ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

고속 데이터베이스 시스템을 위한 컬럼-인지 양분화 기법☆

Column-aware Polarization Scheme for High-Speed Database Systems

변 시 우*
Siwoo Byun

요 약

최근 컬럼-기반 저장 장치는 우수한 입출력 성능으로 고속 데이터베이스 시스템의 진보적인 중요 모델이 되었다. 본 논문에서는, 기존의 가로-기반 저장 모델과 비교 분석하고, 고속 컬럼-기반 데이터베이스 시스템의 유효 성능을 향상시키기 위하여, 플래시 메모리와 어시스트 드라이브를 활용한 새로운 컬럼-인지 스토리지 관리 모델을 제안한다.

컬럼-인지 폴라라이징이라고 하는 본 스토리지 관리 기법은 테이블 컬럼을 활성-컬럼과 비활성-컬럼으로 양분하여 압축 저장하며, 고부하시에 어시스트 드라이브를 활용하여 적체된 저장 연산을 리벨런싱한다. 실험 결과는 본 제안 기법이 컬럼-기반 스토리지의 초당 저장 연산 처리치를 19% 개선하였고, 그 응답 성능도 49%개선되었음을 보였다.

ABSTRACT

Recently, column-oriented storage has become a progressive model for high-speed database systems because of its superior I/O performance. In this paper, we analysis traditional raw-oriented storage model and then propose a new column-aware storage management model using flash memory drive and assist drive to improve the effective performance of the high-speed column-oriented database system.

Our storage management scheme called column-aware polarization improves the performance of update operation by dividing and compressing table columns into active-columns or inactive-columns, and balancing congested update operations using a assist drive in high workload periods. The results obtained from experimental tests show that our scheme improves the update throughput of column-oriented storage by 19 percent, and the response time by up to 49 percent.

☞ keyword : 컬럼-지향 저장(Column-oriented Storage), 컬럼-압축(Column-Compression), 고속플래시메모리(High-Speed Flash Memory), 대용량저장소(Large-Volume Storage)

1. 서 론

최근 산업계에서 고속 데이터 저장 및 신속한 데이터 검색용으로 MIT에서 제안한 세로-저장(Column-Oriented) 데이터 스토리지[1,2]가 부상하고 있다. 세로-저장 데이터 스토리지는 기존의 한 레코드 단위로 연속 저장하는 가로방향-저장 데이터 저장방식과는 완전히 상반되어, 세로의 필드단위(컬럼)로 분리, 저장, 검색하는 새로운 데이터 저장 기법이다.

또한, 최근 스마트 기기를 비롯하여 데이터 서버나 데스크탑, 노트북 등 에서 전통적인 하드디스크 드라이브(HDD) 저장 시스템에서 초고속 플래시 메모리(Flash Memory) 기반의 저장 시스템(SSD)[3,4,5]으로의 이동이 빠르게 진행되고 있다.

세로-저장 스토리지는 테이블 저장시 세로로 분리 저장한다. 예를 들면, 아래의 회사 테이블은 논리적으로 행과 열의 2차원으로 구성되어 있다. 하지만, 실제 물리적인 저장은 엄밀히 말하면, 해당 저장 장치에서 연속적인 일차원 바이트 열로 구성되어 운영체제나 시스템 버퍼와 연동된다.[6]

(표 1.1) 스토리지 데이터의 예

E-Id	Last-name	First-name	Salary
1	Smith	Joe	40,000
2	Jones	Mary	50,000
3	Johnson	Cathy	44,000

* 정 회 원 : 안양대학교 디지털미디어학과 교수
swbyun@anyang.ac.kr
[2011/10/21 투고 - 2011/11/09 심사 - 2012/02/10 심사완료]
☆ "이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 20120002767)"

세로-저장 스토리지는 컬럼의 데이터를 세로로 집합 저장한 후, 그 다음 인접 컬럼의 데이터를 순차적으로 저장하는 방식이므로, 유사성이 높은 데이터들이 서로 군집되어, 압축, 저장, 검색에 매우 효과적인 구조이다. 이러한 물리적 구조는 저장, 검색 외에도 파티셔닝, 인덱싱, 캐싱에도 성능상 큰 영향을 준다. 과거에는 하드 디스크가 대세였고 스토리지도 이에 적응하였으므로, 하드 디스크의 접근 특성을 고려하였고, 당연히 저장 연산 처리에 가장 효율적인 방식인 가로-지향 저장 방식으로 설계되었다. 그러나 이제 고성능 SSD가 속도와 더불어 가격 경쟁력을 갖춤에 따라서, 기존의 HDD와 더불어 차세대 컬럼-저장 시스템으로서 데이터 검색과 스토리지 접근 성능 개선이 필요하게 되었다.

2. 관련 연구

먼저, 새로운 세로-저장 스토리지의 장점을 기존의 가로-지향 스토리지와 비교 분석 하였다.

1) 세로-저장 스토리지는 데이터가 메모리에 올라가면, 같은 타입의 데이터가 순차적으로 군집되어 나열되어 있으므로, 당연히 높은 압축 효율과 대용량의 고속 압축이 가능하다. 또한, 이 압축후 접근할 데이터의 크기가 대폭 줄어든 효과로서, 다량의 집합 연산 수행시 일반 스토리지가 불필요하게 전체 레코드를 읽어야 하는데 비하여, 상대적으로 매우 적게(연관된 컬럼만) 읽어도 되므로 매우 효율적이다.[7]

2) 세로-저장 스토리지는 다량의 레코드에 한 컬럼의 수치를 수정 반영할 때도 상당히 효율적이다. 실제 이런 업데이트 연산이 일상적으로도 자주 발생하는데, 일반 스토리지는 업데이트와 관련 없는 다른 컬럼들을 어쩔 수 없이 스토리지로부터 읽고 메모리상에도 할당시키느라 엄청난 시간과 공간을 낭비한다. 반면에 세로-저장 스토리지는 연산과 무관한 컬럼은 아예 터치하지 않고, 관련된 컬럼에만 압축된 최소한의 수정연산만 수행하므로, 상당히 공간효율적이며, 연산 속도도 매우 빠르다.

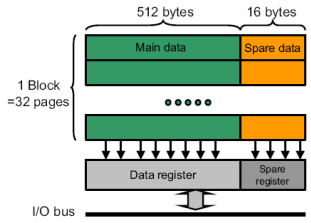
3) 세로-저장 스토리지는 컬럼별로 반복적 파이프라인을 적용할 수 있는데, 이는 캐쉬용량이 크고, 하이퍼쓰레딩이나, 멀티코어 CPU와 같은 최신 컴퓨터 환경에 잘 부합되므로, 병렬처리 효과를 극대화시킬 수 있다.

4) 반면, 일반 스토리지의 장점은 한 레코드의 많은 컬럼이 수정되는 경우나 새로운 데이터가 빈번히 추가되는 경우, 즉 온라인 쓰기 연산이 빈번한 경우에 효과적이다. 즉, 읽기가 적고, 쓰기가 매우 많은 경우는 세로-저장 스토리지가 불리한 조건이 된다.

결론적으로 보면, 쓰기 연산이 매우 빈번히 발생하는 스토리지는 기존의 가로-지향형이 더 적합하다. 반면에, 저장된 데이터가 매우 많고 복잡한 검색이 신속히 수행되어야 하는 대용량 데이터웨어하우스와 같이 신속한 검색이 필요한 경우에도 세로-저장 스토리지가 더 우수하다.

또한, 이미 플래시 메모리 저장 시스템은 고속화, 대용량화 추세에 있다. 현재 하드 디스크 저장 용량이 급속도로 증가하고는 있을지라도(현재는 1TB이상), 디스크의 기계적인 특성(헤드의 이동 시간, 디스크 회전 시간)에 의한 지연시간은 매년 15%정도의 미약한 수준으로 개선되고 있다. 반면에 플래시 메모리 처리 속도는 매년 50% 정도의 빠른 속도로 개선 발전되고 있다. 이러한 기계적 특성에 따른 속도 차이에 의하여 하드 디스크는 점차로 한계 성능에 도달하게 된다. 또한, 이에 따른 전력 소모가 평상시에도 전체 시스템 전력의 20%정도로 매우 큰 편이며, 사용 대기 중에도 전력이 계속 소모된다.[8] 이에 대한 대안으로 최근에 주목받고 있는 저장장치가 바로 플래시 메모리를 장착한 SSD이다. 결론적으로 입출력 성능 및 저전력 측면에서는 플래시 메모리 저장 장치가 매우 우세하다.

다만, 일반 메인 메모리와는 달리, 쓰기와 소거 연산에 상당히 많은 시간이 더 소모되며, 쓰기 횟수가 최대 1,000,000번 정도로 제한되는 고유한 특성이 있다.[5] 일반적으로는 충분한 수명이나, 이 수명을 넘기면 더 이상 쓰기 연산을 수행할 수 없다. 단, 읽기 연산은 계속하여 수행이 가능하다.



(그림 2.1) 플래시 메모리

또한, 플래시는 일반 메인 메모리와 달리 바로 쓰기 연산을 수행할 수 없고, 이전에 미리 블록 단위로 포맷 개념의 소거 연산을 수행한 후 쓰기 연산을 수행할 수 있다. 즉, 플래시 메모리는 PC의 메인 메모리처럼 Update-In-Place(제자리 덮어 쓰기)가 불가능한 Update-Out-Place 구조이다.[5]

세로-저장 스토리지도 시스템 운영을 위하여 반드시 저장 장치가 필요하다. 물론, 이러한 저장 장치로서 기존의 전통적인 하드디스크를 그대로 사용할 수도 있다. 하지만, 최근 본 연구팀은 새롭게 부상 중인 플래시 메모리 저장 기술을 활용하면, 아래와 같은 이유로 최상의 궁합을 만들 수 있음을 발견하였다.

첫째, 플래시 메모리는 그동안 고가였던 이유로 검토 대상에서 제외되었지만, 현재는 충분한 가격경쟁력을 갖추고 있음을 우선 주장한다.

둘째, 플래시 메모리가 최근 빠른 추세로 대용량화 되고 있으며, 읽기 속도가 하드디스크에 비하여 매우 빠른 장점이 있다. (보통 250MB/s로서 3배 이상 빠름) 따라서, 세로-저장 스토리지의 사용처가 대부분 고속 읽기가 필요하므로 매우 적합하다.

따라서 세로-저장 스토리지는 처음에 한번 로딩하면 쓰기 연산이 자주 발생하지 않는 특성이 있고, 이후에도 대체로 검색에 주로 사용되어 쓰기 부담이 상대적으로 적으므로 플래시 메모리 특성에 잘 맞다.

이러한 관점에서, 본 연구에서는 각 컬럼의 저장 특성을 효과적으로 분석, 활용하며, 기존 대용량 HDD와 더불어 고속 SSD의 약점을 보완하고 장점을 최대한 반영하여, 대용량 초고속의 컬럼-지향 데이터베이스에 최적화된 저장 성능을 낼 수 있는 컬럼-인지 저장 기술을 제안하고자 한다.

3. 컬럼-인지 저장 시스템의 제안

3.1 컬럼 데이터 저장을 위한 특성 분석

최근 급부상중인 SSD는 휴대용 고급 노트북이나 고성능 서버용 주요 스토리지로 활용범위를 넓히고 있다. SSD는 고성능 플래시 메모리를 사용하기 때문에, 윈도우 부팅시 1분정도인 일반 하드 디스크보다 읽기 속도가 향상되어 20초에 부팅이 완료되고, PDF등의 문서 읽기를 수행하면 무려 3배 가까운 속도의 향상이 나타난다. 이렇게 높은 접근 성능이 나오는 이유는 SSD는 HDD의 기계적 동작부분이 전혀 없고 거의 전자적으로 작동하기 때문이다. 즉, 하드 디스크의 물리적/기계적인 접근 시간에 해당하는 헤드의 탐색시간(헤드 모터 이동 시간)과 디스크 원판의 회전지연시간(모터의 섹터 접근 시간)이 전혀 없기 때문이다. 다만, 쓰기 성능에 있어서는 아직도 개선의 여지가 크며, 최근 몇 년간의 기술발전으로 하드 디스크에 비해서는 빠르지만, 과부하시는 아직도 가격 대비 아쉬운 성능을 보여준다.

SSD는 크게 두 종류, 즉 멀티 레벨 셀(Multi-level Cell, 약어 MLC)과 싱글 레벨 셀(Single-level Cell, 약어 SLC)로 나눈다. MLC란 멀티플 레벨(multiple level)을 활용하여 한 셀 당 1 비트 이상의 정보를 저장하는 기술이며, SLC란 한 셀당 1비트만을 저장하는 기술이다. MLC의 현재 상용화된 기술로는 한 셀 당 최대 4 상태(state)를 갖게 할 수 있으므로, 셀 당 2 비트의 정보를 저장하게 된다. 따라서 MLC는 SLC에 비하여, 저장 밀도가 높아서 셀 당 제조비용은 적으므로 부품 값이 훨씬 더 저렴하다. 하지만 MLC는 비트 오류를 보상하기 위해, 소프트웨어 복잡도도 증가하고, 일반적으로 셀 당 전력 소모가 더 크며, 속도도 SLC보다 더 느린 단점이 있다. [9]

또한, SSD는 MLC이든지 SLC이든지 상관없이, HDD처럼 기계적 특성에 의한 지연은 없지만, 역시 저장 연산이 집중되면, 적체가 심화되어, 평소 속도보다 느린 속도로 반응한다. 또한, 오래 사용하면 할수록 더티상태의 셀들이 늘어나게 되서 속도저하는 필연적이다. SSD 구입 후 바로 벤치마킹을 했을 경우에는 스펙 이상의 속도가 나오거나 정상적인 속도가 나오지만, 오래 사용하여 저장 영역을 대부분 채운 후에 측정 하면 스펙 이하의 속도가 나오는 경우가 많다. 즉, 사용초기의 경우에는 모든 셀이 클린상태이기 때문에 읽기/쓰기 속도가 매우 잘 나오지만, 사용용량을 가득 채운 후 의 더티 셀이 많은 상태에서 벤치마킹을 수행하면, 바로 덮어쓰기가 불가능한 플래시의 특성상 스펙 이하의 속도가 나오게 된다.

3.2 컬럼-특성 기반 압축 양분화 저장 관리 기법

본 연구에서는 컬럼-데이터베이스의 성능 향상을 위하여, 기존의 SSD 저장 및 검색 기술을 개선하고, 고속 검색용 SSD의 약점을 보완하기 위하여, 저가의 대용량 HDD를 융합하여, 전체적인 시너지 효과를 낼 수 있는 새로운 컬럼-인지 양분화(Column-aware Polarization: CaP) 저장 기법을 제안한다. 먼저, CaP기반 스토리지 제어에서 읽기 연산은 컬럼 데이터가 저장된 해당 드라이브를 통하여 매우 빠르게 수행된다. 즉, 고성능인 MLC SSD의 읽기 속도는 매우 빠르며 메모리 상주 등의 상승효과로 별 병목현상 문제가 없으므로 간략히 서술하며, 본 논문의 이슈인 압축 저장용 쓰기 연산에 대하여 설명하고자 한다.

첫째, 제안하는 CaP기법을 활용하면 전술한 SSD의 성능 저하 및 적체 현상을 크게 감소시킬 수 있다. 즉, 적체가 시작되기 전까지의 일반 부하시는 MLC SSD의 응답속도의 저장 처리율을 최대 피크로 유지하여, 전체적으

로 최고 성능을 뽑아낸다.

둘째, 반대로 MLC SSD(마스터 드라이브)의 쓰기 적체에 의한 과부하와 성능 저하가 시작되면, 어시스트 저장 장치(assist drive)인 HDD가 하이브리드 엔진처럼 개입을 시작하여, MLC의 성능 저하를 가속시키는 비효율 압축 연산을 HDD로 대신 처리하기 시작한다. 개입시점은 적체가 시작되는 시점을 MLC의 큐-길이로 판단한다. 즉, 큐의 길이(대기 작업수)가 길면 적체가 시작된 것인데, 몇 개의 소량 적체는 정상적으로 판단할 수 있으나, 갑자기 계단식으로 수십 개로 크게 급상승한 경우는 심각한 적체가 발생한 것이다. 또한, 적체판단시점은 큐 길이가 급상승하기 조금 이전이 유리하므로, 설정한 임계치보다 낮게 설정한다.

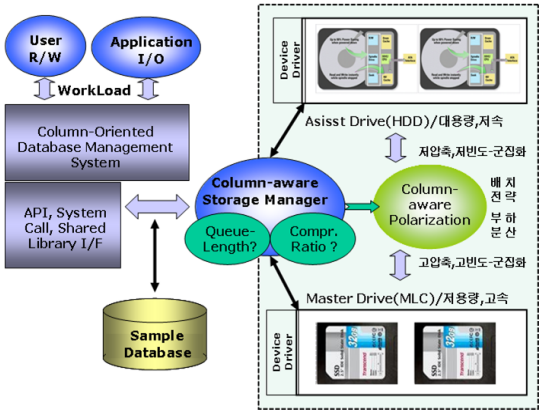
HDD는 상대적으로 입출력은 저성능이긴 하지만, MLC에 적체된 원인이 되는 저압축율로 압축된 컬럼 데이터를 대신 저장하여, 적체 요인을 신속히 제거해주는 역할을 수행하는데, 무엇보다 가격이 매우 저렴하며, MLC에는 저장하기 부담스러운 대용량 데이터를 저장 가능한 장점이 크다.

셋째, 이 적체 제거 과정을 위하여, CaP은 MLC에 처리할 컬럼 데이터와 HDD가 처리할 컬럼 데이터로 두 방향으로 양분하여(Polarize)하여 관리한다. 즉, 저장 요청이 발생된 컬럼 데이터 중에서, 컬럼의 압축률이 높으며 빈번히 쓰기가 이루어지는 컬럼인 고활성-컬럼과 압축이 잘 안되고 접근이 되지 않는 저활성-컬럼으로 양극화시켜서 분류하여 저장 관리한다.

SSD를 구성하는 플래시 메모리는 업데이트가 빈번한 고활성-속성의 컬럼-데이터들을 같은 저장 영역으로 군집화을 유도하면, 개별적으로 수행해야 할 쓰기와 소거 연산을 집합 개념으로 한 번에 몰아서 수행 가능하므로, 전체적인 저장-연산의 오버헤드를 크게 줄일 수 있다. 특히, 컬럼-데이터의 군집화는 일반적으로 마구 섞여 있는 랜덤 저장의 경우보다 상대적으로 소거(클리닝)오버헤드가 줄어서 결국은 SSD의 수명도 연장된다. 또한, SSD의 플래시 메모리는 용량 대비 아직은 고가이므로, 압축이 잘 안되며 접근이 잘 되지 않는 비활성-속성의 컬럼-데이터는 대용량 HDD에 저장하면 매우 경제적이다.

특히, 최근 사용자는 기존의 수치 데이터나 문자 데이터 외에도, 사진 파일이나, 암호화 데이터, 심한 경우 동영상이나 실행파일도 그대로 데이터베이스에 저장하여 쉽게 통합관리를 하고자한다. 이러한 데이터를 통틀어서 BLOB(Binary Large Object)라 하는데, 불행히도 초고속을 추구하는 컬럼 데이터베이스에서는 컬럼내부에서 공통성과 유사성이 별로 없어서, 압축이 불가능하거나, 가능하더라도 시스템에 크게 부담을 주고 데이터의 양도 많아서 성능을 심각하게 저하시키는 요소이다. 즉, 이러한 BLOB나 저압축된 비활성 컬럼들을 결국 MLC의 연산 적체를 가속화하고 장기화시키므로, MLC의 작업 큐에서 빼내어, HDD로 넘겨서, MLC가 최고의 연산 수행 컨디션을 유지시켜준다.

결과적으로 저장 연산의 적체요인이 제거되면서, MLC는 최적의 성능을 다시 발휘하며, 과부하가 해소된다. 적체현상이 사라지면, HDD의 개입도 정지되어 HDD는 비활성모드로 전환되어 전력을 절약하게 된다. 즉, 이러한 어시스트디스크의 보완적인 특성을 잘 활용하면, MLC 드라이브의 급격한 응답지연과 성능 저하현상을 충분히 예방할 수 있다. 압축 알고리즘은 소스가 GPL로 공개되어 확보가 용이하고, 압축효율이 높은 LZO기법[10]을 사용하였고, 제안하는 CaP 기반 저장 시스템의 구성도는 아래와 같다.



(그림 3.1) 제안 시스템 구성도

다음은 CaP 저장 기법의 수행 알고리즘이다.

사용자나 응용 프로그램이 컬럼-데이터베이스(CDB: Column-oriented DataBase) 스토리지에 대한 읽기 또는 쓰기 연산(operation)을 요청한다.

사용자나 응용 프로그램의 입출력 요청이 들어오면, 요청된 연산은 컬럼-인지 저장 관리자(Column-aware Storage Manager: CaSM)를 통하여 CaP 연산으로 전환된다. 그리고 CaSM은 이러한 입출력 요청연산을 자신의 연산-큐의 마지막에 삽입한다. 추후에 지금 요청한 연산이 완료되면 결과 값을 최종적으로 사용자나 응용 프로그램에 리턴 한다.

이 저장 연산-큐에서 한 연산씩 차례로 꺼낸 후 연산의 종류를 분류한다. 만일 읽기 연산이면, (4) CaP_READ을 수행하며, 쓰기 연산이면 (5) CaP_WRITE를 수행한다.

CaP_READ(Column-aware-Polarized_READ):

1) 먼저, MLC Index에 접근하여, Column data의 저장 위치를 파악한다. 저장 위치가 읽기 속도가 빠른 MLC-SSD나 느린 HDD에 저장되어 있으며, 해당 드라이브에서 블록 읽기를 수행한다.

2) CaP 스토리지로부터 블록 읽기 작업이 완료되면, 판독한 값을 LZO_Manager에서 포워딩하여 압축복원 작업을 수행한 후 CaSM에게 리턴 한다.

CaP_WRITE(Column-aware-Polarized_WRITE):

1) 요청된 컬럼-특성이 비압축 모드면, HDD를 타겟으로 설정한다.

2) 컬럼-특성이 압축 모드면, 현재 MLC 저장장치의 큐-길이(Q-Length)를 측정한다.

3) 큐-길이가 임계치 이하면 정상부하로 판단하여, 타겟을 MLC로 설정하고, 임계치 이상이면 과부하로 판단하여, 타겟을 HDD로 설정한다. 본 테스트 환경에서는 실제 과부하는 큐-길이 20에서 발생하지만, 적체이전에 대응하고자 기본 임계치를 10으로 설정하였다.

4) 저장할 컬럼 데이터를 LZO_Manger에서 포워딩하여, Compression 작업을 수행한다.

5) 압축된 컬럼 데이터를 타겟 스토리지의 블록에 기록하고, CaSM에게 저장 완료 메시지를 리턴 한다.

CaP_MOVE(Column-aware-Polarized_MOVE):

1) MLC의 여유 저장 공간이 설정치 이하이거나(여유 공간 확보 필요발생), HDD와 MLC 드라이브가 하나라도 활동 상태이면 계속 대기하고, 유휴 상태이면 아래의 MOVE 작업이 수행된다.

2) MLC에 저장된 컬럼중 가장 접근 빈도가 낮은 컬럼부터 선택하여, MLC 드라이브에서 HDD로 컬럼 데이터를 복사한다.

3) 복사 작업이 성공하면, MLC의 데이터를 삭제(unlink)하고, MLC Index의 위치를 수정한다.

4) MLC에 여유 공간이 충분히 확보될 때까지 1)번~ 3)번을 반복한다. 결과적으로, 접근성이 좋은 컬럼들이 MLC에 남고, 저장 빈도가 낮은 컬럼들은 HDD에 모인다.

(표 3.1) 큐-길이의 변화 추이

저장부하	50	100	150	200	250	300	350..
적체된 큐-길이	0	1	1	3	**10**	27	35

또한, 유휴시를 이용하여, 한꺼번에 HDD와 MLC에서 상호쓰기 작업이 수행되므로, SSD의 경우 모아쓰기를 유도하는 부수적인 효과가 발생하여, SSD이 저장효율과 전력절감 효과가 발생한다. HDD의 경우에도 랜덤한 임의쓰기 패턴이 모여서 한꺼번에 순차쓰기 효과가 있으므로 성능향상에 유리하다.

위에서 큐-길이의 기본값을 10으로 설정한 이유는 (표 3.1)의 선행 실험 결과, 큐-길이가 갑자기 급증하는 지점이 10으로 측정되므로, 이 지점이 임계점으로 설정되었다.

4. 성능 평가

본 연구에서 제안된 CaP 저장 기법의 성능을 검증하기 위하여 실제 환경에서 부하를 주고, 그 실험 결과를 분석해 보았다. 본 연구의 CaP 저장 기법의 드라이브를 CaPD(Column-aware Polarized Drive)라 명하고, 이와 비교된 대상 드라이브는 기존의 하드디스크를 사용하는 저장장치(HDD)와, SSD 중에서 보편적인 MLC-type인 MLC-SSD 저장장치, 고가인 SLC-SSD 저장 장치이다.

4.1 테스트 환경 구축

1) 시스템 환경 구축

실험이 수행된 하드웨어 환경은 HP 8100, I7-2.93GHz QUAD CPU와 메인 메모리 8G, 운영체제는 64bit MS-Windows7을 사용하였다. 개발언어는 비주얼 스튜디오 2008을 사용하였으며, 부수적으로 작업 부하 생성모듈과 물리적인 실험후 통계 자료 수집 및 분석용 CSIM[11] 라이브러리를 사용하였다.

2) 테스트 데이터베이스 구성

본 실험에 앞서서, 실험용 샘플이 되는 대용량의 데이터가 필요한데, 아래와 같이 한 게임포털에서 사용빈도가 높은 한 데이터베이스를 입수하고, 각 컬럼특성을 분석하여 총 100만건의 데이터베이스를 구축하였다.

각 컬럼을 압축하기 위하여, LZO 압축 저장 기법을

(표 4.1) 게임 플레이 테이블의 특성 비교

컬럼명 / 특성	item UseNo	userNo	game code	game cheat	use ... Date ...
압축시간 (ms) /10,000건	30	22	25	8	48
크기(Byte) /10,000건	2,379	659	1,905	192	13,597
압축률	4	1	2	0.3	8

사용하였는데, 컬럼-저장의 군집 특성상 압축률이 8%이하로 매우 좋은 편이다. 특히, 종류가 한정된 코드나 사용자번호가 압축이 잘되며, gamecheat 컬럼처럼 Boolean (True,False) 타입으로 압축이 가장 잘되어 0.3%로 거의 다 압축되었다. 이 테이블의 경우 코드화되어 있어서 비교적 압축이 잘 된 경우이나, 만일 주소속성과 같은 불규칙한 문자열이 많이 포함될수록 압축률이 더욱더 낮아진다.

저장되는 압축컬럼의 크기는 1만건당 수백바이트에서 수십KB 정도이며, 실험에서는 공정하게 하기 위하여 랜덤하게 임의의 컬럼이 선택된다. 또한, 데이터웨어하우스는 읽기특성이 강하고, 테이블의 전체수정은 적으므로, 평균적인 저장 연산의 경우, 전체 100만건중 10만건의 부분갱신으로 기준하였다. 물론, 이러한 부분갱신 범위의 설정은 데이터웨어하우스 사용자의 작업환경에 따라서 조금 다를 수 있으나 실험의 목적과 성능순위에는 영향을 미치지 않는다.

3) 워크로드 모델 및 평가 지표

다음으로 실험을 위한 저장 워크로드 생성 및 결과 분석 모델이 필요한데, 이는 CSIM에서 제공되는 폐쇄형 큐잉 모델(Closed Queuing Model)과 통계 라이브러리를 사용하였다. 이 모듈들을 통하여 초당 일정한 수의 저장 오퍼레이션을 요청하고, 이와 연관된 하부 저장시스템의 처리 시간과 성능을 측정하면 된다. 즉, 본 실험의 주요 성능 관련 평가 지표는 저장 연산 처리치(update throughput)와 그 응답 시간(response time)이다. update throughput은 초당 몇 개의 저장 오퍼레이션이 처리되었는지를 의미하고, response time은 저장 오퍼레이션이 발생한 후 수행완료까지 걸린 시간을 의미한다. 컬럼 데이터베이스의 메모리 내부에서 컬럼 데이터의 업데이트가 발생하면, 이는 해당 SSD나 HDD에서 컬럼 데이터의 저장 연산으로 연동되어 안전하게 수행 완료된다.

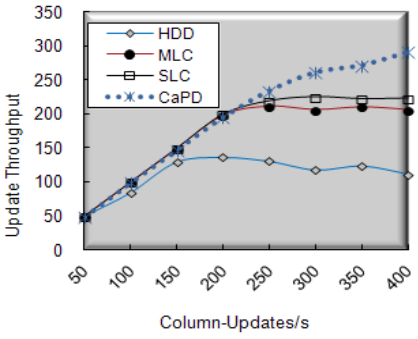
주요 실험 파라미터인 초당 생성된 저장 오퍼레이션의 수는 초당 50개에서부터 50개 단위로 초당 400개까지 8단계로 변화시켜 보았으며, 이는 저장 시스템에 가해지는 작업부하를 의미한다.

4.2 실험 결과 및 분석

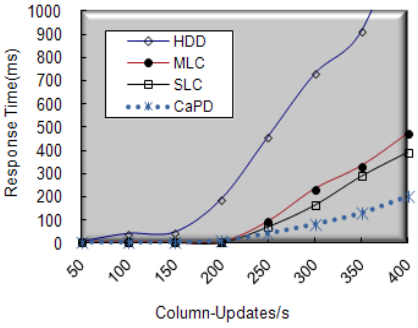
이 실험은 기본적으로 하드 디스크나 SSD 스토리지를 탑재한 일반적인 사용자 시스템에서 부과한 저장 워크로드, 즉 초당 생성된 저장 연산의 수가 시스템 성능에 어떤 영향을 미치는 지를 분석하기 위함이다. (그림 4.1)은 초당 발생된 저장 연산수의 증가에 따른 저장 연산의 처리치를 표시한 그래프이다. 발생된 초당 저장 연산의 수가 늘어날수록 점차로 연산의 처리 결과치가 증가함을 알 수 있다. 또한, 전반적인 저장 연산의 처리 성능을 측정한 결과, 제안 기법인 CaPD가 가장 높게 나타났다.

(그림 4.1)에서 살펴보면, 초당 저장 연산의 수가 대략 150~250개를 넘으면서 각 저장 기법의 성능의 차이가 뚜렷하게 발생한다. HDD의 경우는 업데이트 워크로드 150 연산/초 부하 지점에서 부터 성능이 저하되기 시작하였다. 특이한 점은, MLC 보다 매우 고가인 SLC의 처리 성능이 기대치보다는 낮게, 즉 MLC보다 약간만 더 높게 나타났다. 이유는 스펙에 제시된 저장장치의 벤치마크 성능은 연속적인 입출력만이 벌크로 일어나는 경우의 수치이고, 실제응용의 경우에는 데이터베이스에서 메모리 연산, CPU 처리시간, 특히 압축 등의 부가적인 다양한 프로세스들이 결합되어 있기 때문에 벤치마크시의 표면적인 수치보다는 낮아진다. SLC의 성능은 MLC에 비하여 가격대비 좋은 것은 아니지만, 조금이라도 빠른 응답시간과 고성능이 필요하다면 당연히 가장 빠른 SLC를 사용하여야 한다.

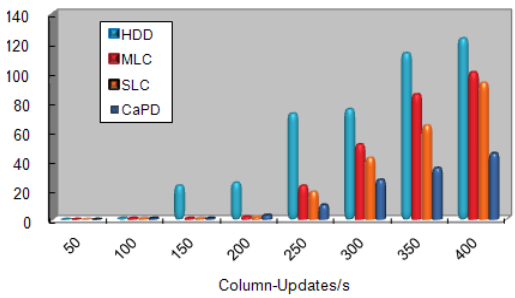
다시 그래프에서 보면 워크로드 250이하의 저장 연산의 부하가 적은 시작 구간에서는 SSD의 느린 쓰기와 소거 연산이 시스템 안에서 수용되어 성능 저하를 일으키지 않는다. 하지만, 이 부하 구간을 지나면서 생성되는 저장 연산의 수가 계속 증가하므로, 플래시 메모리에 하위 레벨의 쓰기 연산의 수도 증가하면서 누적되게 된다. 따라서 중간 구간 이후에서는 저장 연산들이 SSD 컨트롤러의 작업 큐에 과도하게 쌓이면서, 저장 연산 처리가 지체되며 결과적으로 성능이 서서히 저하되게 된다. 이는 (그림 4.3)의 처리 큐에서 대기 중인 저장 연산수의 비교해 보면 명확하다. 즉, SLC, MLC 모두 200 지점 이전에는 1-2 개에 불과하던 누적된 저장 연산수가, 250 지점



(그림 4.1) 초당 저장 연산 처리치의 비교



(그림 4.2) 저장 연산의 응답 시간의 비교



(그림 4.3) 처리 규에서 대기 중인 저장 연산수의 비교

이 되면서, 약 20개로 크게 증가하게 된다. 하지만, 동일한 워크로드의 조건에서도 제안한 CaPD 저장 기법이, 고급형인 SLC보다도 전체 장치비용이 적으면서도, SLC를 포함한 모든 기법에 비하여 성능이 상대적으로 더 높다.

실험결과, 전체 구간에서 즉, 50~400구간을 합산후 평균하여도, CaPD 저장 기법이 다른 기법보다 최소 19% 이상 처리 성능이 더 높다. 특히, 최대 컬럼-데이터 저장 연산 처리치는 CaPD가 291개로 SLC의 223개 보다 31% 정도 더 높다.

그 이유는 위의 저장 성능 저하의 주요 원인인 SSD의 쓰기 연산의 부담에 의한 적체 현상을 CaPD의 컬럼-인지 폴라라이징 전략을 통하여, 임계 이상의 SSD 대기 지연 현상이 발생하면, 적체된 컬럼-저장 연산을 어시스트 디스크로 분산하였기 때문이다. 즉, SSD의 Queue-Length를 모니터링 하여 임계치인 10을 넘으면, 적체가 매우 심각하여, 더 이상 SSD만으로는 성능 유지가 불가능하다고 판단하고, 감당할 수 없는 부하이상의 작업은 저가의 저성능인 어시스트 HDD로 포워딩한다. 이때, 단순히 무작위로 포워딩하지 않고, SSD에 효율적인 저장 연산은 받고, 불리한 저장 연산은 어시스트 HDD에 포워딩한다.

반대로 압축률이 안 좋은 컬럼이거나, 압축이 거의 안 되는 BLOB 타입의 컬럼은 어시스트 HDD에 포워딩한다. 어시스트 HDD는 저성능이긴 하지만, SSD의 적체를 이전받아서 간접적으로 잘 해소시키며, 대용량의 컬럼 데이터를 저가의 비용으로 저장할 수 있어서 매우 효율적이다. 특히 하드 디스크가 일반적인 환경처럼 평소에도 회전하거나, 헤드가 움직이면서, 계속하여 전력을 낭비하지 않고, SSD의 적체시인 과부하시에만 한꺼번에 작동하므로, HDD의 높은 전력소모와 진동, 소음 감소, 수명 연장에 모두 이익이다. 또한, CaPD 기법은 적체가 또 적체를 부르는 악순환을 잘 차단하였음은 (그림 4.3)에서 MLC와 HDD의 큐-길이를 합한 것인 SLC의 큐-길이 보다 훨씬 짧다는 측정결과로 재확인할 수 있다. 결과적으로, 전체적인 대기시간을 크게 감소시킬 수 있었다.

다음으로 (그림 4.2)에서 사용자 프로그램이 요청한 저장 연산에 대한 각 스토리지 저장 기법의 응답시간을 비교해보면, CaPD, SLC, MLC, HDD 순으로 우수하게 나타났다. 먼저, HDD는 기계적 특성에 의하여 느린 응답속도를 보이며, 특히 150구간을 지나면서, 급격히 느려지기 시작한다. 더욱이, 350구간을 지나면서, 1초 이상의 매우 느린 응답시간을 보이므로, 대용량의 고속 분석이 필요한 데이터웨어하우스 분야의 실제 업무에서 사용이 불가능하게 된다. 다른 세 기법은 전체구간에 대하여 과부하 상태시에도 0.48초 이하로 감내할 만한 수준이다.

실험 분석 결과, CaPD는 전구간 평균하여, SLC에 비하여 49% 더 개선된 응답시간을 보였다. 물론, MLC를 사용하므로, 저부하구간인 100지점은 SLC가 4.3ms로서 CaPD의 6.4ms를 앞서지만, 그 차이가 2/1000초로서는 사용자가 전혀 느낄 수 없다. 그러나 200이상의 부하에서부터는 CaPD가 매우 앞서는데, 사용자가 느끼는 차이도 SLC 0.4초 CaPD 0.2 초로서 충분히 속도차를 감지할 수 있다. 이는 부하가 더 심할수록, 여러 명의 사용자가 같

이 사용할수록, 한 사용자라 하더라도 한꺼번에 여러 컬럼이나, 테이블을 갱신할 경우를 고려하면, 두 기법사이에는 수십 초의 차이도 날 수 있다. 결과적으로 CaPD가 전 구간 평균하여, SLC대비 적은 비용을 사용하면서도, 오히려 응답시간은 약 절반정도로 개선되었다.

이 사실은 (그림 4.3)의 대기 큐를 분석해 보면 재확인할 수 있다. 즉, HDD는 150구간에 도달하면, 1-2개 이던 적체 작업수가 갑자기 23개로 늘어난다. SLC와 MLC도 마찬가지로, 250 구간에서 갑자기 19개, 23개로 폭증한다. 이는 원활하게 감내할 임계 작업수를 넘기면, 적체가 적체를 부르는 악순환이 시작됨을 의미한다. 물론, 제안 기법인 CaPD도 계속하여 워크로드를 올리면, 임계치에 도달할 수밖에 없지만, 테스트 전 구간에서 CaPD의 큐의 길이가 다른 세 저장 시스템의 큐의 길이 보다 훨씬 더 짧음을 알 수 있다. 가장 성능이 우수한 고가의 SLC와 비교해 보아도, 전체적인 변화 구간에서 CaPD의 큐-길이는 SLC 대비 0.54배로서 현저한 성능차이를 보였다. 또한, 큐의 길이 그래프의 기울기도 CaPD가 다른 세 저장 시스템 보다 완만하게 증가하여, 전체적으로 큐의 대기 연산수가 훨씬 더 적은 우수한 성능을 보인다.

(표 4.2)는 각 저장 기법의 요약비교를 위하여, 전체 부하구간에서 평균한 응답시간과 평균한 저장 연산 처리치로 재계산한 것이다. 본 CaPD기법은 기존의 가장 우수한 SLC 기법에 비하여, 49% 응답이 신속하며, 저장 처리치도 19% 더 우수한 성능을 보였다.

(표 4.2) 각 저장 기법의 평균 성능의 비교

기법 분류	HDD	MLC	SLC	CaPD
응답시간	730	234	165	84
저장처리치	61	104	111	132

5. 결 론

본 논문에서는 기존의 SSD 저장 및 검색 기술을 개선하고, 고속 검색용 SSD의 약점을 보완하기 위하여, 저가의 대용량 HDD를 융합하여, 전체적인 시너지 효과를 낼 수 있는 컬럼-인지 양분화 저장 기법을 제안하였다.

본 저장 기법의 특징은 중저부하 구간에서는 MLC를 사용하여, SLC에 버금가는 높은 저장 연산 처리율과 빠른 응답시간을 확보하며, 반대로 고부하 구간에서는 주력인 MLC에 어시스트 디스크를 보조로 개입시켜서, MLC가 누적된 저장 부하를 감내하며, 저압축이나 비압축되어 SSD에 상주하기에는 비효율적인 컬럼데이터를 어시스트 디스크로 포워딩한다.

성능 실험 결과, 제안한 컬럼-인지 양분화 저장 기법이 전체 워크로드를 평균하여, SLC 스토리지에 비하여 49% 더 개선된 응답시간을 보였으며, 초당 저장 연산 처리치는 다른 기법보다 최소 19% 이상 우수한 결과를 보였다.

참 고 문 헌

[1] D. Abadi, S. Madden, and M. Ferreira, "Integrating compression and execution in column-oriented database systems", In SIGMOD, pp. 671 – 682, 2006.

[2] D. Abadi, D. Myers, D. DeWitt, and S. Madden, "Materialization strategies in a column-oriented dbms", MIT CSAIL Technical Report. MIT-CSAIL-TR- 2006-078, 2006.

[3] Y. Bae, "Design of High Performance SSD", The KIPS transactions, Vol. 25, No. 6, pp. 18-28, 2007

[4] 변시우, "하이브리드 하드디스크 시스템을 위한 플래시 노드 캐싱 기법", 한국산학기술학회논문지, 제 9권 6호, pp. 1696-1704, 2008

[5] S. Byun, M. Hur, "Flash memory Lock management for portable information systems", International Journal of Cooperative Information Systems, Vol. 15, No. 3, pp. 461-479, Aug. 2006.

[6] A. Halverson, J. Beckmann, and J. Naughton, "A comparison of c-store and row-store in a common framework", Technical Report, UW Madison Department of CS, TR1566, 2006.

[7] S. Harizopoulos, V. Liang, D. J. Abadi, and S. Madden, "Performance tradeoffs in read- optimized databases", In VLDB, pp. 487 – 498, 2006.

[8] A. Roberts, T. Kgil, and T. Mudge, "Integratin NAMD Flash Devices onto Servers", Communications of the ACM, Vol52, No.4, pp. 98-106, 2009.

[9] Micron, "Micron's MLC NAND Flash Webinar", http://www.micron.com/products/nand/mlc-webinar.aspx, 2011.

[10] Oberhumer, "LZO-- a real-time data compression library", http://www.oberhumer.com/opensource/lzo/lzodoc.php

[11] Mesquite, “CSIM2.0 Development Toolkit for Simulation and Modeling”, http://www.mesquite.com/documentation/documents/CSIM20_User_Guide-C.pdf, 2011

◐ 저 자 소 개 ◑

변 시 우

1989년 연세대학교 전산과학과(공학사)
1991년 한국과학기술원 전산학과(공학석사)
1999년 한국과학기술원 정보및통신학과(공학박사)
2000년~현재 안양대학교 디지털미디어학과 교수
관심분야 : 데이터베이스, 저장장치, 스마트 임베디드 시스템
E-mail : swbyun@anyang.ac.kr