하이브리드 저장 시스템을 위한 내장형 노드 캐시 관리

Embedded Node Cache Management for Hybrid Storage Systems

변 시 우*, 허 문 행, 노 창 배
Byun Si Woo, Hur Moon Haeng, Roh Chang Bae

Abstract - The conventional hard disk has been the dominant database storage system for over 25 years. Recently, hybrid systems which incorporate the advantages of flash memory into the conventional hard disks are considered to be the next dominant storage systems to support databases for desktops and server computers. Their features are satisfying the requirements like enhanced data I/O, energy consumption and reduced boot time, and they are sufficient to hybrid storage systems as major database storages. However, we need to improve traditional index node management schemes based on B-Tree due to the relatively slow characteristics of hard disk operations, as compared to flash memory. In order to achieve this goal, we propose a new index node management scheme called FNC-Tree. FNC-Tree-based index node management enhanced search and update performance by caching data objects in unused free area of flash leaf nodes to reduce slow hard disk I/Os in index access processes.

Key Words : Hybrid Hard Disk, Flash Caching, Storage System, Tree Index, Flash Memory

1. 서 론

지난 25년 동안, 하드 디스크는 대부분의 파일 시스템에서 사용되는 절대적인 저장 미디어였다. 그러나 비록 디스크 저장 용량이 급속도로 증가하고는 있을 지라도, 디스크의 기계적인 지연시간은 매년 15%정도로만 개선되고 있다. 반면에 메모리와 CPU는 매년 50% 정도의 빠른 속도로 개선 발전되고 있다. 과거 10년 동안 이러한 접근 속도 차이는 5-6배 이상 벌어지게 되었다[1,2]. 즉, 이러한 기계적 특성에 따른 속도 차이에 의하여 하드 디스크는 점차로 한계 성능에 도달하게 된다.

최근 각종 휴대형 소형 정보기기들이 대중화됨에 따라, 정보 저장용 미디어로 기존의 하드 디스크를 대체하여 플래시 메모리가 보편적으로 활용되게 되었다. 그러나 플래시 메모리와 하드 디스크 모두 보편적인 저장 장치로서 많은 장점을 가지고 있는 반면, 적잖은 단점도 가지고 있다.

먼저, 하드 디스크는 널리 알려진 저장장치로서 고용량 저장 능력과 저렴한 저장 비용이 장점이나 기계적인 내부 구조로서 소음, 전력, 속도, 내충격성 등에서 많은 단점을 내포하고 있다. 반면 플래시 메모리는 이러한 하드 디스크에 비하여 저용량성과 상대적으로 높은 저장비용이 단점이 되고, 기타 측면에서 장점을 가지고 있다.

본 연구에서는 이러한 상반되는 두 저장 장치의 고유한 특성을 상호 보완하여 우수한 저장 및 입출력 성능을 발휘할 수 있도록 하이브리드 저장 시스템에 적합한 인덱스 노드의 빈 공간을 재활용한 데이터 플래시 노드 캐싱 기법을 제안하고자 한다.

저자 소개
* 正 會 員 : 安養大學校 디지털미디어學科 助敎授 · 工博

2. 관련 연구

현재의 데이터 저장 시스템에서 사용되는 일반적인 색인 관리 기법에 대한 기존의 연구를 분류해 보면, 크게 디스크 기반 색인 시스템과 메인 메모리 기반 색인 시스템으로 접근 방향을 나눌 수 있다.

개념적으로 디스크 기반 색인 및 저장 기술의 목표는 디스크의 접근 횟수와 디스크 공간을 최소화하는 것이며, 디스크 I/O를 가장 큰 비용으로 고려한다. 그러나 메모리 기반 색인 및 저장 기술은 디스크의 접근이 없으므로, CPU 수행 시간을 줄이고, 최소한의 메모리 공간을 사용하는 것이 중요하다. 저렴하고 대용량인 디스크 기반 저장 방식이 저장 비용 측면에서는 유리하지만, 아무리 I/O 버퍼를 많이 할당하더라도 속도 측면에서는 메모리 기반 저장 방식보다는 매우 느리다. 그러나 두 방식 모두 플래시 메모리 기반 저장 환경에는 부적합하므로 플래시 메모리의 고유한 특성을 고려하여 새로 개발하여야 한다.

2.1 디스크 기반 색인 관리 시스템

디스크 기반 색인 및 저장 시스템에서는 검색 시에 디스크 접근을 최소화하기 위하여 노드의 크기를 디스크 페이지와 같은 크기나 배수로 설정하고, 되도록이면 많은 엔트리를 한 노드에 넣어야 유리하다. 한 노드에 많은 엔트리가 들어갈 경우 모든 엔트리를 검색해야 하기 때문에 검색 시에 연산 처리 성능은 당연히 저하된다. 그러나 디스크 기반 저장 시스템에서는 이러한 검색 성능 저하보다는 디스크 접근에 의한 성능 저하가 훨씬 더 크기 때문에 한 노드에 많은 엔트

리를 넣는 방향으로 설계한다.[3] 주로 B-Tree, B'-Tree[4] 계열의 인덱스가 많이 사용되며, 공간 색인으로는 R-Tree, R*-Tree[5]계열이 사용되고 있다. R-Tree계열의 인덱스는 일반적으로 디스크 기반 색인으로서 메인 메모리를 사용하지 않는 공간 인덱스로 구현되었다. R-Tree는 삽입 연산, 삭제 연산, 분할이나 병합과 같은 리벨런싱 연산이 수행될 경우, 동일한 위치로 많은 섹터가 판독 또는 재기록 되는 부담이 있다. 디스크 기반 시스템에서는 이러한 연산들의 검색 효율을 위하여 디스크의 연속 섹터에 그룹핑되어 있으며, 디스크 특성을 잘 고려한 R-Tree는 디스크 기반 시스템에서 실제로 매우 효과적이다.

2.2 메인 메모리 기반 색인 관리 시스템

메인 메모리 기반 색인 및 저장 시스템은 디스크 기반 시스템에 비하여 디스크에 접근하는 시간이 대폭 줄어들기 때문에 훨씬 더 빠른 성능을 보여줄 수 있다. 그러나 문제는 시스템 전원이 차단되는 등의 치명적인 장애가 발생할 경우이다. 디스크 기반 저장 시스템은 디스크에 데이터를 일일이 저장하므로 장애에 매우 강하다. 반면에 메인 메모리 데이터베이스는 메모리에 저장된 데이터가 사라지므로 디스크 기반 저장 시스템과는 다른 백업, 복구 방식이 필요하다.

메인 메모리 기반 색인 시스템에서는 일반적으로 T-Tree가 1차원 데이터를 위한 색인으로서 좋은 성능을 보이며, 비교적 적합하다고 알려져 있다.[6] T-Tree는 이진 검색과 높이 균형을 가지고 O(logN)의 트리 순회가 가능한 AVL-Tree의 빠른 검색 특성을 가지고 있으며, 한 노드 안에 여러 개의 데이터를 가지고 저장효율이 좋은 B-Tree의 성질도 함께 가지고 있다. T-Tree는 빠른 처리속도와 메모리 사용의 최적화라는 메인 메모리의 특성에 적합한 구조로 알려져 있다.[7]

디스크 기반 색인은 노드의 접근 횟수가 디스크의 I/O의 수와 같으므로, 트리의 깊이는 성능에 큰 영향을 미쳤다. 삽입/검색 시에는 데이터를 삽입/검색할 노드를 찾기 위하여 비교하면서 내려가는 노드의 개수가 성능에 가장 큰 영향을 준다. 따라서 깊이가 얕고 넓게 퍼진 트리를 써서 삽입/검색 시에 I/O 비용을 최소화 하였다. 반면에, 메모리 기반 색인의 접근 비용은 포인터로 노드의 메모리 주소를 획득하는 비용이므로 크지 않다. 따라서 디스크 기반 색인에서 선호하는 얕고 넓게 퍼진 트리 구조는 더 이상 유용하지 않다. 메모리 기반 색인에서는 디스크 기반 색인과는 달리 노드의 용량을 변화시켜 트리의 깊이와 비교횟수를 조절하여 성능 향상이 가능하다.[7]

3. 하이브리드 특성을 고려한 내장형 노드 캐시 관리

B-Tree 계열의 색인은 데이터의 삽입, 삭제, 검색을 효율적으로 처리하기 위하여 가장 널리 사용되는 색인 구조이다. B-Tree의 삽입, 삭제, 리벨런싱은 많은 노드들이 읽혀지고 쓰여 지게 한다. 기본적으로 B-Tree는 하위 노드에 대한 포인터 정보와 함께 데이터 관련 정보를 한 노드 안에 보관한다. 그러나 B'-Tree는 중간 노드에 데이터 관련 정보를 넣지 않고 리프 노드에 이를 저장한다. 즉, 중간 노드에서는 순수한 색인 정보만을 저장하므로, 색인 자체의 저장 효율이 높아진다. 또한, 리프 노드에서는 색인 검색의 도움 없이 바로 순차적인 접근이 가능한 장점도 있다. 따라서 플래시 메모리의 색인으로서는 기본적인 B-Tree 보다도 B'-Tree가 더 적합하다.

그리고 B'-Tree에서 수많은 임의의 값들을 삽입하고 삭제하는 시뮬레이션을 수행하여 분석한 결과 69%정도 차 있을 때가 가장 효율적인 것으로 분석되었다. 한 노드에 최대한 많이 저장하면 검색 노드수도 줄어들고 저장 효율은 향상되지만, 삽입, 삭제시 노드의 변동이 너무 빈번하여 많은 수의 쓰기 연산을 유도하여 결과적으로 더 손실이 크다. 즉, 평균 점유율(avg fill factor)이 69%일 때 가장 안정되어 인덱스 리벨런싱과 재구성을 하지 않고도 인덱스 연산을 수행할 수 있다. 따라서 인덱스 구성시 이 평균점유율에 맞추어 B'-Tree를 조직하게 된다. 이러한 분석 결과와 기타 B'-Tree에 대한 자세한 이론은 [4]에 구체적으로 잘 설명되어 있다.

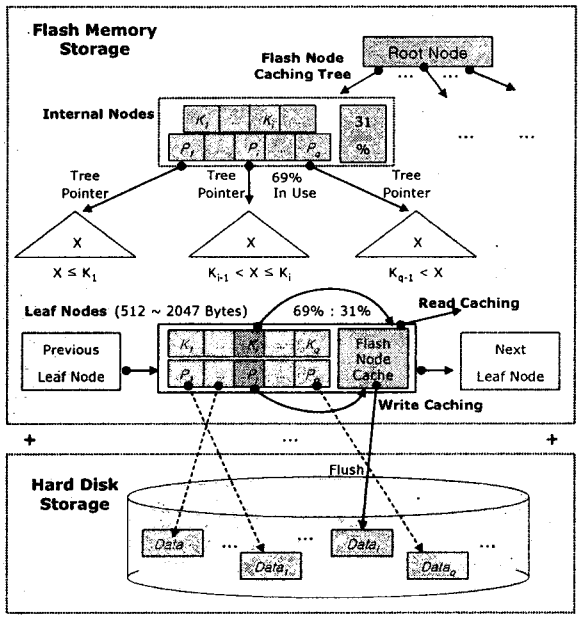
전술한 바와 같이, 플래시 메모리는 하드 디스크에 비하여 고가의 고속 메모리이므로 상대적으로 저장용량이 충분하지 않다. 만일, 하이브리드 하드 디스크에 작은 양의 데이터를 저장한다면, 플래시 메모리에 데이터베이스를 전부 적재하고 고성능 모드로 운영할 수도 있다. 하지만, 데이터의 건수가 상당히 많아진다면, 일부는 하드 디스크로 내려야만 한다. 즉, 데이터베이스에서 매우 빈번히 접근되는 색인은 루트 노드에서부터 순차적으로 플래시 메모리에 유지하고, 하위 리프 노드에 연결된 실제 데이터는 너무 방대하므로 하드 디스크로 내려야 한다. 또한, 실제 데이터 중에서도 일부만이 매우 빈번히 접근되는 공간적, 시간적 로컬리티가 존재하므로 이를 고려하여야 한다.

전술한 바와 같이 비트리 색인은 효율을 위하여 한 노드당 평균 69%정도만을 엔트리(키와 포인터)공간으로 점유하고 있다. 이때, 그 노드의 나머지 31%의 저장 공간은 추후 발생 가능한 엔트리 삽입을 위하여 빈 공간으로 계속 유지하고 있다. 즉, 추후 엔트리 삽입을 위하여 현재는 낭비되고 있는 저장 공간이다.

본 연구에서는 이러한 플래시 메모리의 유휴 공간을 활용하여 인덱스 노드에 데이터를 캐싱하는 플래시 노드 캐싱(Flash Node Caching) 기법을 제안한다. 본 기법의 기본 아이디어는 바로 현재 유휴 공간인 31% 저장 공간을 읽기/쓰기 캐시로 활용하자는 것이다. 플래시 메모리는 고속 저장이 가능하므로 충분히 캐시로서의 성능 개선을 얻을 수 있으며, 기존 휘발성 램(RAM) 방식의 캐시에 비하여 비휘발성 영구 저장이 가능하므로, 갑작스런 전원차단에 따른 데이터 손실도 방지할 수 있다. 더우기 하드 디스크에 비하여 매우 고가인 플래시 메모리는 빈 공간의 낭비 없이 더욱더 효율적으로 활용되어야 한다. 또한, 추후에 해당 노드에 신규 엔트리가 삽입되면, 이때는 이 노드는 캐시 역할을 중단하고, 추가 엔트리를 저장하면 되므로 공간 재활용에 따른 오버헤드가 발생하지 않는다. 그리고 추가 엔트리가 저장될 때 그 노드에 대한 플래시 메모리 쓰기가 한번 수행되는데, 이때 해당 데이터도 그 노드 안에 같이 저장하여 쓰여 진다. 따라서 엔트리에서 포인팅하는 데이터를 캐시에 저장하기 위한 별도의 추가적인 쓰기 연산이 불필요하므로, 플래시 메모리의 쓰기 연산 횟수를 감소시킨다. 읽기보다 느린 플래시 메모리의 쓰기 속도를 고려할 때 이는 곧 하이브리드 저장 시스템의 성능 향상에 기여한다.

결과적으로 본 기법은 빈 공간을 재활용하므로 큰 오버헤

드 없이, 고성능 캐싱 효과와 안정성을 모두 얻을 수 있으므로 저장 시스템의 성능 개선에 크게 기여할 수 있다. 그림 1은 제안된 플래시 노드 캐싱을 지원하는 B+-Tree 인덱스 구조로서, FNC-Tree(Flash Node Caching Tree)라고 하였다.



(그림 1) 플래시 노드 캐싱 트리의 구조

본 기법의 FNC-Tree에서 플래시 메모리의 입출력의 효율을 위하여 한 페이지당 하나의 노드를 수용한다. 데이터 수정 또는 엔트리 삽입이 발생하면 루트 노드에서 중간 노드를 거쳐서 리프 노드에서 도달하며, 리프 노드에서 연결된 하드 디스크의 데이터가 저장되게 된다. 본 기법의 빈 공간을 활용하는 노드는 리프 노드만을 대상으로 하였다. 물론, 루트 노드와 중간 노드를 활용하면 좀 더 효과적이겠지만, 실험결과 리프 노드에 비하여 그 수가 상대적으로 5%이하로 매우 적으면서, 계산과정 너무 복잡하게 만들므로, 본 기법은 데이터에 가장 근접해있는 노드인 리프노드만을 대상으로 하였다.

본 기법의 리프 노드 구조는 일반 리프 노드 구조체에서 이 노드가 현재 캐시 모드로 동작하는지에 대한 변수와 몇 바이트부터 캐시 데이터가 저장되는지에 대한 오프셋 변수를 추가로 포함하고 있다. 그리고 리프 노드에서 데이터가 캐싱되는 엔트리(키+포인터)는 데이터 포인터(pointer_to_data)를 하드 디스크에 있는 데이터 노드가 아닌 바로 자기 자신의 현재 노드를 포인팅하게 된다.

FNC-Tree도 다른 B+-Tree와 마찬가지로 노드의 엔트리 점유율이 평균점유율인 69%에서 운영중에 이를 벗어나서 최소 50%에서 최대 100%까지 변동할 수 있다. 이 변동으로 인하여 빈공간이 줄어서 캐시 데이터의 저장이 불가능하게 되면, 캐싱된 데이터를 하드 디스크로 반영한 후 그 캐시영역을 빈 공간으로 모두 반납하게 된다.

4. 결론

본 논문에서는 기존의 대표적 저장 장치인 하드 디스크와 최근 휴대용 고속 데이터 저장 장치로 많이 사용되는 플래시 메모리를 융합한 개선된 하이브리드 저장시스템을 살펴보았다. 또한, 하이브리드 저장시스템에서 인덱스 연산의 성능 향상을 위한 새로운 인덱스 관리 기법을 제안하였다. 기존의 하드 디스크 및 메모리를 위한 B-Tree 기반 인덱스 기법을 개선하여, 제안 기법은 플래시 리프 노드의 사용되지 않는 여분의 영역을 활용하여 읽기와 쓰기의 데이터 캐싱 효과를 증대시킴으로써 전반적인 시스템 성능을 높일 수 있다.

참 고 문 헌

[1] Andy Wang, Geoff Kuenning, Peter Reiher, and Gerald Popek, "The Conquest File System: Better Performance Through a Disk/Persistent-RAM Hybrid Design", ACM Transactions on Storages, vol. 2, no. 3, pp. 309-348, Aug 2006.

[2] Samsung, What is Hybrid HDD?,http://www.samsung.com/Products/HardDiskDrive/whitepapers/WhitePaper_12.htm, 2007

[3] Cha S. K., J. H. Park, and B. D. Park, "Xmas: An Extensible Main-Memory Storage System," Proc. of 6th ACM Int'l Conference on Information and Knowledge Management, Nov. 1997.

[4] 황규영, 홍의경, 음두현, 박영철, 김진호, "데이터베이스 시스템", 생능출판사, 2000

[5] Beckmann N., H. P. Kriegel, R. Schneider, and B. Seeger, "The R*Tree: An Efficient and Robust Access Method for Points and Rectangles," Proc. of ACM SIGMOD Intl. Symp. on the Management of Data, pp. 322-331, 1990.

[6] Lehman T. J. and Carey M. J., "Query processing in main memory database management systems", in Proc. ACM SIGMOD Conf., Washington, DC, May, 1986.

[7] 이창우, 안경환, 홍봉희, "이동체 데이터베이스를 위한 메인 메모리 색인의 성능 결정 요소에 관한 연구", 정보처리학회 춘계 학술대회 논문집, 제10권 1호, pp. 1575-1578, 2003