분산 XML 데이터베이스에서 질의 컴파일 시 카탈로그 관리 기법의 성능 평가*

장건업° 홍의경
서울시립대학교 컴퓨터과학부
{atlas°, ekhong}@venus.uos.ac.kr

Performance Evaluation of Catalog Management Schemes for Distributed XML Database at the Query Compile Time

Gun Up Jang° Eui Kyeong Hong
Dept. of Computer Science, University of Seoul

요 약

최근 컴퓨팅 환경은 클라이언트-서버(client-server) 환경에서 웹(World Wide Web)을 기반으로 한 분산 컴퓨팅(distributed computint) 환경으로 변화하고 있다. 그에 따라 XML 문서의 사용과 XML 문서의 양이 급속하게 증가하였다. 언제 어디서나 쉽게 필요한 XML 문서에 접근해야하며, 이러한 응용을 위해 짧은 시간 내에 그 정보를 전달할 수 있어야 한다. 이에 따라 분산 환경에서의 XML 문서의 처리가 요구된다. XML 데이터를 분산 데이터베이스의 특성을 이용하여 저장, 관리, 질의하는 분산 XML 데이터베이스 시스템(Distributed XML Database System)의 사용의 필요성이 증가하고 있다. 이에 따라, 사이트의 자치성, 질의 최적화, 데이터의 투명성 등에 큰 영향을 미치는 분산 XML 데이터베이스 시스템에서의 카탈로그 관리 기법의 연구의 필요성이 증가하게 된다.

본 논문에서는 중앙 집중식 카탈로그와 완전 중복식 카탈로그, 분할식 카탈로그를 분산 XML 데이터베이스 시스템에서 CPU 비용, I/O 비용, 동시성 제어, 이단계 완료 프로토콜, 큐잉 지연 등을 모두 고려한 모델을 설계하였고, 이를 시뮬레이터로 구현하여 각 카탈로그 관리 기법들의 성능을 합리적인 환경 설정을 통해 시뮬레이션함으로써 카탈로그 관리 기법들의 성능을 평가하였다.

1. 서 론

최근 컴퓨팅 환경은 과거 1980년대부터 1990년대 중반까지 널리 활용되던 클라이언트-서버 환경에서 웹을 기반으로 한 분산 컴퓨팅(distributed computing) 환경으로 변화하고 있다[1]. 그에 따라 XML[2] 문서의 사용과 XML 문서의 양이 급속하게 증가하였으며, 언제 어디서나 쉽게 필요한 XML 문서에 접근해야 하며, 이러한 응용을 위해 짧은 시간 내에 그 정보를 전달할 수 있어야 한다. 이에 따라 분산 환경에서의 XML 문서의 처리가 요구된다. XML 데이터를 분산 데이터베이스의 특성을 이용하여 저장, 관리하고, 질의를 하는 분산 XML 데이터베이스 시스템(Distributed XML Database Management System)의 사용이 증가하고 있다[3,4]. 따라서 분산 XML 데이터베이스 시스템에서 카탈로그 관리 기법을 연구할 필요성이 증가하게 된다. 카탈로그 관리 기법은 매우 중요한 데이터베이스 관리 기법 중의 하나로 사이트 자치성(site autonomy), 질의 최적화(query optimization), 뷰 관리(view management), 권한 관리 기법(authorization mechanism), 데이터의 투명성 등에 큰 영향을 미친다[5].

분산 XML 데이터베이스 시스템의 카탈로그는 중앙 집중식, 완전 중복식, 분할식으로 구분할 수 있다. 중앙 집중식은 분산된 모든 사이트에 대한 전체 카탈로그가 하나의 중앙 사이트에 저장되는 방식이며, 완전 중복식은 분산된 사이트의 카탈로그가 모든 사이트에 중복해서 저장되는 방식이다. 분할식은 원격 사이트의 객체를 참조하는 질의가 발생하게 되면 지역 사이트로 원격 사이트의 카탈로그의 일부분을 가져와 관리하는 방식이다[6].

본 논문에서는 중앙 집중식 카탈로그와 완전 중복식 카탈로그, 분할식 카탈로그에서 캐시가 없는 방법, 완전 캐시를 하는 방법에 대하여 분산 XML 데이터베이스 시스템에서 CPU 비용, I/O 비용, 동시성 제어, 2-단계 완료 프로토콜, 큐잉 지연 등을 모두 고려한 모델을 설계하였고, 이를 시뮬레이터로 구현하고 합리적인 환경 설정을 통해 시뮬레이션함으로써 카탈로그 관리 기법들의 성능을 평가하였다.

본 논문의 구성은 다음과 같다. 2절에서는 카탈로그 관리 기법과 분산 XML 데이터베이스 시스템에서의 질의 처리 과정에 대해서 알아보고, 3절에서는 시뮬레이션 모델을 소개하고, 4절에서는 카탈로그 관리 기법에 대해 시뮬레이션 모델을 사용하여 성능을 평가한다. 5절에서는 논문의 결론을 맺는다.

* 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음

2. 관련 연구

2.1 카탈로그 관리 기법

카탈로그는 시스템 데이터베이스라고도 하며, 이는 데이터베이스 시스템이 관리하는 여러 객체들에 대한 정보를 포함한다[7]. 객체들에는 릴레이션, 뷰, 인덱스, 사용자 액세스 모듈(수행 가능한 코드), 액세스 권한 등이 포함된다. 카탈로그 자체도 사용자의 데이터 릴레이션과 같이 릴레이션들로 구성된다. 따라서 릴레이션 형태로 저장된 카탈로그들도 질의 언어를 사용하여 액세스할 수 있고, 사용자 릴레이션에 사용되는 것과 동일한 동시성 제어와 회복 기법에 의해 유지된다. 카탈로그는 사용자가 질의에서 정의한 객체를 낮은 수준의 객체 식별자와 객체의 위치에 사상하는 역할을 한다. 카탈로그는 사용자가 질의에서 참조한 데이터베이스 객체들의 스키마를 정의하고 이들 객체에 접근할 수 있는 액세스 경로를 명시한다. 또한 카탈로그는 질의 최적화, 사용자의 데이터베이스 객체에 대한 권한, 데이터베이스 객체들 간의 의존성 등에 대한 통계 자료를 제공한다.

2.1.1 중앙 집중식 카탈로그(Centralized Catalogs)

중앙 집중식 카탈로그는 분산 데이터베이스 시스템을 구성하고 있는 모든 사이트의 카탈로그를 오직 하나의 중앙 사이트에 저장하고 관리하는 방식이다. 중앙 집중식 카탈로그는 신뢰성이나 가용성 면에서 분산 데이터베이스 시스템의 이점을 활용하지 못한다. 이 방법은 지역 사이트에 있는 데이터를 접근할 때도 중앙 사이트에 카탈로그 정보를 요청해야 한다. 중앙의 사이트의 고장이 발생하면 전체 시스템을 사용할 수 없게 되고, 중앙 사이트가 병목지점이 될 가능성이 많다. 이러한 구조는 주로 스타형 네트워크에서만 사용된다[5].

2.1.2 완전 중복식 카탈로그(Fully Replicated Catalogs)

완전 중복식 카탈로그는 분산 데이터베이스 시스템을 구성하고 있는 사이트마다 모든 카탈로그를 저장하는 방법이다. 이러한 완전 중복식 카탈로그는 지역 사이트에서 트랜잭션이 많이 발생할 경우 네트워크 통신비용이 적게 들기 때문에 카탈로그 정보를 참조하는 성능이 좋다. 그러나 중복되어 있는 모든 카탈로그가 일관성 있게 갱신되어야 하기 때문에 쓰기 질의가 비효율적이고, 분산 데이터베이스에 새로운 사이트를 추가하기 쉽지 않다. 또한 한 사이트에서 결코 참조되지 않는 카탈로그 정보도 보유하고 있어야 하기 때문에 공간의 낭비가 있다[5].

본 논문에서는 완전 중복식 카탈로그의 일관된 상태를 유지하기 위해 ROWA(Read One Write All) 알고리즘[8]을 사용한다. ROWA 알고리즘에서는 읽기 질의인 경우, 지역 사이트에 다른 원격 사이트의 카탈로그의 사본 모두를 저장하고 있으므로 저장된 지역 사이트의 카탈로그를 읽으면 된다. 하지만 쓰기 질의인 경우 쓰기 질의의 요청이 들어온 사이트는 카탈로그 사본을 저장하고 있는 모든 사이트에 쓰기 요청 메시지를 전달한다. 쓰기 요청을 받은 사이트는 카탈로그에 배타 로크를 하고 카탈로그를 갱신한다. 만약 배타 로크를 얻지 못한다면 허용될 때까지 기다렸다가 갱신한다.

2.1.3 분할식 카탈로그(Partitioned Catalogs)

분할식 카탈로그에서는 각 사이트가 자신이 저장하고 있는 객체들에 대한 카탈로그 정보만 갖고 있다. 이러한 구조는 지역 데이터만 참조하는 질의가 입력되면 원격 사이트와 통신할 필요가 없어 사이트의 자치성을 높일 수 있는 장점이 있다. 처음에는 원격 카탈로그 정보 중 지역 디스크에 저장된 것이 없지만, 어떠한 사이트도 원격 사이트의 카탈로그 정보를 지역 사이트에 저장할 수 있다. 나중에 동일한 원격 사이트의 객체를 참조하는 질의를 컴파일할 때 원격 사이트를 액세스하지 않고 지역 사이트에 저장된 카탈로그 정보를 사용하여 액세스 플랜(access plan)을 생성한다. 원격 사이트의 카탈로그 정보가 갱신되면 지역 사이트에 있는 카탈로그 정보를 동시에 갱신하지는 않는다. 대신에 액세스 플랜이 최신의 카탈로그 정보를 이용하여 생성되었는지를 확인하기 위해 버전 번호와 함께 원격 사이트의 액세스 플랜을 전송하고 액세스 플랜을 받은 원격 사이트는 버전을 비교하여 유효하지 않으면 새로운 카탈로그 정보를 액세스 플랜을 전송한 사이트로 보낸다[6].

본 논문에서는 분할식 카탈로그 방법 중에서 캐시가 없는 분할식 카탈로그 방법(Partitioned Catalogs Without Caching)과 완전 캐시를 이용한 분할식 카탈로그 방법(Partitioned Catalogs With Full Caching)을 실험하였다. 캐시가 없는 분할식 카탈로그는 지역 사이트에 원격 사이트의 카탈로그 정보를 저장하지 않는 방법이다. 이 방법을 사용하였을 때, 원격 사이트의 데이터를 접근하기 위해서는 매번 원격 사이트의 카탈로그 정보를 참조하여야 한다. 완전 캐시를 이용한 분할식 카탈로그는 원격 사이트의 카탈로그 정보를 지역 사이트에 저장하는 방법이다. 이 방법을 사용하였을 때, 원격 사이트의 데이터를 접근할 때 지역 사이트의 카탈로그 정보가 최신의 정보라면 원격 사이트에 접근 없이 액세스 플랜을 생성할 수 있다[5,6].

완전 중복식 카탈로그와 분할식 카탈로그의 차이점은 다음과 같다. 첫째, 읽기 질의인 경우, 완전 중복식 카탈로그는 항상 모든 원격 사이트의 정보를 저장하고 있으므로, 원격 사이트의 질의가 입력되었을 때 지역 사이트의 카탈로그를 참조하여 질의를 컴파일 하는 것이 가능하다. 그러나 분할식 카탈로그는 참조할 모든 원격 사이트의 카탈로그 정보를 저장하고 있지 않기 때문에, 원격 사이트의 카탈로그 정보를 참조하는 읽기 질의가 입력되면 원격 사이트의 카탈로그 정보를 지역 사이트로 전송받아 질의를 컴파일 하여야 한다. 둘째, 쓰기 질의인 경우, 분할식 카탈로그는 카탈로그의 갱신이 발생하여도 원격 사이트에 저장된 지역 사이트의 카탈로그를 갱신할 필요가 없지만, 완전 중복식 카탈로그는 중복되어 있는 모든 사이트의 카탈로그가 갱신되어야 한다.

2.2 분산 XML 데이터베이스에서의 질의 처리 과정

사용자 질의는 데이터에 대해 읽기 연산을 수행하는지 쓰기 연산을 수행하는지에 따라 읽기 질의 혹은 쓰기 질의로 구분한다. 본 논문에서는 카탈로그를 기준으로 질의 처리 과정에서 카탈로그에 대한 읽기 연산이 수행되면 읽기 질의, 카탈로그에 대한 쓰기 연산이 수행되면

쓰기 질의라고 정의한다. 카탈로그에 대한 읽기 및 쓰기 질의는 표 1에 정리하였다.

본 논문에서는 XML 문서가 관계 데이터베이스 시스템에 분할 저장[9]되어 있다고 가정하고, 그림 1과 같이 XQuery[10] 질의에 대해서는 XQuery 질의를 SQL문으로 변환[11,12,13]하고 변환된 SQL 질의를 컴파일[14,15]하여 액세스 모듈이 생성된다고 가정한다.

표 1. 카탈로그에 대한 읽기 및 쓰기 질의

구분	명령어	기능
읽기 질의	SELECT	릴레이션으로부터 데이터를 검색
	INSERT	릴레이션에 튜플을 삽입
	DELETE	릴레이션으로부터 튜플을 삭제
	UPDATE	릴레이션의 튜플을 갱신
쓰기 질의	CREATE TABLE	테이블을 생성
	DROP TABLE	테이블을 삭제
	ALTER TABLE	테이블정보에 추가, 수정, 삭제
	CREATE INDEX	인덱스를 생성
	DROP INDEX	인덱스를 삭제
	CREATE VIEW	뷰를 생성
	DROP VIEW	뷰를 삭제
	GRANT	사용자에게 권한 부여
	REVOKE	사용자의 권한을 회수
	CREATE SYNONYM	릴레이션이나 뷰의 별명을 정의
	DROP SYNONYM	릴레이션이나 뷰의 별명을 삭제
	UPDATE STATISTICS	최적화 질의 처리를 위해 DBMS가 관리하는 통계 정보를 갱신

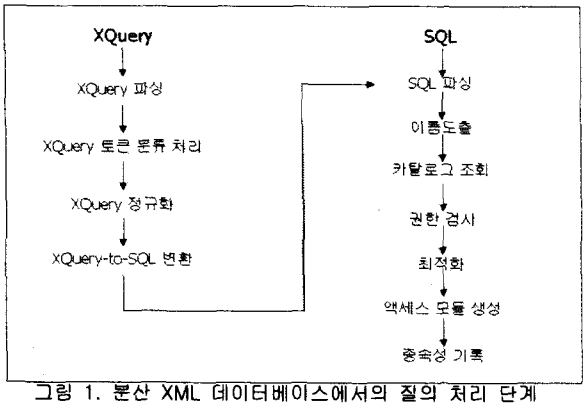

그림 1. 분산 XML 데이터베이스에서의 질의 처리 단계

3. 시뮬레이션

3.1 가정

시뮬레이션 연구를 위해서는 몇 가지 합리적인 가정들이 필요하다. 본 연구에서는 다음과 같은 가정을 바탕으로 카탈로그 관리 기법의 시뮬레이터를 구현하였다.

1. 응용의 질의는 카탈로그 읽기 및 쓰기 질의를 포함한다.
2. 동시성 제어 기법은 2단계 로킹 알고리즘(two-phase locking algorithm(2PL))을 사용한다.
3. 교착 상태 방지를 위하여 wound-wait 알고리즘을 사용한다.
4. 각 트랜잭션의 원자적 완료를 위해 2단계 완료 규약(two-phase commit protocol(2PC))을 사용하다.
5. 데이터 로킹과 접근은 레코드 단위로 한다.
6. 네트워크는 완전히 연결되어 있으며 브로드캐스팅 기능을 갖는다.
7. 사이트의 고장이나 네트워크 연결 정지는 발생 하지 않는다.

3.2 시뮬레이션 모델

분산 XML 데이터베이스 시스템은 근거리 통신망에 연결되어 있는 여러 개의 DB 사이트들로 구성되며, 각 DB 사이트는 그림 2와 같은 구조를 갖는다. 한 사이트는 여러 개의 터미널(또는 여러 명의 사용자), CPU, 디스크와 5개의 큐로 구성된다. CPU는 일반연산 큐나 메시지 큐에 있는 요청들을 처리한다. 일반연산 큐에서 대기하고 있는 요청들은 질의 컴파일, 동시성 제어, 2단계 완료 처리, 카탈로그 읽기/쓰기 등이다. 메시지 큐에서 대기하는 트랜잭션은 다른 사이트로부터 받은 요청들이거나 다른 사이트가 처리해야 할 요청들이다. 메시지 큐에서 대기하는 요청들은 일반연산 큐에서 대기하는 요청들보다 우선순위가 높다. 따라서 CPU는 메시지 큐에서 대기하는 요청이 없는 경우에 일반 연산 큐에서 대기하는 요청을 처리한다.

또한 DB 사이트에는 지연 큐, 채널 큐, 디스큐 큐가 있다. 지연 큐에는 동시성 제어로 인해 지연되는 요청들이 저장된다. 이러한 요청들은 지연 조건이 해결되는 순간에 일반연산 큐로 옮겨진다. 채널 큐에는 네트워크를 통해 다른 사이트로 보내질 요청들이 저장된다. 네트워크 전송이 가능하게 되면 이러한 요청들은 해당 사이트로 순서대로 보내진다. 디스크 큐에는 레코드를 디스크에 쓰려는 요청들과 로그를 디스크에 쓰려는 요청이 대기한다.

하나의 트랜잭션은 다음과 같은 처리 과정을 거친다. 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. 데이터에 대한 로크 허가를 얻지 못한 경우에는 지연 큐에서 대기한다. 지연 큐에서 대기하고 있던 트랜잭션은 기다리던 로크에 대한 허가를 얻으면 CPU의 일반 연산 큐로 가서 대기한다. 트랜잭션의 로그가 필요할 시에는 기록을 하고 로그 버퍼가 다 찼을 경우에는 디스크의 로그 파일에 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.

3.3 성능 지수 및 매개변수

일반적으로 성능을 평가하는 지수(performance index)로서 응답시간이나 처리율(thoughput) 중 하나를 사용한다. 대부분의 경우 두 지수는 비슷한 결과를 나타내므로 본 논문에서는 응답시간을 주된 성능 지수로 선택하였다.

응답 시간은 트랜잭션의 발생 시간으로부터 수행이 완료되어 결과가 반환될 때까지 경과한 시간이다. 트랜잭션이 철회된 경우에는 자동으로 재시작되고 철회된 시점까지의 수행시간은 응답시간에 추가된다. 응답시간 이외의 몇 가지 보조 성능지수로서 큐 대기 시간을 사용했는

데, 큐 대기 시간을 통해 어떤 자원에서 병목이 발생하였나를 파악할 수 있다. 시뮬레이터의 입력으로 들어가는 매개변수는 시스템을 위한 매개변수, 트랜잭션의 생성을 위한 매개변수, 자원 관리를 위한 매개변수로 구성되어 있다. 표 2는 이러한 매개변수를 정리한 것이다.

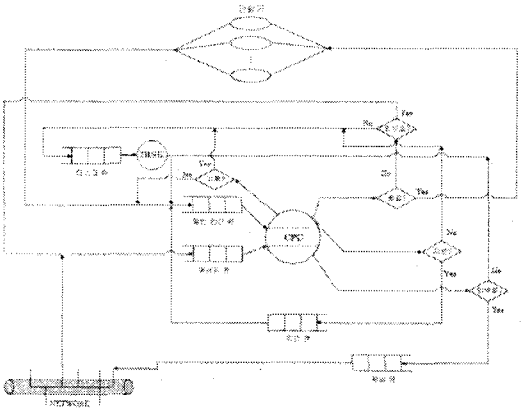

그림 2 분산 XML 데이터베이스의 사이트 모델

표 2. 성능평가를 위한 매개변수

시스템을 위한 매개변수		
NumberOfSite	3~10	분산 XML 데이터베이스 시스템 내의 사이트 수
NumberOfTerminal	1~50	하나의 사이트 내의 터미널의 수
NumberOfRelation	100	하나의 데이터베이스 시스템 내의 릴레이션의 수
NumberOfTuple	2000	하나의 릴레이션의 투플의 수
트랜잭션 생성을 위한 매개변수		
MeanThinkTime	3000ms	터미널들에서의 평균 트랜잭션 생성 시간
ReadXactRatio	0~100%	읽기 질의의 비율
LocalXactRatio	0~100%	지역 사이트에서 질의 접근 비율
AccessTupies	1~200개	트랜잭션 당 접근 레코드 수
자원 관리를 위한 매개변수		
XactCountPerTerminal	100~600개	하나의 터미널에서 처리해야 할 트랜잭션의 수
PageSize	4KB	디스크 페이지의 크기
LogBufferSize	512KB	로그 버퍼의 크기
LogFileSize	4MB	로그 파일의 크기
LogWriteTime	10ms	로그 기록 시간
MessageProcessingTime	2ms	메시지 처리 시간
LockAcquireTime	1ms	록크 획득 시간
LockReleaseTime	1ms	록크 반환 시간
DiskSeekTime	8ms	평균 디스크 탐색 시간
LockingTime	2ms	병행 수행 제어, 데드록의 처리 등에 필요한 CPU 시간
XQueryParsingTime	30ms	XQuery 질의를 파싱하는데 걸리는 시간
XQueryProcessingTime	50ms	XQuery 질의를 처리하는데 걸리는 시간
XQueryNormalizationTime	500ms	XQuery 질의의 정규화에 걸리는 시간
XQueryToSQLTime	1000ms	XQuery 질의를 SQL 질의로 변환하는데 걸리는 시간
SQLParsingTime	10ms	SQL 질의를 파싱하는데 걸리는 시간
CatalogLookupTime	5ms	카탈로그를 조회하는데 걸리는 시간
NamingTime	2ms	분산 질의의 이름을 도출하는데 걸리는 시간
AuthenticationCPUTime	2ms	질의에 대해 권한을 검사하는데 걸리는 시간
OptimizationCPUTime	3ms	질의를 최적화하는데 걸리는 시간
CodeGenerationCPUTime	1ms	코드 생성 시간
CPU Hz	1.8Hz	중앙처리장치의 클럭 수
NetTransTime	0.00008ms	바이트당 전송 시간

3.4 시뮬레이터 구현

본 논문에서는 SimJava[16]언어를 사용하여 카탈로그 관리 기법의 시뮬레이터를 구현하였다. SimJava는 자바 언어를 기반으로 하는 시뮬레이션 패키지로서 Edinburgh대학에서 개발되었다. SimJava는 프로세스 기반 패키지이며 프로세스들은 동시에 수행된다. 각 프로세스는 사건 발생을 기다리거나 일정 시간 동안 대기할 수 있다. 시뮬레이터를 구현한 환경은 Windows 2003 Server로 운영되는 Intel Pentium 4 PC이며, 분산 XML 데이터베이스 시스템이 운영되는 환경은 Intel Xeon Server 시스템과 100MBbps 대역폭의 근거리 통신망을 사용한다고 가정한다.

4. 카탈로그 관리 기법의 성능 평가

카탈로그 관리 기법의 성능을 분석하기 위해 분산 XML 데이터베이스 내의 사이트의 수와 한 사이트 내의 단말기 수를 변화시켰으며 성능지수로는 질의의 컴파일 응답시간을 선택하였다. 편의상 중앙 집중식 카탈로그는 CC로, ROWA 알고리즘은 사용하는 완전 중복식 카탈로그는 ROWA로, 캐시가 없는 분할식 카탈로그는 PCWC로, 완전 캐시를 이용한 분할식 카탈로그는 PCWFC로 나타낸다.

4.1 사이트의 수에 변화에 따른 성능 평가

이 실험에서는 사이트의 수를 3~15개로 변화시켜가면서 카탈로그 읽기/쓰기 질의의 평균 응답시간을 측정하였다. 이 실험에서 분산 XML 데이터베이스 시스템의 한 사이트의 터미널의 수는 10개이며, 카탈로그의 읽기 질의의 비율과 지역 질의의 비율은 80%로 가정하였다. 이 실험의 결과는 그림 3과 그림 4에 나타나 있다. 표 3은 사이트의 수가 7인 경우에 자원 사용 비율과 큐 대기 시간의 비율을 나타낸 표이다.

PCWFC의 경우, 각 사이트는 질의의 80%에 해당하는 카탈로그 읽기 질의를 처리하기 위해 캐시된 카탈로그를 사용하기 때문에 다른 방법에 비해 성능이 좋다. 반면에 PCWC의 경우 각 사이트는 원격 카탈로그가 필요한 경우 항상 원격 사이트의 카탈로그를 가져오므로, 네트워크를 통한 데이터 전송과 메시지 처리를 위한 오버헤드가 발생한다.

ROWA의 경우에는 지역 사이트에 카탈로그의 사본이 있기 때문에 읽기 질의에서는 지역 사이트의 카탈로그를 읽고 컴파일이 이루어지지만, 쓰기 질의에서는 모든 사이트의 카탈로그를 갱신하기 위해서 배타 로크를 얻기 위해 대기하는 큐잉 지연이 발생하고, 메시지 처리와 전송을 위한 오버헤드가 발생하기 때문에 읽기 질의와 쓰기 질의의 응답시간이 차이가 난다.

CC의 경우에는 성능이 가장 나쁘다. 중앙의 하나의 사이트에서 카탈로그 정보를 참조하여야 하기 때문에 큐잉 지연이 발생하고 카탈로그를 저장하고 있는 중앙 사이트의 오버헤드가 크다.

4.2 사이트 당 터미널의 수의 변화에 따른 성능 평가

이 실험에서는 각 사이트 당 터미널의 수를 5~50개로 변화시켜가면서 카탈로그 읽기/쓰기 질의의 평균 응답기간을 측정하였다. 이 실험에서 분산 XML 데이터베이스 시스템의 사이트 수는 7개이며, 카탈로그의 읽기 질의의 비율과 지역 질의의 비율은 80%로 가정하였다. 이 실험의 결과는 그림 5와 그림 6에 나타나 있다.

실험 결과 CC가 가장 성능이 나쁘고, PCWC가 성능이 가장 좋다는 것을 알 수 있다.

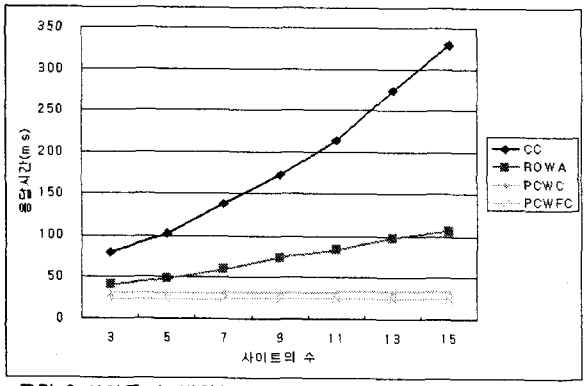

그림 3 사이트 수 변화에 따른 읽기 트랜잭션의 평균 응답시간

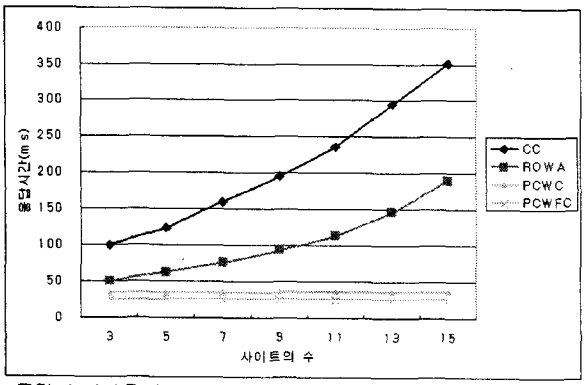

그림 4 사이트 수 변화에 따른 쓰기 트랜잭션의 평균 응답시간

표3. 사이트 수가 7인 경우, 자원 사용 비율과 큐 대기시간

자원 사용 비율(사이트의 수가 7인 경우)

	CC	ROWA	PCWC	PCWFC
CPU cost	0.98%	3.50%	18.88%	16.08%
Disk cost	0.94%	10.07%	40.90%	39.72%
Communication cost	0.06%	0.07%	0.45%	0.39%
Queuing delay	98.02%	86.36%	39.77%	43.81%

큐 대기시간(사이트의 수가 7인 경우)

	CC	ROWA	PCWC	PCWFC
일반연산큐	0.02%	0.40%	8.12%	7.14%
메시지 큐	0.07%	0.20%	1.02%	0.70%
블록 큐	2.72%	2.47%	0.12%	0.09%
디스크 큐	97.19%	96.91%	90.70%	92.04%
채널 큐	0.00%	0.02%	0.04%	0.03%

분할식 카탈로그 방법에서, 읽기 질의 경우에 PCWC가 PCWFC보다 더 우수한 성능을 보인다. PCWC와 PCWFC의 성능 차이는 디스크의 접근 빈도와 재컴파일 비율의 차이 때문이다. 재컴파일 비율은 전체 읽기 질의 중에 재컴파일된 읽기 질의의 비율이다. PCWC에서는 원격 사이트의 카탈로그 정보를 필요할 때마다 요구하게 되므로 디스크에 대한 접근은 일어나지 않고, 재컴파일은 거의 일어나지 않는다. 반면에, PCWFC의 경우에는 원격 사이트의 카탈로그 정보를 캐시로 지역 사이트에 저장하게 되므로 디스크에 접근이 빈번히 일어난다. 디스크에 저장된 원격 사이트의 카탈로그 정보는 다음에 지역 사이트에서 동일한 원격 사이트의 객체에 접근할 때 사용될 수 있다. 하지만 원격 사이트의 객체가 수정이 되면, 지역 사이트의 카탈로그 정보가 최신의 정보가 아니므로, 원격 사이트의 최신의 카탈로그 정보를 지역 사이트로 가져와 새로이 컴파일이 이루어지게 된다. 쓰기 질의인 경우에는 PCWC과 PCWFC의 성능의 차이가 거의 없다.

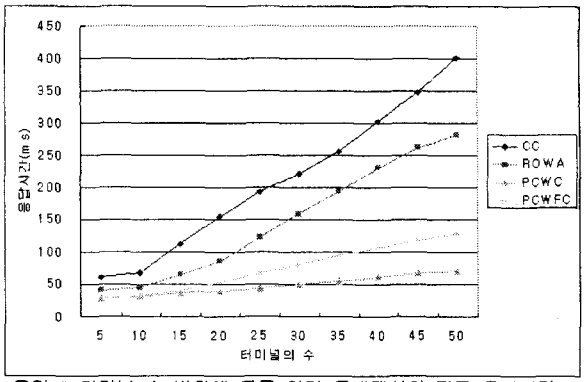

그림 5 터미널 수 변화에 따른 읽기 트랜잭션의 평균 응답시간

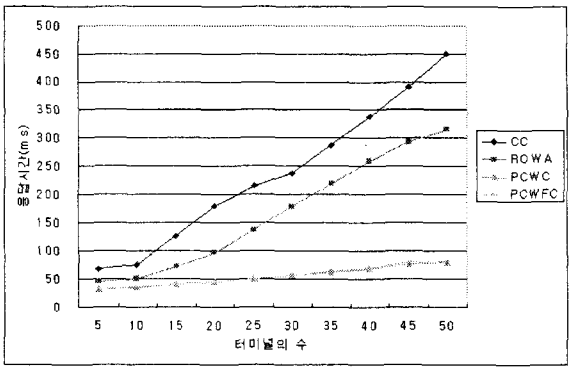

그림 6 터미널 수 변화에 따른 쓰기 트랜잭션의 평균 응답시간

5. 결론 및 향후 연구 방향

중앙 집중식 XML 데이터베이스 시스템을 분산 시스템으로 확장하려 할 때, 중복제어 및 카탈로그의 확장이 필수적이다. 본 연구에서는 분산 XML 데이터베이스 시스템에서 사용 가능한 여러 가지 카탈로그 관리 기법들을 소개하고 성능 평가를 수행하였다. 성능 평가를 수행하기 위해 분산 XML 데이터베이스 모델을 제안하였고, 각 기법에 대해 시뮬레이션을 수행하였다. 시뮬레이션 결과, 중앙 집중식 카탈로그 관리 기법이 가장 성능이 좋지 않았고, 분할식 카탈로그 관리 기법이 가장 성능이 좋은 것으로 나왔다.

본 논문에서 사용된 성능 지수로 읽기/쓰기 트랜잭션의 응답시간을 선택하였지만, 앞으로 자원의 병목 지점을 알아보기 위해서 각 자원마다 소비량을 측정할 필요가 있다. 또한 다양한 성능 지수 및 매개변수를 사용한 성능 평가 및 세부적인 모델에 따른 성능 평가 역시 필

요하다.

6. 참고 문헌

[1] H. Wilhelm, "Information System Integration," Commnications of the ACM 43(6), pp. 32-38, June 2000.

[2] World Wide Web Consortium, Extensible Mark Up Language (XML) 1.0, W3C Recommendation 4, Feb. 2004.

[3] S. Abiteboul, et al., "A Framework Distributed XML Data Management," Int'l Conf. on Extending Database Technoloy, Munich, Germany, March 2006.

[4] S. Hastings, et al., "XML Database Support for Distributed Execution of Data-intensive Scientific Workflows," ACM SIGMOD Record 34(3), pp.50-55, Sept. 2005.

[5] E. K. Hong, "Performance Evaluation of Catalog Architectures in Distributed Database Systems," Ph.D Dissertation, Dept. of Computer Science, KAIST, 1991.

[6] B .G. Lindsay and P. G. Selinger, "Site Autonomy Issue in R*: A Distributed Database Management System," IBM Research Report RJ 2927, San Jose, Calif., Sept. 1980.

[7] R. Elmasri and S. B. Navathe, Fundamental of Database Systems. Addison Wesley, 4th Edition, 2003.

[8] M. Rabinovich and E. D. Lazowska, "An Efficient and Highly Available Read-One Write-All Protocol for Replicated Data Management," Proc. of the 2nd Int'l Conf. on Parallel and Distributed Information Systems, San Diego, US, pp.56-65, Jan. 1993.

[9] J. Shanmugasundaram, et al., "Relational Databases for Querying XML Documents: Limitations and Opportunities," Proc. of the 25th VLDB Conf., Edinburgh, Scotland, pp.302-314, Sept. 1999.

[10] World Wide Web Consortium, XQuery 1.0: An XML Query Language, W3C Candidate Recommendation 8, June 2006.

[11] D. Suciu, "Distributed Query Evaluation on Semistructured Data," ACM Trans. on Database Systems 27(1), pp.1-62, March 2002.

[12] I. Manolescu, et al., "Answering XML Queries Over Heterogeneous Data Source," Proc. of the 27th VLDB Conf., Rome, Italy, pp.241-250, Sept. 2001.

[13] I. Tatarinov, et al., "Updating XML," Proc. of ACM SIGMOD, pp.413-424, Santa Barbara, US, May 2001.

[14] G. M. Lohman, et al., "Query Processing in R*: A Distributed Database Manager," IBM Research Report RJ 3740, San Jose, Calif., Jan. 1983.

[15] D. Kossmann, "The State of the Art in Distributed Query Processing," ACM Computing Survey 32(4), pp.422-469, Dec. 2000.

[16] R. McNab and F. W. Howell, "Using Java for Discrete Event Simulation," in Proc. 12th UK Computer and Telecommunication, Performance Engineering Workshop, Univ. of Edinburgh, pp.219-228, Sept. 1996.