Journal of the Korea Academia-Industrial cooperation Society
Vol. 17, No. 7 pp. 293-300, 2016

http://dx.doi.org/10.5762/KAIS.2016.17.7.293
ISSN 1975-4701 / eISSN 2288-4688

대용량 압축 데이터베이스를 위한 비대칭 색인 관리 기법

변시우, 장석우*
안양대학교 디지털미디어학과

Asymmetric Index Management Scheme for High-capacity Compressed Databases

Si-Woo Byun, Seok-Woo Jang*
Division of Digital Media, Anyang University

요 약 전통적인 기존의 데이터베이스는 높은 성능을 얻기 위하여 느린 하드 디스크에서 관련된 레코드가 연속적으로 저장되어 있는 레코드 기반 모델을 활용한다. 그러나 읽기 집중적인 데이터 분석 시스템을 위해서는 컬럼 기반 압축 데이터베이스가 월등한 판독 성능으로 인하여 더 적합한 모델이 되고 있으며, 최근 플래시 메모리 SSD가 고속 분석용 시스템에 적합한 저장 매체로 선호되고 있다.
본 논문에서는 세로로 저장하는 컬럼 기반 스토리지 모델을 소개하고, 대용량 데이터웨어하우스 시스템을 위한 새로운 인덱스와 데이터 관리 기법을 제안한다. 제안된 인덱스 관리 기법은 두 개의 인덱스를 사용하는 비대칭 인덱스 이중화이며, 갱신용 마스터 인덱스와 판독용 컴팩트 인덱스를 활용하여 특히 읽기가 집중된 빅 데이터베이스에서 우수한 검색 성능을 얻는다. 그리고 본 데이터 관리 기법은 관련된 컬럼 압축과 두 개의 플래시 메모리 SSD를 이중화하여 높은 판독 성능과 처리 안정성에 도움을 준다. 고부하 워크로드 조건의 성능 평가 결과를 기반으로, 본 데이터 관리 기법이 기존 기법보다 검색 처리 및 응답 시간 측면에서 더 우수함을 보이고자 한다.

Abstract Traditional databases exploit a record-based model, where the attributes of a record are placed contiguously in a slow hard disk to achieve high performance. On the other hand, for read-intensive data analysis systems, the column-based compressed database has become a proper model because of its superior read performance. Currently, flash memory SSD is largely recognized as the preferred storage media for high-speed analysis systems.
This paper introduces a compressed column-storage model and proposes a new index and its data management scheme for a high-capacity data warehouse system. The proposed index management scheme is based on the asymmetric index duplication and achieves superior search performance using the master index and compact index, particularly for large read-mostly databases.
In addition, the data management scheme contributes to the read performance and high reliability by compressing the related columns and replicating them in two mirrored SSD. Based on the results of the performance evaluation under the high workload conditions, the data management scheme outperforms the traditional scheme in terms of the search throughput and response time.

Keywords : Compressed database, Asymmetric index duplication, Tree index search, Flash memory

1. 서론

빅 데이터 분석 분야에서 효과적인 컬럼-기반 저장소 [1]는 세로의 필드 단위로 분리, 저장, 검색하는 새로운 방법이다. 판독 위주의 데이터베이스 환경에서는 특정 컬럼에 데이터가 군집되어 있는 컬럼 저장 구조가 효율

본 논문은 안양대학교 2015년 연구년 기간중 연구되었음.
*Corresponding Author : Seok-Woo Jang(Anyang Univ.)
Tel: +82-31-467-0842 email: swjang@anayng.ac.kr
Received April 19, 2016 Revised (1st May 2, 2016, 2nd May 23, 2016)
Accepted July 7, 2016 Published July 31, 2016

적이며, C-Store[2]가 알려져 있다.

또한 최근에는 기업의 중대형 데이터 서버나 데스크톱등 에서 전통적인 HDD에서 고속 SSD 저장 시스템[3]으로의 대체이동이 가속화되었다.(그림1)

기본적으로 세로-저장 스토리지는 테이블 저장시 세로로 분리 저장하므로, 유사성이 높은 데이터들이 서로 군집되어, 압축, 저장, 검색에 매우 효과적인 구조이다. 과거에는 하드 디스크가 대세였고 스토리지도 이에 적응하였으므로, 하드 디스크의 접근 특성을 고려하였고, 당연히 저장 연산 처리에 가장 효율적인 방식인 가로-기반 저장 방식으로 설계되었다.

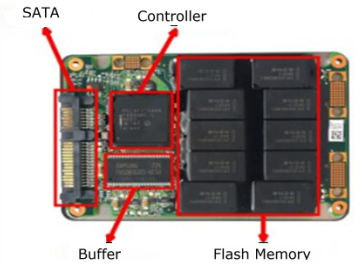

Fig. 1. An architecture of SSD

그러나 SSD가 속도와 더불어 가격 경쟁력을 갖춤에 따라서, 신속한 대용량 데이터 검색이 가능한 효율적인 인덱스 접근 및 관리 기법이 필요하다.

2. 압축 데이터와 기존 인덱스 구조

대용량 압축데이터베이스는 압축, 저장, 검색에 매우 효과적인 구조이다.[2] 기존 인덱스 구조를 살펴보면, 하드 디스크 기반 색인 저장 기법은 검색 시에 디스크 접근을 최소화하기 위하여 노드의 크기를 디스크 페이지와 같은 크기나 그 배수로 설정하고, 많은 엔트리를 한 노드에 넣어야 유리하였다. 검색 성능 저하보다는 디스크 헤드 무빙이나 회전 지연 및 섹터 접근에 의한 성능 저하가 더 크기 때문에 한 노드에 많은 엔트리를 넣는 방향으로 설계한다[4]. 이런 용도로 B-Tree 계열의 인덱스가 사용되었다.

그리고 메모리 기반 색인에서는 디스크 기반 색인과는 달리 노드의 용량을 변화시켜 트리의 깊이와 비교횟수를 조절하여 성능 향상이 가능하다[5].

그러나 기존의 인덱스 관리 구조는 전통적인 스토리지에 초점이 맞추어져 있으므로, 최신 플래시 메모리 스토리지나 읽기가 대부분인 대용량 데이터웨어하우스의 구조에 맞게 개선되어야 한다. 즉, 차세대 플래시 기반의 압축에 효과적인 인덱스 기법과 대용량 검색의 고속화 및 이중화 기법이 필요하다.

3. 새로운 색인 관리 방식의 제안

3.1 기존 인덱스 관리 기법의 특성

보편적으로 안전성을 높이기 위하여 두 디스크를 동일한 내용으로 이중 저장하는 미러링 방식을 사용한다. 그런데, 데이터에 대해서는 비용을 더 투자해서라도 이중화를 하여 데이터의 안정성을 증대시키려고 하는 반면, 이러한 데이터를 사용하기 전에 먼저 접근해야 하는 인덱스에 대해서는 굳이 이중화를 하지 않고 있다. 그 이유는 인덱스는 파괴되더라도 원래 데이터로부터 다시 만들 수 있기 때문이다.

그러나 최근 대용량 데이터가 축적이 늘어나고, 빅 데이터, SNS, 클라우드 환경에서 엄청난 속도로 데이터가 증가하고 있다. 즉, 과거와는 달리 이제 대용량 데이터베이스의 인덱스를 다시 생성하기에는 이제는 너무 많은 시간이 필요하고, 그 지연시간동안 운영은 겨우 되더라도 해당 인덱스가 없으므로 그 검색 성능은 매우 저하되게 된다. 본 연구의 목표는 이러한 인덱스의 기존 관리 개념을 탈피하여, 새로운 방식으로 이중화 저장 관리하여 저비용으로 안정성과 검색성을 높이고자 한다.

또한, 데이터에 접근하기 위해서는 그전에 먼저 인덱스에 접근허가를 얻기 위하여 잠금 락을 설정하게 되면, 다른 프로세스는 이 접근 권한을 얻을 때까지 대기 큐에서 기다려야 한다. 만일, 이러한 대기 상황에서, 다른 검색 인덱스를 사용하게 되면, 대기 지연 시간을 대폭 줄일 수 있다. 특히 데이터가 미러링 되어 중복되어 있는 경우에는 두 경로에서 데이터를 접근할 수 있게 되므로 그 효율을 더 높아질 수 있다. 본 연구에서는 기존의 단일 인덱스가 아닌 서로 다른 두 개의 인덱스 개념을 도입하여 안정성과 성능을 개선하게 된다. 본 제안 기법에서는 디스크 기반 및 메인 메모리 기반 색인 중에서 가장 지속적으로 사용되며 공통적인 모태 속성을 가진 B-Tree 색인을 근간으로 하여 보편성을 살리되, 대용량 압축 데

이터베이스 및 고속 플래시 메모리에 효과적인 색인 관리 및 데이터 검색 기법을 제안한다.

인덱스 관점에서 살펴보며, 메모리 데이터베이스에서 B-Tree도 많이 사용된다.[5] 본 연구에서는 일반 데이터베이스의 기본 색인인 B^+-Tree를 본 컬럼-지향 플래시 메모리 저장 환경에 적합하게 개선하여, 검색 속도를 높이고, 저장 안정성을 개선할 수 있는 색인 이중화 기법인 AID-Tree (Asymmetric Index Duplication Tree) 및 관리 기법을 제안한다. AID-Tree 기법은 이중화와 더불어 B-Tree의 비활용 영역을 제거한 압축과 내장된 압축 인덱스 기반의 고속 검색을 활용한다.

B^+-Tree에서 시뮬레이션을 수행하여 분석한 결과 69%정도 차 있을 때가 성능 상 최적이라고 한다.[6] 본 제안기법에서는 원래의 빈공간이 유지되고 있는 마스터 인덱스를 유지하되, 고속 검색을 위하여 30%의 용량이 줄어든 컴팩트 인덱스를 하나 더 만들어 두 개를 활용하고자 한다. 삽입, 삭제시 발생하는 리벨런싱 효율을 위하여, 마스터 인덱스를 기존처럼 유지하면서, 빈공간이 제거된 컴팩트 인덱스를 통하여 트리 순회의 길이를 대폭 줄여서 인덱스의 검색 깊이를 줄이므로 검색 속도 개선이 가능하다. 또한, 피크타임시 인덱스의 부하를 줄여서 전체 성능과 응답성을 높일 수 있다.

분석 작업이 대부분인 데이터웨어하우스 환경에서는 대용량 데이터베이스에 연산부하가 심하여 피크부하가 빈번히 걸리지만, 사용자가 매우 적기 때문에 많은 유휴시간에 비동기화가 충분히 가능하다. 위와 같은 이유로 비대칭 인덱스 이중화는 검색 성능의 향상과 더불어 안정성도 함께 증대시킬 수 있다.

기존에 읽기 쓰기 연산의 성능 향상을 위한 인덱스 개선 기법으로서 FD-Tree[7], HH-Index[8], BFTL[9] 등이 제안되었다. FD-Tree는 인덱스 트리에 로그 연산을 활용한 자투리 관리 기법이다. 최상단에 위치한 조그만 B-Tree인 헤드 트리(Head-Tree)가 있고, 성능을 향상시키기 위하여 펜스(Fence)라고 하는 포인터를 저장하고 관리한다. 이 기법은 쓰기-최적화를 목적으로 설계되었으며, 특히, 검색 연산이 점점 더 많은 노드를 방문하게 되더라도 임의 쓰기 연산은 조그만 헤드 트리에 한정되도록 되어 있는 점이 특징이다.

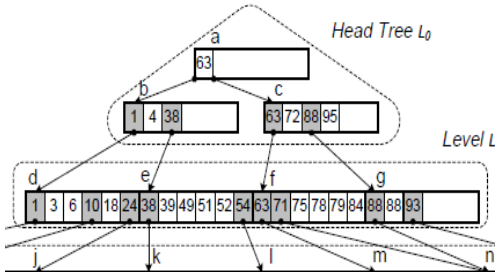

Fig. 2. An example FD-tree

FD-Tree와 본 제안 기법이 모두 인덱스 성능 향상을 목적으로 하지만, FD-Tree는 쓰기 영역을 군집화 시키는 방법으로 접근하지만, 본 기법은 군집화를 통한 갱신 효율 향상이 목적이 아니라 B^+-Tree의 검색 효율 향상으로 설계된 점이 다르다. 특히, 본 제안 기법은 일반 데이터베이스가 아닌 읽기가 압도적으로 많은 분석용 데이터웨어하우스를 위하여 판독-연산 최적화에 중점을 두고 있는 점이 다르다. 또한, 인덱스 관리 측면에서, FD-Tree는 루트 노드와 중간 노드를 수정하지만, 본 기법은 리프 노드를 수정하는 점도 다르다. 즉, 대용량의 읽기가 매우 많은 데이터 분석 분야에서는 본 기법이 더 적합하다.

또한, 최근에 제안된 기법으로서 플래시 메모리를 위한 인덱스 성능 향상 기법으로서 하이브리드 해시 인덱스[8]가 있다. 이 기법은 해시 인덱스의 제자리 갱신시 발생하는 성능 저하를 버켓별 오버플로우 영역을 활용하는데, 이는 기존의 선형 해시 인덱스와 같이 분할 시점을 지연시키고자 하는 것이다. 또한 다중 동적 해시 인덱스는 멀티 레벨 구조를 활용하여 쓰기 연산을 줄였지만, 저장 메모리 버퍼를 사용하여 갑작이 정전이 되면 자료가 손실될 수도 있다. 본 기법은 정전시 해시 관리로 인한 손실이 없으며, 두 종류의 인덱스를 이중화 시켜서 사용하므로 더 안정적이라고 할 수 있다. 또한, 갱신 연산 보다는 검색 위주의 환경에서 더 유리한 점이 다르다.

또한, 플래시 기반 인덱스 성능 향상 기법으로 BFTL이 있다. BFTL 기법은 B-Tree 인덱스에 접근시 아주 세밀하게 갱신 연산을 제어하여 플래시 메모리상의 중복된 쓰기 연산을 줄이는 기법이다. BFTL은 조그만 예약 버퍼(reservation buffer)와 노드 변환 테이블(node translation table)로 운영된다. 응용프로그램이 삽입 삭제를 시도하면, 새로 생성된 레코드는 임시로 예약 버퍼에 들어가서 중복된 쓰기 여부를 검사하여 제거되게 된다. 예약버퍼는 적당한 양의 레코드를 홀드하고 있다가

주기적으로 플러시 된다. 노드 변환 테이블은 B-Tree 노드와 인덱스 노드의 논리적 섹터 주소로서 이전보다 적은 섹터수로 합쳐져서 효율적으로 관리 저장된다.

BFTL이 쓰기 연산에서 개선된 성능을 확보할 수 있지만, 예약 버퍼와 노드 변환 테이블이 하드웨어적으로 추가해야 하는 부담이 있으므로, 소프트웨어적인 본 기법에 비하여 현실적 수용이 어려운 단점이 있는 점이 다르다. 또한, 빈번한 예약버퍼 접근과 변환 테이블 접근으로 검색 부담이 있다. BFTL이 B-Tree의 추가적인 하드웨어를 통하여 쓰기 횟수를 줄이는 반면, 본 기법은 인덱스 이중화과 패킹에 의하여 부하를 소프트웨어적으로 줄이는 점이 다르며, 검색 비용에서 측면에서 유리하다.

3.2 비대칭 이중화 인덱스 관리 기법

3.2.1 컬럼별 인덱스의 비동기 비대칭 이중화

컬럼-지향 데이터베이스의 성능 개선을 위하여 고려하여야 할 또 다른 중요한 사항은 CPU 및 저장 부하의 변동성이다. 특히, 일반 온라인 데이터베이스에 비하여 컬럼-지향 데이터베이스의 데이터웨어하우스는 소수의 인원이 집중적인 피크 로드를 발생했다가, 유휴 시는 CPU 로드가 매우 급격히 감소하므로, 이 변동성이 훨씬 더 크다. 빈번히 접근되는 인덱스를 피크로드시에 실시간으로 바로 동기화시키면 부하가 더욱 가중되므로, 이것 보다는 마스터 인덱스에만 반영하고, 유휴시 동기화가 유리하다.

특히, 부하가 많이 발생할 때 인덱스에도 판독 부하나 동시성 제어를 위한 락이 많이 걸릴 수 있으므로, 두 개의 인덱스를 사용하면, 부하를 많이 감소시킬 수 있다. 더욱이, 대량의 업로드 등으로, 부하가 급증하고, 인덱스에도 수정부하가 급증하고 있는 경우, 읽기 전용으로 슬레이브 인덱스를 사용하여, 대기하고 있는 응용프로그램의 진행을 도와준다면, 결과적으로 시스템 부하의 누적을 줄여서, 시스템의 성능과 안정성 향상에 기여할 수 있게 된다.

3.2.2 컬럼 데이터의 압축 동기 이중화

이중화 원리는 데이터를 압축한 후 두 카피가 동일한 데이터를 실시간으로 동일하게 유지하는 것으로, RAID-1 의 미러링의 개선기법으로 볼 수 있다. 그러나 데이터를 하나씩 저장하지 않고, 1000개 이상의 컬럼-데이터를 모아서 하나의 컬럼-세그먼트 단위로 압축 저장하며, 필요시 압축이 복원되어 판독이 된다는 점이 차이점이다.

물론, 안정성 강화를 위하여 미러링과 같이 분리된 다른 드라이브에 저장되며, 두 드라이브에 모두 안전하게 저장되어야만 전체 저장연산이 완료되는 동기화방식이다. 여기서 인덱스 이중화와는 달리 비동기 저장을 하면 속도는 더욱 개선되겠지만, 최소한 데이터만큼은 완벽하게 안정성이 보장되어야 하므로, 동기화를 선택한다.

3.2.3 마스터 인덱스와 컴팩트 인덱스의 관리

본 비대칭 인덱스 이중화 방법은 원래 기존이 인덱스 역할인 마스터 인덱스와 빈공간이 제거된 압축된 형태의 컴팩트 인덱스의 두 버전을 운영한다. 시스템의 부하가 낮은 경우에는 그냥 마스터 인덱스를 접근한 후 하부의 컬럼 데이터를 읽고 쓰면 되고 정상적인 데이터베이스 성능이 유지된다. 반면에, 대용량 고속 데이터웨어하우스 환경의 특성상 고부하의 검색, 분석, 작업등의 이유로 데이터베이스가 busy인 상태가 되면, 검색에 매우 유리한 컴팩트 인덱스를 집중적으로 사용되게 한다.

특히, 컬럼-지향 데이터베이스의 특성상 과부하시 데이터의 복원속도가 더 떨어져서 전체 시스템이 느려지는 상황이 발생하는데, 이 경우에도 컴팩트 인덱스 버전을 판독하여 실체화 성능을 높일 수 있다. 또한, 데이터 업데이트 작업시 마스터 인덱스의 노드도 갱신될 경우에 컴팩트 인덱스가 동기화 이전의 데이터를 계속 연계시켜주면 검색 서비스가 지연되지 않으므로 성능이 향상된다. 즉, 본 기법은 컬럼-지향 스토리지로서의 검색 성능 개선에서도 기여하며, 갱신 작업 시에도 지연 없이 다른 검색 연산을 수행 할 수 있으며, 컴팩트한 인덱스 크기로, CPU자원 및 스토리지 자원 효율도 높다. 다만, 인덱스를 하나 더 저장해야 하므로, 좀 더 많은 저장 영역을 차지하지만, 그 크기는 실험결과 수백 메가바이트에 불과하므로, 조그만 추가 부담에 비하여 성능 및 안정성 개선효과는 비교할 수 없을 정도로 크다.

또한, 전술한 바와 같이, 컬럼-데이터베이스에서 컬럼별로 데이터를 모아 붙인 적당한 길이의 클러스터별로 압축해야 하는데, 1만 건 이상의 단위로 모아서 압축한다. 보통 1만 건은 일반 가로-지향 데이터베이스에서는 압축이 잘 안되어 큰 용량일 수 있으나, 컬럼-데이터베이스에서 컬럼별로 저장하므로, 압축률이 매우 높다. 물론, 이러한 고효율 압축을 잘 활용한 것이 컬럼-데이터베이

스의 기본 발상이며 큰 장점이기도 하다. 본 연구에서는 플래시 메모리를 통한 검색 고속화와 공간효율을 위하여 중간 노드는 빈 공간 압축하며, 컬럼 데이터가 들어 있는 리프 노드는 빈 공간 압축과 더불어 세그먼트로 나누어 압축한다. 본 실험에 사용된 압축 알고리즘은 Lha나 Gzip 유틸리티의 기반이 되는 lzo 압축[10]이다. 그림3은 본 연구에서 컬럼-지향데이터베이스를 위하여 검색 성능을 개선 제안한 비대칭 인덱스 이중화 (AID index) 기법의 저장 구조이다.

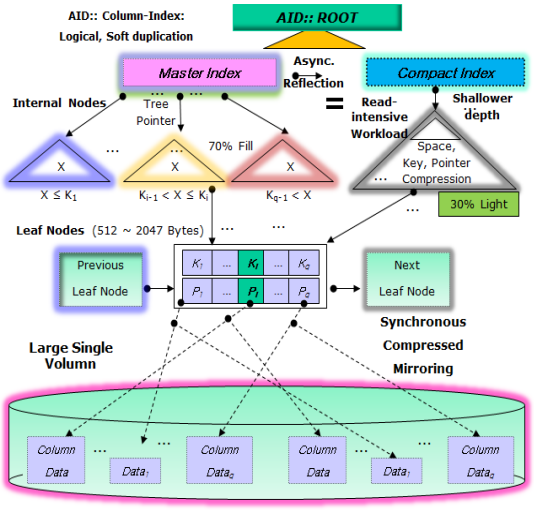

Fig. 3. A structure of AID Index

위에서 컴팩트 노드 압축은 일반적인 압축 알고리즘을 적용하되, 압축 효율을 높이기 위하여, 키 부분과 포인터 부분을 분리하여 압축한다. 다만, 압축의 부담인 압축 시간이 요구되는데, 실제로 CPU와 RAM의 접근 속도가 플래시 메모리의 접근 속도 보다 월등하여 압축시 시간 부담이 크지 않았다. 압축된 노드를 검색할 경우에도 약간의 복원 시간이 필요하지만, 복원 시간이 크지 않아서 검색 성능에 별로 영향을 미치지 않았다.

그림에서 중간노드는 루트 노드에서 포인팅 되어서 다시 다수의 중간노드로 분기될 수 있으며, 최종적으로 리프 노드를 포인팅하게 된다. 본 제안 기법의 중간노드는 노드 구조를 참조하였으나, 비활용 여유 영역을 캐싱이 아닌 키와 데이터 영역으로 빈틈없이 확장하여 사용한다는 점에서 크게 다르다. 본 기법의 중간 노드 탐색 방법은 기존 B^+-Tree와 같이 루트 노드로부터 킷값을 좌우 범위를 비교하여 킷값의 범위를 아래로 좁혀가며 최종적으로 리프 노드까지 인덱스 트리를 찾아 가는 방식이다. 따라서 플래시 메모리 접근 I/O가 대폭 감소함과 동시에 압축 부하도 크게 감소하여 컬럼-지향 데이터베이스의 검색 속도와 응답시간이 자연적으로 개선되게 된다.

4. AID 색인 관리 기법의 성능 분석

4.1 성능 비교 기법의 분류

성능 테스트에서는 제안한 색인 관리 기법을 다른 3가지 기법과 비교 분석하였다. 또한, 포괄적인 성능 분석을 위하여 인덱스 관리 기법과 밀접하게 연계되는 데이터 저장소를 포함하여 측정되었다. 즉, 아래와 같이 명명된 총 4가지 방식의 성능이 분석되었다.

- NoComp 기법: 사용되는 B-Tree 인덱스와 비압축 데이터 저장 방식이다.
- SingleComp기법: NoComp와 같이 B-Tree 인덱스를 사용하나, 컬럼별로 데이터를 압축 저장 하는 방식이다.
- MirrorComp 기법: SingleComp와 같으나, 데이터를 동일한 저장장치에 물리적으로 중복한 안정적 방식이다.
- AID-MrComp 기법: MirrorComp와 같으나, 제안한 AID 기법을 적용하여, Master Index와 Compact Index를 활용하여, 안정성과 검색 성능을 높인 방식이다.

4.2 실험 환경과 실험용 인덱스의 구성

하드웨어 환경은 옥타코어 2.7Ghz에 RAM 8G, 64bit 윈도우즈7 Ultimate을 사용하였다. 또한, 총 10억 건의 데이터를 대상으로 하고, 분석용으로 CSimAPI[11]가 사용되었다. 실험용 트리는 효율을 위하여 한 노드의 크기는 한 페이지의 크기인 512B와 같도록 하였다. 포인터의 크기는 일반적인 포인터 크기인 4바이트에 플래시 메모리 식별자 1바이트를 추가하여 5바이트로 설정하였다. 또한, 키필드의 크기도 10바이트로 설정하였는데, 압축을 하면, 일반적인 정수, 실수 등의 데이터 형을 포함하여 키워드 정도의 문자열을 포함할 수 있는 크기이다.

Table 1. A capacity planning of index node

Node Type	Level	nodes	keys	pointers
root	0	1	22	23
internal	1	23	506	529
internal	2	529	11,638	12,167
leaf	3	12,167	279,841	279,841
All	-	12,720	292,007	292,560

인덱스 트리의 크기는 루트 노드부터 시작하여 리프 노드까지 구성하여 산정해 보았다. 먼저 중간 노드의 크기는 한 페이지의 크기 이하로 할 때, 몇 개의 하위 노드를 포인팅 가능한지, 노드 포인터의 개수를 계산하였다. 계산 결과를 바탕으로 구성하면 아래의 표1과 같다. 이 인덱스 트리에는 총 12,720 개의 노드가 존재하며, 총 6.51 메가바이트의 저장 공간이 필요하다. 또한, 이 색인이 포인트 하는 데이터는 리프 노드의 포인터 개수인 279,841개가 되니, 약 28만개의 컬럼 세그먼트를 식별할 수 있게 된다. 또한, 한 컬럼 세그먼트에 1만개의 압축된 컬럼데이터가 저장되므로, 28억개의 컬럼데이터에 접근할 수 있으며, 실험 범위의 분석데이터를 지원할 충분한 크기이다. 실험 범위에서 한 개의 컬럼 인덱스의 6.51메가바이트이고, 대용량 압축 데이터베이스 전체에 지원할 인덱스 개수가 백 개 정도면 충분하므로, 총 컬럼에 대한 인덱스가 점유하는 공간은 최대 700메가바이트 이하다.

4.3 실험 결과 및 비교 분석

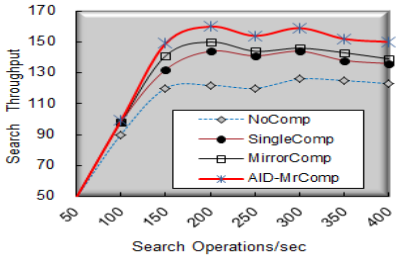

Fig. 4. Throughput of Column Searches

그림4는 초당 처리치를 표시한 그래프인데, 실험결과 AID-MrComp, MirrorComp, SingleComp, NoComp 순으로 우수하게 나타났다. 즉 제안 기법인 AID-MrComp가 가장 높으며, 압축하지 않은 일반적인 저장 방식인 NoComp 이 가장 낮았다. 또한, SingleComp보다 MirrorComp이 유사한 패턴을 보이나 더 높게 나온 것은 둘 다 압축저장을 하여 NoComp보다는 좋아 보이나, MirrorComp가 리프노드에 압축된 데이터를 미러링하여, 읽기 작업 부하를 경감시킨 효과이다.

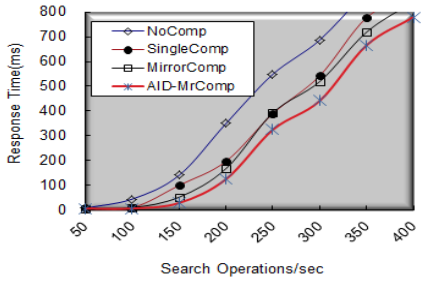

Fig. 5. Response Time of Column Searches

그림5의 응답시간을 분석해 보면, 110개 까지는 컬럼 데이터 검색에 필요한 처리 시간이 소요될 뿐, 그 외의 지연 시간이 없어, 차이가 나타나지 않는다.

전체적으로 작은 부하의 초반 워크로드에서는 미세하게 제안기법이 우세하지만, 중후반 워크 로드부터는 AID-MrComp가 효과가 더욱 우세하게 나타났다. 즉, AID-MrComp가 인덱스 이중화 효과 및 데이터 읽기 부하 감내 능력에서 기존 기법보다 더 우월하므로, 성능이 더 높은 것으로 평가한다. 그러나 210개를 넘으면서 서서히 차이가 발생하기 시작하며, 초당 약 160 개 정도가 최고 부하이다. 하지만, 부하가 심할수록 본 기법이 효율이 더 높다. 즉, 기존의 기법에 비하여 각각 약 10~20% 정도 더 높게 나타났다. 그 이유는 리프노드의 컬럼 데이터를 압축함으로써 읽기모드에서 검색 블록수를 줄이고, 해당 검색 영역에 인덱스를 이중화하여 대용량 데이터 접근시 인덱스 접근 부하를 분산시켜서 목표한 리프노드에 신속하게 접근시킬 수 있었기 때문이다.

기존 하드디스크와는 달리 플래시 메모리의 인덱스는 비인접 영역에도 신속한 랜덤 접근 가능하므로 이중화시에도 하드디스크 보다 더 효율적이고 빠르다. 결과적으로, 이러한 인덱스 이중화 효과로서 본 기법이 MirrorComp 보다 30%정도의 개선된 응답 속도를 나타냈다. 즉, 검색에 수반된 컬럼-인지 인덱스 이중화 효과가 처리 시간에 개선 효과를 주고, 이는 처리 성능치를 개선시킴을 확인할 수 있었다.

최근 중요한 정보를 보관하는 고성능 데이터서버에서 이중화 부하는 저장시스템 안정화를 위한 필연적인 작업

부하이므로 감내해야 한다. 또한, 일반 데이터베이스에서도 갱신 연산은 보통 전체의 10%이하이며, 특히 이미 구축된 데이터를 기반으로 검색 및 분석 업무가 대부분인 데이터웨어하우스 분야에서는 초기의 벌크로드 작업, 주기적인 데이터 업로드작업 외에는 상대적으로 발생빈도가 극히 낮다. 특히, 벌크로드와 업데이트는 일상적인 시간이 아닌 유휴시간에 이루어지므로 실질적인 성능 손실을 사용자가 체감하기는 더더욱 어렵다. 따라서 분석용 데이터베이스 환경에서 매우 빈번한 검색 연산과 매우 드믄 갱신연산의 비대칭적 비중을 고려할 때, AID 기법은 기존 기법에 비하여 소량의 자원추가로 우수한 성능을 얻을 수 있다고 판단된다.

5. 결 론

먼저, 본 논문에서는 고속 검색이 빈번한 대용량 압축 데이터베이스를 위한 컬럼-기반 데이터 저장 관리 기술을 기존의 가로-지향형 일반 데이터베이스와 비교 분석하였다. 또한, 최근 대중화 되고 있는 플래시 메모리기반 SSD 저장 장치를 활용하면 압축률이 높고 검색이 많은 환경에서 효과적임을 보였다.

다음으로, 이러한 분석용 데이터웨어하우스 분야에서 검색 성능과 저장 안전성을 향상 시킬 수 있는 비대칭 인덱스 이중화 기반의 새로운 데이터 저장 기법을 제안하였다. 제안 기법은 기준이 되는 마스터 인덱스와 트리 검색이 빠른 컴팩트 인덱스를 활용하고, 압축시 분할된 데이터 이중화를 통하여, 안정성과 더불어 읽기가 집중된 고부하 워크로드에서 특히 우수한 검색 성능을 제공한다. 마지막으로 성능 평가 모델을 제시하고, 실험용 데이터와 프로세스를 구축하였다. 분석 결과, 본 인덱스 및 데이터 관리 기법이 기존 기법보다 검색 처리치 에서 15% 개선되고, 응답성에서 30% 더 우수하였다.

References

[1] D. Abadi, P. Boncz, P. Alto, “Column-oriented Database Systems,” Proc. of the VLDB, Lyon, France , August pp. 24-28, 2009.
DOI: http://dx.doi.org/10.14778/1687553.1687625

[2] S. Ahn, K. Kim. "A Join Technique to Improve the Performance of Star Schema Queries in Column-Oriented Databases", Journal of Korean Institute of Information Scientist and Engineers, Vol. 40, No.3, pp. 209-218, 2013.6.

[3] Y.Chang, J. Hsieh, and T. Kuo, “Endurance Enhancement of Flash-Memory Storage System: An Efficient Static Wear Leveling Design,” Proc. of the 44th conference on Design automation, San Diego, USA, pp. 212-217, 2012

[4] S. Byun. “Search Performance Improvement of Column-oriented Storages using Compression Index”, Journal of Korea Academia-Industrial, Vol. 14, No.1, pp. 393-401, 2013.
DOI: http://dx.doi.org/10.5762/KAIS.2013.14.1.393

[5] L. Hongjun, N. Yuet Yeung, and T. Zengping, "T-Tree or B-Tree: Main Memory Database Index Structure Revisited", Proc. of 11th Australasian Database Conference, 2000

[6] R. Elmasri and S. Navathe, Fundamentals of Database System, Addison-Wesley, 2010.

[7] Y. Li, B. He, R. J. Yang, Q. Luo, and K. Yi, “Tree indexing on solid state drives,” Proc. of the VLDB, vol. 3, no. 1-2, 2010, pp. 1195-1206.
DOI: http://dx.doi.org/10.14778/1920841.1920990

[8] M. Yoo, B. Kim. and D. Lee "Hybrid Hash Index for NAND Flash Memory-based Storage Systems", Journal of Korean Information Science, Vol. 38, No.2, pp. 120-128, 2012.4.
DOI: http://dx.doi.org/10.1145/2184751.2184819

[9] C. H. Wu, L. P. Chang, and T. W. Kuo, “An efficient B-tree layer for flash-memory storage systems,” Proc. of 9th RTCSA, Tainan City, Taiwan, 2003, pp. 409-430.

[10] M. Oberhumer, LZO” Available From: http://www.oberhumer.com/opensource//lzodoc.php(accessed Feb., 10, 2015)

[11] Mesquite, User’s Guide CSIM20 Simulation Engine (C++ Version), Available From: http://www.mesquite.com//documents/CSIM20_User_Guide-C++.pdf, (accessed Feb., 10, 2015)

변 시 우(Si-woo Byun) **[정회원]**

- 1989년 2월 : 연세대학교 이과대학 전산과학과(공학사)
- 1991년 2월 : 한국과학기술원 전산학과(공학석사)
- 1999년 8월 : 한국과학기술원 전산학과(공학박사)
- 2000년 3월 ~ 현재 : 안양대학교 디지털미디어학부 정교수

<관심분야>
고속 데이터베이스, 대용량 저장장치, 스마트 시스템 등

장 석 우(Seok-Woo Jang) **[정회원]**

- 1995년 2월 : 숭실대학교 전자계학과 (공학사)
- 1997년 2월 : 숭실대학교 컴퓨터학과 (공학석사)
- 2000년 8월 : 숭실대학교 컴퓨터학과 (공학박사)
- 2009년 3월 ~ 현재 : 안양대학교 디지털미디어학과 조교수

<관심분야>
로봇비전, 증강현실, HCI, 비디오 색인 및 검색, 등