View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Repository@USM

Minimizing the Garbage Collection Time in Flash Memory using Efficient Data Allocation Scheme

Amir Rizaan Rahiman
Multimedia Research Group
School of Computer Sciences, Universiti Sains Malaysia,
11800 Penang, Malaysia
amir@fsktm.upm.edu.my

Putra Sumari
Multimedia Research Group
School of Computer Sciences, Universiti Sains Malaysia
11800 Penang, Malaysia
putras@cs.usm.my

Abstract—**Recently, flash memory is becoming a popular data storage device in most of the electronic consumer devices. It has lots of attractive features such as small size and light weight nature, zero noise, solid-state reliability, low power consumption, and better shock resistant. However, its two hardware characteristics, namely, i) out-place updating and ii) garbage collection process are affecting flash memory performance if these characteristics are not well-organized. To overcome these constraints, we propose an efficient page allocation scheme that based on the occurrences of page data in the data access pattern. In the scheme, we have classified the page data into hot and cold data and allocated them into different blocks. The performance of the allocation scheme is confirmed by trace-driven simulations and the merit of the proposed scheme is justified in terms of a number of active block requirements. The number of active block requirement is reduced into 16% in comparison to the existing schemes.**

Keywords-flash memory; garbage collection; active block; trace-driven simulation;

I. Introduction

Currently, flash memory is becoming a popular and the most demanding data storage media in many embedded systems, mobile computing devices and electronic consumer devices (e.g. MP3 players PDA's, hand phones, digital cameras, etc. [1]. In general, flash memory is a non-volatile storage device that needs no power to retain its contents. Furthermore, as a data storage device, it had offered several superiority features such as small and lightweight nature, fast data access, zero noise, solid-state reliability, low power consumption, and better shock resistant [1 – 4].

However, flash memory has two hardware characteristics, namely i) out-place updating scheme, and ii) garbage collection process that will bring several challenges in terms of data management issues. Since flash memory is a form of EEPROM, updating the existing data by overwriting at same physical locations are strictly prohibited. The original data must be erased prior to the updated data can be stored on same physical location. To avoid initiating an erase operation during every update process, out-place updating scheme has been proposed in the literatures [5 – 7]. In out-place scheme, the updated data (latest version) is stored into a new page while the original data are set as *garbage*[1]. However, this updating policy increases the amount of garbage that reduces free spaces when a large number of data updating processes occurs. The garbage collection process is associates with the process of cleaning the garbage when the free space availability in flash memory attained at a certain level of threshold (e.g., 20 – 35% of total space). This cleaning policy can be also initiated periodically [8]. Furthermore, it is time and power consuming process since it is carried out by the erase operation (involve bulk sizes of data), and cannot be executed on specific area (page). Before the erasure process can be initiated, the valid data in the block must be copied into other blocks.

Various studies have been proposed with the aim to improve the garbage collection process performance [8 – 11]. Most studies are focusing on determining a victim block that can minimize the garbage collection time. We refer the garbage collection time to amount of access time required in erasing the victim block. Several keys including block utilization, block locality, erasure counts, age of data, etc., can be used in determining the victim block that minimized the garbage collection time. Although these studies have improved the performance of garbage collection, but they give a more concentration on preserving free space availability. As pointing out by [11], initiating the garbage collection process without a proper allocation scheme could impose approximately 40 seconds of blocking time in time-constraint applications. Besides, few studies have shown that the well-organized data allocation in flash memory could influence the garbage collection performance. Due to this fact, it is motivating us to propose an efficient data allocation scheme that can minimize the garbage collection time. In this paper, we proposed an allocation scheme that distributes the accessed data in flash memory based on the frequency of appearance of each page data. This scheme is inherent from the inspiration of the Frequency Based Replacement (FBR) page replacement policy [12]. We compare the performance of our proposed work with existing allocation schemes in terms of the number of active block requirements by using the similar data access pattern.

The rest of this paper is organized as follows. In Section II, we discuss about flash memory technology and related works that are relevant for this paper. In Section III, we present our

[1]Garbage in flash memory refers to "dead" data and shall not be used anymore.

proposed allocation scheme that based on data appearance frequency. Section IV presents the experimental results to show the performance of proposed scheme and finally, the conclusions of this paper are presented in Section V.

II. Background and Related Works

Recently, flash memory performances, price, and popularity are improving so fast. As reported by Min and Nam, flash memory capacity has been doubled every year for past few years and the price-per-byte is falling [13]. Due to these facts, we believe flash memory will successfully turn into the most demanding storage device, and it will bring numerous innovation and smart applications using it.

A. Flash memory characteristics

Two types of flash memory in the markets are: i) NOR-flash, and ii) NAND-flash. The comparisons for both types have been discussed extensively by [13]. In this paper, we limit our focus on NAND-flash and referred it as flash memory. Flash memory architecture and operational characteristics can be summarized as follows:

- **Block and page architecture.** As shown in Fig. 1, flash memory is organized in terms of a fixed number of blocks where each block evenly contains a fixed number of pages. Each page further contains two areas: i) data area (store the actual data), and ii) spare area (data assisting information e.g., ECC, time, etc.).
- **Asymmetric operational units and access time.** Flash memory offers three types of operational functions, called, i) read, ii) write, and iii) erase [14]. Each function is carried out with asymmetric accessing time and accessing unit (see TABLE I).
- **No in-place update.** Overwriting an existing data in flash memory is performed via out-place scheme rather than in-place scheme.
- **Limited block life-span.** Each block could be tolerant with a limited number of erasure cycles (e.g., 1 million cycles). Wear-leveling policy is employed in order to lengthen block life-cycles [2].

Due to out-place updating scheme, page in flash memory can be in one of the following three states, i) free, ii) valid, and iii) invalid. The free page refers to a page that contains no data, and it is available to store new or the updated data. The valid page is the page that contains the recently version of data while the invalid page contains "dead" data. According to these page states, block in flash memory can be clustered as active or inactive [15]. The active block refers to a block that contains valid pages (at least one valid page) while block that contains either free or invalid pages refers as inactive. The active block cannot be simply erased while the inactive block with invalid pages can be erased at any time since it needs no valid page copying.

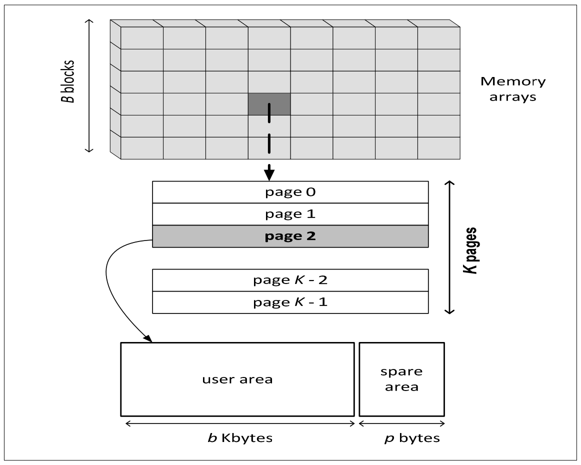

Figure 1. The basic architecture of flash memory.

TABLE I. Flash Memory Properties

Specification	Details
Power consumption	2.70 – 3.60V
Size (h x w x d)	12 mm x 20 mm x 1.2 mm
Page size (KB)	2
Block size (KB)	128
Read Write Erase	25 µsec 200 µsec 1.5 msec
Data endurance	100 K (write and erase cycles)
Data retention	10 years
Noise	~ 0 db

B. Garbage collection process in flash memory

Garbage collection process employed by flash memory to preserve free space availability due to the out-place policy. As illustrated in Fig.2, garbage collection process is realized in three stages.

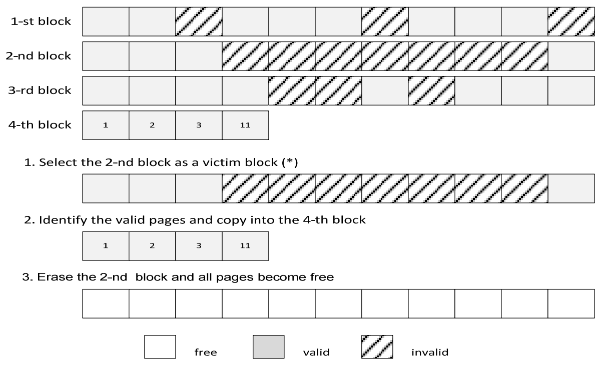
1-st block
2-nd block
3-rd block
4-th block
1
2
3
11
1. Select the 2-nd block as a victim block (*)
2. Identify the valid pages and copy into the 4-th block
3. Erase the 2-nd block and all pages become free
free
valid
invalid

Figure 2. Garbage collection process in flash memory.

First, a victim block is selected by using several parameters such as the block utilization, block erasure count, data age, cleaning cost, etc. Then, after the victim block is recognized, valid pages resided in the block are identified and copied into free pages in other blocks. Upon completion copying the last valid page, the victim block is erased.

Garbage collection time (denoted by λ) refers to total access time required in handling the garbage collection process. It includes several read (ρ) and writes (ω) access time and a single constant erasure (ε) time. Moreover, the access times can be clustered into two principal access times, called erasure-time and copying-time. The erasure-time is a constant time used to erase a block while the mutable copying-time is used for copying the valid pages into new blocks. In short, both access times can be simplified in the following formula.

$$\lambda = \eta * (\rho + \omega) + \varepsilon \qquad (1)$$

Parameter η refers to number of valid pages resided in a victim block. A victim block with more valid pages results in higher garbage collection time and Equation (1) is used for erasing a single block. Furthermore, the total access time required in handling different level of a victim block fraction with a different level of block utilization is shown in Fig. 3.

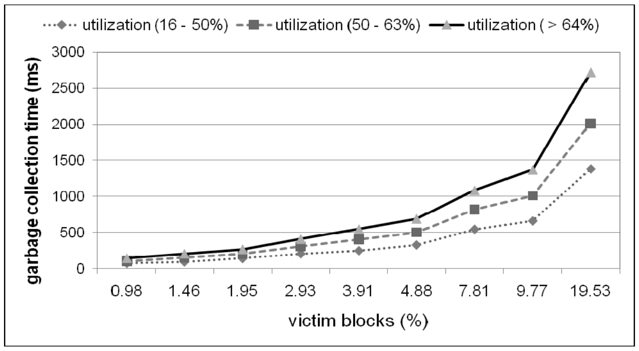

Figure 3. Garbage collection time for different level of victim block fraction.

From Fig. 3, it is clear that garbage collection time is directly proportional to the fraction of victim block. A victim block is an active block since it may contain valid page. When the number of active block increase, the number of block involves in the garbage collection process will also increase. Thus, the number of active block becomes the significant indicator in determining the performance of the garbage collection process.

C. *Page allocation algorithms*

There are many page allocation algorithms with the aim to minimize a number of active blocks have been proposed in the literature [15–16]. We summarize these algorithms in TABLE II.

TABLE II. PAGE DATA ALLOCATION ALGORITHMS

Algorithm	Method	Parameter usage
FCFS (First Come First Serve)	Online	Page arrival position in the data access.
FRFS (First Re-Arrival First Serve)	Offline	Page re-arrival position in the data access.
OFRFS (Online First Re-Arrival First Serve)	Online	Prediction of previous page arrival position history.
BestM (Best Match)	Offline	Length between the arrival time and re-arrival time.

The online allocation algorithm allocates each page in the data access pattern when it appears into flash memory. Conversely, the offline allocation is delaying the allocation of each page until all information of pages is analyzed. The offline allocation results in a minimum number of active blocks in the allocation process, but it needs to know entire information of the pages in the access pattern. Therefore, the offline method is unsuitable for the time constraint applications which need to allocate when the page appears.

The terms of data access pattern refer to a sequence of updating the partitioned page's data [15]. For instance, as shown in Fig. 4(a), file A is partitioned into 10 pages denoted by *a, b, c… j*. After certain time of an interval, several pages (or all pages) have been updated (see Fig. 4(b)). Thus, the data access pattern containing a sequence of updated pages can be shown in Fig. 4(c).

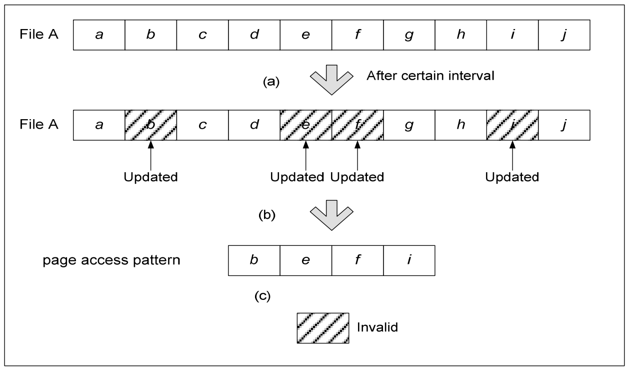

Figure 4. Data access pattern consist of sequence of updated pages.

III. PROBABILITY BASED PAGE ALLOCATION SCHEME

A. *Page popularity distribution*

Previously, the arrival position of each page in data access pattern is used to determine the location of the page. By using the arrival position, the page is stored sequentially on the block.

In our scheme, we propose an allocation scheme that takes into account the popularity of each unique page in data access pattern. Since each file data is partitioned evenly into a fixed number of different pages, each page may appear different times (k) in access pattern. By using the tracing file, the number of each page appears in the access pattern can be knowable first. The most appearance pages are denoted as popular page, while the pages that appear least time are denoted as unpopular page. We use Zipf's [17] popularity distribution to determine the popularity distribution of each page data. In this distribution approach, the page data is sorted in an increasing order from the highest rank (rank 1) to the lowest rank (rank n) based on their frequency of occurrences in the access pattern. The page data with the highest frequency is assigned to rank 1 (the most popular page) while the lowest frequency (unpopular page) is assigned to rank n (n denotes the number of different page data in the access pattern).

B. Allocation architecture

The page data allocation architecture is shown in Fig. 5. The architecture consists of two main components: i) access screening that percolates data access types either read or write, and ii) allocation algorithm that perform the distribution process of the accessed data.

The page data allocation architecture is shown in Fig. 5. The architecture consists of two main components: i) access screening that percolates data access types either read or write, and ii) allocation algorithm that perform the distribution process of the accessed data. The access screening is responsible in filtering the data access type. In flash memory, two types of data requests that can be initiated from the users or host systems are read and write. If a request is read, then the data can be directly retrieved from the memory array since it does not change the page state. If a request is write, then the request need to go through the allocation algorithm before can be access to the memory array.

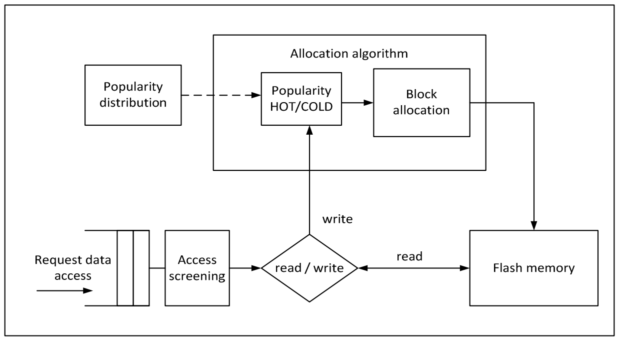

Figure 5. Page data allocation architecture.

The allocation algorithm contains two sub components: i) popularity interpreter, and ii) an allocation engine. The popularity interpreter uses the accessed data popularity probability in determining the state of the accessed data, either hot or cold. The allocation engine performs the block selection process in which the accessed page data fall into. After the block has been determined, the accessed data is stored into first free space of the allocated block. Furthermore, when all pages in a block are invalid (block is inactive), it can be erased directly without the need to waiting until the garbage collection process.

C. Popularity based (PB) allocation scheme

Each partitioned file data is allocated with its own probability value and we refer it as weight. Some page data may share their probability value.

The available page weight is clustered into three groups: i) maximum weight (μ), ii) median weight (γ), and iii) minimum weight (η). The pages data that has weight value greater than or equals to γ are classified as popular hot since it will arrive frequently in the access pattern, while the remaining pages are classified as cold page. When each page arrived, its weight is compared with the probability distribution values to determine in which groups the page fall into.

Furthermore, blocks in flash memory (denoted by β) are divided into two categories: i) hot and ii) cold. Since the hot page will appear frequently in the access pattern, the number of blocks that partitioned into hot should be more than the cold blocks. Specifically, the portion for hot blocks is a fraction of μ and γ value from the probability distribution. In short, the functions given in Equation (2) and Equation (3) denote the number of block portion for hot (φ) and cold (δ).

$$\varphi = (\mu + \gamma) * \beta \qquad (2)$$

$$\delta = 1 - (\mu + \gamma) * \beta \qquad (3)$$

The motivation behind this scheme is that by classifying the pages into hot and cold and allocating them to different blocks, we will eventually turn blocks not to be active for the whole allocation process. When such blocks are active during the whole allocation process, it will increase the number of active blocks. In FRFS scheme, the number of active blocks required for the allocation is minimized because such block could be in the active state for a short period of an interval. Thus, to minimize the number of active block requirement, we try to diminish each block to be active in the whole allocation scheme. We simplify the allocation of each arrival page data as follows:

```
1. for each page in the access pattern.
2.   get page weight ($\alpha$).
3.     if $\alpha >= \gamma$
         set page as hot.
         write page into first free page within blocks
         [0, $\varphi - 1$].
4.     else
         set page as COLD.
         write page into first free page within blocks
         [$\varphi, \beta - 1$]
5.   if new block → active block + 1
6.     If block inactive → active block - 1
7. count maximum active blocks.
```

When all pages in the block become invalid, the block can be erased and the blocks ID within each group are reconfigured back. If current active block ID is i and block $i - 1$ is inactive, block $i - 1$ is erased and block ID i is set to $i - 1$.

IV. Implementation and Performance Evaluation

Using the real data from the tracing file, we perform an experiment in evaluating the performance of the proposed scheme. The tracing file is collected over a personal computer that running general applications (such as Web browser, PowerPoint, Word, Acrobat reader, and email application) [18].

By using the current flash memory specification [14] we run our proposed allocation to determine the number of active blocks required. In the tracing file, the total arrival page data in the access pattern is 45000 containing 12000 different page data. The highest data page appearance is 4780. We compare our proposed allocation scheme with the FCFS scheme. Fig. 6

shows the average number of active blocks required for the tracing file.

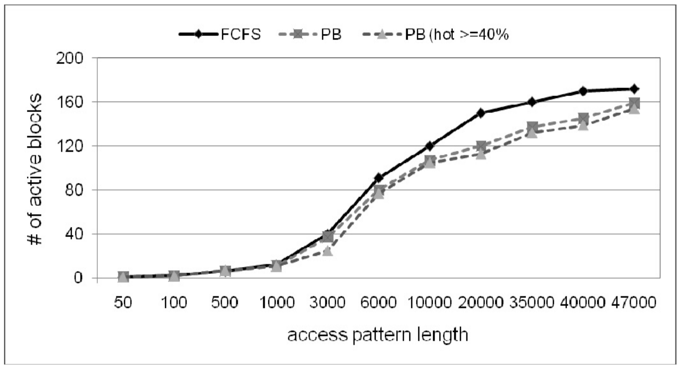

Figure 6. Average number of active block requirement.

From Fig. 6, the PB allocation scheme performs better than the FCFS scheme. The number of active block requirement is reduced at approximately 16% from the FCFS scheme when the length access pattern reaches 6000 arrivals. In addition, when we reduce the hot block group to 40% of the γ value, the number of active block required is reduced. The main reason that PB outperforms the FCFS is because the pages those have weight greater than the γ is stored within same blocks. When we allocate pages that have similar weight into a same group, we eliminate a block from being active all time during the allocation process. By cutting the block lifetime to be active, the number of active blocks can be minimized. The FRFS using this similar approach, but the scheme needs information of all pages in the access pattern and not suitable in online environment.

V. CONCLUSION

This paper discusses the allocation scheme that reduces the access time for garbage collection process. We propose the allocation scheme called probability based (PB) for allocating the arrival pages in the access pattern. From the experiment result, our proposed scheme reduces the number of active blocks required for the allocation. The main idea of using the probability is to minimize the period of each block to be an active state. By minimizing it, the block can become inactive earlier, thus minimize the number of active blocks. The number of active block requirements can increase when the cold pages are stored together with the hot pages. When we do clustering, we allocate different blocks for the cold and hot pages and the number of active block required is minimized.

ACKNOWLEDGMENT

This research was supported in part by the Research University Post-graduate (USM-RU-PGSR) grant (no. 1001/PKOMP/832006) funded by University Sains Malaysia.

REFERENCES

[1] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and J. A. Tauber, “Storage alternatives for mobile computers,“ Proc. 1st Symp. Operating Systems Design and Implementation (OSDI), pp. 25–37, 1994.

[2] L. Chang and T. Kuo,”An efficient management scheme for large-scale flash-memory storage systems,” Proceedings of the 2004 ACM Symposium on Applied Computing (SAC’04), pp. 862–868, 2004.

[3] M. Breeuwsma, M. d. Jongh, C. Klaver, R. v. d. Knijff, and M. Roeloffs, ”Forensic data recovery from flash memory,” Journal of Small Scale Digital Device Forensic, vol. 1, no. 1, pp. 1–17, June 2007.

[4] G. Lawton, "Improved flash memory grows in popularity,” Computer, vol. 39, no.1, pp. 16–18, January 2006.

[5] Memory Technology Devices, http://www.linux-mtd.infradead.org/doc/nand.html.

[6] A. Kawaguchi, S. Nishioka, and H. Motoda,“ A flash-memory based file system,” Proceedings of 1995 USENIX Annual Technical Conference, pp. 155-164, 1995.

[7] E. Gal and S. Toledo, “A transactions flash file system for microcontrollers,” Proceedings of the 2005 USENIX Annual Technical Conference, pp. 89-104, 2005.

[8] M. L. Chiang and R. C. Chang,“ Cleaning policies in mobile computers using flash memory,“ Journal of Systems and Software, vol. 48, no. 3, pp. 213-231, November 1999.

[9] L.-P. Chang,” On efficient wear leveling for large-scale flash-memory storage systems,” Proceedings of the 2007 ACM symposium on Applied computing, pp. 1126 -1130, 2007.

[10] M. Rosenblum and J. K. Ousterhout,“ The design and implementation of a log-structured file system,” ACM Transactions on Computer Systems, vol. 10. no. 1, pp. 26-52, 1992.

[11] V. Malik,“ Jffs2 is broken,” Mailing List of Memory Technology Device (MTD) Subsystem for Linux, June 2001.

[12] J. T. Robinson and M. V. Devarakonda,” Data cache management using frequency based replacement,” Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp. 134–142, May 1990.

[13] S. L. Min and E. H. Nam,” Current trends in flash memory technology,” Asia and South Pacific Conference on Design Automation, pp. 332 – 333, 2006.

[14] K9K8G08U1A & K9F4G08U0A_Data Sheet for 512M Bit and 1G Bit NAND Flash Memory SAMSUNG.

[15] L.-F. Chou and P. Liu,“ Efficient allocation algorithms for flash file systems,” 11th International Conference on Parallel and Distribution Systems, pp. 634 – 641, 2005.

[16] P. Liu, C.–H. Chuang, and J.–J. Wu,” Block-based allocation algorithms for flash memory in embedded systems”, PaCT, pp. 569–578, 2007.

[17] G. K. Zipf, Human Behavior and the Principle of Least-Effort Addison-Wesley . 1942.

[18] http://newslab.csie.ntu.edu.tw/~flash/index.php?SelectedItem=Traces#