플래쉬 메모리 SSD 기반 해쉬 조인 알고리즘의 성능 평가

(Performance Evaluation of Hash Join Algorithm on Flash Memory SSDs)

박 장 우 [†] 박 상 신 [††] 이 상 원 [†††] 박 찬 익 [††††]
(JangWoo Park) (SangShin Park) (SangWon Lee) (ChanIk Park)

요 약 데이터베이스 관리 시스템의 핵심 알고리즘인 해쉬 조인은 해싱을 위한 메모리가 부족한 경우(즉, 해쉬 테이블 오버플로우) 디스크 입출력를 유발하게 된다. 하드디스크를 임시 저장공간으로 사용할 경우, 해쉬 조인의 probing 단계에서 과도한 임의 읽기로 인해 I/O 시간이 성능을 저하시키게 된다. 한편, 플래시메모리 SSD가 저장장치로 각광을 받고 있으며, 머지않아 엔터프라이즈 환경에서 하드디스크를 대체할 것으로 예상 된다. 하드디스크와 달리, 기계적인 동작 장치가 없는 플래시메모리 SSD의 경우 임의 읽기에서 빠른 성능을 보이기 때문에 해쉬 조인의 성능을 크게 향상시킬 수 있다. 본 논문에서는 플래시 메모리 SSD를 해쉬 조인을 위한 임시 저장공간으로 사용할 경우의 몇 가지 중요하고 현실적인 이슈들을 다룬다. 우선, 해쉬 조인의 I/O 패턴을 자세히 설명하고, 하드디스크에 비해 플래시메모리 SSD가 수십 배에 가까운 성능 향상을 보이는 이유를 설명한다. 다음으로, 클러스터 크기(즉, 해쉬 조인 알고리즘에서 사용하는 I/O 단위)가 성능에 미치는 영향을 제시하고 분석한다. 마지막으로, 하드디스크의 경우, DBMS의 질의 최적화기가 산출하는 비용이 실 수행시간과 편차가 클 수 있는데 반해, 플래시메모리 SSD의 경우 비용 산출을 정확히 하게 됨을 실험적으로 보인다. 결론적으로, 플래시메모리 SSD를 해쉬 조인을 위한 임시 저장공간으로 사용할 경우, 빠른 성능과 더불어 질의 최적화기의 비용 산출이 훨씬 더 신뢰할 수 있음을 보인다.

키워드 : 플래시메모리, 해쉬 조인, 오버플로우, 클러스터, 질의 최적화기

Abstract Hash join is one of the core algorithms in databases management systems. If a hash join cannot complete in one-pass because the available memory is insufficient (i.e., hash table overflow), however, it may incur a few sequential writes and excessive random reads. With harddisk as the tempoary storage for hash joins, the I/O time would be dominated by slow random reads in its probing phase. Meanwhile, flash memory based SSDs (flash SSDs) are becoming popular, and we will witness in the foreseeable future that flash SSDs replace harddisks in enterprise databases. In contrast to harddisk, flash SSD without any mechanical component has fast latency in random reads, and thus it can boost hash join performance. In this paper, we investigate several important and practical issues when flash SSD is used as tempoary storage for hash join. First, we reveal the I/O patterns of hash join in detail and explain why flash SSD can outperform harddisk by more than an order of magnitude. Second, we present and analyze the impact of cluster size (i.e., I/O unit in hash join) on performance. Finally, we emperically demonstrate that, while a commerical query optimizer is error-prone in predicting

· 본 논문은 서울시의 지원으로 수행한 서울시 산학연 협력사업(PA090903)의 연구 결과입니다.
· 본 논문은 언어지식베이스(ITRC 지능형 HIC융합 연구센터 3 세부과제, 2005.9 ~ 2013.8, IITA)의 연구 결과입니다.

† 비 회 원 : 성균관대학교 임베디드 소프트웨어학과
sliod@naver.com
†† 학생회원 : 성균관대학교 임베디드 소프트웨어학과
park.sangsin@gmail.com
††† 정 회 원 : 성균관대학교 정보통신공학부 교수
wonlee@ece.skku.ac.kr
†††† 비 회 원 : 삼성전자 Flash WS 그룹 수석연구원
ci.park@samsung.com

논문접수 : 2010년 6월 25일
심사완료 : 2010년 9월 13일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제11호(2010.11)

the execution time with harddisk as temporary storage, it can precisely estimate the execution time with flash SSD. In summary, we show that, when used as temporary storage for hash join, flash SSD will provide more reliable cost estimation as well as fast performance.

Key words : Flash memory, Hash join, Overflow, Cluster, Query Optimize

1. 서 론

조인 연산은 데이터베이스에서 가장 자주 사용되고 시간이 많이 걸리는 핵심 알고리즘이며, 대표적으로 중첩반복 조인(nested loop join), 병합정렬조인(merge-sort join), 해쉬 조인(hash join) 등의 세 가지 방법이 사용된다. 그 중 해쉬 조인은 일반적으로 가장 좋은 성능을 보이는 것으로 알려져 있다[1]. 그런데, 해쉬 조인의 문제점 중의 하나는 조인 수행 시 필요로 하는 메모리가 부족할 때 발생하는 해쉬 테이블 오버플로우(hash table overow)현상이며, 이로 인하여 질의 최적화기의 부정확한 비용 산출과 조인 수행 도중 메모리 부족 등의 문제점을 야기한다[2]. 이 문제점을 피하기 위하여 해쉬 조인에서는 하드디스크를 이용한 오버플로우 분해(resolution) 또는 오버플로우 방지(avoidance) 기법을 사용한다[3]. 그럼에도 불구하고, 해쉬 테이블 오버플로우가 발생한 경우, 조인을 위한 임시 저장장치로의 디스크 입출력이 유발되며, 특히, 이때 발생하는 입출력 유형은 주로 순차 쓰기(sequential write)와 임의 읽기(random read)이다. 따라서, 하드디스크를 조인을 위한 임시 저장장치로 사용하는 경우, 임의 읽기를 위해 기계적인 장치인 디스크 헤드가 많이 움직여야 하고 그로 인해 급격한 성능 저하 현상이 일어난다.

한편, 플래시메모리는 빠른 접근 지연 시간, 내구성, 가벼운 무게, 작은 크기, 저전력 등의 장점들로 인하여 스마트폰, MP3 플레이어, 노트북, 이동식 디스크 등 여러 분야에서 각광받고 있다. 플래시메모리 관련 기술이 빠른 속도로 발전하고 있기 때문에 머지않아 엔터프라이즈 환경의 하드디스크를 대체할 것으로 예상된다. 특히, 하드디스크와 달리, 기계적인 동작 장치가 없는 플래시 메모리 SSD는 임의 읽기에 있어 아주 뛰어난 성능을 제공하기 때문에, 조인 시 발생하는 임시데이터의 저장 공간으로 해쉬 조인의 성능을 크게 향상시킬 수 있다[4]. 본 논문에서는 플래시메모리 SSD를 해쉬 조인을 위한 임시 저장 공간으로 사용할 경우의 몇 가지 중요하고 현실적인 이슈들을 다룬다. 우선 해쉬 조인의 동작 원리 및 I/O 패턴을 자세히 설명한 후, 성능 측정을 위한 몇 가지 수학 모델을 해쉬 조인 수행 과정에 기반하여 제시하고 실험한다. 이를 통해 하드디스크에 비해 플래시메모리 SSD가 수십 배에 가까운 성능 향상을 보이는 이유를 설명한다. 또한, 해쉬 조인에서 사용하는 I/O 단위인 클러스터 크기에 따라, 하드디스크와 플래시메모리 SSD의 성능의 변화를 보이고 그 이유를 설명한다. 하드디스크 기반의 시스템에서 정렬(sort) 연산이 클러스터의 크기가 성능에 큰 영향을 줄 수 있으며, 또한 정렬과 해쉬의 대칭성으로 인하여 해쉬 역시 클러스터의 크기에 따라 성능에 영향을 받는 것이 이전 연구에서 확인되었다[3,5]. 하드디스크와 플래시메모리 SSD는 유리한 I/O 유형 및 탐색지연시간이 다르기 때문에 기존 하드디스크를 기반 한 것과 다른 분석이 필요하다. 플래시메모리 SSD를 기반으로 병합 정렬 연산의 클러스터 크기에 따른 성능 변화는 이미 확인되었다[4]. 본 논문에서는 플래시메모리 SSD를 기반으로 한 해쉬 조인 연산의 클러스터 크기에 따른 성능 변화를 분석한다. 마지막으로, 하드디스크의 경우 그 기계적인 특성으로 인해서 질의 최적화기(query optimizer)가 정확한 비용을 예측하기 힘든데 반해, 플래시메모리 SSD의 경우 해쉬 조인의 비용을 정확히 산출할 수 있음을 실험적으로 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 하드디스크와 플래시메모리 SSD의 특징을 비교하고, 상용 데이터베이스에서 해쉬 조인 연산의 동작원리를 설명하고, 기존 관련 연구를 고찰한다. 3장에서는 해쉬 조인시 I/O 패턴을 구체적으로 설명하고 이를 기반으로 해쉬 조인의 비용 모델을 제시한다. 4장에서는 임시 저장공간으로 하드디스크와 플래시메모리 SSD를 사용한 성능 평가 결과를 제시하고 분석한다. 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

2.1 하드디스크 와 플래시메모리 SSD의 특징

1장에서 설명한 것과 같이 하드디스크는 기계적으로 구성 되어있다. 따라서 데이터를 읽어들일 때 읽기/쓰기 헤드가 움직여야하며 디스크 회전 시간을 기다려야한다. 이러한 기계적 대기 시간은 초당 7200번 회전하는 하드디스크의 경우, 그 평균적인 시간은 표 1에 나타난다. 이에 반해 플래시메모리 SSD는 기계적인 부분이 없는 전자적 장치이기 때문에 하드디스크에 비해 대기 시간이 크게 작으며 데이터 위치에 구애받지 않고 항상 동일한 대기 시간에 입출력 요청을 마칠 수 있는 장점이 있다. 몇 년 전까지 하드디스크는 읽기와 쓰기 연산 모두가 평균적으로 동일한 대기 시간에 수행되는 반면, 플

표 1 하드디스크, 플래시 SSD I/O 성능 비교

저장 장치	하드디스크	플래시 SSD
평균 대기 시간	4.5ms	0.1ms
데이터 전송 속도	100MB/sec	230MB/sec(읽기) 180MB/sec(쓰기)

† 하드디스크: Western Digital WD1600AAJS;
† 플래시 SSD: Commercial Flash memory SSD(SLC)

래시메모리 SSD는 덮어쓰기 연산에 대한 취약점으로 인하여 읽기 연산이 빠르고 쓰기 연산이 느린 단점이 있었다. 하지만 근래 들어 플래시메모리 SSD의 FTL에 대한 연구가 계속 되었으며 그 결과, 현재는 기존의 단점이 많이 개선되어 읽기와 쓰기 연산이 비슷한 대기 시간에 수행된다. 뿐만 아니라 최신 플래시메모리 SSD의 경우 그림 1과 같이 다중 채널 구조를 확장함을 통해 초당 데이터 전송 속도에 대한 개선도 이루어져 두 연산 모두 하드디스크 보다 좋은 성능을 발휘하게 되었다.

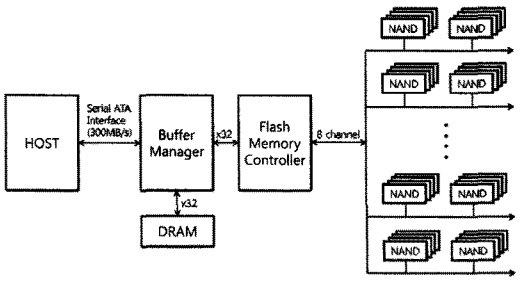


그림 1 일반적인 플래시메모리 SSD 아키텍쳐

2.2 상용 데이터베이스에서의 해쉬 조인 동작 원리

상용 데이터베이스에서 해쉬조인은 그림 2의 과정으로 동작한다. 먼저 테이블을 몇 개의 파티션으로 나눌 것인가를 결정한다. 파티션의 숫자는 대부분의 경우[1] 다음 공식에 따라 결정된다[6].

공식 1 : (Number of Partitions) · C ≤ Favm · M

파티션의 개수는 클러스터의 크기(C)와 사용 가능한 메모리의 크기(M)에 의하여 결정된다. Favm은 실질적으로 사용가능한 메모리의 크기를 구하기 위한 것으로 이를 통해 비트맵 벡터, 비동기 입출력 등 기타 용도로 사 용되는 메모리는 제외된다.

Favm은 일반적으로 0.8로 정의된다[6]. 해쉬 조인을 수행하는 환경에 따라 클러스터의 크기는 매번 달라지기 때문에 파티션 숫자를 계산하기 위해서 설계한 각각의 메모리 크기에 맞춰 적절한 클러스터의 크기를 구한다. 이에 대한 공식은 다음과 같다[7].

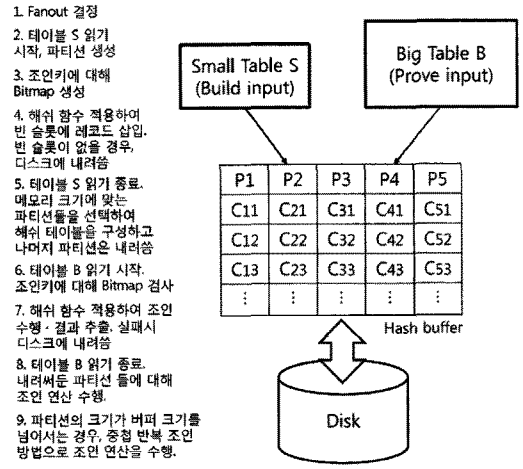


그림 2 해쉬 조인 동작 원리

공식 2 : R/M ≤ Po2(M/C)

R은 작은 테이블 S의 크기, M은 이전 공식에서 사용가능한 메모리(M)에 Favm을 적용한 값을 나타낸다. C는 해쉬 조인시 사용하는 클러스터의 크기로 데이터베이스 표준 블록 크기에 해쉬 조인 수행 시 한 번에 읽어오는 블록 숫자를 곱한 값이다. Po2는 괄호 안의 값보다 작은 2의 최대 거듭제곱을 나타낸다.

다음으로 작은 테이블 S를 읽어 들인다. 읽어 들인 레코드의 조인키 값에 첫 번째 해쉬 함수 h1을 적용하여 해당하는 파티션에 레코드를 삽입한다. 이때 두 번째 해쉬 함수 h2 역시 조인키 값에 미리 적용하여 추후 조인 단계에서 사용할 수 있도록 그 값을 저장해둔다. 또한 조인키 값에서 중복된 것을 제외한 유일한 값들에 대해 비트맵을 생성한다. 그림 2와 같이 각 파티션은 클러스터의 모임으로 구성되며 레코드를 각 파티션에 삽입할 때 파티션이 가득 차 빈 공간이 없는 경우, 이를 임시 저장공간에 내려쓴다. 이것이 해쉬 테이블 오버플로우 현상이다. 작은 테이블을 모두 읽어 들인 후 메모리에 해쉬 테이블을 생성할 수 있는 크기를 가진 파티션들을 찾아 이를 해쉬 테이블로 구성하고 나머지 파티션은 디스크에 내려쓴다. 해쉬 테이블을 구성할 때는 이전 단계에서 h2를 통해 미리 저장해둔 값을 사용한다.

그리고 나서 큰 테이블 B를 읽어들인다. 먼저 읽어들인 레코드의 조인키 값이 이전에 생성한 비트맵에 존재

[1] 일부 경우, 파티션의 숫자가 공식에서 산출한 것과 다를 수 있으나 그 오차는 크지 않다.

하는지 확인하여 존재하지 않으면 해당 레코드 값을 무시한다. 비트맵을 통과한 레코드는 그 조인키 값에 h1을 적용하여 해당하는 파티션을 찾고 h2 역시 적용하여 값을 추출, 미리 구성한 해쉬 테이블과 조인 연산을 수행한다. 조인에 성공한 경우에는 결과 값을 출력하고 반대로 조인에 실패한 경우는 앞서 S 테이블을 읽어들이는 과정에서 해당 레코드가 속한 파티션을 디스크에 내려썼는지 확인, 내려썼다면 해당 레코드 역시 디스크에 내려쓰고 내려쓰지 않았다면 해당 레코드 값을 무시한다.

B 테이블을 모두 읽어들였으면 마지막으로 디스크에 내려써둔 값에 대한 조인 연산을 수행한다. 이 때 테이블 S,B의 동일한 파티션을 읽어들이되 테이블의 순서에 관계없이 크기가 작은 파티션을 먼저 메모리로 읽어들인다. 그 후 나머지 파티션을 읽어 들이며 위와 동일하게 조인 연산을 수행한다. 메모리의 크기가 작아 테이블 S의 파티션도 메모리에 들어가지 않는 경우, 해당 파티션을 보다 작은 단위로 나눠 읽어들이며 중첩 반복 조인과 유사한 방법으로 연산을 수행한다[7].

2.3 관련 연구

2.3.1 데이터 분포에 따른 하드디스크 탐색 시간 변화

하드디스크는 두 세장의 디스크 판으로 구성된다. 각 디스크 판은 원형의 트랙으로 나누어지고 트랙은 다시 섹터로 나누어지는데 이 섹터에 데이터가 기록된다. 각 디스크 판은 읽기/쓰기 헤드를 가지고 있으며 데이터를 읽거나 쓸 때는 이 헤드가 트랙사이를 움직여 원하는 데이터가 기록된 트랙으로 이동된다. 헤드가 이동하는 거리(seek distance)에 따라 데이터 탐색 시간이 달라진다[8]. 그 후 디스크 판이 회전되면서 원하는 데이터를 읽거나 쓴다. 이러한 하드디스크의 기계적인 구성은 임의로 퍼져있는 데이터를 읽어들일 때, 동일한 회수의 I/O가 발생하더라도 각 데이터의 디스크 상의 위치 분포에 따라 소요되는 시간이 달라지는 현상을 유발할 수 있다.

2.3.2 기존 연구와의 차이점

앞서 언급 했듯이, 현재의 대부분의 하드디스크에서 최소한의 물리적인 I/O 단위는 섹터이며, 운영체제나 데이터베이스시스템 등에서는 일반적으로 여러 개의 연속된 섹터들을 구성된 클러스터라는 단위로 I/O를 수행한다. 지금까지 주로 하드디스크를 저장 장치로 사용해 온 데이터베이스 분야에서, 클러스터 크기가 데이터베이스의 특정 알고리즘의 성능에 영향을 미치는 것으로 알려져 있다. 예를 들어, 대용량 데이터를 외부 정렬(external sorting)하는 경우가 대표적이다. 외부 정렬에서, 클러스터 크기가 작으면 정렬 연산 시 한 번에 병합되는 데이터 집합이 많기 때문에 정렬 단계가 줄어드는 반면, 클러스터의 크기가 크면 한 번에 병합되는 데이터 집합이 줄어들고 정렬 단계가 늘어나 그에 따른 시간 지연이 발생한다. 하지만 클러스터가 크면 한 번에 보다 많은 데이터를 가져올 수 있고 디스크 탐색 횟수가 줄어들어 이로 인한 시간 단축 또한 발생하기 때문에 상황에 따라 클러스터의 크기를 적절하게 조절 해야한다. 플래시메모리 SSD를 정렬을 위한 임시 저장공간으로 사용할 경우, 병합 정렬 연산의 클러스터 크기에 따른 성능 변화를 측정한 연구도 존재한다[4].

한편, 해쉬 조인과 관련해서, Graefe는 하드디스크를 임시 저장장치로 사용하는 경우, 클러스터 크기가 클수록 해쉬 조인의 성능에 유리함으로 보였고[3], 하드디스크와 플래시메모리 SSD를 임시 저장공간으로 사용하여 해쉬 조인 연산의 성능을 비교한 기존 연구가 존재한다[9]. 특히, [9]에서는 adhoc 조인 알고리즘[10]을 기반으로 여러 조인 연산을 구현하여 다양한 상황에 대하여 실험하였고, 하드디스크의 경우 전체 조인 수행 시간에서 입출력 시간이 가장 큰 비중을 차지한 반면에 플래시메모리 SSD의 경우 중앙 처리 장치의 처리 시간이 가장 큰 비중을 차지하는 사실을 확인하여 플래시메모리 SSD가 해쉬 조인에 있어서 하드디스크보다 유리함을 보였다.

본 논문의 연구 내용은 다음 두 가지 측면에서 기존 연구들과 차별화된다. 우선, ad hoc 알고리즘에 기반 한 해쉬 조인의 입출력 유형은 순차 읽기와 임의 쓰기인데 반해, 본 논문에서 대상으로 하는 해쉬 조인의 입출력 유형은 순차 쓰기, 임의 읽기로 I/O 유형이 다르다. 이러한 차이는 임의 읽기에 유리하고 임의 쓰기에 불리한 플래시메모리 SSD의 특성상 해쉬 조인의 성능에 차이를 유발한다. 다음으로, 기존 논문에서는 해쉬 조인에 사용하는 메모리를 별도로 할당하지 않았기 때문에 버퍼 교체 정책에 큰 영향을 받는 반면, 본 논문에서는 조인에 사용하는 메모리를 PGA(Program Global Area)에 별도로 할당하기 때문에 버퍼 교체 정책에 거의 영향을 받지 않고 해쉬 조인에서 유발되는 순수한 I/O의 회수와 양이 성능에 결정적인 영향을 미친다.

3. 해쉬 조인 IO 비용 모델

본 장에서는 2장에서 설명한 해쉬 조인의 동작원리를 바탕으로, 우선 해쉬 테이블 오버플로우가 발생 할 경우, 임시 저장공간에 발생하는 I/O 패턴을 설명한다. 그리고, probing 단계에서 메모리 크기에 따라 조인 테이블의 파티션을 반복적으로 읽어 들여야 하는 횟수를 산출하는 예측 모델을 설명한다. 마지막으로, 해쉬 조인 시 사용하는 I/O 단위인 클러스터 크기에 따른 탐색 횟수(seek count)를 산출하는 해쉬 조인의 I/O 비용 모델을 제시한다.

3.1 해쉬 테이블 오버플로우 I/O 패턴

그림 3은 작은 테이블 S를 90MB로 설정하고 큰 테이블 B를 180MB로 설정한 상태에서 해쉬 테이블 오버플로우 현상이 발생한 경우 디스크 입출력 유형이다. 메모리 크기를 작게 설정했기 때문에 그림과 같이 먼저 두 테이블(S,B) 모두 거의 대부분의 파티션을 디스크에 순차적으로 내려쓴다. 가장 왼쪽의 순차 쓰기 유형은 그 때 발생한다. 그 후 작은 테이블 S의 각 파티션들에 대응되는 큰 테이블 B의 파티션 들을 임의 읽기로 읽어 들이면서 조인 연산을 수행한다. 그림 상단의 대부분을 차지하고 있는 임의 읽기 유형이 그것을 나타내고 있고 그 하단 부분의 빗금 형태의 순차 읽기 유형은 작은 테이블 S의 각 파티션을 순차적으로 읽어 들이는 것을 나타낸다.

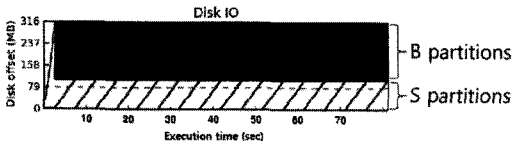


그림 3 해쉬 조인 수행 시 임시 저장공간의 I/O 패턴

3.2 파티션 읽기 횟수 예측 모델

실험에 앞서 각 테스트케이스의 성능을 예측 할 기준이 필요하다. 이번 장에서는 해쉬 조인 구동 원리에 기반한 수학적 모델을 설계한다. 또한, 메모리의 크기를 1MB에서 256KB까지 감소시켜 오버플로우 현상을 점점 증가시키는 7가지 테스트케이스를 제시하고 각 메모리 크기에 설계한 모델을 적용시켜 결과를 예측한다.

앞서 언급했듯이 해쉬 조인을 수행할 때 메모리의 크기가 작으면 각 파티션을 디스크로 내려쓴다. 추후 내려쓴 파티션을 다시 읽어들일 때 파티션의 크기보다 메모리의 크기가 작으면 중첩 반복 조인 방법을 이용하여 조인 연산을 수행한다. 일반적인 중첩 반복 조인은 반복 횟수가 늘어날수록 조인 수행 시간이 오래 걸리게 된다. 해쉬 조인도 중첩 반복 조인 방법을 사용하면 반복 횟수가 늘어나는 만큼 조인 수행 시간이 증가한다. 해쉬 조인에서 중첩 반복 횟수를 구하는 공식은 다음과 같다[6].

$$\text{공식 } 3 : N = (S/F)/Favm \cdot M$$

작은 테이블의 크기(S)를 파티션의 숫자(F)로 나누고 그것을 사용 가능한 메모리의 크기로 나누면 각 파티션에 대해 몇 번의 반복을 수행하는지를 구할 수 있다. 모든 파티션에 대해 동일한 횟수의 반복을 수행하므로 이는 곧 산출된 반복 횟수(N)만큼 큰 테이블을 반복하여 읽어들이는 것을 뜻한다.

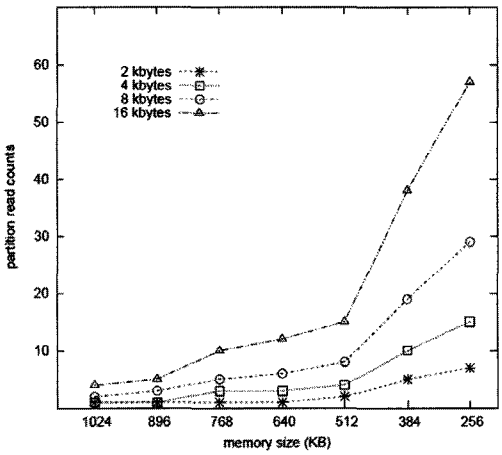


그림 4 메모리 및 클러스터 크기별 반복 읽기 횟수

그림 4는 다양한 메모리 크기 및 클러스터 크기에 공식 1,2,3을 적용하여 계산한 파티션 반복 읽기 횟수를 보여주고 있다. 메모리 및 클러스터 크기가 바뀌면, 2장에서 설명한 파티션 분할 공식에 따라 파티션의 숫자가 변하게 되고, 파티션 숫자가 변경되면 위 공식들에 의해 큰 테이블 B의 파티션을 읽는 횟수(N)가 달라지게 되므로 읽는 데이터의 양이 달라진다. 그림에서 알 수 있듯이, 메모리 크기가 작을수록 파티션 반복 횟수가 급격하게 증가함을 알 수 있고, 한편, 클러스터 크기에 비례해서 파티션 반복 횟수가 증감함을 알 수 있다. 플래시메모리 SSD에서 메모리 및 클러스터 크기에 따른 해쉬 조인 성능은 그림 4와 비슷할 것이라 예상된다.

3.3 클러스터 크기에 따른 탐색 횟수 예측 모델

지금까지는 메모리 크기와 클러스터 크기에 따른 파티션의 반복 횟수를 예측하는 모델을 설명하였다. 하지만, 이 모델은 해쉬 조인시 발생하는 I/O의 양을 산출하는 공식으로써 수행시간과 직접적인 상관관계는 없다. 대신, 이 예측 모델을 바탕으로 해쉬 조인시 필요한 디스크 탐색 횟수(Seek Count)를 정확하게 예측하는 모델이 필요하다.

그림 3과 같이 오버플로우 현상이 발생했을 때 그 수행시간에 가장 큰 영향을 미치는 요소는 테이블을 읽어들이는 읽기 시간이다. 즉, 해쉬 조인의 가장 큰 시간 비용 요인은 임의 읽기 부분이다. 데이터를 임의로 읽어들이면 읽어들이는 클러스터 마다 매번 탐색이 일어나고 각 탐색마다 탐색시간, 회전 지연시간, 데이터 전송시간 등의 시간 지연 요소가 발생하게 되어 순차 읽기에 비해 성능이 떨어진다. 이러한 사실에 기반하여 다음과 같은 수학적 가정을 세운다. 각 테스트케이스에 대한 해쉬 조인 성능은 임의 읽기에서 발생한 탐색 횟수의 합과 비례한다. 따라서 작은 테이블을 S, 큰 테이블을 B라 할 때,

해쉬 조인의 성능은 다음 공식을 통해 예측할 수 있다.

공식 4 : Total S·C = Table S S·C+Table B S·C×N

S·C는 클러스터 탐색 횟수(Seek Count)를 의미하고, N은 공식 1에 따라 큰 테이블 B의 파티션이 읽어들여지는 횟수를 나타낸다. 각 테이블을 임의 읽기로 읽어들이기 때문에 전체 테이블의 크기를 클러스터 크기로 나누어 해당 테이블의 탐색횟수를 구한다. 그림 5는 공식 4를 통해 계산한 각 메모리 및 클러스터 크기별 탐색 횟수를 보여주고 있다.

메모리 크기가 줄어들수록 오버플로우 현상이 증가하여 탐색 횟수가 늘어나고, 해쉬 조인의 수행시간은 탐색 횟수에 비례한다. 그림 5(a) 그래프는 하드디스크 성능에 영향을 주는 탐색 횟수와 플래시메모리 SSD 성능에 영향을 주는 N을 반영하고 있으므로 하드디스크와 플래시메모리 SSD 모두 그림 5(a)와 유사한 경향의 성능 저하가 나타날 것이다.

이제 클러스터 크기에 따른 해쉬 조인의 성능을 예측한다. 작은 테이블 S는 90MB, 큰 테이블 B는 180MB로 설정한다. 먼저 디스크 입출력에서 발생하는 시간 지연 요인을 각각 탐색 시간은 S.T, 회전지연 시간은 R.T, 데이터 전송 시간은 D.T라 정의하고 탐색 횟수를 X축, 조인 수행 시간을 Y축이라 정의한다.

플래시메모리 SSD는 탐색 시간과 회전 지연 시간이 무시되므로 2KB 클러스터로 10번 읽어들이는 것과 10KB 클러스터로 2번 읽어 들이는 것이 동일한 성능을 가진다. 따라서 조인 수행 시간은 다음과 같이 계산된다.

공식 5 : Y=(D.T)X

공식 5에서 플래시메모리 SSD에서는 클러스터 크기 변경은 성능에 거의 영향을 미치지 않는 것을 보인다. 다만 앞서 언급했듯이 클러스터 크기에 따라 전체 파티션을 읽어들이는 횟수가 달라지므로, 읽어들이는 데이터량 변화에 따른 성능 변화는 그림 4에 나타났다.

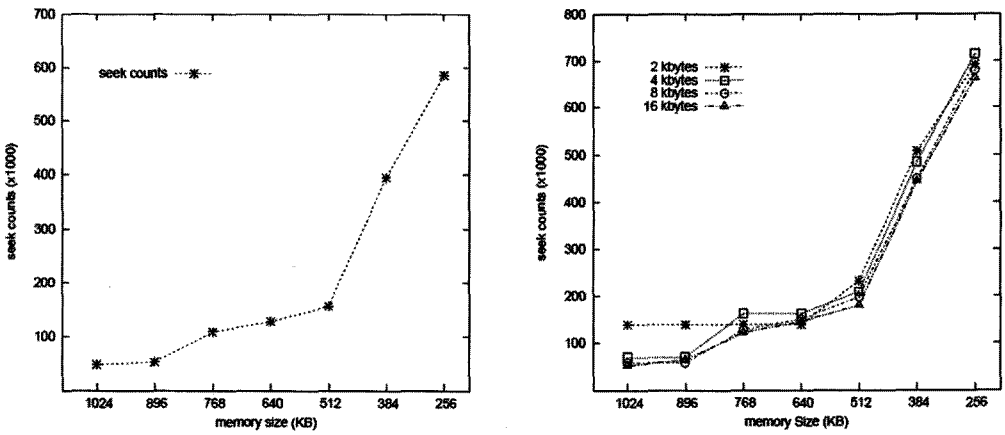
하드디스크의 경우, 각 탐색에 대하여 탐색 시간, 회전 지연 시간, 데이터 전송 시간 등의 3가지 요소 모두가 성능에 영향을 미치게 된다. 특히, 탐색을 위해 이동해야하는 트랙 수 등 탐색거리(seek distance)에 따라 탐색시간이 달라진다. 하드디스크의 조인 수행 시간은 다음과 같은 식으로 계산할 수 있다.

공식 6 : Y=(S.T + R.T + D.T)X

하드디스크는 탐색 시간과 회전 지연 시간에 큰 영향을 받는다. 따라서 클러스터의 크기가 작아져 탐색 횟수가 늘어나거나 크기가 커져 탐색 횟수가 줄어드는 경우 그에 따라 해쉬 조인 성능도 증감하게 된다. 클러스터의 크기가 변경되면 파티션을 읽는 횟수(N)도 변화하게 되므로 이 또한 탐색횟수에 영향을 미친다. 그림 5(b)는 클러스터 크기별로 계산한 탐색횟수이다. 하드디스크는 메모리 크기가 큰 경우에는 클러스터 크기가 크면 성능에 유리한 반면, 메모리 크기가 작은 경우에는 기존 연구 결과와 달리 클러스터의 크기에 상관없이 비슷한 성능을 낼 것으로 예측된다.

4. 성능평가

성능평가에서는 상용 데이터베이스의 임시 저장공간으로 하드디스크와 플래시메모리 SSD를 장착하고 3장에서 설정한 7가지 메모리 크기에 대하여 각각 해쉬 조



(a) 메모리 크기별 탐색 횟수(클러스터 크기: 4KB) (b) 클러스터 크기별 탐색 횟수 변화

그림 5 메모리 및 클러스터 크기별 탐색 횟수

인 수행 시간이 얼마나 걸리는지를 실험하고 결과를 분석한다. 또한, 동일한 메모리 크기에 대해 클러스터 크기를 변화시켜가며 조인 수행시간을 측정한 결과를 제시한다.

4.1 성능평가 환경

성능평가 환경은 다음과 같다. AMD 1.86GHz 듀얼코어 프로세서, 2GB 메모리가 장착된 Linux Kernel 2.6 시스템에 상용 데이터베이스를 설치하였다. 그리고 하드디스크와 플래시메모리 SSD를 각각 상용 데이터베이스의 임시 저장공간에 로우 디바이스(Raw Device)로 장착하였다. 하드디스크는 160GB용량에 7200RPM의 상용 하드디스크를 사용하였고, 플래시메모리 SSD는 50GB 용량에 데이터 전송속도가 230/180MBytes/s의 최신 상용 플래시메모리 SSD를 사용하였다. 그리고 두 저장장치 모두 SATA 인터페이스로 연결하였다. 3장에서 가정한 것과 같이 각각 90MB, 180MB 용량의 두 테이블을 조인 테이블로 사용하였으며 두 테이블 모두 4B 임의 키에 90 B 데이터로 구성된 94B 크기의 레코드 들로 이루어져있다. 실험도구로 Linux의 blktrace [11]를 사용하여 각 저장장치의 읽기/쓰기 트레이스를 추출하였고 해당 로그에 seekwatcher[12]를 사용하여 조인 수행시간을 측정하고 입출력 유형을 그래프화 하였다.

4.2 성능평가 결과

4.2.1 예측 결과와 성능 비교

그림 6은 플래시메모리 SSD와 하드디스크 각각을 임시 저장공간으로 사용하여, 메모리 크기를 1MB에서 256KB까지 128KB씩 감소시키면서 두 예제 테이블의 해쉬 조인 질의를 수행하는 데 걸리는 시간을 보여주고 있다.

그림 6(a)에서 알 수 있듯이, 임시 저장공간으로 플래시메모리 SSD를 사용하면 하드디스크를 사용한 것에 비해 9배에서 20배가량 좋은 성능을 보인다. 3장에서 기술한 바와 같이 메모리 크기가 줄어들수록 하드디스크가 플래시메모리 SSD에 비해 조인수행 시간이 크게 증가하는 것을 확인할 수 있다. 그림 6(b) 그래프는 플래시메모리 SSD의 수행시간을 상세히 확대하여 보여주고 있는데 3장의 그림 5 그래프의 기울기와 거의 완벽하게 일치한다. 즉, 예측한 비용과 성능이 일치한다. 반면, 하드디스크의 그래프(a)는 예측한 비용 그래프와 다른 기울기를 보인다. 그 이유는 하드디스크의 불규칙적인 초당 탐색 횟수에 기인한다. 실험도구인 seekwatcher [12]로 저장장치가 초당 몇 번의 읽기/쓰기 명령을 완료하였는지 확인할 수 있는데 이는 그림 7에 나타난다. 플래시메모리 SSD는 모든 메모리 크기에서 동일한 초당 탐색횟수를 보이는 반면, 하드디스크는 메모리 크기에

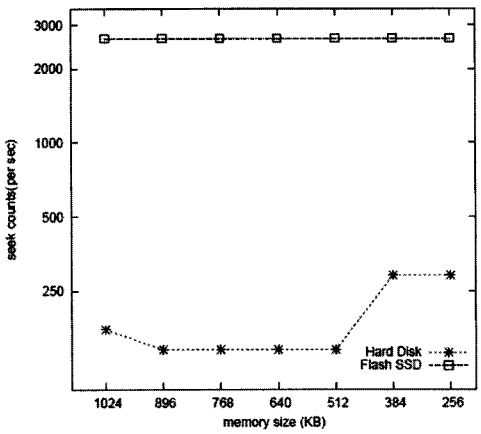
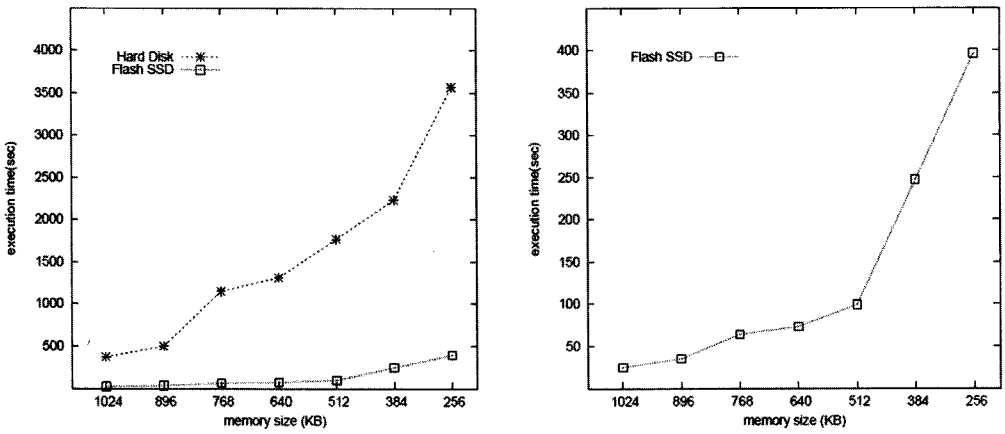


그림 7 HDD vs. SSD 초당 평균 탐색횟수



(a) HDD vs. SSD 수행 시간 비교 (b) SSD 수행 시간 (상세)

그림 6 메모리 크기에 따른 해쉬 조인 수행시간

따라 초당 탐색 횟수가 가변적이다. 이것은 메모리의 크기에 따라 디스크에 기록되는 데이터의 분포가 달라지고, 하드디스크의 실린더탐색 거리에 따라 탐색 시간이 크게 달라지기 때문이다[8]. 이러한 하드디스크의 불규칙적인 탐색 횟수로 인해 3장에서 산출한 비용과 다른 성능 그래프가 나타나게 된다.

4.2.2 질의 비용과 성능의 불일치 오류

그림 8은 각 저장장치별 계산 비용 그래프이다.

상용 데이터베이스의 질의 최적화기는 질의의 비용을 산정할 때 저장 장치로부터 수집한 I/O 성능 통계치를 바탕으로 계산하기 때문에 저장 장치에 따라 그래프의 절대값은 서로 다를 수 있다. 하지만 상대적인 측면에서 두 저장 장치 모두 동일한 기울기의 비용 그래프를 산출했다. 질의 최적화기는 저장 장치의 종류에 상관 없이 각 메모리 크기에 대해 동일한 비율의 성능 저하가 일어날 것이라 예상한 것이다. 하지만 하드디스크의 성능 저하는 질의최적화기가 예측한 성능 저하와 다른 기울기의 그래프를 보인다. 이러한 오류는 적절한 수행 질의를 탐색하는데 문제가 될 뿐더러 데이터베이스 관리자로 하여금 질의 최적화기가 질의에 대하여 부정확한 비용을 계산한다고 생각하도록 만들 수 있다. 반면, 플래시메모리 SSD는 초당 처리하는 입출력 숫자가 균일하기 때문에 하드디스크와 달리 질의 최적화기가 산정한 비용 그래프와 실제 성능 그래프가 일치한다.

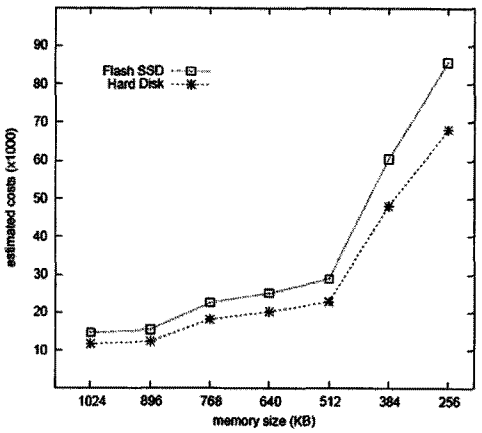
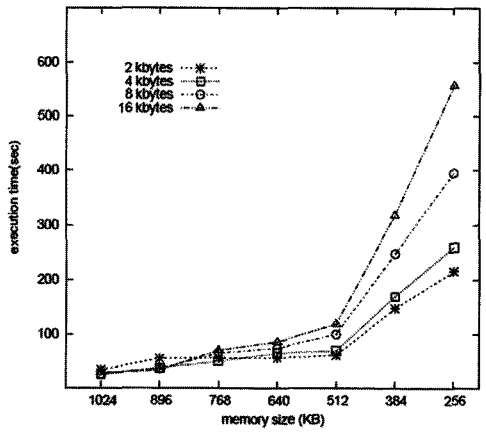


그림 8 질의 최적화기 산출 비용

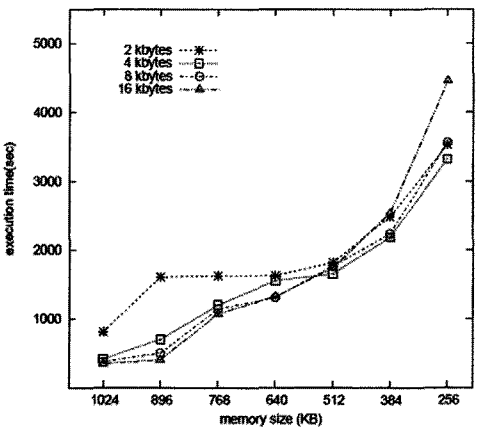
4.2.3 클러스터 크기에 따른 성능 변화

그림 9(a)는 플래시메모리 SSD의 클러스터 크기별 성능 그래프이다. 플래시메모리 SSD는 앞서 언급한 것과 같이 탐색횟수보다는 I/O 데이터의 양에 영향을 받는다. 3장에서 계산한 총 파티션 읽기 횟수 그래프 그림 4과 그림 9(a)의 성능 그래프를 비교하면 전반적인 경향이 비슷하다. 플래시메모리는 탐색 시간과 회전 지연시간이 없기 때문에 필요한 I/O 데이터의 양이 비슷하면 클러스터의 크기에는 거의 영향을 받지 않는다. 1MB-768KB 부분에서 결과가 나타난다. 반면, 메모리 크기가 줄어들수록 데이터의 양이 늘어나고 클러스터의 크기가 클수록 처리할 데이터의 양은 더욱 늘어나게 되어 성능이 떨어질 것이라 예상했는데 그것을 512KB-256KB 부분에서 확인된다. 결국, 플래시메모리 SSD는 클러스터의 크기가 작은 것이 해쉬 조인 성능에 유리하다.

그림 9(b)은 하드디스크의 클러스터 크기별 성능 그래프이다. 하드디스크는 탐색 시간과 회전 지연시간이 존재하기 때문에 클러스터의 크기가 클수록 성능에 유리하다. 그래프의 1MB-768KB에서 그러한 경향을 보인다. 다만 메모리가 작아지고 클러스터의 크기가 커질수록 처리해야하는 데이터의 총량도 늘어나기 때문에 클



(a) 플래시메모리 SSD



(b) 하드디스크

그림 9 클러스터 크기별 해쉬조인 성능 변화

러스터를 크게 사용해서 얻어지는 이점이 늘어나는 데이터량에 상쇄되어 512KB 부터 모든 클러스터 크기가 비슷한 성능을 나타낸다. 즉, 클러스터의 크기가 크다고 항상 유리한 것이 아니다. 이러한 결과는 앞에서 예상한 그림 5(b)와 전반적으로 일치한다. 데이터베이스 관리 시스템에서 임시 저장공간으로 하드디스크를 사용한다면 메모리가 충분할 때는 클러스터의 크기가 큰 것이 유리하지만 메모리가 부족할 때는 그렇지 않은 상황도 발생할 수 있으므로 사용 가능한 메모리 등 운영 환경을 고려한 적절한 클러스터 크기의 조절이 필요하다.

5. 결론 및 향후 연구

본 논문에서는, 해쉬 조인의 임시 저장공간으로 플래시메모리 SSD를 사용하면 하드디스크를 사용한 경우보다 9배에서 20배 정도의 성능향상을 있음을 보였고, 그 이유가 해쉬 조인의 성능에 큰 영향을 미치는 임의 읽기 때문임을 설명하였다. 또한, 임시 저장공간으로 하드디스크를 사용하는 경우, 하드디스크의 기계적 특성으로 인해 해쉬 조인의 성능과 질의 최적화기가 예측한 비용이 일치하지 않는 오류가 발생할 수 있는 반면, 플래시 메모리 SSD의 경우 실제 조인 성능과 예측 비용이 거의 일치함으로 보였다. 마지막으로, 클러스터 크기에 따른 각 저장장치별 성능 변화를 살펴보았는데, 플래시메모리 SSD는 작은 크기의 클러스터가 유리하다. 하드디스크는 사용가능한 메모리가 충분한 경우 클러스터가 클수록 유리하지만, 그렇지 않을 경우, 기존 연구 결과와 달리, 클러스터 크기에 무관하게 비슷한 성능을 나타내었다. 결론적으로, 다수의 사용자가 해쉬 조인을 사용하는 일반적인 데이터베이스 환경에서, 하드디스크 대신, 플래시메모리 SSD를 임시 저장공간으로 사용하고, 클러스터 크기 튜닝을 통해서 상당한 성능 개선 효과를 낼 수 있을 것이다.

지난 30여년 동안 데이터베이스 조인 비용 모델과 질의 최적화기는 하드디스크를 저장장치로 가정하고 개발, 발전되어 왔다. 따라서, 플래시메모리 SSD가 데이터베이스 저장장치로 사용될 경우에, 다양한 조인 방법들의 비용 모델들을 재고찰해야 하고, 비용 모델의 변화가 질의 최적화기에 미치는 영향에 대한 향후 연구가 필요하다.

참 고 문 헌

[1] L.D.Shapiro, "Join processing in database systems with large main memories," *ACM Trans.Database Syst.*, vol.11, no.3, pp.239-264, 1986.

[2] Hansjorg Zeller, Jim Gray, TANDEM COMPUTERS, "Hash join algorithms in a multiuser environment," *technical report 90.4*, Fed. 1990.

[3] G. Grafe, A. Linville, and L.D.Shapiro, "Sort versus hash revisited," *IEEE Transactions on Knowledge and Data Engineering*, vol.6, no.6, pp. 934-944, 1994.

[4] S.-W. LEE, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim, "A case for flash memory ssd in enterprise database applications," *in SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1075-1086, 2008.

[5] G.Graefe, Univ.of Colorado, Boulder, USA, Dept. Comput. Sci., "Parallel external sorting in volcano," *technical report 459*, June 1990.

[6] Center of Expertise, Oracle Worldwide Customer Support., "Hash joins, implementation and tuning release 7.3," *technical report*, Mar. 1997.

[7] Oracle, "Oracle enterprise manager database tuning with the oracle tuning pack," http://download.oracle.com/docs/B1050101/ em.920/a86647/index.htm.

[8] S. W.Schlosser, J. Schindler, S. Papado-manolakis, M. Shao, A. Ailamaki, and G. R.Christos Faloutsos, "On multidimensional data and modern disks," *Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies* - vol.4, pp.225-238, 2005.

[9] J. Do and J. M.Patel, "Join processing for flash ssds:remembering past lessons," *Data Management On New Hardware*, pp.1-8, 2009.

[10] L. M.Haas, M. J.Carey, M. Livny, and A. Shukla, "Sing the truth about ad hoc join costs," *The VLDB Journal-The International Journal on Very Large Data Bases*, vol.6, no.3, pp.241-256, 1997.

[11] "blktrace." http://linux.die.net/man/8/blktrace.

[12] "seekwatcher." http://oss.oracle.com/mason/seekwatcher

박 장 우

2009년 건국대학교 소프트웨어과 졸업 (학사). 2009년~현재 성균관대학교 임베디드 소프트웨어학과 석사과정. 관심분야는 데이터베이스 시스템 구조 및 개발, 플래시 메모리

박 상 신

2008년 한국기술교육대학교 컴퓨터공학과(학사). 2008년~2009년 한국정보통신기술협회 2009년~현재 성균관대학교 임베디드 소프트웨어학과 석사과정 관심분야는 데이터베이스 시스템 구조 및 개발, 플래시 메모리

이 상 원

1991년 서울대학교컴퓨터학과(학사). 1994년 서울대학교컴퓨터학과(석사). 1999년 서울대 학교컴퓨터학과(박사). 1999년~2001년 한국오라클. 2001년~2002년 이화여대 BK21 계약교수. 2002년~현재 성균관대학교 정보통신공학부 부교수. 관심분야는 flash memory DBMS

박 찬 익

1995년 서울대학교 컴퓨터공학과(학사) 1997년 서울대학교 컴퓨터공학과(석사) 2002년 서울대학교 전기컴퓨터공학과(박사). 2002년~현재 삼성전자 메모리사업부 수석 연구원. 관심분야는 Flash memory 기반 스토리지 아키텍처