맵리듀스를 이용한 다중 조인의 효율적인 처리 기법

최연정, 박진경, 이기용
숙명여자대학교 컴퓨터과학부
e-mail : {cyj, jinkyung93, kiyonglee}@sookmyung.ac.kr

Efficient Processing of Multi-Way Joins using MapReduce

Yeunjung Choi, Jinkyung Park, Ki Yong Lee
Division of Computer Science, Sookmyung Women's University

요 약

맵리듀스(MapReduce)는 대용량 데이터의 병렬 처리에 사용되는 프로그래밍 모델이다. 조인(join)은 둘 이상의 테이블에서 동일한 애트리뷰트 값을 가지는 레코드들을 결합하는 연산으로, 데이터베이스 분야에서 가장 중요한 연산 중 하나이다. 본 논문은 맵리듀스를 이용하여 다중 조인(multi-way)을 효율적으로 처리하는 방법을 제안한다. n 개 테이블의 다중 조인을 처리하기 위해 기존 방법은 2-way 조인을 수행하는 맵리듀스 잡을 $(n-1)$번 수행하거나, 레코드들을 중복시켜 n 개 테이블의 조인을 1 개의 맵리듀스 잡으로 한 번에 처리한다. 하지만 전자는 맵리듀스 잡을 $(n-1)$번 수행해야 하며, 후자는 레코드들을 상당히 많이 중복시켜야 한다는 단점이 있다. 본 논문은 레코드를 전혀 중복시키지 않고도 $\lceil \log_2 n \rceil$개의 맵리듀스 잡만으로 다중 조인을 효율적으로 처리하는 방법을 제안한다. 실험을 통해 제안 방법은 기존 방법에 대해 다중 조인을 더 빠르게 처리함을 보인다.

1. 서론

최근 웹 검색이나 SNS(Social Networking Service) 등 다양한 분야에서 데이터 양이 크게 급증하는 한편, 증가 속도는 점점 커지고 데이터 형태 또한 다양해지면서 이러한 데이터를 표현하기 위해 소위 빅데이터(Big Data)라는 개념이 등장하였다[1].

빅데이터를 쉽고 빠르게 분석하고 처리하기 위해 맵리듀스(MapReduce)라는 프로그래밍 모델이 제안되었다[2]. 맵리듀스는 다수의 컴퓨터에 분산되어 저장된 데이터를 병렬적으로 처리하는 기술로서, 사용자가 수행하고자 하는 작업을 맵(Map)과 리듀스(Reduce)라는 두 개의 함수로 표현하기만 하면 이를 다수의 컴퓨터가 자동으로 병렬적으로 수행한다. 사용자가 한 쌍의 맵과 리듀스로 표현한 작업을 맵리듀스 잡(job)이라 한다. 한편 조인(join)이란 둘 이상의 테이블에서 지정된 애트리뷰트의 값이 동일한 레코드들을 결합하는 연산으로, 데이터베이스 분야에서 가장 중요한 연산 중 하나이다.

본 논문에서는 맵리듀스를 사용하여 세 개 이상의 테이블 들을 조인하는 다중 조인(multi-way join)을 효율적으로 처리하는 방법을 제안한다. 맵리듀스로 두 개의 테이블을 조인하는 방법에 대해서는 이미 여러 기법들이 제안되었으나, 아직 여러 개의 테이블들을 조인하는 다중 조인에 대해서는 충분한 연구가 이루어지지 않았다. 맵리듀스로 n 개의 테이블을 조인하는 기존 방법에는 크게 두 종류가 있다. (1) 순차 2-way 조인: n 개의 테이블을 조인하기 위해 2 개의 테이블을 조인하는 맵리듀스 잡을 $(n-1)$번 순차적으로 수행한다. (2) 단일 n-way 조인[3]: 테이블의 레코드들을 중복시켜 n 개 테이블의 조인을 1 개의 맵리듀스 잡으로 한 번에 처리한다. 하지만 (1)은 $(n-1)$개의 맵리듀스 잡을 순차적으로 수행해야 하며, (2)는 테이블의 레코드들을 상당히 많이 중복시켜야 한다는 단점이 있다.

본 논문에서 제안하는 방법은 레코드들을 전혀 중복시키지 않고도 단 $\lceil \log_2 n \rceil$개의 맵리듀스 잡만으로 n 개의 모든 테이블들을 조인한다. 제안 방법은 레코드들을 전혀 중복시키지 않기 때문에 맵리듀스에서 중간결과를 전송하고 정렬하는데 드는 비용이 크게 감소된다. 또한 $(n-1)$개보다 더 적은 $\lceil \log_2 n \rceil$개의 맵리듀스 잡만을 사용하기 때문에 계산 속도가 크게 향상된다. 실험을 통해 제안 방법은 기존 방법에 대해 다중 조인을 더 빠르게 처리함을 보인다.

본 논문의 구성은 다음과 같다. 2 장에서는 맵리듀스와 조인에 대한 배경 지식을 소개하고, 3 장에서는 맵리듀스를 이용한 다중 조인에 대한 관련 연구를 설명한다. 4 장에서는 제안 방법을 설명한 후, 5 장에서 실험 결과를 제시한다. 마지막으로 6 장에서 결론을 맺는다.

2. 배경 지식

맵리듀스를 사용하기 위해 사용자는 자신이 수행하고자 하는 작업을 맵과 리듀스라는 두 단계의 함수로 나누어 표현한다. 첫 번째 단계의 맵 함수는 처리하고자 하는 데이터를 담고 있는 입력 파일로부터 (key, value) 형태의 입력을 받아, 각 (key, value)에 대해 결과 값으로 하나 이상의 (key, value)들을 내보낸다. 맵

리듀스는 맵 함수가 결과로 내보낸 (key, value)들을 정렬하여 같은 key 값을 가지는 (key, value)들을 모아 각 key 값에 대해 (key, [value1, value2, …]) 형태의 입력을 만든 뒤, 이를 리듀스 함수로 전달한다. 두 번째 단계의 리듀스 함수는 (key, [value1, value2, …]) 형태의 입력을 받아, 각 (key, [value1, value2, …])에 대해 결과 값으로 하나 이상의 (key, value)들을 내보낸다. 리듀스 함수가 출력한 (key, value)들이 맵리듀스 잡의 최종 수행 결과가 된다.

한편 조인이란 둘 이상의 테이블에서 지정된 애트리뷰트(조인 애트리뷰트)의 값이 동일한 레코드들을 결합하는 연산으로, 맵리듀스를 사용하여 2-way 조인을 처리하는 방법에 대해서는 이미 많은 연구가 이루어졌다. 맵리듀스를 사용하여 두 개의 테이블 A, B 간의 2-way 조인 $A \bowtie B$ 를 처리하는 일반적인 방법은 다음과 같다[4]. 맵 함수는 A 와 B 의 각 튜플 t 을 입력으로 받아, t 의 조인 애트리뷰트 값을 key 로 하고, 어떤 테이블의 튜플인지를 나타내는 태그가 부착된 t 를 value 로 하는 (key, value)를 결과로 내보낸다. 맵리듀스는 같은 조인 애트리뷰트 값, 즉 같은 key 값을 가지는 (key, value)들을 모아 이를 (key, [value1, value2, …]) 형태로 리듀스 함수로 전달한다. 리듀스 함수는 조인 애트리뷰트 값이 key 로 동일한 튜플들 value1, value2, …의 테이블 태그를 보고 서로 다른 테이블에서 온 튜플들을 결합하여 조인 결과를 내보낸다.

3. 관련 연구

맵리듀스로 n 개의 테이블을 조인하는 다중 조인에 대한 기존 방법은 크게 두 종류가 있다.

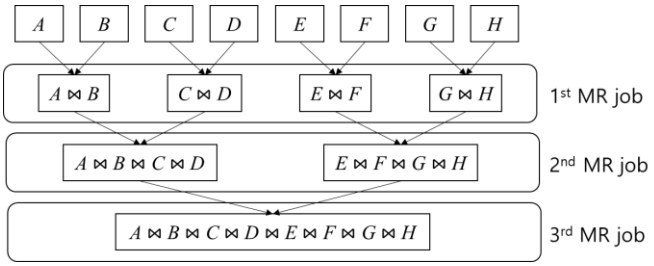
(1) 순차 2-way 조인: n 개의 테이블 중 2 개의 테이블을 먼저 조인하고, 그 결과를 다른 테이블과 조인한다. 이러한 방법을 모든 테이블들이 조인될 때까지 반복한다. 각 조인을 수행할 때는 앞서 기술한 2-way 조인 방법을 사용한다. 따라서 총 $(n-1)$개의 맵리듀스 잡이 순차적으로 수행된다.

(2) 단일 n-way 조인: 3 개의 테이블 $A(a, b)$, $B(b, c)$, $C(c, d)$ 간의 다중 조인 $A \bowtie B \bowtie C$ 를 계산한다고 하고 b 와 c 를 조인 애트리뷰트라고 하자. 맵 함수는 A, B, C 의 각 튜플 t 에 대해 t 의 조인 애트리뷰트 값, 즉, b 와 c 애트리뷰트의 값 (v_b, v_c)를 key 로 하고, 어떤 테이블의 튜플인지를 나타내는 태그가 부착된 t 를 value 로 하는 (key, value)를 결과로 내보낸다. 이 때, t 가 A 의 튜플인 경우 t 는 c 애트리뷰트가 없으므로 c 애트리뷰트가 가질 수 있는 모든 가능한 값 v_c 에 대해 (v_b, v_c) 형태의 key 를 만들어 각각에 대해 (key, value)를 내보낸다. t 가 C 의 튜플인 경우에도 앞과 유사하게 t 는 b 애트리뷰트가 없으므로 b 애트리뷰트가 가질 수 있는 모든 가능한 값 v_b 에 대해 (v_b, v_c) 형태의 key 를 만들어 각각에 대해 (key, value)를 내보낸다. 리듀스 함수는 같은 조인 애트리뷰트 값 (v_b, v_c)를 가지는 튜플들로부터 테이블 태그를 보고 서로 다른 테이블에서 온 레코드를 결합하여 조인 결과를 내보낸다. 따라서 1 개의 맵리듀스 잡만으로 n 개의 테이블이 한 번에 조인된다.

하지만 위 두 방법은 각각 문제점을 가지고 있다. (1)은 $(n-1)$개의 맵리듀스 잡을 순차적으로 수행해야 하므로 병렬성이 떨어져 수행 시간이 길어진다. 한편 (2)는 맵 함수가 A 와 C 테이블의 레코드들에 대해서 각각 c 와 b 애트리뷰트가 가질 수 있는 값들의 수만큼 (key, value)를 중복하여 내보낸다. 따라서 맵 함수가 결과로 내보내는 (key, value)의 개수가 많아져 맵리듀스가 그들을 정렬하고 전송하는데 드는 비용이 커진다.

4. 제안 방법

본 장에서는 제안 방법을 설명한다. n 개 테이블의 다중 조인을 계산하기 위해, 제안 방법은 첫 번째 맵리듀스 잡에서 n 개의 테이블을 두 개씩 묶어 총 $n/2$ 개의 쌍으로 분할하고, 각 쌍에 대해 하나씩 총 $n/2$ 개의 2-way 조인을 하나의 맵리듀스 잡으로 한 번에 계산한다. 두 번째 맵리듀스 잡에서는 $n/2$ 개의 조인 결과를 두 개씩 묶어 총 $n/4$ 개의 쌍으로 분할하고, 각 쌍에 대해 하나씩 총 $n/4$ 개의 2-way 조인을 하나의 맵리듀스 잡으로 한 번에 계산한다. 이러한 방식으로 진행하여 마지막 맵리듀스 잡은 2 개의 조인 결과에 대해 2-way 조인을 수행하여 최종 조인 결과를 얻는다.



(그림 1) 제안 방법

(그림 1)은 제안 방법을 나타내는 그림이다. 8 개의 테이블 $A(a, b)$, $B(b, c)$, $C(c, d)$, $D(d, e)$, $E(e, f)$, $F(f, g)$, $G(g, h)$, $H(h, i)$간의 다중 조인 $A \bowtie B \bowtie C \bowtie D \bowtie E \bowtie F \bowtie G \bowtie H$ 를 계산하는 예로 제안 방법을 설명한다. 여기서는 b, c, d, e, f, g, h 가 조인 애트리뷰트가 된다. 첫 번째 맵리듀스 잡은 $A \bowtie B$, $C \bowtie D$, $E \bowtie F$, $G \bowtie H$ 의 4 개의 2-way 조인을 한 번에 계산한다. 맵 함수는 A, B, C, E, F, G, H 의 각 튜플 t에 대해 t의 조인 애트리뷰트 값과 그 값이 어떤 조인 애트리뷰트의 값인지를 나타내는 태그를 연결한 값을 key 로 하고, 어떤 테이블의 튜플인지를 나타내는 태그가 부착된 t 를 value 로 하는 (key, value)를 결과로 내보낸다. 리듀스 함수는 동일한 조인 애트리뷰트에 대해 동일한 값을 가지는 튜플들을 받아 서로 다른 테이블에서 온 레코드를 결합하여 해당 2-way 조인에 대한 결과를 내보낸다. 예를

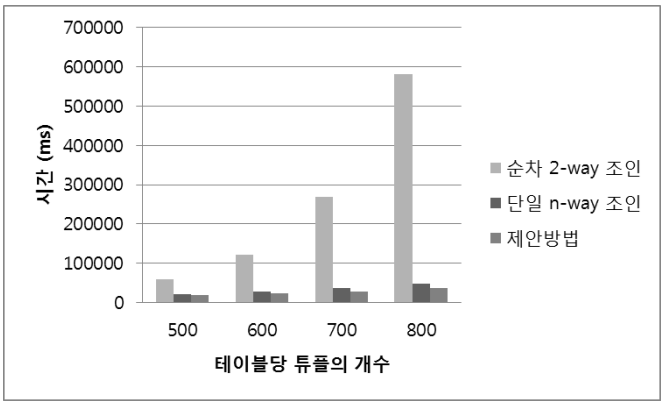
들어 리듀스 함수가 (2:*b*, [*t*1, *t*2, …])와 같은 입력을 받은 경우, *t*1, *t*2, …를 조인한 결과는 조인 애트리뷰트 *b*의 값이 2 인 튜플들을 조인한 결과이므로 $A \bowtie B$ 의 부분 결과를 나타내게 된다. 이와 유사하게 두 번째 맵리듀스 잡은 $A \bowtie B$ 와 $C \bowtie D$ 의 결과로부터 $A \bowtie B \bowtie C \bowtie D$ 를, $E \bowtie F$ 와 $G \bowtie H$ 의 결과로부터 $E \bowtie F \bowtie G \bowtie H$ 를 구하는 두 개의 2-way 조인을 한 번에 계산한다. 마지막 세 번째 맵리듀스 잡은 $A \bowtie B \bowtie C \bowtie D$ 와 $E \bowtie F$ 와 $G \bowtie H$ 의 결과를 2-way 조인하여 최종 결과를 얻는다.

따라서 제안 방법은 *A, B, C, E, F, G, H* 의 레코드들을 전혀 중복하지 않고도, 단 $\lceil \log_2 n \rceil$개의 맵리듀스 잡만으로 이들에 대한 다중 조인을 계산할 수 있다. 제안 방법은 레코드들을 전혀 중복시키지 않으므로 맵 함수가 출력한 (key, value)를 정렬하고 전송하는데 드는 비용이 크게 감소된다. 또한 $(n-1)$개보다 더 적은 $\lceil \log_2 n \rceil$개의 맵리듀스 잡만을 사용하기 때문에 병렬성이 향상되어 계산 속도가 빨라진다.

5. 실험 결과

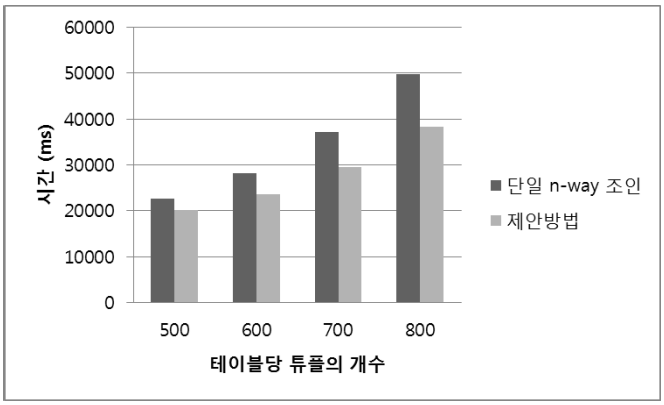
본 장에서는 제안 방법의 성능을 순차 2-way 조인 방법과 단일 *n*-way 조인 방법과 비교한 실험 결과를 보인다. 성능 척도로는 다중 조인을 계산하는데 드는 전체 시간을 기준으로 하였다. 실험은 Amazon EC2 서비스[5]에서 제공하는 4 대의 컴퓨터로 구성된 클러스터를 사용하였으며, 맵 태스크와 리듀스 태스크의 수는 각각 4 개와 6 개로 하였다.

실험 데이터로는 가상의 데이터를 생성하였다. 실험에서는 4 개의 테이블 *A*(*a*, *b*), *B*(*b*, *c*), *C*(*c*, *d*), *D*(*d*, *e*)를 생성하여, 이들 간의 다중 조인 $A \bowtie B \bowtie C \bowtie D$ 의 계산에 드는 시간을 측정하였다. 각 레코드의 모든 애트리뷰트의 값은 [0, 100] 사이의 자연수 중 임의로 선택하였으며, 각 테이블의 튜플 개수는 500 개에서 800 개까지 증가시켰다.



(그림 2) 실험 결과

(그림 2)는 순차 2-way 조인 방법과 단일 *n*-way 조인 방법, 그리고 본 논문에서 제안한 방법으로 $A \bowtie B \bowtie C \bowtie D$ 를 계산하는 데 걸린 총 시간을 나타낸다. 3 장에서 설명한 바와 같이, 순차 2-way 조인은 $(n-1)$ 개의 맵리듀스 잡을 순차적으로 수행해야 하므로 각 맵리듀스 잡을 시작하는 시간 증가 및 병렬성의 저하에 따라 성능이 크게 저하된다. 단일 *n*-way 조인은 맵리듀스 잡의 수는 1 개로 최소화하였지만, 맵 함수가 (key, value)를 중복해서 내보내므로 이들을 정렬하고, 리듀스 함수로 전송하는 비용이 증가하게 된다. 그에 비해 제안 방법은 첫 번째 맵리듀스 잡에서 $A \bowtie B$ 와 $C \bowtie D$ 를 동시에 계산하여 병렬성을 높이면서도 전혀 중복을 발생시키지 않는다. 두 번째 맵리듀스 잡에서는 $A \bowtie B$ 와 $C \bowtie D$ 를 다시 2-way 조인으로 조인하므로, 이미 조인 결과로 작아진 입력을 사용하면서도 여전히 중복은 발생시키지 않는다. 그 결과 제안 방법은 기존 방법에 비해 좋은 성능을 보인다.



(그림 3) 제안 방법과 단일 *n*-way 조인과의 비교

(그림 3)은 제안방법과 단일 *n*-way 조인 방법만을 비교한 것으로, 튜플의 개수가 많아질수록 제안 방법의 성능이 상대적으로 더 좋아지는 것을 볼 수 있다. 따라서 제안 방법이 튜플 수가 증가함에 따라 단일 *n*-way 조인에 비해 확장성을 가지고 있음을 알 수 있다.

6. 결론

본 논문은 맵리듀스 환경에서 다중 조인에 대한 효율적인 계산 방법을 제안하였다. 제안 방법은 기존의 단일 *n*-way 조인 방법에서 발생하는 중복을 제거하면서도, 중복을 발생시키지 않는 기존의 순차 2-way 조인 방법과 비교해서도 맵리듀스 잡의 수를 $(n-1)$개에서 $\lceil \log_2 n \rceil$개로 크게 줄였다. 그에 따라 병렬성이 향상되는 한편, 네트워크 전송 및 데이터 정렬에 드는 비용이 감소되어 다중 조인 계산에 드는 총 시간이 줄어들게 된다. 실험을 통해 제안 방법이 기존 방법에 비해 다중 조인을 더 빠르게 계산함을 확인하였다.

Acknowledgment

본 논문은 2014 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2012R1A1A1001269)

참고문헌

[1] Big Data, http://en.wikipedia.org/wiki/Big_data

[2] Jeffrey Dean, Sanjay Ghemawat, "MapReduce: simplified data processing on large clusters," In Proceedings of OSDI '04, pp. 137-150, 2004.
[3] Foto N. Afrati and Jeffrey D. Ullman, “Optimizing Multiway Joins in a Map-Reduce Environment”, IEEE Transactions on Knowledge & Data Engineering, vol. 23, no. 9, pp. 1282-1298, 2011.
[4] Spyros Blanas, Jignesh M. Patel, Vuk Ercegovac, Jun Rao, “A Comparison of Join Algorithms for Log Processing in MapReduce”, In Proceedings of ACM SIGMOD, 2010.
[5] Amazon Web Services, http://aws.amazon.com/ec2/