논문 2017-12-46

NAND Flash 메모리 기반 해시 충돌 처리 기법에 관한 연구

(Study of Hash Collision Resolution Scheme for NAND Flash Memory)

박 웅 규, 김 성 철, 온 병 원, 정 호 열, 최 규 상*
(Woong-Kyu Park, Sung-Chul Kim, Byung-Won On, Ho-Youl Jung, Gyu Sang Choi)

Abstract : In this paper, we show shortcomings of separate chaining scheme by way of experiments with NAND flash memory and improve the performance with merge chaining scheme which is proposed in this paper. We explain this merge chaining scheme and explain how to improve the performance of search operation. Merge chaining scheme shows better performance at insert and search operation compare to separate chaining scheme.

Keywords : NAND flash memory, Hash table, Hash collision resolution, Data structure, Closed addressing scheme

Ⅰ. 서 론

플래시 메모리는 EEPROM (Elecrically Erasable Programmable read-Only Memory)로서 비휘발성의 특징이 있고 RAM과 같이 읽기, 쓰기가 가능하고, 강한 충격 저항, 빠른 읽기 속도 등의 장점으로 워크스테이션, 서버의 BIOS, Laptop, 휴대전화, 디지털카메라, 개인 휴대용 단말기, 임베디드 시스템 등에서 널리 사용되고 있다 [1].

하지만 제자리 갱신 (in-place update)이 불가능하여 쓰기 전 삭제 (erase-before-write)를 수행해야하고, 하드디스크와 같은 다른 저장장치에 비해 제한된 삭제 회수 등의 단점으로 플래시 메모리를 그대로 저장장치로 사용하기는 힘든 단점이 있다 [2]. 이러한 단점을 보완하고 하드디스크처럼 인터페이스하기 위해 플래시 전환 계층 (Flash Translation Layer,FTL) [3, 4, 5, 6]을 사용하거나 저널링 파일 시스템 (Journaling File System) [7, 8, 9]이 사용된다. 이를 통해 블록 내의 페이지 활용도를 최대한 높이고 삭제 회수를 줄여야 한다.

NAND 플래시 메모리는 읽기 속도가 가장 빠르고 삭제는 쓰기보다 속도가 더 느리다. 또한 삭제는 페이지들의 집합인 블록 단위로 실행된다. 제한된 삭제 회수에 도달하는 마모고장 (Wear-out)이 발생하면 해당 블록은 더 이상 사용할 수 없으므로 낸드 플래시 메모리의 사용기간 및 속도를 늘리기 위해서는 쓰기 균등화 정책 (Wear-Leveling) [10, 11, 12]이 필요하다.

본 논문은 위의 NAND 플래시 메모리의 특징에에 적합한 자료 구조에 대한 논문으로 데이터 베이스와 같이 어플리케이션에서 사용하는 인덱스 자료 구조에 대한 내용이다.

* Corresponding Auther (castchoi@ynu.ac.kr)
Received: June 3 2017, Revised: Aug. 21 2017, Accepted: Sep. 26 2017.
W.K. Park, H.Y. Jung, G.S. Choi: Yeungnam University
S.C. Kim: The IMC
B.W. On: Kunsan University
이 논문은 2017년도 정부 (미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구 (NRF-2016R1A2B4007498)이며, 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터 육성 지원사업의 연구결과로 수행되었으며 (IITP-2017-2016-0-00313), 2016년도 산업통상자원부 및 산업기술평가관리원 (KEIT) 연구비 지원에 의한 연구임 (No.10063130, 익스트림 트랜잭션 및 분산 확장성을 제공하는 대용량 비휘발성 메모리 (SCM) 기반의 차세대 인메모리 빅 데이터베이스 시스템 상용 기술 개발).

ⓒ IEMEK J. Embed. Sys. Appl. 2017 Dec. 12(6) 413-424
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2017.12.6.413

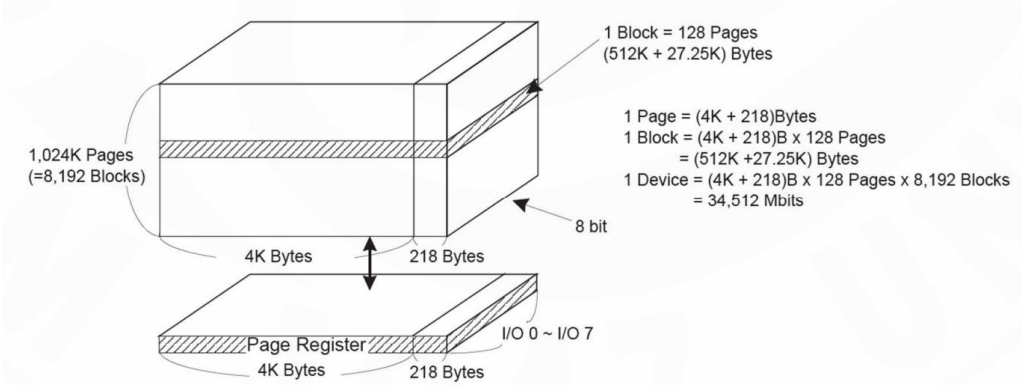

그림 1. NAND 플래시 메모리 구성
Fig. 1 Composition of NAND flash memory

인덱스 자료 구조는 모든 레코드의 키를 모아 차례로 정렬하는 방법이다. 하지만 레코드의 숫자가 증가함에 따라 많은 양의 데이터를 차례로 모두 관리하기에는 비효율적인 단점이 있어 현재는 트리 인덱스 자료 구조[13, 14, 15]나 해시 인덱스 자료 구조[16, 17, 18]를 많이 사용한다.

해시 인덱스 자료 구조는 레코드의 키를 해시 함수를 통해 얻게 되고 자료가 증가함에 따라 충돌(Collision)이 일어날 가능성이 있다. 이러한 충돌을 처리하는 방법으로는 개방 주소 기법 (Open addressing scheme)과 폐쇄 주소 기법 (Closed addressing scheme)의 두가지 카테고리의 방법이 있다.

개방 주소 기법을 플래시 메모리에 그대로 적용시킬 경우 생성되는 버켓의 개수가 많아져 버퍼의 적중률을 떨어뜨려 읽기가 쓰기보다 빠른 플래시 메모리에서도 부담 (Overhead)으로 작용해 높은 성능을 기대하기 어렵다. 이러한 단점을 보완하기 위해 NAND 플래시 메모리에 효율적인 해시 충돌 처리 기법인 병합 연쇄 기법 (Merge chaining scheme)을 제안하고 추가 제안 사항을 실험을 통해 제안하고자 한다.

본 논문의 구성은 Ⅱ장에서 NAND 플래시 메모리의 특징과 단점 보완을 위한 기법과 기존의 인덱스 자료 구조에 대해 소개하고 Ⅲ장에서는 NAND 플래시 메모리 기반의 어플리케이션에서 기존의 인덱스 자료 구조에 대한 문제의 원인과 그 실험 그리고 제안하는 충돌 처리 기법을 활용한 자료 구조에 대해 실험을 통해 단점을 보완함을 증명하고 Ⅳ장에서는 제안하는 기법에 대한 결론을 맺는다.

표 1. NAND 플래시 메모리 속도
Table 1. Operation time for NAND flash memory

	Sequencial Read	Random Read	Write	Block Erase
Access time	50ns	60μs	800μs	1.5ms

Ⅱ. 관련 연구

1. NAND 플래시 메모리

플래시 메모리는 1971년 미국 Intel 사에서 FAMOS (Floating gate Avalanche injection MOS)라고 불리는 EPROM (Erasable Programmable Read Only Memory)를 처음 발표하고 1984년 일본 Toshiba사에 의해 플래시 메모리가 발표되었다. NAND 플래시 메모리 [2]는 표 1 과 같이 무작위 접근이 순차 읽기 보다 훨씬 느리고 최소 읽기 및 쓰기의 단위는 그림 1과 같이 페이지 단위로 되어 있지만 빠른 쓰기 속도, NOR에 비교하면 느리지만 하드디스크보다 빠른 읽기 속도, 작은 셀 크기 등의 장점으로 NOR 플래시 메모리보다 NAND 플래시 메모리가 대용량 저장장치로 더 많이 쓰이고 있다.

하지만 제자리 갱신 불가, 제한된 삭제 가능 횟수, 읽기, 쓰기, 지우기의 속도 차이 등의 특징으로 NAND 플래시 메모리를 그대로 저장 장치로 활용하기에는 성능저하 및 마모 고장에 의한 수명 단축이 우려된다. 이러한 단점을 보완하기 위해 플래시 전환 계층 [3, 4, 5, 6]이나 JFFS (Journaling Flash File System) [7], YAFFS (Yet Another

Flash File System) [8], EXT4 (Ended File System 4) [9] 와 같은 저널링 파일 시스템을 사용하여 삭제 를 최대한 줄이고 쓰기 균등화 정책 [10, 11, 12]을 통해 내구성 (Endurance)을 한계까지 최대한 활용하여 수명을 늘리는 방법이 사용되고 있다.

2. 인덱스 자료 구조

인덱스 자료 구조는 레코드를 통해 저장 장치에 원하는 데이터를 빠르게 찾을 수 있는 자료 구조이다. 가장 기본적인 인덱스 자료 구조는 모든 레코드를 차례로 정렬하는 방법이다. 인덱스는 데이터 파일보다 일반적으로 크기가 작아 인덱스를 탐색하는 것이 데이터 전체를 차례로 탐색하는 것 보다 효율적이다. 하지만 레코드의 수에 따라 그 범위가 너무 넓어 많은 양의 데이터를 관리하기 위해 대부분의 데이터베이스와 파일시스템에서는 트리[13, 14, 15]나 해시 인덱스 자료 구조[16, 17, 18]를 많이 사용한다. 트리 구조 방법은 레코드 수와 저장 형태 및 위치에 따라 검색 시간이 많이 소요되고 최선과 최악의 시간 차이가 많이 발생할 수 있어 큰 데이터 셋에서는 효율적이지 못한 문제점이 있다. 해시는 비교가 아닌 계산 방법으로 키의 계수적인 성질을 이용하여 레코드를 검색하는 방법이다. 일정한 속도와 삽입 및 제거가 쉬워 다양한 분야에서 활용되고 있다.

3. 해시 인덱스 자료 구조

해시 인덱스 자료 구조는 해시테이블을 이용하고 레코드를 저장할 수 있는 공간인 슬롯, 그리고 그 슬롯들의 집합인 버켓, 그리고 그 버켓의 위치를 저장하고 있는 디렉토리로 이루어져 있다. 키의 값을 해시 함수를 통해 난수 (Random number)로 바꾸어 레코드가 저장된 주소를 계산을 통해 산출하여 데이터를 찾는 방식이다. 해시 함수를 사용하게 되면 서로 다른 레코드가 같은 해시 값을 갖는 충돌 (Collision)이 발생하게 되는데, 이를 해결하기 위해 다양한 충돌 처리 기법 (Collision resolution scheme)이 제안되었다.

2.3.1 개방 주소 형

개방 주소 형에는 선형 검색 기법 (Linear probing scheme) [19], 2차 검색기법 (Quadratic probing scheme) [20], 무작위 검색 기법 (Random probing scheme) [21]이 있다. 개방 주소형은 충돌이 일어나는 레코드들을 처리할 때 충돌이 발생한 레코드들에 대한 연쇄가 없다는 특징이 있으며 본 논문에서는 자세히 다루지 않는다.

선형 검색 기법은 충돌이 발생한 위치에서 가장 가까운 곳의 빈 슬롯에 충돌이 발생한 레코드를 저장하는 방법이다. 식 1과 같이 가장 가까운 후속 슬롯 주소 $H_i(K)$를 확인하고 빈 슬롯이 있을 때까지 해당 슬롯에 저장한다.

$$H_i(K) = H_0(K) + i \ (i = 1,2,.....) \quad (1)$$

선형 검색 기법은 간단하지만, 레코드가 한곳으로 모이는 군집 현상 (Clustering)이 발생하기 쉬우며, 계산된 슬롯 주소에서 직접 검색하지 못하고 여러 슬롯을 읽어야 할 가능성이 많다. 또한 레코드의 삭제로 빈 슬롯이 생겼을 때 군집현상의 끝인지 아니면 하나의 빈 슬롯인지 확인할 수 없는 단점도 있다.

2차 검색 기법과 무작위 검색 기법에 대해서는 본 논문에서는 활용하지 않으므로 심도 깊게 다루지 않는다.

2.3.2 폐쇄 주소 형

폐쇄 주소형은 충돌이 발생하면 같은 해시를 가지고 있는 레코드들을 연결 리스트 기법으로 연결하는 체인 (Chain) 방식이다. 개방 주소 형과는 달리 충돌이 일어난 부분에 대해서 연쇄로 묶어 두는 방법을 사용하며 이에 따라 좀더 복잡한 처리 과정이 있을 수 있으나 레코드 검색에 충돌이 일어난 레코드만 검색하므로 검색이 더 빠른 효과를 볼 수 있다. 폐쇄 주소 형에는 분리 체인 기법 (Separate chaining scheme) [22], 독립 연쇄 기법 (Independent Chaining scheme) [23], 합병 연쇄 기법 (Coalesced Chaining) [24]이 있다.

분리 연쇄 기법은 각 슬롯을 헤드 노드 (Head node)로 취급하고 레코드의 키를 저장하는 부분과 노드간의 연결 역할을 하는 인덱스 부분을 하나의 슬롯으로 구성하고 헤드 노드의 버켓당 하나의 연결 리스트를 만든다. 각 슬롯은 저장장치에 독립적으로 위치하게 된다.

그림 2와 같이 8에 대한 나머지 연산을 해시함수로 사용하는 해시 인덱스 자료 구조에서는 0과 16의 키에 대한 해시 값이 같아 충돌이 일어나면 각 레코드를 연결 리스트로 연결해 충돌을 처리하는 기법이다.

독립 연쇄 기법은 저장 공간을 두 개의 영역으로

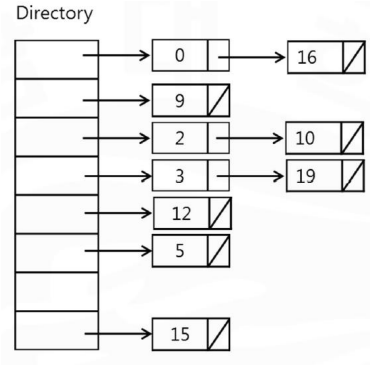

그림 2. 분리 연쇄 기법
Fig. 2 Structure of separate chaining scheme

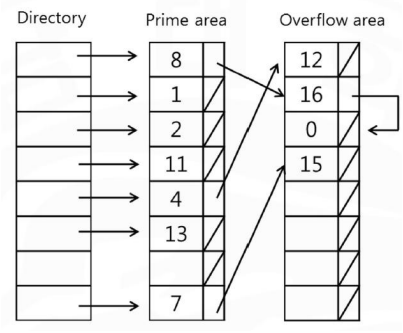

그림 3. 독립 연쇄 기법
Fig. 3 Structure of Independent chaining scheme

구분하여 충돌이 발생하지 않은 레코드들로 구성된 기본 구역 (Prime area)과 충돌이 발생한 레코드들을 저장하는 충돌 처리 구역으로 (Overflow area)로 구성된다. 즉 Prime Area에는 소켓에 정해진 해시 값의 레코드만 존재하며 해당 해시 값의 레코드가 없으면 해당 공간은 비어있게 된다. 먼저 기본 구역의 슬롯 주소에 슬롯이 비어있으면 해당 해시에 대한 레코드가 없는 것으로 처리한다. 충돌 처리 구역에서는 충돌이 일어났을 때 저장되는 영역이며 여러 번 충돌이 일어났을 때에도 충돌 처리 구역의 빈 소켓이 있을 경우 이를 활용 한다. 그림 3과 같이 8에 대한 나머지 연산을 통해 8, 0, 16이 충돌하여 가장 처음에 충돌이 없었던 8의 인덱스를 갖는 레코드는 기본구역에 0과 16은 충돌 구역에 저장되었으며 기본구역의 순서와 상관없이 저장되고 충돌 구역에서 또다시 충돌이 일어날 경우 다음 빈 소켓의 위치에 레코드를 저장한다.

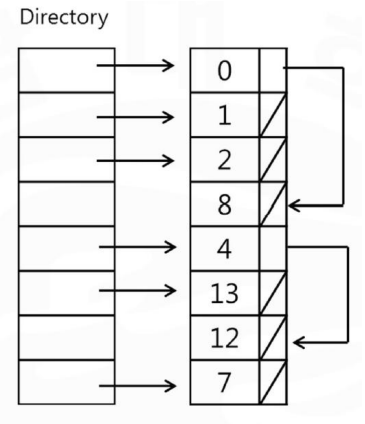

그림 4. 합병 연쇄 기법
Fig. 4 Structure of Coalesced chaining scheme

합병 연쇄 기법은 충돌이 발생 하였을 때 별도의 기억 장소를 이용하지 않고 해시 테이블 내의 빈 슬롯을 찾아 해결하는 방법이고, 직접 연쇄 기법이라고도 한다. 그림 4와 같이 0과 8, 4와 12는 8의 나머지에 대한 해시 함수에서는 같은 해시 값을 갖고 충돌이 발생한 경우 충돌한 레코드에 대한 위치에 대한 정보를 저장한다. 합병 연쇄 기법은 별도의 기억 공간이 필요 없는 장점이 있지만, 충돌 발생 시 빈 슬롯을 찾는 과정, 인덱스 부분을 조정하는 알고리즘이 복잡하다. 충돌이 발생한 레코드가 정확한 위치가 아닌 다른 소켓에 저장되므로 검색에 어려움이 발생하고, 늦게 삽입 되는 코드를 자신의 소켓 주소에 저장하기 위해 조정이 자주 일어난다.

합병 연쇄 기법은 개방 주소형과는 달리 검색할 때는 원하는 키를 찾을 때까지 연쇄로 연결된 인덱스를 따라 다음 레코드를 읽고 만약 다음 인덱스의 값이 NULL 일 때 검색을 종료한다.

Ⅲ. 제안하는 해시 충돌 처리 기법

NAND 플래시 메모리 기반의 해시 충돌 처리 기법인 분리 연쇄 기법의 문제점을 실험을 통해 제기하고 병합 연쇄 기법이라는 NAND 플래시 메모리에서 효율적인 해시 충돌 처리 기법을 제안하고 그 장단점을 실험을 통해 설명한 후 검색이 느린 단점을 보완하기 위한 다양한 방법을 실험을 통해 제안하고 증명하고자 한다.

표 2. 실험환경
Table 2. Experiment environment

CPU	Intel Core 2 Duo 3.0GHz
RAM	2G Bytes
Storage interface	Serial ATA-II
SSD	Samsung SSD 840
OS	Linux Fedora 18
File System	EXT4

그림 5. 레코드 구성
Fig. 5 Composition of a record

1. 분리 연쇄 기법의 문제점

NAND 플래시 메모리의 최소 읽기 단위는 페이지 (Page)로 되어 있다. 페이지는 약 4 Kbytes, 8 Kbytes와 같이 word (4bytes)에 비해 매우 큰 단위이며 수 byte 단위의 작은 하나의 레코드를 읽기 위해 페이지 단위를 계속 읽고 쓰는 것은 손해가 매우 크다. 가장 이상적인 해시테이블은 레코드의 수가 맞는 버켓과 슬롯으로 구성된 것이지만 레코드가 계속 입력 또는 삭제되며 변화하는 인덱스 자료 구조의 특성상 버켓의 크기를 지정하기 어려운 문제가 있다. 변하는 레코드 수에 가장 좋은 결과를 보여주는 것은 버켓의 크기를 NAND 플래시 메모리의 크기와 같게 하는 것이다. 하지만 이 방법은 충돌이 많이 발생하지 않는 해시테이블에서는 실제 필요한 크기보다 훨씬 더 많은 저장 공간을 사용한다는 단점이 있다. 가장 좋은 해시 함수는 충돌이 최대한 드물게 일어나고 분포가 잘 되어있는 상태이지만, 분리 연쇄 기법의 버켓 크기를 기존의 하드디스크의 섹터 단위 (512bytes)보다 훨씬 큰 플래시 메모리의 페이지 단위로 설정하는 것은 저장장치의 자원 낭비가 심해지고, 메모리 버퍼 적중률 (Hit Rate)를 떨어뜨려 오히려 전체적인 해시 테이블의 성능이 저하될 수도 있다. 그리고 충돌이 매우 많이 발생하는 해시는 다른 해시 함수를 사용하는 재 해싱을 통해 충돌이 드물게 발생하게 하는 것이 더 효과적이라고 알려져 있다.

표 2와 같은 실험 환경에서 그림 5와 같이 24 bytes의 키와 4 bytes의 인덱스로 레코드를 구성한다. 메모리 버퍼는 8M bytes로 설정하였으며, 페이지 교체 전략에 대해 언급이 되지 않는 실험의 경우 LRU (Least Recently Used) 방식을 사용하였다.

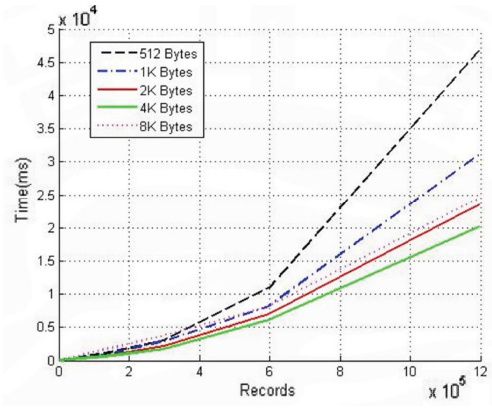

그림 6. 분리 연쇄 기법의 삽입 소요 시간
Fig. 6 Total elapsed time for insert operation in separate chaining

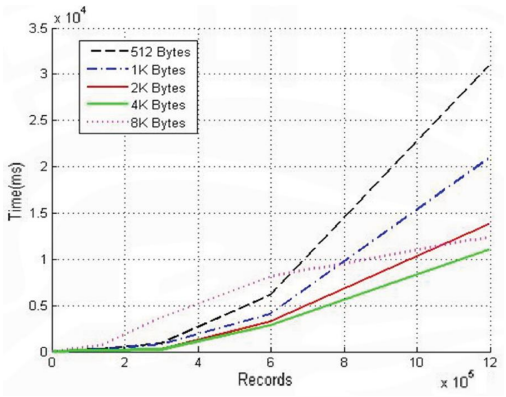

그림 7. 분리 연쇄 기법의 검색 소요 시간
Fig. 7. Total elapsed time for insert operation in separate chaining

삽입 (Insert) 에는 레코드들을 전부 삽입하는데 걸리는 시간, 검색 시간에는 삽입된 레코드들을 모두 검색하는데 걸리는 시간을 나타낸다.

아래의 그림 6, 7, 8의 실험 결과는 분리 연쇄 기법에서 충돌이 매우 많이 발생하도록 해시 함수의 범위를 줄여서 실험한 결과이다. 최저 평균 0개의 충돌과 592개의 충돌이 발생하도록 레코드의 개수를 조정하여 실험한 결과는 다음과 같다.

그림 6은 분리 체인 기법에서 레코드 개수와 버켓의 크기에 따른 삽입에 걸리는 시간을 나타내는 그래프이다. 실험에 사용된 저장장치의 최소 읽기, 쓰기의 단위인 페이지와 크기가 같은 4K에서 가장 좋은 성능을 발휘한다.

그림 7은 분리 분리 연쇄 기법에서 레코드 개수

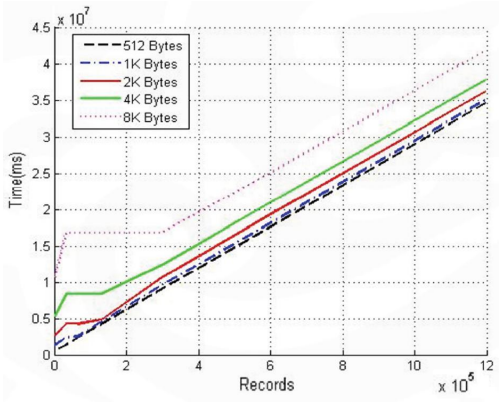

그림 8. 분리 연쇄 기법의 파일 크기
Fig. 8 Total file size of separate chaining

와 버켓의 크기에 따른 검색 시간을 나타내는 그래프이다. 그림9와 마찬가지로 4K Bytes의 크기에서 가장 좋은 성능을 발휘한다.

그림 8은 분리 연쇄 기법에서 레코드의 개수와 버켓의 크기에 따른 파일 크기에 대해 나타내는 그래프이다. 버켓의 크기를 크게 설정하면 할수록 많은 저장 공간이 필요로 하게 된다. 특히 레코드의 개수가 작고 충돌이 드물게 일어나는 상황에서는 불필요한 저장 공간을 많이 소모하게 된다.

페이지의 크기와 같은 4 Kbytes로 버켓의 크기를 설정한 분리 연쇄 기법이 기장 좋은 성능을 보이지만 충돌이 한 버켓을 채울 정도로 많은 충돌이 발생하기 전까지는 불필요한 저장 공간을 많이 소모하게 된다.

본 논문에서 실험 내용에 대한 결과는 버켓의 크기를 페이지 크기와 같게 설정한 것을 대상으로 한다.

2. 제안하는 해시 충돌 처리 기법

기존의 분리 연쇄 기법에는 두 가지 문제점이 있다. 불필요한 저장 공간이 많이 필요로 하고, 그에 따라 버퍼 적중률이 감소하여 성능이 저하되는 단점이다. 이러한 단점을 줄이기 위해 병합 체인 기법을 제안하고자 한다.

병합 체인 기법은 그림 9와 같이 한 버켓에 하나의 해시에 대한 레코드들을 저장하는 것이 아니라 여러 개의 해시에 대한 레코드들을 삽입할 수 있도록 하였다. 버켓 내 평균 하나의 해시에 하나의 슬롯을 배분하도록 하였다. 즉 버켓에 슬롯이 4개 있으면 4개의 해시가 한 버켓에 저장되고 해당

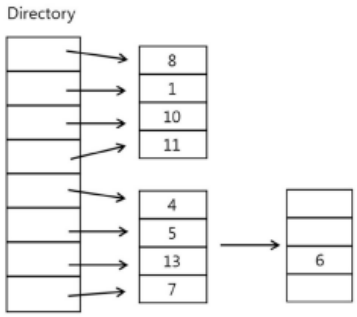

그림 9. 병합 연쇄 기법
Fig. 9 Merge chaining scheme

인덱스를 디렉토리에 저장하게 된다. 과잉 현상이 발생하면 기존의 분리 연쇄 기법과 같이 버켓을 연결 리스트로 연결하여 충돌을 처리했다. 제안하는 방법인 병합 체인 기법은 여러 해시에 대한 레코드를 갖고 있기 때문에 검색 성능을 향상시키기 위해 버켓 내의 레코드들을 선형 검색 기법을 적용하여 검색 성능을 향상 시키고자 했다. 아래의 그림 9와 같이 8,1,10,11의 키를 갖는 레코드가 해시가 모두 다르다고 하더라도 한 버켓에 저장된다. 그리고 5와 13의 키를 갖는 레코드가 해시가 같다고 가정할 때 만약 버켓에 빈 슬롯이 있으면 그 슬롯에 해당 레코드를 쓰게 된다. 쓰기 횟수를 최대한 줄이기 위해서 해당 슬롯이 해시를 갖는 레코드가 오더라도 위치를 바꾸지 않는 방법을 적용하였다.

3. 검색 성능 향상을 위한 합병 연쇄 기법 적용

병합 연쇄 기법은 충돌이 많이 발생하지 않는 해시테이블에서는 좋은 성능을 보이지만 검색의 경우 평균 3회 이상의 충돌이 발생 시 검색 속도가 기존의 분리 연쇄 기법에 비해 느려지게 된다. 그 이유는 버켓의 레코드를 검색하기 위해 선형 검색 기법을 사용했지만 해시가 다른 레코드를 검색할 가능성이 너무 크고 검색하고자 하는 레코드를 찾기 위해 해시가 다른 많은 레코드들 검색해야 한다. 그리고 여러 개의 해시를 모아서 하나의 버켓을 생성하기 때문에 과잉현상으로 연결 리스트가 너무 많이 생성되게 된다. 빈 버켓을 검색하기 위해 디렉토리의 시작 버켓부터 원하는 레코드가 저장된 버켓까지 너무 많은 버켓을 읽어야 하므로 읽기가 훨씬 빠른 NAND 플래시 메모리에서도 원하는 데이터를 찾기까지 오래 걸려 메모리 버퍼 적중률이 60%가 넘지 않을 경우 기존의 분리 연쇄 기법에

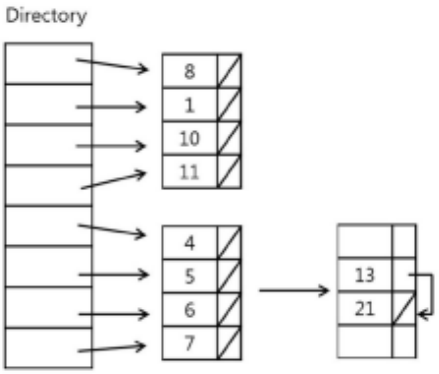

그림 10. 합병 연쇄 기법을 적용한 병합 연쇄 기법
Fig. 10 Structure of merge chaining scheme with coalesced chaining scheme for slot

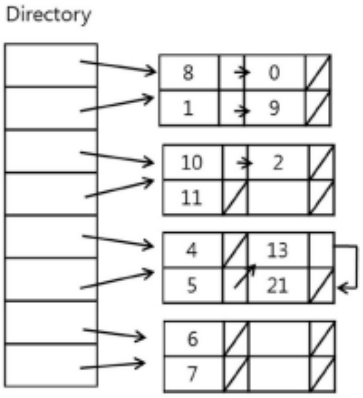

그림 11. 버켓 분할 기법을 적용한 병합 연쇄 기법
Fig. 11 Merge chaining with split

비해 나쁜 검색 성능을 보인다. 같은 이유로 결국 삽입에서도 충돌이 매우 많이 일어나게 될 경우 성능이 역전되는 현상이 발생하게 된다. 삽입에서 성능이 역전되는 현상이 훨씬 늦게 일어나는 이유는 낸드 플래시 메모리의 경우 읽기 에 비해 쓰기가 훨씬 느려서 병합 연쇄 기법의 경우 메모리 버퍼 적중률이 분리 연쇄 기법보다 훨씬 높아 쓰기 횟수를 줄여 읽기가 많이 발생하는 병합 연쇄 기법에서도 평균 충돌 횟수가 삽입에서는 레코드 수가 더 많아야 분리 연쇄 기법의 성능이 더 좋은 결과가 나타나게 된다.

버켓 내 합병 연쇄 기법을 적용하기 위해 새로운 인덱스 공간이 필요하므로 버켓 내의 슬롯을 줄여야 한다. 본 실험에서는 기존의 병합 연쇄 기법의 경우 146개의 슬롯을 사용했지만, 버켓 내의 합병 연쇄 기법을 적용하기 위해서 6개의 슬롯을 줄이는 대신 관계없는 레코드를 읽지 않아도 되므로 검색에서 약간의 성능 향상을 볼 수 있다. 그림 10과 그림 9를 비교해보면 6의 키를 가지는 레코드의 위치가 다름을 알 수 있는데 6의 키의 지정된 슬롯에 이미 다른 레코드가 있어도 그 슬롯의 해시가 정확한 위치가 아닐 경우 6번 키의 레코드로 대체하고 기존의 13번 키의 레코드는 버켓의 빈 슬롯에 저장하고 빈 슬롯이 없다면 과잉 현상의 발생이므로 새로운 버켓을 생성하여 연결 리스트로 묶는다. 그리고 6,13,21의 키에 대한 해시가 같을 경우 버켓 내에 인덱스를 통해 다음 버켓을 연결하여 해시가 다른 레코드는 읽지 않게 되어 검색 성능을 향상시킬 수 있다. 하지만 레코드 교체, 인덱스 추가가 필요함에 따라 삽입 성능이 떨어지는 단점이 있고, 43% 이하의 메모리 적중률을 보이는 환경에서는 기존의 분리 연쇄 기법보다 검색 성능이 떨어지게 된다.

4. 병합 연쇄 기법을 위한 메인 메모리 버퍼 관리 정책

검색 성능을 향상시키기 위한 다른 접근은 메모리 버퍼 관리 정책이다. 병합 연쇄 기법은 과잉 현상이 매우 빨리 일어나게 된다. 디렉토리에 직접 연결된 버켓을 읽을 가능성은 매우 높지만 연결 리스트의 가장 마지막에 있는 버켓을 읽을 가능성은 매우 낮다. 이렇게 디렉토리와 가까이 있는 버켓의 우선순위를 높여 메모리 버퍼에서 제거 (Evict)되지 않는다면 메모리 버퍼 적중률을 높여 더 빠른 검색 성능을 보일 수 있다. 하지만 여전히 과잉 현상이 매우 빠르게 일어나는 단점으로 충돌이 많이 발생하는 큰 데이터 셋에서는 분리 연쇄 기법이 더 나은 성능을 나타낸다.

5. 검색 성능 향상을 위한 버켓 분할 기법

병합 연쇄 기법의 특징을 과잉 현상이 매우 빠르게 나타나게 되는 이유는 여러 개의 해시가 하나의 버켓을 사용하기 때문에 기존의 분리 연쇄 기법보다 저장 공간의 활용도는 뛰어 나지만, 충돌이 많이 발생하는 큰 데이터 셋에서는 성능 저하의 가능성이 매우 크다. 이러한 단점을 보완하기 위해 버켓 분할 기법을 적용한 병합 연쇄 기법을 제안한다. 충돌이 많이 발생하게 되면 하나의 버켓을 공유하는 범위를 줄이도록 버켓의 레코드들을 나누어 레코드

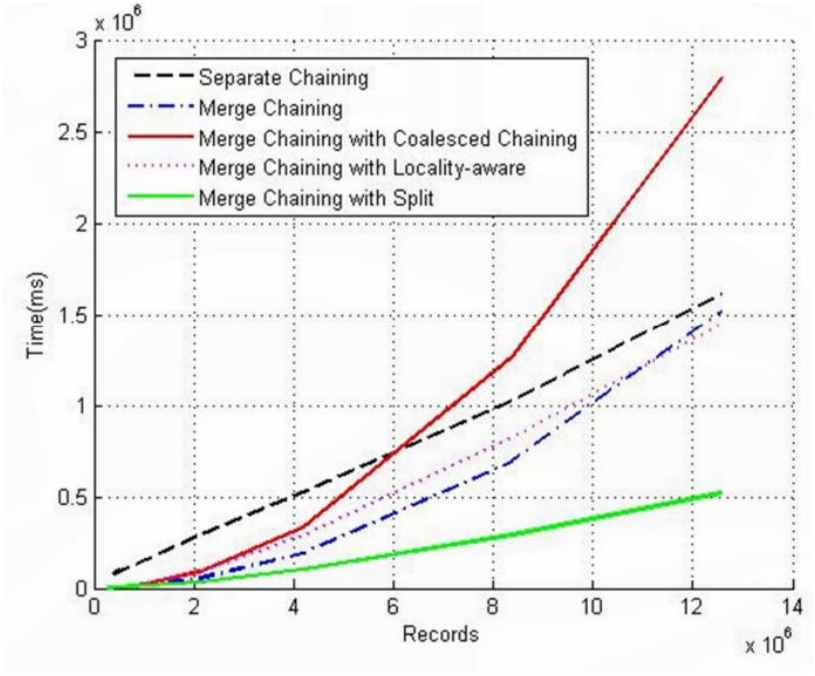

그림 12. 제안 기법의 삽입 소요 시간
Fig. 12 Total elapsed time of insert operation in proposed scheme

들을 분산 시켜 과잉 현상을 막고 디렉토리의 내용을 새로이 바꾸는 방법을 사용한다. 해시 충돌이 계속 발생하여 분할이 계속 이루어져 하나의 버켓에 대해 하나의 해시에 대한 내용만 있게 된다면 그 형태는 기존의 분리 연쇄 기법과 거의 흡사한 형태의 해시 테이블이 된다.

충돌이 매우 많이 발생할 경우 재 해싱과 디렉토리의 크기 조정으로 해결하는 것이 가장 좋은 방법이지만, 그렇지 못하여 과잉 현상이 발생하게 되면 검색 성능을 좋게 유지 하고 싶을 경우 검색 성능 유지를 위해 여러 개의 기존 버켓이 하나의 과잉현상이 나타난 버켓을 가리키도록 한다. 하지만 이 과정에서 과잉 현상에 분할이 일어나게 될 경우 범위 내 모든 버켓을 업데이트해야하므로 삽입에서의 성능저하가 일어나게 된다. 또 다른 방법으로는 분리 연쇄 기법의 형태를 그대로 유지하는 방법이다 .그림 11과 같이 과잉 현상이 일어난 후에는 완벽한 난수로 동시에 버켓을 생성하지 않는 이상 동시에 커다란 불필요한 저장 공간이 소모되지는 않는다.

그림 11은 그림 10에 레코드가 추가되어 더욱 확장된 경우의 분할 기법에 대한 그림이다. 0과 9의 레코드가 추가 되면서 분할이 발생하였고 2의 레코드가 새로이 추가 되었다 기존의 합병 연쇄 기법과 같이 여전히 해시에 맞지 않는 슬롯의 위치에 있더라도 빈 슬롯이 있으면 버켓 내에 인덱스를 추가하여 저장 공간의 사용을 줄일 수 있다.

그림 12, 13 그리고 14는 기존의 분리 연쇄 기법, 병합 연쇄 기법, 합병 연쇄 기법을 적용한 병합 연쇄 기법, 메모리 버퍼 관리 정책을 적용한 병합 연쇄 기법 그리고 버켓 분할 기법을 적용한 병합 연쇄 기법의 성능에 대해 나타내는 그래프이다. 그림 12는 제안 기법의 삽입 성능, 그림 13은 제안 기법의 검색 성능 그리고 그림 14는 제안 기법의 파일 크기에 대해 나타내고 있는 그래프다. 버켓 분할 기법을 적용한 병합 연쇄 기법이 최적의 파일 크기는 아니지만 가장 좋은 삽입 및 검색 성능을 보인다.

Ⅳ. 결 론

본 논문에서는 NAND 플래시 메모리에 효과적인 해시 충돌 처리 기법을 포함한 해시 인덱스 자료구조를 제안 하고 그 충돌 처리 기법에 대해 실험을

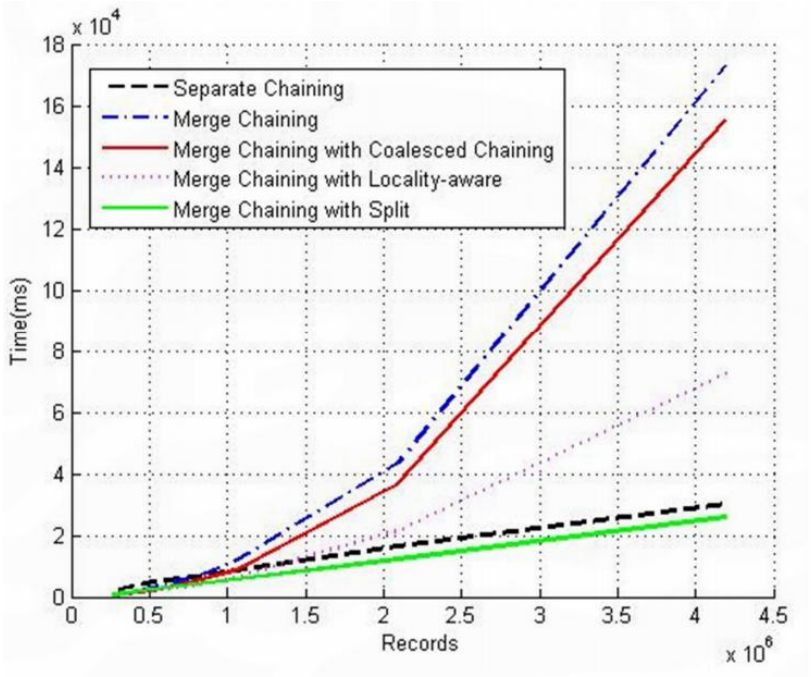

그림 13. 제안 기법의 검색 소요 시간
Fig. 13 Total elapsed time of search operation in proposed schemes

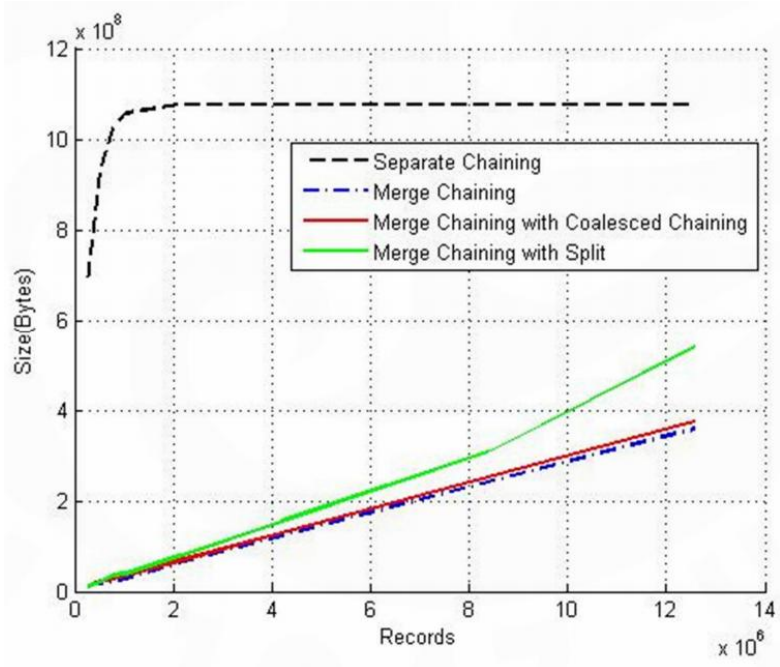

그림 14. 제안 기법의 파일 크기
Fig. 14 Total file size of proposed schemes

통해 성능을 높일 수 있는 방안에 대해 연구 하였다. 병합 연쇄 기법에 합병 레코드 연쇄 기법, 지역성에 따른 페이지 교체 알고리즘, 버켓 분할 기법을 적용하여 기존의 분리 연쇄 기법 보다 적은 저장 공간 사용, 빠른 삽입, 빠른 검색이 가능함을 보였다. 일반적인 변화하는 상황에서의 낸드 플래시 메모리를 적용한 시스템의 어플리케이션에서의 해시 인덱스 자료 구조에서는 버켓 분할기법을 적용한 병합 연쇄 기법이 가장 효과적인 성능을 발휘 할 수 있다.

References

[1] E. Gal, S. Toledo, "Algorithms and Data Structures for Flash Memories," ACM Computing Surveys, Vol. 37, No. 2, pp. 138-163, 2005.

[2] Samsung Electronics Company, K9PDG08U5D datasheet 2G x 8Bit NAND Flash Memory.

[3] S.Y. Kim, S.I. Jung, "A log-based Flash Translation Layer for Large NAND Flash Memory," Proceedings of International Conference on Advanced Communication Technology, Vol. 3, pp. 1641-1644, 2006.

[4] S.W. Lee, D.J. Park, T.S. Chung, D.H. Lee, S. Park, H.J. Song, "A log Buffer-based Flash Translation Layer Using Fully-associative Sector Translation," ACM Transactions on Embedded Computing Systems, Vol. 6, No. 3, 2007.

[5] J. Kim, J.M. Kim, S.H. Noh, S.L. Min, Y. Cho, "A Space-efficient Flash Translation Layer for Compact Flash Systems," IEEE Transactions on Consumer Electronics, Vol. 48, No. 2, pp. 366-375, 2002.

[6] J.U. Kang, H. Jo, J.S. Kim, J. Lee, "A Superblock-based Flash Translation Layer for NAND Flash Memory," Proceedings of ACM & IEEE International Conference on Embedded software, pp. 161-170, 2006.

[7] D. Woodhouse, "JFFS: The Journaling Flash File System," Ottawa Linux Symposium, 2001.

[8] A. One, "YAFFS: Yet Another Flash File System," 2002.

[9] A Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, L. Vivier, "The new ext4 File System: Current Status and Future Plans," Proceedings of Linux Symposium, Vol. 2, pp. 21-33, 2007.

[10] Y.H. Chang, J.W. Hsieh, T.W. Kuo, "Endurance Enhancement of Flash-memory Storage Systems: an Efficient Static Wear Leveling Design," Proceedings of ACM Design Automation Conference, pp. 212-217, 2007.

[11] L.P. Chang, "On Efficient Wear Leveling for Large-scale Flash-memory Storage Systems," Proceedings of ACM symposium on Applied computing, pp. 1126-1130, 2007.

[12] Y. Li, B. He, R.J. Yang, Q. Luo, K. Yi, "Tree Indexing on Solid State Drives," Proceedings of VLDB Endowment, Vol. 3, No. 1-2, pp. 1195-1206. 2010.

[13] P. O'Neil, E. Cheng, D. Hawlick, E. O'Neil, "The log-structured Merge-tree (LSM-tree)," Springer Acta Informatica, Vol. 33, No. 4, pp. 351-385, 1996.

[14] C.H. Wu, L.P. Chang, T.W. Kuo, "An Efficient B-tree Layer for Flash-memory Storage Systems," Springer Real-Time and Embedded Computing Systems and Applications, pp. 409-430, 2004.

[15] E. Malalla, "Two-way Hashing With Separate Chaining and Linear Probing," McGill University, 2004.

[16] X. Li, Z. Da, X. Meng, "A new Dynamic Hash Index for Flash-based Storage," Proceedings of International Conference on Web-Age Information Management, pp. 93-98, 2008.

[17] C.W. Yang, K.Y. Lee, M.H. Kim, Y.J. Lee, "An Efficient Dynamic Hash Index Structure for NAND Flash Memory," IEICE Transactions on fundamentals of electronics, communications and computer sciences, Vol. 92, No.7, pp. 1716-1719, 2009

[18] S.H. Lim, C. Lee, K.H. Park, "Hashing Directory Scheme for NAND Flash File System," Proceedings of International Conference on Advanced Communication Technology, Vol. 1, pp. 273-276, 2007.

[19] B. Pittel, “Linear Probing: the Probable Largest Search Time Grows Logarithmically With the Number of Records,” Journal of algorithms, Vol. 8, No. 2, pp. 236-249, 1987.
[20] A. Fiat, M Naor, “Implicit O(1) Probe Search,” Proceedings of ACM symposium on Theory of computing, pp. 336-344, 1989.
[21] M. V. Ramakrishna, “Analysis of Random Probing Hashing,” Information Processing Letters, Vol. 31, No. 2, pp. 83-90, 1989.
[22] E. Malalla, “Two-way Hashing With Separate Chaining and Linear Probing,” McGill University, 2004.
[23] G.E. Blelloch, D. Golovin, “Strongly History-independent Hashing With Applications,” Proceedings of IEEE Symposium on Foundations of Computer Science, pp. 272-282, 2007.
[24] J.S. Vitter, “Analysis of the Search Performance of Coalesced Hashing,” Journal of the ACM, Vol. 30, No. 2, pp. 231-258, 1983.

Woong-Kyu Park (박 웅 규)

He received M.S. degree in department of information and communication engineering from Yeungnam University, Korea, in 2014. He is a researcher in Wi-Media Regional Innovation Center, Yeungnam university. His research interests Data structure, Non-volatile Memory, intelligent vehicle and Image processing.
Email: pwkyu@ynu.ac.kr

Sung-Chul Kim (김 성 철)

He received B.S. degree in department of information and communication engineering from Yeungnam University, Korea, in 2012. He received M.S. degree in department of information and communication engineering from Yeungnam University, Korea, in 2014. His research Interests Non-volatile memory, Data base.
Email: borame30@ynu.ac.kr

Byung-Won On (온 병 원)

He received Ph.D. degree in department of computer engineering from Pennsylvania State University, in 2007. Currently, he is professor in the department of statistics computer science in Kunsan National University, Korea. His research Interests Data mining, Data base, information retrieval, Big data
Email : bwon@kunsan.ac.kr

Ho-Youl Jung (정 호 열)

He is received the Ph.D. degree in Electronics Engineering from the Institut National des Sciences Appliquées de Lyon (INSA de Lyon), France, in 1988. He is currently a Professor in the Department of Information and Communications Engineering, Yeungnam University, Korea. Both teaching and research interests include digital signal/image processing, intelligent vehicles, and nonlinear systems.
Email: hoyoul@yu.ac.kr

Gyu Sang Choi (최 규 상)

He received Ph.D. degree in department of computer engineering from Pennsylvania State University, in 2005. Currently, he is a professor in the department of information and communication engineering from Yeungnam University, Korea. His research Interests Non-volatile memory, Operating system, Storage system, intelligent vehicles, and nonlinear systems.

Email: castchoi@ynu.ac.kr