論文 95-6-6-11

클라이언트-서버와 주기억 데이타베이스 시스템 환경에서의 개선된 회복 기법

正會員 林 海 喆*, 崔 義 仁**

Improved Recovery Techniques in a Client-Server and Main-Memory Database System Environment

Eui-In Choi*, Hae-Chull Lim** Regular Members

본 연구는 한국 과학 재단의 '94 핵심 전문 연구비(941-0900-060-2) 지원으로 수행되었음

要 約

이 논문에서는 클라이언트-서버와 주기억 데이타베이스 환경에서의 회복 기법을 제안하였다.

제안 기법의 특징은 기존의 기법과는 달리 서버가 주기억 데이타베이스와 배터리 백업 장치 로그를 유지하게 하여 백업과 회복시 발생하는 디스크 입출력을 최소화 할 수 있도록 하였으며, 클라이언트가 주기적으로 자신의 상태를 파악하여 서버에게 전달하므로써 서버가 클라이언트에 의해 갱신된 데이타 페이지등의 정보를 항상 파악할 수 있도록 하였고, 안전 기법을 제안하여 시스템 파손시 회복 동작의 처리가 단순하고 신속하게 처리되도록 하였다. 그리고, 기존의 퍼지 검사점 기법을 개선하여 검사점 실행 동안에도 데이타베이스의 일치성이 보장되도록 하였다. 또한, 재수행 로그가 검사점이 실시될때마다 기록되지 않으므로 검사점 실행에 따른 오버헤드를 감소시켰다.

ABSTRACT

In this paper, Improved techniques are proposed with respect to recovery and maintaining consistency of memory cache in a Client-Server and Main-Memory database environment.

The key features of our scheme are as follows : 1) Server contains only main memory databases and battery backup device log; this minimizes the number of i/o activities for backup and recovery. 2) Clients perform checkpointing; this makes the server acknowledge the dirtied page by the client and it is used by the server to recover them efficiently. 3) Restart processing is simpli-

*홍익대학교 컴퓨터공학과
Dept. of Computer Engineering, Hongik Univ.
**명지전문대학 전자계산학과
論文番號 : 95079-0221
接受日字 : 1995年 2月 21日

fied and made very fast by using proposed safety technique. 4) Database consistency is guaranteed even in the meantime of checkpointing by improving traditional Fuzzy checkpointing.

We do not require the redo-log to be flushed on every checkpoint. This reduces the checkpoint overhead and time taken for checkpoint, which is important when pages are checkpointed individually.

Ⅰ. 서 론

최근 워크스테이션의 기능이 강화되고 대용량의 데이타를 고속으로 전송하는 통신 장비의 개발로 클라이언트-서버 환경에 대한 관심이 많아짐에 따라 클라이언트-서버 환경이 주요 시스템으로 등장하고 있다. 이러한 클라이언트-서버 환경에 데이타베이스 시스템을 유지 관리하는 경우를 클라이언트-서버 DBMS라 한다. 일반적으로 클라이언트-서버 DBMS에서는 서버가 데이타베이스와 로그의 관리를 전담하고 클라이언트에서는 응용 프로그램을 실행하여 서버의 로드(load)를 줄여 줄 수 있기 때문에 최근에 많은 연구가 진척되고 있다. 그리고 실시간 시스템이나, 이동 통신과 같은 빠른 처리를 요하는 시스템의 경우에는 주어진 시간내에 트랜잭션의 빠른 처리와 응답이 필요하다. 하지만 기존의 클라이언트-서버 환경에서의 데이타베이스 시스템은 디스크에 데이타베이스를 저장하여 처리하므로써 빠른 응답을 하기 어려운 실정이다. 최근들어 이런 환경에 주기억 데이타베이스 시스템을 응용하는 연구가 많이 진행되고 있다.[14,15] 본 논문도 서버의 환경을 주기억 데이타베이스 시스템으로 구성하여 트랜잭션의 빠른 처리를 하도록 제안하였다. 그리고 이런 시스템을 운영할 경우에 발생될 수 있는 시스템 파손과 같은 경우에 대한 회복 기법을 제시하고자 한다. 클라이언트-서버 환경은 데이타 페이지에 대한 갱신이 주로 클라이언트에서 이루어지고, 서버는 데이타베이스와 로그만을 관리하며, 각각의 클라이언트는 자신의 메모리 캐쉬에 데이타베이스를 부분적으로 가지고 있으며, 한 클라이언트가 소유하고 있는 데이타베이스는 다른 클라이언트에서 동시에 보유하고 있지 않으며, 클라이언트와 클라이언트 간의 데이타베이스의 전달은 서버를 통해서 이루어진다. 그러므로 클라이언트-서버 환경에 맞는 회복 기법이 필요한 실정이다.

또한, 클라이언트-서버 DBMS를 운영할 경우에는 다음과 같은 문제점이 발생되는데 첫째 클라이언트에서 데이타 페이지가 갱신(dirty)되어졌을 때, 서버의 데이타 페이지는 갱신되지 않을 수 있고, 둘째 클라이언트가 로그 레코드를 서버에 전송하는 시기를 결정하기 어렵고, 셋째 클라이언트로부터 로그 레코드와 갱신된 데이타 페이지가 각각 서버에 따로 도착할 경우, WAL(Write Ahead Logging) 프로토콜을 유지하기가 어렵고, 마지막으로 백업 연산 동안 디스크 입출력으로 인한 서버의 오버헤드가 증가하는 등의 문제점이 있다.[6]

이 논문에서는 클라이언트 및 서버 사이트에서 시스템 파손이 발생하는 경우에 대한 빠른 회복을 위해, 1) 안전 기법을 제안하여 회복 동작 수행시 기존의 기법은 로그(log) 분석, 재수행(redo), 철회수행(undo) 세 단계로 이루어지던 것을 재수행만 실시하고, 재수행 단계를 감소시켜 기존의 기법보다 빠른 회복이 가능하도록 하였으며, 2) 클라이언트에서 갱신된 로그 레코드가 서버에 도착할 때 서버는 갱신 페이지 테이블에 해당 데이타 페이지가 갱신된 것으로 기록하여 로그 레코드와 갱신 페이지가 각기 따로 도착할 때의 문제점을 해결하였고, 3) 검사점실행(checkpointing) 측면에서는 여러 검사점 기법중 검사점 수행시 동기화 비용이 적고 트랜잭션의 처리가 중단되지 않는(퍼지 검사점 기법이 트랜잭션, 액션 일치 기법에 비해[9]) 퍼지 검사점을 개선하여(데이타베이스의 불일치가 발생한다는 단점) 데이타베이스의 일치성을 보장 할 수 있도록 하였으며 또한, 클라이언트에서도 주기적으로 자신의 상태 정보를 파악하여 전송하도록 하여 클라이언트 사이트의 파손시 서버가 빠른 회복을 하도록 하였고, 4) 서버의 데이타베이스 구성을 디스크 베이스로 하지않고 주기억 데이타베이스로 구성하여 디스크에 대한 입출력을 감소시켜 실시간 시스템과 같이 빠른 응답을 요구하는 시스템을 만족시키도록 하였다.

또한, 이 연구에서는 클라이언트-서버 간의 관계를 N:1 로 가정하여, 클라이언트-서버 사이의 갱신 동작

및, 파손시의 회복 동작에 대해서도 기술하였다.

본 논문의 구성은 2장에서는 클라이언트-서버 DBMS를 구분하는 여러가지 기준에 대하여 설명하고, 3장에서는 클라이언트-서버 환경의 대표적인 회복 기법으로 소개된 ESM-CS의 회복 기법을 기술한 뒤, 4장에서는 본 논문에서 제안하는 시스템의 구조를 설명하고, 5장에서는 제안된 갱신 동작, 회복기법 에 대해 기술하고, 6장에서는 관련 연구와의 비교를, 7장에서는 결론을 유도한다.

Ⅱ. 클라이언트-서버의 시스템 환경

클라이언트/서버 DBMS를 분류하는 기준은 크게 세가지로 나누어 볼 수 있다. 첫째는 클라이언트-서버 간의 데이타 전송 단위이고, 둘째는 클라이언트-서버 간의 질의 처리 방식에 대한 것이며, 세째는 클라이언트가 자체 캐쉬를 가지고 있느냐 하는 것이다.

클라이언트-서버 사이의 데이타 전송 단위를 기준으로 분류해 보면 전송의 단위를 화일, 페이지 또는 세그먼트, 그리고 객체나 튜플과 같이 논리적 단위의 데이타를 주고 받는 시스템으로 구분 할 수가 있다. 이 세가지 전송 방법에서 일반적으로 페이지 단위로 전송하는 것이 가장 우수한 방법으로 평가되고있다.[5]

질의의 처리 방식에 있어서도 클라이언트가 질의를 서버에게 전송하여 그 결과를 돌려받는 질의 운송(query-shipping) 시스템과 클라이언트가 질의 처리에 필요한 특정 데이타를 서버에게 요청하여 클라이언트가 질의를 처리하는 데이타 운송(data-shipping) 시스템으로 구분 될 수 있다. 질의 운송 시스템은 일반적인 중앙집중식 또는 분산 데이타베이스 시스템, 클라이언트가 메모리 캐쉬 기능이 없는 시스템, 그리고 트랜잭션 처리 구조가 비슷한 시스템에서 사용된다. 이러한 질의 운송 시스템에서는 중앙 집중식 데이타베이스 시스템을 클라이언트-서버 환경의 데이타베이스 시스템으로 전환하기가 용이하다는 점과 클라이언트와 서버간의 통신에 있어서 클라이언트가 보낸 질의에 대해 만족되는 결과만 서버가 클라이언트로 전송하면 되므로 통신 오버헤드가 적다는 장점이 있다. 그러나, 서버에 많은 오버헤드가 부과된다는 단점이 있다. 한편, 클라이언트가 서버에게 질의 처리에 필요한 특정 데이타를 요구하는 데이타 운송 시스템은 클라이언트 워크스테이션의 기억장치 이용율과 처리율을 높이고, 서버의 병목 현상을 최소화 할 수 있다는 장점이 있다. 이런 시스템은 각각의 클라이언트가 데이타 캐쉬 기능과 처리능력을 지니는 방법으로써 성능 측면에서 효율적이라고 할 수 있다.

클라이언트-서버 시스템에서의 시스템 구조는 클라이언트가 메모리 캐쉬를 가지지 않은 시스템과 클라이언트가 메모리 캐쉬나 디스크 캐쉬를 가지고 운영하는 시스템으로 구분 할 수가 있다. 클라이언트가 메모리 캐쉬를 가지지 않은 시스템은 일반적인 상용 DBMS을 말하며 데이타베이스의 관리 및 처리를 서버가 전담하는 시스템이다. 그러나 클라이언트가 메모리 캐쉬나 디스크 캐쉬를 가진 경우에는 서버의 오버헤드를 줄이고, 클라이언트와 서버 사이의 데이타 전송에 따른 비용을 절감한다는 장점이 있으나 클라이언트가 메모리 캐쉬에 대한 데이타 일치성을 유지하는데 어려움이 있다.[3,4]

본 논문에서는 클라이언트와 서버 사이의 데이타 전송은 페이지 단위로 전송하며, 질의의 처리 방식도 클라이언트가 메모리 캐쉬 기능을 갖고 질의에 필요한 특정 데이타를 서버에게 요구하여 그 데이타를 클라이언트로 전송하여 처리하는 데이타 운송 방식으로 운영하고, 클라이언트가 메모리 캐쉬만을 가지고 있다고 가정하였다.

Ⅲ. 관련 연구

이 논문에서 기본 골격으로 선정한 ESM-CS(Client-Server Exodus Storage Manager system)는 최근에 Wisconsin Univ.에서 Franklin 등이 개발한 시스템으로 클라이언트-서버 환경에서의 데이타베이스 시스템으로는 대표적인 시스템이며 다른 많은 연구에서도 인용하였다.[6] 또한, 본 논문의 데이타 전송 및 질의의 처리 방식도 이 연구에서 제안한 시스템과 유사한 환경을 갖기 때문에 본 연구를 위한 기본 모델로 채택하였다. ESM-CS는 사용자의 응용 프로그램과 서버에 연결된 클라이언트들로 구성된다. 각각의 클라이언트들은 독립적으로 수행되는 하나의 프로세스로서 자신의 메모리 캐쉬와 록 캐쉬를 가지고 있으며, 한번에 하나의 트랜잭션을 처리한다. 서버는 전체 데이타베이스와 로그를 관리하며, 록 관리 및 시스템 파손시 회복 동작을 제공한다. ESM-CS 시스템의 구조는 그림 1과 같다.

ESM-CS는 ARIES(Algorithm for Recovery

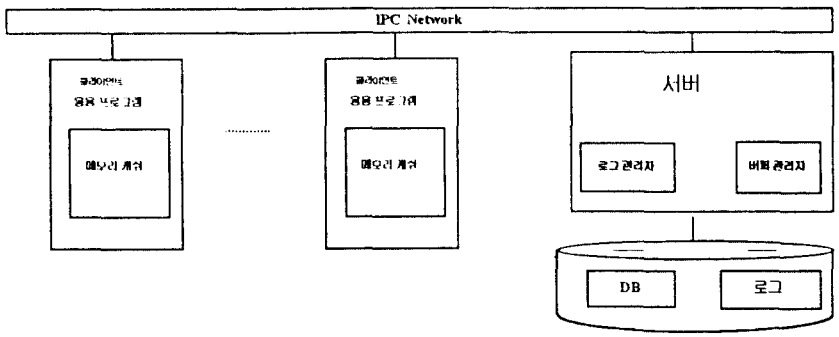

그림 1. ESM-CS 구조
Fig. 1. ESM-CS structure

and Isolation Exploiting Semantics)라는 디스크 베이스 회복 기법을 클라이언트-서버 환경에 맞게 개선하였고, WAL 프로토콜을 보장하는 회복 기법이다.[12] WAL 프로토콜이란 갱신된 데이타 페이지가 디스크 데이타베이스 사본에 기록되기 전에 갱신에 대한 로그가 안정된 로그에 먼저 기록되는 것을 보장하는 기법을 말한다. WAL 프로토콜을 구현하기 위해 ESM-CS는 클라언트가 갱신 연산에 따른 로그를 갱신된 데이타 페이지보다 먼저 서버에 전송하여 기록하도록 하였다.

ESM-CS 환경에서 WAL 프로토콜을 사용하려면 클라이언트에서 갱신된 데이타 페이지를 서버로 전송할 때 데이타 페이지 변경에 따라 생성된 로그 레코드를 먼저 서버로 전송해야 하나 클라이언트에서 생성된 로그 레코드의 LSN이 어떤 클라이언트에서 생성된 것인지 서버는 알 수 없게 된다. 따라서, 클라이언트에서 LRC(Log Record Counter, 한 페이지에 해당하는 모든 레코드에 대해 유일한 값을 갖고 단조 증가하는 성질을 갖는다)를 이용하여 서버에게 데이타 페이지를 전송 할 때 LRC와 pageLRC를 같이 전송하여 이 문제를 해결하였고, ARIES가 무조건 철회를 실시하는데 비하여 조건부 철회(conditional undo)를 실시하여 무조건 철회에서 발생하는 데이타베이스의 일치성 문제를 개선하였다. 또한, 재수행은 클라이언트가 기록하고 있는 LRC와 갱신된 데이타 페이지가 서버로 전송될 때 보내지는 recLSN을 이용하여 실시한다.

그러나 ESM-CS는 서버가 갱신된 데이타 페이지가 도착해야만 DPT에 기록하기 때문에 재수행시에 문제가 발생되어 데이타베이스의 일치성을 보장 못하는 문제점이 있다. 만약 클라이언트의 트랜잭션이 해당 데이타 페이지에 대한 갱신 연산을 수행하고, 로그 레코드를 먼저 서버로 전송했을 경우를 생각해보자. 클라이언트는 그 후 해당 갱신 연산 결과가 반영된 데이타 페이지를 서버에게 전송하고, 서버는 검사점을 실시하고 있다면 로그에는 기록이 반영되지만 데이타베이스에는 기록이 이루어지지 않아 재수행이 필요한 경우 이를 수행 할 수 없다는 문제점이 발생한다.

이런 문제점외에도 클라이언트-서버 환경에서 서버의 시스템 구성이 디스크 베이스 시스템으로 구성되어 갱신된 데이타 페이지의 처리 및 로그의 처리로 인하여 많은 디스크 입출력을 증가시켜 서버의 오버헤드 및 병목 현상을 증가시키고 있다.

이 논문에서는 이런 문제점의 해결을 위해 서버의 시스템 구성을 주 기억 장치로, 로그 처리를 디스크 로그의 사용을 배제하고 배터리 백업 로그를 사용하므로써 많은 디스크 입출력을 배제하여 서버의 병목 현상과 오버헤드를 감소시킬 수 있도록 하였다.

Ⅳ. 시스템 구조

이 논문에서 제안하는 시스템 구조는 ESM-CS와는 달리 디스크 베이스가 아닌 데이타베이스를 서버의 주기억 장치에 저장하고, 로그를 디스크에 유지하지 않고 배터리 백업 로그를 이용하므로써 디스크 입출력을 줄이고, 클라이언트가 메모리 캐쉬를 사용하도록 하여 서버

의 부하를 감소시키며, 시스템 고장시 회복 속도를 개선하는 방향에 중점을 두었다. 시스템 고장시에 발생하는 로그의 손실 문제는 배터리 백업 로그와 클라이언트에서의 상태 정보 전송을 통하여 해결하였다. 또한, 로그는 클라이언트에서 발생하는 모든 갱신 연산에 대해 서버가 클라이언트로부터 해당 로그 레코드를 전송받아 배터리 백업 저장 장치에 있는 로그에 기록하도록 하였다. 그리고 데이타베이스 백업시 데이타베이스 내의 모든 데이타를 동시에 기록하지 않고 페이지 단위로 데이타를 기록하도록 구성하여 더욱 효율적인 운영이 이루어지도록 설계하였다. 본 논문에서 제안하는 시스템의 구성도는 그림 2와 같다.

4.1 서버의 구성

4.1.1 서버의 주기억 장치 구성

서버의 주기억 장치에는 전체 데이타베이스와 배터리 백업 로그, 각각의 트랜잭션에 대한 개인 로그를 저장하고, 배터리 백업 로그에는 주기억 장치의 데이타베이스에 변경이 가해진 재수행 로그 레코드를 유지한다. 그러므로 시스템 파손시 로그 분석 및 재수행, 철회수행으로 이루어지는 로그 처리가 철회수행은 처리하지 않고, 로그 분석 시간은 감소되어 빠른 회복이 이루어진다. 또한 서버는 갱신된 데이타 페이지 테이블을 이용하여 운영한다. 로그는 개인 로그와 배터리 백업 로그를 사용하여 관리하고, 재수행(redo)과 철회수행(undo)으로 나뉘어서 개인 로그에 관리한다. 단, 로그 버퍼에는 정상완료된 레코드 만을 기록한다. 철회수행 로그는 사용자에 의해 수행 중이던 트랜잭션을 철회하고자 할 때만 사용하고 배터리 백업 로그에는 기록하지 않는다.

4.1.2 서버의 디스크 구성

서버의 디스크에는 데이타베이스 사본과 검사점 실행시의 정보를 유지한다. 데이타베이스는 맵 테이블을 이용하여 검사점 실행시의 정보와 갱신된 데이타베이스의 내용을 기록한다. 검사점 실행시의 정보를 유지하는 디스크 데이타베이스는 세도우(shadow) 기법과 핑퐁(ping-pong) 기법[2,7]을 변형하여 포인터를 이용하여 검사점이 실행될 때마다 검사점 위치 포인터를 이용해서 데이타베이스 내의 변경된 데이타 페이지의 위치를 변경해 주고 주기억 장치에서는 갱신 비트를 사용하여 갱신된 데이타 페이지만 디스크 데이타베이스에 기록하도록 하였다. 그러므로 핑퐁 기법과 같이 두 개의 데이타베이스 사본을 이용하여 운영하는 것 보다 기억 공간의 낭비를 줄이도록 하였다.

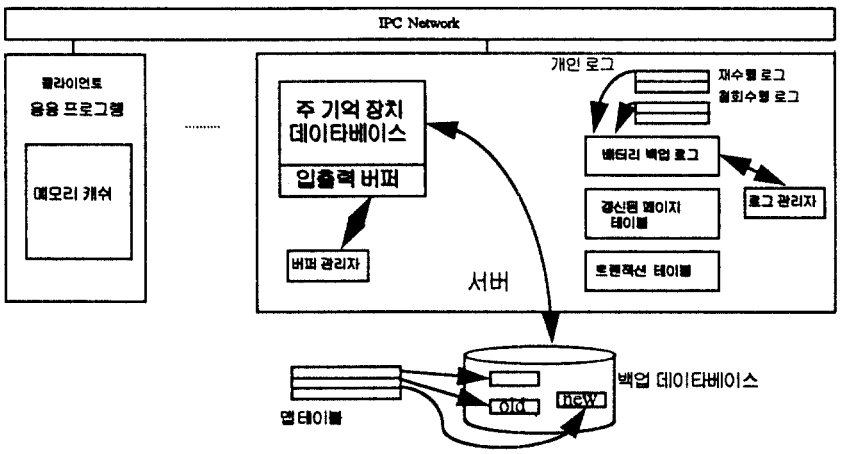

그림 2. 시스템 구성도
Fig. 2. System structure

4.2 클라이언트의 구성

클라이언트는 서버와는 달리 메모리 캐쉬를 보유하고 있는데 이 메모리 캐쉬는 트랜잭션 처리시 필요한 데이타 페이지를 서버로부터 전송받아 갱신 연산을 실행한 후 갱신 연산이 일어난 페이지를 보관하는 역활을 한다. 그리고 클라이언트가 자신에 대한 상태 정보를 메모리 캐쉬에 저장하고, 그 정보를 이용하여 주기적으로 서버에게 알려준다.

4.3 서버에서의 검사점 실행 및 회복 동작

본 논문에서 제안한 검사점 실행은 기존의 퍼지 검사점의 문제점을 개선하여 실행하였다. 개선된 퍼지 검사점은 기존의 검사점 기법인 트랜잭션 일치와 액션 일치 검시점 기법이 검사점 실행을 위해 동기화의 비용이 요구되고, 트랜잭션의 처리가 일시 중단되는 등의 문제점이 발생되고 있으나 퍼지 검사점 기법은 일시 중단 문제나 동기화에 따른 비용 문제를 해결하였다.[10,11] 그러나 퍼지 검사점 기법은 위 두 기법의 문제점은 해결하였으나 트랜잭션 처리의 원자성을 보장하지 못해 검사점 실행 이후 데이타베이스의 일치성을 유지하지 못한다는 단점이 발생하므로 본 연구에서는 이러한 단점을 해결하기 위해서 검사점 실행이 실시될 때 처리가 완료되지 않은 트랜잭션은 곧 바로 디스크에 기록하지 않고 주기억 장치의 버퍼에 일시 저장했다가 트랜잭션 처리가 완료될 때 디스크 데이타베이스에 기록하므로써 데이타베이스의 일치성을 보장하도록 하였다. 검사점실행은 배터리 백업 로그에 일정한 양의 로그 레코드가 기록되거나 일정 시간이 경과하면 검사점실행을 실시해 데이타베이스 백업을 발생시키는 방법으로 실행한다. 또한 데이타베이스를 페이지로 나누어 각 페이지 대해 변경 유무를 조사하여 변경된 페이지에 대해서만 디스크에 기록한다. 따라서, 검사점 실행시 모든 데이타베이스에 대한 접근이 중지되는 것이 아니고, 디스크에 기록 할 필요가 없는 페이지에 대해서는 트랜잭션 처리가 가능하다.

검사점실행 측면에서는 서버와 마찬가지로 클라이언트가 자신의 상태 정보를 파악하여 서버에게 주기적으로 자신의 정보를 전송하도록 하여 클라이언트가 파손될 경우에 서버가 클라이언트에 대한 정보를 파악하고 있어 빠른 회복을 할 수 있도록 하였다. 또한, 메모리 캐쉬의 상태 즉, 클라이언트가 갱신한 데이타 페이지 정보와 현재 활동 중인 트랜잭션의 상태를 기록하여 검사점 실행 후 그에 대한 정보를 서버에게 전송한다. 서버는 모든 클라이언트에 대한 가장 최신의 정보를 유지하고 있으며 자신이 파손될 경우에 대비해 주기적으로 검사점을 실행하는데 DPT(Dirty Page Table)의 상태, 트랜잭션 테이블에 대한 정보를 기록한다.

시스템이 파손될 경우에 회복 동작에서 재수행 처리 단계를 감소시키기 위해 안전 기법을 제안하였다. 안전 기법은 우선 다음과 같은 가정을 필요로 한다. 정상 처리동안 주기억 데이타베이스는 디스크 데이타베이스에 갱신 결과를 주기적으로 전달하는데 이때 디스크 데이타베이스 페이지와 정상 완료된 데이타베이스 페이지가 같다면 그 페이지를 안전하다고 한다. 역으로 그 페이지의 내용이 같지 않으면 그 페이지를 불안전하다고한다. 이때 트랜잭션이 바로 직전에 정상 완료된 페이지는 불안전한 페이지로 볼 수 있다. 또한, 페이지가 갱신된 뒤 디스크 데이타베이스에 기록이 이루어지면 갱신된 페이지는 안전하다고 한다. 본 기법은 각각의 데이타베이스 페이지를 안전 상태로 유지하므로써 데이타베이스의 일치성을 보장하도록 하였다. 이를 위해 시스템 파손시 회복 동작이 이루어지는 동안 불안전한 페이지에 대한 접근은 그 페이지에 재수행이 실행되어 그 페이지가 일치성이 보장될 때 까지 중지하도록 하며 안전한 페이지는 바로 재적재를 실시하는 방법이다.

Ⅴ. 갱신 동작 및 회복 알고리즘

이 절에서는 클라이언트가 해당 데이타 페이지를 갱신할 때 서버와 클라이언트간의 갱신 동작, 클라이언트나 서버 사이트 파손시 회복 동작에 대하여 각각의 경우로 나누어 기술하겠다.

5.1 클라이언트-서버 간의 갱신 동작

클라이언트-서버는 해당 데이타 페이지에 대한 갱신 동작을 다음과 같이 실시한다. 아래의 동작은 모든 갱신 연산 동작이 정상완료된다는 가정하에서 실시된다. 알고리즘은 다음과 같다.

〔클라이언트에서의 갱신 동작〕

1. 갱신에 필요한 해당 페이지가 메모리 캐쉬에 존재하는 가를 조사한다.
2. 없으면 서버에게 요청한다.

3. 요청한 페이지가 도착하거나 메모리 캐쉬에 존재하면 갱신 연산을 실시한다.
4. 갱신에 해당하는 로그 레코드를 생성하여 클라이언트의 LRC 필드 값과 갱신된 페이지가 포함된 위치의 page_LSN 필드 값에 현재 Client_LSN 값을 기록한다.
5. 클라이언트는 생성된 로그 레코드를 Client_LSN을 이용하여 서버에게 전송한다.

여기서 Client_LSN은 각각의 클라이언트에게 주어진다.

〔서버에서의 갱신 동작〕

1. 클라이언트로부터 전송받은 로그 레코드를 Client_LSN을 통하여 로그 화일에 기록한다.
2. 해당 트랜잭션 테이블을 트랜잭션 식별자(TransID)를 이용하여 조사한다.
   2.1 테이블에 없으면 트랜잭션 식별자를 이용하여 등록한 뒤 LastLSN(가장 최근 것의 LSN)과 UndoNxtLSN을 로그 레코드의 LSN에 기록한 뒤, PrevLSN 값을 0으로 초기화한다.
   2.2 테이블에 있으면 PrevLSN에 LastLSN을 기록한뒤, LastLSN과 UndoNxtLSN에 LSN을 기록한다.
3. 로그 레코드의 pageID를 이용하여 DPT를 검색한다.
   3.1 pageID가 존재하면 갱신 연산이 정상완료된 것으로 기록한다.
   3.2 pageID가 존재하지 않으면 이를 등록한 후 갱신 연산이 정상완료된 것으로 기록한다.

5.2 클라이언트-서버의 파손시 회복 기법

〔클라이언트 파손시의 동작〕

1. 서버는 모든 클라이언트에 대해 가장 최근의 검사점 실행 상태를 유지하고 있다.(클라이언트는 주기적으로 자신의 상태를 파악하여 서버에 아직 기록이 안된 갱신된 데이타 페이지와 현재 활동중인 트랜잭션의 정보를 서버에게 전송한다)
2. 서버는 해당 클라이언트의 상태 정보를 보고 처리를 시작한다.
3. 서버는 자신의 로그를 조사한다.
4. 로그가 정상완료 레코드를 포함할 경우에만 재수행을 실시한다.
5. 서버가 주어진 시간안에 반응을 못하면 서버의 파손으로 간주하여 서버가 회복될 때 까지 기다린다
   (이 경우 클라이언트는 메모리 캐쉬에 있는 데이타 페이지에 대해 록(lock)을 해제하는 것이 효율적이다)

〔서버 파손시의 동작〕

1. 클라이언트가 수행중인 모든 트랜잭션의 실행을 중지하는 메시지를 전송한다.
2. 클라이언트는 자신의 메모리 캐쉬에 있는 내용을 삭제한다.
3. 서버는 모든 클라이언트에 대해 록을 걸어 접근을 금지시킨다.
4. 모든 클라이언트가 상태 정보 파악후 가지고 있는 상태 정보를 전송하도록 요청한다.
5. 서버의 데이타베이스가 주기억 장치이므로 디스크 데이타베이스 사본을 재적재한다.
6. 각 클라이언트가 보내온 상태 정보와 배터리 백업 로그를 이용하여 분석한 뒤 재수행을 실시한다.

Ⅵ. 관련 연구와의 비교

6.1 성능 분석

본 논문에서 제안한 개선된 검사점 수행의 성능을 평가하기 위하여 SLAM Ⅱ[1]와 〔8,11〕을 이용하여 시뮬레이션을 수행하였다. 시뮬레이션에서는 디스크 데이타베이스 사본에 백업시 발생하는 CPU 오버헤드를 기존의 검사점실행과 제안된 검사점실행을 비교하므로써 행해졌다.

6.1.1 실험 환경

시뮬레이션 도구로서는 SLAM Ⅱ 언어를 사용하였으며[1], 주요 매개변수는 표 1에 나타냈으며 〔8,11〕의 것을 주로 이용하였다.

6.1.2 비교 분석

트랜잭션 로드에 따라 부가되는 CPU 오버헤드 측면에서 시뮬레이션을 수행한 결과 그림 3과 같이 본 논문에서 제안된 기법이 트랜잭션 일치 검사점보다는 우수한

표 1. 주요 매개 변수
Table 1. parameters

매개변수	값	매개변수	값
로그페이지 크기	1024 words	데이타베이스크기	256 Mwords
전송시간	7.46ms	레코드크기	32 words
주기억장치버퍼복사시간	1.1738ms	세그먼트크기	8192 words
페이지 접근시간	21.76ms	페이지크기	1800
평균탐색시간	16ms	도착율	1000 TRs/sec.
평균지연시간	8.3ms	갱신연산수	5 rec/TRs
입출력 오버헤드	1000 instructions	워드당 바이트	4 bytes
입출력 지연시간	0.03 sec.	트랜잭션실패율	0.05

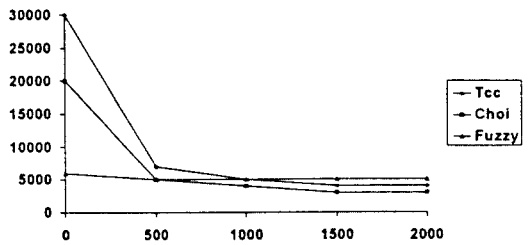

그림 3. 트랜잭션 로드에 따른 cpu 오버헤드
Fig. 3. CPU Overhead with transaction load

것으로 나타났다. 여기에서 퍼지 검사점 기법과 본 논문의 기법은 디스크에 백업용 사본을 하나를 유지하고, 트랜잭션 일치 검사점은 디스크에 백업용 사본을 두 개를 유지하는 핑퐁 기법을 이용하였다. 트랜잭션 일치 검사점의 경우 핑퐁 기법을 사용하는 이유는 검사점 수행시 트랜잭션의 처리가 일시 중단되기 때문에 데이타베이스의 일치성과 재수행, 철회수행을 위해서이다.

6.2 관련 연구와의 비교 분석

본 장에서는 ESM-CS, ARIES/CSA[13], 그리고 본 논문에서 제시한 회복 기법을 비교하였다. 이 논문에서 제안된 기법은 ESM-CS, ARIES/CSA와는 서버의 구성, 클라이언트 간의 페이지 전송, 로그 처리등 시스템 환경의 차이가 있다. 그러나 이 연구에서는 두 시스템과는 다른 측면에서 회복 기법을 제시하였고 다음과 같은 특징이 있다.

1. 서버의 시스템 구성을 주기억 장치 데이타베이스 시스템으로 제안하였다.
2. 로그의 처리도 배터리 백업 로그를 이용하였다.
3. 클라이언트도 자신의 상태를 파악하도록 하였다.
4. 로그 레코드가 서버에 도착시 해당 데이타 페이지가 갱신 된 것으로 간주한다.
5. LRC를 보완하기 위해 Client_LSN을 사용하였다.

따라서, 본 연구와 ESM-CS, ARIES/CSA를 비교하면 표 2과 같다. 또한 비교에 의하면 본 연구는 다음과 같은 장점을 갖는다.

1. 클라이언트가 주기적으로 최근 상태 정보를 서버에 전송하여 서버가 해당 클라이언트의 최근 상태 정보를 유지하므로써 클라이언트 사이트의 파손시 빠른 회복이 이루어진다.
2. 로그 레코드가 서버에 도착할 때 데이타가 갱신된 것으로 간주하므로써 재수행 이상(anomaly)이 발생되지않는다.
3. 클라이언트가 로그 레코드를 생성할 때 생성된 로그 레코드의 LSN을 서버의 로그에 전송해야 하고, 여러 클라이언트가 생성한 로그 레코드를 서버에 동시에 전송하면 서버와의 많은 통신 비용과 지연 시간이 발생하여 LSN을 알기 어렵다. 이를 해결하기 위해서 Client_LSN을 제안하여 성능을 향상시켰다. Client_LSN은 각 클라이언트가 해당 데이타 페이지를 갱신할때 자신의 pageLSN에 같

표 2. 비교 분석
Table 2. comparative analysis

	ESM-CS	ARIES/CSA	이 논문의 방법
서버의 환경	디스크 데이타베이스	디스크 데이타베이스	주 기억 데이타베이스
로그 구성	디스크 로그	디스크 로그	배터리 백업 로그
로그 처리 방법	재수행, 철회수행	재수행, 철회수행	재수행만 처리
데이타 페이지 갱신 여부	로그와 데이타 페이지 도착시	로그 도착시	로그도착시
로그 처리로 인한 오버헤드	크다	중간	적다
회복 기법	ARIES 확장	ARIES 확장	주기억 데이타베이스 기법 확장
회복시 오버헤드	중간	중간	적다
클라이언트 구분 여부	메시지 교환	전역 록 관리 LSN 변형	Client_LSN 제안
클라이언트 구분에 따른 오버헤드	통신비용증가	LRC 비교 비용 증가	로그 크기 증가
완료 확인	완료 갱신 페이지 리스트 이용	commit_LSN 이용	DTP에 등록 확인
클라이언트 LSN	pageLRC사용	Local_Max_LSN 사용	Clint_LSN 사용

이 기록하므로써 갱신된 기록을 서버에 전송시 자신의 Client_LSN과 로그 레코드를 비교하여 작은 값을 갖는 Client_LSN을 서버에 전송하는 방법이다.

4. 검사점 실행에서는 퍼지 검사점 기법을 개선하여 제시하므로써 검사점 실행에 따른 비용을 감소시키고, 데이타 베이스의 일치성을 보장하였으며 검사점 실행시 트랜잭션의 처리가 중단되지 않기 때문에 트랜잭션 처리율을 높였다.
5. 회복시에도 재수행 처리에 있어 안전 기법을 이용하여 재수행 처리 횟 수를 줄임으로써 다른 회복 기법에 비하여 빠른 회복이 이루어 지도록 하였다.

본 논문에서는 클라이언트-서버 DBMS에서 시스템 운영시 발생하는 여러가지 문제점들을 제시하고, 효율적인 회복 기법 및 운영 기법을 제안하였다. 본 연구는 서버의 시스템 구성을 주 기억 장치로 로그의 처리도 배터리 백업 장치를 이용하므로써 서버의 오버헤드 감소와 디스크 입출력을 감소시켰다. 기존의 기법들이 단순히 시스템 파손시에 로깅을 이용하여 데이타베이스를 일치된 상태로 복구하는 방법만 제시한 것에 비하여 본 연구는 클라이언트와 서버 사이트의 파손시에 대한 회복 동작을 기술하였다. 그리고 클라이언트가 서버에게 주기적으로 자신의 상태 정보를 전송하므로써 클라이언트의 파손시 빠른 회복이 이루어지도록 하였다.

Ⅶ. 결론 및 추후 연구 방향

현재까지는 클라이언트-서버 환경에서의 회복 기법이 많은 연구가 이루어지지 않고 있는 실정에서 ARIES 회복 기법을 클라이언트-서버 환경에 맞게 적용한 [6,13]을 통해 본 연구와 비교를 하였다.

앞으로의 연구는 본 논문에서 제시한 기법에서 발생될 수 있는 오버헤드를 최소로 하기 위한 계속적인 연구와 클라이언트-서버 DBMS 환경에서 본 연구에서 제안한 기법을 적용시 발생되는 장점의 정당성을 위해 시뮬레이션 기법을 이용하여 본 연구의 효율성을 입증하기 위해 ESM-CS, ARIES/CSA와의 성능 분석을 실시할 것이다.

참고문헌

1. A.Alan, B.Pritsker, "Introduction to Simulation and SLAM Ⅱ," A Halsted Press Book, John Wiley & Sons, Third Edition, 1986.

2. Bernstein, P. A., Hadzilacos, V., and Goodman, N., "Concurrent Control and Recovery in Database Systems," Addison-Wesley, 1987.
3. A.Delis and N.Roussopouls, "Performance Comparison of Three Modern DBMS Architectures," IEEE TSE, 19, 12, pp.120-138, 1993.
4. A.Delis and N.Roussopouls, "Modern Client-Server DBMS Architectures," Proc. ACM SIGMOD RECORD, 20, 3, Sep. 1991.
5. D.DeWitt et. al., "A Study of three Alternative Workstation-Server Architectures for Object Oriented Database Systems," Proc. VLDB, pp.107-121, 1990.
6. M.Franklin et. al., "Crash Recovery in Client-Server EXODUS," Proc. ACM SIGMOD, pp.165-174, 1992.
7. J. Gray, & Reuter, A., Transaction Processing : Concepts and techniques, Morgan Kaufmann, San Maeteo, 1993.
8. Le Gruenwald & M. H. Eich, "Simulation of Main Memory Database Recovery," SIMULATION, January. 1993.
9. T. Haerder & A. Reuter, "Principles of Transaction-Oriented Database Recovery," ACM Computing Surveys 15, 4, December. 1983.
10. Robert B. Hagmann, "A Crash Recovery Scheme for a Memery-Resident Database System," IEEE Trans on Com, Vol. C-35, No. 9, December. 1986.
11. Kenneth Salem & Hector Garcia-Molina, "Checkpointing Memory-Resident Databases," IEEE Data Eng., 1989.
12. C. Mohan & Don Haderle, et al, "ARIES : A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," ACM TODS, Vol. 17, No. 1, March. 1992.
13. C.Mohan and I.Narang, "ARIES/CSA: A Method For Database Recovery in Client-Server Architectures," Proc., ACM SIGMOD., pp.55-66, 1994.
14. M. Singhal, "Issues and Approaches to Design of Real-Time Database Systems," ACM SIGMOD RECORD, Vol. 17, No. 1, 1988.
15. S. H. Son, "Real-Time Database Systems: Issues and Approaches," ACM SIGMOD RECORD, Vol. 17, No. 1, 1988.
16. 최의인, 임해철, "주기억 장치 데이타베이스 시스템에서의 효율적인 회복 기법," '95 동계 데이타베이스 학술대회 발표 논문집, 3, 1995.
17. 최의인, 임해철, "클라이언트-서버 환경에서의 개선된 회복 기법," 1994년 한국 통신학회 추계 종합학술 발표회 논문집 13권 2호.

林 海 喆(Hae-Chull Lim) 정회원

1976년 : 서울대학교 계산통계학과(이학사)
1978년 : 한국과학기술원 전자계산학과(이학석사)
1988년 : 서울대학교 컴퓨터공학과(공학박사)
1978년~1981년 : 현대엔지니어링
1989년~1990년 : 미국 플로리다 대학 방문교수
1981년~현재 : 홍익대학교 컴퓨터공학과 교수
※주관심 분야 : 객체지향 DB, 실시간 DB, GIS, 분산 DB

崔 義 仁(Eui-In Choi) 정회원

1982년 : 한남대학교 계산통계학(이학사)
1984년 : 홍익대학교 전자계산학과(이학석사)
1995년 : 홍익대학교 전자계산학과(이학박사)
1985년~1988년 : 공군 장교 근무
1992년~현재 : 명지전문대학 전자계산학과 조교수
※주관심 분야 : 실시간 DB, 클라이언트/서버 DBMS, 객체지향 DB