고가용성을 위한 주기억장치 DBMS ALTIBASE™의 이중화 관리 기법☆

Management of Database Replication in Main Memory DBMS ALTIBASE™ for High Availability

이 규 웅*
Lee, Kyu Woong

요 약

ALTIBASE™ 시스템은 메인 메모리를 주 저장장치로 사용하는 관계형 주기억장치 DBMS이다. 본 논문에서는 최근 데이터베이스 응용들의 요구사항으로 부각되고 있는 데이터베이스의 고가용성과 실시간 데이터베이스 시스템의 높은 트랜잭션 처리율을 동시에 보장하기 위하여 ALTIBASETM 시스템의 구조 및 설계에 대하여 기술한다. 특히, 고가용성 보장을 위한 ALTIBASETM 시스템의 이중화 기능에 대하여 논의한다. 이중화 기법의 분류를 통해 다른 기법과의 비교를 하였으며, 이중화 관리를 위한 프로세스 구조 및 이중화 통신 모델, 프로토콜에 대하여 설명한다. 또한, 다양한 실험을 통하여 트랜잭션 처리율을 측정하였으며, 이중화 기능을 지원하기 위해 요구되는 오버헤드를 독립 시스템의 성능과 비교하여 측정하였다.

Abstract

ALTIBASE™ is the relational main-memory DBMS in which a main memory is primarily used as the main storage device. We present the database replication strategies and techniques of the ALTIBASETM system in order to meet the requirement of high availability and efficient transaction processing. Our process architecture for replication management and its communication model are proposed, and database replication protocols are also described. We show the experimental result of transaction processing rate with various DBMS parameters and overall performance of database replication system as compared to standalone system.

Keyword : Main Memory, DBMS, Replication, Transaction, Consistency

1. 서 론

실시간 산업 분야 뿐만 아니라 컴퓨터를 이용하는 전반적인 일반 산업분야에서도 빠른 성능과 응답을 요구하는 데이터베이스 응용 시스템들이 증가하고 있다. 이러한 요구사항을 만족하기 위해서 디스크 기반의 데이터베이스 시스템들은 디스크에 위치한 모든 사용자 데이터를 메인 메모리에 적재하여 사용할 수 있는 대체 방안을 제시하고 있으나, 순수한 주기억장치 데이터베이스 관리 시스템(main memory DBMS) 만큼의 충분한 성능과 기능을 제공하지 못한다[1,2]. 또한 실시간 데이터베이스 시스템은 군사 응용 목적과 같은 특정한 응용 시스템에 맞춰 개발되었고, 일반 응용에는 적절하지 못한 기능들을 포함하고 있다. 즉, 최근의 빠른 성능을 요구하는 데이터베이스 응용들은 종료시한(deadline) 개념을 갖는다기 보다, 전체적으로 우수한 트랜잭션의 처리를 요구하기 때문에 순수 실시간 데이터베이스 시스템을 적용하는데 어려움이 많다[3]. 따라서, 범용 디스크 기반의 데이터베이스 기능은 그대로 유지하면서, 실시간 시스템과 동등한 트랜잭션 처리 성능

* 정 회 원 : 상지대학교 컴퓨터정보공학부 교수
leekw@sangji.ac.kr(제 1저자)
[2004/01/28 투고 - 2004/02/19 심사 - 2004/09/09 심사 완료]
☆ 본 논문은 2004년도 상지대학교 교내 연구비 지원에 의한 것임.

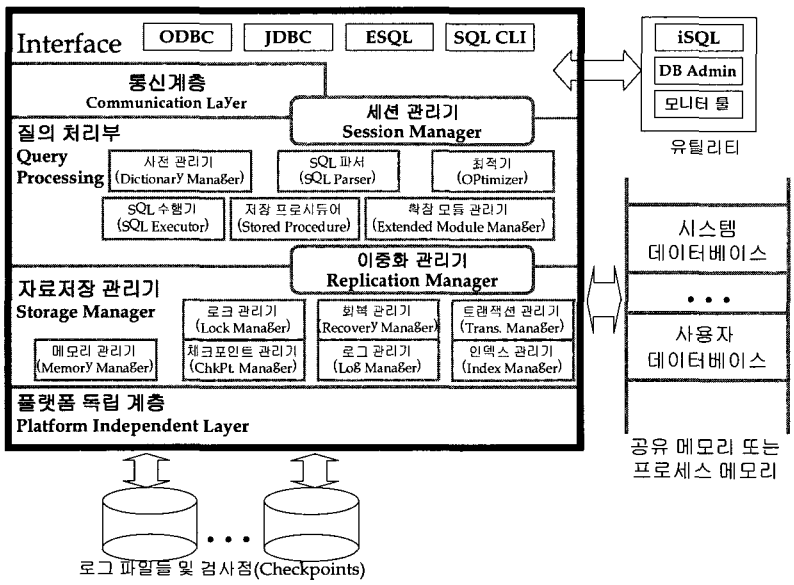
을 제공하고 기존 응용을 쉽게 이식하기 위해서, 디스크 기반 데이터베이스 시스템의 주저장장치인 디스크를 메인 메모리로 대체한 주기억장치 DBMS의 수요가 증가하고 있다.

ALTIBASE™ 시스템은 메인 메모리를 주 저장장치로 사용하는 관계형 데이터베이스 시스템으로서 범용적 데이터베이스 응용 뿐만 아니라 특정한 실시간 응용 목적에도 부합할 수 있는 주기억장치 DBMS이다. 또한 최근 데이터베이스 응용들의 요구사항으로 부각되고 있는 것은 어떠한 상황에서도 서비스가 중단되지 않아야 한다는 무정지 데이터베이스 서비스, 즉 데이터베이스의 고가용성 기능이다. 데이터베이스의 고가용성 기능에 대한 요구를 만족시키기 위해서는 트랜잭션 처리에 대한 성능이 저하될 수 밖에 없는 상반관계(trade-off)에 있는 요구사항이다. 따라서 이러한 고가용성 기능과 효율적인 트랜잭션 처리 기능을 모두 만족시킬 수 있는 주기억장치 데이터베이스 시스템의 설계가 필요하다. 따라서 본 논문에서는 주기억장치 DBMS인 ALTIBASE™의 시스템 구조를 먼저 설명한 후, 고가용성을 제공하기 위한 ALTIBASE™의 이중화 기법과 그 구현 내용을 기술한다. 기존 분산 시스템이나 데이터베이스 시스템에서 사용하는 이중화 기법들은 트랜잭션의 성능보다는 안정성을 위주로 제안된 방법들이므로, 주기억상주 DBMS에서 그대로 적용하기에는 많은 문제점들을 갖고 있다. 본 논문에서 이러한 주기억상주 DBMS용 이중화 기법을 설명하고, 그 구현내용을 기술한다.

본 논문의 구성은 다음과 같다. 2장에서 ALTIBASE™ 시스템의 구조 및 구성요소에 대하여 기술하고 주요 기능에 대하여 설명한다. 3장에서 고가용성을 제공하기 위한 ALTIBASE™의 이중화 관리 기법과 설계 방안을 제안한다. 제안한 이중화 관리 기법을 적용하였을 때, 기존 시스템과의 성능 및 독립 서버와의 성능 평가 결과를 4장에서 설명한다. 끝으로 5장에서 본 논문의 결론을 맺는다.

2. ALTIBASE™ 구조적 특징 및 시스템 모형

ALTIBASE™ 시스템은 크게 사용자 및 프로그래머 인터페이스 부분과 질의처리부, 트랜잭션



〈그림 1〉 ALTIBASE™ 시스템 구성도

처리를 위한 자료저장 관리기로 구분할 수 있다. 그림 1은 ALTIBASE™ 시스템의 시스템 구조를 도식화하여 나타내고 있다. 그림 1의 가장 상위 계층에 위치하는 인터페이스 부분은 다양한 산업 표준 응용 프로그래밍 인터페이스를 제공하고 있다. X/OPEN CLI 표준 사양을 따른 SQL CLI 인터페이스, ODBC 및 JDBC 인터페이스, 내재된 SQL 프로그래밍을 지원하는 ESQL 인터페이스, 서버 내장형 SQL 기반 인터페이스를 제공한다.

질의 처리부는 데이터베이스 표준언어인 SQL 2를 완벽하게 지원하며 데이터 사전 관리기, 최적기, SQL 수행기, 저장 프로시듀어, 확장 모듈 관리기 등의 세부 구성요소를 갖는다. 기존 주기억장치 데이터베이스 시스템들이 SQL2의 일부분만을 지원하여 응용 프로그램의 제작에 어려움을 주는 반면 ALTIBASE™ 시스템의 질의 처리부는 SQL 2를 지원하는 질의 최적기, 중첩 반복 죠인, 해쉬 및 정렬을 이용한 죠인 연산 지원, 정렬-합병(sort-merge) 죠인 연산을 완벽하게 지원하는 질의 처리기를 포함하고 있다.

ALTIBASE™ 시스템의 자료저장 관리기는 데이터베이스 서버 프로세스를 구성하는 핵심 모듈로서 트랜잭션 처리를 위한 로크 관리기, 회복 관리기, 로그 관리기 등의 여러 세부 관리기들에 의해 구성된다. ALTIBASE™ 자료저장 관리기의 구성요소 중 로크 관리기는 고성능 트랜잭션 처리를 위한 개선된 다중 병행수행 제어 방법을 제공한다. 기존 다중버젼 병행수행 제어방법과는 달리 데이터의 로킹 단위(granularity)를 세분화하여, 즉, 테이블 단위와 레코드 단위로 구분하여 다중 버전 병행수행 제어 방법[4,5]을 사용한다.

ALTIBASE™의 회복 관리기는 트랜잭션의 ACID 속성을 완벽하게 지원하므로, 비록 주기억장치 DBMS라 할지라도 트랜잭션의 영속성(durability)을 위해 디스크와의 동기화 작업을 한다[7,8]. 따라서 로크 레코드를 위한 디스크 동기화 작업은 주기억장치 DBMS의 성능을 저하하는 주요 요소중 하나이다. ALTIBASE™ 시스템의 회복 관리기는 로그 레코드에 따른 성능 저하를 없애기 위해 메모리 맵드 파일이나 메모리 버퍼를 로그 버퍼로서 선택적으로 사용할 수 있도록 한다. 또한 응용의 특징에 따라 트랜잭션의 영속성보다 성능을 더 중요시하는 경우, 로그 작업에 의한 오버헤드를 줄일 수 있도록 세 가지 로깅 레벨을 지원하여 디스크 동기화 수준을 제어할 수 있다.

본 논문에서 집중적으로 언급할 데이터베이스 이중화는 고가용성을 목적으로 물리적으로 분리되어 있는 여러 데이터베이스에 변경 내용을 전달하여 동일한 데이터베이스 상태를 유지하는 것을 목적으로 한다. ALTIBASE™ 시스템의 이중화는 점대점(point-to-point) 방식을 기본적으로 사용하며, 변경 로그를 기반으로 하는 이중화 기법을 적용한다. 원격 서버에서는 점대점 방식으로 전송 받은 변경로그를 분석하여 실행 계획(execution plan)으로 재작성하여 수행하는 방식의 이중화 기능을 구현하고 있으며, 또한 N-way 이중화을 제공하여 네트워크 구조의 이중화 위상을 제공한다.

3. 고가용성을 위한 ALTIBASE™ 이중화 관리 기법

3.1 데이터베이스 이중화 기능 분류 및 특징

실시간 데이터베이스 응용 시스템 분야나 임무결정적(mission critical)인 응용 시스템을 다루는 산업 분야에서는 데이터의 성능이나 일관성보다 시스템의 가용성을 더욱 중요시 하는 경우가 많다[11]. 이와 같은 고가용성을 얻기 위하여 데이터베이스 이중화(database replication) 기법이 지속적으로 연구되어 왔다. 데이터베이스 이중화는 오류-극복(fail-over) 구조에서 가용성을 증가시키는 기본적인 방법으로 사용되어 왔으며, 또한 원격지 서버의 접속 필요성을 줄임으로 인해 성능을 향상시킬 수 있는 방법으로도 제공되어 왔다.

그러나 실시간 주기억상주 데이터베이스 시스템의 주 목적은 가용성 보다 예측가능한 응답시간과 성능이라 할 수 있다. 따라서, 가용성과 안정성을 위해서 부가적인 디스크 요청 작업이 증가할 수록 주기억상주 데이터베이스 시스템의 본 목적을 위반할 수 있게된다. 따라서 두가지 목적, 즉, 가용성과 빠른 성능을 모두 만족하기 위한 이중화 기법을 시스템에 구현해야만 한다.

전형적인 분산 데이터베이스 환경에서 데이터베이스 이중화 기법은 데이터 변경 내용을 전달하는 방법에 의하여 "lazy"와 "eager" 기법으로 분류되며, 변경의 주체 즉, 데이터의 소유자에 따라 그룹 방식 또는 마스터 방식으로 분류된다[9, 10]. "eager" 기법은 한 객체에 대한 변경 내용이 그 트랜잭션의 일부로 인식되어 수행되어 진다. 즉, 트랜잭션 수행 중에 발생한 변경은, 발생 즉시 모든 이중화 서버로 전달되어 연쇄적으로 변경내용이 반영되어야 하는 기법이다. 반면 "lazy" 기법은 트랜잭션의 수행이 완전히 완료된 후에, 그 변경 사실에 대한 새로운 트랜잭션을 작성하여 각 노드에게 전달하는 기법으로 각 노드마다 또 다른 새로운 트랜잭션이 수행되어 지는 것으로 간주된다. 표 1은 변경 내용의 전달방식에 따른 두 기법의 비교를 보이고 있다.

또한, 객체의 소유권 및 변경의 주체에 따라서 그룹 기법과 마스터 기법으로 분류할 수 있다. 즉, 객체를 변경을 할 수 있는 노드가 결정되어 있어서, 그 노드에서만 객체를 변경할 수 있고, 그 변경 내용을 다른 모든 노드에게 전달하는 마스터 기법의 방식과, 임의의 노드가 임의의 객체를 변경할 수 있으며, 그 변경사실을 모든 노드에게 전달하는 그룹 기법 방식이 있다. 따라서, 그룹 기법에서는 아무 노드나 객체를 변경하고, 변경 사실에 대한 갱신 트랜잭션을 발생할 수 있으나, 마스터 기법에서는 주 사본(primary copy)을 소유한 노드에서만 그 사본에 대한 변경이 가능하고, 그 변경 내용을 전달할 수 있다. 주 사본이 아닌 복사본을 소유하고 있는 모든 원격 서버들은 그 사본 객체에 대해서 판독연산만 가능하다. 이러한 객체 변경 소유권에 따른 기법의 비교는 표 2와 같다.

변경 내용 전달 방식과 객체 변경 소유권의 기준에 따라 데이터베이스 이중화 기법들을 표 3과 같이 분류할 수 있다.

위와 같은 분류에 따라 ALTIBASE™ 시스템은 "lazy" 기법과 마스터 기법을 사용하는 이중화 기능을 제공하며 다음과 같은 특징을 제공한다. 첫째, 기존 디스크 기반 시스템에서 제공되는 수준의 가용성과 확장성을 제공한다. 시스템 오류

〈표 1〉 변경 내용 전달 기법에 따른 비교

	전역적 완료 (global commit)	일관성[6] (전역적 직렬성)	응답 시간	상호 조화 (reconciliation)	데드로크 비 율
Eager	2단계 완료, 3단계 완료	엄격함	늦음	불필요	높음
Lazy	지역적 완료	약화됨	빠름	필요	낮음

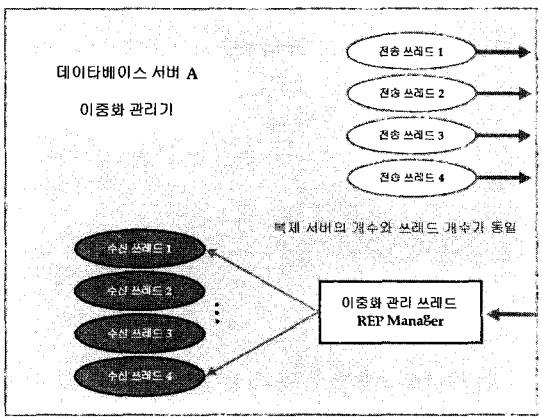
〈표 2〉 객체 변경 소유권에 따른 기법 비교

	변경 가능 노드	전역 로크 처리	응용 유연성	상호 조화 (충돌 해결)	데드로크 비 율
그룹	임의 노드에서 변경 가능	필요	높음	필요, 타임스탬프 기반	높음
마스터	마스터만 변경 가능	불필요	낮음	부분적 필요	낮음

발생시 이중화 기능을 갖는 제 2의 서버가 즉시 서비스가 가능하도록 이중화 작업을 지속적으로 수행하며, 이러한 이중화 기능을 여러 시스템에 이중화할 수 있는 수평적 확장성 또한 지원한다. 둘째, 이중화 작업시 발생할 수 있는 데이터 불일치성을 시스템의 부하없이 무결성 데이터로 유지하게 한다. 즉, 하나의 데이터가 동시에 여러 서버 내에 존재하므로, 다중 접근시에 발생하는 데이터 충돌 문제가 발생하는데, ALTIBASE™ 시스템은 데이터 충돌에 대해, 불일치 레코드를 탐지하는 비교 정책, 일치(sync) 정책등을 이용한 추적 감시(audit) 기능을 이용하여 이러한 데이터 불일치를 해결할 수 있도록 한다. 셋째, 이중화 작업을 하지 않는 독립서버 대비 우수한 성능을 제공한다. ALTIBASE™ 시스템은 이중화를 수행하는 오버헤드가 작아 독립 서버 시스템으로 수행할 때와의 성능 비교시 95% 이상의 성능을 유지할 수 있다. 넷째, 지역성 및 부하 분산 기능을 제공한다. 원격 서버의 성능이 지역 서버의 성능에 영향을 미치지 않도록 데이터 이중화 작업을 수행하도록 하였으며, 다중 서버 운영환경에서 서비스 하는 트랜잭션들을 응용 프로그램 수준에서 두 그룹 이상으로 나누어, 각 그룹의 트랜잭션들을 해당 서버에서 수행되도록 한 후, 각 서버의 변경내용을 서로 반영하는 부하 분산 기능을 제공하여 서버의 부하를 분산시킬 수 있도록 하였다.

3.2 ALTIBASE™ 이중화 관리기 구조 및 통신 모델

이중화 관리 쓰레드는 이중화 서버들간의 통신을 담당하는 이중화 관리 쓰레드(REP_Manager)와 이중화 데이터를 원격 서버에 보내는 전송 쓰레드(Rep_Sender), 원격 서버의 이중화 데이터를 전달 받아 자신의 데이터베이스에 반영하는 수신 쓰레드(Rep_Receiver)로 구성된다. 전송 쓰레드는 타 서버의 이중화 관리 쓰레드와의 연결을 통해 타 서버의 수신 쓰레드와 통신을 하게 되며, 수신 쓰레드는 자신의 이중화 관리 쓰레드와 접속하여 타 서버의 전송 쓰레드와 통신을 하게 된다. ALTIBASE™ 시스템의 이중화 관리기는 그림 2와 같은 쓰레드 구조로 구성된다.



〈그림 2〉 한 노드에서의 이중화 관리 쓰레드 구조

〈표 3〉 이중화 기법들의 분류

변경 시점 / 변경 장소	Eager	Lazy
마스터 기법	- ALTIBASETM의 초기 방안 - INGRES - ORACLE Synchronous Replication	- ALTIBASETM 이중화 기법 - SYBASE 이중화 서버 - IBM Data Propagator - ORACLE Placement 전략
그룹 기법	- ROWA/ROWAA(쿼럼 방식) (Read-one/Write-all, Read-one/ Write-all-available) - ORACLE Synchronous Replication	- Timesten - ORACLE Symmetric Replication (asynchronous)

위와 같은 노드 구조를 기반으로 3개의 이중화 서버들로 구성되는 이중화 시스템이 그림 3에 나타나 있다.

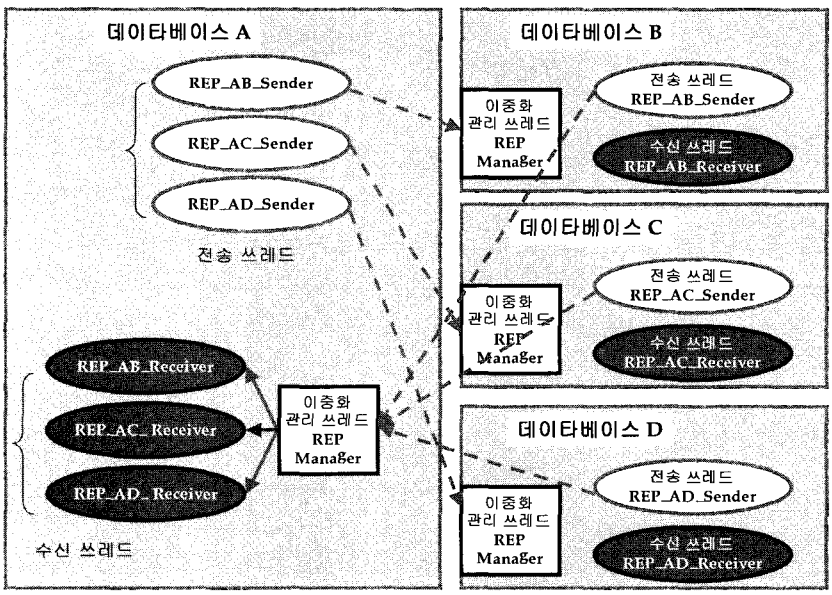
ALTIBASE™ 시스템은 한 지역 서버가 최대 16개의 원격 서버에 대하여 이중화를 수행할 수 있다. 그림 3에 나타난 바와 같이 지역 서버 A에는 이중화 데이터를 전송하기 위한 전송 쓰레드(REP_A*_Sender)가 생성되고, 원격 서버(지역 서버 A 입장에서 서버 B, C, D)에는 이에 대응하는 수신 쓰레드(REP_A*_Receiver)가 생성된다. 각각의 서버 B, C, D 입장에서는 서버 A가 원격 서버로 간주된다. 전송 쓰레드와 수신 쓰레드들 간의 동기화는 이중화 관리기에 의해 수행되며, 동기화가 수행되면, 각 송, 수신 쓰레드간의 직접 전송이 이루어 진다. 이중화 관리기는 각 데이터베이스 마다 하나씩 존재하게 되며, 데이터베이스 서버 구동시 자동으로 실행되게 된다.

ALTIBASE™ 시스템은 이중화에 참여하는 노드들의 역할에 따라 여러 가지 이중화 통신 모델을 지원한다. 이중화 기능에 참여하는 각 노드들은 다음 표 4와 같은 세 가지 서버 유형으로 분류할 수 있다.

〈표 4〉 이중화 기능을 위한 통신 모델의 서버 유형

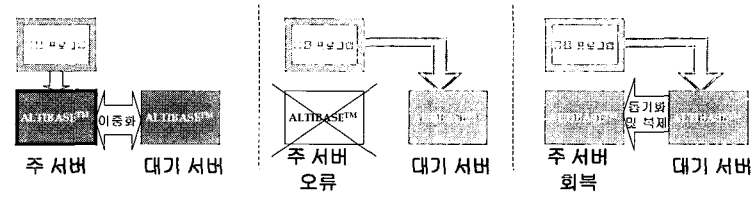
서버 유형	기능 및 역할
주 서버 (primary server)	다른 원격 서버에 데이터베이스를 이중화만 하고 원격 서버로부터 데이터 이중화를 위한 변경 트랜잭션 요구를 받지 않는 서버
대기 서버 (standby server)	하나 또는 그 이상의 주 서버로부터 데이터 이중화를 위한 변경 트랜잭션 요청을 받아서 자신의 데이터베이스에 반영하는 서버로 정상 운영 모드에서는 데이터베이스 서비스를 하지 않는 서버이다.
활동 서버 (active server)	주 서버와 대기 서버의 역할을 동시에 하는 서버로서, 데이터 이중화를 위한 변경 트랜잭션 요구를 받아 자신의 데이터베이스에 반영하면서 동시에 자신의 데이터 변경 사항을 이중화하기 위한 변경 트랜잭션을 원격 서버로 전송하는 서버

위에 기술한 표 4의 서버 구성에 따라 ALTIBASE™ 시스템의 이중화를 위한 통신 모델을

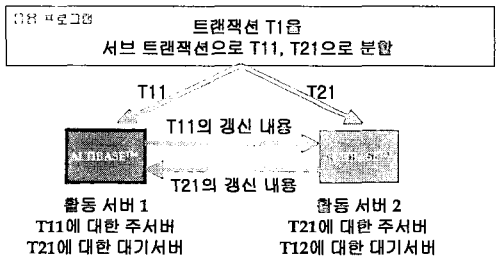


〈그림 3〉 4개의 노드로 구성되는 이중화 관리 쓰레드

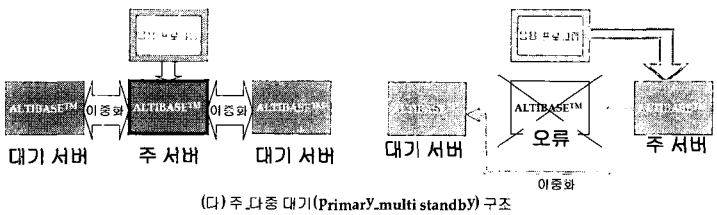
그림 4와 같이 네 가지로 분류할 수 있다. 주-대기 서버 구조는 데이터베이스 이중화 뿐만 아니라 분산 시스템의 전형적인 이중화 모형으로서, 정상적인 운영 모드에서 주 서버는 일반적인 데이터베이스 서비스를 제공하는 것처럼 보이나, 실제로 자신의 변경 내용을 트랜잭션화 하여 대기 서버로 전달하고 있으며, 대기 서버는 데이터베이스 서비스는 하지 않은 채, 변경 내용을 자신의 데이터베이스에 반영하게 된다. 주 서버에 오류가 발생하면, 주 서버의 서비스를 받던 응용들을 대기 서버로 옮겨 대기 서버에서 새로운 서비스를 시작하게 된다. 주 서버의 오류가 복구 되면 그



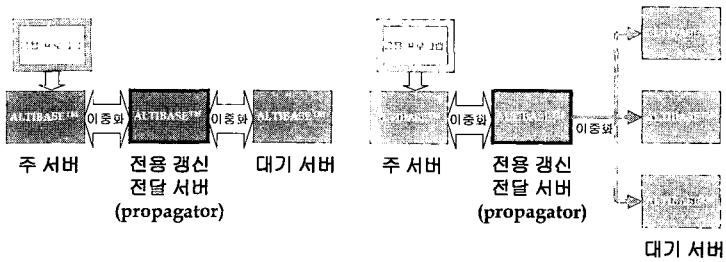
(가) 주-대기(Primary-standby) 구조



(나) 활동-활동(active-active) 구조



(다) 주-다중 대기(Primary-multi standby) 구조



(라) 전용 갱신 전달 서버(ProPagator)를 이용한 다중 대기(Primary-multi standby) 구조

〈그림 4〉 ALTIBASE™ 시스템의 이중화 통신 모델

동안의 대기 서버의 변경 트랜잭션을 주 서버에 보내 이중화 작업을 계속 수행한다. 그 후, 양 서버의 역할을 바꾸어 계속 수행하거나, 주 서버와 대기 서버의 역할을 바꾸어 수행하기도 한다.

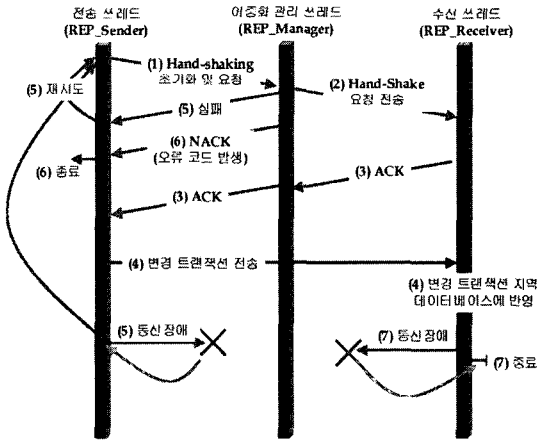
활동-활동 서버 이중화 통신 모델은 응용의 업무를 분리하여 처리하거나 부하 분산을 위해서 주로 사용된다. 변경된 데이터 갱신 내용을 서로 상대편 서버에 이중화하는 작업을 수행하고, 응용에서 발생하는 트랜잭션을 두 그룹으로 분리하여 양 서버에서 수행하게 한 후, 서로의 변경 트랜잭션을 교환하여, 두 서버 그룹의 데이터 변경 내용을 일치시켜 상호 이중화 작업을 하도록 한다. 즉, 응용 수준에서 트랜잭션 T1을 T11과 T21으로 분할 시킨 후, 각 서브 트랜잭션을 그림 4의 (나) 모델에 적용하여, T11을 활동 서버 1에게 T21을 활동 서버 2에게 전송시켜 각각의 서버에서 서브 트랜잭션을 수행시킨 후, 결과를 조합하여 최종결과를 얻을 수 있게 된다. 이 때, 트랜잭션 T11에 대해서는 활동 서버 1이 주 서버, 활동 서버 2가 대기 서버 역할을 하게 되면, 반대로 트랜잭션 T21에 대해서는 활동 서버 2가 주 서버, 활동 서버 1이 대기 서버 역할을 하게 된다. 데이터베이스 이중화 기능을 위해서 활동 서버 1에서 수행된 T11의 변경 내용은 활동 서버 2로 전송되고, 마찬가지로 활동 서버 2에서 수행된 T21의 변경 트랜잭션 내용은 활동 서버 1로 전송되어, 상호 이중화 기능을 갖도록 수행하는 한편, 두 서버를 이용한 부하 분산 기능을 얻도록 한다. 이 통신 모델은 트랜잭션의 부하 분산을 통한 성능 향상을 얻을 수 있는 반면, 이중화를 위한 변경 트랜잭션 수행에 따른 오버헤드를 갖는다.

주-다중 대기 통신 모델은 성능에 민감하고 높은 가용성을 요구하는 응용에 적합한 모델로서, 하나의 주 서버와 두 개 이상의 대기 서버로 구성된다. 이 구조는 주-대기 서버와 기본적으로 같은 구성을 갖지만, 대기 서버의 수가 많다는 점에서 시스템의 가용성을 더욱 높일 수 있다. 이 구조는 높은 고가용성을 얻을 수 있어, 치명적 손실을 입을 수 있는 응용 시스템에 적합한 모델이지만, 비용이 많이 드는 단점을 갖는다.

다중 대기 서버에 변경 트랜잭션을 전달하기 위해 소요되는 주 서버의 오버헤드를 절감시키기 위해, 그림 4의 (라)와 같은 전용 변경 전달 서버(propagator)를 운영하여 주-다중 대기 서버 모델을 운영할 수 있다. 변경 내용을 전달하기 위한 전담 서버는 대기 서버의 개수에 따라 자유롭게 계층적 구조를 설정할 수 있다.

3.3 이중화 프로토콜 및 알고리즘

이중화 기능의 수행은 이중화 관리기의 전송 쓰레드(REP_Sender), 이중화 관리 쓰레드(REP_Manager), 수신 쓰레드(REP_receiver)의 상호 협조 및 동기화 작업으로 수행된다. 이중화를 위한 이중화 프로토콜을 도식화 하면 그림 5와 같다.



〈그림 5〉 이중화 프로토콜

사용자가 이중화 작업을 시작하면, 전송 쓰레드가 생성되고, 전송 쓰레드는 원격 서버의 이중화 관리 쓰레드에게 핸드 쉐이크 연결을 요구하게 된다(그림 5의 (1)). 이중화 관리 쓰레드는 수신 쓰레드를 생성시키고, 수신 쓰레드에게 핸드 쉐이크 연결 상대를 알린다(그림 5의 (2)). 수신 쓰레드가 전송 요구를 성공적으로 전달 받게 되

면, 이중화 관리 쓰레드에 ACK 메시지를 전송하고, 차례로 이중화 관리 쓰레드는 전송 쓰레드에 성공 메시지를 전달한다(그림 5의 (3)). 전송 쓰레드로 ACK 메시지가 완전히 전송되면, 전송 쓰레드와 수신 쓰레드는 직접 연결을 통해 변경 트랜잭션(XLOG)을 전송하게 되고, 수신 쓰레드는 지역 데이터베이스에 트랜잭션의 내용을 반영한다(그림 5의 (4)). 변경 트랜잭션은 로그를 기반으로 이중화 시작점(LSN)부터 현재까지(XLSN)의 변경 로그를 기반으로 트랜잭션화 하여 전송하게 된다.

반면에, 전송 쓰레드가 통신 장애나 이중화 관리 쓰레드의 수신 오류 등의 이유로 인해 연결이 제대로 성립되지 않는 경우 그림 5의 (5)와 같이 연결 재시도를 수행한다. 그러나 이중화 관리 쓰레드와의 연결은 성공하였으나, 이중화 스키마의 불일치와 같은 논리적 오류 및 사용자 오류에 의해 NACK 메시지를 받게 되면 그림 5의 (6)과 같이 전송 쓰레드는 종료된다. 또한 수신 쓰레드는 이중화 관리 프로토콜 수행 중 어떤 시점에서라도 통신 장애가 발생하는 경우 즉시 종료하게 된다(그림 5의 (7)). 이 프로토콜상의 이중화 송신 쓰레드와 수신 쓰레드의 간략한 알고리즘은 다음과 같다.

○ 송신 쓰레드 알고리즘

```
[1 단계] 수신 쓰레드와의 연결
1-1 수신 쓰레드와 연결 시도 ;
1-2 if ( connected )
        이중화 시작점 설정
        2 단계 시작 ;
1-3 else if ( 사용자 또는 시스템에 의한 이중
            화 종료 요구)
        이중화 종료 ;
1-4 else
        1단계 처음부터 재수행(연결 재시도) ;
[2 단계] 변경 로그 작성
2-1 if ( 사용자 또는 시스템에 의한 이중화 종
        료 요구)
        이중화 종료 ;
2-2 if ( 현재 로그(LSN) == 변경 로그(XLSN) )
        이중화 종료 ;
2-3 else
        XLSN 위치에서 로그 획득 ;
        switch(로그의 유형)
        case '트랜잭션 시작 로그' :
            이중화 관리 테이블에 현재 트
            랜잭션 등록 ;
            2단계 처음부터 재시작 ;
        case '트랜잭션 종료 로그' :
            if ( 전송 버퍼 == FULL)
                    3단계 수행 ;
            종료 로그를 전송 버퍼에 저장 ;
            이중화 관리 테이블에서 현재
            트랜잭션 삭제 ;
            2단계 처음부터 재시작
        case '데이타 변경 로그' :
            if ( 전송 버퍼 == FULL)
                    3단계 수행 ;
            변경 로그를 전송 버퍼에 저장 ;

2-4 XLSN 증가

[3 단계] 전송 버퍼 송신
3-1 수신 쓰레드 측으로 전송 버퍼 전송
3-2 2 단계부터 재수행.
```

○ 수신 쓰레드 알고리즘

```
[1 단계] 수신 및 이중화 테이블에 반영
1-1 if ( 송신 쓰레드와 연결)
        전송 버퍼에 수신된 XLOG를 읽음 ;
1-2 switch( XLOG의 로그 유형)
        case '트랜잭션 시작' :
            이중화 트랜잭션 관리 테이블에
            등록 ;
            1-1부터 재수행 ;
        case '트랜잭션 종료' :
            이중화 트랜잭션 관리 테이블에
```

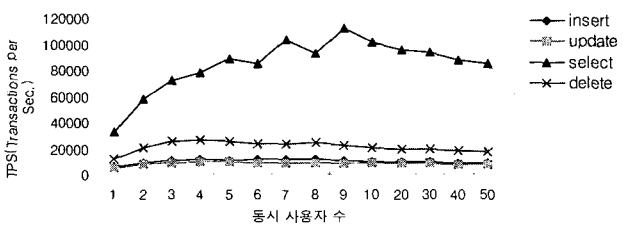
서 삭제 ;
1-1부터 재수행 ;
case '갱신 로그' :
데이터베이스 XLOG 반영;
1-1부터 재수행 ;

4. 이중화 기능의 성능 평가

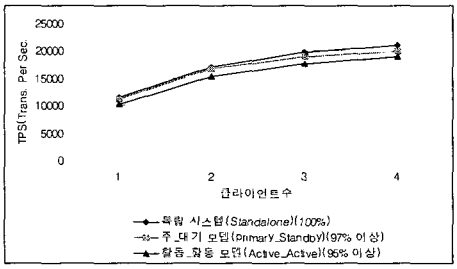
본 절에서는 ALTIBASE™ 시스템의 트랜잭션 수행에 대한 성능평가를 보인다. 특히 여러 종류의 시스템 부하에 따른 TPS(Transaction per Second) 값을 집중적으로 평가하여 시간제약적인 응용 분야에 적합한 시스템임을 보인다. 본 절에서 수행한 모든 실험은 400MHz CPU 4개와 4G 바이트 메모리를 보유한 "Sun Enterprise 3500" 플랫폼과 "Solaris 2.5.8" 운영체제 하에서 수행하였다. 또한 실험 트랜잭션은 ALTIBASE™ 시스템에서 제공하는 저장 프로시듀어(native stored procedure) 인터페이스를 사용하여 구현하였다. 따라서 트랜잭션의 데이터베이스 요구는 모두 질의 처리기를 거쳐 최적 수행 계획에 따라 수행되며, 네트워크 지연에 따른 간섭은 실험에 평가되지 않는다. 실험에 사용된 트랜잭션은 모두 4종류, 검색, 삽입, 변경 그리고 삭제 트랜잭션이며, 대상 테이블은 "number", "real", "varchar"등의 여러 가지 속성들로 구성되는 총 20개의 속성을 갖는 단일 테이블이다. 동시 사용자 수는 실험에 따라 단일 사용자에서 50명의 사용자로 변화를 주었으며, 레코드의 개수는 실험에 따라 총 10,000개에서 500,000개의 레코드로 구성하였다. 검색 트랜잭션의 경우 모든 속성을 검색하도록 수행되었으며, 삽입 및 변경 트랜잭션들도 모든 속성들에 대해 삽입 및 변경 작업을 수행하도록 구성하였다.

그림 6은 동시 사용자 수의 변화에 따른 TPS 결과의 변화를 측정한 실험 결과이다. 동시 사용자 수가 증가 할수록, 처리해야 할 트랜잭션의 수가 증가하게 되어 TPS의 수가 증가함을 알 수 있다. CPU 처리능력 임계치에 근접해 가면서 초당 처리할 수 있는 트랜잭션의 수가 점점 증가하게 됨을 알 수 있으나 동시 사용자 수가 10명 이상이 되는 지점에서부터 CPU 처리능력이 부족하여 TPS 수치가 일정 수준을 유지하거나 약간 감소함을 알 수 있다. 검색 트랜잭션을 제외한 갱신, 삽입, 삭제 트랜잭션을 혼합하여 수행시킨 이 실험에서 동시 사용자수에 크게 변화없이 일정한 트랜잭션 처리율을 보임을 알 수 있으며, 이 측정 결과는 상용 주기억장치 DBMS에 비해 우월한 수치임을 확인할 수 있다.

그림 7은 ALTIBASE™ 시스템의 이중화 구조에 대한 트랜잭션 처리율을 평가한 것이다. 독립 서버와 주-대기 서버 이중화 모델의 트랜잭션 처리율을 비교하면, 이중화 기능을 지원하는 경우의 트랜잭션 처리율이 다소 감소하고 있음을 알 수 있으나, 이중화 기능을 사용하는 장점에 비해 무시할 만한 성능 차이를 보인다. 활동-활동 이중화 구조에서도 다소 감소된 트랜잭션 처리율을 보이지만, 독립 서버가 제공하는 트랜잭션 처리율의



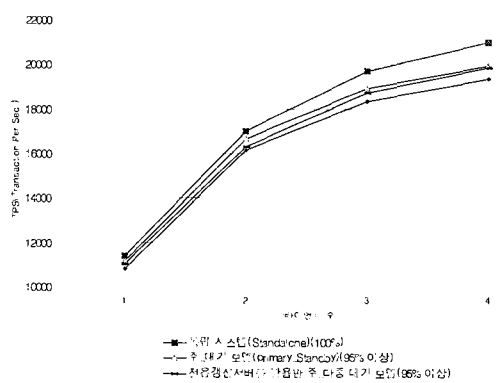
〈그림 6〉 동시 사용자 수 대 트랜잭션 처리율



〈그림 7〉 이중화 구조에 따른 트랜잭션 처리율

95%이상의 성능을 보임을 알 수 있다. 따라서, 어떠한 이중화 모델을 사용하더라도 독립 시스템 성능의 5% 이하의 오버헤드로서 이중화 기능을 수행할 수 있음을 알 수 있다.

또한, ALTIBASE™ 시스템이 지원하는 이중화 모델중에서 안정성을 더욱 중요시 하는 주-다중 대기 모델의 경우에는 주 서버가 응용 서비스를 하면서, 동시에 다수의 대기 서버로 갱신내용을 전달해야 하므로 주 서버의 오버헤드가 커질 수 있다. 따라서 이러한 경우의 성능 차이를 독립 시스템과 주-대기 서버의 모델과 비교하여 그림 8에 표현하였다. 본 실험에서는 2개의 대기 서버를 운영하여 주 서버가 2개의 서버로 갱신내용을 전달하도록 하였다. 이 실험 결과에서 알 수 있듯이, 주-다중 대기 서버 모델은 독립 서버의 트랜잭션 처리율에 비해 8% 미만의 오버헤드를 가져 92% 이상의 성능을 보일 수 있음을 알 수 있다. 주-대기 서버 모델의 성능과 비교해 볼 때 약 97%이상의 성능이다. 그러나, 이러한 갱신 내용 전달에 대한 오버헤드를 제거하기 위해 전용 갱신 전달 서버(propagator)를 활용한 실험에서의 성능 측정 결과는 주-대기 서버 모델과 유사한 성능을 보였으며, 이는 독립 시스템의 성능과 비교할 때 95%이상의 트랜잭션 처리율을 보임을 알 수 있다.



〈그림 8〉 주-다중 대기 서버 및 전용 갱신 전달 서버 모델의 트랜잭션 처리율

결국, 본 실험을 통해 측정된 결과는 다른 상용 주기억 상주 DBMS 들에 비해 우세한 실험 결과임을 알 수 있으며, 데이터베이스 이중화 기능 또한 심각한 오버헤드 없이 지원가능 함을 알 수 있다.

5. 결 론

ALTIBASE™ 시스템은 메인 메모리를 주 저장장치로 사용하는 관계형 데이터베이스 시스템으로서 범용적 데이터베이스 응용 뿐만 아니라 특정한 실시간 응용 목적에도 부합할 수 있는 주기억장치 DBMS이다. 본 논문에서는 최근 데이터베이스 응용들의 요구사항으로 부각되고 있는 데이터베이스의 고가용성 기능의 지원과 실시간 데이터베이스 시스템과 유사한 트랜잭션 처리율을 동시에 보장하기 위하여 ALTIBASE™ 시스템의 구조와 이중화 기능에 대하여 언급하였다. 특히, 이중화 기법의 분류를 통해 ALTIBASE™ 제시하는 이중화 기법과 타 기법과의 비교를 하였으며, 이중화 관리를 위한 프로세스 구조 및 이중화 통신 모델, 프로토콜 등에 대하여 기술하였다. 현재 ALTIBASE™ 시스템은 버전 3.0을 상용 제품으로 출시하였으며, 성능 및 안정성, 고가용성을 요구하는 응용 시스템들에 범용적으로 활용되고 있다.

그러나, 주기억장치 DBMS의 대용량 자료처리에 대한 단점을 해결하기 위한 방안이 아직 제시되지 않아, 범용 DBMS로서 자료 처리 용량의 한계를 극복하지 못하고 있다. 본 ALTIBASE™ 시스템의 향후 연구 및 개발 방향은 디스크 기반 DBMS와 주기억장치 DBMS를 혼합하여 사용할 수 있는 비등속 다중 저장 장치 데이터베이스 시스템을 개발하는 것이다.

참 고 문 헌

[1] P. Bohannon, J. Parker, R. Rastogi, S.

Seshadri, A. Silberschatz, and S. Sudarshan, "Distributed Multi-Level Recovery in Main-Memory Databases", Proc. of the International Conference on Parallel and Distributed Information Systems, 1996.

[2] H. Garcia-Molina and K. Salem, "Main Memory Database Systems : An Overview", IEEE Transactions on Knowledge and Data Engineering, 4(6), 1993.

[3] P. Bohannon, D. F. Lieuwen, R. Rastogi, A. Silberschatz, S. Seshadri, and S. Sudarshan, "The Architecture of the Dali Main-Memory Storage Manager", Multimedia Tools and Applications, 4(2), 1997.

[4] D. Agrawal and V. Krishnaswamy, "Using Multiversion Data for Non-Interfering Execution of Wirte-Only Transactions", Proc. of the ACM SIGMOD International Conference on Management of Data, 1991.

[5] P. M. Bober and M. J. Carey, "Multiversion Query Locking", Proc. of the 18th Conference on Very Large Database, 1992

[6] K. Ramamritham and P. K. Chrysanthis, "A Taxonomy of Correctness Criteria in Database Applications", VLDB Journal, 5(1), 1996.

[7] H. Garcia-Molina and K. Salem, "Main Memory Database Systems : An Overview", IEEE Transactions on Knowledge and Data Engineering, 4(6), 1993.

[8] H. V. Jagadish, A. Silberschatz, and S. Sudarshan, "Recovering Main Memory Lapses", Proc. of the 19th Conference on Very Large Databases, 1993.

[9] Bettina Kemme and Gustavo Alonso, "A New Approach to Developing and Implementing Eager Database Replication Protocols", ACM Transactions on Database Systems, 25(3), Sep., 2000.

[10] Jim Gray, Pat Helland, Patrick O'Neil, and Dennis Shasha, "The Dangers of Replication and a Solution", Proc. of the AMC SIGMOD International Conference on Mangement of Data, 1996

[11] M. Wiesmann, F. Pedone,A. Shiper, B. Kemme, G. Alonso, "Understanding Replication in Databases and Distributed Systems", Proc. of the 20th International Conference on Distributed Computing Systems, 2000.

[12] P. A. Bernstein, V. Hadzilacos, and N. Goodman. "Concurrency Control and Recovery in Database Systems", Addison-Wesley Publishing Company, 1987

◐ 저 자 소 개 ◑

이 규 웅(Lee, Kyu Woong)
1990년 한국외국어대학교 전자계산학과(이학사)
1992년 서강대학교 대학원 전자계산학과(공학석사)
1998년 서강대학교 대학원 전자계산학과 (공학박사)
2000년 한국전자통신연구원 인터넷서비스 연구부 선임 연구원
2000년~현재 상지대학교 컴퓨터정보공학부 조교수
관심분야 : 트랜잭션 처리, 자료저장 시스템, 분산 및 실시간 DB
E-mail : leekw@sangji.ac.kr