분산 주기억장치 데이타베이스에서 컴파일 시 카탈로그 관리 기법의 성능평가*

정한라+ 홍의경
서울시립대학교 컴퓨터· 통계학과
{hrjung93, ekhong}@venus.uos.ac.kr

Performance Evaluation of Catalog Management Schemes for Distributed Main Memory Databases : Compilation Cases

Han-ra Jeong+ Eui-kyeong Hong
Dept. of Computer Science and Statistics, University of Seoul

요 약

분산 DBMS에 대한 연구는 디스크에 데이타가 저장되어 있는 환경을 가정한 관계 DBMS에서 주로 진행되어 왔다. 디스크에 데이타가 저장되어 있다고 가정하는 시스템은 질의 최적화, 버퍼 관리기, 인덱스 관리 기법 등 여러 가지 측면에서 주기억장치 DBMS와 크게 다르기 때문에 이런 분산 DBMS에서 연구된 결과들을 그대로 주기억장치 상주 DBMS의 분산 시스템에 적용하기에는 어려움이 있다. 본 논문에서는 이러한 주기억장치 상주 중앙 집중형 DBMS를 분산 시스템으로 확장할 때 고려해야 할 여러 문제 중 캐시의 유무에 따른 카탈로그의 구조에 대해 살펴보고 시뮬레이션을 통해 카탈로그 관리기법에 대한 성능을 평가한다. 카탈로그 관리 기법의 성능평가 대상으로는 사이트의 자치성을 고려하여 분할된 카탈로그 방식을 택하였다. 실험의 결과는 캐시를 이용하는 카탈로그가 캐시를 사용하지 않는 카탈로그보다 좋은 성능을 나타냈다.

1. 서 론

최근 들어, 실시간으로 데이타를 처리해야하는 응용이 등장함에 따라 고성능의 데이타베이스 시스템이 필요하게 되었다. 그리고 주기억장치 용량의 증가와 가격의 하락으로 인해 디스크에 있던 데이타의 전체 혹은 일부를 주기억장치에 저장할 수 있게 되었다.

논리적으로 연관성을 지닌 주기억장치 데이타베이스가 서로 다른 사이트에 분산되어 전산망으로 연결되어 있는 시스템을 분산 주기억장치 데이타베이스 시스템이라고 한다. 그리고, 분산 주기억장치 데이타베이스 또한 저장된 데이타의 효율적인 관리를 위해 카탈로그 정보를 유지한다.

분산 데이타베이스에서의 카탈로그 관리 기법을 살펴보면 크게 중앙 집중식 카탈로그, 완전 중복식 카탈로그, 분할식 카탈로그로 구분할 수 있다. 여기서는 사이트의 자치성[3]이 높고 성능이 좋은 분할식 카탈로그의 세 가지 변형 기법에 대해 성능을 평가한다.

본 논문의 구성은 다음과 같다. 2절에서는 성능을 평가하기 위한 시뮬레이션 모델을 제시한다. 3절에서는 카탈로그 관리 기법에 대해 소개한 후 성능을 평가한다. 마지막으로 4절에서는 본 논문의 결론을 맺는다.

* 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음

2. 시뮬레이션

2.1 가정

시뮬레이션 연구를 위해서는 몇 가지 합리적인 가정들이 필요하다. 본 연구에서는 다음과 같은 가정을 바탕으로 각 알고리즘의 시뮬레이터를 구현했다.

1. 응용의 질의는 카탈로그 읽기 및 쓰기 질의를 포함한다.
2. 동시성 제어 기법은 2단계 로킹 알고리즘(two-phase locking algorithm:2PL)[1]을 사용한다.
3. 교착상태 방지를 위해 wound-wait 알고리즘[4]을 사용한다.
4. 각 트랜잭션의 원자적 완료를 위해 2단계 완료 규약(two-phase locking protocal:2PC)[1]을 사용한다.
5. 데이타를 로킹하거나 접근하는 단위는 페이지이다.
6. 망은 완전히 연결되어 있고 방송 기능을 가진다.
7. 사이트 고장이나 망 분할이 없다.

2.2 시뮬레이션 모델

분산 주기억장치 데이타베이스에서 각 사이트의 모델은 그림 2.1과 같다. CPU에는 메시지 큐와 일반 연산 큐가 있다. 메시지는 다른 사이트에서 들어온 정보나 목적 사이트로 보낼 정보로서 데이타베이스 시스템과 인터럽트 처리 루틴과 같은 별도의 프로그램에 의해 처리되므로 높은 우선 순위를 부여한다. 따라서 CPU는 메시지 큐에 기다리고 있는 메시지가 없는 경우에는 일반 연산 큐에서 트랜잭션을 처리한다.

하나의 트랜잭션은 다음과 같은 처리 과정을 가진다.

각 단말기에서 생성된 트랜잭션은 CPU의 서비스를 받기 위해 일반 연산 큐에 들어가 대기한다. CPU에서 필요한 서비스를 받은 후 다음에 받을 서비스의 종류에 따라서 채널 큐, 또는 메시지 큐로 가서 대기한다. 데이타에 대한 로크 허가를 얻지 못한 경우에는 지연 큐에서 대기한다. 지연 큐에서 대기하고 있던 트랜잭션은 기다리던 로크에 대한 허가를 받으면 CPU의 일반 연산 큐로 가서 대기한다. 트랜잭션의 로그가 필요할 시에는 주기억장치에 기록을 하고 로그버퍼가 다 찼을 경우에는 디스크의 로그파일에 저장한다. 이런 흐름을 반복해서 필요한 연산의 수행을 완료한 트랜잭션은 단말기로 완료 신호를 보낸다. 그러나 철회된 트랜잭션은 처음부터 다시 연산을 수행하기 위해 CPU의 일반 연산 큐로 가서 대기한다.

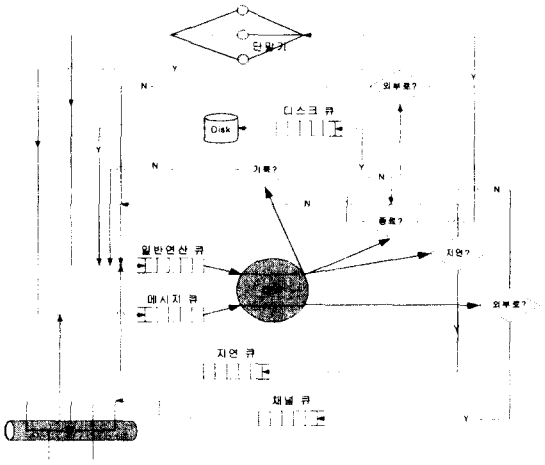

그림 2.1 데이타베이스 사이트 모델.

2.3 성능지수 및 매개변수

일반적으로 성능을 평가하는 지수(performance index)로서 응답시간이나 처리율(throughput) 중 하나를 사용한다. 대부분의 경우 두 지수는 비슷한 결과를 나타내므로 본 실험에서는 응답시간을 주된 성능지수로 선택했다.

응답시간은 트랜잭션의 발생시간으로부터 수행이 완료되어 결과가 반환될 때까지 경과된 시간이다. 트랜잭션이 철회된 경우에는 자동으로 재 시작되고 철회된 시점까지의 수행시간은 응답시간에 추가된다. 응답시간 이외의 몇 가지 보조 성능지수로서 큐 대기시간을 사용했는데, 큐 대기 시간을 통해 어떤 자원에서 병목이 발생하였나를 파악할 수 있다.

시뮬레이터의 입력으로 들어가는 매개변수는 시스템을 위한 매개변수, 트랜잭션의 생성을 위한 매개변수, 자원관리를 위한 매개변수로 구성되어 있다. 표 2.1은 이러한 매개변수를 요약한 것이다.

2.4 시뮬레이터 구현

본 논문에서는 SimJava[5] 언어를 사용하여 각 알고리즘의 시뮬레이터를 구현하였다. SimJava는 자바언어를 기반으로 하는 시뮬레이션 패키지로서 Edinburgh대학에서 개발되었다. SimJava는 프로세스기반 패키지(process-oriented package)이며 프로세스들은 동시에 수행된다. 각 프로세스는 사건 발생을 기다리거나 일정시간동안 대기할 수 있다. 시뮬레이터를 구현한 환경은 WindowNT 5.0으로 운영되는 Intel Pentium-III PC이며, 분산 주기억장치 데이타베이스 시스템이 운영되는 환경은 SunEnterprise 450 시스템과 100MBps 대역폭의 근거리 통신망을 사용한다고 가정하였다.

표 2.1 성능 평가를 위한 매개변수.

시스템을 위한 매개변수		
NumOfSites	3~15 개	사이트 수
NumOfTerminals	3~15 개	사이트당 터미널 수
NumOfPages	5000페이지	전체 페이지 수
트랜잭션 생성을 위한 매개변수		
MeanThinkTime	5000msec	트랜잭션 생성 빈도
ReadXactRatio	0~100%	읽기 연산의 비율
AccessPages	1~20페이지	트랜잭션당 접근 페이지 수
LocalQueryRatio	0~100%	지역 사이트에서 질의 접근 비율
자원 관리를 위한 매개변수		
XactCountPerTerminal	100~500개	한 터미널이 처리해야 할 트랜잭션 개수
PageSize	4KB	페이지의 크기
LogBufferSize	128~512KB	로그 버퍼의 크기
LogFileSize	1MB	로그 파일의 크기
LogWriteTime	20msec	로그 기록 시간
MessageProcessingTime	0.4msec	메시지 처리 시간
LockAcquireTime	0.05msec	로크 획득 시간
LockReleaseTime	0.05msec	로크 반환 시간
DataReadingTime	0.2msec	데이타 읽는 시간
ParsingTime	0.001sec	질의 파싱 시간
NameTime	0.0002sec	네이밍(Naming) 시간
CatalogLookupTime	0.0005sec	카탈로그 찾는 시간
AuthorityCheckingTime	0.0002sec	권한 검사 시간
OptimizationTime	0.003sec	질의 최적화 시간
CodingTime	0.001sec	코드 생성 시간
CPU Hz	480MHz	중앙 처리 장치의 클럭 수
NetTransTime	0.00008msec	네트워크 전송 시간

3. 카탈로그 관리 기법 및 성능평가

카탈로그 관리 기법 중 본 논문에서는 분할식 카탈로그에 대해 성능을 알아본다.

3.1 점진적 캐시를 이용한 분할식 카탈로그(Partitioned Catalogs With Incremental Caching : PCWIC)

각 사이트는 자신이 가지고 있는 데이타 객체에 대한 카탈로그 정보를 유지한다. 이러한 구조는 지역 데이타에 대한 질의가 발생하면 원격 사이트와 통신할 필요가 없어 사이트의 자치성[3]을 높일 수 있는 장점이 있다. 다른 사이트가 가진 데이타와 의존성이 있는 경우 성능을 높이기 위해 원격 사이트의 카탈로그 정보를 지역 사이트에 캐시한다. 질의가 참조하려는 원격 사이트의 카탈로그 정보가 지역에 있으면 캐시만을 이용하여 액세스 계획을 생성한다. 원격 사이트의 카탈로그 정보가 변경

되면 지역 사이트에 캐시되어 있는 카탈로그 정보를 갱신하지는 않는다. 대신에 액세스 계획이 최신의 카탈로그 정보를 이용해서 생성되었는지 확인하기 위해 버전 번호를 비교하여 유효하지 않으면 새로운 캐시 정보를 트랜잭션이 발생한 사이트로 전송한다[2].

3.2 캐시가 없는 분할식 카탈로그(Partitioned Catalogs Without Caching : PCWC)

캐시가 없는 분할식 카탈로그는 지역 사이트에 캐시를 하지 않는다는 점을 제외하면 점진적 캐시를 이용한 분할식 카탈로그와 비슷하다. 원격 사이트의 데이타를 접근하기 위해서는 매번 원격 사이트의 카탈로그 정보를 참조하여야 한다.

3.3 완전 캐시를 이용한 분할식 카탈로그(Partitioned Catalogs With Fully Caching : PCWFC)

완전 캐시를 이용한 분할식 카탈로그는 처음부터 원격 사이트의 카탈로그 정보가 지역 사이트에 캐시가 되어 있다는 점을 제외하면 점진적 캐시를 이용한 분할식 카탈로그와 비슷하다. 이런 구조가 현실적이지는 않지만 캐시가 성능에 미치는 영향을 살펴보기 위해 처음부터 캐시를 가지고 있다고 가정하였다.

3.4 성능 분석

카탈로그의 성능을 분석하기 위해 시스템 내의 사이트 수와 한 사이트 내의 단말기 수를 변화시켰으며 성능지수로는 응답시간을 선택했다. 각 실험에서 별도의 언급이 없는 한 시스템 내의 사이트 수는 7개, 한 사이트 내의 단말기 수는 7개, 로그 버퍼의 크기는 512KB, 읽기 트랜잭션의 비율과 지역 내 질의 비율은 모두 80%이다.

실험 결과를 보면 사이트 수의 변화와 터미널 수의 변화에 따른 응답시간 모두 PCWFC가 좋음을 알 수 있다. 이유는 읽기 트랜잭션의 비율이 쓰기 트랜잭션보다 높으므로 지역 사이트에 캐시되어 있는 정보를 이용하면 네트워크 전송 및 메시지 처리 시간이 들지 않기 때문에 PCWFC가 가장 좋은 성능을 나타내고, PCWC는 트랜잭션이 발생할 때마다 카탈로그 정보를 요청하므로 네트워크 전송 및 메시지 처리시간으로 인해 나쁜 성능을 나타내기 때문이다.

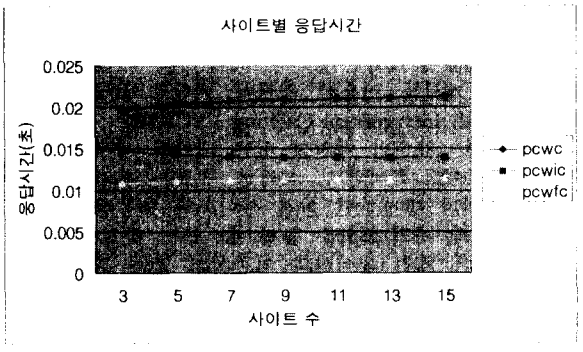

그림 3.1 사이트 수의 변화에 따른 응답시간.

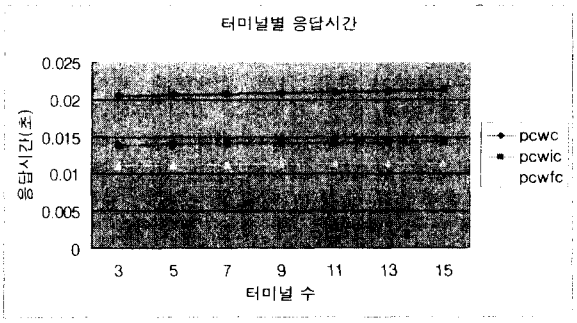

그림 3.2 터미널 수의 변화에 따른 응답시간.

4. 결론 및 향후 연구 방향

중앙 집중식 주기억장치 데이타베이스를 분산 시스템으로 확장하려 할 때 중복제어 및 카탈로그의 확장이 필수적이다. 이중 카탈로그 관리기법은 분산 주기억장치 데이타베이스의 성능에 큰 영향을 미친다. 본 논문에서는 여러 가지 카탈로그 관리 기법 중에서 사이트의 자치성이 뛰어나며 성능이 좋은 분할식 카탈로그의 세 가지 변형인 캐시가 없는 분할식 카탈로그, 점진적 캐시를 이용한 분할식 카탈로그, 완전 캐시를 이용한 분할식 카탈로그의 성능을 비교 평가했다.

본 논문에서 사용된 성능지수로 전체 트랜잭션의 응답시간을 선택하였으나 앞으로 읽기의 가용성을 분석하기 위해 읽기와 쓰기 연산의 성능을 구분하고 자원의 병목지점을 알아보기 위해서 각 자원마다 소비량을 측정할 필요가 있다. 또한 다양한 성능 지수 및 매개변수를 사용한 성능평가 및 세부적인 모델에 따른 성능평가 역시 필요하다.

5. 참고 문헌

[1] J.Gray, "Notes on Data Base Operating Systems," Operating Systems: An Advanced Course, Lecture Notes in Computer Science:60, Edited by: R.Bayer, et al., Springer-Verlag, 1979.

[2] B.G.Lindsay, Object Naming and Catalog Management for a Distributed Database Manager, IBM Research Report RJ2914, San Jose, Calif., Aug. 1980.

[3] B.Glindsay and P.G.Selinger, "Site Autonomy Issues in R*: A Distributed Database Management System," IBM Research Report RJ2927, San Jose, Calif., Sept. 1980.

[4] D.J.Rosenkrantz, et al., "System Level Concurrency Control for Distributed Database Systems", ACM Trans. on Database Systems 3(2), June 1978, PP.178-198.

[5] R. McNab and F.W. Howell, "Using Java for Discrete Event Simulation," in Proc. 12th UK Computer and Telecomm. Performance Engineering Workshop, Univ. of Edinburgh, pp. 219-228.