페이지-서버 객체지향 DBMS에서의 PLT를 이용한 회복기법

조 성 제†

요 약

기존의 데이터베이스관리시스템(Database Management System : DBMS)은 대부분 메인 프레임 컴퓨터 이상의 컴퓨터와 같은 강력한 컴퓨터에 설치되고, 이 컴퓨터에 터미널을 연결하여 운용하는 방식이 대부분이었다. 최근 들어 저렴하고 강력한 워크스테이션과 초고속의 통신장비 등장으로 클라이언트-서버 DBMS구조는 많은 사람들에 의해서 연구되고 있다. 그러나, 클라이언트-서버 DBMS의 고장회복에 관한 연구는 아직까지 깊이있게 연구되지 않고 있는 실정이다. 본 논문은 클라이언트-서버 환경 중 페이지-서버 환경의 회복기법에 관해서 논한다. 제한된 기법은 서버파손이 발생할 경우 페이지별로 로그일련번호 부여로 신속히 회복을 수행할 수 있고, 클라이언트에서 철회동작을 수행함으로써 시스템 병렬성을 사용하였다. 그리고 기존의 방법과 달리 재수행 로그레코드만을 서버로 전송하므로 WAL(Write Ahead log) 규약을 위한 오버헤드가 감소되었다.

Recovery Technique using PLT in a Page-Server Object Oriented DBMS

Sung-Je Cho†

ABSTRACT

Recently, with a drop in memory chip price and rapid development of mass storage memory chip technology, research on client-server Database Management Systems has gained wide attention. This Paper discusses the recovery technique in a page-server OODBMS environment. Although most researchers have studied client-server systems, the recovery technique has been a poor researches. In this paper, client transfers only the completed log records to the server and resolves the problems in the current recovery techniques. In addition, the server manages only the execution of completed log records suggesting a simple recovery algorithm, Client uses system concurrency, fully, by acting to abort actions and it suggests a page unit recovery technique to reduce time that is required in the whole database.

키워드 : 클라이언트-서버(client-server), 페이지-서버(page-server), OODBMS, PLT

1. 서 론

강력한 워크스테이션들과 서버들로 구성된 네트워크가 다양한 응용 프로그램들의 실행 환경이 됨에따라 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 DBMS는 클라이언트 프로세스가 수행되는 여러 개의 워크스테이션과 데이터베이스를 관리하는 서버 프로세스가 동작하는 데이터베이스 서버로 구성된다. 클라이언트-서버 환경에서는 중앙 집중식 데이터베이스 시스템과 달리 각각의 클라이언트들이 자신의 컴퓨터를 지니고, 그 컴퓨터에서 트랜잭션을 발생시켜 수행한다. 그러므로, 실시간 접근을 위해서 클라이언트들의 컴퓨터에 서버의 데이터를 가져온 후 클라이언트가 직접 접근하는 방법을 사용할 수 있다. 이러한 접근 방법은 클라이언트-서버 데이터베이스 시스템 구조 중 서버에 특정 데이터를 요청하는 방법에 해당한다[5]. 서버에 데이터를 요청하는 시스템은 그 성격에 따라 페이지 서버와 객체 서버로 나뉠 수 있다. 이 기법들은 클라이언트 워크스테이션의 기억장치 이용율과 처리율을 높여서 서버의 병목현상을 최소화 할 수 있다는 장점이 있다. 또한 각각의 클라이언트가 기억장치와 처리능력을 지니므로 트랜잭션 수행을 고려할 때 효율적이라 할 수 있다. 그러나 클라이언트가 서버로 특정 데이터를 요청하는 구조는 전통적인 회복 기법과 비교해 볼 때 몇 가지 문제점을 가지고 있는데, 그것은 클라이언트-서버의 구조적 문제로 인한 것이다. 즉, 데이터베이스의 갱신 동작이 클라이언트에서 이루어지고, 서버는 갱신에 대한 로그와 갱신 발생 전 상태의 데이터베이스

† 정 회 원 : 국립 한국전통문화학교 교수
논문접수 : 2002년 9월 30일, 심사완료 : 2002년 12월 16일

를 보유한다. 그러므로 기존의 중앙 집중식 데이터베이스 시스템에서 사용하는 회복 방법으로는 효율적 회복 동작이 이루어지기 어렵다. 본 논문에서는 기존기법들의 문제점을 살펴보고, 새로운 기법을 제안하고자 한다. 기존기법들의 문제점을 나열하면 다음과 같다.

첫째, 각 페이지에 대한 로그일련번호 부여를 서버가 수행함으로써 병목현상이 발생한다.

둘째, 수행 완료한 로그정보와 페이지를 서버에 전송할 때 WAL 규약을 준수해야한다. 이를 위한 로그레코드 일련번호 부여에 대한 오버헤드 발생한다.

셋째, 서버가 철회동작을 수행함으로써 시스템의 병렬성을 사용하지 못하였다.

넷째, 회복시 로그 분석(LSN, LRC, PageLSN)에 대한 오버헤드가 상당히 발생된다.

본 논문에서는 클라이언트-서버 기반으로 하는 페이지-서버 환경에서의 효율적인 회복 동작에 대해서 제안한다. 제안하는 회복 기법은 트랜잭션 모델을 설계하여 ESM-CS 기법 문제점을 개선하고자 한다. 논문의 2장에서는 회복기법의 관련연구를, 나머지 장에서는 본 논문이 제안하는 페이지-서버 데이터베이스 모델, 로깅, 회복동작, 그리고 성능분석에 대해서 기술한다. 마지막 장에서는 앞으로 이루어져야 할 연구방향을 제시한다.

2. 관련 연구

LAN이 급속하게 발전하고, 또한 워크스테이션이 아주 빠르고, 값이 저렴해짐에 따라 DBMS의 구조가 클라이언트-서버 모델화 되기 시작했다. 처음에는 완전한 분산 DBMS로의 의도 강했지만, 너무 복잡하고 현실적으로 완전한 구현이 힘들므로 LAN으로 연결 가능한 지역 내에서는 클라이언트-서버 DBMS가 자리잡기 시작했다. 객체지향 DBMS가 상품화되기 시작하고, 컴퓨터 처리에 있어서 클라이언트-서버 구조가 보편화됨에 따라 클라이언트-서버 구조의 객체지향 DBMS가 나타났다. 객체지향 DBMS 구조는 크게 객체-서버, 페이지-서버 방법으로 분류할 수 있다. 객체-서버 DBMS 모델은 클라이언트-서버 사이의 정보 전송 단위가 객체이다. 이 구조에서는 서버가 객체의 의미를 파악할 수 있기 때문에 서버에서 객체에 메소드를 적용할 수 있다. O2, ORION, Gemstone 모델은 초기 프로토타입들이 이 구조를 채택하고 있다. 이 구조에서는 객체지향 DBMS의 대부분의 기능들이 클라이언트와 서버에 중복되어 있다. 이 모델은 한번의 전송에 여러 개의 객체를 보내지 못하므로 시스템의 성능이 저하될 수 있고, 클라이언트와 서버간의 동일한 객체에 대해서 서로 다른 사본을 가질 수 있으므로 서버에서 메소드 실행시는 이러한 비일관성 문제를 해결해야 하는 부담이 있다.

페이지-서버 DBMS 모델은 서버가 단지 페이지만을 관리하므로 서버에서 객체를 인식할 필요가 없다. 이 구조에서는 클라이언트와 서버간의 전송단위는 페이지가 된다. Object-Store, O2 등이 이 구조를 채택하고 있다. 이 구조에서는 서버는 페이지 버퍼와 페이지, 파일, 동시성 제어, 회복 등을 담당하는 입출력 레벨의 저장 시스템으로 구성되며, 객체지향 DBMS의 상위 기능들은 대부분 클라이언트 프로세스에서 담당하게 된다. 이 모델은 서버는 객체의 의미를 알 수 없으므로 동시성 제어와 회복에만 전념하면 된다. 따라서, 하나의 서버 프로세스가 다수 클라이언트 프로세스를 효율적으로 지원할 수 있다. 그리고, 클라이언트-서버 DBMS에서는 데이터를 갱신하는 트랜잭션은 클라이언트에서 수행되고, 고장회복시에 수행되는 고장회복 작업은 서버에서 수행된다. 그러므로, 클라이언트-서버 구조에서의 고장회복 알고리즘은 기존의 중앙 집중식 DBMS에서의 고장회복이 복잡하다. 그러나, 아직까지 이 분야에 대한 깊은 연구가 되고 있지 않은 실정이다. 대부분 고장회복은 기존의 ARIES 기법을 클라이언트-서버 DBMS로 확장시키는 연구가 대부분이다. 본 논문에서는 ARIES 회복기법, ESM-CS 고장 회복 기법, Extending ARIES-CS 기법 등을 살펴보고자 한다.

ARIES 회복기법은 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. [10]을 고장 회복 알고리즘으로 사용하는 시스템은 파손 발생 후 재가동시 분석 단계, 재수행 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 재수행 단계, 철회 단계를 수행한다.

ESM-CS 고장 회복 기법(Client-Server Exodus Storage Manager System)은 최근에 Wisconsin 대학에서 개발한 시스템으로 클라이언트-서버 환경에서의 데이터베이스 시스템으로는 대표적 시스템이며 다른 많은 연구에도 인용되었다[5]. 이 시스템은 각각의 클라이언트들은 독자적으로 수행되는 하나의 프로세서로서 자신의 메모리 캐쉬와 록 캐쉬를 가지고 있으며 한번에 하나의 트랜잭션을 처리한다. 클라이언트에서의 WAL 규약의 구현을 위해서 LSN(Log Sequence Number)을 사용한다. LSN을 사용시 클라이언트-서버 사이에 많은 메시지 전송이 필요하다. 이러한 문제점을 해결하기 위해 LRC(Log Record Counter)를 사용하여 서버에서 데이

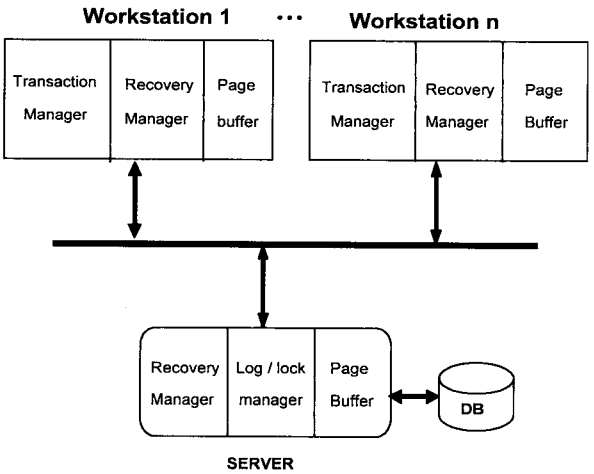
터 페이지 전송시 LRC와 pageLRC를 같이 전송한다. 또한 ARIES가 무조건 철회를 실시하는데 비하여 조건부 철회를 실시하여 무조건 철회시 데이터베이스 일치성 문제를 개선하였다.

Extending ARIES-CS(Extending ARIES for Client-Server DBMS)[11]은 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 비일관성 극복을 위해 [5]와 같이 조건부 철회를 사용하였다. [11]은 로그 레코드가 서버에 도착했을 때 해당 페이지가 dirty된 것으로 간주한다.

3. 트랜잭션 관리 시스템

트랜잭션은 DBMS, 분산 시스템, 그리고 기타 트랜잭션 처리 시스템에서 이미 오래 전부터 잘 알려진 개념이다. 트랜잭션은 데이터의 신뢰도가 매우 중요하고, 이러한 데이터를 여러 사람이 동시에 사용되는 시스템에서 사용되는 작업의 단위가 된다. 대부분의 데이터베이스관리 시스템에서는 동시성제어 기능과 고장회복기능을 트랜잭션 단위를 이용하여 구현한다.

본 논문의 시스템 환경은 ARIES, EXTENDING-ARIES, ESM-CS와는 달리 새로운 트랜잭션 모델을 제시 하였다. 전체 시스템 구성도는 (그림 3-1)과 같다.



(그림 3-1) 전체 시스템 구성도

3.1 서버의 구성

객체지향 DBMS가 상품화되기 시작하고, 컴퓨터 처리에 있어서 클라이언트-서버 구조가 보편화됨에 따라 클라이언트-서버 구조의 객체지향 DBMS가 나타났다. 본 논문에서는 클라이언트-서버 DBMS에서 전송단위가 페이지로 구성되는 페이지-서버 객체지향 DBMS 모델을 설계하였다.

페이지-서버환경에서 서버의 역할은 데이터베이스 관리, 회복관리, 록관리, 동시성 처리기, 그리고, 백업처리기를 관리 담당한다. 클라이언트 프로세스는 사용자 응용프로그램과 클라이언트 라이브러리로 구성되며 하나의 트랜잭션을 실행한다. 서버는 데이터베이스와 로그에 대한 저장장치를 보유하며, 서버 프로세스는 데이터와 인덱스에 동시성제어, 페이지 할당, 그리고, 회복기능을 지원하는 역할을 담당한다. 서버 프로세스는 여러개의 클라이언트 프로세스에서 요청을 지원한다. 로깅 기법은 지연기법을 사용한다. 지연기법은 갱신된 연산을 가지므로 재수행 만을 실시하는 장점을 가지고 있다. 그러므로, 시스템 파손시 로그 분석 및 재수행, 철회수행으로 이루어지는 로그 처리가 철회수행은 처리하지 않으므로 빠른 회복이 이루어진다. 각 클라이언트들이 트랜잭션을 commit한 후, 서버에게 재수행 로그 레코드를 전송한다. 이때 서버는 클라이언트로부터 전송받은 재수행 로그 레코드를 기록하기 위해 로그 버퍼가 필요하다. 이 로그버퍼는 시스템 파손시 회복 동작 때 필요하고, 또한 시스템 백업이 사용된다. 그리고, Page Log Table(이하 PLT라고 한다)은 어떤 트랜잭션이 하나 이상의 페이지에 접근 할 수 있는 환경에서 page별 재수행 동작을 위해 각 페이지별 로그정보를 유지 관리한다. 회복 관리기(recovery manager)는 클라이언트-서버 환경에서 예기치 못한 파손이 발생할 경우 데이터베이스의 일치성이 유지되게 회복 동작이 반드시 필요하게 된다. 이 때 회복 관리기는 시스템 재적재(reloading) 작업을 수행한 후, 로그버퍼를 이용하여 PLT를 구성한다. 그리고, PLT를 이용하여 페이지별로 회복동작을 수행하는 일을 관리 담당한다. 클라이언트는 로그 버퍼와 페이지 버퍼로 구성되어 있다. 로그는 철회 수행 로그레코드(undo log record)와 재수행 로그 레코드(redo log record)를 유지 관리하며, 페이지 버퍼는 서버에서 전송받는 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 로그 정보를 서버에 전송하기 위한 레코드이다.

3.2 트랜잭션 모델

3.2.1 트랜잭션 정의

본 논문에서 가정하는 페이지-서버 객체지향 데이터베이스시스템에서는 전송단위가 페이지이고, 트랜잭션은 클라이언트에서만 처리한다. 한 트랜잭션의 처리가 시작될 때 로그에 <Ti, starts>라는 로그레코드를 기록하고, 트랜잭션 처리시마다 데이터의 갱신을 기록하기 위해 <Ti, data item, new

value>의 로그 레코드 형식을 로그에 추가한다. 그 트랜잭션이 부분 종료시 <Ti, commit>라고 로그에 기록 한 뒤, 그 다음에 변경된 값이 클라이언트 로그버퍼에 기록한다. 이 과정을 하나의 트랜잭션이라 한다.

3.2.2 트랜잭션의 처리 순서 및 과정

(그림 3-1)에서 보면 서버에서 클라이언트로 화살표가 있는데, 이것은 트랜잭션을 처리하기 위해 데이터 아이템이 속한 페이지를 서버로부터 전송받는 것을 나타낸다. 그리고, 반대 방향의 화살표는 클라이언트에서 처리된 트랜잭션의 결과를 서버로 보내는 것을 의미한다.

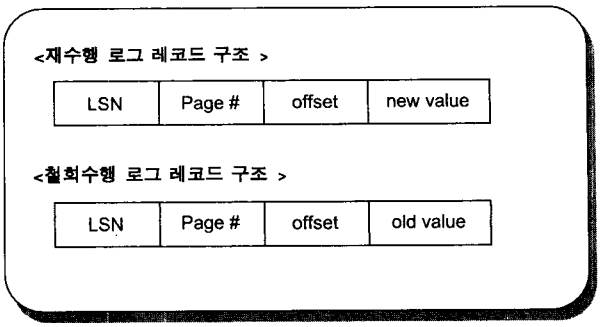
4. 페이지-서버 DBMS의 회복 기법

기존의 페이지-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 사실로부터 해당 페이지가 dirty 되었음을 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 보내주어야 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

4.1.1 클라이언트내의 로그 구조

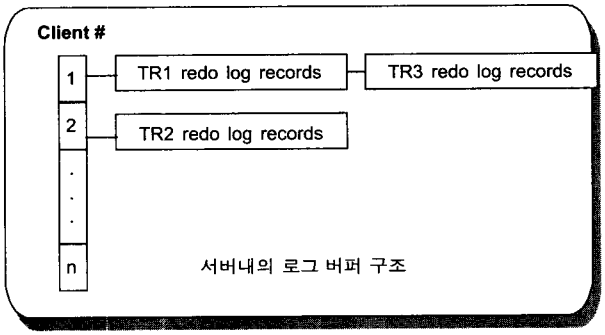
첫째, 클라이언트는 트랜잭션을 처리하기 위해 필요한 데이터 아이템이 속한 페이지를 서버로부터 전송받아 로그버퍼에 저장하고, 둘째, 트랜잭션 처리기에 의해 로그버퍼에 있는 페이지를 이용하여 트랜잭션 수행한 후, 로그 레코드를 생성하여 로그 버퍼에 저장한다. 클라이언트 내에는 하나의 트랜잭션을 저장할 수 있는 로그버퍼가 존재하고 있고, 로그버퍼는 재수행 로그버퍼와 철회수행로그버퍼로 구성되어 있다. 그 구조 (그림 4-1)과 같다.



(그림 4-1) 로그 레코드 구조

4.1.2 서버로그 구조

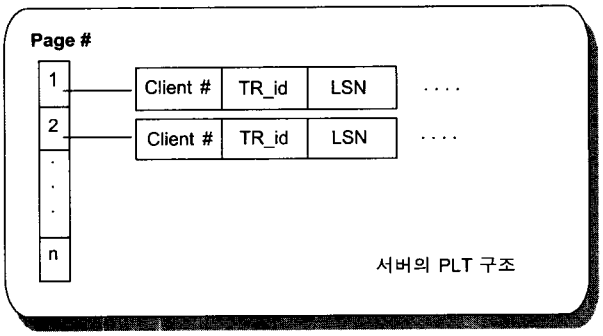
클라이언트별로 로그를 분리시켜 저장하는데 그 이유는 로그레코드에 빠른 접근과 클라이언트별로 로그번호를 부여함으로써 유일성과 증가성을 보장할 수 있기 때문이다.



(그림 4-2) 로그 버퍼 구조

4.1.3 PLT

하나의 트랜잭션이 하나이상의 페이지를 접근할 수 있는 환경에서 페이지별 재수행을 동작하기 위한 로그정보구조이다. 즉, 페이지별 재수행을 보장하기 위한 것이고, 그 구조는 (그림 4-3)과 같다.



(그림 4-3) 서버 PLT 구조

클라이언트번호는 로그레코드가 저장된 클라이언트 위치이고, TR-id는 서버에서 트랜잭션의 도착순서를 부여하기 위

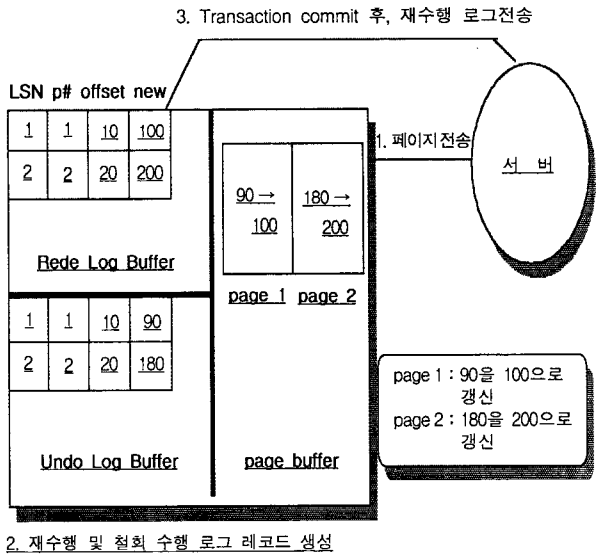
한 필드이다. 그리고, LSN은 해당페이지를 갱신시킨 트랜잭션내의 로그레코드의 일련번호를 한다.

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 클라이언트들로부터 받은 재수행 로그들을 각각 클라이언트별로 분리하여 (그림 4-2)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(T_id), 그리고 트랜잭션의 수행 시작 로그레코드의 LSN 값으로 구성된다. 클라이언트가 서버에 페이지 요청한 경우의 로깅 동작은 다음과 같다.

4.2.1 클라이언트가 서버에 페이지를 요청한 경우

어떤 클라이언트가 서버에 페이지를 요청한 경우 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. 재수행 동작을 한다. 서버는 요청한 페이지를 클라이언트에게 준다.



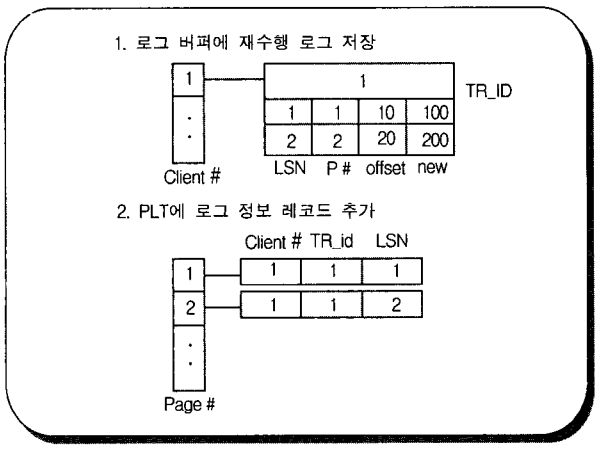
(그림 4-4) 클라이언트 로깅동작 구조

클라이언트의 로깅동작의 알고리즘은 다음과 같다.

① 클라이언트는 자신이 사용하고자 하는 페이지들을 서버로부터 전송 받는다.
② 클라이언트는 자신의 트랜잭션을 수행하면서 재수행 로그레코드와 철회수행 로그 레코드을 생성해서 각각의 로그버퍼에 분리시켜 저장한다.
③ 만약, 트랜잭션 철회시에는 자신의 철회 수행 로그버퍼를 이용해 직접 철회동작을 수행한다.
④ 트랜잭션 수행 완료시에는 서버로 재수행 로그버퍼내의 로그레코드들을 보내고, 철회수행 로그버퍼는 제거한다.

4.2.2 서버의 로깅동작

아래의 로깅동작은 클라이언트로 부터 로그레코드를 수신 후 서버는 로그버퍼에 클라이언트 별로 로그를 저장한다. 그리고, 로그레코드 정보를 이용하여 PLT을 생성한다.



(그림 4-5) 서버 로깅동작 구조

알고리즘은 다음과 같다

① 클라이언트로부터 재수행 로그레코드들을 받은 서버는 로그버퍼의 해당 클라이언트 위치에 유일성이 보장되는 트랜잭션ID를 배정하여 저장한다.
② 저장한 로그레코드들의 Page#필드를 조사해서 PLT를 구성한다.
③ 다른 클라이언트로부터 Page 신청을 받을 경우, PLT 상에 해당 Page 위치의 로그정보를 이용해 Page 단위의 재수행 동작을 수행한 후, 클라이언트로 전송한다.

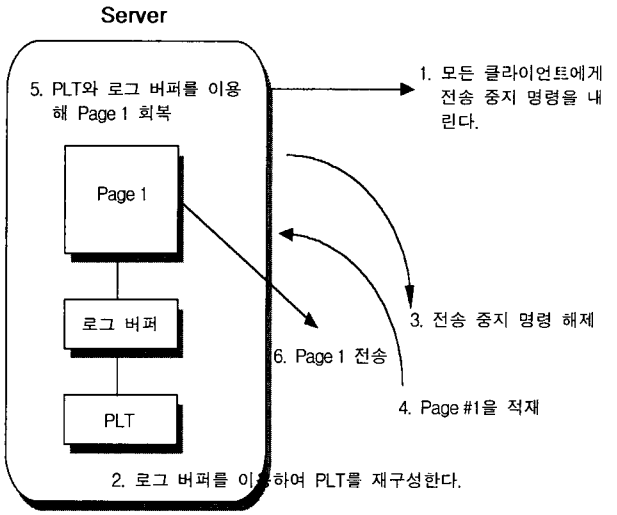
4.3 회복 동작

클라이언트-서버 환경에서 예기치 못한 파손이 발생할 경우 데이터베이스의 일치성이 유지되지 못하게 되므로 데이터

베이스 시스템의 회복 동작은 반드시 필요하게 된다. 본 절에서는 서버의 데이터베이스 시스템이 파손되었을 경우 회복 동작에 대해서 알아본다.

4.3.1 서버의 데이터베이스 시스템 파손시 회복 동작

서버가 파손 후 재가동 되면 분석 단계에서 로그 버퍼를 분석하여 PLT를 재구성한다. 서버는 클라이언트에게 전송 중지명령 해제신호를 보낸다. 그 후, 클라이언트로 부터 페이지 요청을 받은 후 서버는 신청한 페이지를 PLT에 존재하면 트랜잭션 로그 레코드의 시작 주소를 이용해서 트랜잭션 단위로 재수행 동작을 수행한다. 클라이언트에게 요청된 페이지를 전송하고, PLT에 존재하지 않으면 데이터베이스로 페이지를 이동하여 클라이언트에게 전송한다. 주기억장치로 이동되지 않은 페이지에 대해서는 클라이언트로 부터 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 4-6) 회복동작 구조

시스템 파손 후 회복동작의 알고리즘은 다음과 같다.

① 서버는 모든 클라이언트에게 중지명령을 지시한다.
② 로그버퍼를 이용하여 PLT를 재구성하고 클라이언트에게 해제명령을 내린다.
③ 클라이언트로 부터 페이지 신청을 받을 경우 해당 페이지를 주기억장치로 가져온 후, PLT를 이용해 해당 페이지만을 회복시켜 클라이언트로 전송한다.
④ 클라이언트로 부터 트랜잭션의 재수행 로그레코드가 올 경우에는 로그버퍼에 로그 구조를 저장시키고, PLT에 로그 레코드를 생성해 저장한다.

5. 성능 분석 및 평가

본 논문에서 실험 모델에 채택하는 기법 및 환경은 다음과 같다. 먼저 클라이언트/서버 구조 환경에서 본 논문과 같은 페이지/서버 구조인 EMS-CS와 비교 분석한다. 로깅 로그와 로그 버퍼, 디스크 로그를 가지고 있는 경우로 나누어 적용한다. 회복 동작은 서버에 로그버퍼와 페이지 로그 테이블, 그리고 데이터베이스를 기지고 있는 경우로 적용한다.

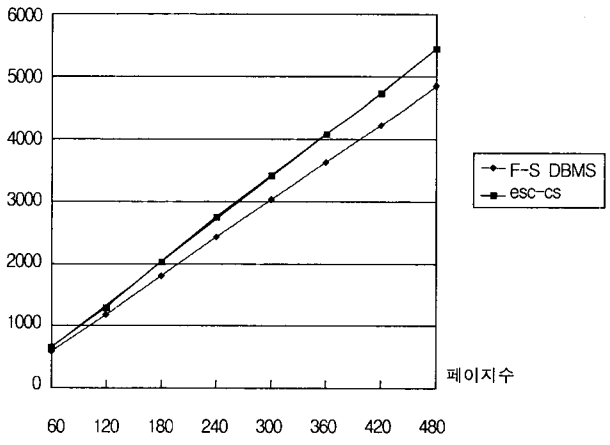
5.1 실험 성능 측정

본 연구에서 제안한 회복 모델은 SLAM Ⅱ의 모델로 표현하였다. 실험은 SUN-SPARC-10 환경에서 이루어졌다. 성능 분석 모델은 이 논문에서 제안한 시스템 환경을 기초로 하여 각 시뮬레이션 단계마다 약간의 변형과 매개 변수에 변화를 주어 실행하였다. 시뮬레이션의 목적은 제안된 모델의 회복시간, 트랜잭션 처리시간으로 나누어 성능을 평가한다. 회복시간은 시스템이 파손된 시점에서부터 클라이언트들이 요구하는 페이지들을 전송하는데 걸리는 총 시간으로 한다.

그리고, 트랜잭션 처리시간은 트랜잭션이 생성되어 수행한 후 로그 레코드를 서버로 전송하는데 걸리는 총 시간으로 한다. 여기서 동시성제어를 제외한다.

5.2 로깅 기법을 이용한 회복시간

(그림 5-1)은 회복하여 서비스시간(초 단위)이 0에서 60000까지 증가됨에 따라 cpu 로드를 나타낸 것이다. 본 비교에서는 트랜잭션 처리시간은 고려하지 않고, 단지 시스템이 파손시 서버를 회복하여 클라이언트에게 해당 페이지를 전송하는데 걸리는 시간을 분석하였다. 이 실험결과 전체적으로 본 기법이 EMS-CS 기법 보다 약간 우세한 것으로 분석되었다. 그 이유는 첫째는 클라이언트들이 신청 한 페이지를 로그버퍼를



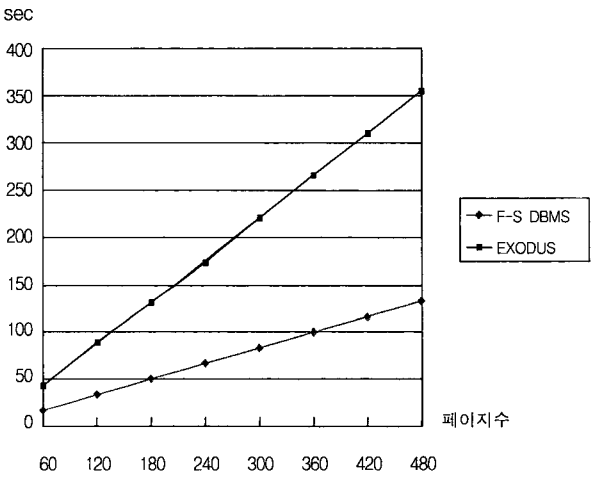
(그림 5-1) 로깅을 이용한 회복시간

이용하여 먼저 재적재 한 후 회복하므로 시간이 단축되었다. 둘째는 EMS-CS 기법의 회복동작은 로그분석, 재수행, 철회 동작 등으로 처리 시간이 너무 많이 걸리고, 본 기법은 간단한 로그분석과 재수행 동작만으로 이루어진다. 그래서 페이지 수의 증가 따른 약간의 개선이 되었다.

5.3 로깅 기법을 이용한 트랜잭션 처리시간

(그림 5-2)는 페이지의 크기가 변화됨에 따라 부가되는 CPU로드 측면에서 시뮬레이션을 수행한 결과이다. 각 기법을 비교하면 제안된 기법이 EMS-CS 기법 보다는 약간 우수한 것으로 나타났다. 트랜잭션의 처리시간(초 단위)은 0에서 4000까지 증가됨에 따라 cpu 로드를 나타낸 것이다. 본 비교에서는 트랜잭션 처리시간 동안 시스템 파손없이 정상적으로 작동됨을 가정하였다.

(그림 5-2)는 전체적으로 제한된 기법이 EMS-CS 기법 보다 우세한 것으로 분석되었다. 그 이유는 첫째는 클라이언트에서 트랜잭션이 발생하면 해당 페이지를 서버에게 요구한다. EMS-CS 기법은 서버에 있는 DPT(Dirty Page table)을 이용하여 전송하므로 DPT에 대한 로드가 많은 반면 본 기법은 PLT를 이용하므로 신속히 클라이언트에게 전송 할 수 있다. 둘째는 EMS-CS 기법은 로그일련번호와 로그 레코드 번호를 사용하므로 로드가 많은 반면 본 기법은 PLT를 사용하여 간단히 부여 할 수 있다.



(그림 5-2) 로깅 기법을 이용한 트랜잭션 처리시간

6. 결 론

본 논문에서는 클라이언트-서버 데이터베이스 환경에서 시스템 운영시 발생하는 여러 가지 문제점들을 제시하고, 효율적인 회복기법을 제안하였다. 워크스테이션의 보급이 증가하고 고속의 네트워크 통신기술의 발전으로 페이지-서버 환경의 데이터베이스 시스템이 요구됨에 따라, 기존의 기법의 회복 알고리즘의 문제점 파악하여 개선된 회복기법을 제시하고, 간단한 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.

앞으로의 연구는 본 논문에서 평가 분석한 결과를 중심으로 오버헤드를 최소하기 위한 계속적인 연구가 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., Dentist, 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, pp.641-654, 1993.

[12] M.Carey et.al., "Data Caching Tradeoffs in Client-Server DBMS Architectures," Proc. ACM SIGMOD, pp.596-609, 1991.

[13] Y. Wang and L.Rowe, "Cache Consistency and Concurrency Control in a Client/Server DBMS Architecture," Proc. ACM SIGMOD, pp.367-376, 1991.

[14] 이강우, 김형주, "클라이언트-서버 DBMS로의 ARIES 확장", 정보과학회논문지, 제21권 제4호.

조 성 제

e-mail : sjcho@nuch.ac.kr

1997년 홍익대학교 대학원 전자계산학과 (이학박사)

1995년~2000년 경주대학교 컴퓨터공학과 조교수

2000년~현재 국립 한국전통문화학교 조교수

관심분야 : 클라이언트-서버 DBMS, 트랜잭션관리, 객체지향 DBMS