대용량 데이터 처리를 위한 질의 컬럼셋과 수평 파티션의 통합 방법

정문영, 이태휘, 김성수, 송혜원, 원종호
전자통신연구원 빅데이터인텔리전스연구부
e-mail : {mchung, taewhi, sungsoo, hwonsong, jhwon}@etri.re.kr

Integrating Query Column-Sets and Horizontal Partitions on Very Large Data

Moonyoung Chung, Taewhi Lee, Sung-Soo Kim, Hyewon Song and Jongho Won
Electronics and Telecommunications Research Institute (ETRI)

요 약

분산된 데이터에 대한 질의 처리에서는 중간 데이터를 전송하는 단계에서 많은 디스크 I/O 및 네트워크 트래픽을 야기할 수 있다. 따라서, 질의에 필요하지 않은 데이터를 미리 필터링하면 불필요한 I/O 및 네트워크 전송을 줄일 수 있어 질의 처리 성능을 높일 수 있다. 이 논문에서는 질의 컬럼셋과 수평 파티션 방법을 통합하여 질의 처리에 불필요한 데이터를 초기 단계에 미리 필터링하여 질의 처리 성능을 높이는 방법을 제안한다.

1. 서론

SQL-on-Hadoop 은 하둡 분산 파일 시스템 (HDFS)에 저장된 대용량 데이터에 대해 빠른 SQL 질의 처리를 제공하는 시스템으로 하이브, 임팔라, Tajo 등이 있다. HDFS 기반의 시스템의 분산된 데이터의 느린 처리 속도를 향상시키기 위해서 실체화 뷰 (Materialized view), 질의 컬럼셋 (Query Column Sets) [1], 파티션 등의 기술들이 질의 처리를 가속화 하기 위한 방법들로 제안되고 있다.

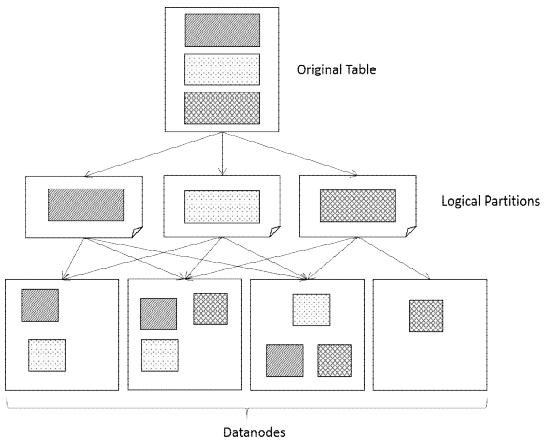
질의 컬럼셋이란 질의 워크로드를 분석하여 질의의 WHERE, GROUP BY, HAVING 절 등에 자주 사용되는 컬럼들을 물리적으로 실체화 한 것으로, 추후 질의가 들어왔을 때 질의 컬럼셋을 이용하여 질의 처리 속도를 높일 수 있다[2].

한편, SQL-on-Hadoop 에서 컬럼 파티션이나 해쉬 파티션과 같은 수평 파티션도 데이터를 미리 필터링하여 질의 처리 속도를 향상시키기 위한 방법으로 널리 사용되고 있다. 질의 컬럼셋은 테이블을 수직으로 나누어 실체화 한 것이라면, 파티션은 원래의 테이블을 파티션 키를 이용하여 수평으로 나누어 물리적으로 저장하는 기술이므로 파티션에 컬럼셋을 적용하면 데이터를 처리할 때, 원래의 테이블의 전체 데이터를 처리하지 않고 질의에서 필요로 하는 데이터에 해당하는 파티션만 이용하므로 빠른 성능을 기대할 수 있다. 특히, 분산된 대용량 데이터를 처리할 때는 노드 사이에 분산되어 있는 데이터를 이동하는 셔플 연산에서 성능 저하가 일어나므로 데이터 처리의 초기 단계에서 필요 없는 데이터를 필터링하여 처리속도를 크게 향상시킬 수 있다.

이 논문에서는 질의 컬럼셋과 수평 파티션을 효과적으로 통합하여 질의 처리 성능을 높이기 위한 방법을 제안한다.

2. 관련 연구

데이터베이스의 파티셔닝이란 테이블을 파티션이라 부르는 작은 파트로 물리적으로 나누는 것이다. 질의가 들어왔을 때 테이블 전체의 데이터를 처리하지 않고 질의에 필요한 데이터베이스 파티션만 이용하여 처리하면 되므로 질의 성능을 높일 수 있다. 특히, SQL-on-Hadoop 과 같이 대용량 데이터를 분산 저장하는 시스템에서는 질의를 처리하기 위해 대용량의 데이터를 노드 사이에서 전송하는 비용이 매우 높다.



(그림 1) 분산 환경에서 테이블의 파티션

그림 1 은 분산된 대용량 데이터에 대한 테이블을 파티션 하였을 때 데이터가 어떻게 저장되는지 보여준다. 사용자가 지정한 범위 등에 의해 원래 테이블이 논리적 파티션으로 나뉘어지고, 논리적 파티션은 여러 노드에 물리적으로 중복 분산되어 저장된다. 질의가 들어오면 질의 처리에 필요한 데이터가 들어 있는 파티션에 대해서만 스캔이나 조인과 같은 연산을 수행하면 된다. 따라서, 데이터를 처리하는 노드로 필요한 데이터 파티션만 이동시키면 되므로 불필요한 디스크 I/O 나 네트워크 전송을 줄일 수 있다.

3. 분산된 데이터에 대한 질의 컬럼셋과 수평 파티션의 통합 방법

질의 컬럼셋을 이용하여 질의를 가속화하는 기술은 첫째, 질의 워크로드를 분석하는 모듈과 둘째, 질의 컬럼셋을 생성하는 모듈로 구성된다 [2]. 질의 워크로드 분석에서는 질의 히스토리와 통계 정보를 이용하여 과거에 자주 같이 이용되었던 질의 컬럼셋들이 이후에도 자주 같이 이용될 것이라는 가정으로, 질의 컬럼셋을 추천한다. 질의 컬럼셋 생성 모듈은 질의 컬럼셋을 캐시 테이블로 구성하고, 질의가 들어오면 원래의 테이블이 아닌 캐시테이블을 이용하여 질의를 처리한다. 이러한 방법은 질의에 대한 데이터를 미리 필터링하여 질의 성능을 높일 수 있다.

분산된 데이터에 대한 질의 컬럼셋과 수평 파티션의 통합은, 물리적으로 수평 파티션된 데이터에 대한 캐시 테이블을 두어 대용량 데이터에 대한 질의 처리 성능을 높이는데 목적이 있다.

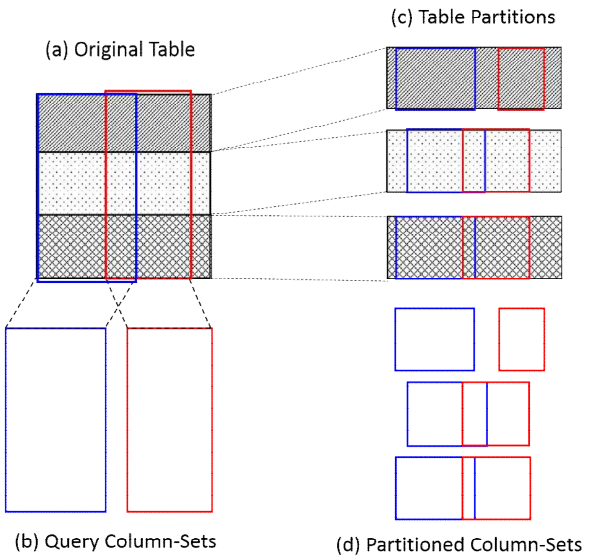
이를 위해서, 파티션 테이블에 대해서 컬럼셋을 캐시 테이블로 구성하는데 이를 파티션 컬럼셋이라고 하자.

그림 2(b)는 질의의 조건절 (WHERE, HAVING, GROUP BY) 등에 자주 사용되는 컬럼셋에 대한 캐쉬 테이블이다. 질이 컬럼셋을 이용하면 질의 처리를 데이터 양을 줄어들어 처리 속도의 높일 수 있다. 수평 파티션(c)은 범위, 컬럼, 해쉬값등으로 테이블을 물리적으로 파티션한 것이다. 파티션 컬럼셋(d)은 질의 컬럼셋과 수평파티션을 통합하여 파티션에 대한 질의 컬럼셋을 캐시 테이블로 실체화한 것이다. 파티션 컬럼셋을 이용하면 질의에 대한 결과셋을 초기 단계에서 더 많이 필터링하여 질의 성능을 높일 수 있다.

파티션 컬럼셋을 구성하는 방법은 질의 워크로드를 분석하는 모듈과 파티션 컬럼셋을 구성하는 두 가지 단계로 구성된다.

질의 워크로드 분석 모듈은 질의 컬럼셋을 구성할 때에 자주 같이 사용되는 컬럼들을 질의 컬럼셋으로 추천하는 것과 차이가 있다. 파티션 컬럼셋을 위한 질의 워크로드 분석 모듈은 먼저, 조건절을 분석하여 파티션키와 수평 파티션을 먼저 추천하거나, 혹은 사용자가 생성한 수평 파티션이 있다고 가정한다. 다음으로, 각 파티션에 대해서 자주 같이 사용되는 컬럼셋을 분석하여 파티션 컬럼셋을 구성한다. 따라서, 그림 2(d)와 같이 파티션에 따라 다른 질의 컬럼셋을 구성할 수 있다.

파티션 컬럼셋 구성 모듈은 질의 워크로드 분석 모듈에서 추천한 대로 파티션 컬럼셋을 캐시 테이블로 구성한다. 질의가 들어오면 질의에서 필요한 데이터에 해당되는 파티션 컬럼셋에 맞게 질의를 재구성하여 처리한다. 따라서 질의에서 필요로 하지 않는 파티션 컬럼셋은 초기 단계에서 필터링 되어, 불필요한 디스크 I/O 나 네트워크 전송을 줄일 수 있다. 특히 셔플이 여러 번 반복되는 질의에서 성능 향상의 효과를 높일 수 있다.



(그림 2) 파티션 컬럼셋

4. 결론

본 논문에서는 SQL-on-Hadoop 시스템에서 분산된 데이터에 대한 질의 처리 성능을 높이기 위해서, 수직으로 파티션하는 질의 컬럼셋과 데이터를 수평으로 파티션하는 기술을 통합하는 파티션 컬럼셋을 제안하였다. 과거의 질의 워크로드를 분석하여 파티션 컬럼셋을 구성하고, 후에 들어오는 질의를 분석하여 파티션 컬럼셋에 맞게 질의를 재구성하여 처리하여, 처리할 데이터의 양을 줄여 질의 성능을 높였다.

Acknowledgment

이 논문은 ETRI R&D 프로그램("듀얼모드 배치-쿼리 분석을 제공하는 빅데이터 플랫폼 핵심 기술 개발, 16ZS1400")의 일환으로 수행됨.

참고문헌

[1] Agarwal, Sameer, et al. "BlinkDB: queries with bounded errors and bounded response times on very large data." Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013.

[2] Sung-Soo Kim, Taewhi Lee, Moonyoung Chung, and Jongho Won. Sweet KIWI: Statistics-Driven OLAP Acceleration using Query Column Sets. In Proceedings of the 19th International Conference on Extending Database Technology (EDBT 2016), pp. 680-681, 2016.3.