플래시 메모리 환경에서 클러스터링 방법과 비 클러스터링 방법의 성능 분석

(Performance Analysis of Clustering and Non-clustering Methods in Flash Memory Environment)

배 덕 호 † 장 지 웅 ††
(Duck-Ho Bae) (Ji-Woong Chang)
김 상 욱 †††
(Sang-Wook Kim)

요 약 플래시 메모리는 기존 저장 매체와는 달리 읽기 연산에 비해 쓰기 연산의 수행비용이 매우 크고, 저장된 데이타에 대한 직접 갱신이 불가능한 고유의 특성이 있다. 본 논문에서는 플래시 메모리 환경이 클러스터링 방법과 비 클러스터링 방법에 미치는 영향을 분석한다. 이를 통해 디스크 환경과는 달리 플래시 메모리 환경에서는 비 클러스터링 방법이 더 적합하다는 것을 보인다. 또한, 플래시 메모리 환경에서 비 클러스터링 방법이 가진 문제점을 지적하고, 이를 기반으로 플래시 메모리 환경에서 레코드 관리 방법의 설계 시 고려해야 할 사항들을 제안한다.

· 본 연구는 2007년도 정부(과학기술부)의 재원으로 한국과학재단(R01-2007-000-11773-0)과 2007년도 정부재단(교육인적자원부 학술연구조성사업비)으로 학술진흥재단(KRF-2007-314-D00221) 및 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업(IITA-2008-C1090-0801-0040)의 연구비 지원을 받았습니다.
· 이 논문은 제34회 추계학술대회에서 '플래시 메모리 환경에서 클러스터링 방법과 논 클러스터링 방법의 성능 분석'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 한양대학교 전자컴퓨터통신 연구원
smith@zion.hanyang.ac.kr
†† 정 회 원 : 한국산업기술대학교 게임공학과 교수
jwchang@kpu.ac.kr
††† 종신회원 : 한양대학교 전자컴퓨터통신 교수
wook@hanyang.ac.kr

논문접수 : 2008년 1월 3일
심사완료 : 2008년 5월 28일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제14권 제6호(2008.8)

키워드 : 플래시 메모리 DBMS, 클러스터링 방법, 비 클러스터링 방법, 빈 공간 리스트

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 of flash memory affect the performance of clustering and non-clustering in record management, and shows that non-clustering is more suitable in flash memory environment, which does not hold in disk environment. Also, we discuss the problems of the existing non-clustering method, and identify considerable designing factors of record management method in flash memory environment.

Key words : Flash memory DBMS, Clustering, Non-clustering, Free space list

1. 서 론

플래시 메모리는 기존의 저장 매체들과는 다른 플래시 메모리만의 고유한 특성을 가진다. 첫째, 플래시 메모리는 쓰기 연산의 수행 속도가 읽기 연산에 비해 매우 느리다[1,2]. 둘째, 플래시 메모리에서는 오버헤드가 매우 큰 소거 연산(erase operation)이 존재한다. 소거 연산은 잦은 쓰기 연산, 특히 기존의 페이지를 갱신하는 쓰기 연산에 의해 발생한다[3]. 이렇듯 플래시 메모리 환경에서 쓰기 연산을 감소시키는 것이 매우 중요하다.

최근 플래시 메모리의 용량이 증가함에 따라 하드 디스크를 대체하는 경향이 많아지고 있다. 이에 따라 플래시 메모리에 직접 데이타를 저장, 관리하는 연구가 필요하게 되었다. 그러나 지금까지는 플래시 메모리에 대용량의 데이타를 저장, 관리하는 방법에 대한 연구가 거의 이루어지지 않았다. 대부분의 경우 디스크 기반 DBMS에서 고안된 방법을 그대로 플래시 메모리에 적용하는 수준이었다[4]. 그러나 이는 플래시 메모리의 특성을 잘 반영하지 못하여 DBMS의 성능 저하의 원인이 되었다. 그러므로 플래시 메모리의 특성을 고려한 레코드 관리 방법, 인덱스 방법, 백업 기법 등에 대한 연구가 필요하다.

특히 레코드 관리 방법은 레코드의 삽입, 삭제, 배치 등을 결정하는 자료구조로서, DBMS의 성능에 큰 영향을 미친다. 디스크 기반 DBMS의 대표적인 레코드 관리 방법에는 클러스터링 방법(clustering method)과 비 클러스터링 방법(non-clustering method)이 있으며, 범위 질의의 성능이 우수한 클러스터링 방법이 주로 사용되었다.

그러나 클러스터링 방법은 삽입되는 레코드의 키 값에 따라 저장 위치가 결정되므로, 레코드 삽입 시 쓰기 연산을 빈번히 수행하게 된다. 이는 쓰기 연산의 수행 속도가 매우 느린 플래시 메모리 환경에서는 심각한 문

제를 발생시킨다. 반면, 읽기 연산의 빠른 수행 속도로 인해 비 클러스터링 방법의 느린 범위 질의 성능은 보완되며, 버퍼의 효과로 인해 쓰기 연산과 소거 연산의 수가 줄어들게 되어 비 클러스터링 방법의 성능이 클러스터링 방법에 비해 우수해질 것으로 예상할 수 있다.

레코드 관리 방법의 성능에 큰 영향을 미치는 또 다른 중요한 요소에는 빈 공간 리스트(free space list) 관리가 있다. 디스크 기반 DBMS에서는 관리의 편의를 위해 리스트 관리에 필요한 포인터들을 페이지 내에 저장한다[5]. 이로 인해 리스트 변경은 쓰기 연산을 유발하게 된다. 빈 공간 리스트 또한 플래시 메모리 환경에서는 심각한 문제를 발생시킨다.

그러나 비 클러스터링 방법 역시 플래시 메모리의 특성을 고려한 것은 아니기 때문에 여러 가지 문제점이 발생한다. 본 논문에서는 이러한 문제점들을 분석하여 원인을 규명하고, 플래시 메모리 환경에서 레코드 관리 방법을 설계할 때 고려하여야 할 사항들에 대하여 제안한다.

본 논문의 구성은 다음과 같다. 제2절에서는 플래시 메모리의 특성을 분석하고, 레코드 관리 방법에 미치는 영향을 논의한다. 제3절에서는 기존의 레코드 관리 방법을 고찰하여 플래시 메모리 환경에서 새롭게 부각되는 문제점들을 분석한다. 제4절에서는 성능 평가를 통하여 비 클러스터링 방법의 우수성을 규명한다. 제5절에서는 추후 레코드 관리 방법의 설계 시 고려되어야 할 사항들에 대하여 제안한다. 마지막으로 제6절에서 결론을 맺는다.

2. 플래시 메모리의 특성과 레코드 관리 방법

일반적으로 플래시 메모리에서 읽기와 쓰기 연산은 512바이트 크기의 페이지 단위로 이루어진다. 그리고 소거 연산은 연속된 여러 페이지로 구성된 16K바이트 블록 단위로 이루어진다[6]. 표 1은 플래시 메모리, 주기억장치(DRAM), 그리고 하드 디스크에서 읽기, 쓰기, 소거 연산의 수행 시간을 나타낸 것이다. 플래시 메모리의 쓰기 연산은 읽기 연산에 비해 약 13배의 시간이 소요되고, 소거 연산은 읽기 연산에 비해 130배의 시간이 소요되는 것을 알 수 있다[7].

하드 디스크는 읽기 연산과 쓰기 연산의 속도 차이가 없으므로 레코드 관리 방법을 설계할 때에 이러한 사항이 전혀 고려되지 않았다. 그러나 플래시 메모리에서는 쓰기 연산을 최소화하려는 노력이 필요하다.

플래시 메모리에서는 저장된 데이타를 직접 갱신하는 것이 불가능하다. 그러므로 데이타를 갱신하기 위해서는 새로운 데이타를 플래시 메모리의 다른 영역에 기록하고, 기존의 데이타를 무효화하는 방법을 사용한다[4]. 갱신 연산을 수행하면 기존의 페이지는 무효 페이지(invalid page)가 되며, 무효 페이지를 다시 사용하기 위해서는 먼저 소거 연산을 수행하여야 한다. 이렇듯, 갱신 연산을 최소화하여 무효 페이지를 줄이면, 소거 연산을 줄일 수 있다.

표 1 여러 가지 저장 장치에서 각 연산의 수행 시간

	읽기 연산	쓰기 연산	소거 연산
NAND Flash memory	15㎲/512B	200㎲/512B	2ms/16KB
DRAM	100ns/1B	100ns/1B	-
Hard disk	12.4ms/512B	12.4ms/512B	-

3. 플래시 메모리 환경에서 기존의 레코드 관리 방법의 고찰

3.1 빈 공간이 존재하는 페이지의 관리 방법

대부분의 페이지에는 추후에 삽입될 레코드를 위해 남겨두었던 공간이나 삭제로 인해 발생된 공간이 존재한다. 디스크 기반 DBMS에서는 이러한 빈 공간에 새로운 레코드를 저장하여 저장 공간 사용 효율(space utilization)을 높이는 방법을 사용한다. 이를 위해 빈 공간이 존재하는 모든 페이지들을 이중 연결 리스트(doubly linked list) 형태로 관리하며[5], 이를 빈 공간 리스트라 부른다.

그러나 기존의 빈 공간 리스트를 그대로 플래시 메모리에 적용하면 많은 문제점이 발생한다. 일반적으로 빈 공간 리스트는 관리의 편의를 위해 포인터를 페이지 내에 저장한다. 포인터가 페이지에 저장되면, 리스트를 변경하기 위해 추가적인 쓰기 연산을 수행하여야 한다.

또한, 이중 연결 리스트 구조로 인해 전, 후 페이지의 포인터를 변경하는 최대 2번의 추가적인 쓰기 연산이 필요하며, 이는 플래시 메모리 환경에서 심각한 성능 저하의 요인이 된다.

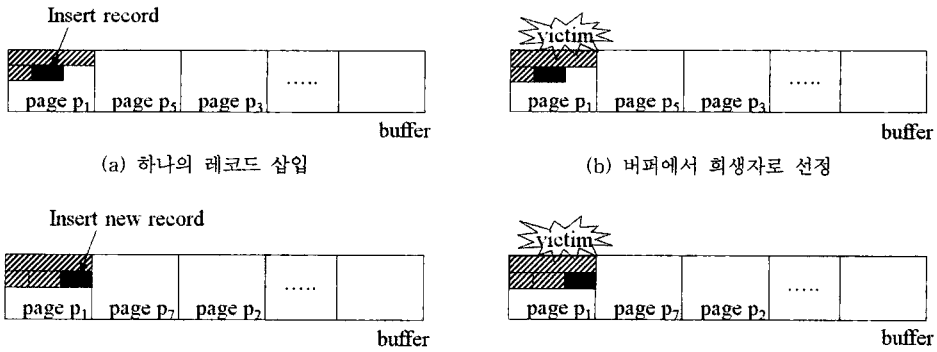
3.2 기존의 레코드 관리 방법

3.2.1 클러스터링 방법

클러스터링 방법은 유사한 키 값을 가지는 레코드들을 인접한 페이지에 저장하는 방법이다[5]. 다음은 클러스터링 방법의 레코드 삽입 과정이다.

- 단계 1. 삽입할 레코드와 가장 유사한 키 값을 가지는 레코드가 저장된 페이지를 탐색한다.
- 단계 2. 탐색된 페이지에 삽입할 레코드를 저장할 공간이 있을 경우, 해당 페이지에 레코드를 저장한다.
- 단계 3. 레코드를 저장할 수 없는 경우, 빈 공간 리스트의 첫 번째 페이지에 레코드를 저장하거나, 혹은 새로운 페이지를 할당 받아 레코드를 저장한다.

클러스터링 방법은 범위 질의를 처리할 때 함께 액세스될 확률이 높은 레코드들을 인접한 위치에 저장하기 때문에 범위 질의 성능은 우수하나 플래시 메모리의 특



(a) 하나의 레코드 삽입 (b) 버퍼에서 희생자로 선정

(c) 새로운 레코드 삽입 (d) 버퍼에서 희생자로 재선정

그림 1 동일한 페이지에 반복적으로 쓰기 연산을 수행

성 상 읽기 연산은 매우 빠르며, 이로 인해 얻는 성능의 향상은 미미하다.

반면, 일반적으로 레코드의 키 값은 불규칙적이므로, 삽입되는 n개의 레코드들은 n개의 서로 다른 페이지에 저장되며, 이로 인해 n번의 쓰기 연산이 발생한다. 이렇듯, 레코드 삽입으로 인한 빈번한 쓰기 연산은 플래시 메모리 환경에서의 성능을 매우 심각하게 저하시킨다.

3.2.2 비 클러스터링 방법

비 클러스터링 방법은 삽입되는 레코드의 키 값을 고려하지 않고 빈 공간 리스트의 첫 번째 페이지에 레코드를 저장하는 방법이다[5]. 다음은 비 클러스터링 방법의 레코드 삽입 과정이다.

- 단계 1. 빈 공간 리스트의 첫 번째 페이지를 탐색한다.
- 단계 2. 탐색된 페이지에 삽입할 레코드를 저장할 공간이 있을 경우, 해당 페이지에 레코드를 저장한다.
- 단계 3. 레코드를 저장할 수 없는 경우, 빈 공간 리스트의 다른 페이지에 레코드를 저장하거나, 혹은 새로운 페이지를 할당 받아 레코드를 저장한다.

비 클러스터링 방법은 레코드의 저장 위치가 키 값과 무관하게 결정되기 때문에 범위 질의 성능이 나쁘다. 그러나 플래시 메모리의 특성 상 성능 저하는 심각하지 않다.

이와 반대로 비 클러스터링 방법은 여러 개의 레코드들을 삽입하는 경우 클러스터링 방법에 비해 읽기, 쓰기 연산의 횟수가 감소할 수 있다. 비 클러스터링 방법은 삽입되는 레코드들이 동일한 페이지에 저장될 경우, 삽입이 일어나는 페이지는 계속 버퍼에 올라와 있을 확률이 높으며, 이로 인해 읽기, 쓰기 연산이 감소한다. 또한, 비 클러스터링 방법에서 버퍼의 효과로 인해 감소하는 쓰기 연산의 대부분은 페이지를 갱신하는 쓰기 연산이며, 이로 인해 소거 연산 또한 크게 감소한다.

3.3 비 클러스터링 방법의 재분석

플래시 메모리 환경에서는 비 클러스터링 방법이 클러스터링 방법에 비해 성능이 우수할 것으로 예상된다. 그러나 비 클러스터링 방법 역시 의도하지 않은 버퍼의 효과 때문일 뿐, 플래시 메모리의 특성에 적합하도록 설계되지 않았다. 이로 인한 성능 저하 요인이 많이 존재한다.

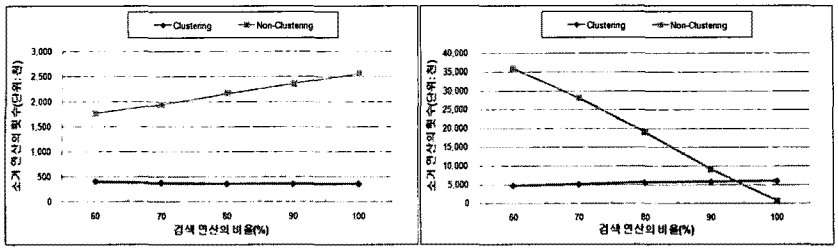
플래시 메모리 환경에서 비 클러스터링 방법의 문제점은 다음과 같다. 첫째, 버퍼 교체 전략의 영향으로 동일한 페이지에 반복적으로 쓰기 연산을 수행하는 현상을 발생할 수 있다. 이는 그림 1을 통해 알 수 있다. 페이지 p_1은 버퍼를 통해 하나의 레코드가 삽입된 후, 버퍼 교체 전략에 의해 희생자(victim)로 선정되어 쓰기 연산이 수행될 수 있다. 그러나 페이지 p_1은 빈 공간이 존재하므로, 삽입되는 레코드는 다시 페이지 p_1에 저장된다. 페이지 p_1은 다시 버퍼로 올라와야 하며, 레코드가 저장된 후, 다시 희생자로 선정될 수 있다. 이렇듯 비 클러스터링 방법에서 최악의 경우 삽입한 레코드 개수만큼의 쓰기 연산이 발생할 수 있다.

둘째, 기존의 빈 공간 리스트는 레코드 삭제로 인한 빈 공간 리스트의 변경이 빈번히 발생할 수 있다. 어떤 페이지에서 레코드가 삭제되면, 해당 페이지는 새롭게 빈 공간이 생겼으므로, 빈 공간 리스트에 추가되어야 한다. 따라서 빈 공간 리스트의 시작 부분의 페이지, 즉, 새로운 레코드들을 저장할 페이지는 빈 공간이 거의 존재하지 않으며, 이러한 페이지들은 한, 두 개의 레코드 삽입만으로 페이지가 가득 차게 된다. 이렇듯 기존의 빈 공간 리스트는 변경이 빈번히 일어나며, 이를 관리하기 위한 오버헤드가 크다.

4. 성능 비교 실험

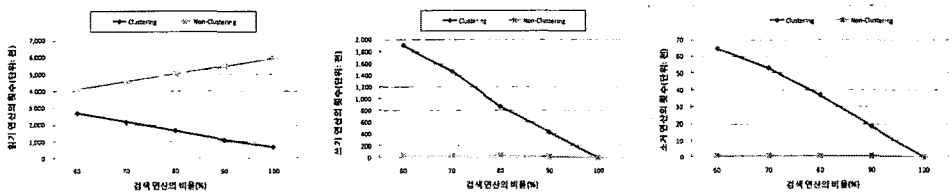
4.1 실험 환경

본 논문에서는 실험을 위해 플래시 메모리 개발 프레임 워크의 플래시 메모리 에뮬레이터[8]를 사용한다. 에뮬레이터는 플래시 메모리의 동작을 모방하며, 모의 성능 측정 기능을 제공한다. 삼성 2G NAND 플래시 메모리[6]를 기준으로 플래시 메모리 용량을 설정한다.



(a) 디스크 환경 (b) 플래시 메모리 환경

그림 2 저장 매체에 따른 검색 연산의 비율 변화에 대한 실험 결과



(a) 읽기 연산 (b) 쓰기 연산 (c) 소거 연산

그림 3 플래시 메모리에서의 검색 연산의 비율 변화에 대한 실험 결과

전체적인 실험의 구성은 다음과 같다.[1] 실험에서는 20만 개의 벌크로드 후, 20만 개의 레코드 검색, 삽입, 삭제 연산을 수행한다. 벌크로드를 수행할 때, 클러스터링 방법은 각 페이지의 70%만 채우며, 비 클러스터링 방법은 각 페이지의 빈 공간을 남겨두지 않고 가득 채운다. 실험을 수행할 때, 전체 검색 연산 중 80%는 하나의 레코드 검색 연산을, 나머지 20%는 범위 질의를 수행하며, 선택도(selectivity)가 각각 0.01%와 0.05%인 범위 질의를 수행한다. 또한, 하나의 페이지에 최대 20개의 레코드가 삽입 가능 하도록 레코드의 크기를 설정하였다.

실험의 성능 척도는 읽기, 쓰기, 소거 연산의 총 횟수를 측정하며, 각 연산의 횟수에 각각의 가중치를 곱한 값을 더해 전체 연산의 비용을 계산한다. 이 때, 쓰기 연산은 13, 소거 연산은 130의 가중치를 부여한다[7].

본 실험은 크게 두 가지로 구성된다. 실험 1에서는 클러스터링 방법과 비 클러스터링 방법의 성능을 각각 디스크 환경과 플래시 메모리 환경에서 측정하여 비교한다. 이를 위해 레코드의 검색, 삽입 연산 수행 할 때 검색 연산의 비율을 60%, 70%, 80%, 90%, 100%로 변화시키며 성능을 측정한다. 실험 2에서는 클러스터링 방법과 비 클러스터링 방법의 빈 공간 리스트 관리 오버헤드를 측정한다. 이를 위해 레코드의 검색, 삽입, 삭제 연산을 수행 할 때, 전체 연산 중 검색 연산의 비율을 80%로, 나머지 20%의 삽입, 삭제 연산 중 삽입 연산의 비율을 80%로, 삭제 연산의 비율을 20%로 설정하여 성능을 측정한다. 이 외에도 레코드의 크기와 DBMS의 버퍼의 개수를 변경하며 실험하였으나, 전체적인 경향이 유사하므로 본 논문에서는 생략한다.

4.2 실험 결과

실험 1. 클러스터링 방법과 비 클러스터링 방법의 성능 비교

본 실험에서는 저장 매체로 디스크와 플래시 메모리를 사용하는 경우 각각에 대하여 클러스터링 방법과 비 클러스터링 방법의 성능을 측정한다. 실험 1의 결과는 그림 2, 3을 통해 알 수 있다. 그림 2, 3에서 그래프의 x축은 매개 변수인 검색 연산의 비율의 변화를 나타내고, y축은 측정한 성능 척도를 나타낸다.

그림 2는 검색 연산의 비율을 증가시키며 측정한 전체 연산의 비용 변화를 나타낸 것이다. 실험 결과, 디스크 환경에서는 클러스터링 방법의 성능이 우수한 반면, 플래시 메모리 환경에서는 검색 연산의 비율이 95% 이하일 때 비 클러스터링 방법의 우수하다. 이는 비 클러스터링 방법의 나쁜 범위 질의 성능에도 불구하고, 레코드를 삽입하기 위한 쓰기, 소거 연산의 비용이 클러스터링 방법에 비해 매우 적어, 오히려 비 클러스터링 방법의 성능이 우수한 것으로 나타났다.

클러스터링 방법은 범위 질의 성능의 우수함으로 인해, 검색 연산의 비율이 95% 이상일 때는 오히려 클러스터링 방법의 성능이 우수하다. 이는 그림 3에서 더욱 확실히 알 수 있다.

그림 3은 그림 2의 결과를 읽기, 쓰기, 소거 연산의 횟수에 따라 각각 나타낸 것이다. 그림에서 읽기 연산의

1) 본 논문에서는 실험 설계 과정에서 다음의 논문을 참조하였다[9].

성능은 클러스터링 방법이 우수하다. 그러나 쓰기, 소거 연산의 성능은 비 클러스터링 방법이 버퍼의 효과로 인해 매우 우수하다.

실험 2. 빈 공간 리스트의 관리 오버헤드 분석

실험 2는 클러스터링 방법과 비 클러스터링 방법에서 삭제 연산으로 인한 빈 공간 리스트 관리 오버헤드를 측정한다. 실험 2의 결과는 표 2를 통해 알 수 있다.

표 2는 전체 쓰기 연산 중 빈 공간 리스트 관리만을 위해 수행한 쓰기 연산의 비율을 나타낸 것이다. 측정 결과, 클러스터링 방법에서는 7%, 비 클러스터링 방법에서는 14.4%로 비 클러스터링 방법에서 빈 공간 리스트 관리 오버헤드가 매우 크다는 것을 알 수 있다. 그러므로 이러한 오버헤드를 줄이면 비 클러스터링 방법의 성능이 크게 향상될 것이라고 예상할 수 있다.

표 2 빈 공간 리스트 관리 오버헤드

레코드 관리 방법	빈 공간 리스트 관리 오버헤드
클러스터링	7%
비 클러스터링	14.4%

5. 레코드 관리 방법 설계 시 고려해야 할 사항

플래시 메모리는 레코드 관리 방법을 설계함에 있어 기존의 저장 매체에는 없던 여러 가지 고려사항이 발생한다. 본 절에서 플래시 메모리 환경에서 레코드 관리 방법을 설계할 때 고려할 사항들을 제안한다.

첫째, 레코드들을 저장할 때 키 값에 의한 클러스터링을 고수할 필요가 없다. 플래시 메모리 환경에서는 읽기 연산의 수를 줄여서 얻는 성능의 향상보다는 클러스터링을 유지하기 위해 증가된 쓰기 연산으로 인한 성능 저하가 더욱 크다. 따라서 쓰기 연산의 수를 줄이는 것이 레코드 관리 방법의 성능 향상에 더 효율적이다.

둘째, 레코드들이 삽입되는 페이지는 가능한 많은 레코드들을 삽입한 후 쓰기 연산을 수행하여야 한다. 이 경우, 여러 번의 레코드의 삽입을 한 번의 쓰기 연산으로 처리할 수 있어 쓰기 연산을 크게 줄일 수 있다. 그러나 버퍼 교체 전략의 영향으로 레코드를 더 삽입할 수 있음에도 불구하고 버퍼에서 삭제되어 쓰기 연산을 수행하는 경우가 많았다. 따라서 추후 레코드 관리 방법을 설계할 때에는 이를 고려하여야 한다.

셋째, 빈 공간을 관리하는 오버헤드를 줄여야 한다. 기존의 빈 공간 리스트는 포인터 관리 오버헤드가 매우 크다. 또한 페이지의 빈 공간의 크기를 고려하지 않아 리스트의 잦은 변경이 발생하였다. 따라서 빈 공간이 큰 페이지에 레코드를 저장하는 것이 좋다. 이 경우, 빈 공간 리스트 변경 오버헤드를 줄일 수 있어, 레코드 관리 방법의 성능이 더욱 우수해질 것이다.

6. 결 론

플래시 메모리는 읽기, 쓰기, 소거 연산의 수행 속도의 차이가 크다. 특히, 쓰기 연산은 비용이 가장 큰 소거 연산을 유발하고, 플래시 메모리의 수명을 단축시키는 요인이다. 따라서 플래시 메모리에서는 쓰기 연산의 수를 감소시키는 것이 중요하다.

본 논문의 공헌은 다음과 같다. 첫째, 플래시 메모리의 특성이 클러스터링 방법과 비 클러스터링 방법에 미치는 영향을 분석하고, 실험 1을 통하여 비 클러스터링 방법의 성능의 우수함을 규명하였다. 둘째, 플래시 메모리 환경에서 나타나는 비 클러스터링 방법의 구조적인 문제점들을 지적하고, 실험 2를 통하여 이를 규명하였다. 셋째, 분석한 내용을 기반으로 추후 플래시 메모리 환경에 적합한 레코드 관리 방법을 설계할 때 고려해야 할 사항들을 제안하였다.

참 고 문 헌

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

[2] S. Lee and B. Moon, "Design of Flash-Based DBMS: An In-Page Logging Approach," In Proc. ACM Int'l. Conf. on Management of Data, ACM SIGMOD, pp. 55-66, 2007.

[3] A. Kawaguchi, S. Nishioka, and H. Motoda, "A Flash-Memory Based File System," In Proc. USENIX Technical Conf. on Unix and Advanced Computing Systems, pp. 155-164, 1995.

[4] C. Wu, L. Chang, and T. Kuo, "An Efficient B-Tree Layer for Flash-Memory Storage Systems," In Proc. Int'l. Conf. on Real-Time and Embedded Computing Systems and Applications, RTCSA, Vol.LNCS 2968, pp. 409-430, 2003.

[5] J. Gray and A. Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1995.

[6] Samsung, 2G NAND Flash Memory, http://www.samsung.com/products/semiconductor/NANDFlash/, 2007.

[7] K. Yim, "A Novel Memory Hierarchy for Flash Memory Based Storage Systems," Journal of Semiconductor Technology and Science, Vol.5, No.4, pp. 262-269, 2005.

[8] S. Kim et al., "A Development Framework for Reliable Flash Memory Software," SK Telecommunications Review, Vol.15, No.4, pp. 638-646, 2005.

[9] J. Rao and K. Ross, "Making B+-Trees Cache Conscious in Main Memory," In Proc. ACM Int'l. Conf. on Management of Data, ACM SIGMOD, pp. 475-486, 2000.