플래시 메모리 환경에서 PostgreSQL 비용 모델의 성능 연구

이종백○ 이상원
성균관대학교
hundredbag@skku.edu, swlee@skku.edu

A Study on PostgreSQL Cost Model Performance on Flash Memory

Jongbaeg Lee○ Sang-won Lee
Sungkyunkwan University

요 약

PostgreSQL의 비용 기반의 옵티마이저(Cost-based Optimizer)는 비용 모델(Cost model)을 이용하여 생성 가능한 쿼리 실행 계획들의 예상 비용을 계산하고, 그 중 가장 저렴한 비용의 실행 계획으로 쿼리가 동작하도록 한다. PostgreSQL의 비용 모델에서는 하나의 페이지를 디스크로부터 읽는 비용을 나타내기 위해 random_page_cost와 seq_page_cost 파라미터를 이용한다. 하드디스크를 이용하도록 설계된 PostgreSQL에서는 임의 읽기를 의미하는 random_page_cost이 순차 읽기를 의미하는 seq_page_cost의 4배로 설정된다. 그러나 빠른 임의 접근 속도를 제공하는 플래시 메모리 SSD가 하드디스크를 대체함에 따라 PostgreSQL 비용 모델에서의 디스크 IO 파라미터, 저장 장치, 그리고 쿼리 실행 계획의 관계에 관한 연구의 필요성이 증가하였다. 본 논문에서는 저장장치로 하드디스크와 SSD를 이용할 때, random_page_cost 값을 변경해가며 TPC-H 벤치마크를 수행하고 그 결과를 분석한다. 성능 평가 결과, TPC-H 벤치마크 중 쿼리 실행 계획이 변경되는 쿼리의 경우, 하드디스크를 이용할 때와 SSD를 이용할 때 모두 순차 스캔이 우수한 성능을 보이지만, SSD에서 그 성능 하락이 크게 완화됨을 확인할 수 있다.

1. 서 론

데이터베이스 시스템에서 쿼리 옵티마이저는 SQL 쿼리를 효율적으로 수행하기 위한 최적의 실행 계획을 생성하는 역할을 한다. PostgreSQL[1]의 옵티마이저는 쿼리 실행 계획들의 예상 비용을 비교하고, 그 중 가장 저렴한 실행 계획을 최종 실행 계획으로 선택한다. 이 때, 비용의 산정을 위해 데이터베이스 내부의 오브젝트 통계뿐 아니라 시스템의 성능과 관련된 통계가 이용된다.

PostsgreSQL에서는 디스크 IO 비용 산정을 위해서 임의 읽기 비용을 의미하는 *random_page_cost*, 그리고 순차 읽기 비용을 의미하는 *seq_page_cost* 파라미터를 이용한다. 디스크 IO의 총 비용은 읽기 패턴에 따른 파라미터 값에 읽어야 하는 페이지의 수를 곱하여 계산된다. 두 파라미터의 기본 값은 하드디스크의 느린 임의 접근 속도를 고려하여 *random_page_cost*가 *seq_page_cost*에 비해 4배 크게 설정된다.

그러나, 새로운 주류 저장장치로 자리 잡고 있는 플래시 메모리 SSD는 하드디스크에 비해 높은 임의 접근 속도를 제공한다. 이러한 특징은 데이터베이스 시스템에서 임의 접근 기반의 스캔 방법인 인덱스 스캔(Index scan)이 순차 스캔(Sequential scan)에 비해 높은 성능을 보일 수 있는 기회를 제공할 수 있으며, 쿼리 옵티마이저에서 실행 계획을 선택할 때 이 점을 고려할 필요가 있다.

본 논문에서는 저장장치로 하드디스크를 이용할 때와 SSD를 이용할 때, *random_page_cost* 값에 따른 TPC-H 벤치마크를 이용하여 실행 계획 및 성능을 측정하고 그 결과를 분석함으로써 PostgreSQL 옵티마이저에서 디스크 IO 파라미터, 저장 장치 종류, 그리고 쿼리 실행 계획의 관계에 관하여 연구한다. 실험 결과, TPC-H의 22가지 쿼리 중 9가지에서 실행 계획이 변경되었으며, 하드디스크를 이용할 때와 SSD를 이용할 때 모두 순차 스캔을 이용하는 것이 인덱스 스캔에 비해 더 좋은 성능을 보이는 것으로 나타났다. 그 중, 느린 실행 계획과 빠른 실행 계획의 차이가 가장 심한 쿼리의 경우, 하드디스크에서 약 70배의 성능 차이가 나는 것에 비해, SSD에서는 약 6.5배의 성능 차이가 발생하는 것을 확인할 수 있었다. 이를 통해, 저장장치로 SSD를 이용하는 것이 잘못된 비용모델로 인한 성능 하락을 크게 완화시킴을 알 수 있다.

2. 관련 연구

2.1 PostgreSQL의 쿼리 수행과 옵티마이저

그림 1은 PostgreSQL에서 쿼리가 수행되는 과정을 나타낸다. 애플리케이션에서 PostgreSQL로 쿼리를 전달하면 파서(Parser)는 전달받은 쿼리의 문법을 확인하고, 쿼리를 트리 구조로 변환하여 옵티마이저에 전달한다. 옵티마이저는 전달받은 쿼리 트리를 이용해 실행 가능한 계획들을 생성하고, 계획들의 예상 비용을 비교하여 가장 저렴한 비용의 실행 계획을 쿼리 익스큐터(Executor)에 전달한다. 마지막으로, 쿼리 익스큐터는 실행 계획을 바탕으로 쿼리를 수행한다.

PostgreSQL에서는 비용 기반의 옵티마이저(Cost-based Optimizer)를 이용한다. 비용 기반의 옵티마이저는 비용 모델(Cost model)을 통해 쿼리 실행 계획들의 비용을 산정하고, 각 계획의 비용을 비교하여 가장 저렴한 비용의

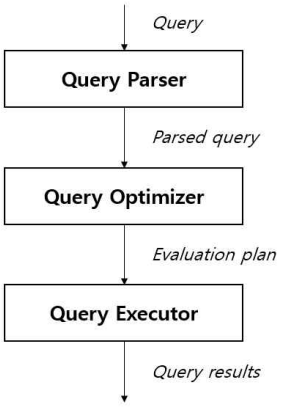

그림 1 PostgreSQL의 쿼리 수행 과정

실행 계획을 선택한다. 이 때, 비용의 산정을 위해 오브젝트 통계(레코드 개수, 블록 개수 등) 및 시스템 통계(CPU 속도, 디스크 IO 속도 등)가 이용된다.

표 1은 PostgreSQL의 비용 모델에서 이용하는 다섯 가지 시스템 통계 파라미터를 나타낸다. 그 중 디스크로부터 페이지를 읽어오는 것에 관하여 두 가지 파라미터가 사용된다. 첫 째로, *seq_page_cost*은 디스크 페이지를 순차적으로 읽을 때의 비용을 의미한다. 둘째로, *random_page_cost*는 디스크 페이지를 임의의 순서로 읽을 때의 비용을 의미한다. 두 파라미터의 기본 값은 *random_page_cost* = 4, *seq_page_cost* = 1로 설정되며, 디스크 IO의 총 비용은 읽기 패턴 파라미터와 읽어야 하는 페이지의 수를 곱하여 계산된다. 즉, 읽어야 할 페이지의 수가 동일할 때, PostgreSQL에서는 임의 읽기가 순차 읽기에 비해 4배 높은 비용으로 예측된다.

표 1 PostgreSQL 비용 모델의 비용 파라미터

파라미터	설명
seq_page_cost	순차적 디스크 페이지 읽기 비용
random_page_cost	랜덤 디스크 페이지 읽기 비용
cpu_tuple_cost	하나의 튜플을 처리하는 비용
cpu_index_tuple_cost	하나의 인덱스 엔트리를 처리하는 비용
cpu_operator_cost	연산의 수행 비용

2.2 플래시 메모리와 쿼리 실행 계획

PostgreSQL을 포함하는 기존 데이터베이스 시스템들 대부분은 하드디스크에서 동작하는 것을 가정하여 설계되었다. 이러한 점은 2.1절에서 설명한 것처럼 쿼리 옵티마이저에서 디스크에 페이지에 접근할 때, 임의 접근 방법의 비용을 순차 접근 방법의 비용보다 더 높게 책정하는 것으로부터 알 수 있다.

그러나, 플래시 메모리 SSD는 전자적인 방식으로 동작하기 때문에 기계적 부품으로 동작하는 하드디스크에 비해 매우 빠른 임의 접근 성능을 보인다. 데이터베이스 시스템에서 테이블의 데이터에 접근하는 대표적인 두 가지 방법은 디스크 페이지를 순차적으로 접근하는 순차 스캔, 그리고 인덱스를 통해 임의의 순서로 페이지에 접근하는 인덱스 스캔이다. 플래시 메모리 상에 동작하는 데이터베이스 시스템의 연구 중, [2]에서는 SSD의 빠른 임의 접근 속도라는 장점을 더욱 잘 활용하기 위하여 인덱스 스캔에서 중복된 페이지에 접근하는 문제를 제거함으로써 인덱스 스캔의 성능을 향상시킨다. 같은 관점에서, 본 논문에서는 SSD의 빠른 임의 접근 속도라는 장점이 쿼리 옵티마이저의 비용 모델에 적용되었을 때의 쿼리 실행 계획 변화 및 성능에 관하여 연구하고자 한다.

3. 성능 평가

성능 평가에서는 데이터베이스의 저장장치로 플래시 메모리 SSD와 하드디스크를 이용하는 각각의 경우에 대하여, PostgreSQL의 *random_page_cost*를 1, 2, 4, 8, 16, 32로 변화시켜가며 TPC-H 벤치마크의 쿼리를 수행한다. 또한

3.1 실험 환경

표 2 실험 환경

운영체제	Ubuntu 16.04.3 LTS
프로세서	Intel Xeon E5-2670V3 2.3GHz 24 Core (48 Thread)
저장장치	Samsung SSD 850 Pro 256GB WDC WD10EZEX 1TB
데이터베이스	PostgreSQL 9.4.15

본 논문에서는 WDC WD10EZEX 1TB 하드디스크와 Samsung SSD 850 Pro 256GB를 저장장치로 이용한다. 성능 평가에는 TPC-H[3] 벤치마크를 이용하였고, 데이터베이스의 크기는 1GB, 그리고 PostgreSQL의 버퍼 크기는 32MB로 설정하였다. 성능 평가를 위해 TPC-H의 벤치마크의 Q1부터 Q22까지 22가지의 쿼리 중 수행되지 않는 Q15를 제외한 21가지의 쿼리를 수행하였다. 자세한 실험 환경은 [표 2]과 같다.

3.2 실험 결과

그림 2는 저장장치로 하드디스크를 이용할 때와 SSD를 이용할 때의 *random_page_cost* 값에 따른 TPC-H 벤치마크의 수행 시간을 나타낸다. *random_page_cost* 값이 작을수록 디스크 페이지에 대한 임의 접근 비용이 낮게 산정되기 때문에 인덱스 스캔의 비용이 낮아지므로 순차 스캔 대신 인덱스 스캔을 이용하는 경향이 나타난다.

21가지의 TPC-H 쿼리를 수행한 결과, 그림 2에 나타난 9가지 쿼리를 제외한 나머지 쿼리에서는 실행 계획이 변경되지 않으며, 따라서 *random_page_cost*를 변경하여도 성능 차이가 나타나지 않는다.

성능 차이가 나타나는 쿼리들의 경우, 저장장치의 종

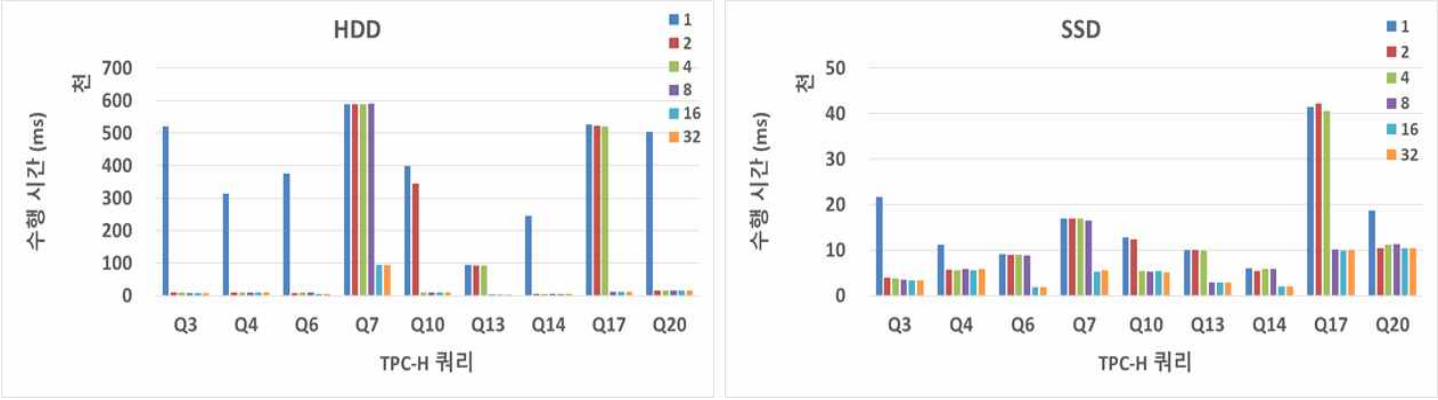

그림 2 저장장치 및 *random_page_cost* 값에 따른 TPC-H 벤치마크의 수행 시간

류에 관계없이 *random_page_cost*의 값이 작을 때 성능이 느려지는 경향이 나타난다. 쿼리 실행 계획을 확인해본 결과, *random_page_cost* 값이 작아짐에 따라 크기가 큰 테이블에 대한 인덱스 스캔이 수행되어 성능이 하락함을 알 수 있다. 이러한 경향은 하드디스크를 저장장치로 이용할 때 더욱 심각해지는데, 가장 성능 차이가 적은 Q7의 경우에도 빠른 실행 계획과 느린 실행 계획의 성능 차이는 약 6.2배로 나타나며, 가장 성능 차이가 많이 나는 Q3에서는 빠른 실행 계획과 느린 실행 계획의 성능 차이가 약 70배로 측정된다. 그에 비해 SSD를 저장장치로 이용할 때, Q7에서는 빠른 실행 계획과 느린 실행 계획의 성능 차이가 약 3배로 측정되며, 하드디스크에서 가장 심각한 성능 차이를 보였던 Q3의 경우에도 그 성능 차이는 약 6.4배로 측정된다. 이를 통해 SSD를 이용하는 경우 성능 하락이 완화되는 것을 확인할 수 있다.

Q6과 Q14의 경우 흥미로운 점이 추가로 발견된다. Q6과 Q14를 수행할 때, *random_page_cost*를 1로 설정하면 인덱스 스캔이 수행된다. 그리고 *random_page_cost*이 2, 4, 8일 때에는 비트맵 힙 스캔(Bitmap Heap Scan)이 수행되며, *random_ page_cost*이 16 이상일 경우 순차 스캔이 수행된다. 비트맵 힙 스캔은 임의 접근을 줄이기 위해 테이블 블록 번호 순서를 정렬하고 그 순서로 페이지에 접근하는 방법이다. 하드디스크를 이용할 때, 느린 임의 접근 속도로 인하여 인덱스 스캔을 이용할 때 가장 느린 성능이 나타나며, 비트맵 힙 스캔은 순차 스캔보다 조금 느린 성능을 보인다. 그에 비해, SSD를 이용할 때에는 SSD의 빠른 임의 접근 속도로 인하여 인덱스 스캔의 성능이 향상되어 비트맵 힙 스캔과의 성능 차이가 줄어드는 것을 확인할 수 있다.

4. 결 론

본 논문에서는 저장장치로 하드디스크를 이용할 때와 SSD를 이용할 때, *random_page_cost* 값을 1에서 32까지 변화시키며 TPC-H 벤치마크 수행하고 그 결과를 분석한다. 실험 결과, 실행 계획에 변화가 있는 쿼리의 경우, 하드디스크를 이용할 때와 SSD를 이용할 때 모두 순차 스캔을 이용하는 것이 인덱스 스캔을 이용하는 것에 비해 높은 성능을 보이는 것을 확인하였다. 또한, 저장장치로 SSD를 이용하는 것이 잘못된 비용 모델로부터 얻어진 느린 실행 계획으로 인한 성능 하락을 크게 완화시킴을 확인하였다.

사 사

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 SW컴퓨팅산업원천기술개발사업(SW스타랩)의 연구결과로 수행되었음 (IITP-2015-0-00314).

참 고 문 헌

[1] PostgreSQL, “http://www.postgresql.org/”
[2] Lee, Eun-Mi, Sang-Won Lee, and Sangwon Park. “Optimizing index scans on flash memory SSDs.“ ACM SIGMOD Record 40.4 (2012): 5-10.
[3] TPC-H, “http://www.tpc.org/tpch/”