플래시 메모리 환경에서 클러스터링 방법과 논 클러스터링 방법의 성능 분석

배덕호* 장지웅°** 김상욱*
한양대학교 전자컴퓨터통신공학부*, 한국산업기술대학교 게임공학과**
smith@zion.hanyang.ac.kr, jwchang@kpu.ac.kr°, wook@hanyang.ac.kr

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

Duck-Ho Bae* Ji-Woong Chang°** Sang-Wook Kim*
Dept. of Electronics and Computer Engineering, Hanyang University*
Dept. of Game & Multimedia Engineering, Korea Polytechnic University**

I. 서론 플래시 메모리는 기존의 저장 매체들과는 다른 플래시 메모리만의 고유한 특성을 가진다. 첫째, 플래시 메모리는 쓰기 연산의 수행 속도가 읽기 연산에 비해 매우 느리다[1]. 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 환경에서는 쓰기 연산을 감소시키는 것이 매우 중요하다. 본 논문에서는 이러한 플래시 메모리의 특성이 레코드 관리 방법에 미치는 영향을 분석한다.

II. 플래시 메모리 환경에서 기존의 레코드 관리 방법의 고찰 디스크 기반 DBMS의 대표적인 레코드 관리 방법에는 클러스터링 방법(clustering method)과 논 클러스터링 방법(non-clustering method)이 있다. 클러스터링 방법은 범위 질의 시 함께 액세스될 확률이 높은 레코드들을 인접한 위치에 저장하기 때문에 범위 질의 성능은 우수하여 주로 사용되었다[2]. 그러나 플래시 메모리의 특성 상 읽기 연산은 매우 빠르며, 이로 인해 얻는 성능의 향상은 미미하다. 반면, 레코드를 삽입할 위치를 탐색하는 작업이 선행되어야 하므로 삽입 연산의 오버헤드가 크다. 또한, 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. 따라서 클러스터링 방법은 플래시 메모리 환경에 적합하지 않은 방법이다.

반면, 논 클러스터링 방법은 레코드의 저장 위치가 키 값과 무관하게 결정되기 때문에 범위 질의 성능이 나쁘다[2]. 그러나 플래시 메모리의 특성 상 성능 저하는 그리 심각하지 않다. 이와 반대로 논 클러스터링 방법은 여러 개의 레코드들을 삽입하는 경우 클러스터링 방법에 비해 쓰기 연산의 횟수가 감소할 수 있다. 만약 레코드가 저장되는 페이지가 버퍼에 올라와 있으면 별도의 액세스 없이 레코드를 저장할 수 있다. 논 클러스터링 방법은 삽입되는 레코드들이 동일한 페이지에 저장될 확률이 높으며, 삽입이 일어나는 페이지가 계속 버퍼에 올라와 있을 확률이 높다. 이로 인해 쓰기 연산이 크게 감소한다. 감소하는 쓰기 연산의 대부분은 페이지를 갱신하는 쓰기 연산이며, 이로 인해 소거 연산 또한 크게 감소한다. 이렇듯 논 클러스터링 방법은 플래시 메모리에 적합한 레코드 관리 방법이다.

레코드 관리 방법의 성능에 큰 영향을 미치는 또 다른 중요한 요소에는 빈 공간 리스트(free space list) 관리가 있다. 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[2], 이를 빈 공간 리스트라 부른다. 빈 공간 리스트는 클러스터링 방법과 논 클러스터링 방법 모두 관리한다[2]. 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. 이렇듯, 플래시 메모리 환경에서는 페이지 내에서 빈 공간 리스트를 관리하는 기존의 방법은 관리 오버헤드가 매우 크다.

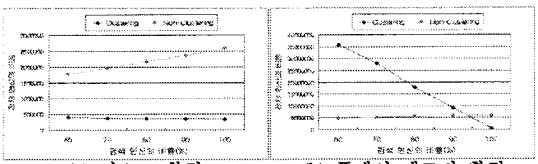
플래시 메모리 환경에서는 논 클러스터링 방법이 클러스터링 방법에 비해 전반적인 성능이 우수하다. 그러나 논 클러스터링 방법 역시 플래시 메모리의 특성에 적합하도록 설계되지 않아 성능 저하 요인이 많이 존재한다. 첫째, 버퍼 교체 전략의 영향으로 동일한 페이지에 반복적으로 쓰기 연산을 수행하는 현상을 발생할 수 있다. 페이지 p_1에 하나의 레코드가 삽입된다고 가정하자. 페이지 p_1은 버퍼를 통해 레코드가 삽입된 후, 버퍼 교체 전략에 의해 희생자(victim)로 선정되어 쓰기 연산이 수행될 수 있다. 만약 페이지 p_1에 빈 공간이 존재한다면, 삽입되는 레코드는 다시 페이지 p_1에 저장된다. 페이지 p_1은 다시 버퍼로 올라와야 하며, 레코드가 저장된 후, 다시 희생자로 선정될 수 있다. 이렇듯 논 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.

III. 성능 분석 본 실험은 크게 두 가지로 구성된다. 실험 1에서는 클러스터링 방법과 논 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 논 클러스터링 방법

의 빈 공간 리스트 관리 오버헤드를 측정한다. 빈 공간 리스트는 레코드 삭제로 인한 변경이 매우 빈번히 일어나며, 이를 관리하기 위한 오버헤드가 매우 커진다. 이를 측정하기 위해 레코드의 검색, 삽입, 삭제 연산을 수행 할 때, 전체 연산 중 검색 연산의 비율을 80%로 고정한 후, 나머지 20%의 삽입, 삭제 연산 중 삭제 연산의 비율이 20%일 때의 성능을 측정한다.

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

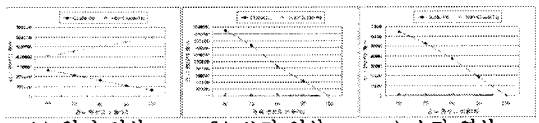
그림 1은 검색 연산의 비율을 증가시키며 측정한 전체 연산의 비용 변화를 나타낸 것이다. 실험 결과, 디스크 환경에서는 클러스터링 방법의 성능이 우수한 반면, 플래시 메모리 환경에서는 논 클러스터링 방법의 우수하다. 클러스터링 방법이 범위 질의 성능이 우수함에도 불구하고, 레코드를 클러스터링하기 위한 쓰기, 소거 연산의 비용이 매우 커서 오히려 논 클러스터링 방법의 성능이 우수한 것으로 나타났다.



(a) 디스크 환경. (b) 플래시 메모리 환경.

그림 1. 저장 매체에 따른 실험 결과.

그림 2는 그림 1의 결과를 읽기, 쓰기, 소거 연산의 횟수에 따라 각각 나타낸 것이다. 그림에서 읽기 연산의 성능은 클러스터링 방법이 우수하나, 쓰기, 소거 연산의 성능은 논 클러스터링 방법이 매우 우수하다. 클러스터링 방법과 논 클러스터링 방법의 쓰기 연산의 횟수는 매우 큰 차이를 보이는데, 이는 논 클러스터링 방법은 버퍼의 효과로 인해 쓰기 연산의 횟수를 크게 줄일 수 있으며, 이로 인해 데이터를 갱신하는 쓰기 연산이 크게 줄어 두 방법 간의 소거 연산의 횟수 또한 매우 큰 차이를 보이게 된다.



(a) 읽기 연산. (b) 쓰기 연산. c) 소거 연산.

그림 2. 플래시 메모리에서의 실험 결과.

실험 2. 빈 공간 리스트의 관리 오버헤드 분석 실험 2는 클러스터링 방법과 논 클러스터링 방법에서 삭제 연산으로 인한 빈 공간 리스트 관리 오버헤드를 측정한다. 실험 2의 결과는 표 1을 통해 알 수 있다.

표 1은 전체 쓰기 연산 중 빈 공간 리스트 관리만을 위해 수행한 쓰기 연산의 비율을 나타낸 것이다. 측정 결과, 클러스터링 방법에서는 7%, 논 클러스터링 방법에서는 14.4%로 논 클러스터링 방법에서 빈 공간 리스트 관리 오버헤드가 매우 크다는 것을 알 수 있다. 그러므로 논 클러스터링 방법은 빈 공간 리스트 관리 오버헤드를 줄이면 성능이 크게 향상될 것이라고 예상할 수 있다.

표 1. 빈 공간 리스트 관리 오버헤드

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

IV. 레코드 관리 방법 설계 시 고려해야 할 사항 본 절에서는 플래시 메모리 환경에서 레코드 관리 방법을 설계할 때 고려할 사항을 제안한다. 첫째, 레코드들을 저장할 때 키 값에 의한 클러스터링을 고수할 필요가 없다. 쓰기 연산의 오버헤드가 매우 큰 플래시 메모리 환경에서는 읽기 연산의 수를 줄여서 얻는 성능의 향상보다는 클러스터링을 유지하기 위해 증가된 쓰기 연산으로 인한 성능 저하가 더욱 크다. 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. 셋째, 빈 공간을 관리하는 오버헤드를 줄여야 한다. 기존의 빈 공간 리스트는 페이지의 빈 공간의 크기를 고려하지 않아 리스트의 잦은 변경이 발생하였으며, 이를 관리하기 위한 오버헤드가 매우 컸다. 따라서 빈 공간이 많이 존재하는 페이지에 레코드를 저장하여 쓰기 연산의 수를 줄어야 한다.

V. 결론 본 논문에서는 플래시 메모리 환경에서의 클러스터링 방법과 논 클러스터링 방법의 성능을 분석하였다. 첫째, 플래시 메모리의 특성이 클러스터링 방법과 논 클러스터링 방법에 미치는 영향을 분석하고, 실험을 통하여 논 클러스터링 방법의 성능의 우수함을 규명하였다. 둘째, 플래시 메모리 환경에서 나타나는 논 클러스터링 방법의 문제점들을 지적하고, 실험을 통하여 이를 규명하였다. 셋째, 분석한 내용을 기반으로 추후 플래시 메모리 환경에 적합한 레코드 관리 방법을 설계할 때 고려해야 할 사항들을 제안하였다.

감사의 글 본 논문은 제주대학교를 통한 정보통신부 및 정보통신진흥원의 대학 IT연구센터 지원사업(IITA-2005-C1090-0502-0009) 및 2007년도 정부(과학기술부)의 재원으로 한국과학재단(R01-2007-000-11773-0)의 연구비 지원을 받았습니다.

참고 문헌

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

[3] 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.