

# 國立交通大學

資訊學院 資訊學程

碩士論文

以快閃記憶卡增加硬碟使用效率方案

Performance Enhancement of Hard Drives

Using Flash-Memory Cards

研究生：張少農

指導教授：張立平 教授

中華民國九十七年 12 月

以快閃記憶卡增加硬碟使用效率方案

Performance Enhancement of Hard Drives  
Using Flash-Memory Cards


研究生：張少農

Student : Shao-Nung Chang

指導教授：張立平教授

Advisor : Dr. Li-Ping 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  
December 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年 12 月

# 以快閃記憶卡增加硬碟使用效率方案

研究生：張少農

指導教授：張立平教授

國立交通大學

資訊學院

資訊學程碩士班

## 摘 要

近年來隨著科技技術的日新月異，筆記型電腦的效能及電池續航力也隨著提升。以 CPU 為例，1995 至今，CPU 時脈速度已從 100MHz 大幅增加至現在的 3800MHz，整整快了 40 倍。而其他零組件如快取記憶體、顯示晶片，匯流排頻寬…等等，也都有驚人的發展。但唯獨硬碟的發展非常緩慢。無論是在耗電量或者是使用效率方面，從 1995 年代到現在，只成長 2 至 3 倍。

近幾年來，Flash 記憶體技術的成熟，在效率及耗電量，都較硬碟來得更出色，而以往最令人詬病的價格，也隨技術的成熟大幅下降。另外在使用壽命上重複讀取次數也有所提升，已經可以到達 100 萬次以上的重覆讀寫。這些瓶頸被克服之後，Flash 的使用已經成為儲存裝置的明日之星。

因此本篇論文的目的是希望提供一個方法，利用 Flash 先天上的優勢，與硬碟互相結合。使得硬碟在耗電量或使用效率上能獲得大幅的改善。

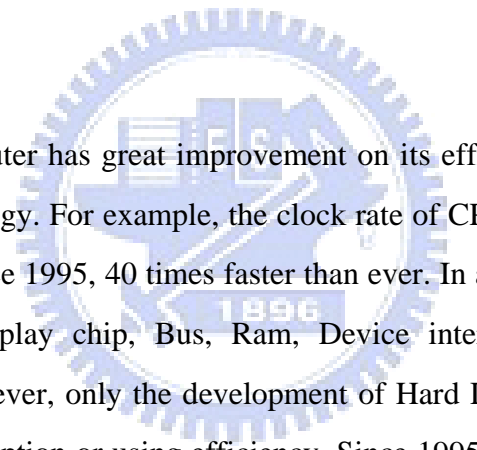
# Performance Enhancement of Hard Drives Using Flash-Memory Cards

Student : Shao-Nung Chang

Advisors : Dr. Li-Ping Chang

Degree Program of Computer Science  
National Chiao Tung University

## ABSTRACT



Recently, notebook computer has great improvement on its efficiency and battery life along with the ever-changing technology. For example, the clock rate of CPU has dramatically increased from 100MHz to 3800MHz since 1995, 40 times faster than ever. In addition, other components of computer, such as Cache, display chip, Bus, Ram, Device interface, and so on also have astonishing improvement. However, only the development of Hard Disk (HD) is behind schedule no matter for the power consumption or using efficiency. Since 1995, it has only developed 2 to 3 times better than before.

During previous years, the technology readiness level of Flash Memory has become better than HD in terms of efficiency and power consumption. Even its controversial prices have substantially dropped with following the increasing level of technology readiness. Moreover, the frequencies of repeated read & write have also improved for its service life which has already reached the mark of over one million frequencies of repeated read & write. Thus, the use of Flash has become an emerging star of storage device after these bottlenecks have been resolved.

Therefore, the target of this thesis tends to offer a solution which is able to use the natural advantage of Flash and to mutually combine it with HD in order to make great improvement in HD's power consumption and using efficiency.

## 誌 謝

在交大求學的這段期間很感謝各位師長的諄諄教誨，和各位同學的相互切磋研究，讓我可以順利的完成我的學業及相關論文的研究。首先要感謝張立平教授擔任我的指導教授，對於我的論文研究和專業知識都不辭辛勞的給予許多的指導和協助，尤其我在資料分析及論文的研究方法上並不是很成熟，在論文研究的學習過程中，經過教授的指導成長很多。不但得到了許多專業的知識，並且學習到了研究分析的方法。再來是感謝和我一起分組作報告的同學們，雖然大家都是以在職生的身份同時忙於工作和課業，但也因為同學都是來自不同的領域，在學識的增長方面相當有幫助，並且在分組的過程中，學到如何利用個人不同的技能來完成一個專案，使每個人都能發揮所長。所以在我們的相互激勵和配合之下，我們的課業方便都能夠順利完成。另外也要感謝我公司的同事，黃冠誠，周峰義，在我就學期間，在事業上幫我負擔一些工作，讓我能有更多時間在我的課業上。

最後感謝交大能提供這樣的環境和機會，讓我在出社會之後仍有進修的機會，不至於停滯不前，只知道目前工作相關的事情。希望今天在交大所得到的，有一天能回饋到交大。

# Contents

摘	要 .....	iii
ABSTRACT	iv	
誌	謝 .....	v
Contents	vi	
第 1 章	簡 介 .....	1
第 2 章	動 機 .....	4
2.1	Performance model(Flash memory card vs HD) .....	4
2.2	相關工作 .....	6
2.3	問題與挑戰 .....	7
第 3 章	演算法 .....	9
3.1	演算法概觀 .....	9
3.2	Sampling 演算法 .....	12
3.2.1	存取行為之分析(feature extraction) .....	12
3.2.2	取樣頻率調整(sampling-frequency adjustment) .....	14
3.3	壽命控制演算法(Lifetime control Algorithm) .....	16
3.4	震盪避免演算法演算法(Threshing avoidance algorithm) .....	18
3.5	系統控制流程 .....	20
3.6	演算法實作議題 .....	28
第 4 章	實驗結果 .....	30
4.1	實驗環境 .....	30
4.2	硬碟與快閃記憶體效能模型 .....	32
4.3	效能指標 .....	35
4.4	讀取效能 .....	36
4.5	能源消耗 .....	38
4.6	快閃記憶體 endurance 的影響 .....	40
第 5 章	結 論 .....	42
附錄 1	硬碟概觀 .....	46
I.	硬碟簡介 .....	46
II.	硬碟構造 .....	46
III.	硬碟組織 .....	47
IV.	硬碟傳輸介面 .....	48
附錄 2	快閃記憶體概觀 .....	49
I.	快閃記憶體簡介 .....	49
II.	快閃記憶體分類 .....	49
III.	快閃記憶體原理 .....	50
IV.	快閃記憶體技術瓶頸 .....	52

## FIGURE CONTENTS

圖表 1	隨機讀取比較圖(HARD DISK VS FLASH MEMORY).....	4
圖表 2	PERFORMANCE IMPROVEMENT OF HARD DISK WITH FLASH MEMORY (HIT RATE 100%).....	7
圖表 3	SYSTEM STRUCTURE OF OUR THESIS.....	11
圖表 4	CYLINDERS ACCESS STATUS.....	13
圖表 5	SYSTEM ARCHITECTURE.....	21
圖表 6	MAIN ROUTINE PROCEDURE.....	22
圖表 7	RANKING PROCEDURE.....	23
圖表 8	SWAP PROCEDURE.....	24
圖表 9	RE-SAMPLING PERIOD ADJUSTMENT PROCEDURE(1).....	26
圖表 10	RE-SAMPLING PERIOD ADJUSTMENT PROCEDURE (2).....	26
圖表 11	LIFETIME CONTROL PROCEDURE.....	27
圖表 12	CYLINDER COUNT ARRAY.....	28
圖表 13	READING REQUEST COUNT ARRAY OF FLASH MEMORY.....	29
圖表 14	RESPOND TIME OF SEQ/RANDOM WRITE HD.....	33
圖表 15	RESPOND TIME OF SEQ/RANDOM READ HD.....	33
圖表 16	RESPOND TIME OF SEQ WRITE HD/FLASH.....	33
圖表 17	RESPOND TIME OF SEQ READ HD/FLASH.....	33
圖表 18	RESPOND TIME OF SEQ/RANDOM READ FLASH.....	33
圖表 19	RESPOND TIME OF SEQ/RANDOM WRITE FLASH.....	33
圖表 20	RESPOND TIME OF RANDOM READ HD/FLASH.....	34
圖表 21	RESPOND TIME OF RANDOM WRITE Hd/FLASH.....	34
圖表 22	RESPOND TIME OF RANDOM WRITE Hd /SEQ WRITE FLASH.....	34
圖表 23	PERFORMANCE OF OUR ALGORITHM(FLASH SIZE : 512 MB).....	37
圖表 24	PERFORMANCE OF OUR ALGORITHM(FLASH SIZE : 256 MB).....	37
圖表 25	PERFORMANCE OF OUR ALGORITHM(FLASH SIZE : 1024 MB).....	37
圖表 26	PERFORMANCE OF OUR ALGORITHM(FLASH SIZE : 128 MB).....	37
圖表 27	POWER CONSUMPTION OF OUR ALGORITHM(4).....	39
圖表 28	POWER CONSUMPTION OF OUR ALGORITHM(1).....	39
圖表 29	POWER CONSUMPTION OF OUR ALGORITHM(2).....	39
圖表 30	POWER CONSUMPTION OF OUR ALGORITHM(3).....	39
圖表 31	POWER SAVING RATIO.....	39
圖表 32	INFLUENCE OF FLASH ENDURANCE(4).....	41
圖表 33	INFLUENCE OF FLASH ENDURANCE(3).....	41
圖表 34	INFLUENCE OF FLASH.....	41
圖表 35	INFLUENCE OF FLASH ENDURANCE(2).....	41
圖表 36	FLASH MEMORY CELL 構造圖.....	50
圖表 37	FLASH MEMORY 寫入示意圖.....	51

圖表 38	FLASH MEMORY ERASE 示意圖.....	51
圖表 39	FLASH MEMORY 讀取示意圖.....	52





# 第1章 簡介

電腦在現代人生活中佔了很重要的角色，有很多工作幾乎沒有電腦，便無法運作，而電腦系統中除了 CPU 以外最重要的就是硬碟機。硬碟的功用是用來儲存大量資料。一旦硬碟損壞，電腦就無法使用任何資料，所以硬碟可以說是最重要的元件之一。但是硬碟在物理技術上，已經到了一定的極致水準，鮮少再有革命性的突破，唯一稱的上意義性突破的當屬「垂直記錄技術」。垂直記錄技術的出現，使硬碟在磁碟密度的表現上再次獲得大幅推升的空間。但是跟硬碟效能相關的物理技術，舉凡磁頭感應技術、讀寫臂換軌技術、主軸轉速等等技術，再進步的可能性卻很小。影響硬碟性能因素非常多，其中又以平均搜尋時間(seek time)，以及主軸轉速影響最大。平均搜尋時間指的是硬碟臂從一個磁軌移動到目的磁軌所花的時間(關於硬碟介紹請參閱附錄 I)，搜尋時間長短取決於硬碟臂的移動速度。另一項影響硬碟效率的主要因素是硬碟馬達轉速，硬碟馬達轉速會決定硬碟 spin up/down 的時間以及從碟片存取資料的時間。目前無論是硬碟臂的移動速度或者是馬達轉速方面都已經到了極致，難有突破性的發展。因此除非更改硬碟的架構，否則只有在硬碟存取方式做改進。

快閃記憶體(NOR型或NAND型)是1984年於東芝公司於半導體學術會議上首次發表的(更多關於快閃記憶體資料請參閱附錄II)。相較於傳統硬碟，快閃記憶卡具有體積小、重量輕、可靠性及耐用性好，抗衝擊、抗震動能力強，消耗功率低等優點，最重要的是在某些狀態下對於資料的存取，快閃記憶卡勝過傳統硬碟許多。在快閃記憶卡的架構設計上，並不需要加入馬達。而馬達這個硬碟原件雖然在傳統硬碟設計上扮演重大角色，但有其致命的弱點。第一是抗震性差，其次是消耗功率大，而快閃記憶體因為不需要馬達，所以比起傳統硬碟，快閃記憶體較能抗衝擊、抗震動、可靠程度也更高。另外在消耗功率方面，由於沒有馬達，因此快閃記憶體在消耗功率方面的表現較傳統硬碟來得亮麗許多。另外在存取資料速率方面，快閃記憶體在隨機讀取資料時，勝過傳統硬碟許多。但快閃記憶體並非沒有缺陷，它最令人詬病的就是讀寫次數的限制以及儲存資料儲存年限，因此快閃記憶體使用年限問題在系統使用上會是個極需克服的重點。

從兩種儲存記憶體的優缺點看起來，傳統硬碟有快閃記憶體不可取代的要素，但快閃記憶體又有硬碟所比不上的性能以及優點，所以結合兩種儲存裝置看起來是可行的做法。如果我們能在傳統硬碟較弱的部分以快閃記憶體的優點來取代，比如以快閃記憶體較快的隨機讀取效率來取代硬碟較差的隨機讀取效率，又或者以及存取資料時，花費耗電量較少的快閃記憶體來取代存取資料需花費較多耗電的傳統硬碟，那麼我們是不是可以發展出新的資料存取模式，而這種資料存取模式同時具有快閃記憶體以及硬碟的優點，有較佳的存取資料速度、較少的耗電量，較佳的穩定性…等等。同時又避免兩者的缺點，比如穩定性差、讀寫次數限制以及資料保存期限短、耗電量大…等等。因此本篇論文將朝結合兩種記憶體的方向來思考，構思一種新的儲存系統架構。由於快閃記憶體價格遠比硬碟來得貴，因此為了較佳的價格容量比，我們思考中的系統將包含一個較小的快閃記憶卡，以及一個容量較大的傳統硬碟。然後以一個驅動程式來控制這兩種硬體。如此，既不用更改現有硬體架構，又可以達到增進效率以及漸少硬碟耗電量的效果。

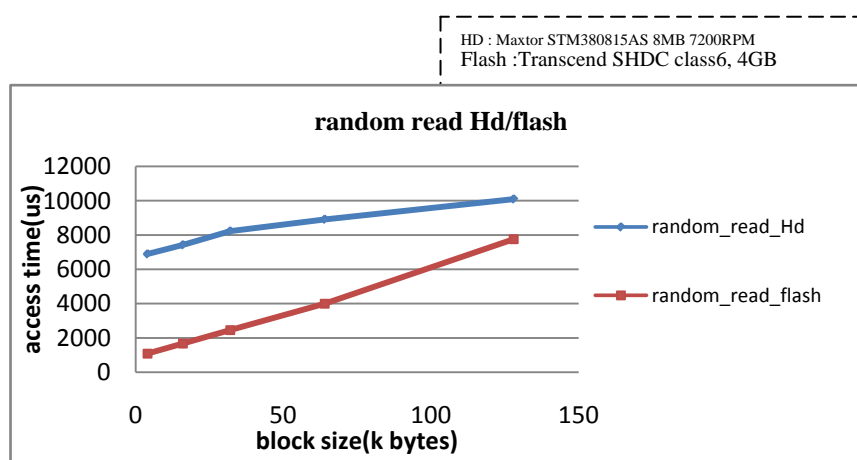
一旦要結合兩種記憶體勢必會遇到一些困難，首先我們面臨的困難是，在我們的整個系統架構中，必須決定何時要使用快閃記憶體？何時要使用傳統硬碟？因為要使系統有較好的效率以及較少的耗電量，我們必須讓我們系統使用快閃記憶的頻率較高。所以必須讓系統未來要讀取的資料盡量在事先就將資料由硬碟搬入快閃記憶體中，但是要知道系統未來將讀取哪些資料卻是非常困難的一件事，這是我們首先必需解決的問題。另外我們該在何時將這些資料由硬碟搬入快閃記憶體中，才能獲得最大的改善效果，又是另一個需要思考的問題。另一方面，快閃記憶體由於先天物理上的因素，存在著資料保存期限，以及讀寫次數受限的問題。以目前作業系統以及軟體的設計來看，會有許多無法避免的隨機讀寫狀況，如果是應用在商業伺服器的話，情況更是明顯，一旦我們使用快閃記憶體來取代硬碟，資料在快閃記憶體與硬碟之間的搬移，無可避免的會使快閃記憶體的讀寫次數增加，減短系統使用壽命。系統在面臨最嚴苛的持續讀寫狀況下，壽命可能不到幾個月便告終結，因此我們演算法必須克服這個問題。

由於經過實驗測試，快閃記憶體的隨機存取效率要勝過傳統硬碟，因此我們系統將會將所有的存取型式分為兩種，隨機存取以及循序存取。當隨機存取時以快閃記憶體作為儲存裝置，當循序存取以傳統硬碟作為儲存裝置。可是如果要做到這種存取方式，我們必須知道哪些資料必須放在快閃記憶體，哪些資料必須放在傳統硬碟，這是一個很大的挑戰，我們將以”以小觀大”的概念來做為我們的解決方法，所謂以小觀大，即是觀察一小段時間的讀取資料，來預測往後一段時間，哪些資料可能被讀取的方式來決定哪些資料該被放進快閃記憶體中，詳細的統計方法以及統計時機將在後面章節做討論。另一個我們系統必須解決的問題是，快閃記憶體有讀寫次數的限制，如果為了讀取效率的關係，增加讀寫次數將會使系統的使用年限縮短，為了使系統的使用年限不會過短，因此我們必須限制快閃記憶體的寫入次數，在後面章節我們也會針對這個問題做討論，嘗試以讀寫次數配額的觀念來解決這個問題。提升效率以及延長使用年限的做法似乎會有所衝突，因此我們必須以演算法來取得最佳平衡點。

本文組織將分文四章節，在第一章我們將會針對本篇論文的概念及組織做些基本介紹。第二章則會分為三個部份，第一部分會介紹本篇論文為什麼會想要這樣做，第二部分會介紹之前的一些其他人的相關做法，第三部分會介紹我們系統這個作將朝遇到哪些問題，一旦解決這些問題將得到甚麼好處。第三章會分四個部分介紹，分別介紹我們演算法的原理、系統架構、演算法詳細作法、我們演算法的優點。第四章則會介紹我們的實驗。我們的實驗將會以模擬的方式來進行，實驗參數會以市面上常見的硬碟以及快閃記憶體來設定。而實驗的讀寫資料方面將會以一個普通 PC 長時間的存取資料作為我們的輸入資料。針對我們演算法在系統效率，耗電以及快閃記憶體的寫入次數來做討論。另外在附錄部分會簡單介紹硬碟的原理構造，以及快閃記憶體的原理構造以及其技術瓶頸。

## 第2章 動 機

### 2.1 Performance model(Flash memory card vs HD)



圖表 1 隨機讀取比較圖(Hard disk vs flash memory)

從傳統硬碟以及快閃記憶體的 performance mode(圖表 1)可以看出硬碟在隨機讀取方面表現得非常差，這是因為硬碟在作隨機讀取時，需要不時的移動硬碟臂的關係，因此硬碟的搜尋時間將會影響整體效率，因此硬碟在隨機讀取的能力非常不好，但是當讀取資料是循序資料時，由於硬碟臂移動的次數以及距離非常小，因此其搜尋時間將會非常小，整體讀取時間也會較快閃記憶體快。而快閃記憶體就沒這方面的困擾，因為快閃記憶體在做隨機讀取時並不需要做機械方面的移動，因此快閃記憶體的搜尋時間非常快，遠勝過硬碟的循序存取時間。所以快閃記憶體讀取隨機資料取速度遠勝於傳統硬碟。當讀取循序資料時，由於硬碟的硬碟臂移動次數及距離小，搜尋時間相對較快，因此在讀取循序資料方面，快閃記憶體要較傳統硬碟來得慢。

我們定義硬碟由靜止啟動到達可以讀取資料的動作稱之為 spin up，硬碟由可以讀取資料到達完全靜止的動作 spin down。當硬碟 spin up 開始到完成，由於硬碟馬達的特性關係，當馬達由靜止到達讀取資料所需的速度時，需要由低轉速慢慢加速到高轉速，因此需要較多的時間來完成加速，同時所耗費的電能也較多。當硬碟 spin down 時，也需要同樣的時間來慢慢減速到完全停止，因此假如系統讀取資料導致硬碟 spin up/down 次數過於頻繁，將會耗費大量的電能以及時間。而快閃記憶體由於沒有馬達的關係，因此有較佳的耗電量、也不需要 spin up/down 的時間。

綜觀上面的因素，我們可以歸納出一些結果。並且可以針對這些結果推論出一些方向來解決硬碟的效率及耗電問題。第一個方向是減少硬碟的搜尋時間(seek time)。但是要如何減少硬碟的搜尋時間呢?由於硬碟的搜尋時間與硬碟臂的來回次數與距離相關，因此要減短搜尋時間，就要減少資料存取時硬碟臂搜尋的次數以及硬碟臂全部的搜尋距離。這樣做的同時，還可以減少耗電量。另一個增加硬碟效率與減少耗電量的方向是減少硬碟 spin up/down 的次數，由於硬碟 spin up/down 所耗費的時間以及電能是最大的，因此假如我們能藉由減少硬碟 spin up/down 的次數，來達到增加硬碟效率與減少耗電量的目的。

要達到減少硬碟 spin up/down 的目的以及減少硬碟平均的搜尋時間有兩種做法。第一種有效方法是將寫入資料的次序重新排序。由於硬碟在存取資料時，硬碟臂將會隨著資料的所在來回移動，因此如果能將讀寫資料的次序重新排列，讓硬碟臂盡量在移動一次完成時，就將讀寫資料處理完畢，如此便能減少硬碟搜尋時間以及 spin up/down 的次數，很不幸的是這種方法只能用在寫入資料時，因為讀此資料有即時性，並不適用。另一種做法則是以較快的記憶體來當 Cache。在硬碟讀寫資料時，如果可以到較快的記憶體作讀寫，等到有需要時再到硬碟做讀寫，便能減少硬碟搜尋時間，同時也能減少硬碟 spin up/down 的次數。

本篇論文的方向將採取以快閃記憶體來當作 Cache 的方式來做為解決硬碟問題的方案。在這幾年來，也有很多相關的研究採取類似的做法，但都是針對硬碟的寫入需求來提出解決方法，但由於對硬碟讀取預測有相當的困難性，因此針對硬碟讀取的解決方案較少人提出，其做法通常是採取將被讀取資料的附近資料也一起搬進快閃記憶體的作法。但這種作法雖然有效，但是效果有限。而本篇論文雖然也採取以快閃記憶體來當作 Cache 的方式來做為解決硬碟問題的方案，但我們將採取預測硬碟的讀取資料的方式來作為硬碟解決方案。這是本篇論文與其他論文最大的不同。

## 2.2 相關工作

關於這如何增加硬碟效率以及減少耗能的作法及研究非常多，大概可以歸類為三種概念。第一種做法，以管理硬碟 Spin-Up, Spin-Down 的觀念來減少耗電，硬碟因為省電的關係，在一段時間沒有對硬碟作存取動作時，在一段時間後，硬碟將會 Spin-Down, 等到需要存取硬碟時，硬碟才會 Spin-Up。而硬碟的 Spin-Up、Spin-Down 是最浪費電能及時間的，一旦頻繁的 Spin-Up、Spin-Down 將耗費大量的時間及電能。” Adaptive Disk Spin-down Policies for Mobile Computer” [7]，這篇論文中提出一個方法將硬碟 Spin-Up, Spin-Down 的時機做一個的管理。

另外一種做法，以重新排列的方式存取順序增加硬碟效率，在 "A buffer Cache management scheme exploiting both temporal and spatial localities" [2] 中提出一種做法。因為存取 Hard Disk 最浪費時間的就是硬碟臂來回的移動，以及磁片的旋轉。因此作者認為假如能夠將隨機 Access request 變成循序的 Access request。就可以達到增加效率，減低耗電。

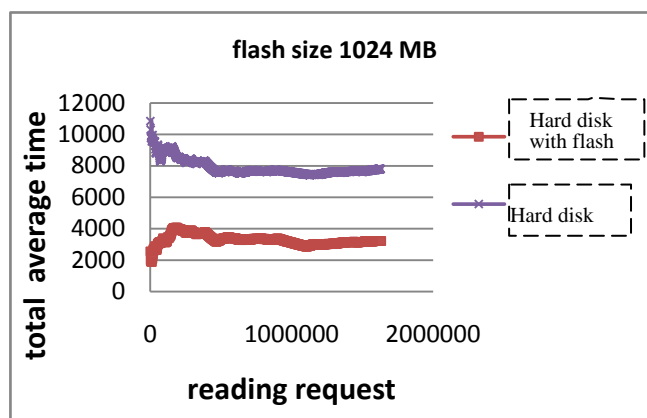
“Reducing Hybrid Disk Write Latency with Flash-Backed I/O Requests” [6]、” Flushing Policies for NVCache Enabled Hard Disks” [9]，雖然做法不盡相同，但是原理是一樣的。

第三種作法，以高效能低電能記憶體來當硬碟快取，減少耗電，增加效能。” SmartSaver: turning flash drive into a disk energy saver for mobile computers" [4]。作者分別對讀寫方面來進行，他將快閃記憶體分成三個部份 1. Cache area: 使用這個區域來管理 Cache 資料，並且發展了一套評估方法來評估哪些 Sectors 該被留在 Cache 中。2. Prefetching area: 當某一個 sector 被搬進 Cache 中時，因為 space locality 的關係，有可能附近的 sector 也有可能馬上被讀到。因此作者提出一個簡單的方式評估該搬多少資料進入這塊 buffer。3. Write back area : 當要寫資料時先將資料寫進這塊區域中，等到硬碟 spin-up 時，再將所有資料一起寫入。最後再依據目前狀態去動態調整這三塊區域的大小。另外在 "Energy-Efficient and Performance-Enhanced Disks Using Flash-Memory Cache" [1], 也提出另一種做法，他使用一個 Flash 管理機制去管理 Flash 的使用。

## 2.3 問題與挑戰

由於我們的系統將採取讀取隨機資料時，以快閃記憶體來取代傳統硬碟的做法，那麼我們將會遇到很多挑戰。首先就是如何將讀取資料在讀取之前先放到快閃記憶體中，其次就是快閃記憶體的讀寫限制問題，另外就是我們演算法所產生的邊際效應。

要達到我們系統所設定以快閃記憶體來取代硬碟的隨機讀取，首先要面臨的挑戰是我們必須預測系統在接下來將會讀取哪些資料。另外由於未來得不可確定性，我們很難去知道未來系統將會讀取何種資料，而將這些資料預先搬進快閃記憶體中。由於我們演算法系統裡快閃記憶體相當類似 PC 系統體的 Cache 所扮演的角色，因此我們可以一般管理 Cache memory 的演算法來做資料的預測嗎？答案是否定的。因為 PC 系統 Cache memory 的關係，同一地區 (sector) 在短時間內在被讀取的機率非常低，因此如果將以一般的 Cache 演算法作為被讀過的資料搬進快閃記憶卡及將資料搬出快閃記憶體的依據，將會使得我們快閃記憶體的命中率非常低，所以我們不能以一般的 Cache 演算法來作資料的 swap in/out。我們必須發展我們自己的預測演算法來解決這個問題。圖表 2 是以假設命中率 100% 的狀態下，我們以快閃記憶體來取代傳統硬碟隨機讀取的方式 (紅線)，與完全以傳統硬碟來做讀取資料 (藍線)，所做的比較，我們可以發現，假如我們資料 pre-fetch 演算法命中率不錯的話我們將在硬碟效能上，獲得不小的提昇。



圖表 2 performance improvement of Hard disk with flash memory (hit rate 100%)

目前主流快閃記憶體技術目前仍為 NAND 與 NOR 技術，兩者都各有其適用對象與特性，但是兩者都有其物理特性的限制，NAND 最高只有 10 萬次(近年來已有企業開發出可以讀寫 100 萬次的快閃記憶體，但是價格非常貴)左右的讀寫壽命，NOR 甚至更低。在我們系統中，由於須將資料預先搬進快閃記憶體中，因此假如我們做資料 swap in/out 的次數過多的話，也代表我們必須重複寫入快閃記憶體的次數增加，而一旦我們的寫入次數在短時間內快速增加到超過快閃記憶體的重複寫入次數的話，我們系統的使用年限將會受到限制，因此我們必須對我們系統寫入快閃記憶體的次數做出限制，以面避免使用年限的過短，但是一旦我們限制 swap in/out 次數的話，又會面臨整個演算法對於系統效率提升的程度不佳。因此我們的系統須有一個機制來處理這個問題，既讓我們的系統的使用年限可以令人接受，同時整體系統的效能也可以獲得不錯的提升。

當我們設計演算法解決上面這兩個問題時，同時也可能會產生一些邊際效應。例如 swap in/out 衍生的 threshing 問題(我們定義:threshing 現象是當我們演算法判定需將某些資料搬進快閃記憶體時，但是這些資料對於往後的讀取命中率並無幫助)，我們可能因為這些 threshing 現象發生頻率過高，反而使得我整體系統效能犯而降低。因此我們必須有些演算法機制來避免這個問題。如果我們能成功避免這個問題的話，我們將可以減少系統 swap in/out 的次數。這樣做可以有兩個好處，首先，減少不必要的 swap in/out 的次數，可以減少我們 swap in/out 所需付出的話費，使整體系統效能有所提升。另外減少 swap in/out 還可以減少快閃記憶體的寫入次數，進而延長快閃記憶體的使用年限，使得我們系統可以使用的更長久。



## 第3章 演算法

經過前面幾章的敘述，讀者應該對這篇論文想要解決的問題，以及這些問題發生的原因，以及之前其他學者相關的作法，有相當的了解。接下來我們將在這章介紹我們的系統架構，相關演算法以及原理。在我們的演算法中將針對硬碟讀取的預測所將遇到的問題一一提出解決方法。因此在本章節將針對以下幾個方向做出討論。演算法概觀、資料 Sampling、快閃記憶體 endurance 限制、Threshing、演算法實作。

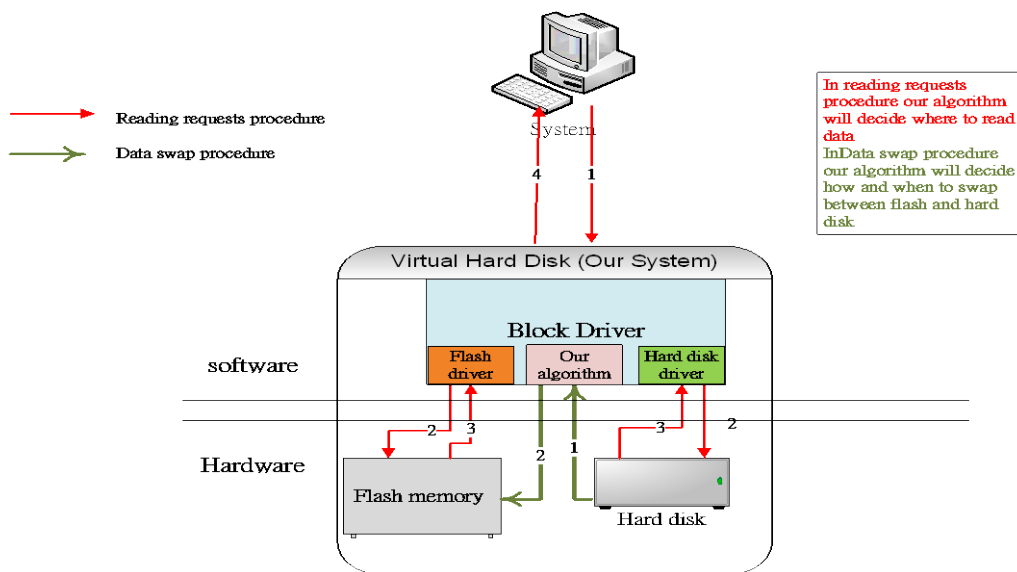
### 3.1 演算法概觀

我們的演算法系統會是一個虛擬硬碟，當系統讀寫硬碟時，我們系統會攔截系統 I/O 資料，由我們的系統(block driver)取代原本的硬碟驅動程式。我們的系統架構如圖表 3 所示，硬體部分包含一個快閃記憶體，以及一個硬碟。軟體部分(稱為 Block Driver)包含 3 個部份，分別為一個控制快閃記憶體以及硬碟的 Driver，以及我們的演算法本身。

我們的硬體包含兩個部份，一個傳統硬碟以及一個快閃記憶體。傳統硬碟在我們系統中扮演的角色為儲存裝置，儲存系統裡所有的讀寫資料，當系統在快閃記憶體中找不到所需的資料時，才會到硬碟中讀取資料，如果在快閃記憶體中可以讀取到所需資料時，硬碟會一直維持在 spin down 狀態，硬碟維持啟動的時間長短將會跟我的演算法命中率相關。而快閃記憶體在我們的系統中扮演的角色較類似 PC 系統裡的快取記憶體。在快閃記憶體中，將會存放我們演算法所預測的資料，也就是我們預測硬碟中將來會被系統讀取的資料，根據我們演算法會在適當時機點開始對未來系統的讀取資料做出預測，然後會將這些演算法所預測的資料搬進快閃記憶體中，當系統在未來讀取資料時，將會先到快閃記憶體中讀取，當系統所需的資料不在快閃記憶體中時，才會到傳統硬碟中讀取資料。

我們演算法軟體的部份是一個 Block driver，用來取代系統原本的硬碟驅動程式，攔截系統對虛擬硬碟的 I/O 動作，分為 3 個部份，其中包含兩個需由系統提供的部份，分別為硬碟的驅動程式，以及快閃記憶卡的驅動程式。硬碟的驅動程式用來控制我們系統中的硬碟資料讀寫；快閃記憶卡驅動程式則用來控制快閃記憶卡的資料讀寫。第三部份則是我們演算法部份。我們的演算法在 Block driver 中有三種作用，分別是仲裁、統計系統讀取行為、預測系統未來讀取資料的角色，當系統 I/O 進來時，我們演算法會判斷要到快閃記憶體或是硬碟中讀取資料；另一個功能是針對系統讀取行為資料的統計，以做為預測將來資料的依據；另外演算法會根據統計資料，預測將來系統將會讀取哪些資料。我們演算法會根據快閃記憶體的 endurance 限制以及系統效率，決定何時開始針對讀取資料做出預測，並將演算法預測出的資料搬進快閃記憶體中。

因此我們的演算法系統會分為兩個部份。第一部份處理正常的讀取動作（圖表 3 紅色部份），我們演算法會將系統 I/O 過濾，一般的 I/O 讀寫將會被直接到硬碟當中讀寫資料。當判斷系統讀取資料為隨機讀取時，我們的演算將會啟動機制，先判斷系統所讀取的資料是否在快閃記憶體中，若資料已在快閃記憶體中，則到快閃記憶體中讀取資料；若資料不在快閃記憶體中，則到硬碟中讀取資料，完成系統讀寫資料的程序，同時將資料被讀取的頻率記錄下來，做為系統預測的憑據。演算法第二部份（圖表 3 綠色部份），當系統須做資料預測時機來到，系統會針對目前資料被讀取頻率的統計資料，預測出未來將被讀取的 Cylinder 是那些，並根據演算法評估這些 Cylinders 是否值得被搬進快閃記憶體中，若 Cylinders 值得被搬進快閃記憶體中，我們演算法才將這些 Cylinders 搬進快閃記憶體中。在這同時若本次預測時機是在硬碟 Hot period 開始時，我們演算法還必須依據系統讀取效能以及快閃記憶體 endurance 限制，設定下次系統須在次做預測的時機，在系統讀取效能以及快閃記憶體 endurance 限制之間取得最佳的平衡點。



圖表 3 System structure of our thesis



## 3.2 Sampling 演算法

根據前面章節所述，我們演算法要以快閃記憶卡來增進硬碟的效能，我們必須要將硬碟將來要讀取的資料(硬碟讀取頻率較高的資料)，預先搬進快閃記憶體中。因此我們將會面臨兩個問題，首先我們必須預先知道那些資料是電腦系統將來會讀取的，其次我們演算法必須決定何時才是將這些資料搬進快閃記憶體的最好時機，因此我們設計取樣演算法(Sampling algorithm)來解決這兩個問題。在下面將分為兩個部份，存取行為分析(feature extraction)及取樣頻率調整(sampling-frequency adjustment)討論。

### 3.2.1 存取行為之分析(feature extraction)

要決定哪些資料該預先被搬進快閃記憶體中是非常困難的，因為沒有人能保證電腦系統在未來將會讀取哪些資料，即使是觀察過去的讀取資料來預測未來讀取資料，因為電腦系統中含有快取記憶體，所以效果也不好；另外是當我們預測到未來電腦系統可能會讀哪些資料時，我們該怎麼評估這些資料時否值得被搬進快閃記憶體呢？，這些都是我們演算法必須先克服這個困難。

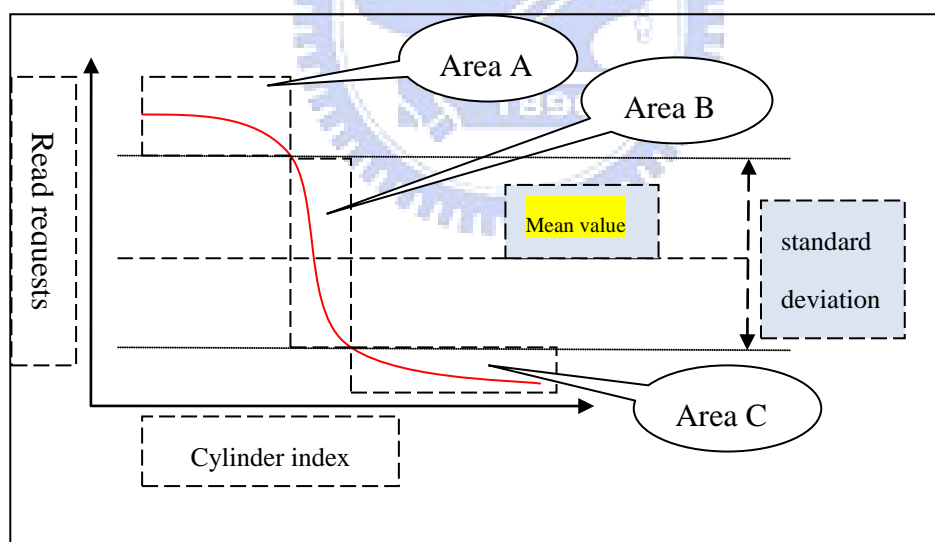
該搬哪些資料進快閃記憶體中呢？我們的想法是，我們以Cylinder為基本單位<sup>1</sup>觀察硬碟的讀取行為(圖表 4)，我們可以發現在同一個Hot period當中，常被讀取的Cylinder群以及不常被讀取的Cylinder群會有一定程度的落差。我們將所有Cylinder分為三類，第一類是常被讀取的Cylinder群(Area A)，第二類是過渡Cylinder群(Area B)第三類是不常被讀取的Cylinder群(Area C)；我們的做法是只將Area A部分的資料搬進快閃記憶體中，Area B由於佔的數量並不太大，而Area C的部分因為被讀取的次數也較少，我們也不做任何處理。要如何決定哪些Cylinder屬於Area A？在我們的演算法是以所有的Cylinders被讀取次數的標準差做為基準，因為標準差可以代表相當程度的資料分散程度。

---

<sup>1</sup>由於系統快取記憶體的影響，同一個 sector 短時間內被讀取的機會並不高，所以我們以 Cylinder

我們的具體作法是，在每一次 Hot period 一開始時每隔一段時間去(間隔時間如何決定，將在後面章節討論)搬一次資料，在重新取樣時，先統計同一個 Hot period 開始至目前為止每個 Cylinder 被讀取的次數，然後根據被讀取的次數來求出標準差，然後將被讀取的次數超過標準差的 Cylinders，作為我們預定搬進快閃記憶體中的 Cylinders，這些預定搬移的 Cylinders 必須還要考慮 Threshing 的影響，才能決定我們真正要搬移的 Cylinders。

理論上，在同一個 Hot period 之間，電腦系統通常做相類似的事情，因此其對 Cylinders 的讀取行為會相類似，因此我們理論上可以以同一個 Hot period 內前面的讀取行為來代表整個 hot period 的讀取行為。依據我們的取樣演算法，我們可以相當程度的預測到一個 hot period 內的讀取行為，從而達到改進硬碟效能的目的，但是這樣作的同時，將會產生副作用 (Threshing)，我們將在 3.4 節討論這個議題。



圖表 4 Cylinders access status

### 3.2.2 取樣頻率調整(sampling-frequency adjustment)

作Cylinder搬移的另一個問題是，何時才是將我們預測的資料搬進快閃記憶體的最佳時機？由於在Hot period中每個cylinder被讀取的密集程度並不一樣，因此有可能在不同的時間點會有不同Cylinders migration<sup>2</sup>現象產生，因此我們必需每隔一段時間就去重新統計一次目前狀況，將合適的Cylinders搬進快閃記憶卡中。如果太早將資料搬進有可能會影響我們預測資料的命中率，如果太晚將資料搬進快閃記憶體，又會變的不符效益。因此如何決定時機也是我們取樣演算法的另一項挑戰。

要決定取樣週期並不容易，有兩點因素會影響我們決定取樣週期。第一個決定的因素是Cylinders搬移的次數。當取樣週期越長，Cylinders搬移的次數也越少；取樣週期越短，Cylinders搬移的次數也越多。而Cylinders搬移的次數越多也代表著效率越差，反之Cylinders swap次數越少代表著效率越好。第二個因素則是讀取平均反應時間。當取樣週期越短，偵測到Cylinders migration的機會越多，因此預測Cylinders越準，因此讀取命中率也越高，效率也越好。反之效率也越差。因此由上述兩點可知取樣週期的長短分別會影響平均反應時間以及cylinder搬移次數，當取樣週期越短，會增加Cylinder搬移次數但會增加命中率，取樣週期越長會減少Cylinder搬移次數但會減少命中率。如何去取得平衡是我們演算法的重點。電腦系統每次讀取的全部平均反應時間，是Cylinders搬移時間加上讀取平均反應時間。我們的想法是依據讀取的全部反應時間動態調整取樣週期。

我們的作法是當我們要決定第N個Hot period時我們先觀察第N-1、N-2個Hot period，例如當我們在第N-2個Hot period時將取樣週期縮短，發現第N-1個Hot period讀取的全部反應時間變少，我們會將第N個Hot period取樣週期繼續縮短，反之如果發現第N-1個Hot period讀取的全部反應時間變長了，我們會將第N個Hot period的取樣週期增加，以此類推。

---

<sup>2</sup> Cylinders migration:由於每個Cylinder被讀取的時間點不一樣，因此在同一個hot period中不同的時間點去看Cylinders被讀取的排名可能會有所變動，我們稱這種變動現象為Cylinders migration

另外由於快閃記憶體的 Endurance 問題，我們的須改成依據 Cylinder 的搬移次數來動態調整取樣週期，調整方法如同上面所述。

理論上，依照我們的演算法，我們可以依據最佳效率來動態決定 Cylinder 搬移到快閃記憶體的最佳時機。因為我們演算法可以在平均反應時間以及 cylinder 搬移次數當中取得平衡點。但是我們的演算法會產生一個副作用，因為依照最佳效率來動態決定取樣週期，有可能會為了效率造成 Cylinder 搬移次數過多，導致快閃記憶體讀寫次數過多，而使的系統使用期限縮短。如何解這個問題我們將在 3.4 節作討論



### 3.3 壽命控制演算法(Lifetime control Algorithm)

由於快閃記憶體先天上架構上的限制，在寫入快閃記憶體上會有次數限制，因此假設我們 Sampling 演算法寫入快閃記憶體太多次將會限制系統的使用壽命。所以我們的演算法必須有種機制(throttling algorithm)避免上述問題。要從根本徹底解決快閃記憶體的寫入次數限制是不可能的(原因請參閱附錄二)。所以我們演算法將在讀寫次數上做限制。

在我們取樣演算法中為了取的效能的改進，需要對快閃記憶體作寫入動作，但寫入過多會造成系統使用期限縮短，因此我們必須限制我們的 Cylinder 搬移次數。我們的想法是要在快閃記憶體的最大讀寫次數限制下取得最佳的讀取效能，所以使用 quota 的觀念設計我們的壽命控制演算法。我們的想法是，我們必須判斷目前的可寫入額度，舉例來說：我們假設我們系統硬碟使用年限為 n 年，快閃記憶體可被讀寫次數為 K 次，快閃記憶體容量為 C 個 Cylinders，因此  $AccessTimes_{One\ Day} = K / (365 * n)$ ，假設系統已經被使用 d 天總共搬移次數為  $K_d$  次(單位為 cylinder/每次)，理論上到第 d 天可以搬移次數  $Quota_d = AccessTimes_{One\ Day} * d(\text{天}) * C(\text{cylinder 個數})$ ，所以理論上我們第 d 天 可以使用的配額  $quota = Quota_d - K_d$ ，將每天 Quota 以時間為比例來算出目前可用 quota。譬如當我這個 hot period access 時間為第 d 天，時間為 t 小時。我就可以算出目前可用的 swap 次數

$$quota_{now} = \frac{quota}{24} * t。$$

我們的做法是，在我們每一次重新取樣時，必須先依照上面理論，求出當下可以搬移的次數額度，當求出理論上的搬移次數額度時( $quota_{now}$ )，假設系統目前已經搬移的次數為  $K_{now}$ ，將  $K_{now}$  與可搬移次數  $quota_{now}$  做比較，若  $K_{now}$  小於  $quota_{now}$ ，因為還沒達到可使用的搬移次數額度。因此演算法以效率來動態改變 resample period 以求達到最佳效率。若  $K_{now}$  大於  $quota_{now}$ ，因為已經超過搬移次數額度，演算法以搬移次數來動態改變取樣週期。



理論上來說，當取樣週期越長，Cylinder 搬移次數會越少，反之當取樣週期越短，Cylinder 搬移次數會越多，因此我們可以有效控制 Cylinder 搬移次數，避免超過快閃記憶體使用年限。而且依照我們演算法的理論，我們可以調整取樣週期，達到在快閃記憶體的讀寫限制下的最佳效率。但是我們的壽命控制演算法的執行次數將會對我們效率造成影響。我們以目前的快閃記憶體製造技術來看，通常可以允許重複寫入快閃記憶體約 100000 次，甚至更新的技術可以達到 1000000 次之多。以我們的模擬實驗 (4.6 節) 來看，要達到 10 年的使用年限且快閃記憶體的讀寫限制 (約 100000 次) 是非常不容易的。因此我們的演算法可以忽略快閃記憶體 Endurance 對我們效率上的影響。



### 3.4 震盪避免演算法演算法(Threshing avoidance algorithm)

經過上面的演算法，理論上可以有效增加效率，但過程中將會產生一些不必要的搬移動作。原因是在同一個 hot period 時間內所有的 cylinders 被讀取的頻率可能差不多，在此時做 swap 的動作部沒有多大意義。在下面兩種情形可能會發生，第一種情形，當求出的標準差過小，其物理意義代表 cylinders 被讀取頻率差不多。第二種情形，當從硬碟搬移到快閃記憶卡的 cylinders 被讀取次數與要被從快閃記憶體被移除的 cylinders 相差不多時。此時將 Cylinders 資料做交換，對效率的影響並不大，但是反而會多浪費時間在處理資料交換，反而減低效率。

我們的想法是，我們需要一個 filter 值來做為我們判斷到底該不該搬資料的基準。但是這個值到底該如何決定呢?在這裡我們以每次 Swap 的時間相當於被命中幾次作為我們的 filter 值；也就是當兩個 Cylinder 的次數相差超過這個值，我們才將這兩個 Cylinder 作交換。推論公式如下：

已知參數：

$T_{read\_hd}$ : time of read hard disk / once

$T_{read\_flash}$ : time of read flash memory card / once

$T_{swap}$ : time of swap / once

N: number of read times:

推論公式：

$$\text{Total read time (hd, flash)} < \text{Total read time(hd)} \dots\dots\dots(1)$$

$$\rightarrow T_{read\_hd} \times 1 + T_{swap} + T_{read\_flash} \times (N - 1) < T_{read\_hd} \times N \dots\dots(2)$$

$$\rightarrow T_{swap} + T_{read\_flash} \times (N - 1) < T_{read\_hd} \times (N - 1)$$

$$\rightarrow T_{swap} < T_{read\_hd} \times (N - 1) - T_{read\_flash} \times (N - 1)$$

$$\rightarrow T_{swap} < (T_{read\_hd} - T_{read\_flash}) \times (N - 1)$$

$$\rightarrow T_{swap} \div (T_{read\_hd} - T_{read\_flash}) < (N - 1)$$

$$\rightarrow N > (T_{swap} \div (T_{read\_hd} - T_{read\_flash})) + 1 \dots\dots\dots(3)$$

公式(1)表示某個 Cylinder 被讀取的 N 次，有快閃記憶體的讀取時間必

須小於只純粹讀取硬碟的時間。公式(2)代表有快閃記憶體讀取  $N$  次的時間為讀取一次硬碟的時間加上將 Cylinder 搬進快閃記憶體的時間再加上讀取  $N-1$  次快閃記憶體的時間。公式(3)經過化簡過後，代表理論  $N$  必須大於  $(T_{\text{swap}} \div (T_{\text{read}_{\text{hd}}} - T_{\text{read}_{\text{flash}}})) + 1$  次，作 swap 才有效用，因此我們以這個值作為 filter。

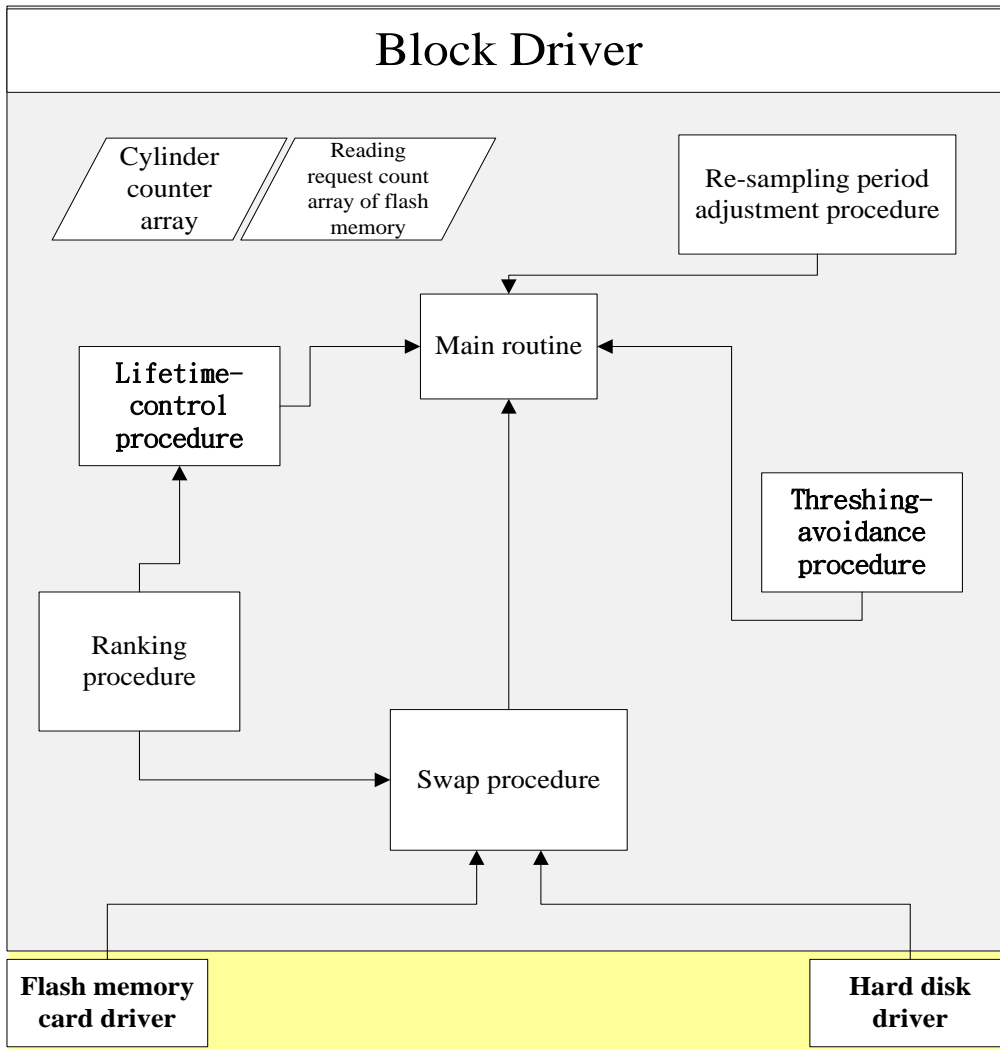
我們的具體做法是每一次 hot period 開始時，我們會求出 filter 值。而參數的如下  $T_{\text{read}_{\text{hd}}}$ 、 $T_{\text{read}_{\text{flash}}}$  可由快閃記憶體的規格估計得出，或由演算法執行過程中統計得到； $T_{\text{swap}}$  由演算法執行過程中統計取得。當系統在同一個 hot period 內重新取樣時，在系統決定將 Cylinder 搬進快閃記憶體時，若 Cylinder 被讀取次數小於 filter，或搬進記憶體之 Cylinder 與搬出記憶體之 Cylinder 被讀取次數相差小於 filter，則放棄將 Cylinder 搬進快閃記憶體中。

以我們的理論來看，應該可以減低演算法過程中產生的不必要搬移次數，但是能無法完全避免，這是因為目前被讀取頻率較高的 Cylinders，只能說往後被讀取的機率很高，但不能保證一定會被讀取。而我們震盪避免演算法可以減低大部份的不必要的搬移次數。

### 3.5 系統控制流程

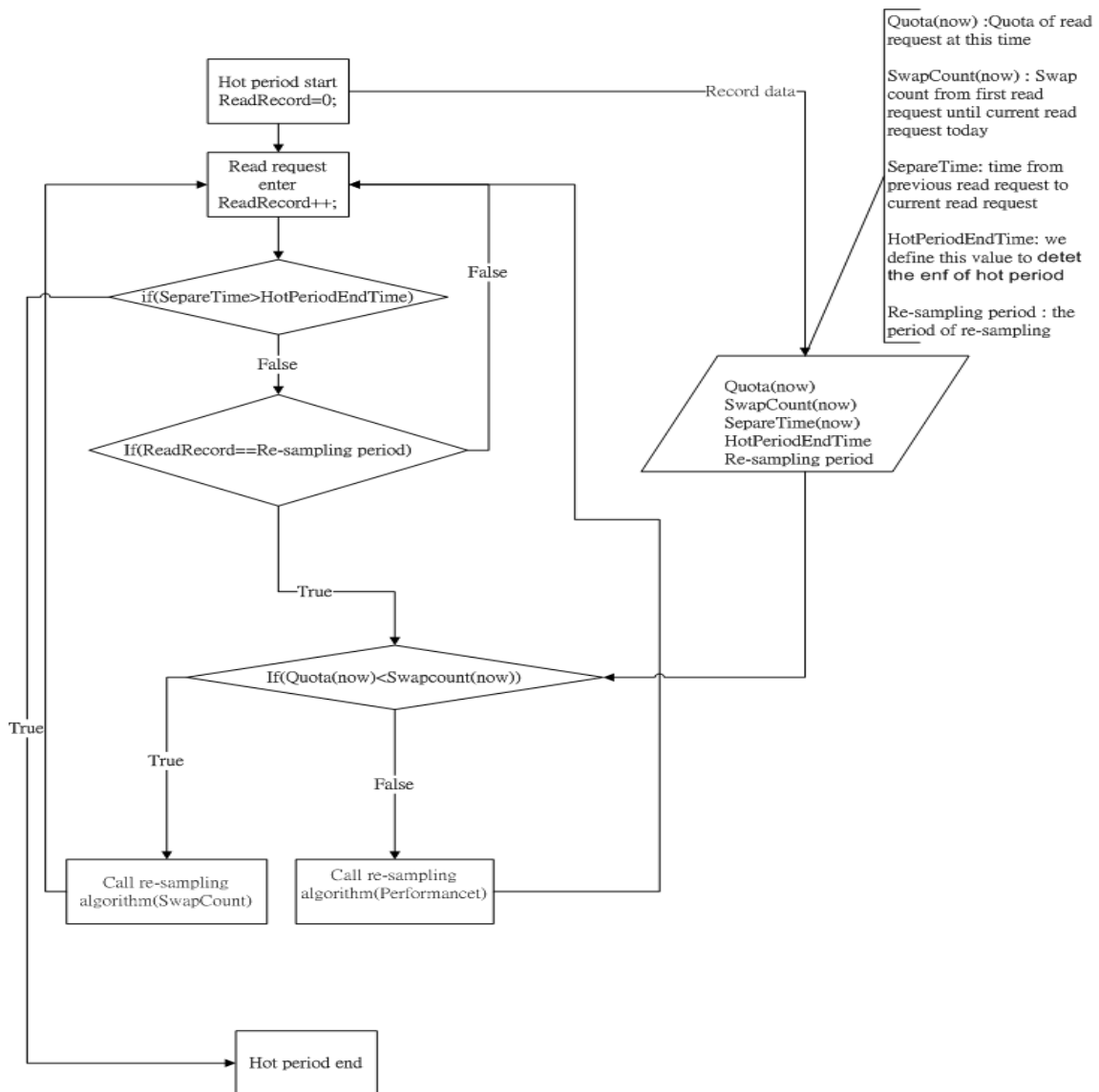
由於我們的演算法並不複雜，因此在程式實作方面其實也很簡單。並不需要複雜的運算、也不需要複雜的流程。我們演算法的實作可以分為幾個部分，分別為 Ranking procedure、Swap procedure、Threshing-avoidance procedure、Re-sampling period adjustment procedure、Lifetime-control procedure、main routine，以及兩個簡單的資料結構。在實作上，我們的系統並不複雜，整個系統只需要兩個簡單的資料結構以及六個簡單的子程式，另外需要外部提供兩個 driver，讓我們演算法可以控制系統的硬碟以及快閃記憶卡。圖表 5 為整個系統的架構圖，箭頭所指方向為 Procedure 會被哪些 Procedure 所使用，如 Lifetime-control procedure 會被 Main routine procedure 所呼叫。

整個系統會分作三個時機分別有不同的流程。首先是一般的讀取資料時，一開始時會進入 main routine 中，main routine 會決定要到硬碟或快閃記憶體中取的資料，並且呼叫 Ranking procedure 記錄所有 Cylinder 被讀取的次數。第二個時機點是當系統判斷 hot period 開始時，main routine 會呼叫 Lifetime-control procedure 判斷本次 hot period 應該是用 Cylinder 搬移次數或是整體效率來調整取樣週期。當決定調整方式後呼叫 Re-sampling period adjustment procedure 來調整我們的取樣週期。調整取樣週期後，呼叫 Threshing-avoidance procedure 來取得 filter 值。第三個時機點是重新取樣，當系統判斷重新取樣時機時，main routine 會呼叫 Swap procedure 並依據 Hot period 一開始求得的 filter 來搬移預測資料到快閃記憶體中，在 Swap procedure 會實作取樣演算法以及依據 filter 來決定要搬哪些資料到快閃記憶體。



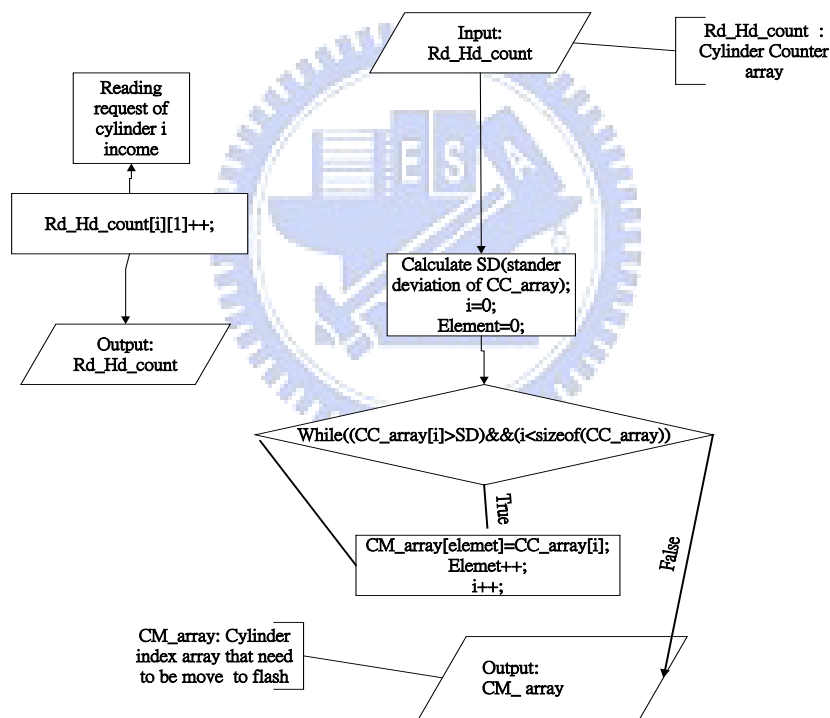
圖表 5 System architecture

Main routine 是我們演算法的主要流程。整個功能分為三部份。當一般讀取時會判斷從快閃記憶體或是硬碟中拿資料。當 Main routine 判斷 Hot period 開始時，Main routine procedure 會根據 Lifetime-control procedure 來決定要以 swap 次數或者以整體效率為考量來啟動 Re-sampling period adjustment procedure。詳細流程如圖表 6 所示。若判斷為重新取樣時機時則呼叫 Swap procedure。



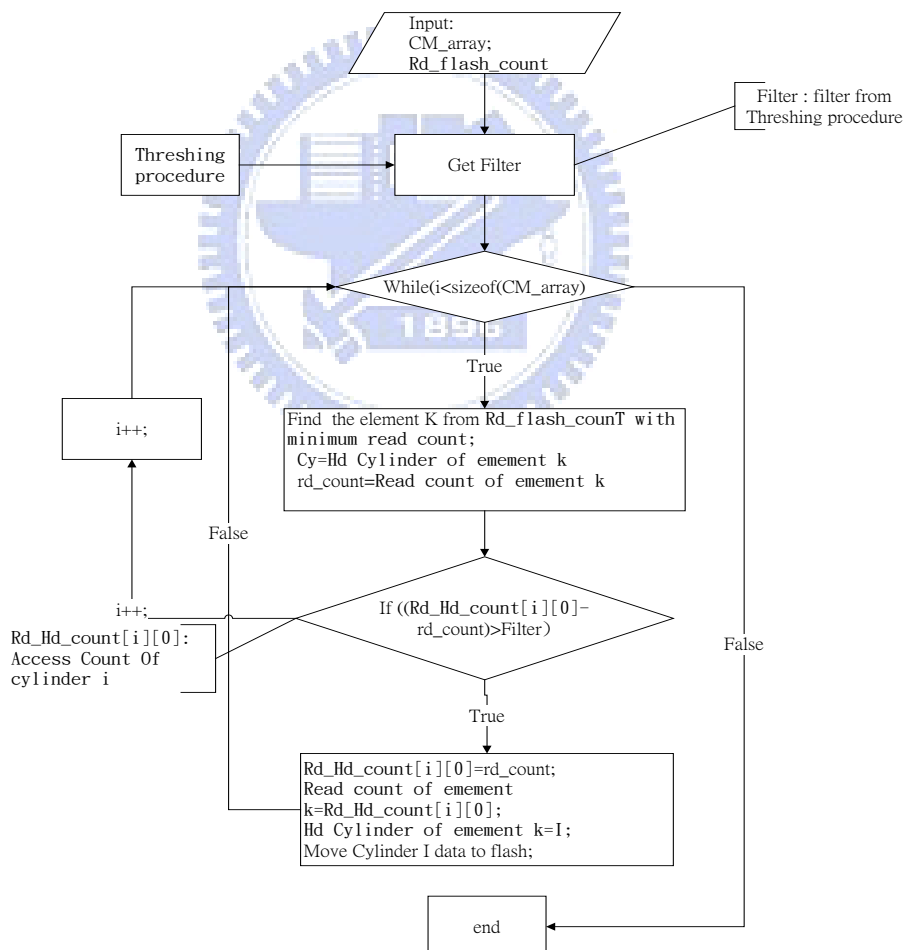
圖表 6 Main routine procedure

在 Ranking procedure 需實做兩個部份，第一部份，當平常 reading request 發生時，我們採用 Cylinder 的被讀取次數作為我們的依據值。當每 Cylinder  $i$  被讀取時，若 Cylinder  $i$  在硬碟中，將  $Rd\_Hd\_count[i]$  的 Access Count+1，若 Cylinder  $i$  在快閃記憶體中，則將  $Rd\_flash\_count$  中 Hd Cylinder= $i$  的 Read Count+1。第二部份，在每次 resample 時我們必須統計目前每個 Cylinder 被 Access 的次數，以求出 Cylinders 標準差。在求出標準差後，我們須判斷標準差是否大於 filter 值(由 filter 演算法求出)。若大於 filter 時，找出所有 Cylinders 被讀取次數大於標準差的 Cylinders，並記錄在  $CM\_Array$  中，演算法流程如圖表 7 所示。



圖表 7 ranking procedure

Swap procedure 的功能是決定要搬哪些資料到外閃記憶體中，這個 procedure 將在系統 resample 時被 main routine 呼叫，它的 Input 是 ranking algorithm 所產生的 CM\_Array 以及 Rd\_flash\_count。這個 procedure 主要是處理硬碟與快閃記憶體的 Swap。Procedure 會由 Threshing avoidance 演算法取得 filter 值，並根據這個值判斷是否要將 CM\_Array 中的 Cylinders 搬進快閃記憶體中。當要將 Cylinder i 搬到快閃記憶體時會將，將 Rd\_Hd\_count[i] 的 Access Count 值寫到 Rd\_flash\_count 中 Hd Cylinder=i 相對應的 Read count 中，當系統將 Cylinder i 從快閃記憶體移除時，將 Rd\_flash\_count 中 Hd Cylinder=i 的 Read Count 值寫到 Rd\_Hd\_count[i] 相對應的 Access count 中。流程圖如圖表 8。



圖表 8 Swap procedure

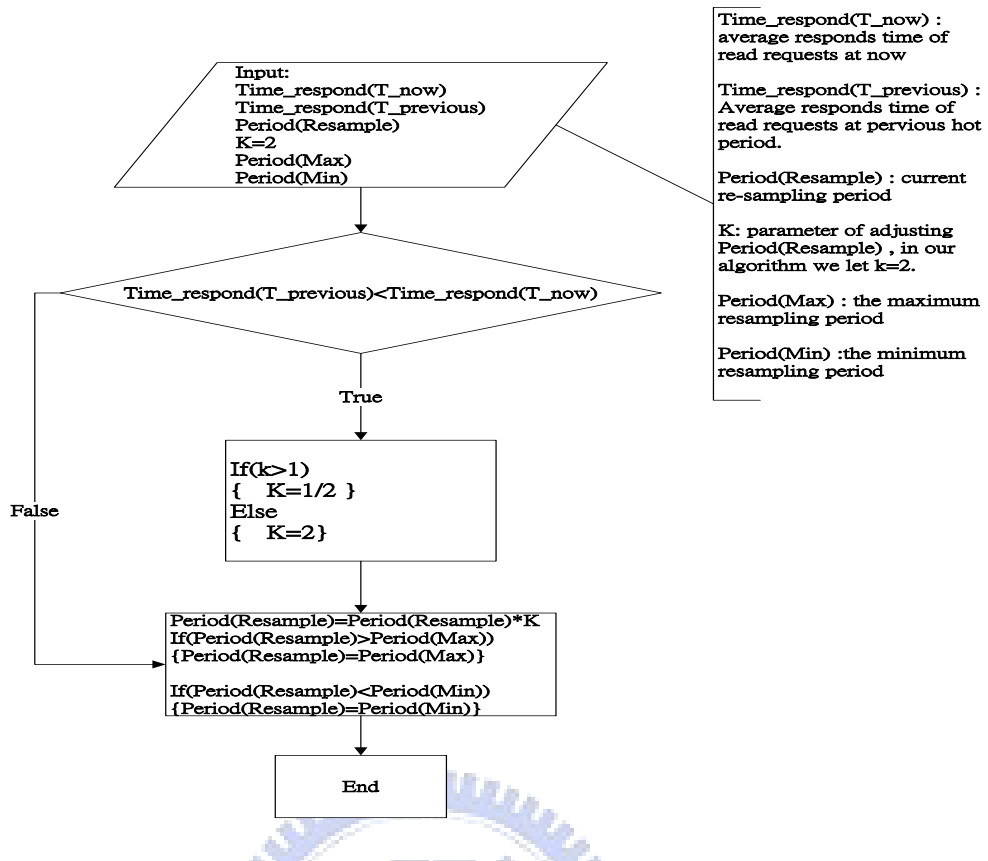


我們演算法的第三部份是 Threshing avoidance procedure，這個 procedure 主要的作用在求出 filter 值，在 hot period 一開始時被 main routine 所呼叫，傳入參數為搬入一個 Cylinder 到快閃記憶體的時間，平均讀取硬碟的時間，平均讀取快閃記憶體的時間。目前這些參數我們參考硬碟以及快閃記憶體的規格來求出理論值。

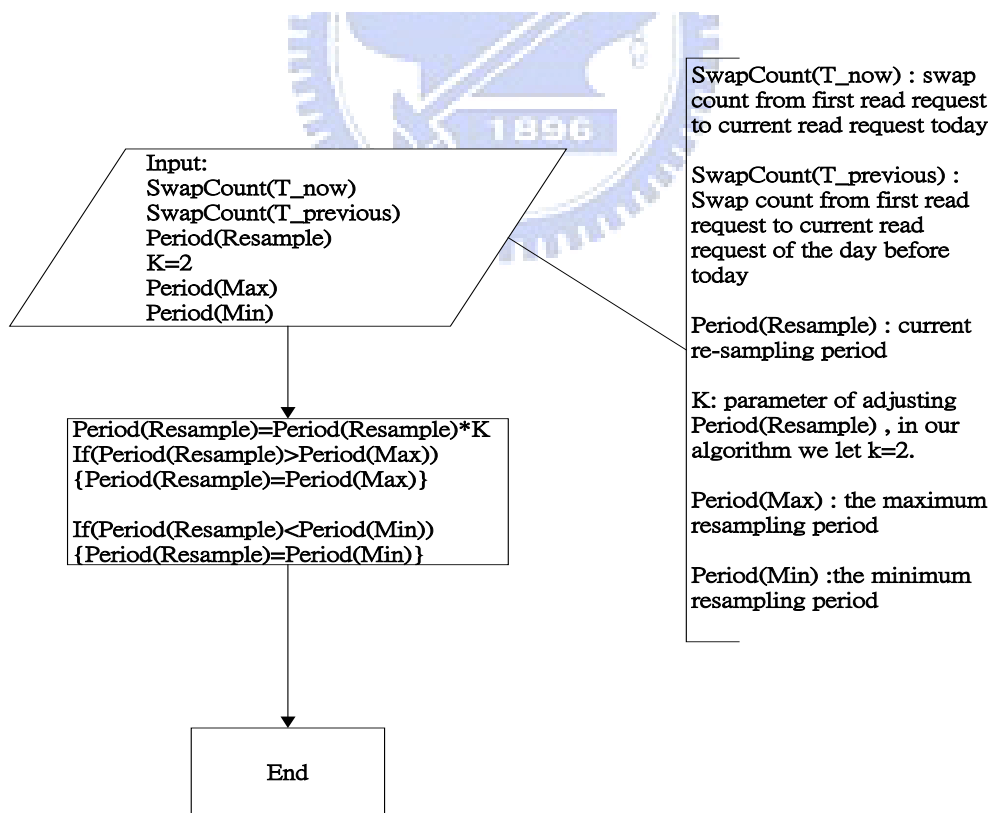
我們演算法的第四部份為 Re-sampling period adjustment procedure。這個 procedure 用來決定下次 hot period 的 resample period 長短。在 hot period 一開始時被 main routine 所呼叫。有兩個部份需要實作，分別是根據整體效率(圖表 9) 或根據 swap 次數(圖表 10)來動態調整取樣週期長短。

當我們以整體效率(圖表9)來動態調整resample period長短時，main routine 會傳入參數為前兩次的讀取平均反應時間(Time\_respond(T\_now)，Time\_respond( T\_previous ))、目前取樣週期 (Period(resample))、最大的取樣週期(Period(Max))、最小取樣週期(Period(Min))、調整值(K:程式中設為2，會將取樣週期以2倍或1/2倍的方式調整)。程式會依據演算法來求得最佳效率的取樣週期。

當我們以整體效率(圖表 10)來動態調整 resample period 長短時，main routine 會傳入參數為前兩次的 Cylinder 搬移次數(SwapCount(T\_now)，(SwapCount ( T\_previous ))、目前取樣週期 (Period(resample))、最大的取樣週期(Period(Max))、最小取樣週期(Period(Min))、調整值(K:程式中設為 2，會將取樣週期以 2 倍或 1/2 倍的方式調整)。程式會依據演算法來求得最符合快閃記憶體寫入限制的取樣週期。

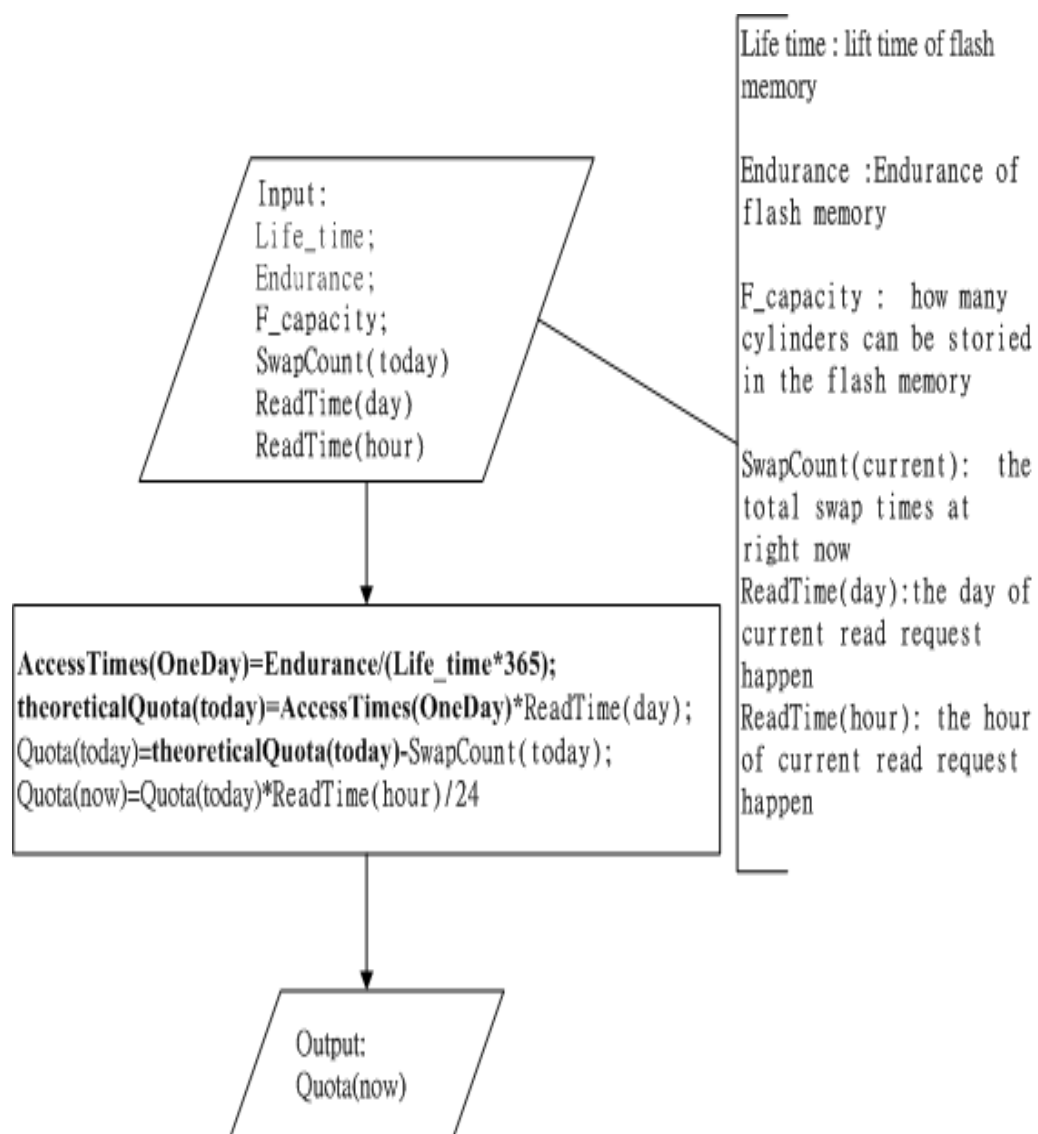


圖表 9 Re-sampling period adjustment procedure(1)



圖表 10 Re-sampling period adjustment procedure (2)


我們演算法的第五部份是 Lifetime-control procedure(throttling algorithm)。這個 procedure 會實作 Lifetime-control algorithm。Lifetime-control procedure 中是由 main routine 呼叫，main routine 會將系統參數系統使用年限(Life\_time)、快閃記憶體寫入次數(Endurance)、快閃記憶體可容納的 Cylinder 數(F\_capacity)、今天開始到目前已經寫入的次數，本次 Hot period 開始的時間(ReadTime(today)，ReadTime(Hour))，然後重新計算。流程如圖表 11



圖表 11 Lifetime control procedure

### 3.6 演算法實作議題


依照我們演算法，我們將會需要兩個資料結構。分別用來記錄讀取過程中的每個 Cylinder 被讀取頻率。第一種資料結構是 Reading request count array(以 Rd\_Hd\_count 表示)，它在我們演算法中的目的是用來記錄 Cylinder 在每次 hot period 的被讀取次數，如圖表 12。這個資料結構將包含兩個參數，第一個參數是 Cylinder index: 表示 Cylinder 在硬碟中的索引。第二個參數是 Access Count, 被用來記錄目前硬碟中每個 Cylinder 被讀取的次數。當系統在每次讀取 Cylinder i 時，我們演算法會將 Rd\_Hd\_count[i] 的 Access Count 值加 1。當每次 hot period 結束時，演算法會將 Rd\_Hd\_count 中 Access Count 值重設為 0。當系統將 Cylinder i 從硬碟 swap 到快閃記憶體時，會將 Rd\_Hd\_count[i] 的 Access Count 值寫到 Rd\_flash\_count 中 Hd Cylinder=i 相對應的 Read count 中，當系統將 Cylinder i 從快閃記憶體 swap 到硬碟時，會將 Rd\_flash\_count 中 Hd Cylinder=i 的 Read Count 值寫到 Rd\_Hd\_count[i] 相對應的 Access count 中。



Cylinder index	Access count
Cylinder0	125
Cylinder1	64
Cylinder2	0
Cylinder3	0
Cylinder4	0
Cylinder5	32
Cylinder6	11
Cylinder7	146
Cylinder8	55
Cylinder9	66
Cylinder10	00
Cylinder11	23
Cylinder12	11
.....	.....

圖表 12 Cylinder count array

我們演算法第二種資料結構使用於記錄快閃記憶體中 Cylinders 在 hot period 被讀取的頻率(以 Rd\_flash\_count 表示)。如圖表 13。這個資料結構將包含三個參數，第一個參數是 flash cylinder index，用於表示在快閃記憶體中的 Cylinder 索引。第二個參數是 Hd Cylinder，被用來記錄儲存於快閃記憶體中的 Cylinder 是屬於硬碟中的哪個 Cylinder。第三個參數表示每個儲存於快閃記憶體中 Cylinder 被讀取次數。當系統在每次讀取快閃記憶體中 Cylinder i 時，我們演算法會將 Rd\_flash\_count 中 Hd Cylinder=i 的 read Count 值加 1。當每次 hot period 結束時，演算法會將 Rd\_flash\_count 中 read Count 值重設為 0。當系統將 Cylinder i 從硬碟 swap 到快閃記憶體時，會將 Rd\_Hd\_count[i] 的 Access Count 值寫到 Rd\_flash\_count 中 Hd Cylinder=i 相對應的 Read count 中，當系統將 Cylinder i 從快閃記憶體 swap 到硬碟時，會將 Rd\_flash\_count 中 Hd Cylinder=i 的 Read Count 值寫到 Rd\_Hd\_count[i] 相對應的 Access count 中。



Flash Cylinder	Hd Cylinder	Read count
Cylinder 0	Cylinder 409	10
Cylinder 1	Cylinder 508	8
Cylinder 2	Cylinder 1048	2
Cylinder 3	Cylinder 1096	0
Cylinder 4	Cylinder 4	1
Cylinder 5	Cylinder 1	94
Cylinder 6	Cylinder 814	16
Cylinder 7	Cylinder 812	32
Cylinder 8	Cylinder 706	15
Cylinder 9	Cylinder 408	1
.....	.....	.....

圖表 13 Reading request count array of flash memory

## 第4章 實驗結果

根據我們演算法，在理論上對效率以及耗電量方面，應該會有不錯的效果。但不知在實際上的效果怎麼樣？在這章我們將用實驗來佐證我們的演算法。我們將以軟體模擬的方式，來測量實際的數據，並根據這些數據結果來分析我們的演算法的效果以及造成的影響。

### 4.1 實驗環境

由於我們將以軟體模擬的方式來驗證我們的演算法。所以我們將在這節介紹我們的所有實驗環境。首先，在硬體環境部份，我們使用的 CPU 為 intel 的 T7500，具有 2.2GHZ 的 CPU 頻率，這顆 CPU 還具有 4MB 的 L2 快取。在 RAM 的部分則是使用 4GB 的 DRAM；另外模擬程式我們則使用 Microsoft Visual Studio 2008 來發展我們的模擬程式；另外作業系統我們選擇微軟的 Vista 作為我們的模擬環境；另外我們將會記錄一顆 20GB 的硬碟長時間的讀寫動作(約 26 天)來做為我們的模擬程式的讀寫資料。

另外在我們硬碟模擬的部份，我們以一般市面上常見的 3.5 吋硬碟規格作為我們的模擬硬碟規格。在硬碟轉速方面我們選擇一般常見的規格 7200 轉；在硬碟 Header 及 Cylinder 部份，由於一般市面上硬碟 head 為 16 到 255，cylinder 為 1024 到 16384 由於資料分析的方便，我們將硬碟參數定義 header 為 255，Cylinder 為 2609；硬碟資料傳輸方面，我們選擇 SATA2 的理論傳輸速度 300MB/sec；硬碟 Seek time 的部分參考 reference[12] 論文裡所建立的硬碟 seek time model<sup>3</sup>；由於我們的測試資料為一個 20GB 硬碟的長期讀取記錄(約 26 天)，因此我們的硬碟容量也設定為 20GB。

在快閃記憶體模擬的部份，我們選用 Transcend 快閃記憶卡，記憶晶片規格是 SHDC class6，容量為 4GB。以程式來測試記憶卡的平均讀寫數據。我們測試程式使用的 API 為 CreateFile(……)、WriteFile(……)、ReadFile(……)來讀寫資料。選用這些 API 主要是因為讀寫資料可以不經

---

<sup>3</sup>關於 seek time 的模擬方法我們將以 reference[12] 中建立的硬碟 Model 來求得。

$d < 616, 3.45 + 0.59\sqrt{d}$

$d \geq 616, 10.8 + 0.012d$

block-device buffer Cache，所以可以避免其他因素影響讀寫時間，另外讀寫動作的前後分別以 intel CPU 的 RDTSC 指令來讀取 CPU Time Stamp，所以可以精確的測量出讀寫動作所花的 CPU 指令週期，再根據 CPU 執行頻率換算出精確的讀寫時間。求得數據如下，隨機讀取資料時間為 30us、循序讀取資料時間為 29us、隨機寫入資料時間為 699us、循序讀取資料時間為 43us。我們以這些數據做為我們程式模擬時，快閃記憶體的讀寫數據。



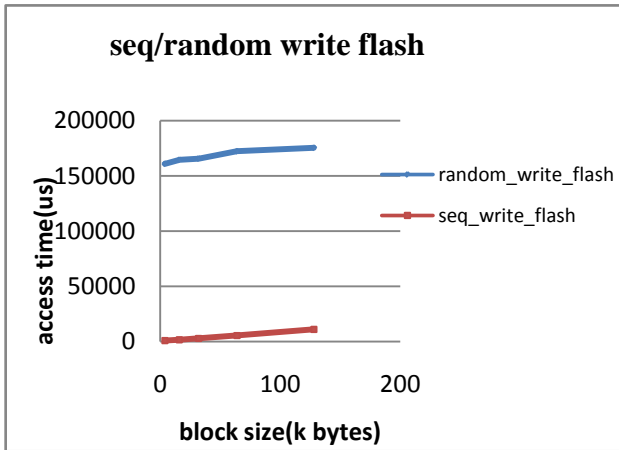
## 4.2 硬碟與快閃記憶體效能模型

在這節我們將測試市面上常見的硬碟與快閃記憶卡，建立其效能模型，並且分析硬碟與快閃記憶體的效能模型，從中得出我們論文的方向與演算法基礎。我們選擇的硬碟型號為 Maxtor STM380815AS、內建 8MB 的 Buffer、硬碟轉速為 7200RPM、傳輸規格為 SATA、平均 seek time 為 9ms。快閃記憶體方面我們選擇 Transcend 快閃記憶卡，記憶晶片規格是 SHDC class6，容量為 4GB。以程式將測試資料分別對硬碟以及快閃記憶卡作讀寫動作，並測量其平均時間。讀寫的方法分為循序寫入資料、循序讀取資料、隨機寫入資料以及隨機讀取資料。寫入的資料大小分別為 4k、16k、32k、64k、128k。

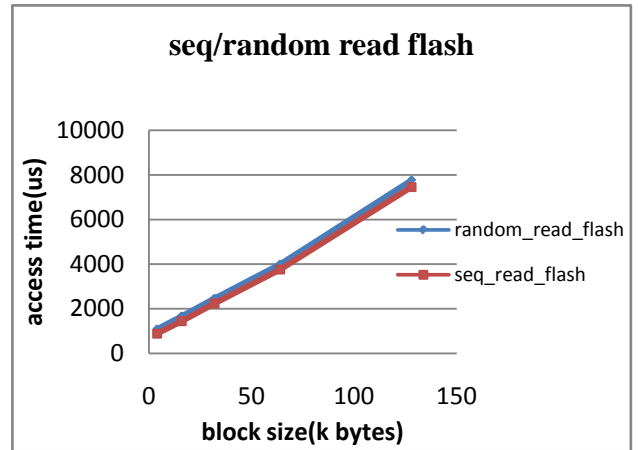
經過測試後我們得出結果如圖表 14、圖表 15、圖表 16、圖表 17、圖表 18、圖表 19、圖表 20、圖表 21、圖表 22。從這些數據可以得到一些結論。第一個結論，我們可以所有圖表看出無論是硬碟或記憶體，無論讀取資料或寫入資料，以循序的方式都會比隨機的方式來得快，但是由圖表 18 我們可以知道快閃記憶體的隨機讀取資料跟循序讀取資料是一樣快的。第二個結論，由圖表 19、圖表 21 可以發現隨機寫入快閃記憶體是最花時間的，第三點，由圖表 17 我們可以看出循序讀取資料時，從硬碟讀取資料要比從快閃記憶體讀取快，而且相差的程度非常明顯，但是若是在隨機讀取狀態下可以明顯的看出，快閃記憶體比硬碟快非常多(圖表 20)。第四點，在寫入小量的資料下循序寫入快閃記憶體比隨機寫入硬碟要快很多(圖表 22)。

從第三點的結論，我們可以發現隨機讀取快閃記憶體資料比隨機讀取硬碟資料快，而在循序讀取下結果卻相反。因為單以讀取速率來講硬碟要比快閃記憶體快，因此在循序讀取資料時硬碟要較快閃記憶體快。但是在隨機讀取的狀態下，硬碟的讀取速度需額外考慮一項因素，硬碟的 seek time。由於隨機資料可能散布在硬碟的每個 Cylinder 中，因此隨著讀取資料的不同，硬碟必須來回移動以取得正確的資料，因此 seek time 的平均時間會加長，所以在隨機讀取狀態下，快閃記憶體要較硬碟快得多。由於在隨機讀取狀態下，快閃記憶體較硬碟快非常多，而一般 PC 系統隨機讀取的機會其實非常多，因此假如我們在隨機讀取資料的情況下，能以讀取快閃記憶體的來取代讀取硬碟，那麼我們有機會能改善我們的讀取效能。我們論文的方向將朝這個方向進行。

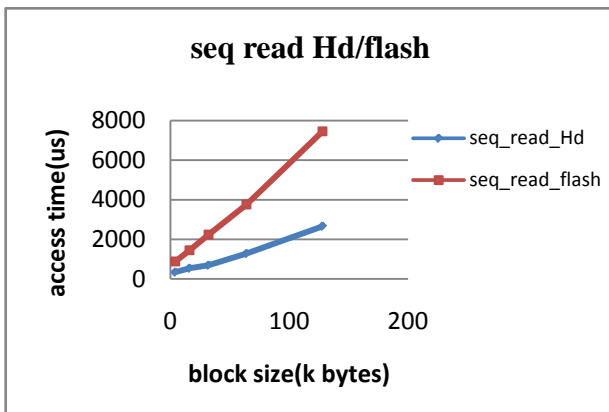




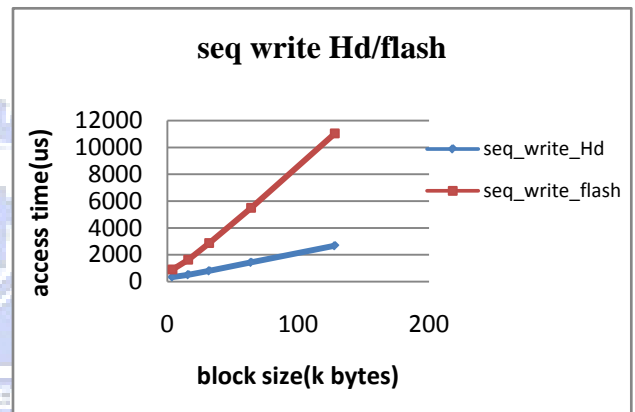
圖表 19 respond time of seq/random write flash



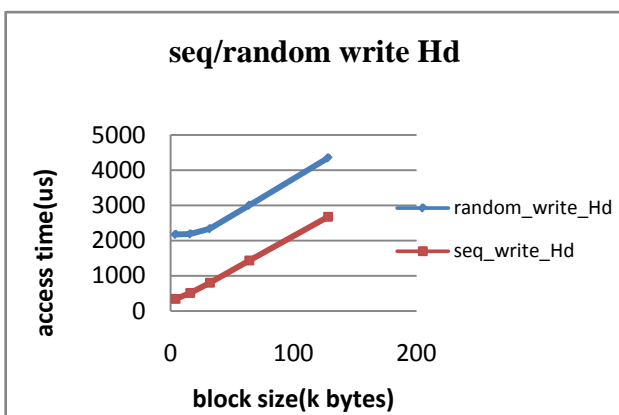
圖表 18 respond time of seq/random read flash



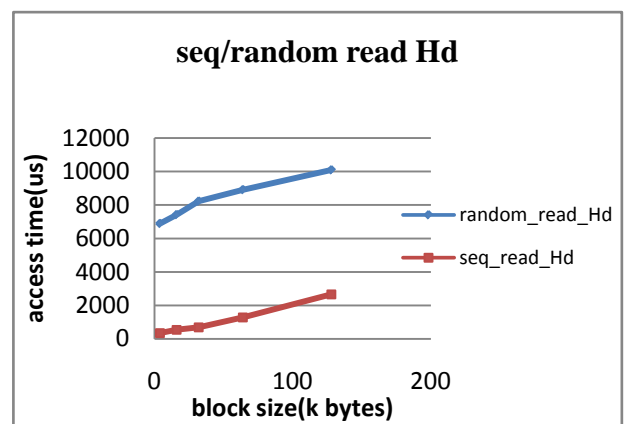
圖表 17 respond time of seq read Hd/flash



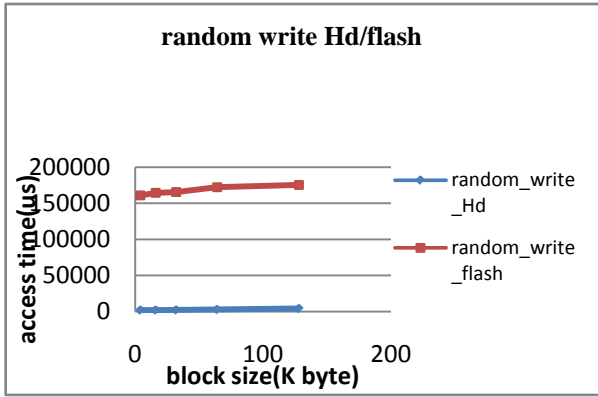
圖表 16 respond time of seq write Hd/flash



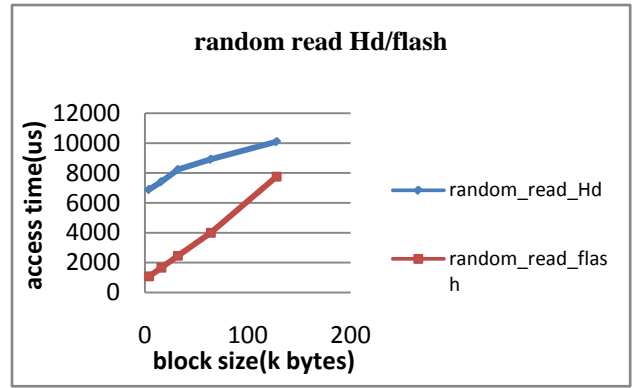
圖表 14 respond time of seq/random write Hd



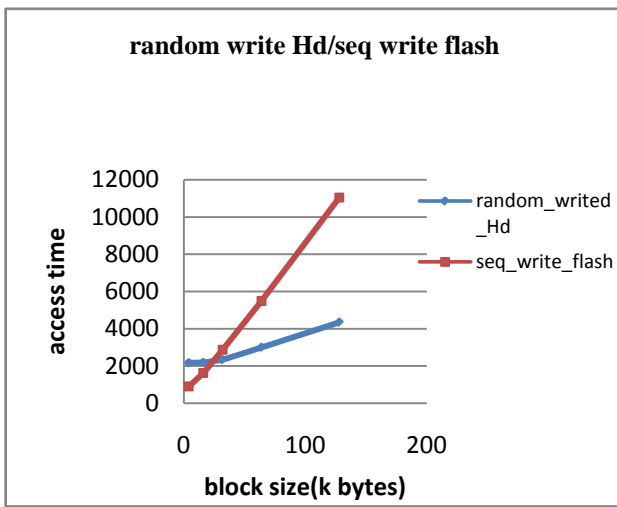
圖表 15 respond time of seq/random read Hd



圖表 21 respond time of random write Hd/flash



圖表 20 respond time of random read Hd/flash



圖表 22 respond time of random write Hd /seq write flash

### 4.3 效能指標

經過理論的推演，在理論上將對系統有所改善，但是應用在實際上到底效果如何？因此我們設計了 3 個實驗，以軟體模擬的方式分別針對幾個因素來觀察，藉以了解我們演算法對系統的效應如何。我們實驗要觀察的因素分別是效能、耗電量、讀取命中率、快閃記憶體 endurance 的影響。

首先是關於演算法對於系統效能的影響。在理論上，我們演算法對於效能有不錯的改善效果。因此我們首先要觀察的是效能的實際影響。我們將以讀取資料的平均反應時間(average responds time<sup>4</sup>)作為我們的評判標準，當讀取資料的平均反應時間越短，代表演算法改進硬碟效能的效果越好，反之若讀取資料的平均反應時間越長，代表演算法改進硬碟效能的效果越不好。另外會以四種不同的演算法作為我們的對照組。我們將在 4.4 節做說明。

第二步我們要觀察的指標是演算法對系統耗電量的影響。理論上由時演算法會減少硬碟的讀取，改成由耗電量較少的快閃記憶體讀取資料，因此會減少整個系統的耗電量。因此我們設計實驗，以觀察讀取動作的累積耗電量來判斷演算法對系統耗電量的影響。如果累積耗電量越多代表效果越不好，若累積耗電量越少則表示我們演算法對減少耗電量的效果越好。另外我們將以傳統硬碟的耗電量來做為我們的對照組。

第三步我們要觀察的指標是快閃記憶體 endurance 對於我們演算法效率以及對使用年限的影響。如之前章節所述快閃記憶體有讀寫次數的限制，而我們為了做讀取預測須對快閃記憶體作讀寫，因此我們將會影響系統的使用年限。因此我們必須觀察快閃記憶體 endurance 限制的影響。我們會針對不同容量的快閃記憶體觀察(128MB、256MB、512MB、1024MB)，觀察快閃記憶體 endurance 的影響。另外我們會觀察快閃記憶體 endurance 次數對系統效能的影響。如何能夠在不影響使用年限又能達到不錯的效率。

---

<sup>4</sup> 我們統計從第一筆讀取資料到現在時間 T 為止的讀取資料平均反應時間來做為在時間 T 的平均反應時間。

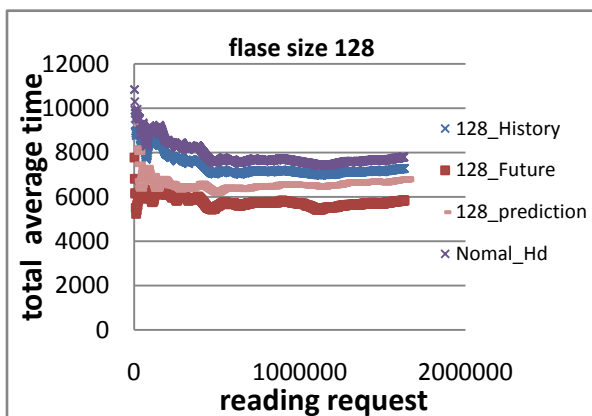
## 4.4 讀取效能

在這個實驗我們要觀察的是演算法對效能改善的效果。我們有三個對照組。分別是 Future 演算法(xxx\_Future)、History(xxx\_History)演算法、一般傳統硬碟(Nomal\_Hd)。Future 演算法，我們以每個 hot period 為單位，預先去統計目前這個 Hot period 被讀取最多的 cylinders 並將這些 Cylinders 在每個 hot period 一開始就搬進快閃記憶體。History 演算法則是以每個 hot period 為單位，統計前一個 Hot period 被讀取最多的 cylinders，將這些 Cylinders 在每次 hot period 開始時就搬進快閃記憶體。而第三個對照組則是一般傳統硬碟，並沒有快閃記憶體作為 Cache。

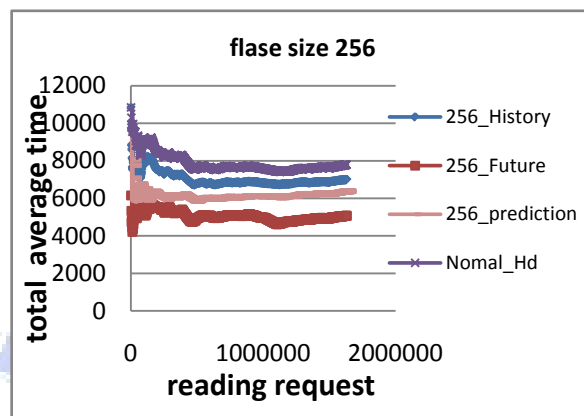
首先我們由實驗結果(圖表 23，圖表 24，圖表 25，圖表 26)可以發現效率最好的是 Future 的演算法，但是這種演算法在現實世界中是無法實作的，原因是我們無法預窺未來，因此無法預先將準確資料搬進快閃記憶體中。而以看 history 的方式以及我們的演算法是現實中比較可能的方式，搬進快閃記憶體中的資料是有可能會在未來被讀取的資料。由實驗結果可以看出，我們的演算法效率又比 history 演算法好。而且當快閃記憶體容量越大時，我們演算法的預測效果越好。這可能是因為快閃記憶體容量越大，則可以儲存在快閃記憶體裡的資料越多，所以整體命中率也可能越高。

還有由上面資料也可以看出，快閃記憶體 容量越大，我們演算法跟 Future 的演算法差異越來越大，Future 的演算法我們可以視為最好的讀取命中率，當容量越大時，可以搬進快閃記憶體的資料越多，而這些資料中會包含一些被讀取次數排名較後面的資料。這部分的資料因為被讀取次數差異並不明顯，因此我們的演算法在預測這種資料方面表現較差，因此會與最佳演算法 Future 差異越大。但是當容量越小時，可以搬進快閃記憶體的資料越少，但這些資料都是屬於被讀取次數排名較前面的資料，我們演算法對這種資料的命中率越好，因此我們的演算法會與 Future 演算法差異越小。所以我們可以知道對被讀取次數越多的資料，我們的演算法預測命中率會越好，但是對被讀取次數越少的資料，演算法命中率會越差。

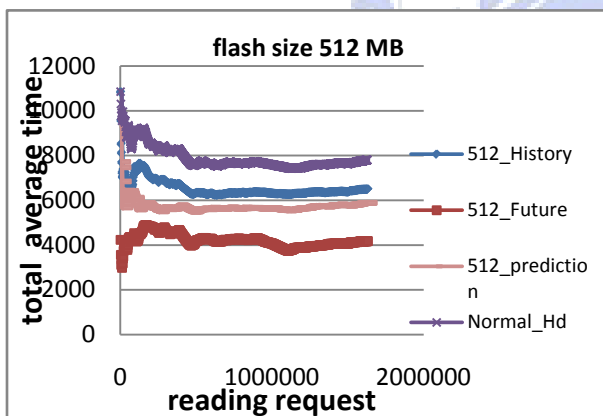
根據我們目前的實驗結果，當快閃記憶體容量為 128MB 時可以增加的效能為 12.45%、容量為 256MB 時可以增加的效能為 17.182%、容量為 512MB 時可以增加的效能為 23.468%、容量為 1024MB 時可以增加的效能為 33.175%。以效能來看，當快閃記憶體容量為 1024MB 時效果最好。



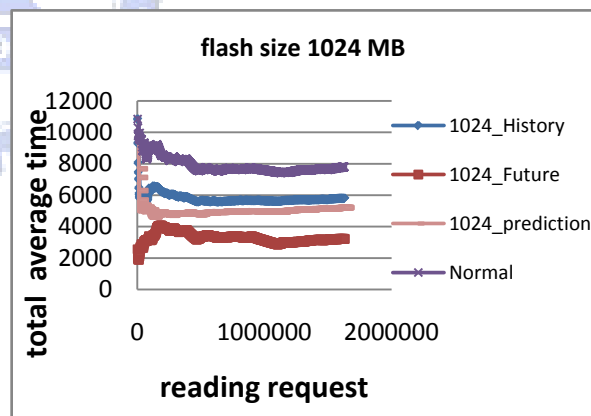
圖表 26 performance of our algorithm(Flash size : 128 MB)



圖表 24 performance of our algorithm(Flash size : 256 MB)



圖表 23 performance of our algorithm(Flash size : 512 MB)



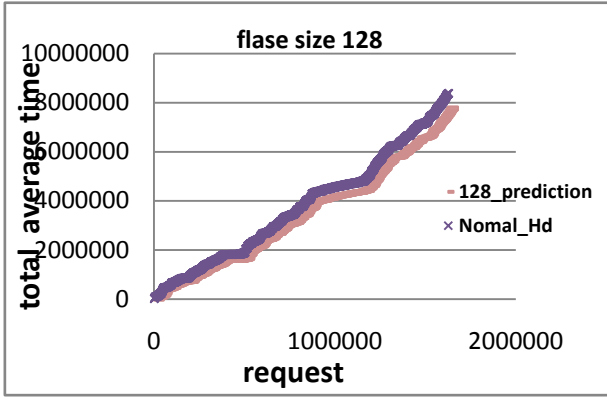
圖表 25 performance of our algorithm(Flash size : 1024 MB)

## 4.5 能源消耗

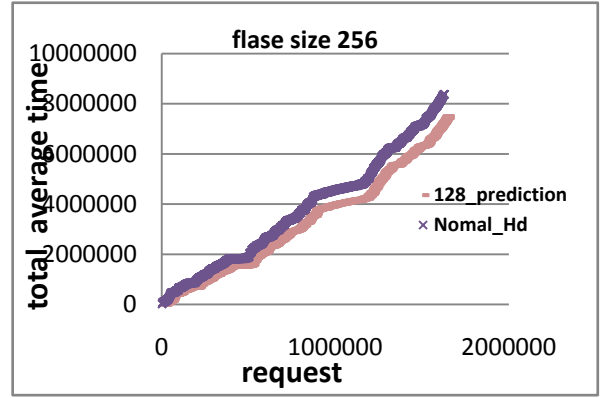
理論上我們的演算法有不錯的命中率，因此部份的讀取動作，我們可以以耗電量較少的快閃記憶體來取代耗電量較大的硬碟，因此理論上會整體耗電量有所貢獻。因此在這個實驗我們要觀察的是演算法實際上對減少系統耗電量的效果。我們以一般傳統硬碟(Normal\_Hd)讀取動作做為我們的對照組，比較以我們演算法作讀取動作所累積耗電量，以及一般傳統硬碟對相同的讀取動作的累積耗電量。看看我們的演算法減少耗電量的程度如何。由於快閃記憶體的寫入方式限制(請參考附錄 2)，一般快閃記憶卡中都有 garbage collection 的設計，因此也會花費一些能源。在我們的模擬中只統計讀寫次數所花費的耗電，並沒有直接對快閃記憶卡中內部的 garbage collection 活動作能耗的估計。

由實驗結果(圖表 27、圖表 28、圖表 29、圖表 30)可看出，當快閃記憶體容量越大則我們演算法對減少系統耗電量的效果越好，當快閃記憶體容量越小，則我們演算法在對減少系統耗電量的效果越差。這是因為快閃記憶體容量越大，則儲存在快閃記憶體裡的資料越多，因此命中率也相對越高。命中率越高會有兩個影響，第一命中率越高表示讀取快閃記憶卡的次數也越多，因此硬碟 spin up/down 次數比起一般傳統硬碟可以有效減少，因此我們可以減少系統花費在硬碟 spin up/down 的耗電。另外，由於讀取快閃記憶卡的次數越多，表示讀取硬碟次數也越少，因此硬碟在做 seek 動作次數也會相對減少，由於 seek 動作是由硬碟臂的來回移動來完成，因此我們演算法可以有效減少 seek 動作耗電量的花費。另外，讀取硬碟次數的減少，我們可以減少硬碟碟片的轉動，因此我們可以減少讀取動作所花費的耗電量。

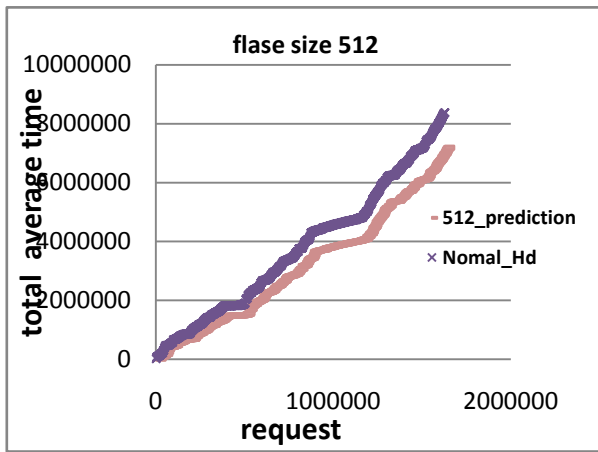
由實驗的數據顯示(圖表 27、圖表 28、圖表 29、圖表 30)，我們的演算法在快閃記憶體容量為 128MB 時，我們演算法可以節省約 7% 的耗電；快閃記憶體容量為 256MB 時，可以節省約 11% 的耗電；快閃記憶體容量為 512MB 時，可以節省約 14% 的耗電；當快閃記憶體容量為 1024MB 時，我們的演算法甚至可以節省約略達到 19% 的耗電量(圖表 31)。我們演算法可以在節省耗電量方面有不錯的表現。



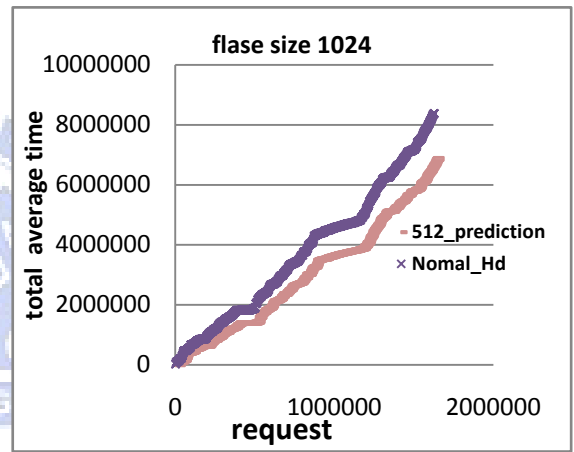
圖表 28 power consumption of our algorithm(1)



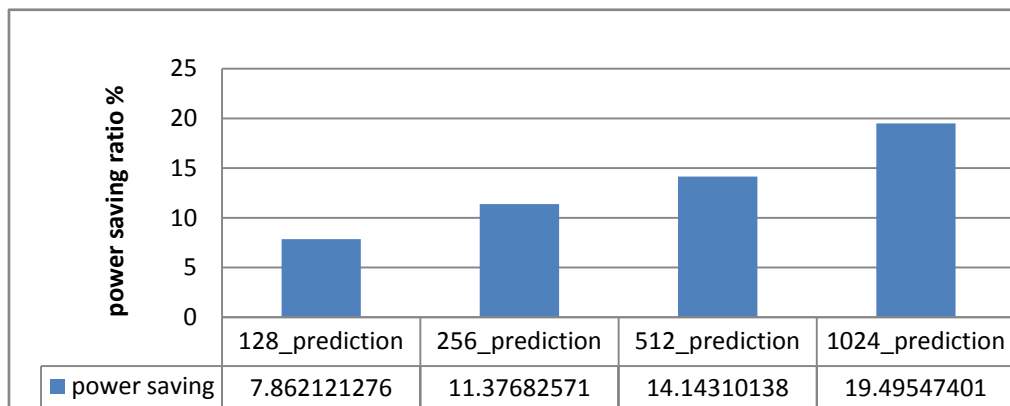
圖表 29 power consumption of our algorithm(2)



圖表 30 power consumption of our algorithm(3)



圖表 27 power consumption of our algorithm(4)



圖表 31 power saving ratio

## 4.6 快閃記憶體 endurance 的影響

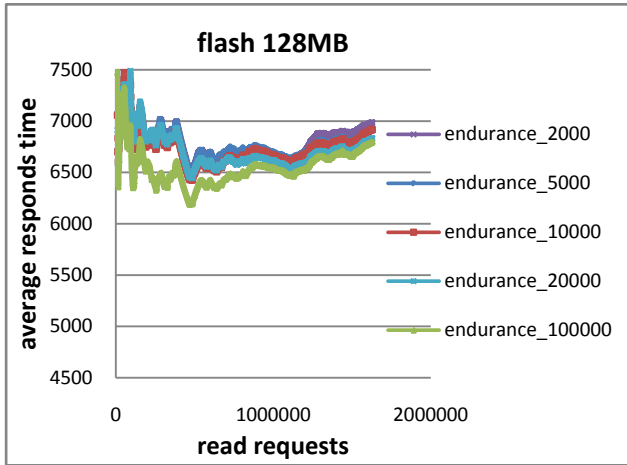
由於快閃記憶體有 endurance 的限制，因此在我們的演算會將以縮小 re-sample period(以減少 swap data 次數)的方式來減低快閃記憶體 swap in/out 次數，以避免快閃記憶體導致系統使用年限過短的問題。因此我們設計這個實驗想要從中觀察 endurance 對整體效率的影響。我們分別針對不同的快閃記憶體容量以及不同的快閃記憶體 endurance 次數來觀察對我們系統的影響。由模擬過程中我們已知當快閃記憶體 endurance 等於 100000 時我們的系統並不會發生配額不夠的情形，因此不會有使用 throttling algorithm 的機率出現。我們將以快閃記憶體 endurance 等於 100000 作為我們的對照組。

首先由實驗結果(圖表 32、圖表 33、圖表 34、圖表 35)，我們可以歸納出第一個結論，當快閃記憶體相同時，可以發現 endurance 的限制次數越大，則效率越好 endurance 次數越小，則效率越差。這是因為我們演算法以配額的觀念來做為調整 swap 次數的依據，當 endurance 次數越大，發生超過配額的機率越小。所以 endurance 次數與整體效率會成反比。因此我們選用的快閃記憶體 endurance 將會影響我們演算法對整體效率改善的效果。

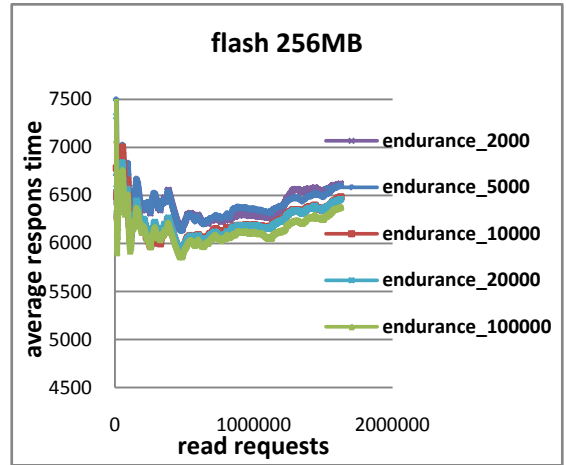
第二點，當我們觀察圖表 32 時可以發現快閃記憶體 endurance 次數等於 10000 次、20000 次以及 100000 次的平均反應時間是相同的，由於我們已知快閃記憶體 endurance 次數等於 100000 次實並不會發生配額不夠的情形，加上前面第一點結論，所以我們也可以推斷出 endurance 次數等於 10000 次、20000 次都沒有發生配額不夠的情形。但是當我們觀察圖表 34(快閃記憶體容量時 128MB)時，可以發現不同的快閃記憶體 endurance 次數，它的平均反應時間並不一樣。也就是說當快閃記憶體等於 128MB 時，除了快閃記憶體 endurance 次數等於 100000 次不會發生超過配額等情形以外，其餘都發生超過配額的情形。其他如圖表 33、圖表 35 都有類似情形。

所以我們可以得出一個結論，當我們演算法的快閃記憶體容量越小時，所使用的快閃記憶體 endurance 次數需要越大，對整體的效能越好，當快閃記憶體容量越大時，快閃記憶體 endurance 次數限制就沒有這麼嚴格。

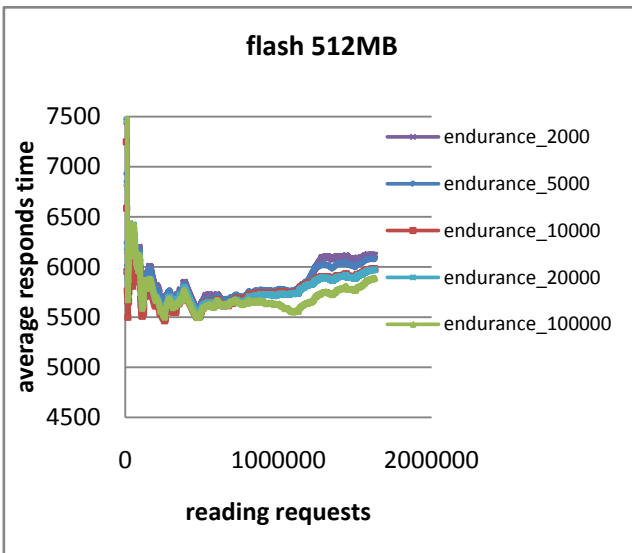




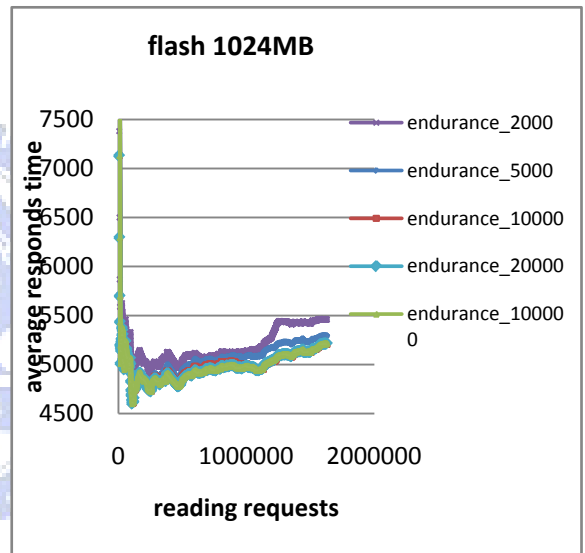
圖表 34 influence of flash endurance(1)



圖表 35 influence of flash endurance(2)



圖表 33 influence of flash endurance(3)



圖表 32 influence of flash endurance(4)

## 第5章 結論

由之前幾章的理論推演以及實驗模擬結果，我們可以證明” 結合快閃記憶卡與硬碟來增加效率與減少耗電量” 這個觀點是可行的。在之前也有人提出相關研究，但是這些研究都是針對硬碟的寫入以及硬碟的 spin up/down 方面來提出想法，因為 PC 系統 Cache memory 的影響，使得讀取行為很難被準確預測出來，所以針對硬碟的讀取行為來最佳化的研究很少看到，最多只有當讀取行為發生時，將附近的資料一起搬入快閃記憶體的方式來做為改善硬碟的方法。而直接針對讀取行為來做的預測，以改善硬碟做為研究方向更是沒有。本篇論文與其他論文最大的不同是，我們提出一套理論來預測硬碟的讀取行為。

依照我們的演算法可以得到幾個優點，首先，依照我們的架構，我們的演算法並不需要複雜的計算，整個演算法過程中，只有在計算標準差時，需要稍為複雜一點的運算，因此在演算法執行時，不需花費太多額外的系統效能；另外我們的系統不需複雜的資料結構以及程式流程，整個演算法只需要兩個資料結構以及 6 個 Function，因此在演算法實作方面，並不需花費太多的心力；另一個好處是使用我們的演算法架構，並不需要額外的硬體，也不需更改硬碟的硬體架構，只需安裝一個特殊的 block driver，因此對於使用者來說，不用額外的金錢花費；。依照我們的模擬結果，當我們快閃記憶體容量為 1024MB 時我們的讀取效能可以增加約 33.175%；我們的耗電量(不計快閃記憶體 Garbage collection 耗能)可以減少約 19.45。可以證明並且在改善硬碟效率與減少耗電量方面，我們的演算法可以有亮麗的表現

但是我們的演算法實驗的部分只有使用一份讀取的記錄資料，但是電腦的用途其實有很多種，其讀寫特性並不相同，譬如作為資料庫的電腦與 3D 畫圖所使用的電腦，其讀取的行為模式其實並不相同，而我們的演算法是否能適用在所有不同用途的電腦呢?當在不同用途的電腦上，是否也可以得出相同的效果呢?這是本篇論文比較不足之處，因此為了驗證我們演算法的可靠度，在未來我們必須要蒐集各種不同電腦的讀寫行為，使用我們的演算法來看其結果，以補足我們實驗不足的地方。另外由於我們只有將演算

法以模擬的方式來驗證其功能，因此我們只能說理論上可行，但是在真實的環境中是否可行?因此在未來我們也希望將我們的演算法實作在真實的平台上，以驗證我們演算法的可行性。



# Reference

- [1] Jen-Wei Hsieh, Tei-Wei Kuo, Yu-Chung Huang, Po-Liang Wu, "Energy-Efficient and Performance-Enhanced Disks Using Flash-Memory Cache", *International Symposium on Low Power Electronics and Design*, Pages: 334 - 339, Portland, USA, August 27 - 29, 2007
- [2] Xiaoning Ding, Song Jiang, Feng Chen, "A buffer Cache management scheme exploiting both temporal and spatial localities", *ACM Transactions on Storage*, Article No. 5, ACM New York, NY, USA, June 2007
- [3] Paul M. Greenawalt, "Modeling Power Management for Hard Disks", *Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*, Pages: 62-66, IEEE Computer Society Washington, January 31 - February 02 1994
- [4] Feng Chen, Song Jiang, Xiaodong Zhang, "SmartSaver: turning flash drive into a disk energy saver for mobile computers", *International Symposium on Low Power Electronics and Design*, Pages: 412 - 417, Bavaria, Germany, October 04 - 06, 2006
- [5] Timothy Bisson, and Scott A. Brandt. "Reducing Energy Consumption with a non-volatile storage Cache", *Proceedings of the 1st International Workshop on Software Support for Portable Storage (IWSSPS)*, IEEE, 2005
- [6] Timothy Bisson and Scott A. Brandt. "Reducing Hybrid Disk Write Latency with Flash-Backed I/O Requests", *Proceedings of the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, IEEE, 2007.
- [7] Fred Douglass, Brian N. Bershad, Brian N. Bershad, "Adaptive Disk Spin-down Policies for Mobile Computer", *Proceedings of the 2nd Symposium on Mobile*

- and Location-Independent Computing, Pages: 121 - 137 , April 10 - 11, 1995*
- [8] Li-Pin Chang, ” Hybrid Solid-State Disks : Combining Heterogeneous NAND Flash in Large SSDs” , *with EDA Technofair Design Automation Conference Asia and South Pacific, Pages 428-433, Seoul, Korea January 21 - 24, 2008*
- [9] Timothy Bisson , Scott A. Brandt. “Flushing Policies for NVCache Enabled Hard Disks” , *In Proceedings of the 15th NASA / 24rd IEEE Conference on Mass Storage Systems and Technologies (MSST), IEEE, 2007*
- [10] Athanasios E. Papathanasiou, Michael L. Scott , “Energy efficient prefetching and caching” , *USENIX Annual Technical Conference, Boston, June 27 - July 02, 2004*
- [11] Young-Jin Kim, Kwon-Taek Kwon, Jihong Kim , “Energy-efficient disk replacement and file placement techniques for mobile systems with hard disks” , *Symposium on Applied Computing , Pages: 693 - 698, Seoul, Korea March 11 - 15, 2007*
- [12] Chris Ruemmler, John Wilkes , “An introduction to disk drive modeling” , *Computer, Volume 27, Issue 3 , Pages: 17 - 28, IEEE Computer Society Press Los Alamitos, CA, USA , March 1994*
- [13] 電子月刊 2005 年十月號, ” Nand 快閃記憶體介紹, 連浩明、李明修、謝光宇”
- [14] 電子材料 2006 年 4 月號, ” 快閃記憶體原理與新進發展, 張廖貴樹、王啟照”

# 附錄 1 硬碟概觀

## I. 硬碟簡介

硬碟是電腦上使用堅硬的旋轉碟片為基礎的儲存設備。它在平整的磁性表面儲存和搜尋數位數據。信息通過離磁性表面很近的讀寫頭，由電磁來改變極性方式被電磁寫到旋轉碟片上。信息可以通過相反的方式讀回。

## II. 硬碟構造

### A. 磁頭:

磁頭是硬碟技術中最重要和最關鍵的一環。硬碟的讀、寫卻是兩種截然不同的操作，為此，這種二合一磁頭在設計時必須要同時兼顧到讀/寫兩種特性，從而造成了硬碟設計上的侷限。而 MR 磁頭 (Magnetoresistive headers)，即磁阻磁頭，採用的是分離式的磁頭結構。這樣，在設計時就可以針對兩者的不同特性分別進行優化。另外，MR 磁頭是通過阻值變化而不是電流變化去感應信號幅度，因而對信號變化相當敏感，讀取資料的準確性也相應提高。而且由於讀取的信號幅度與磁軌寬度無關，故磁軌可以做得很窄，這也是 MR 磁頭被廣泛應用的最主要原因。目前，MR 磁頭已得到廣泛應用，而採用多層結構和磁阻效應更好的材料製作的 GMR 磁頭 (Giant Magnetoresistive heads) 也逐漸普及。

### B. 碟片:

硬碟儲存資料的地方，隨著技術的演進，新製作材料的發現，單碟所含的資料量已經進步到 334GB<sup>5</sup> per media。

### C. 主軸馬達:

主軸馬達是硬碟磁片 Rotation 的時間的決定要素，目前的規格有 4200rpm、5400rpm、7200rpm、10000rpm、15000rpm 等幾種規格。而馬達型態又可分液態軸承跟滾珠軸承，使用液態軸承有下列好處：是避免了金屬面的直接磨擦，噪聲及溫度被減至最低；是油膜可有效吸收震動，使抗震能力增加。是理論上無磨損，壽命無限長

---

<sup>5</sup> 資料來源三星網站。 <http://www.samsung.com/us>

D. 驅動臂：

乘載讀寫磁頭，並可在磁軌間移動。因此驅動臂的移動速度決定硬碟的 Seek 時間，太頻繁的移動，將造成 respond time 的大幅提升，因此這也是硬碟速度無法提升的主要原因。

### III. 硬碟組織

A. 磁軌：

當磁片旋轉時，磁頭若保持在一個位置上，則每個磁頭都會在磁片表面劃出一個圓形軌跡，這些圓形軌跡就叫做磁軌。磁片上的資訊便是沿著這樣的軌道存放的。相鄰磁軌之間並不是緊挨著的，這是因為磁化單元相隔太近時磁性會相互產生影響，同時也為磁頭的讀寫帶來困難。

B. 磁區：

磁片上的磁軌若被等分為若干個區段，這些區段便稱為磁片的磁區，一個磁區可以存放 512 個位元組的資訊，磁碟機在讀取和寫入資料時，通常以磁區為單位。

C. 柱面(Cylinder)：

硬碟通常由重疊的一組碟片構成，每個盤面都被劃分為數目相等的磁軌，並從外緣的“0”開始編號，具有相同編號的磁軌形成一個圓柱，稱之為磁片的柱面。磁片的柱面數與一個盤面上的磁軌數是相等的。由於每個盤面都有自己的磁頭，因此，盤面數等於總的磁頭數。所謂硬碟的 CHS, 即 Cylinder (柱面)、Head (磁頭)、Sector (磁區)，只要知道了硬碟的 CHS 的數目，即可確定硬碟的容量。

硬碟的容量=柱面數×磁頭數×磁區數×512B。

#### IV. 硬碟傳輸介面

硬碟按傳輸介面不同，大致分為 ATA 和 SATA 以及 SCSI 和 SAS。

##### A. ATA:

全名 Advanced Technology Attachment，是用傳統的 40-pin 並列數據線連接主機板與硬碟的，外部介面速度最大為 133MB/s，因為並列線的抗干擾性太差，且排線太佔空間，不利電腦散熱，將逐漸被 SATA 所取代。

##### B. SATA:

全名 Serial ATA，也就是使用 serial port 的 ATA 介面，因抗干擾性強，且對數據線的長度要求比 ATA 低很多，支持熱插拔等功能，已越來越為人所接受。SATA-I 的外部介面速度已達到 150MB/s，SATA-II 更將升至 300MB/s，SATA 的前景很廣闊。而 SATA 的傳輸線比 ATA 的細得多，有利於機殼內的空氣流通，以利散熱。

##### C. SCSI:

全名為 Small Computer System Interface（小型機系統介面），歷經多世代的發展，從早期的 SCSI-II，到目前的 Ultra320 SCSI 以及 Fiber-Channel（光纖通道），接頭型式也有多種。SCSI 硬碟廣為工作站級個人電腦以及伺服器所使用，因為它的轉速快，可達 15000 rpm，且資料傳輸時佔用 CPU 運算資源較低，但是單價也比同樣容量的 ATA 及 SATA 硬碟昂貴。

##### D. SAS:

全名 Serial Attached SCSI 是新一代的 SCSI 技術，和 SATA 硬碟相同，都是採取序列式技術以獲得更高的傳輸速度，可達到 3Gb/s。此外也透過縮小連接線改善系統內部空間等。



## 附錄 2 快閃記憶體概觀

### I. 快閃記憶體簡介

自2000年底Flash被發明和應用以來，快閃記憶被應用的頻率迅速增加。最近十年來日益普及的行動電話、數位相機、PDA等都已廣泛利用快閃記憶體作為永久性儲存記憶體。二十多年的發展過程中，快閃記憶體技術經過了多次變革和發展。但其變化的總體趨勢一直都是：存儲容量越來越大、資料讀寫速度越來越快、性能價格比越來越高。

### II. 快閃記憶體分類

全球 Flash 記憶體的技術主要掌握在 AMD、ATMEL、Fujitsu、Hitachi、Hyundai、Intel、Micron、Mitsubishi、Samsung、SST、SHARP、TOSHIBA，由於各自技術架構的不同，分為幾大陣營。

#### A. NOR

NOR 技術（亦稱為 Linear 技術）Flash 記憶體是最早出現的快閃記憶體，目前仍是多數供應商支援的技術架構。與其它快閃記憶體技術相比，具有可靠性高、隨機讀取速度快的優勢，因此適合用在儲存程式碼。但由於 NOR 技術快閃記憶體的擦除和程式設計速度較慢，因此擦除和程式設計操作所花費的時間很長，在純資料存儲和檔存儲的應用中，NOR type 顯得相當弱勢。

#### B. DINOR

DINOR(Divided bit-line NOR)技術是 Mitsubishi 與 Hitachi 公司發展的技術，DINOR 技術快閃記憶體和 NOR 技術一樣具有快速隨機讀取的功能，按位元組隨機程式設計的速度略低於 NOR，而 block erase 速度快於 NOR flash。

#### C. NAND

NAND type 快閃記憶體 是由 Samsung、Toshiba 和 Fujitsu 共同推動。這種結構的 Flash 適合於純資料儲存和檔案儲存，主要作為 SmartMedia 卡、CompactFlash 卡、PCMCIA ATA 卡、SSD 的存儲單元，並成為 Flash Disk 的儲存核心。NAND type Flash 具有以下特點：以頁為單位進行讀和程式設計操作，1 頁為 256 或 512B(位元組)，以

Block 為單位進行 erase，1 個 Block 為 4K、8K 或 16KB，其 block erase 時間是 2ms；而 NOR 技術的 Block Erase 除時間達到幾百 ms。資料、位址採用同一匯流排，實現串列讀取。隨機讀取速度慢。晶片尺寸小，Pin 腳少，是成本最低的固態記憶體，將很快突破每百萬位元組 1 美元的價格限制

#### D. UltraNAND

AMD 與 Fujitsu 共同推出的 UltraNAND 技術。它與 NAND 標準相容：擁有比 NAND 技術更高的可靠性。儘管 UltraNAND 技術具有優勢，但在當前的市場上仍以 NAND 技術為主流。

#### E. AND

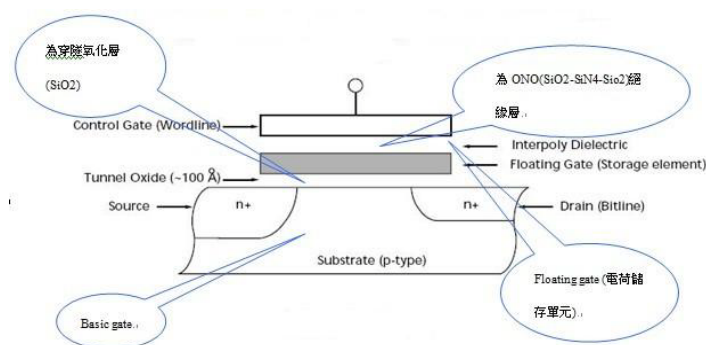
AND 技術是 Hitachi 公司的專利技術。Hitachi 和 Mitsubishi 共同支援 AND 技術的快閃記憶體。目前，在資料和檔案儲存領域中是另一種佔重要地位的 flash 儲存技術。Hitachi 公司用該技術製造 128MB 的 MultiMedia 卡和 2MB 的 PC-ATA 卡，用於智慧型電話、個人數位助理、掌上型電腦、數位相機、可攜式攝像機、可攜式音樂播放機等。

#### F. EEPROM 快閃記憶體

EEPROM 具有很高的靈活性，可以隨機讀寫（不需要擦除，可直接改寫資料），但存儲密度小，單位成本高。

### III. 快閃記憶體原理

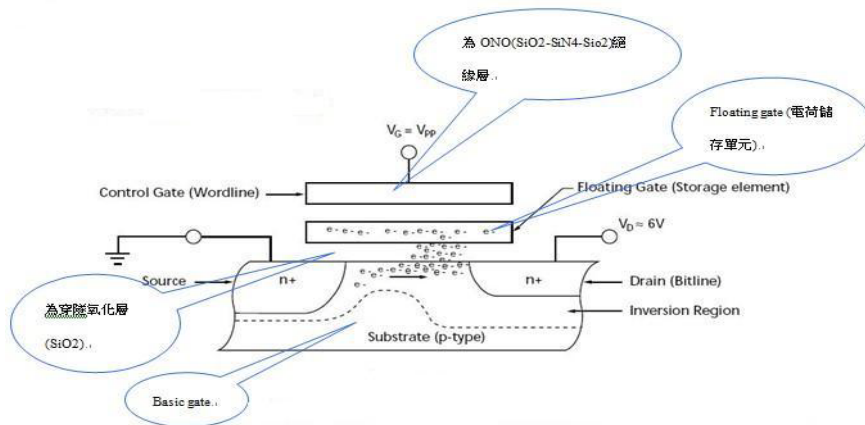
Flash 是由 Memory cell 組合而成。依照不同的組合方式，可分成不同的 Flash，如 Nand flash、Nor flash…。Flash cell 構造如圖表 36：



圖表 36 flash memory cell 構造圖

資料來源：請參考 reference[13]

## A. Flash Programming

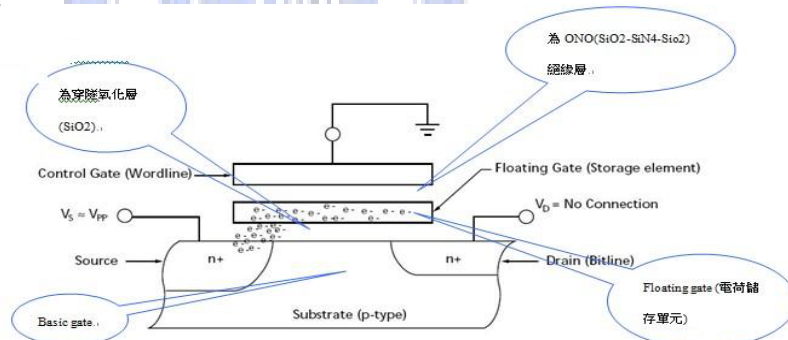


圖表 37 flash memory 寫入示意圖

資料來源：請參考 reference[13]

如圖表 37 所示，當要 program flash cell 時，將 Control Gate 上的 VG 連接到一組高電壓(通常為 12V~20V)，而 Basic Gate 的 Drain 極接上一組低電壓(0~6V)，由於壓差的關係使的電子突破穿隧氧化層，使 floating gate 可保持一個電壓  $V_t$ ，當平常不接電壓時由於 Floating gate 由絕緣層包裹起來，因此電子可以保存在 Floating gate 中，達到保存電荷(資料)的目的

## B. Flash Erase

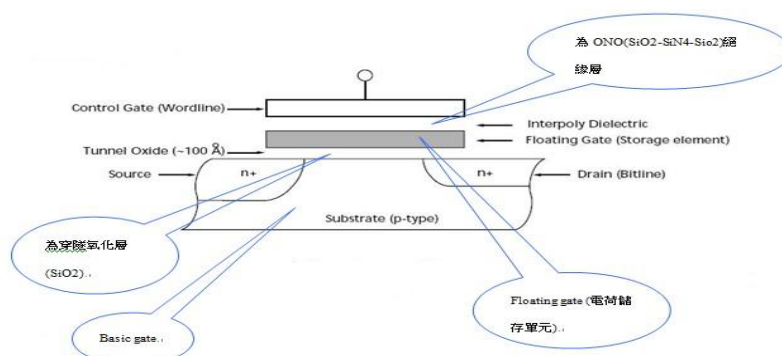


圖表 38 flash memory Erase 示意圖

資料來源：請參考 reference[13]

如圖表 38: 當要 Erase flash cell 時，將 Control Gate 上的 VG 接地，而 Basic Gate 的 Source 極接上一組高電壓(18~12V)，由於壓差的關係使的電子突破穿隧氧化層，使的 floating gate 中電子被釋放出來達到 Erase 的目的。

## C. Flash Read



圖表 39 flash memory 讀取示意圖

資料來源：請參考 reference[13]

如圖表 39: 當要 Read flash cell 時，將 Control Gate 上的 VG 接地，而 Basic Gate 的 Source 極接上一組電壓(約 1V)，如果 Floating gate  $V_t$  為 High 則 Substrate 為 P Type，因此可以偵測到電流。若 Floating gate  $V_t$  為 Low 則 Substrate 為 N Type<sup>6</sup>，因此可以偵測不到電流。藉由偵測 Substrate 的電流則可以讀出 memory cell 的值。

## IV. 快閃記憶體技術瓶頸

### A. Endurance 及 Retention

影響快閃記憶體最重要的因素為 Endurance 及 Retention

- ◆ Endurance：指 Flash 的耐久度，即 Flash 的讀寫次數。一般 Flash 在讀寫次數超過定值後，會變成非常不穩定，甚至會影響資料的存取。
- ◆ Retention：指記憶體中資料保存的年限，通常規格為 10 年。

### B. Read Write disturb

- ◆ 現象：快閃記憶體在偶然的機會下會造成 Access 不正確。
- ◆ 原因：
  1. SILC(漏電流)。
  2. CHE 應力。

<sup>6</sup> N, P type 為半導體的兩種架構，讀者可參考其他半導體理論的著作，再回頭來參考本文。

## 自 傳

我來自中部彰化市，畢業於東海大學資訊系。家中人口簡單，生活單純，父親從商，母親家管，家境小康。由於父母親對於女子的教育與學習，總是有著相當的期盼，因此家中的孩子，總也不負父母的期望，在各行各業中，均能學以致用貢獻所長

在我求學過程，一路平順，在學期間受到學校自由的學習風氣及老師們尊重學生的想法，讓我在學生生涯中，受益頗深，同時也讓我學會，可以站在不同的角度去思考事情，這樣的訓練讓在往後自主及邏輯思考增益不少。

從小我就對電子產品有濃厚的興趣，常常拆卸家中的電子產品並加以重新組合，由於個人的性向所趨，父母也尊重我的想法，因此，我在興趣與學習合而為一的理念下，便朝向與資訊有關的產業去學習。我大學唸的是東海大學資訊科學與工程，主要的學習項目的大部分集中在資訊產業有關知識上，由於學校對於基礎科學相當重視，因此我在相關課程的基礎理論的專業知識，可謂相當紮實，這樣的經驗，讓我在面對不同的工作環境與工作內容，都可以在最短的時間內上手。

在過去的職場經驗中，歷經兩年的 ERP 開發工程師，因此讓我熟悉 ERP 觀念與系統開發流程（在專業技能上熟 JAVA, application server, database, JSP, three tier 架構）。另外我有 3 年的韌體工程師經驗，專長在於以 RTOS 為平台的 Embedded 系統開發，開發的產品為 MFP。由於系統的需要，曾經接觸的部分有(DSP 影像處理, flash memory 韌體開發, ARM(ARM9, ARM7)的韌體開發, RTOS(Nucleus, threadx, File System, USB 韌體開發, MEMORY CARD 韌體開發)。

在這個瞬息萬變的產業中，保持彈性與充沛的精力，加上時時不停學習的心，更是讓自己永不停頓的重要能源；在我的生命中，不斷的學習更是挹注我生命成長的重要養份，由於有感於自己的能力與學歷的不足，所以在 95 報考交通大學的資訊學院碩士在職專班。很幸運的以免於口試的成績考上專班，並且請張立平教授擔任指導教授，從事跟工作相關的研究，深刻的體會到學問與工作相互結合與驗證。

我是以積極的態度來面對自己的人生，會努力的爭取所有的機會。而最好的投資就是自己。我相信目前的學歷不會是我的學習終點，仍然規劃 MBA 或者是博士的深造機會。我不知道未來我會不會成功，但是我

希望在機會來到時，我已經準備好了。

