Journal of Systems Architecture 58 (2012) 221–232

Contents lists available at SciVerse ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

An efficient method for record management in flash memory environment

Duck-Ho Bae [a], Ji-Woong Chang [b], Sang-Wook Kim [a,*]

[a] Department of Electronics and Computer Engineering, Hanyang University, 17 Haengdang-dong, Seongdong-gu, Seoul 133-791, Republic of Korea
[b] Department of Game & Multimedia Engineering, Korea Polytechnic University, 2121 Jeongwang-dong, Siheung-si, Kyonggi-do, Republic of Korea

ARTICLE INFO

Article history:
Received 25 May 2011
Received in revised form 4 February 2012
Accepted 27 March 2012
Available online 4 April 2012

Keywords:
Flash memory DBMS
Record management
FARM
Group write
k-LASL

ABSTRACT

Flash memory has its unique characteristics: the write operation is much more costly than the read operation, and in-place updating is not allowed. In this paper, we analyze how these characteristics affect the performance of clustering and non-clustering methods for record management, and show that non-clustering is more suitable in flash memory environment. Also, we identify the problems of the existing non-clustering method when applied to flash memory environment without any modification, and propose an effective method for record management in flash memory databases. This method, which is basically based on the non-clustering method, tries to store consecutively inserted records in the same page in order to make it possible to process them with only one write operation. In this paper, we call this method *group write*. Moreover, we propose two novel techniques for achieving efficient group writes: (1) dedicated buffers for group writes and (2) free space lists managed in main memory for maintaining only those pages having large free space. Our method greatly improves the write performance of database applications running in flash memory. For performance evaluation, we conduct a variety of experiments. The results show that our method achieves speed up by up to 1.67 times compared with the original non-clustering method.

1. Introduction

Characterized by low power consumption, non-volatility, and high mobility, flash memory is widely used as storage media for mobile devices [7]. Unlike other storage media, flash memory has its unique characteristics: first, the difference between speeds of read and write operations is significant with flash memory [4,8]. Second, the physical characteristic of flash memory makes the in-place update impossible. To update data, its containing page[1] has to be invalidated first, and then newly updated data is written onto a different physical page [28,30]. Third, flash memory requires the erase operation to resume the invalidated page. The erase operation requires a very high cost and occurs mainly due to frequent write operations, particularly when updating existing pages [14]. Therefore, it is very important to reduce the number of write operations in flash memory.

Recently, as a capacity of flash memory grows rapidly, the application of flash memory as alternative storage media to disk is increasing. This naturally raised the necessity of studies for flash-based DBMSs to find out ways that allow to efficiently store and manage massive data directly on flash memory.

With a proper software layer such as Flash Translation Layer (FTL), which makes flash memory behave as a block device like disk, we can employ the same methods used in conventional disk-based DBMSs without any modification [16]. However, conventional disk-based DBMSs do not consider the characteristics of flash memory at all, this approach fails to provide good performance.

In order to overcome such limitations, researches on flash-based DBMSs have been performed. A typical storage system in a conventional DBMS consists of five layers: storage, buffer, index, record, and scan management layers [9]. Previous studies mainly focused on the prior three layers, i.e., flash-aware methods for storage [7,15], buffer [18–20], and index [1,17] management. However, to the extent of our knowledge, there have been no previous studies on a record management method that handles the insertion, deletion, and placement of records, and significantly affects the overall performance of a DBMS.

Two types of record management methods are widely used in disk-based DBMSs: clustering and non-clustering [9]. With the clustering method, records with similar key values are stored at physically adjacent pages. Whereas, with the non-clustering method, records are stored at pages having available space regardless of their key values. In disk-based DBMSs, the clustering method is more popular than the non-clustering one because the performance of range queries can be significantly improved if the records of similar key values are stored in the same or physical adjacent pages [9].

* Corresponding author. Tel./fax: +82 2 2220 4567.
E-mail addresses: dhbae@agape.hanyang.ac.kr (D.-H. Bae), jwchang@kpu.ac.kr (J.-W. Chang), wook@hanyang.ac.kr (S.-W. Kim).

[1] There are two types of pages: a logical page and a physical page. For brevity, we simply refer to a logical page as a page.

1383-7621/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2012.03.004

Since the location of a record in the clustering method is totally determined by its key value, the insertion of n records generally places them into n different pages, respectively, hence requiring n page accesses. As a result, it normally incurs n write operations. Such a task would not pose any problem in disk environment. However, it causes serious problems in flash memory environment due to the speed difference between read and write operations [2]. Furthermore, they incur frequent erase operations since most of write operations for record insertions are to update pages.

On the other hand, in case of the non-clustering method, its inefficiency in range query processing would be compensated by the high speed of a read operation in flash memory. The non-clustering method is also expected to show superior performance of record insertions owing to the buffering effect that reduces the number of write operations and also the number of erase operations [2]. Through extensive preliminary experiments, this paper indeed shows the superiority of the non-clustering method over the clustering counterpart in overall performance in flash memory.

However, there still arise several problems with the non-clustering method because it was not specifically designed to cater for the characteristics of flash memory. First, a number of write operations may be required in the same page, due to buffer replacement. Second, the free space list is frequently updated in case of the repetition of record insertions and deletions.

The management of a free space list is used in both clustering and non-clustering methods for maintaining those pages that contain free space for future record insertions [9]. Since pointers to form a list are stored within pages in disk-based DBMSs [9], this makes any modification of the list necessitate write operations on pages, too. Thus, the management of a free space list could incur serious performance degradation in flash memory environment.

In this paper, we propose an effective Flash-Aware Record Management (FARM) method. This method is basically based on the non-clustering method, and tries to store consecutively inserted records in the same page, thereby making it possible to store them with only one write operation. In this paper, we call this method as *group write*. For efficient support of group writes, we propose a dedicated buffer for group write and a free space list for maintaining in main memory only a small number of pages having large free space. To show the superiority of the proposed method, we conduct a variety of experiments for performance evaluation. The results show that our method achieves speed up by up to 67% compared with the original non-clustering method.

This paper consists of the followings: in Section 2, characteristics of flash memory and related work on flash-based DBMSs are introduced. And the difference between related work and our proposed method is explained. In Section 3, as existing record management methods, clustering and non-clustering methods are explained and their new problems arising in flash memory environment are addressed. In Section 4, performance analysis on clustering and non-clustering methods in flash memory environment are presented. In Section 5, our new method is described in detail. In Section 6, performance analysis is conducted to verify the superiority of our proposed method. Finally, in Section 7, this paper is wrapped up with conclusions.

2. Related work

2.1. Flash memory: a background

In this section, we introduce the characteristics of flash memory. Read and write operations are carried out in a 512 B page while an erase operation is done in a 16 KB block that is composed of multiple pages [2]. Table 1 shows the execution time of read, write, and erase operations in flash memory, main memory (DRAM), and hard disk [25–27]. In flash memory, a write operation and an erase operation require their processing time longer than a read operation, 10 times and 75 times, respectively.

Table 1
Processing time taken for each operation type in various storage media.

	Read operation	Write operation	Erase operation
NAND flash	20 μs/512 B	200 μs/512 B	1.5 ms/16 KB
DRAM	39 ns/1 KB	39 ns/1 KB	–
Hard disk	5.7 ms/1 MB	5.7 ms/1 MB	–

Recent studies showed that the speed gap between read and write operations would further increase 3.5–10 times once flash memory is fragmented [6]. Therefore, even if the initial speed gap between read and write operations is getting smaller owing to the recent hardware and software techniques, the minimization of write operations is still crucial for improving overall system performance.

In flash memory, stored data cannot be in-place updated. This characteristic causes a high-level of overhead as follows: first, since store the newly updated data onto a different page, if there is no free page for allocation in the same block, then all the valid pages in the block have to be copied to another block.[2] Second, frequent page updates give rise to erase operations. When a page update operation is executed, the page having old data is invalidated. In order to use this invalid page again, we need to make it cleared by an erase operation.

When a block is erased, all the valid data within the block needs to be copied to another block first, which is also very costly [17]. Furthermore, the number of erase operations allowed in a block is limited [31]. Once this limit is reached, the block can no longer be used. Therefore, the erase operation shortens the service life of flash memory.

Consequently, one has to consider a way to reduce the number of write operations, particularly update operations, when designing a record management method for flash memory. Based on this observation, we investigate clustering and non-clustering methods, both of which are widely used for record management, in the view point of flash memory.

2.2. Flash Translation Layer

Flash Translation Layer (FTL) makes flash memory appear to a block device like disk, and is typically stored within flash memory [16]. One of the important roles of FTL is to map a *logical* page address, used by a file system or a DBMS, to a *physical* page address in flash memory. Owing to the mapping, we can apply the same methods used in conventional disk-based DBMSs to flash memory environment without modifications.

The mapping can be managed at the page-level, block-level, or hybrid [3,10,16,22].

- In *page-level mapping* [10], a logical page can be mapped to a physical page in any location of flash memory. Therefore, page-level mapping can greatly reduce *valid pages copying cost*. However, the size of the page-level mapping table is too large to be managed in main memory, it is hardly used in embedded environment.
- In *block-level mapping* [3], a logical block, which consists of a set of logically adjacent pages, is mapped into a physical block in flash memory. A logical page is mapped to a fixed offset of its

[2] This happens only when block-level mapping and hybrid mapping are used in FTL as an address assigning method and does not happen in page-level mapping. However, page-level mapping is hardly used in embedded environment where the amount of main memory in flash disk is limited.