데이타베이스 공유 시스템에서 병렬 해쉬 조인 알고리즘의 구현

김창현[0] 조행래
영남대학교 컴퓨터공학과
chkim@com.ne.kr, hrcho@yu.ac.kr

Implementation of Parallel Hash Join Algorithms in a Database Sharing System

Changhyun Kim[0] Haengrae Cho
Dept. of Computer Engineering, Yeungnam University

요 약

기존에 제안된 대부분의 병렬 조인 알고리즘들은 데이타베이스가 여러 처리 노드에 분할되어 저장되는 데이타베이스 분할 시스템을 가정하였다. 데이타베이스 분할 시스템은 다수의 노드들을 연결할 수 있으며 지리적으로 분산된 환경도 지원할수 있다는 장점을 갖지만, 데이타베이스 공유 시스템에 비해 부하 분산이나 시스템 가용성이 떨어진다는 단점을 갖는다. 본 논문에서는 데이타베이스 공유 시스템에서 병렬 질의 처리기를 위한 병렬 해쉬 조인 알고리즘을 구현한다. 이를 위하여, 데이타베이스 공유 시스템에 적용 가능하도록 병렬 질의 처리기를 구성하고 병렬 해쉬 조인 알고리즘의 처리 과정에 대해 설명 한다.

1. 서 론

저렴한 처리 노드들을 연동하여 고성능 트랜잭션 처리 및 복잡한 질의에 대한 빠른 응답 시간을 지원하기 위한 병렬 데이타베이스 시스템의 구조로 데이타베이스 분할 시스템(Database Partition System : DPS)과 데이타베이스 공유 시스템(Database Sharing System : DSS)이 제안되었다[1,2,3]. DPS는 전체 데이타베이스를 분할하여 각 노드가 데이타베이스의 일정 분할을 관리하는 구조이며, DSS는 고속의 네트워크로 연결된 각각의 처리 노드가 디스크 레벨에서 하나의 데이타베이스를 공유한다. DPS는 다수의 노드들을 연결할 수 있으며 데이타베이스가 지리적으로 분산된 환경도 지원할 수 있다는 장점을 갖지만, 하나의 노드가 고장날 경우 그 노드가 관리하고 있는 데이타베이스 분할은 사용할 수 없다는 단점을 갖는다. 뿐만 아니라, 노드들간의 부하 분산이 어렵고 확장성도 떨어진다.

본 논문에서는 본 연구의 선행연구로 DSS를 위해 제안된 병렬 조인 알고리즘들을 구현한다. 현재까지 제안된 대부분의 병렬 조인 알고리즘들은 DPS를 가정하였으며, DSS를 위한 병렬 조인 알고리즘들은 거의 제안되지 않았다[4,5]. 물론 DPS에서 제안된 기본 개념들이 DSS에서도 적용될 수 있지만, DSS에서는 각 노드들이 전체 데이타베이스를 액세스할 수 있으므로 훨씬 다양하고 효율적인 병렬 조인 알고리즘의 개발이 가능하고 동적 부하분산이 용이하다.

2. 병렬 해쉬 조인 알고리즘

본 절에서는 기존의 해쉬 조인 알고리즘을 병렬화한 병렬 해쉬 조인 알고리즘에 대하여 설명한다[6]. Parallel Grace Hash Join(PGHJ)는 Grace Hash Join(GHJ)을 병렬화한 것이다.

PGHJ의 분할 단계는 각 노드가 자신에게 할당된 릴레이션 부분들에 대해 해쉬 함수를 적용하여 버킷을 생성하고, 버킷 오버플로우가 발생하면 디스크에 기록한다. 분할 단계가 완료되면 메모리에 남아있는 버킷 내용들을 디스크에 기록한 후, 작업 관리자에게 생성된 모든 버킷에 대한 버킷 정보[버킷식별자(B_I), 버킷 크기(B_S), 처리되지 않고 남은 버킷 크기(B_U), 페이지 식별자 리스트(P_B)]를 전송한다. 작업 관리자는 플래그먼트 단위로 버킷을 분할하여 각 노드에서 조인을 실행한다.

작업 관리자의 처리 과정은 다음과 같다.

① 각 노드로부터 버킷 정보[B_I, B_S, B_U, P_B]를 전송 받아 작업 테이블을 만든다.

② B_U를 메모리 크기에 따라 프래그먼트 단위로 분할한다. 이후 작업 테이블에 프래그먼트 정보[F_I, F_S, F_A, P_F]를 기록한다.

③ 노드로부터 작업 요청 메시지가 전송되면, 할당되지 않은 프래그먼트를 작업 요청 노드에게 할당하고 [B_I, F_I, F_S, P_B, P_F]정보를 포함한 메시지를 전송한다. 즉, 조인을 위한 내부 릴레이션(R)의 프래그먼트 정보 및 외부 릴레이션(S)의 버킷 정보를 전송한다.

분할 단계에서의 처리 과정은 다음과 같다.

① 조인 애트리뷰트 JA_R을 해싱하여 R을 |B|개의 버킷으로 분할(해쉬 함수 = h_1) 한후 디스크에 저장한다.

② 조인 애트리뷰트 JA_S를 동일한 해쉬 함수 h_1을 이용하여 |B|개의 버킷으로 분할한후 디스크에 저장한다.

③ R과 S의 버킷들에 대해 [B_I, B_S, B_U, P_B] 메시지를 작업 관리자에게 전송한다.

조인 단계에서의 처리과정은 다음과 같다.

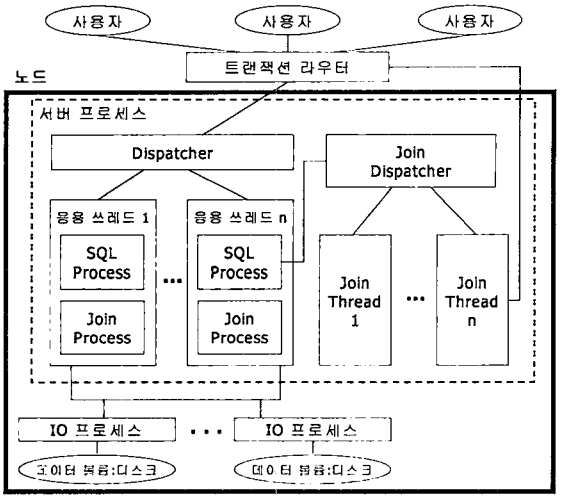
① 다음에 처리할 프래그먼트 정보를 작업 관리자에게 요청
② while (작업 관리자가 [B_I, F_I, F_S, P_B, P_F] 메시지 전송)
③ for each (P_F에 속하는 R의 페이지 p)
④ p를 디스크에서 판독한후 p에 저장된 튜플 r을 해쉬 함수 h_2로 해싱하여 해쉬 테이블에 등록
⑤ endfor
⑥ for each (P_B에 속하는 S의 페이지 p)
⑦ p를 디스크에서 판독한후 p에 저장된 각각의 튜플 s를 h_2로 해싱하여 해쉬 테이블 검사
⑧ 동일한 해쉬 값을 갖는 r과 s에 대해 $r.JA_R = s.JA_S$일 경우 (r, s) 출력
⑨ endfor
⑩ 다음에 처리할 프래그먼트 정보를 작업 관리자에게 요청
⑪ endwhile

3. 병렬 해쉬 조인 알고리즘 구현

본 절에서는 병렬 해쉬 조인 알고리즘들을 데이타베이스 공유 시스템 환경에서 실행할수 있도록 시스템을 설계하고 구현한다. 먼저 전체 시스템 구조에 대해 설명하고, 병렬 해쉬 조인 처리 과정에 대해 상세히 설명한다.

3.1 병렬 해쉬 조인 처리를 위한 시스템 구조

사용자로부터 질의가 입력되면 질의를 파싱한후, 최적화를 거쳐 관계 대수 식을 실행하게 된다. 조인 연산을 병렬 처리하기 위해 PGHJ 알고리즘을 사용한다. <그림 1>은 병렬 해쉬 조인 처리를 위한 단일 시스템 구조이다.

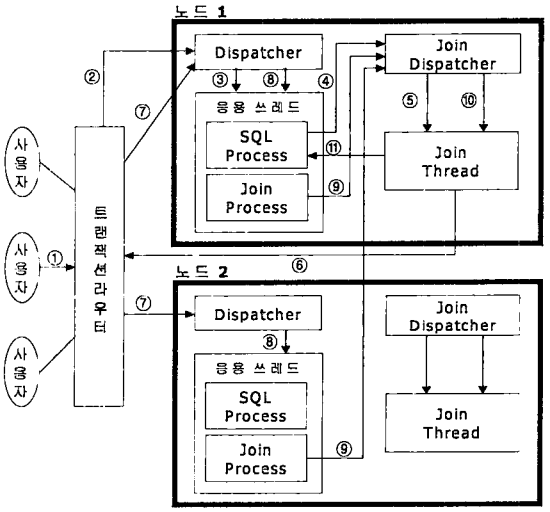


<그림 1> 단일 노드에서의 병렬 해쉬 조인 구조

트랜잭션 라우터는 UDP 소켓 인터페이스를 이용하여 사용자로부터 질의를 입력받아 적절한 노드로 질의를 전송한다. 서버는 요청된 서비스를 실행하기 위해 하나의 Dispatcher 쓰레드와 여러개의 응용 프로그램 쓰레드, 그리고 데이타 볼륨 수만큼의 IO 프로세스로 구성되어 있다. 이때, Dispatcher 쓰레드는 요청된 서비스와 응용 프로그램을 실행하기 위한 쓰레드간의 바인딩을 지원하는 역할을 담당한다. 응용 프로그램 쓰레드는 호출된 응용 프로그램을 실행하는 역할을 담당한다. 병렬 조인처리를 위해 하나의 Join Dispatcher 쓰레드와 여러개의 Join 쓰레드, 그리고 응용프로그램 쓰레드에 있는 SQL Process와 Join Process가 사용된다. Join Dispatcher 쓰레드는 요청된 조인과 조인 처리를 위한 조인 쓰레드간의 바인딩을 지원하는 역할을 한다. Join 쓰레드는 조인처리를 실행하는 역할을 담당한다.

3.2 병렬 해쉬 조인 처리 과정

단일 노드 시스템을 여러 노드로 확장한 그림이 <그림 2>에 나타난다. 전체 구조를 간략화 해서 전체 동작 과정을 알기 쉽게 나타내었다. 각 단계별로 필요한 자료구조와 동작 과정에 대해서 자세히 설명한다.



<그림 2> 병렬 해쉬 조인의 처리 과정

단계 ① : 사용자로부터 입력받은 질의를 트랜잭션 라우터에게 전송한다.

단계 ② : 트랜잭션 라우터는 적절한 노드를 선택해서 해당 노드의 Dispatcher 쓰레드에게 전송한다.

단계 ③ : Dispatcher 쓰레드는 응용 프로그램 쓰레드 중 질의 처리에 관한 SQL Process를 생성한다.

단계 ④ : 질의 처리중 조인연산이 필요하면 버킷 정보 파일을 생성한후 Join Dispatcher에게 전송한다. 조인 연산을 실행하기 위해 먼저 각 릴레이션을 동일한 해싱 함수를 사용하여 버킷 단위로 분할한다. 예를 들어 릴레이션이 R과 S가 존재한다고 가정할 때 R과 S를 동일한 해싱함수를 사용하여 버킷 단위로 분할한뒤 버킷 정보를 버킷 정보 파일에 저장한다. Join Dispatcher에게 전송되는 메시지에는 조인 요청을 뜻하는 JN_START 태그, 현재 쓰레드의 쓰레드 식별자, 버킷 정보 파일이 생성되어 있는 데이타베이스 번호와 파일명, 결과 저장 파일이 생성되어 있는 데이타베이스 번호와 파일명이 포함된다.

단계 ⑤ : Join Dispatcher는 조인 요청을 수신하는 역

할을 하며, UDP 포트 하나를 열고 대기하고 있다가 이 포트로 들어오는 메시지를 수신한다. 메시지의 종류에 따라 두가지 기능을 한다. JN_START이면 조인 요청이므로 새로운 Join 쓰레드를 생성하고 수신한 메시지를 그대로 전송한다. 생성한 Join 쓰레드의 쓰레드 식별자는 다시 메시지로 만들어 조인을 요청한 쓰레드에게 전송한다. 즉, 조인을 요청한 쓰레드가 결과를 수신할 수 있도록 하기 위해 Join 쓰레드의 쓰레드 식별자를 넘겨준다. JN_RESULT이면 조인 결과이므로 해당 Join 쓰레드에게 메시지를 넘겨준다. 여기에 해당하는 단계는 단계 ⑩이다.

단계 ⑥ : Join 쓰레드는 버킷 정보를 분할해서 트랜잭션 라우터에게 전달한다. 전달되는 메시지 구조에는 조인 처리를 위한 JN_PROCESS 태그, 결과를 전송할 때 해당 Join 쓰레드를 찾기 위한 Join 쓰레드의 쓰레드 식별자, 노드 번호, 조인을 할 버킷 번호가 포함된다.

단계 ⑦ : 트랜잭션 라우터는 노드 번호와 버킷 번호에 따라 해당 노드의 Dispatcher 쓰레드에게 전송한다.

단계 ⑧ : 각 노드의 Dispatcher 쓰레드는 전송된 메시지가 조인 처리를 뜻하는 JN_PROCESS이면 응용 프로그램 쓰레드중 Join Process를 생성해서 수신된 메시지를 전송한다.

단계 ⑨ : Join Process는 실제 조인을 처리하는 부분으로 버킷 정보 파일에서 버킷 정보, 조인에 참여하는 조인 리스트등을 읽어서 해당 버킷의 조인을 처리한다. 결과는 결과 저장 파일에 저장해서 Join Dispatcher 쓰레드에게 전송한다. 전송되는 메시지에는 결과를 뜻하는 JN_RESULT 태그와 Join 쓰레드의 쓰레드 식별자가 포함된다.

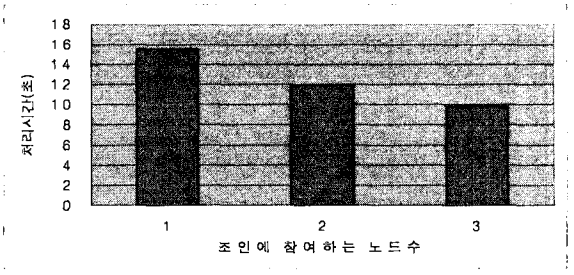
단계 ⑩ : Join Dispacher 쓰레드는 수신된 메시지가 JN_RESULT이면 모든 조인처리가 완료 되었음을 나타낸다. 수신된 메시지에는 Join 쓰레드의 쓰레드 식별자가 있으므로 해당 Join 쓰레드에게 메시지를 넘겨 준다.

단계 ⑪ : Join 쓰레드는 결과가 완료되었다는 메시지를 각 노드로부터 모두 받으면 조인을 요청한 쓰레드에게 처리가 완료되었다는 메시지를 전송한다. 결과 메시지를 전송받은 조인 요청 쓰레드는 결과 저장 파일을 읽어서 리스트 형태로 만들어서 조인 결과를 반환한다.

3.3 병렬 해쉬 조인 알고리즘의 성능

본 절에서는 조인에 참여하는 노드수에 따른 조인 처리시간에 대해 성능평가를 한다. 노드수를 1개에서 3개로 증가하면서 시간을 측정한 그래프가 <그림 3>에 나타난다.

실험을 수행한 시스템은 SunOS 4.1.4를 탑재한 Sun Sparc Server 시스템으로, SuperSPARC 80MHz 2 CPUs 1대(시스템1), SuperSPARC 80MHz 1 CPU 2대(시스템2, 시스템3)이다. 노드수가 1개 일때는 시스템1을 사용했고, 노드수가 2개 일때는 시스템1과 시스템2를, 노드수가 3개 일때는 시스템1, 시스템2, 시스템3을 사용하였다. 각 노드수에 따른 버킷 할당량은 노드수가 1개 일때는 버킷 전부를, 노드수가 2개 일때는 버킷 할당비율을 3:1로, 노드수가 3개 일때는 2:1:1로 하였다. 테이블수는 2개이고, 각 테이블의 레코드수는 4000개로 하였다.



<그림 3> 노드수에 따른 조인 처리 시간

실험결과를 보면 조인에 참여하는 노드수가 증가함에 따라 조인 처리 시간이 현저히 감소하는 것을 알 수 있으며, 노드수가 1개인 경우에 비해 노드수가 3개인 경우 약 60%의 성능 향상을 보였다.

4. 결론 및 향후 연구 방향

대부분 기존 병렬 조인 알고리즘들은 DPS를 가정하여 제안되었다. 그러나 DPS에 비해 DSS는 모든 노드들이 데이타를 공유하므로 병렬 조인 처리에 훨씬 적합하다는 장점을 갖는다. 실제 조인 연산에 있어서도 단일 노드에서 실행하는 것 보다 여러 노드에 나누어서 병렬적으로 처리함으로써 성능향상을 가져왔다.

현재까지 개발한 시스템은 버킷을 분할 할때 한 노드에서만 분할 하였는데, 이것을 여러노드에 나누어서 처리하면 성능향상이 기대된다. 뿐만 아니라, 버킷 분할 개수도 고정되어 있는데, 조인에 참여하는 노드수에 따라 동적으로 할당해주면 성능이 향상될 것으로 기대된다.

마지막으로, PHHJ[6]에 대해서도 구현할 예정인데, PHHJ는 PGHJ보다 성능이 우수 할것으로 기대된다.

[참고 문헌]

[1] M. Abdelguerfi and K. Wong, *Parallel Database Techniques*, IEEE Comp. Society Press, 1998.

[2] D. DeWitt and J. Gray, "Parallel Database Systems : The Future of High Performance Database System," *Comm. ACM*, Vol.35, No.6, pp.85-98, 1992.

[3] E. Rahm, "Parallel Query Processing in Shared Disk Database Systems," *Proc. 5th Int. Conf. High Performance Transactions Syst.*, 1993.

[4] H. Lu, B. Ooi and K. Tan, *Query Processing in Parallel Database Systems*, IEEE Comp. Society Press, 1994.

[5] D. Schneider and D. DeWitt, "A Performance Evaluation of Four Parallel Join Algorithms in a Shared Nothing Multiprocessor Environment," *Proc. ACM SIGMOD*, pp.110-121, 1989.

[6] 문애경, 조행래 "데이타베이스 공유 시스템에서 동적 부하분산을 지원하는 해쉬 조인 알고리즘들의 성능 평가," 한국정보처리학회 논문지 제6권 제12호, 1999.