OLTP 환경에서의 플래시 SSD 기반 해시 조인 성능에 대한 고찰

구동현*, 심준현*, 김강년*, 이상원*
*성균관대학교 컴퓨터공학과
e-mail:smwindy@naver.com

Hash Join Performance on Flash SSD in OLTP Environment

Dong-Hyun Koo*, Jun-Hyeon Sim*, Kang-Nyeon Kim*, Sang-Won Lee*
*Dept. of Information and Communication Engineering,
Sungkyunkwan University

요 약

OLTP(online transaction processing) 환경은 다수의 사용자의 질의 및 요청을 처리하기 위한 데이터베이스 환경으로서 신속하고 정확한 질의 처리가 요구된다. 조인 연산은 이러한 데이터베이스 관리 시스템에서 자주 처리하게 되는 질의 가운데 하나이며, 그 중에서도 해시 조인은 현재 가장 좋은 성능을 보인다고 알려진 조인 알고리즘이다. 이 논문에서는 직접 해시 조인을 구현하여 읽기 및 쓰기 버퍼의 크기가 제한되었을 때 SSD와 하드디스크에서 해시 조인의 성능을 비교하고, 나아가 다수의 사용자가 동시에 여러 개의 조인 연산을 요청했을 때의 상황을 시뮬레이션 하여 SSD에서의 최적화 방안을 생각해본다.

1. 서론

조인 알고리즘은 데이터베이스 관리 시스템에서 질의 처리에 자주 사용되며 시간이 많이 걸리는 연산이다. 상용 데이터베이스 관리 시스템에서 사용되는 조인 알고리즘에는 정렬-병합 조인(sort-merge join), 중첩 반복 조인(nested loop join), 해시 조인(hash join) 등이 있는데, 이 가운데 해시 조인은 현재 가장 좋은 성능을 보인다고 알려져 있다. 해시 조인은 데이터베이스 공간에 저장된 두 개의 테이블을 읽어들여 각 테이블을 동일한 해시 키(hash key)를 갖는 파티션으로 분류한 뒤, 해시 키가 같은 파티션끼리 비교하여 두 레코드를 하나로 통합하는 작업이다. 이 때, 개별 파티션의 용량이 버퍼 공간보다 큰 경우 디스크 입출력이 발생하게 되는데, 특히 다수의 사용자가 하나의 데이터에 접근하게 되는 OLTP(online transaction processing) 환경인 경우, 개별 사용자 당 사용할 수 있는 메모리가 줄어[1]들어 디스크 입출력이 더욱 빈번하게 발생하게 된다. 해시 조인에서 주로 발생하는 입출력 패턴은 파티션 쓰기 단계에서 발생하는 순차 쓰기와, 파티션 간의 키 값 비교 시에 발생하는 랜덤 읽기이다.

한편, 최근 하드디스크를 대체할 새로운 대용량 저장 매체로 각광받고 있는 플래시 SSD는 낸드플래시 메모리를 집적하여 만든 전자식 저장 장치이다. 플래시 SSD는 하드디스크의 기계적 특성으로 인해 밀리 초 단위로 제한되었던 데이터 접근 시간을 전자적 접근을 사용하여 마이크로 초 단위까지 개선하였는데, 이는 디스크 입출력 발생 시 데이터 접근 시간(latency)이 줄어드는 결과를 가져왔다. 이는 지금까지 주로 하드디스크의 순차 읽기 및 쓰기를 극대화하는 방향으로 이루어졌던 IO 최적화 정책이 변화해야 함을 시사한다. 이 논문에서는 플래시 SSD의 높은 랜덤 읽기, 순차 쓰기 성능을 활용하여 해시 조인 처리 성능을 개선할 수 있음을 보인다.

이후의 구성은 다음과 같다. 2장에서는 실험에 사용한 그레이스 해시 조인(grace hash join) 알고리즘과 Flash SSD에 대해서, 3장에서는 실험에 사용된 하드웨어 및 소프트웨어 환경에 대하여 설명한다. 4장에서는 읽기 및 쓰기 버퍼 크기에 따른 하드디스크와 플래시 SSD의 해시 조인 성능을 분석하고, 멀티스레드로 OLTP 환경을 시뮬레이션한 결과를 보인다. 마지막으로 5장에서는 실험 결과를 요약, 향후 과제를 논의하며 논문을 마친다.

2. 관련 연구

2.1 해시 조인

가장 기본적인 해시 조인 알고리즘은 두 개의 테이블 중 크기가 더 작은 테이블을 기준 테이블(build input)로 선택하고 버퍼 크기만큼씩 읽어들여 해시 테이블을 생성하고, 생성된 해시 테이블에 대하여 다른 테이블 전체를 탐색하면서 키 값의 일치 여부를 확인한다. 이때 탐색 테이블(probe input)은 해시 테이블이 생성 될 때마다 매번 전체 데이터를 탐색해야 되는데, 이는 불필요한 입출력을 반복해서 발생시키게 되므로 성능 면에서 바람직하지 않다.

실험에서 사용한 그레이스 해시 조인은 위의 단순 해시 조인 알고리즘을 개선한 것으로, 해시 함수 F1을 설정하여 두 개의 테이블을 모두 F1에 의해 생성된 해시 키 값에 따라 파티션으로 분할하여 디스크에 기록한다. 파티션이 끝나면 같은 해시 키를 갖는 파티션 쌍을 선택하여 기준 테이블은 해시 함수 F2를 사용하여 해시 테이블을 생성하고 탐색 테이블은 해당 테이블을 탐색, 일치하는 값을 찾아낸다. 이는 각 파티션이 F1에 의해 분리 되어 있기 때문에 중복된 데이터를 반복해서 탐색하지 않도록 해주어 불필요한 입출력이 발생하지 않도록 해준다[2]. 이때 각각의 파티션의 기준 테이블은 주어진 버퍼 공간 안에 들어오도록 균등하게 나뉘어야 된다. 만약 나뉘어진 기준 테이블이 주어진 버퍼 공간보다 클 경우 해당 파티션을 처리하는 방법에 따라 중첩 반복 해시 조인(nested loop hash join), 정렬 병합 해시 조인(sort-merge hash join), 재귀 해시 조인(recursive hash join) 등으로 불리는데 이는 추가적인 입출력을 발생 시키게 되므로 성능의 저하를 가져온다.

2.2 플래시 메모리 SSD

디스크 기술이 한계에 봉착함에 따라 플래시 메모리 SSD가 차세대 저장 장치로 주목 받고 있다. 플래시 메모리 SSD는 디스크와 달리 기계 장치가 없기 때문에 충격에 강하고 임의 접근 시간도 빠르다. 이에 따라 휴대용 기기 뿐만 아니라 고성능 저장 장치를 요구하는 OLTP 환경의 데이터베이스 서버에서도 플래시 메모리 SSD를 사용하는 사례가 늘어나고 있다.

최신 플래시 메모리 SSD는 성능 향상을 위해 내부적으로 병렬 구조를 취하고 있다.[3] 상위 소프트웨어에서 SSD의 병렬성을 최대한 활용하는 것이 최적화의 중요한 요소이다. 따라서 디스크 기반으로 설계된 기존의 데이터베이스가 기계적 회전과 헤드의 움직임의 최소화하려는 최적화방식을 SSD 관점에서 재고찰할 필요가 있다.

3. 실험 환경

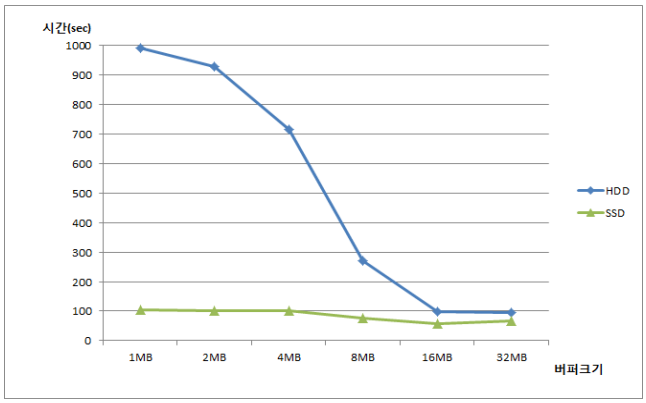
실험 하드웨어는 1.98 GHz AMD 애슬론 듀얼 코어 프로세서와 1.7 GB 메모리가 설치된 시스템을 사용하였다. 저장 매체로는 각각 Seagate 250 GB, 7200rpm 하드 디스크와 32 GB Intel SLC 플래시 SSD를 모두 SATA2 인터페이스로 연결하여 사용하였다. 실험은 64비트 리눅스 운영체제(커널 버전 2.6.35)에서 이루어졌으며, 멀티스레드 프로그래밍에는 리눅스에서 제공하는 pthread 라이브러리를 사용하였다. 저장 매체에 읽고 쓸 때에는 파일 시스템을 별도로 거치지 않고 open, read, write 등의 시스템 호출에 O_DIRECT 플래그를 두어 블록 장치로 직접 입출력을 수행하였는데, 이는 파일 시스템 캐시가 사용되어 요청한 IO가 실제로 발생하지 않는 경우를 방지함으로써 IO 성능 측정을 불투명하게 만드는 요소를 제거하기 위함이다.

조인에 사용한 테이블은 TPC-H 2.13.0의 LINEITEM 테이블과 ORDERS 테이블이며, 조인에 사용한 키 필드는 ORDERKEY를 사용하였다. 키 필드는 4 바이트 Integer 형으로, 나머지는 임의의 데이터를 덧대어 총 128 바이트 고정폭 레코드로 변환하였다. LINEITEM 테이블은 6,001,216 개의 128 바이트 레코드를 가진 732 MB 테이블로 변환하였으며, ORDERS 테이블은 1,500,000 개의 128 바이트 레코드를 가진 183 MB 테이블로 변환하였다. 실험에서 구현한 그레이스 해시 조인(grace hash join) 알고리즘의 동작 원리에 따라 이 중 크기가 더 작은 ORDERS 테이블이 기준 데이터(build input)로 선택 되도록 구현하였다. 파티션 분할이 끝난 후 탐색 단계에서도 각각의 인풋을 선택할 때 더 작은 파티션이 기준 테이블(build input)로 선택 되도록 Dynamic role reversal을 구현하였다.

4. 본문

4.1 읽기 버퍼 크기에 따른 성능 변화

그림 1은 해시 조인에서 사용되는 읽기 버퍼의 크기를 1M부터 32M까지 변화시켜가면서 SSD와 HDD에서의 성능을 측정한 것이다. 읽기 버퍼는 파티션 크기를 결정짓기 때문에, 읽기버퍼가 커지면 각 파티션의 크기가 늘어나 개수는 줄어든다. 이에 따라 랜덤 접근이 줄어든다.

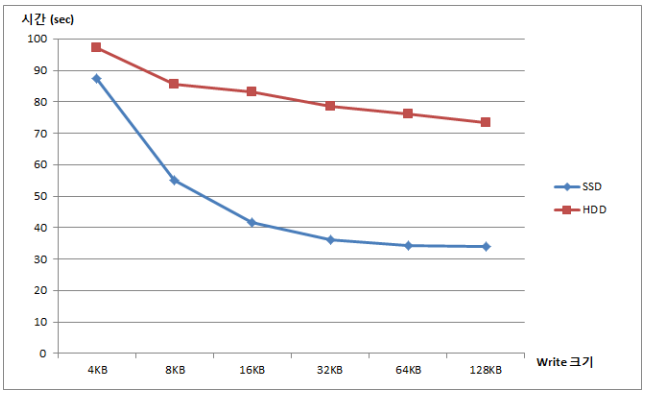


(그림 1) 읽기 버퍼 크기에 따른 성능 변화

실험 결과 HDD는 버퍼 크기가 1MB일 때와, 16MB일 때 약 10배의 성능 차이를 보였다. 버퍼 크기가 작을수록 랜덤 접근이 많이 발생하기 때문에, 랜덤 접근 성능이 취약한 하드 디스크는 버퍼 크기에 민감한 특성을 보인다. 반면 SSD는 버퍼크기에 큰 영향을 받지 않는 모습을 보이는데 이는 기계적 지연이 없는 SSD 특성 상, 접근 패턴에 따라 큰 성능 차이가 없기 때문으로 해석 된다.

4.2 쓰기 버퍼 크기에 따른 성능 변화

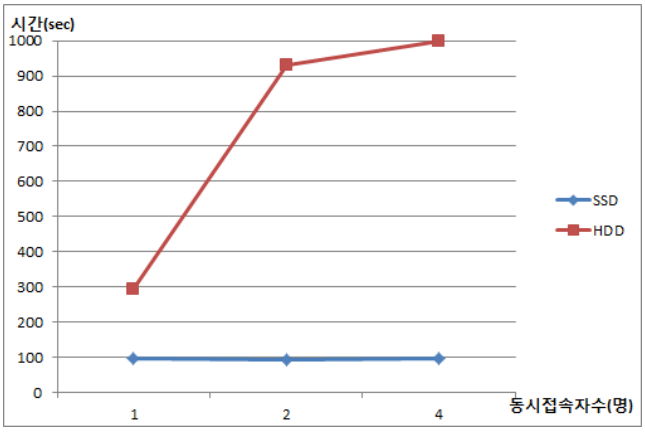
그림 2는 해시 조인에서 사용되는 쓰기 버퍼의 크기를 4K부터 128K까지 변화 시켜가면서 SSD와 HDD에서의 성능을 측정한 것이다. 실험하는 동안 읽기 버퍼의 크기는 16M로 고정하였다.



(그림 2) 쓰기 버퍼 크기에 따른 성능변화

실험결과 SSD는 최신 SSD의 특징인 병렬성의 가동에 따라 4K에서 64K까지 약 2배(100%)의 성능의 증가가 보였다. HDD도 IO유닛이 커지면서 디스크 헤드의 기계적인 지연시간이 줄어들어 약 20%의 성능의 향상이 있었으나 SDD의 성능의 증가를 따라오지는 못하였다.
한편, SSD는 쓰기버퍼 64K 시점에서 더 이상의 성능 증가가 보이지 않았는데 이 시점에서 실험에 사용한 SSD의 병렬성이 최대한으로 발휘된 것으로 볼 수 있다.

4.3 동시 사용자 수에 따른 성능 변화



(그림 3) 동시 사용자수에 따른 평균 성능 변화

그림 3은 SSD와 HDD에서 동시에 접근 하는 사용자를 1명에서 4명까지 변화시키며 각 조인별 평균 성능을 측정해 본 것이다. 실험 하는 동안 총 읽기버퍼의 크기는 8M로, 쓰기 버퍼의 크기는 4K로 고정하였다.

동시 사용자수가 늘어날수록 다수의 사용자가 한정된 버퍼를 공유하기 때문에 개별 사용자에게 할당되는 가용 버퍼 감소한다. 그림 3에서 사용자가 2명일 때 읽기버퍼의 크기는 각 4M씩 할당 되고 이때 SSD와 HDD의 성능 차이는 약 9배 이상인 것으로 관찰된다. 그림 1에서 읽기 버퍼가 4MB일 때 성능 차이가 7배였던 점을 감안하면, 동시적인 랜덤 접근이 많아지는 환경에서 HDD의 한계가 추가적인 성능 차이를 가져온 것으로 보인다.

5. 결론

조인 알고리즘은 데이터베이스 관리 시스템에서 질의 처리에 자주 사용되며 시간이 많이 걸리는 연산이다. 조인은 자주 사용되는 만큼 그 속도를 높이기 위해 많은 방법이 연구되었는데, 그레이스 해시 조인은 그 중 한 방법이고 이 논문에서는 그레이스 해시 조인을 직접 구현하여 HDD와 SDD에서 비교 테스트를 함으로 좀더 SSD에 최적화 된 방법을 연구해 보았다.

이 논문에서는 실험을 위해 해시 조인에 사용되는 읽기 버퍼의 크기와 쓰기 버퍼의 크기 그리고 동시 사용자의 수를 변화 시켜가면서 SSD와 HDD에서의 수행 시간을 비교 해 보았다. 실험 결과 SSD는 읽기 버퍼의 크기가 작은, 즉 파티션의 크기가 작고 수가 많은 상황에서 랜덤 접근이 많이 발생함에도 균일한 성능을 보여주었고, 쓰기 버퍼의 크기 변화에서는 SSD의 병렬성을 사용할 수 있는 시점까지 성능의 큰한 성능을 보여주었동시 사용자의 수가 늘어난 경우에도 HDD가 사용자당 버퍼크기가 있는든 것 이상으로 성능의 저하를 보이는데 반해 SSD는 성능의 저하를 보이지 않았다.

이와 같은 실험들을 통해 HDD와 SDD를 비교해 본 결과 동시에 여러 명령을 처리하기위해 랜덤 접근이 늘어나는 OLTP 환경에서 SSD의 우위성을 확인할 수 있었다. 또 SSD에서의 해시조인 성능의 최적화를 위해 읽기 버퍼의 크기보다 SSD 자체의 특성인 병렬성을 최대화 시킬 수 있도록 쓰기 버퍼의 크기를 조절하는 것이 효과 적임을 확인 할 수 있었다.

이 논문에서는 성능에 영향을 미치는 요인을 세가지 상정하여 실험을 진행 했는데 앞으로 SSD에서 좀 더 최적화 할 수 있도록 더 많은 요인을 고려한 실험이 필요하다. 또 이번 실험에 사용된 그레이스 해시 조인보다 최신 해시 조인 알고리즘인 하이브리드 해시 조인(Hybrid hash join)에 대한 성능 평가를 수행할 계획이다.

참고문헌

[1] Mark Gurry and Peter Corrigan. “Oracle Performance Tuning”. O’REILLY

[2] LEONARD D. SHAPIRO. "Join Processing in Database Systems with Large Main Memories". ACM Transactions on Database Systems, Vol. 11, No. 3, Sep 1986, Pages 239-264

[3] Yoon Jae Seong, Eyee Hyun Nam, Jin Hyuk Yoon, Hongseok Kim, Jin-Yong Choi, Sookwan Lee, Young Hyun Bae, Jaejin Lee, Member, IEEE, Yookun Cho, Member, IEEE, and Sang Lyul Min, Member, IEEE. “Hydra: A Block-Mapped Parallel Flash Memory Solid-State Disk Architecture”. IEEE TRANSACTIONS ON COMPUTERS, VOL. 59, NO. 7, JULY 2010 905