Data & Knowledge Engineering 72 (2012) 83–102

Contents lists available at SciVerse ScienceDirect

Data & Knowledge Engineering

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

AD-LRU: An efficient buffer replacement algorithm for flash-based databases

Peiquan Jin, Yi Ou, Theo Härder *, Zhi Li

School of Computer Science and Technology, University of Science and Technology of China, China
Department of Computer Science, University of Kaiserslautern, D-67663 Kaiserslautern, Germany

ARTICLE INFO

Article history:
Received 17 June 2010
Received in revised form 14 September 2011
Accepted 16 September 2011
Available online 7 October 2011

Keywords:
Flash memory
Database buffer management
Replacement policy
Flash-based DBMS

ABSTRACT

Flash memory has characteristics of out-of-place update and asymmetric I/O latencies for read, write, and erase operations. Thus, the buffering policy for flash-based databases has to consider those properties to improve the overall performance. This article introduces a new approach to buffer management for flash-based databases, called AD-LRU (Adaptive Double LRU), which focuses on improving the overall runtime efficiency by reducing the number of write/erase operations and by retaining a high buffer hit ratio. We conduct trace-driven experiments both in a simulation environment and in a real DBMS, using a real OLTP trace and four kinds of synthetic traces: random, read-most, write-most, and Zipf. We make detailed comparisons between our algorithm and the best-known competitor methods. The experimental results show that AD-LRU is superior to its competitors in most cases.

1. Introduction

1.1. Problem Statement

In recent years, flash memory greatly gained acceptance in various embedded computing systems and portable devices such as PDAs (personal digital assistants), HPCs (handheld PCs), PMPs (portable multimedia players), and mobile phones because of low cost, volumetric capacity, shock resistance, low-power consumption, and non-volatile properties [1,2,3]. Among the two types of flash memory, NAND and NOR, the NAND flash memory is more often used for mass-storage devices.[1] In the following text, we just use the term *flash memory* or *flash* to indicate the *NAND flash memory*.

However, flash memory has many properties differing from magnetic disk, e.g. write-once, block erasure, asymmetric read/write speed and limited block erase count. Flash memory usually consists of many blocks and each block contains a fixed set of pages [5]. Read/write operations are performed on page granularity, whereas erase operations use block granularity. Write/erase operations are relatively slow compared to read operations. Typically, write operations are about ten times slower than read operations, and erase operations are about ten times slower than write operations [6]. Data in a page cannot be updated in-place, i.e., when some data in a page has to be modified, the entire page must be written into a free page slot and the old page content has to be invalidated. Hence, flash always requires *out-of-place updates*. Furthermore, updating a page will cause costly erase operations performed by some garbage collection policy [7], in case that no enough free pages exist in flash memory. Hence, increasing the number of writes will accompany even more erase operations, as shown in previous experimental studies [6].

* Corresponding author at: Department of Computer Science, University of Kaiserslautern, D-67663 Kaiserslautern, Germany.
E-mail address: haerder@informatik.uni-kl.de (T. Härder).

[1] In some special use cases, e. g., in PMPs, it is even desirable to store program code on NAND flash [4], but, in this article, we consider the more common use of flash as a data storage device.

0169-023X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2011.09.007

Since flash memory has become a serious disk alternative, traditional DBMSs should support flash storage devices and provide efficient techniques to cope with flash I/O properties. Among those techniques, DBMS buffering has first received much attention from the research community because of its effectiveness in reducing I/O latencies and thus improving the overall DBMS performance. Traditional (magnetic-disk-based) buffering algorithms do not consider the differing I/O latencies of flash memory, so their straight adoption would result in poor buffering performance and would demote the development of flash-based DBMSs [8]. The use of flash memory requires new buffer replacement policies considering not only buffer hit ratios but also replacement costs incurring when a dirty page has to be propagated to flash memory to make room for a requested page currently not in the buffer. As a consequence, a replacement policy should minimize the number of write and erase operations on flash memory and, at the same time, avoid to worsen the hit ratio which otherwise would lead to additional read operations.

The above flash challenges of DBMS buffering are not met by traditional buffer replacement algorithms. Most of them focus on hit-ratio improvement alone, but not on write costs caused by the replacement process. Recently, LRU-WSR [6] and CFLRU [8] were proposed as the new buffering algorithms for flash-based DBMSs. These algorithms favor to first evict clean pages from the buffer so that the number of writes incurring for replacements can be reduced. Of course, this is a very important idea concerning flash DBMS buffering, and we will also partially observe this policy but with a critical revision of the replacement process. However, CFLRU and LRU-WSR do not exploit the frequency of page references, which will result in an increase of both write count and runtime. Furthermore, since both algorithms exploit the LRU concept and use only a single LRU queue, both are not scan-resistant [9], i.e., the buffer will be totally polluted with sequentially referenced pages when some kind of scan operation is performed.

1.2. Our contributions

In this article, we present an efficient buffer replacement policy for flash-based DBMSs, called AD-LRU (Adaptive Double LRU), which focuses on reducing the number of write/erase operations as well as maintaining a high buffer hit ratio. The specific contributions of our article can be summarized as follows:

(1) We present the novel AD-LRU algorithm for the buffer management of flash-based DBMSs (see Section 3), which not only considers the frequency and recency of page references but also takes into account the imbalance of read and write costs of flash memory when replacing pages. Moreover, AD-LRU is self-tuning to respond to changes in reference patterns, as frequency and recency of page references may fluctuate in varying workloads.
(2) We run experiments both in a flash simulation environment and in a real DBMS to evaluate the efficiency of AD-LRU by using different types of workloads (see Section 4). The experimental results show that AD-LRU maintains a higher hit ratio for the Zipf workload, whereas, for the other workloads, its results are comparable to those of the three competitor algorithms. In both types of experiments, our algorithm outperforms them considering both write count and overall runtime.

1.3. A brief outline of the paper

The remainder of this article is organized as follows: In Section 2, we sketch the related work. In Section 3, we present the basic concepts of the AD-LRU approach to flash DBMS buffering. Section 4 describes the details about the experiments and the performance evaluation results. Finally, Section 5 concludes the article and outlines our future work.

2. Related work

In this section, we briefly introduce flash storage systems (see Section 2.1) and then review the replacement algorithms for magnetic-disk-based (see Section 2.2) and those for flash-based DBMSs (see Section 2.3).

2.1. Flash memory and flash storage system

Flash memory is a type of EEPROM, which was invented by Intel and Toshiba in 1980s. Unlike magnetic disks, flash memory does not support update in-place, i.e., previous data must be first erased before a write can be initiated to the same place. As another important property of flash memory, three types of operations can be executed: read, write, and erase. In contrast, magnetic disks only support read and write operations. Moreover, all granularities and latencies of read and write operations differ for both device types.

Compared to magnetic disks, flash memory has the following special properties:

(1) It has no mechanical latency, i.e., seek time and rotational delay are not present.
(2) It uses an out-of-place update mechanism, because update in-place as used for magnetic disks would be too costly.
(3) Read/write/erase operations on flash memory have different latencies. While reads are fastest, erase operations are slowest. For example, a MICRON MT29F4G08AAA flash chip needs 25 μs/page, 220 μs/page, 1.5 ms/block for the read/write/erase latencies, respectively [10].
(4) Flash memory has a limited erase count, i.e., typically 100,000 for SLC-based NAND flash memory. Read/write operations become unreliable when the erase threshold is reached.