고속 플래시 스토리지를 위한 핫-콜드 인덱스의 성능 평가

변시우*, 허문행*
*안양대학교 디지털미디어공학과
e-mail:swbyun@anyang.ac.kr

A Performance Evaluation of Hot-Cold Index for High-Speed Flash Storages

Siwoo Byun*, Moonhaeng Hu*r
*Dept of Digital Media, Anyang University

요 약

데스크탑 및 이동형 컴퓨터의 저장 장치를 지원하는 플래시 메모리는 비휘발성, 낮은 전력소모, 빠른 데이터 접근 속도 등의 장점이 있다. 하지만, 일반 RAM 메모리에 비하여 상대적으로 느린 연산 특성을 고려하여 기존의 전통적인 인덱스 관리 기법을 개선할 필요가 있다. 이를 위하여, 본 논문은 CHC-Tree 라고 하는 압축된 핫-콜드 클러스터링에 기반하는 새로운 인덱스 관리 기법을 제안한다. CHC-Tree는 인덱스 노드를 핫-콜드 세그먼트로 분류하며, 인덱스 노드의 키와 포인터를 압축한다. 또한, 실험 결과를 통하여 기존의 B-Tree 기반의 인덱스 관리 기법보다 인덱스 검색 및 인덱스 수정 연산에서 더 우수함을 확인하였다.

1. 서 론

최근 각종 휴대형 소형 정보기기들이 대중화됨에 따라, 정보 저장용 미디어로 기존의 하드 디스크를 대체하여 플래시 메모리 저장 장치가 보편적으로 활용되게 되었다[1,2,5]. 그러나 플래시 메모리는 저장 장치로서 많은 장점을 가지고 있는 반면, 수명이 제한되고, 쓰기 속도가 느린 적잖은 단점도 가지고 있다. 이러한 단점들로 인하여, 기존의 인덱스 관리 기법들은 플래시 메모리에 쉽게 적용이 되지 못하였다. 따라서, 플래시 메모리의 제약점을 효과적으로 극복할 뿐 아니라, 그 고유의 장점을 효과적으로 활용하는 새로운 인덱스 관리 구조와 저장 시스템의 연구가 필요하다.

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.

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 엔트리를 넣는 방향으로 설계한다.[4]

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.

2. 본 론

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, 평균점유율(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 [3]에 구체적으로 잘 설명되어 있다.

본 연구에서는 인덱스 노드를 접근 패턴에 따라서 자주 수정되는 "핫-노드"와 드물게 수정되는 "콜드 노드"로 두 가지로 분류한다. 각 인덱스 노드는 이러한 핫-콜드 분류를 위하여 수정빈도시간을 저장하고 있다. 또한, 인덱스 영역을 핫 클러스터와 콜드 클러스터로 분류한다. 핫-클러스터는 핫-노드만을 포함하는 핫-세그먼트들로 구성되어 있다. 마찬가지로 콜드-클러스터는 콜드-노드를 포함하는 여러 개의 콜드-세그먼트로 클러스터링되어 있다. 본 연구에서는 핫-클러스터와 콜드-클러스터의 비율은 1:2로 설정하였는데, 이는 핫-클러스터의 크기는 콜드-클러스터의 반이면서, 두 배로 빈번히 수정된다는 의미이다. 개별 사용자 환경에 따라서 성능최적화를 위하여 두 클러스터 간의 비율은 다르게 조정 가능하다.

플래시 메모리는 쓰기 연산 전에 지우기 연산을 수행하여, 빈 세그먼트를 확보한 후 실제 쓰기 연산을 수행된다. 이때, 이러한 빈 세그먼트를 확보하기 위하여 세그먼트 단위로 클리닝이 수행된다. 만일, 세그먼트 클리닝을 수행하지 않으면, 플래시 메모리 상에서 조각난 데이터들이 여기 저기 불필요하게 떠다니면서 그 데이터가 속한 세그먼트들의 업데이트 횟수를 증가시키기 때문이다.

본 연구에서는 이러한 데이터의 접근 패턴을 분석한 후, 정적인 속성의 데이터는 정적인 콜드 세그먼트에 클러스터링하고, 빈번히 수정되는 동적인 속성의 데이터는 동적인 핫 세그먼트에 저장하여, 전체적으로 세그먼트 업데이트 횟수를 줄이고자 한다. 보통 데이터가 수정된 블록을 오염된(dirty) 블록이라고 하는데, 세그먼트 내에 더티블럭이 모이면 클리닝을 수행하게 된다. 비유를 하자면, 활동이 심한 개구쟁이들을 한 방에 넣고, 정적인 아이들을 다른 방에 분류하여 배치하면, 방 청소횟수를 줄일 수 있는 것과 같다.

여기서, 콜드 세그먼트내의 데이터는 내부적으로는 압축되어, 고가의 플래시 메모리의 저장 공간을 늘려줌과 동시에 세그먼트의 클리닝시 압축효과로 한번에 더 많은 더티 데이터를 클리닝하게 된다. 다만, 핫 세그먼트는 매우 빈번히 수정되어 콜드 세그먼트에 비하여 압축의 효용성이 낮으므로, 압축하지 않고 그대로 저장된다. 이러한 핫-콜드 압축 클러스터링을 통한 세그먼트 클리닝 감소로 인하여, 결과적으로 인덱스 연산의 성능이 향상되고, 더불어 플래시 메모리의 수명도 연장된다.

3. 성능 평가

본 시뮬레이션 수행시 CHC-Tree 기법(CHCTR)과 비교된 대상 색인은 가장 보편적으로 사용하는 B^+-Tree기법(BTR_N)이며, 전술한 바와 같이 최적성능이 나오는 트리 노드의 평균 점유율(69%)을 기본으로 하였다. 또한 트리가 거의 차 있는 최고밀도 저장의 경우(97%)의 B^+-Tree기법(BTR_F)도 포함하여, 총 세 가지 색인을 비교 시험하였다. 실험 도구는 CSIM 시뮬레이션 언어와 Microsoft Visual C++를 사용하였다. 실험이 수행된 하드웨어 환경은 펜티엄 쿼드CPU와 메인 메모리 2G, 하드디스크 500G이며, 운영체제는 윈도우 2003 서버를 사용하였다.

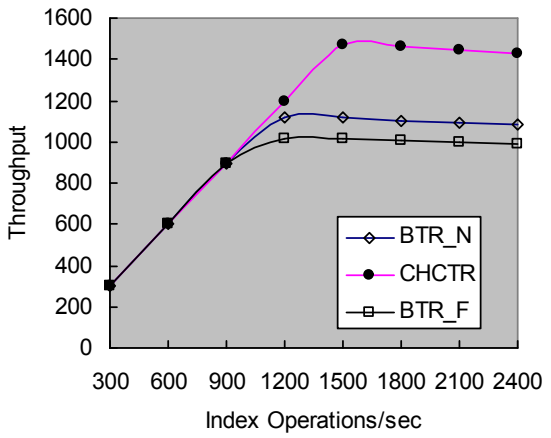
다음으로 플래시 메모리 데이터베이스를 위한 시뮬레이션 모듈이 필요한데, 사용된 모델은 CSIM에서 제공되는 폐쇄형 큐잉 모델(Closed Queuing Model)이다. 이 모델을 통하여 초당 일정한 수의 읽기 쓰기의 인덱스 연산을 생성하고, 이와 연관된 인덱스 트리가 검색 또는 수정되면서 발생한 시스템의 처리 시간과 성능을 측정하면 된다.

플래시 메모리 데이터베이스 운영 환경을 위한 시뮬레이션의 주요 성능 평가 지표는 시스템의 인덱스 연산에 대한 연산 처리치(throughput)와 응답시간(response time)이다. 이러한 연산 처리치는 초당 몇 개의 인덱스 연산이 처리되었는지를 의미하고, 응답시간은 인덱스 연산을 요청한 후 수행완료까지의 걸린 시간을 의미한다.

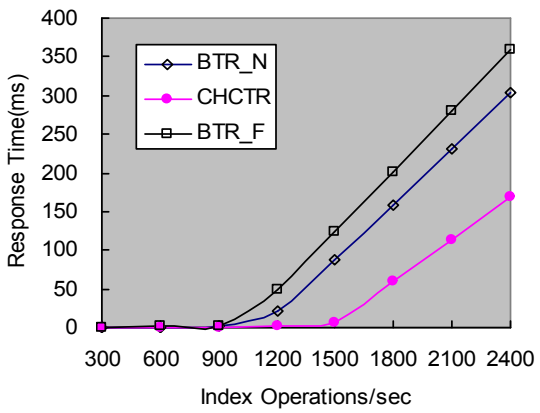
주요 시뮬레이션 파라미터는 초당 생성된 인덱스 연산의 수, 읽기 연산 수행시간, 쓰기 연산 수행 시간, 소거 연산 수행시간 등이다. 초당 생성된 인덱스 연산

의 수는 검색연산의 경우는 1500개에서부터 500개 단위로 5,000개까지 변화시켜 보았으며, 갱신 연산의 경우는 300개에서부터 300개 단위로 2,400개까지 변화시켜 보았다. 이는 모두 시뮬레이션 시스템에 가해지는 작업 부하를 의미한다. 플래시 메모리의 읽기 연산 수행 시간은 16㎲로 설정하였고, 쓰기 연산 수행 시간은 250㎲로 설정하였으며, 소거 시간은 세그먼트당 2㎳로 설정하였다. 각 연산 수행 시간은 제품 홈페이지[5]에서 제시한 자료이다. 그 외의 구체적인 설정 값은 아래 표와 같다.본 연구에서 핫 클러스터와 콜드 클러스터의 설정 비율은 전술한 데로 1:2이므로, 상대적인 hot-ratio 값은 33%로, hot-ratio값은 67%로 설정하였다. 또한, 전체 인덱스 접근 연산에서 검색 연산이 50%를 차지하고, 삽입 연산이 25%, 삭제연산이 25%로하여 구성된다.

본 연구의 시뮬레이션에서는 제안한CHCTR 색인을 기존의 BTR_N 및 BTR_F 색인과 비교 분석하였으며, 검색 및 갱신 부하에 따른 각 기법의 처리 성능과 응답 시간을 분석하였다. 또한, 전체 인덱스 연산 중 검색 연산은 50%, 삽입 및 삭제 연산은 각각 25%로 하여 기본구성비로 실험하였다.



[그림 1] Index Operation Throughput



[그림 2] Average Response Time

그림1은 초당 발생된 인덱스 연산의 수(num_OPs)의 증가에 따른 각 기법의 처리치의 그래프이며, 그림2는 평균 응답시간 그래프이다. 그림2에서 보면, 발생된 초당 인덱스 연산의 수가 늘어날수록 점차로 응답시간이 점차로 증가함을 알 수 있다. 이는 주로 작업 부하 증가에 따라서, 인덱스 연산의 집중에 의한 것이다. 또한, 그림1에서 보는 바와 같이, 전반적인 인덱스 연산의 처리 성능을 측정한 결과, CHCTR 이 BTR_N 과 BTR_F 보다 더 높게 나타났다.그림1에서 보면, 초당 인덱스 연산의 수가 대략 900개까지는 기법들간의 별 성능 차이를 보이지 않는다. 이는 인덱스 연산의 부하를 각 기법들이 충분히 수용 가능함을 의미하며, 그림2의 응답시간을 분석해 보면, 900개 까지는 데이터 검색에 꼭 필요한 처리 시간이 소요될 뿐, 그 외의 지연 시간이 없어서, 성능 차이가 나타나지 않는다. 그러나 인덱스 연산의 수가 1,200~1,500개를 넘으면서 각 기법의 성능은 낮아지면서, 각 기법 별로 성능 차이가 나기 시작한다. 이는 초당 인덱스 연산수의 증가에 의한 인덱스 접근 적체가 성능에 영향을 크게 미치는 주요 요소임을 의미하며, 이 수치 이상으로 인덱스 접근 연산을 활성화시키는 것이 성능 향상에 도움이 되지 않음을 의미한다. 여기서 플래시 메모리상의 인덱스 노드 검색은 상대적으로 고속이므로 아직 이 구간에서는 성능 저하의 원인이 아니며, 인덱스의 삽입 및 삭제 연산에 의하여 야기되는 플래시 메모리의 느린 저장(지우기,쓰기) 속도가 주요 원인이다.

그림1의 그래프에서 보면 본 CHCTR 기법이 느린 저장 연산을 최소화하여 기존 기법보다 처리 성능이 우세하다고는 하지만, 인덱스 연산 부하가1,500를 넘으면, 과도한 부하로 인하여, 서서히 성능이 저하됨을 알 수 있었다. 그림2에서 인덱스 접근 부하가 증가하여도, 900이하의 구간에서는 별로 응답지연이 발생하지 않으나, 이 구간을 넘게 되면 적체현상이 발생하여 평균응답시간이 빠르게 증가함을 알 수 있었다. 전체 구간에서 CHCTR 의 커브가 기존 기법에 비하여 완만하면서도 더 늦게 증가하면서 상대적으로 빠른 응답속도를 보였다. CHCTR은 인덱스 노드의 부하가 초당 2,400일 때, 기존 기법대비 23%의 개선된 처리 결과를 보였는데, 이러한 성능차이로 압축 핫-콜드 클러스터링의 효과를 확인할 수 있었다.

보다 세부적으로 CHCTR 기법의 효과를 분석하기 위하여, 혼합된 인덱스 연산을 검색 연산과 업데이트 연

산(insert and delete)으로 다시 분리하여 실험해 보았다. 실험결과 BTR_F 가 가장 우수한 성능을 나타내었는데, 그 이유는 노드당 평균 점유율이 97%로서 세 기법 중 가장 높으므로, 한번에 가장 많이 읽을 수 있기 때문이다. 즉, BTR_F는 최고의 저장효율을 가지고 있는데, 이는 결과적으로 인덱스 트리의 깊이를 짧게 만들어서 노드 접근 횟수가 줄었기 때문으로 분석된다. BTR_F는 5000개의 인덱스 검색 부하 시에 본 CHCTR 기법의 보다 6% 더 우수한 성능을 보였다. 업데이트 연산은 인덱스 검색 연산에 비하여 상대적으로 매우 인텐시브한 접근 특성을 가진다. 이유는 업데이트 연산에는 매우 느린 지우기와 쓰기를 동반하는 삽입 연산이 다수 포함되어 있기 때문이다. 이러한 인텐시브 저장 연산이 수행되는 실험 환경에서는 당연히 본 CHCTR이 전체 구간에서 더 높은 성능을 보인다. 실험결과 특히, 900개 이상의 초당 인덱스 부하 상태에서 기존 기법에 비하여 26%정도의 훨씬 더 높은 성능을 보였다. 이러한 성능의 우위는 압축된 콜드 클러스터에서 지우기 및 쓰기 연산이 줄어들어서, 결과적으로 전체적인 인덱스 노드 업데이트 연산수가 감소하였기 때문으로 분석된다. 이 사실은 압축 핫-콜드 클러스터링으로 인하여 역시 응답시간도 더 개선되었음을 확인할 수 있었다. 실제로, CHCTR의 평균응답시간은 기존 기법에 비하여 최대 38%가 개선되었다.

4. 결 론

본 논문에서는 플래시 메모리 기반 저장 시스템의 인덱스 연산 성능을 높이기 위하여 CHC-Tree 기반의 새로운 인덱스 관리 기법을 제안하였다. 기존의 하드 디스크 및 메모리를 위한 B-Tree 기반 인덱스 기법을 개선하여, 제안 기법은 트리의 인덱스 노드를 핫-콜드 클러스터로 분리하여 다른 클러스터로 저장하였고, 키와 포인터를 재배열하여 오프셋 압축함으로써 플래시 메모리의 저장 연산 수를 대폭 줄여서, 전반적인 저장성능을 높일 수 있었다. 또한, 성능 확인을 위하여 큐잉방식의 시뮬레이션 모델을 제시하였다. 특히, 부하가 심한 저장 환경에서 실험한 결과, 전체적인 인덱스 처리 성능이 기존 기법에 비하여 23% 이상 개선됨을 확인하였다. 본 CHC-트리 기반 인덱스 관리 기법은 플래시 메모리나 SSD가 장착된 최신 데스크톱 컴퓨터 및 중대형 데이터 서버 등에서 폭넓게 활용 가능하다.

참고 문헌

[1] 김태웅, 류준길, 박찬익, “리눅스 환경에서 Solid-State Disk 성능 최적화를 위한 디스크 입출력요구 변환 계층”, 한국차세대컴퓨팅학회 논문지, 49-57, 2009년 6월.

[2] 정승국, 고대식, 차세대 스토리지 SSD 기술 동향, ETRI 주간기술동향 통권 1369호, 2008.10.

[3] 황규영, 홍의경, 음두현, 박영철, 김진호, "데이터베이스 시스", 생능출판사, 2000

[4] 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.

[5] Samsung, "what is NAND Flash based SSD?", http://www.samsung.com/global/business/semiconductor/products/flash/Products_FlashSSD.html , 2009.