論文 96-21-7-17

클라이언트-서버 주기억 데이타베이스 환경에서의 트랜잭션 관리

正會員 조 성 제*, 김 경 창**, 김 기 룡**

Transaction Management in a Client-Server Main Memory DB Environment

Sung-Jae Cho*, Kyung-Chang Kim**, Ki-Ryong Kim** *Regular Members*

※본 연구는 1995년 한국 과학재단 핵심 전문 연구(과제 번호 951-0908-005-2) 연구과제 연구비에 의하여 연구되었음

요 약

클라이언트-서버 DBMS구조는 많은 사람들에 의해서 연구되고 있으나 고장회복과 동시성 제어에 관한 연구는 아직까지 미흡한 상태다. 기존의 회복 기법은 클라이언트에서 생성된 로그 레코드들과 해당 데이터 페이지들을 서버에 전송 함으로써 발생하는 문제점들에 대한 해결점이 필요하였다. 본 논문에서는 클라이언트가 서버에 수행 완료된 로그 레코드들만을 전송하여 기존의 회복 기법에서 발생하였던 문제점을 제거한다. 또한 서버는 수행 완료된 로그 레코드들만을 관리하여 서버 파손 발생시 분석, 재수행 동작만을 수행하는 간단한 회복 알고리즘을 제안한다. 클라이언트에서 철회 동작을 함으로써 시스템의 병렬성을 높이고 서버 파손시 전체 데이터베이스 회복에 소요되는 시간을 줄이기 위해서 페이지 단위의 회복 기법도 제안한다. 또한 기존의 동시성 제어기법은 같은 페이지를 동시 사용하지 못함으로써 불필요한 대기시간(waiting time)이 증가된다. 본 논문에서는, 기존의 기법과 달리, 같은 페이지 내에 다른 데이터를 접근하는 트랜잭션에게 페이지를 허용함으로써 불필요한 대기시간을 최소화 할 수 있도록 하여 시스템의 병렬성을 향상시켰다.

Ⅰ. 서 론

기존의 데이터베이스 관리 시스템에서는 트랜잭션 처리를 할 때 디스크 장치에 저장된 데이터를 접근하기 때문에 수행 속도가 늦고 처리 시간을 예측하기 어렵다. 현실적으로 수행 속도가 빠른 실시간 응용에 기존 데이터베이스 시스템을 이용하는 것은 많은 문제점을 안고 있다. 이를 극복하기 위하여 주기억 데이터베이스 환경 구축이 바람직할 수 있다. 결국 실시간 응용에 적용될 수 있는 데이터베이스 시스템에 대한 연구가 절실히 필요하게 된다.

*경주 대학교 컴퓨터공학과
Dept. of Computer Science Kyong-Ju Univ.
**홍익대학교 전자계산학과
Dept. of Computer Science Hong-Ik Univ.
論文番號:96148-0511
接受日字:1996年 5月 11日

한편, 강력한 워크 스테이션들과 서버들로 구성된 네트워크가 다양한 응용 프로그램들의 실행 환경이 됨에 따라 최근에는 상업적, 연구목적의 데이터베이스 시스템들도 이러한 환경 속에서 실행가능 하도록 구성되고 있다. 이런 환경 하에서 구성된 데이터베이스 시스템을 클라이언트-서버 데이터베이스 시스템이라고 한다. 클라이언트-서버 환경에서는 중앙집중식 데이터베이스 시스템과 달리 각각의 클라이언트들이 자신의 컴퓨터를 지니고, 그 컴퓨터에서 트랜잭션을 발생시켜 수행한다. 그러므로, 실시간 접근을 위해서 클라이언트 컴퓨터에 서버의 데이터를 가지고 온 후 클라이언트가 직접 접근하는 방법을 사용할 수 있다. 이러한 접근 방법은 클라이언트-서버 데이터베이스 시스템구조중 서버에 특정 데이터를 요청하는 방법에 해당한다[5]. 서버에 데이터를 요청하는 시스템은 그 성격에 따라 페이지 서버와 객체 서버로 나뉠 수 있다. 이 방법은 클라이언트 워크 스테이션의 기억 장치 이용률과 처리 율을 높여서 서버의 병목 현상을 최소화 할 수 있다는 장점이 있다. 또한 각각의 클라이언트가 기억 장치와 처리 능력을 지니므로 성능을 고려할 때 효율적이라 할 수 있다. 그러나 클라이언트가 서버로 특정 데이터를 요청하는 구조는 전통적인 회복 기법과 비교해 볼때 몇가지 문제점을 가지고 있는데, 그것은 클라이언트-서버의 구조적 문제로 인한 것이다. 즉, 데이터베이스의 갱신 동작이 클라이언트에서 이루어지고, 서버는 갱신에 대한 로그와 갱신 발생전 상태의 데이터베이스를 보유한다. 또한 클라이언트는 자신의 버퍼를 갖는다. 그러므로 기존의 중앙집중식 데이터베이스 시스템에서 사용하는 회복 방법으로는 효율적 회복 동작과 동시성 제어가 이루어지기 어렵다. 본 논문에서는 주기억 데이터베이스를 기반으로 하는 클라이언트-서버 환경에서의 효율적인 회복 동작과 동시성 제어 기법을 제안한다. 본 논문의 2장에서는 회복 기법과 동시성 제어 기법의 관련 연구를, 3장에서는 본 논문이 제안하는 클라이언트-서버 주기억 데이터베이스에서의 트랜잭션관리 시스템에 대해서 기술하고 4, 5장에서는 회복 동작과 동시성 제어에 대한 내용을 언급한다. 6장에서는 기존방법과 비교분석하고, 마지막 장에서는 앞으로 이루어져야 할 연구방향을 제시한다.

Ⅱ. 관련 연구

본 장에서는 클라이언트-서버 환경에서 기존의 회복 기법과 동시성 제어 기법에 대해서 알아본다.

2.1 ARIES 고장 회복 기법

ARIES는 단순하고 강력하기 때문에 중앙 집중식 데이터베이스 시스템의 고장 회복 기법으로 많이 사용하고 있다[10]. 이러한 장점 때문에 클라이언트-서버 데이터베이스 시스템의 고장 회복 기법을 설계하는데 기본 모델로 사용하였다. ARIES 고장 회복 알고리즘을 사용하는 시스템은 파손 발생 후 세 가지의 분석 단계, 재 수행 단계, 철회 단계를 수행한다. 분석 단계에서는 최근의 검사점 로그 레코드부터 시스템 파손 전 로그 레코드까지 차례로 읽어 트랜잭션 진행 과정을 분석한다. 분석 단계에서 얻어진 결과를 이용하여 재 수행 단계, 철회 단계를 수행한다.

2.2 ESM-CS 고장 회복 기법

Exodus 저장 관리자(Client-Server Exodus Storage Manager System : ESM-CS)는 최근에 Winsconsin 대학에서 개발한 시스템으로 클라이언트-서버 환경에서의 데이터베이스 시스템으로는 대표적 시스템이며 다른 많은 연구에도 인용되었다[5]. 이 시스템은 각각의 클라이언트가 독자적으로 수행되는 하나의 프로세서로서 자신의 메모리 캐쉬와 락 캐쉬를 가지고 있으며 한번에 하나의 트랜잭션을 처리한다. 클라이언트에서의 Write-Ahead-Log(이하 WAL라고한다) 규약의 구현을 위해서 로그 일련 번호(Log Sequence Number : LSN)을 사용한다. LSN을 사용시 클라이언트-서버 사이에 많은 메시지 전송이 필요하다. 이러한 문제점을 해결하기 위해 로그 레코드 수(Log Record Counter : LRC)를 사용하여 서버에서 데이터 페이지 전송시 LRC와 pageLRC를 같이 전송한다. 또한 ARIES가 부조건 철회를 실시하는데 비하여 조건부 철회를 실시하여 부조건 철회시 데이터베이스 일치성 문제를 개선하였다.

2.3 Extending ARIES-CS 기법

확장 ARIES-CS(Extending ARIES for Client-Server DBMS : Extending ARIES-CS)[11]는 ARIES 고장

기법을 클라이언트-서버 환경으로 확장하였다. [5]와는 다르게 클라이언트에서만 LRC를 이용하여 WAL 규약을 구현하였다. 로그 레코드와 해당 데이터 페이지의 서버 도착 시각의 차이로 인한 비 일관성 극복을 위해 [5]와 같이 조건부 철회를 사용하였다. [11]은 로그 레코드가 서버에 도착했을 때 해당 페이지가 Dirty된 것으로 간주한다.

2.4 2-Phase Locking 기법

기존 기법의 대부분은 트랜잭션이 승인(Commit)될 때 클라이언트가 변경된 데이터 및 해당로그를 모두 서버로 보내고 2단계 승인 프로토콜등을 이용하여 서버와 해당 클라이언트 간에 승인을 완료한다. 클라이언트 트랜잭션은 데이터를 읽거나 변경하는데 적절한 록을 서버로부터 획득한 후 해당 작업을 수행한다. 따라서 서버 DBMS가 록 관리자 역할을 수행한다.

2.5 Callback Locking기법

이 기법은 클라이언트가 데이터 록(Lock Mode)을 함께 캐쉬하고 있어, 트랜잭션이 종료되더라도 록을 소유하고 있어 그 페이지에 대한 유효성을 보장하는 기법이다[13]. 이 기법은 클라이언트가 데이터 페이지에 접근시 매번 서버에게 데이터 페이지 유효성을 검사할 필요가 없다. 그러나, 읽기 전용록(Shared Lock)을 가진 데이터 페이지에 트랜잭션을 변경하고자 할 때 전용록(Exclusive Lock)을 요청 해야한다. 이 경우 서버는 해당 데이터가 캐쉬 되어 있는 다른 모든 클라이언트 사이트에 메세지를 보내어 캐쉬된 록을 회수(Callback)해야 하는 문제점이 있다. [13]에서는 읽기 록 회수기법(Callback Read Locking)사용하였고, [13]에서는 읽기 및 변경용 록을 캐쉬하는 쓰기 록 회수 기법(Callback Write Locking)을 사용하였다.

2.6 No-waiting Locking 기법

이 기법은 클라이언트 사이트에 캐쉬된 데이터의 정확성 점검을 서버의 응답을 기다리지 않고 바로 처리에 들어가는 기법이다[8]. 이 기법의 문제점은 여러 사이트에 캐쉬되어 있을 때 트랜잭션 철회율이 많아진다.

Ⅲ. 트랜잭션 관리 시스템

본 논문의 시스템 환경은 ARIES기법, Extending-ARIES기법, ESM-CS기법 와는 달리 서버의 구성은 디스크 데이터베이스가 아닌 주기억 데이터베이스로 구성하여 디스크의 입출력을 감소 하였고, 백업로그는 디스크에 유지하지 않고 배터리 백업로그를 이용하므로써 디스크 입출력을 제거하였다. 클라이언트

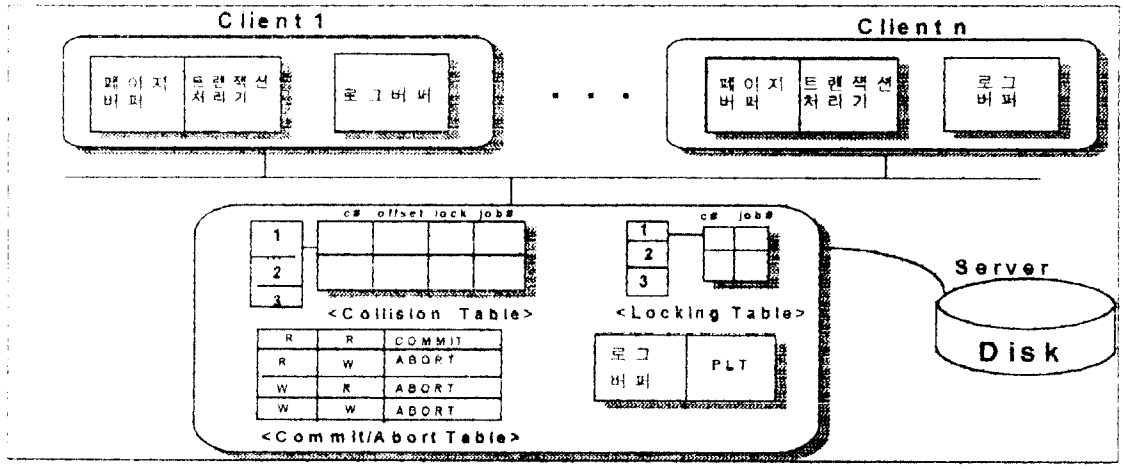

그림 3-1. 전체 시스템 구성도

는 서버의 오버헤드를 감소하기 위해 트랜잭션 처리와 철회(Abort) 동작을 수행하도록 설계하였다. 전체 시스템 구성도는 〈그림3-1〉와 같다. 본 논문에서는 페이지-서버 환경이고, 같은 페이지를 여러 클라이언트가 동시에 사용할 수 있도록 설계하였고, 모델에 대한 가정은 다음과 같다.

〈가정1〉 클라이언트-서버사이의 전송단위는 페이지이다.

〈가정2〉 클라이언트에서는 동시에 하나의 트랜잭션만 처리한다.

〈가정3〉 하나의 트랜잭션은 여러 페이지를 사용할 수 있다.

3.1 서버의 구성

클라이언트-서버 환경에서 서버의 역할은 데이터베이스 관리, 회복관리, 록관리, 동시성 처리, 그리고 백업처리를 관리 담당한다. 기존 기법은 데이터베이스를 디스크에 저장하여 관리 하므로서 디스크 입출력으로 빠른 응답을 할수 없다는 문제점이 제기된다. 본 논문은 이런 문제점을 제거하기 위해 주기억 장치에 전체 데이터베이스를 관리하고 디스크에는 백업용 데이터베이스를 저장 유지 하도록 구성 하였다.

본 논문에서의 로깅 기법은 지연기법을 사용한다. 지연기법은 갱신된 연산을 가지므로 재수행만을 실시하는 장점을 가지고 있다. 그러므로, 시스템 파손시 로그 분석및 재수행, 철회 수행으로 이루어지는 로그처리가 철회 수행은 처리하지 않으므로 빠른 회복이 이루어 진다.

1) 로그버퍼와 PLT

각 클라이언트들이 트랜잭션을 Commit한후, 서버에게 재수행 로그 레코드를 전송한다. 이때 서버는 클라이언트로 부터 전송받은 재수행 로그 레코드를 기록하기 위해 로그 버퍼가 필요하다. 이 로그 버퍼는 시스템 파손시 회복 동작때 필요하고, 또한 시스템 백업시 사용된다.

그리고, Page Log Table(이하 PLT라고 한다)는 어떤 트랜잭션이 하나 이상의 페이지에 접근 할수 있는 환경에서 페이지별 재수행 동작을 위해 각 페이지별 로그정보를 유지 관리한다.

2) 회복 관리기(recovery manager)

클라이언트-서버 환경에서 예기치 못한 파손이 발생할 경우 데이터베이스의 일치성이 유지되게 회복 동작이 반드시 필요하게 된다. 이 때 회복 관리기는 시스템 재적재(reloading) 작업을 수행한후, 로그버퍼를 이용하여 PLT를구성한다. 그리고, PLT를 이용하여 페이지별로 회복동작을 수행하는 일을 관리 담당한다.

3) 동시성 처리기(Concurrency control manager)

여러 클라이언트에서 동시에 같은 데이터 페이지를 요구시 한 클라이언트에게 데이터 페이지를 배정하고 나머지 클라이언트는 수행이 완료 될때까지 기다리게하는 동시성 제어가 필요하다.

본 논문에서는 같은 페이지내에 다른 데이터 아이템을 접근 할수 있도록 동시성 제어기법을 제안 하였다.

4) 백업 처리기

시스템 파손시 빠른 회복을 위해 주기적으로 데이터베이스를 안정된 저장 장치에 기록 하여야 한다. 이때 백업 처리기는 버퍼크기와 PLT크기를 고려하여 일정한 주기로 백업처리를 담당한다.

5) Collision Table

각 페이지별로 데이터 아이템을 사용한 내역을 기록하고 충돌여부를 검사할때 사용한다. 그리고, lock Mode의 충돌 발생시 Commit/Abort Table을 이용하여 트랜잭션을 Commit 또는 Abort 할것인가를 결정한다.

6) Locking Table

각 페이지별로 디렉토리를 구성하고, 그 페이지를 어떤 클라이언트가 어떤 작업 순으로 사용하고 있는지를 기록하여 관리하고 있다. 이 Table는 여러 클라이언트들이 데이터 페이지를 사용하고 있을때 같은 데이터 페이지의 충돌여부를 조사하기 위해 사용한다.

7) Commit/Abort Table

Collision Table에서 데이터 아이템의 충돌 발생시 이 Table에 의해서 Commit/Abort 여부를 결정한다.

3.2 클라이언트의 구성

클라이언트는 로그버퍼(Log Buffer)와 페이지버퍼(Page Buffer)로 구성 된다. 로그버퍼은 철회 수행 로그 레코드(Undo Log Record)와 재수행 로그 레코드(Redo Log Record)를 유지 관리하며, 페이지버퍼는 서버에서 전송 받는 데이터 페이지를 저장하기 위한 기억 공간이다. 철회수행 로그 레코드는 트랜잭션을 철회시 로그 레코드를 이용하여 클라이언트가 철회(Abort)를 직접 수행한다. 재수행 로그 레코드는 트랜잭션 수행 완료시, 그 결과의 로그정보를 서버에 전송하기 위한 레코드이다.

3.3 트랜잭션 모델

1) 트랜잭션 정의

본 논문에서 가정하는 클라이언트-서버 데이터베이스 시스템에서는 전송단위가 페이지이고, 트랜잭션은 클라이언트에서만 처리하는 것으로 가정한다.

한 트랜잭션의 처리가 시작될때 로그에 〈Ti, starts〉라는 로그레코드를 기록하고, 트랜잭션처리시 데이터의 갱신을 기록하기 위해 〈Ti, data item, new value〉의 로그 레코드 형식을 로그에 추가 한다. 그 트랜잭션이 부분 종료시 〈Ti, Commit〉라고 로그에 기록 한 뒤, 그 다음에 변경된 값이 클라이언트 로그버퍼에 기록한다. 이 과정을 하나의 트랜잭션이라 한다.

2) 트랜잭션의 처리 순서 및 과정

〈그림3-1〉에서 보면 서버에서 클라이언트로 화살표가 있는데, 이것은 트랜잭션을 처리 하기위해 데아타 아이템이 속한 페이지를 서버로 부터 전송 받는 것을 나타낸다. 그리고, 반대 방향의 화살표는 클라이언트에서 처리된 트랜잭션의 결과을 서버로 보내는 것을 의미하며, 알고리즘을 살펴 보면 다음과 같다.

1. 클라이언트는 트랜잭션을 처리하기 위해 필요한 데이터 아이템이 속한 페이지를 서버로부터 전송 받아 로그버퍼에 저장
2. 트랜잭션 처리기에 의해 로그버퍼에 있는 페이지를 이용하여 트랜잭션 수행한후, 로그 레코드를 생성하여 로그 버퍼에 저장
3. 트랜잭션이 Commit 직전에 Locking Record을 생성하여 서버로 보냄
4. 서버는 Collision Table, Locking Table, 그리고 Commit/Abort Table을 이용하여 Lock Mode 충돌 여부 조사하여 클라이언트에게 Commit 또는 Abort 메세지 전송
5. Commit 일때 재수행 로그 레코드을 서버로 전송하고 철회 수행 레코드는 제거하고, Abort 일때 철회/재수행 로그 레코드을 모두 제거
6. 서버는 재수행 로그 레코드를 전송 받으면, Log Buffer 해당위치에 로그을 저장하고 PLT을 생성함

Ⅳ. 회복 기법

기존의 클라이언트-서버 환경에서의 회복 기법[5][11]은 클라이언트에서 생성된 로그 레코드와 페이지를 WAL 규약을 이용하여 서버에 전송하였다. 이렇게 함으로써 발생하는 문제점들을 해결하기 위해서 서버에서는 ARIES의 무조건 철회 동작을 수정한 조건부 철회 동작을 수행해야 한다[5][11]. 또한 Dirty된 페이지의 서버 도착 사실로 부터 해당 페이지가 dirty되었음을 DPT(dirty page table)에 기록하기 때문에 발생하는 문제점도 해결해야 한다[5]. 이와 같은 문제점 해결이 필요하게 된 것은 클라이언트는 로그 레코드와 페이지를 서버에 전송하기 때문이다. 항상 로그 레코드는 그 레코드의 생성과 관련된 해당 페이지들을 함께 서버로 보내 주어야 한다. 본 논문은 로그 레코드와 해당 페이지를 서버에 보냄으로 발생하는 문제들을 제거하기 위하여 클라이언트는 서버에 로그 레코드만을 전송하도록 제안한다. 또한 기존의 회복 기법에서는 철회(Abort) 동작을 서버가 함으로써 시스템의 병렬성을 충분히 사용하지 못하였다[5].

본 논문에서는 시스템 병렬성을 충분히 사용하기 위하여 클라이언트에서 철회(Abort)동작을 수행하도록 제안한다.

4.1 로그 구조

1) 클라이언트내의 로그 구조

클라이언트내에는 하나의 트랜잭션을 저장할 수 있는 로그버퍼가 존재하고 있고, 로그버퍼는 재수행 로그버퍼와 철회수행 로그버퍼로 구성되어 있다. 그 구조는 〈그림4-1〉과 같다.

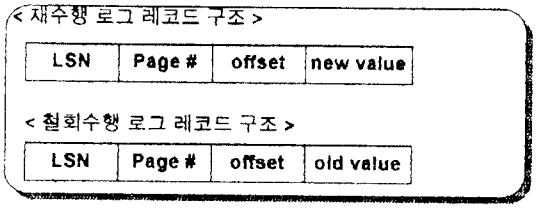

그림 4-1. 로그 레코드 구조

2) 서버 로그 구조

로그버퍼는 배터리 백업 사용을 가정하고, 그 구조는 클라이언트별로 로그를 분리시켜 저장하는데 그 이유는 로그레코드에 빠른 접근과 클라이언트별로 로그번호를 부여함으로써 유일성과 증가성을 보장할 수 있기 때문이다. 그 구조는 〈그림4-2〉와 같다.

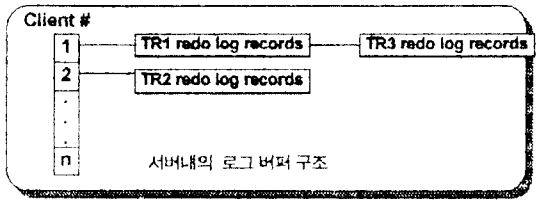

그림 4-2. 로그 버퍼 구조

3) PLT

하나의 트랜잭션이 하나이상의 페이지를 접근할수 있는 환경에서 페이지별 재수행을 동작하기위한 로그 정보 구조이다. 즉, 페이지별 재수행을 보장하기 위한 것이고, 구조는 〈그림4-3〉과 같다.

클라이언트 번호는 로그레코드가 저장된 클라이언트 위치이고, TR-id은 서버에서 트랜잭션의 도착순서를 부여하기위한 필드이다. 그리고, LSN은 해당페이지를 갱신시킨 트랜잭션내의 로그레코드의 일련번호를 의미한다.

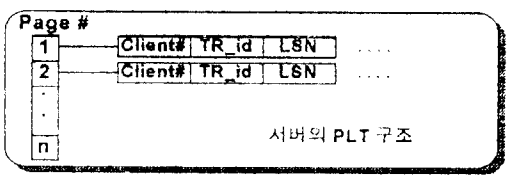

그림 4-3. 서버 PLT 구조

4.2 로깅 동작

클라이언트에서는 트랜잭션을 수행하면서 재수행 로그들과 철회 수행 로그들을 만들어 자신의 로그 버퍼에 저장한다. 만약 트랜잭션 철회시에는 클라이언트 자신이 지닌 철회수행 로그 레코드들을 이용해서 자신이 직접 철회 동작을 수행한다. 트랜잭션 수행 완료시에는 [그림 4-1]의 로그 레코드 형식과 같은 재수행 로그들만을 서버로 전송시키고 서버는 클라이언트로 부터 받은 재수행 로그들을 각각 클라이언트별로 분리하여 [그림4-2]와 같은 로그 버퍼의 구조에 해당 클라이언트 로그 공간의 마지막 위치에 저장한다. 서버는 수행 완료된 트랜잭션의 로그 레코드들에 대한 정보를 PLT에 저장한다. PLT는 [그림 4-3]의 구조와 같이 각각의 페이지별로 분리되어 저장된다. PLT에서 저장하는 로그 레코드들에 대한 정보의 구조는 우선 트랜잭션이 발생된 클라이언트의 번호(C#), 트랜잭션의 식별자(TR id), 그리고 트랜잭션의 수행시작 로그레코드의 LSN값으로 구성된다. 클라이언트가 서버에 페이지 요청한 경우의 로깅 동작은 다음과 같다.

1) 클라이언트가 서버에 페이지를 요청한 경우

어떤 클라이언트가 서버에 페이지를 요청한 경우 페이지 번호(Page Number:P#)를 가지고 PLT에서 해당 페이지의 트랜잭션 정보를 찾아간다. PLT에 해당 페이지가 존재하면 C#, T id, LSN 정보를 이용하여 로그 버퍼에서 해당 페이지의 로그 레코드들에 대해 재 수행 동작을 한다. 서버는 요청한 페이지를 클라이언트에게 준다.

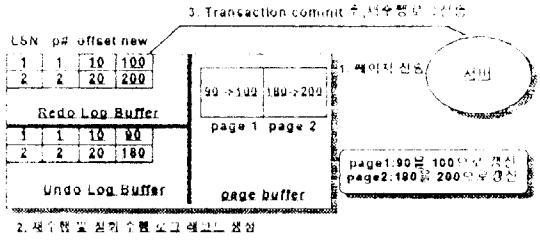

그림 4-4. 클라이언트 로깅동작 구조

클라이언트의 로깅동작의 알고리즘은 다음과 같다.

1. 클라이언트는 자신이 사용하고자 하는 페이지들을 서버로부터 전송받는다.
2. 클라이언트는 자신의 트랜잭션을 수행하면서 재수행 로그레코드와 철회수행 로그레코드를 생성해서 각각의 로그버퍼에 분리시켜 저장한다.
3. 만약, 트랜잭션 철회시에는 자신의 철회수행 로그버퍼를 이용해 직접 철회동작을 수행한다.
4. 트랜잭션 수행완료시에는 서버로 재수행 로그버퍼내의 로그레코드들을 보내고, 철회수행 로그버퍼는 제거한다.

〈그림4-4〉는 클라이언트가 서버에게 페이지를 전송 받아 트랜잭션 수행 동작 구조를 도시하였다.

2) 서버의 로깅동작

아래의 로깅동작은 클라이언트으로부터 로그 레코드를 수신후 서버는 로그버퍼에 클라이언트 별로 로그를 저장한다. 그리고, 로그레코드 정보를 이용하여 PLT을 생성한다. 그 로깅동작 구조는 〈그림4-5〉와 같다.

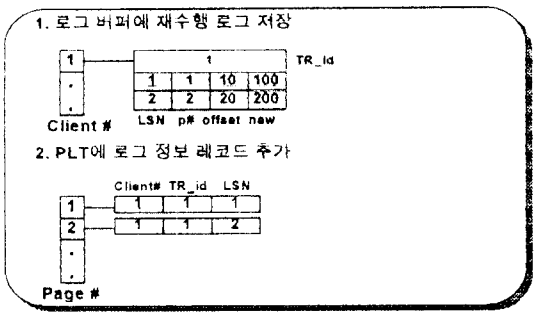

그림 4-5. 서버 로깅동작 구조

알고리즘은 다음과 같다

1. 클라이언트로부터 재수행 로그레코드들을 받은 서버는 로그버퍼의 해당 클라이언트 위치에 유일성이 보장되는 트랜잭션ID를 배 정하여 저장한다.
2. 저장한 로그레코드들의 P# 필드를 조사해서 PLT를 구성한다.
3. 다른 클라리언트로부터 페이지신청을 받을 경우, PLT상에 해당 페이지위치의 로그정보를 이용해 페이지단위의 재수행 동작을 수행한후, 클라이언트로 전송한다.

만약, 클라이언트가 페이지1 신청할 때 서버의 로깅동작은 그림[4-6]과 같다.

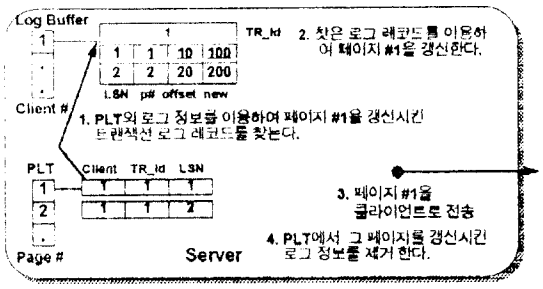

그림 4-6. 서버 로깅동작 구조

4.3 회복 동작

클라이언트-서버 환경에서 예기치 못한 파손이 발생할 경우 데이터베이스의 일치성이 유지되지 못하게 되므로 데이터베이스 시스템의 회복 동작은 반드시 필요하게 된다. 본 절에서는 서버의 데이터베이스 시스템이 파손되었을 경우 회복 동작에 대해서 알아본다.

1) 서버의 데이터베이스 시스템 파손시 회복 동작

서버가 파손후 재가동 되면 분석 단계에서 로그 버퍼를 분석하여 PLT를 재구성한다. 서버는 클라이언트에게 전송 중지명령해제신호를 보낸다. 그후, 클라이언트로 부터 페이지 요청을 받은 후 서버는 신청한 페이지를 PLT에 존재하면 트랜잭션 로그 레코드의 시작 주소를 이용해서 트랜잭션 단위로 재 수행 동작을 수행한다. 클라이언트에게 요청된 페이지를 전송하고, PLT에 존재하지 않으면 주기억장치로 페이지를 이동하여 클라이언트에게 전송한다. 주기억장치로 이동되지 않은 페이지에 대해서는 클라이언트로부터 페이지 사용 요청을 받지 않은 시간에 PLT를 이용하여 파손 발생직전 상태로 회복한다. 페이지 단위의 시스템 회복 동작으로 전체 데이터베이스 시스템을 회복하는 것보다 효율적인 회복 동작을 한다.

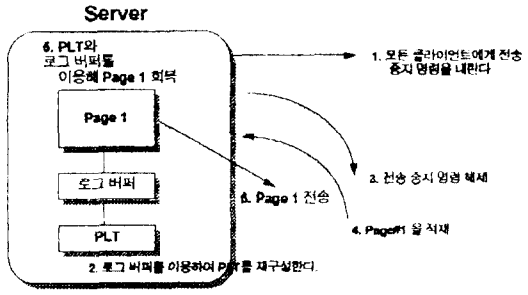

그림 4-7. 회복동작 구조

시스템 파손후 회복동작의 구조는 [그림4-7]에 있으며 알고리즘은 다음과 같다.

1. 서버는 모든 클라이언트에게 중지명령을 지시한다.
2. 로그버퍼를 이용하여 PLT를 재구성하고 클라이언트에게 해제명령을 내린다.
3. 클라이언트로부터 페이지 신청을 받을 경우 해당페이지를 주기억장치로 가져온 후, PLT를 이용해 해당 페이지만을 회복시켜 클 라이언트로 전송한다.
4. 클라이언트로부터 트랜잭션의 재수행 로그레코드가 올 경우에는 로그버퍼에 로그구조를 저장시키고, PLT에 로그레코드를 생성해 저장한다.

V. 동시성 제어 기법

기존의 클라이언트-서버 환경에서의 동시성 제어 기법은 페이지 단위 록(lock)이므로 여러 클라이언트들이 같은 페이지를 동시에 사용하지 못하는 단점이 있다. 즉 같은 페이지 내에 다른 데이터를 접근하고자 할 때는 트랜잭션들의 불필요한 대기시간이 발생하는 문제점이 있다. 또한, [8][11]은 데이터 변경시 모든 클라이언트 사이트에 메세지를 보내어 캐쉬된 록을 회수해야 하는 문제점이 있고, 트랜잭션 철회율이 많아지는 단점이 있다. 본 논문에서는 클라이언트-서버 사이의 전송단위는 기존의 기법과 같이 페이지 단위이고, 록 단위는 데이터 아이템 단위로 하여 같은 페이지를 여러 클라이언트가 사용함으로써 시스템 병렬성과 트랜잭션 처리 시간을 단축하는 새로운 동시성 제어 기법을 제안한다. 그리고, 제안된 기법을 이용하여 서버와 클라이언트 사이 동작을 알고리즘으로 표현하였다.

5.1 시스템 모델 및 가정

클라이언트-서버 환경에서 동시성 제어 시스템은 〈그림3-1〉과 같다. 동시성 제어 모델에 대한 가정은 다음과 같다.

〈가정1〉 먼저 완료한 트랜잭션이 우선 작업순위를 부여받는다.

〈가정2〉 클라이언트들이 같은 페이지에 있는 동일한 데이터 아이템을 사용할 경우를 충돌상태이고, 그 외 경우를 비충돌 상태라고 정의한다.

〈가정3〉트랜잭션은 클라이언트에서만 처리하는 것으로 정의한다.

5.2 클라이언트의 구성

각 클라이언트는 트랜잭션 처리에 필요한 데이터 페이지를 서버로 부터 전송받아 트랜잭션 완료 직전에 Lock Record를 생성하여 서버에 전송한다. 서버는 이 Lock Record 정보를 가지고 Locking Table에 가서 충돌 여부를 조사하여 클라이언트에게 Commit 또는 Abort 메세지를 전송한다.

C#	P#	Offset	lock-field	J#

그림 5-1. lock record

구성내용을 보면 다음과 같다.

c# : 트랜잭션을 처리한 클라이언트 사이트

p# : 트랜잭션이 사용한 페이지 번호

offset : data item 의 값

lock-field : lock Mode

j# : 작업의 실행 순위

5.3 서버의 구성

서버에는 Lock의 충돌 여부를 검사하기 위해 Locking Table, Collision Table, Commit/Abort Table로 구성되어 있다. 첫째 Collision Table은 각 페이지별로 데이터 아이템을 사용한 내역을 기록하고 충돌여부를 검사할 때 사용한다. 그리고, Lock Mode의 충돌 발생시 Commit/Abort Table을 이용하여 트랜잭션을

Commit 또는 Abort 할 것인가를 결정한다. 둘째 Locking Table은 각 페이지별로 디렉토리를 구성하고, 그 페이지를 어떤 클라이언트가 어떤 작업 순으로 사용하고 있는지를 기록하여 관리하고 있다. 이 Table은 여러 클라이언트들이 데이터 페이지를 사용하고 있을 때 같은 데이터 페이지의 충돌여부를 조사하기 위해 사용한다. 셋째, Commit/Abort Table은 Collision Table에서 데이터 아이템의 충돌 발생시 이 Table에 의해서 Commit/Abort 여부를 결정한다. 예를 들면, 데이터 아이템 100을 Read한후, 작업을 완료하고, Collision Table에 Lock을 기록하였다. 그 다음, 클라이언트 3에서 데이터 아이템100을 이용하여 갱신연산후 Commit직전에 Collision Table을 조사했을 때, 충돌발생시 이 트랜잭션을 정상 완료작업으로 처리한다. 그러나, 위의 예에서 클라이언트3 작업이 먼저 완료된 후 클라이언트1이 완료되었을 때는 비정상 작업으로 간주하여 Abort한다. 본 논문에서 가정하는 클라이언트-서버 데이터 베이스 시스템에서는 전송단위가 페이지이고, 트랜잭션은 클라이언트에서만 처리하는 것으로 가정한다.

5.4 동시성 제어 처리 순서 및 과정

〈그림3-1〉에서 보면 서버에서 클라이언트로 화살표가 있는데, 이것은 트랜잭션을 처리 하기 위해 데이터 아이템이 속한 페이지를 서버로 부터 전송 받는 것을 나타낸다. 그리고, 반대 방향의 화살표는 클라이언트에서 처리된 트랜잭션의 결과를 서버로 보내는 것을 의미한다. 클라이언트-서버 사이에 같은 페이지를 동시에 처리 할 수 있도록 제안된 기법의 알고리즘은 다음과 같으며 [그림5-2]에서 나타내고 있다.

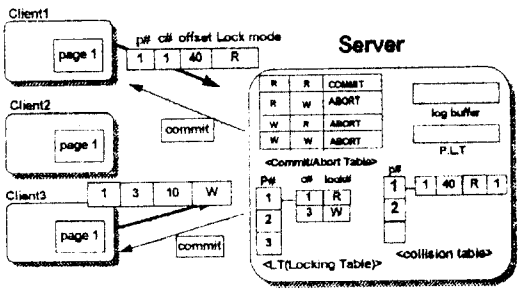

그림 5-2. 동시성 제어 구조

1. 클라이언트는 서버에게 페이지신청
2. 서버는 Collision Table에 C#, P#, 그리고 페이지별 작업순위를 기록
3. 서버는 해당 페이지를 클라이언트에게 무조건 전송
4. 클라이언트는 트랜잭션을 Commit하기 직전 Locking Record를 생성하여 서버에게 전송

5.1 Collision Table에 해당 페이지 존재시 Lock 충돌 여부 조사

- 충돌시 : 1) Commit/Abort Table 조사 : Commit 또는 Abort 여부결정
  2) Locking Table에서 같은 P#를 갖는 Entry 존재 시에는 Lock Records만 제거, 그렇지 않은 경우에는 Lock Records와 Lock Table제거
- 비 충돌시 : Collision Table에 같은 P#를 갖는 Entry존재 시에는 Lock Table에 Lock Records기록
  그렇지 않은 경우에는 Lock Records 제거

5.2 Collision Table이 존재하지 않는 경우에는

- 만약 Locking Table에서 같은 P#를 갖는 Entry존재 시에는 Lock Records를 Lock Table에 기록
- 그렇지 않은 경우에는 Lock Records 제거

6. 해당 클라이언트에게 Abort 또는 Commit 메세지 전송
7. 트랜잭션 Commit 또는 Abort 수행
8. 서버는 Collision Table에서 해당 Entry 제거

Ⅵ. 비교 분석

6.1 회복기법의 비교

본 장에서는 본 논문이 제안한 클라이언트-서버 데이터베이스 고장 회복 기법과 기존의 연구로 ESM-CS[5]와 비교 분석한다. 본 논문이 제안한 회복 알고리즘의 특징을 살펴보면 다음과 같다.

- 본 논문에서는 서버에서 관리하는 로그 공간을 클라이언트 별로 따로 관리한다.
- 본 논문에서 서버는 수행 완료된 트랜잭션들의 정보를 PLT에 저장하여 관리한다.

- 클라이언트는 수행 완료된 트랜잭션 로그레코드만을 서버로 전송한다.
- 기존의 방법[5]에서는 서버에서 클라이언트의 철회(Abort) 동작을 하지만 본 논문에서는 클라이언트에서 철회 동작을 한다.

본 논문에서 제안한 방법을 기존의 회복 기법과 비교한 때 다음과 같은 장점을 갖게 된다.

1. 클라이언트-서버 환경에서 서버 파손이 발생한 경우, 클라이언트 별로 로그 버퍼를 따로 관리하면, 클라이언트가 사용 중이었던 페이지에 대한 회복이 보다 효율적이 된다.
2. 서버의 파손 후 재 가동시, 전체 데이터베이스를 회복후, 페이지 요청을 받은 데이터 페이지부터 회복하여 클라이언트로 넘겨 준 후 PLT를 이용하여 페이지 요청이 없을 경우 나머지 페이지들을 회복함으로써 보다 효율적인 회복 동작을 하게 된다.
3. 기존의 회복기법[5], [11]들은 로그 레코드와 페이지를 서버로 전송시켰다. 본 논문이 제안한 회복 기법에서는 수행 완료된 재수행로그 레코드만을 전송하여 기존의 회복 기법에서의 문제점을 해결하였다.
4. 본 논문의 회복 기법은 클라이언트에서 수행 완료된 로그 레코드만을 서버로 전송하기 때문에 회복 동작시 분석, 재 수행 동작만을 수행한다.
5. ESM-CS 회복 기법[5]에서는 서버가 철회(Abort) 동작을 수행함으로써 시스템의 병렬성을 사용하지 못하였다. 본 논문은 클라이언트에서 철회(Abort)동작을 수행함으로써 시스템 병렬성을 충분히 고려하였다.

6.2 동시성 제어기법의 비교

본 장에서는 본 논문이 제안한 클라이언트-서버 데이터 베이스 동시성 제어 기법과 기존의 관련된 연구[12], [13]을 비교 분석한다. 본 논문이 제안한 동시성 제어 알고리즘의 특징을 살펴보면 다음과 같다.

- 본 논문에서는 클라이언트-서버 사이의 전송단위는 페이지이고, 록킹 단위는 데이터 아이템으로 구성되어 있다.
- 본 논문에서 클라이언트들이 동시에 같은 페이지를 사용할 수 있다.
- 클라이언트는 수행완료된 트랜잭션을 서버에 있는 Collision Table을 이용하여 충돌여부를 조사한 후, 신속히 처리한다.
- 기존의 방법[8][11]과 달리 데이터와 록을 분리 관리한다.

본 논문에서 제안한 방법을 기존의 동시성 제어 기법과 비교할 때 다음과 같은 장점을 갖게 된다.

1. 클라이언트-서버 환경에서 페이지 별로 록 정보를 따로 관리하여 클라이언트가 사용 중이었던 페이지에 대한 정보를 파악하여 충돌여부를 신속히 결정한다.
2. 기존 기법과 달리, 같은 페이지를 여러 클라이언트들이 동시에 접근을 허용함으로써 시스템 병렬성이 향상된다.
3. 기존의 동시성 제어기법[8][11]들은 데이터와 록을 함께 캐쉬하고 있어 변경시 다른 클라이언트에 있는 캐쉬된 록을 회수(callback)해야한다. 본 논문이 제안한 동시성 제어 기법에서는 낙관적 기법을 사용하여 기존의 문제점을 해결하였다.
4. 본 논문의 전송 단위가 페이지이므로 클라이언트와 서버사이의 병목 현상이 감소되고, 록킹 단위가 데이터 아이템이므로 충돌을 줄일 수 있다.
5. 기존 기법과 달리, 충돌 검사시 먼저 페이지별 검사후 데이터 아이템 단위로 검사함으로써 충돌여부를 신속히 처리 할 수 있다.

Ⅶ. 결 론

본 논문에서는 클라이언트-서버 데이터베이스 환경에서 시스템 운영시 발생하는 여러가지 문제점들을 제시하고, 효율적인 동시성 제어 기법과 회복기법을 제안하였다. 워크스테이션의 보급이 증가하고 고속의 네트워크 통신기술의 발전으로 클라이언트-서버 환경의 데이터베이스 시스템이 요구됨에 따라, 기존의 디스크 기반 데이터베이스 시스템의 회복 알고리즘의 문제점을 파악하여 개선된 회복기법을 제시

하고, 회복 알고리즘을 제시하였다. 비록 클라이언트에서 철회 동작을 수행해야 하지만, 높은 처리 기술을 가진 클라이언트를 최대한 활용하고, 서버의 작업량을 줄임으로써 보다 신속하게 클라이언트의 서비스 요청을 처리한다.

그리고, 여러 클라이언트들이 동시에 페이지를 사용할 수 있도록 트랜잭션 처리 시스템을 설계하고, 알고리즘을 제시하였다. 그 결과 시스템 병렬성을 향상시키고 트랜잭션 처리시간을 단축 할 수 있게 되었다. 앞으로의 연구는 본 논문에서 제시된 방법과 기존 방법의 효율적인 비교를 위한 성능 평가가 필요하다.

참 고 문 헌

1. Bernstein, P., Hadzilacos, V., and Goodman, N., "Concurrency control and Recovery in Database Systems", Addition-Wesley, 1987.
2. Carey, M., Dewitt, D., Richardson J., Schekita, E., "Storage Management for Objects in EXODUS", in Object-Oriented Concepts, Databases, and Applications, W. Kim F. Lochovsky, eds., Addition-Wesley, 1989.
3. Carey, M., Frankin, M., Liviny, M., Schekita, E., "Data Caching Tradeoffs in Client-Server DBMS Architecture", Proc. ACM SIGMOD Conf., Denver, Jun 1991.
4. Daniels, D., Spector A., Thompson, D., "Distributed Logging for Transaction Processing", Proc. ACM SIGMOD Conf. San Francisco, May, 1987.
5. Franklin, M., Zwilling, M., Tan, C., Carey, M., Dewitt, D., "Crash Recovery in Client-Server EXODUS", TR #1081, Comp Sci Dept., Univ. of Wisconsin-Madison, Mar. 1992.
6. Haerder, T., Reuter, A., "Principles of Transaction Oriented Database Recovery-A Taxonomy", Computing Surveys, Vol. 15, No. 4, Dec., 1983.
7. Stonebraker, M., "Architecture of Future Database Systems", Data Eng., Vol. 13, No. 4., Dec. 1990.
8. Wang, Y., Rowe, L., "Cache consistency and Concurrency Control in a Client-Server DBMS Architecture", Proc. ACM SIGMOD Donf., Denver, June 1991.
9. The Committee for Advanced DBMS Function, "Third Generation Data Base System Maifesto", SIGMOD Record, Vol. 19, No. 3, Sept. 1990.
10. Mohan, C., Handerle, D., Lindsay, B., Pirahesh, H., Schwarz, P., "ARIES: A Transaction Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging", IBM Research Report RJ16649, IBM ARC, NOV., 1990.
11. M. Franklin et. al., "Local Disk Caching for Client-Server Database Systems," Proc. VLDB, 1993, pp. 641-654.
12. M. Carey et. al., "Data Caching Tradeoffs in Client-Server DBMS Architectures," Proc. ACM SIGMOD, 1991, pp. 596-609.
13. Y. Wang and L. Rowe, "Cache Consistency and Concurrency Control in a Client/Server DBMS Architecture," Proc. ACM SIGMOD, 1991, pp. 367-376.
14. 이강우, 김형주, "클라이언트-서버 DBMS로의 ARIES 확장", 한국 정보 과학회 논문지 제21권 4호

趙 成 濟(Sung Jae Cho) 정회원

1985년 : 서울산업대학교 전자계산학과 학사

1987년 : 동국대학교 전자계산학과 석사

1995년 : 홍익대학교 전자계산학과 박사과정 수료

1991년~1995년 2월 : 영동전문대학 전자계산학과 조교수

1995년 3월~현재 : 경주대학교 컴퓨터공학과 전임강사

※주관심분야 : 클라이언트-서버 데이터베이스, 객체지향 데이터베이스, 멀티미디어 시스템

金 庚 昶(Kyung Chang Kim) 정회원

1978년 : 홍익대학교 전자계산학과 졸업(이학사)

1980년 : 한국과학기술원 전산학과 졸업(이학석사)

1990년 : University of Taxas at Austin 전산학과(이학박사)

1990년 6월~1991년 3월 : 미국해군대학원 객원교수
1991년 3월~현재 : 홍익대학교 컴퓨터공학과 조교수
※주관심분야 : 클라이언트-서버 데이터베이스, 객체지향 데이터베이스, 멀티미디어 시스템

金 淇 龍(Kil Ryong Kim) 정회원
현재 : 홍익대학교 컴퓨터공학과 교수
※주관심분야 : 클라이언트-서버 데이터베이스, 데이터 통신 자료처리
1993년 8월호 참조