주기억 장치 데이타베이스 시스템을 위한 디스크 그룹 완료 프로토콜

(A Disk Group Commit Protocol for Main-Memory Database Systems)

이 인 선 † 염 헌 영 ††
(In-Seon Lee) (Heon-Young Yeom)

요 약 주기억 장치 데이타베이스(Main-Memory DataBase : MMDB) 시스템은 데이타의 모든 부분이 주기억 장치에 상주하는 데이타베이스 시스템으로 트랜잭션의 연산 작업중 데이타로 인한 디스크 입출력이 발생하지 않아 기존의 디스크 기반 데이타베이스 시스템에 비해 월등히 향상된 시스템 성능을 가진다. 이러한 MMDB시스템에서 트랜잭션 완료를 위한 디스크 로깅은 필수 불가결한 요소인 한편 트랜잭션 수행 과정중 유일한 디스크 작업이 되어 트랜잭션 전체 수행 시간의 많은 부분을 차지하게 되므로 시스템 전체 성능을 위해서는 완료 과정의 개선이 아주 중요한 연구 과제가 된다. 로깅 과정의 개선책으로는 여러 방안이 제안되고 있으며, 어떠한 하드웨어를 가정하지 않는 완료과정 개선책중 대표적인 것으로 선-완료(pre-commit)프로토콜과 그룹 완료(group commit) 프로토콜을 들 수 있다. 그러나, MMDB시스템에 이 프로토콜들을 적용하여 성능 변화를 분석한 연구는 아직까지 미미한 실정이다. 본 연구에서는 그룹 완료 프로토콜을 자료 경쟁 수준을 고려하지 않고 MMDB시스템에 적용할 때 교착 상태가 발생 가능함을 밝히고, 실시간으로 다양하게 변하는 자료 경쟁 수준을 가지는 MMDB시스템에 적합한 그룹 완료 프로토콜인 디스크 그룹 프로토콜 방식을 제안하였다. 그리고, 실질적이고 구체적인 모의실험을 실시하여 그룹 완료 프로토콜은 MMDB시스템의 성능을 위해 효과적인 개선 방식이며, 본 논문에서 제안한 디스크 그룹 완료 프로토콜은 시스템의 자료 경쟁 수준을 반영할 필요없이 항상 우수한 성능을 가짐을 보였다. 또한 선-완료프로토콜은 단독으로 쓰일 때는 시스템 성능이 개선되지 않으며, 그룹 완료 프로토콜의 보조 수단으로 적용될 때에만 시스템 성능 개선에 효과적임을 밝혔다.

키워드 : 주기억 장치 데이타베이스 시스템, 완료프로토콜, 그룹 완료, 선-완료

Abstract Main-Memory DataBase(MMDB) system where all the data reside on the main memory shows tremendous performance boost since it does not need any disk access during the transaction processing. Since MMDB still needs disk logging for transaction commit, it has become another bottleneck for the transaction throughput and the commit protocol should be examined carefully. There have been several attempts to reduce the logging overhead. The pre-commit and group commit are two well known techniques which do not require additional hardware. However, there has not been any research to analyze their effect on MMDB system. In this paper, we identify the possibility of deadlock resulting from the group commit and propose the disk group commit protocol which can be readily deployed. Using extensive simulation, we have shown that the group commit is effective on improving the MMDB transaction performance and the proposed disk group commit almost always outperform carefully tuned group commit. Also, we note that the pre-commit does not have any effect when used alone but shows some improvement if used in conjunction with the group commit.

Key words : main-memory database system, commit protocol, group commit, pre-commit

† 비 회 원 : 신구대학 컴퓨터정보처리과 교수
inseon@shingu.ac.kr
†† 종신회원 : 서울대학교 전기및컴퓨터공학부 교수
yeom@cse.snu.ac.kr
논문접수 : 2003년 10월 28일
심사완료 : 2004년 6월 25일

1. 서 론

기존의 디스크 기반 데이타베이스(Disk Resident DataBase : DRDB)시스템은 데이타베이스 원본이 디스크에 있으며, 트랜잭션 처리에 필요한 데이타만을 주기억 장치로 가져와 작업을 한다. 이에 반해 모든 데이타

가 주기억 장치에 상주하는 주기억 장치 데이타베이스(Main-Memory Data Base : MMDB)시스템은 트랜잭션의 연산 수행시 데이타를 위한 디스크 입출력이 발생하지 않아 DRDB시스템보다 단축된 응답시간과 월등히 향상된 트랜잭션 처리능력을 가지게 된다[1]. 최근 RAM이 급속도로 대용량화되는데 비해 가격은 저렴해지고 있어 MMDB시스템의 상용화가 이루어지고 있으며[2-4], 한층 더 MMDB 시스템의 성능을 향상시키고자 하는 여러 논의가 한창 진행중이다. MMDB시스템의 성능을 향상시키기 위해서는 평상시 트랜잭션 처리 과정에서 유일한 디스크 출력 작업인 로깅(logging)과정을 살펴보아야 할 것이다.

데이타베이스 시스템에서는 완료된 트랜잭션의 지속성을 유지하기 위하여 갱신에 관한 로그를 반드시 안전한 저장 장치에 옮겨놓은 후에 트랜잭션이 완료되었다고 사용자에게 알려야 한다(Redo-rule:[5]). 그러나, [6]에서 알 수 있듯이 DRDB시스템에서도 전체 트랜잭션 수행 시간중 대부분을 차지하는 것은 CPU처리가 아닌 디스크에 대한 입출력 작업이며, DRDB시스템에 비해 데이타에 대한 디스크 입출력이 전혀 없는 MMDB시스템에서는 필수적이면서도 유일한 디스크 출력 작업인 로깅이 전체 트랜잭션 수행시간의 많은 부분을 차지하게 되어 시스템 성능의 병목으로 작용한다.

로그 정보의 디스크 출력에 대한 해결책으로 많은 방식들이 시도되고 있다. 이러한 시도로는 크게 주기억 장치의 휘발성이라는 단점을 없애려는 방향과 로깅 과정에서의 성능을 개선시키고자 하는 방향으로 나눌 수 있다.

전자의 방안중 하드웨어적인 해결책으로는 플래쉬 메모리와 같은 비휘발성 메모리를 사용하여 로그 정보를 저장하거나[7], 컴퓨터에 전원이 끊어졌을 경우 주기억 장치의 내용을 모두 디스크에 저장할 동안 전원을 공급시켜주는 UPS를 부착하는 것[8]들이 있다. 그러나 이 방식들은 시스템의 하드웨어적인 에러에 대해서만 로깅 정보를 보호할 수 있다는 한계가 있다. 이런 한계에 대한 새로운 시도로 운영체제가 고장이 나더라도 주기억 장치의 일부분을 마치 디스크처럼 안전하게 사용하는 방식을 고안하여[9] 데이타베이스 시스템의 버퍼 캐쉬로 사용하는 디자인을 제시한 연구가 있다[10]. 그러나, 이 방식은 기존의 운영체제와 DBMS의 디자인에 변경을 가해야 하며, 이로 인한 시스템 성능의 저하를 가져올 수 있다.

후자의 로깅 정보의 디스크 출력 과정의 성능 개선안으로는 선-완료(Pre-commit) 프로토콜과 그룹 완료(Group commit) 프로토콜을 들 수 있다. 선-완료 프로토콜은 갱신에 관한 로그 정보가 디스크로 출력되기를 기다리지 않고 연산 작업이 모두 끝난 직후에 바로 그 트랜잭션이 보유한 로크들을 해제하는 것으로 선-완료 프로토콜을 수행하는 트랜잭션의 응답시간에는 변화가 없지만 이 트랜잭션이 보유한 로크를 기다리는 다른 트랜잭션의 로크 대기 시간을 감소시킴으로써 전체적으로 시스템 성능을 향상시킬 수 있는 장점이 있다[11,12].

그룹 완료 프로토콜은 개별적으로 트랜잭션의 로그 정보를 디스크로 출력하지 않고 주기억 장치의 전역 로그 버퍼에 여러 트랜잭션들의 로그 정보를 축적하였다가, 예를 들어 전역 로그 버퍼가 가득 찼을 때 이를 한꺼번에 디스크에 출력함으로써 시스템 전체적으로 물리적인 디스크 출력 횟수를 감소시키고자 하는 방안이다[12]. 이 두 프로토콜은 특정 하드웨어를 부착하거나, 시스템을 크게 변경하지 않고서도 바로 적용할 수 있다는 장점이 있다.

MMDB시스템에서의 로깅 과정 개선책에 대한 기존의 연구를 살펴보면 다음과 같다. MMDB가 논의되던 초창기 논문들에서는 특별한 하드웨어의 부착을 가정하여 로깅 정보를 이 하드웨어에 기록하는 것으로 문제를 해결하였다[13-18]. 이후 특별한 하드웨어를 가정하지 않을 경우 많은 시스템에서 선-완료프로토콜을 채택하였으며[11,14,19,20], 그룹 완료프로토콜에 대해서는 성능에 대한 아무런 선행되는 연구도 없이 막연히 성능에 도움을 줄 것이라는 추측으로 가정하거나[14,19-21], 또는 그룹 완료 프로토콜이 시스템 전체적인 성능엔 도움을 줄 수 있으나, 개별 트랜잭션들의 응답시간을 늦추는 역효과가 있어 비용 측면에서 좋지 않을 것이라고 언급하기만 하였을 뿐[10,11], 현재까지 이 프로토콜들을 MMDB시스템에 적용하여 구체적이고 실질적인 성능 분석이 이루어지지 않고 있다.

이 시점에서 디스크 출력이 전체 시스템의 커다란 과부하가 되는 MMDB시스템에 여러 완료 개선 프로토콜들을 적용할 경우 어떤 문제점이 발생할 수 있는 지에 대한 상세한 고찰이 먼저 이루어져야 하며, 또한 완료 개선 프로토콜들을 적용함으로써 시스템의 성능 향상 정도 및 개별 트랜잭션 수행 시간에 미치는 영향들에 대해 구체적인 성능 분석이 이루어지는 것이 타당할 것이다. 또한 그룹 완료 프로토콜의 경우 수행 주기가 중요한 요소인데 기존의 DRDB시스템 연구에서는 시스템의 자료 경쟁 수준을 반영하여 동적으로 수행 주기를 구하거나[22], 일정한 로그 버퍼 크기를 기준으로 하였다[21]. 이는 기존의 데이타베이스 시스템에 비해 자료 경쟁 수준이 높으며, 실시간으로 다양하게 변하는 MMDB시스템에 적용하기에는 여러 문제점이 있으며, MMDB시스템에 적합한 새로운 그룹 완료 프로토콜을 필요로 한다.

본 논문의 2장에서는 MMDB 시스템의 구조와 구성 요소들에 대해 기술하고, 3장에서 선-완료 프로토콜과 그룹 완료 프로토콜에 대한 지금까지 수행된 관련 연구들에 대해 상세하게 고찰하고, 4장에서는 MMDB시스템에 그룹 완료 프로토콜을 적용할 때 발생 가능한 교착상태에 대해 분석한다. 5장에서는 MMDB시스템에 맞추어 새롭게 제안하는 "디스크 그룹 완료" 프로토콜에 대해 설명한다. 이 프로토콜은 시스템의 자료 경쟁 수준이나 로그 버퍼 크기가 아닌 성능에 병목이 되는 디스크의 작업 주기에 맞추어 그룹 완료를 수행하는 것이다. 6장에서 다각적이고 구체적인 모의실험을 실시하여 여러 완료 프로토콜들의 성능을 분석하며, 7장에서 결론을 맺는다.

2. MMDB시스템의 구조

MMDB 시스템은 하나 이상의 CPU와 휘발성 주기억 장치(RAM)로 구성되며, 주기억 장치의 일부분을 전역 로그 버퍼로 사용한다. 주변 장치로는 로그 정보를 따로 보관하는 로그 디스크와 데이타베이스 백업을 위한 디스크들이 있으며, 모든 데이타는 주기억 장치에 상주하는 것으로 가정한다. MMDB시스템의 구조 및 구성 요소가 그림 1에 도식되어 있다.

트랜잭션은 여러 클라이언트에서 동시에 생성되어 트랜잭션 관리기(Transaction Manager:TM)로 보내지며, TM은 설정된 동시성 제어 방식에 의해 큐에 도착한 트랜잭션들의 연산들을 수행한다. 연산 작업은 레코드 단위로 이루어지며, 주어진 인덱싱 기법에 따라 저장된 레코드를 찾아 읽기/쓰기 작업을 수행한다. 트랜잭션에서 발생한 갱신은 지연된 갱신(delayed update)방식이 아닌 즉각적인 갱신(immediate update)방식에 의해 주기억 장치에 있는 데이타베이스에 바로 반영되는 것으로 가정한다.

모든 트랜잭션은 시작되면서 개별적인 취소 / 재 수행(undo/redo) 로그를 가지고, 쓰기 연산을 수행하기 전

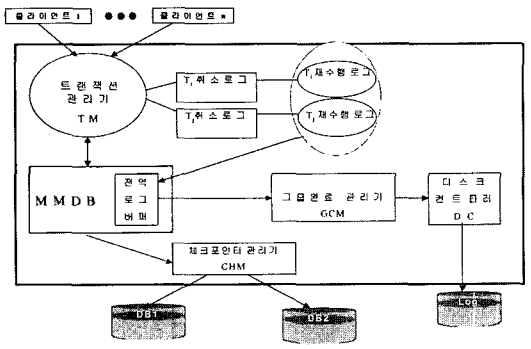


그림 1 MMDB시스템 구조

취소 로그가 작성되고, 데이타베이스의 쓰기 작업이 이루어진 후 재 수행 로그가 기록된다[93]. 모든 연산 작업이 끝나면 TM은 그 트랜잭션의 재 수행 로그들만을 로그 디스크로 출력시키기 위해 디스크 출력 시스템 콜을 호출한다. 디스크 컨트롤러(Disk Controller:DC)로부터 디스크 출력 작업이 끝났음을 전달받으면, TM은 이를 트랜잭션에게 전달함으로써 트랜잭션의 수행이 완료된다. 이때 TM은 완료된 트랜잭션이 개별적으로 가지고 있던 취소 로그도 삭제한다.

만약 그룹 완료를 수행하는 경우 연산 작업을 끝내면 TM은 이를 그룹완료 관리기(Group Commit Manager:GCM)에게 알린다. 그러면 GCM은 해당 트랜잭션의 재 수행 로그만을 전역 로그 버퍼로 옮기고, 미리 설정된 그룹 완료 수행 주기가 되면 전역 로그 버퍼에 취합된 여러 트랜잭션들의 재 수행 로그들을 로그 디스크로 출력하기 위해 디스크 출력 시스템 콜을 호출한다. DC로부터 디스크출력 작업이 완료되었음을 전달받으면 GCM은 출력된 트랜잭션 목록을 TM에게 전달하고, TM은 출력된 트랜잭션들의 로크를 해제하고, 취소 로그를 주기억 장치에서 삭제한 후 해당 트랜잭션들이 완료되었음을 사용자에게 알린다. 위와 같은 절차에 의해 전체 MMDB시스템의 재 수행 로그는 로그 디스크에 쓰여지고, 완료 작업중인 트랜잭션의 로그들만이 주기억 장치의 전역 로그 버퍼에 있게 된다. 시스템 에러가 발생한 경우 신속한 복구를 위해 평상시 갱신된 데이타들을 디스크로 백업하는 체크포인팅 작업은 체크포인팅 전담 관리기(Checkpoint Manager: CHM)에 의해 수행된다. CHM은 퍼지 체크포인팅을 하며, 두개의 백업을 번갈아 갱신하는 핑퐁 체크포인트를 하는 것으로 한다[4,21].

3. 관련 연구

3.1 선-완료(pre-commit) 프로토콜

선-완료 프로토콜은 트랜잭션의 연산 작업이 모두 끝나고 완료하기 전에 그 트랜잭션이 보유한 로크들을 먼저 해제하게 하는 것으로 로그들의 디스크 출력이 트랜잭션간의 일관성 순서대로 이루어지는 경우에 사용할 수 있으며[23], 읽기-전용 트랜잭션중에서 완료중인 트랜잭션이 수정한 데이타를 전혀 읽지 않은 것들을 골라 먼저 완료시키기 위해서는 여러 부가적인 처리가 필요하다[19]. 이 프로토콜은 선-완료한 트랜잭션의 응답 시간에는 변화가 없지만, 이 트랜잭션이 보유한 로크를 기다리는 다른 트랜잭션들의 로크 대기 시간을 감소시킴으로써 전체적으로 시스템 성능을 향상시킬 수 있다[12].

선-완료 프로토콜과 관련하여 모의실험을 한 연구로는 분산 DRDB시스템에 선-완료 프로토콜을 적용함으로써 발생하는 연쇄 철회(cascading abort)를 예방하기

위해 일 단계만 선-완료를 적용하는 방안이 있으며, 모의 실험을 통하여 시스템 성능이 향상되었음을 보였다[24]. MMDB 시스템에서는 서론에서 언급한 바와 같이 초창기 연구들에서는 선-완료 프로토콜의 적용을 가정하기만 할 뿐 구체적으로 적용한 연구는 없다. 최근의 MMDB시스템의 동시성 제어에 관한 연구중에 DB전체에 대해 로크 하나만을 두어 결국 트랜잭션들을 순차 수행시키면서 좀 더 빠른 트랜잭션 완료를 위해 선-완료를 도입한 "Almost-serial"방식이 있다[25]. 이 연구에서는 로깅이 필요없는 읽기-전용 트랜잭션들을 먼저 완료시키면서 발생할 수 있는 비일관성을 해결하기 위해 타임스탬프(Timestamp)와 상호배제 배열(Mutex Array)을 사용하였으며, 이를 상용시스템 "DataBlitz"에 구현하고, 성능 분석을 하였다.

3.2 그룹 완료(group-commit) 프로토콜

그룹 완료 프로토콜은 개별적으로 트랜잭션의 로그 정보를 디스크로 출력하지 않고, 주기억 장치의 전역 로그 버퍼에 여러 트랜잭션들의 로그를 모아 두었다가, 미리 정해진 주기가 되면 모아진 로그들을 한번에 일괄적으로 디스크로 출력함으로써 시스템 전체적으로 디스크 출력 횟수를 감소시키고자 하는 방안이다[12].

DRDB시스템에서의 그룹 완료 프로토콜은 IMS "Fast-Path"에서 처음으로 구현하였으며[26], [27]에서는 Cedar 파일 시스템(CFS)이 내장하고 있는 하드웨어-레벨의 복구를 개선하는 동시에 더 나은 안전성과 성능을 위해 로깅과 그룹 완료 프로토콜을 채택하였다. [22]에서는 대량의 트랜잭션이 발생하는 DRDB시스템에서 자료 경쟁 수준을 반영한 그룹 완료 타이머 산출 공식을 제시하였다. 그러나, 이 연구에는 CPU 큐잉시간이 트랜잭션 처리 시간의 대부분을 차지하는 시스템을 모델로 한 것으로 전체 트랜잭션들의 CPU 큐잉시간을 줄이기 위한 수단으로 그룹 완료 프로토콜을 도입하였다. 또한 디스크 작업 시간을 전혀 고려하지 않고, CPU 경쟁 수준만을 고려한 그룹 완료 타이머 값을 산출하였기 때문에 디스크 로깅 시간을 감소시키기 위한 MMDB시스템에서의 그룹 완료 프로토콜에 적용할 수 없는 연구 결과이다.

MMDB분야에서의 그룹 완료 프로토콜의 적용은 [14]에서 제안하였으며, 초창기 논문들에서는 서론에서 밝힌 바와 같이 단순하게 그룹 완료의 적용을 가정하거나, 또는 트랜잭션 응답시간에 역효과가 있을 거라는 예측을 했을 뿐이다. 최근의 MMDB시스템에 관한 연구나 상용시스템에서도 그룹 완료 프로토콜을 채택하고 있다고만 할뿐[4,28,29], 그룹 완료의 주기나 적용한 결과에 대해서는 구체적인 언급이 없다.

4. MMDB시스템에 그룹 완료 프로토콜의 적용

4.1 교착 상태 발생 예

최근의 MMDB시스템에서는 수행되는 응용 분야의 성향에 따라 DB전체에 하나의 로크를 두어 트랜잭션들을 순차 수행시키거나, 또는 다단계 동시성을 제공하기도 한다[28]. 2단계 로킹 방식을 사용할 경우 연산 대상인 데이타가 모두 주기억 장치에 있으므로 연산 작업이 신속하게 수행되어 로크 경쟁이 그리 심하지 않은 경우에는 로크 관리 부담, 문맥 전환 비용 등을 줄이기 위해 로크 단위를 레코드보다는 원하는 레코드가 포함된 리레이션으로 하는 것이 적당할 수 있다[12]. 이러한 환경에서 그룹 완료를 적용하면서 수행되는 트랜잭션의 특징에 대한 분석없이 주어진 전역 로그 버퍼가 가득 찼을 때를 그룹 완료의 수행 주기로 하는 경우[21]에 아래와 같이 교착 상태가 발생할 수 있다.

- 이 시스템이 가질 수 있는 최대 사용자수를 5명으로 하고, 연산 대상이 되는 레코드가 포함된 리레이션을 단위로 로크를 획득하는 것으로 가정한다.
- 사용자$_3$, 사용자$_1$이 요청한 트랜잭션 T_3, T_1은 모든 연산 작업을 끝내고, 재수행로그가 전역 로그 버퍼에 저장되어 있는 상태이다.
- 사용자$_2$가 요청한 T_2 트랜잭션은 아래와 같은 연산들을 수행한 후 전역 로그 버퍼에 재수행 로그를 추가하였으나, 전역 로그 버퍼가 차지 않아 그룹 완료는 시행되지 않고 있는 상태이다.

```
read relation# 100
update relation# 200
commit
```

- 사용자$_4$가 요청한 T_4 트랜잭션은 아래 연산을 요청하였다.

```
update relation# 100
```

- 사용자$_0$가 요청한 T_0 트랜잭션은 아래 연산을 요청하였다.

```
read relation# 200
```

- 위의 두 연산은 완료중인 트랜잭션 T_2가 로크를 가지고 있는 리레이션들에 대한 연산 요청이므로, 트랜잭션 T_4, T_0는 로크 대기 리스트에 들어가게 되고, 전체 시스템에는 더 이상 수행중인 트랜잭션이 없게 되어 추가로 연산 작업을 마치고 전역 로그 버퍼를 채울 트랜잭션이 없게 되므로 전체 시스템은 교착 상태에 빠지게 된다.

4.2 교착 상태 해결 방안

위의 경우와 같은 교착 상태를 해결하기 위해서는 현재 수행중인 트랜잭션들이 모두 완료중이거나 로크 대기 상태이며, 일정 시간이 경과해도 새로운 트랜잭션이 발생하지 않는 경우 전역 로그 버퍼가 가득 차지 않았더라도 강제로 그룹 완료를 수행하여야 한다. 이를 위해서는 임의의 트랜잭션이 로크 요청 대기 리스트에 들어갈 때마다 시스템의 상황을 체크하여 강제 그룹 완료 여부를 결정하도록 시스템을 수정하여야 한다. 또는 그룹 완료 프로토콜과 함께 선-완료 프로토콜을 병행하여 완료중인 트랜잭션들이 모두 로크를 해제하여 이 로크를 대기하는 트랜잭션을 없애서 교착 상태를 해결하는 것이다. 그러나, 이 경우에도 시스템 성능을 위해서는 전역 로그 버퍼가 일정 시간 이상 차지 않으면 강제로 그룹 완료를 수행하는 것이 나을 수도 있다. 더불어 이런 일이 자주 반복되면 전역 로그 버퍼의 크기를 적절하게 조절하는 작업이 뒤따라야 한다. 또한 선-완료를 적용할 때는 읽기-전용 트랜잭션의 처리를 위한 알고리즘이 구현되어야 한다. 그러나, MMDB시스템의 구축 사례를 보면 주식의 시세 조회나 통신 분야의 콜 데이타 연결이나 다양한 부가서비스 제공 또는 무선 통신 기기의 현재위치 제공 서비스 등으로[4] 짧은 길이의 트랜잭션들이 신속하게 수행되어야 하는 한편 동시에 수행되는 사용자 수는 예측하기가 어려워 시스템 성능을 최고로 하는 그룹 완료 주기를 설정하기는 어려운 문제이다.

5. 디스크 그룹 완료(Disk Group Commit) 프로토콜

5.1 제안 배경 및 동작 과정

모든 연산을 끝낸 트랜잭션은 갱신에 관한 재 수행 로그들이 디스크에 출력되어야만 완료할 수 있다. 로깅 정보의 디스크 출력은 로그만을 전용으로 저장하는 로그 전용 디스크를 가정한다면 순차출력이므로 탐색시간(Seek time)은 필요하지 않다. 하지만 로깅할 때마다 소요되는 회전 지연 시간(Latency time)은 생략할 수 없으며, 현재의 디스크 기술로도 이 시간은 MMDB시스템의 트랜잭션 수행 시간에 영향을 미치므로[30], 여러 개 트랜잭션의 로그 정보를 취합하여 함께 출력하는 그룹 완료 프로토콜이 시스템 성능에 도움을 줄 것이다. 그렇다면 최적의 시스템 성능을 위해서는 그룹 완료의 가장 중요한 결정 사항인 시행 주기를 정해야 한다. 기존의 DRDB시스템에서는 상수로 정한 전역 로그 버퍼 크기나 또는 동적으로 변하는 자료 경쟁 수준을 감안한 CPU소요시간을 기준으로 한다. 그러나, MMDB 시스템의 경우 연산 작업중에는 디스크 입출력이 없으므로 성능에 병목이 되는 디스크 작업 성능에 초점을 맞추게 되면 트랜잭션들의 자료 경쟁 수준을 염두에 두지 않아도 되므로 훨씬 간단하면서도 효과적일 수 있다. 즉, 디스크가 쉬지 않고 출력을 수행하고, 한 번의 출력을 수행할 때 이전 디스크 출력 이후부터 지금까지 축적된 모든 트랜잭션들의 로그를 한번의 디스크 출력 연산으로 해결하는 것이 시스템 성능에 최선이 될 수 있다. 이러한 관점에서 우리가 새롭게 제안하는 프로토콜이 디스크 출력 주기에 맞추어 동작하는 그룹 완료 프로토콜이다. 디스크의 출력 주기에 기준한 그룹 완료 프로토콜은 4절에서 설명한 교착 상태가 발생하지 않는 이점 또한 있다. 디스크 그룹 완료 프로토콜의 연산 수행 과정은 아래와 같다.

<트랜잭션의 연산 수행 과정>

1. 트랜잭션 T_i가 시작되면 활동 트랜잭션 목록(active transaction list)에 T_i가 추가된다.
2. <Start T_i>라는 로그 레코드가 트랜잭션의 개별 취소 / 재 수행 로그버퍼(undo/redo log buffer)에 쓰여진다.
3. T_i가 수행되면서 발생한 자료의 갱신에 대한 로그는 트랜잭션의 개별 로그 버퍼에 기록된다.
4. 모든 연산 작업이 끝나면 완료가 시작된다.

<디스크 그룹 완료 과정>

5. 트랜잭션이 모든 수행을 끝내면 트랜잭션 관리기 TM은 <Commit T_i>로그를 개별 재 수행 로그 버퍼에 기록한다.
6. 이때 선-완료프로토콜이 적용될 수 있다.
7. TM은 디스크 그룹 완료 관리기 GCM에게 트랜잭션 T_i의 완료를 요청한다.
8. GCM은 아래의 두 가지 작업중 하나를 수행한다.
   - 8.1 디스크가 쉬고 있는 상태이고, 전역 로그 버퍼도 비어 있다면 트랜잭션 T_i의 로그를 바로 디스크에 출력하도록 디스크 컨트롤러에게 요청한다. 출력 작업이 끝났음을 통보받으면 TM에게 트랜잭션 T_i의 완료를 알린다.
   - 8.2 디스크가 현재 출력 작업중이면 트랜잭션 T_i의 ID를 완료 목록 <*Commit List:* CL_j>에 추가하고, 개별 로그 버퍼에 있는 재 수행 로그를 전역 로그 버퍼로 옮긴다. 디스크 컨트롤러는 디스크 출력 작업이 끝나고 전역 로그 버퍼가 비어 있지 않다면 전역 로그 버퍼에 취합된 로그들을 한번의 디스크 출력 연산으로 로그 전용 디스크에 있는 로그 파일에 일괄적으로 출력시킨다. 출력 작업이 끝나면 완료 목록 <CL_j>에 속하는 트랜잭션들의 완료를 GCM에게 전달하고, GCM은 이를 다시 TM에게 알린다.
9. TM은 GCM으로부터 완료 목록 <CL_j>의 완료를 통보받으면 해당 트랜잭션들의 완료를 사용자들에게 알

린다.

< 디스크 그룹 완료 이후 과정 >

10. TM은 완료된 트랜잭션들의 모든 로크를 해제하고, 각 트랜잭션의 개별 로그 버퍼에 남아 있던 취소 로그를 주기억 장치에서 삭제시킨다.

5.2 시스템 변경 사항

일반적으로 소형의 MMDB시스템은 디스크 출력 부분을 운영체제에서 제공하는 시스템 콜을 호출함으로써 해결한다. 그러면 정확하게 MMDB시스템에서 원하는 시각에 바로 디스크 출력이 이루어지는 것을 보장할 수 없으며, 디스크 그룹 완료 프로토콜은 디스크의 출력 상태를 직접 알고 조작해야 하므로 로그 파일은 운영체제의 파일시스템을 사용하지 않고 직접 조작, 출력하는 파일시스템을 구축하여야 한다.

6. 모의 실험

6.1 MMDB 시스템 모의실험 모델

제안한 디스크 그룹 완료 프로토콜의 구체적인 성능을 측정하기 위하여 MMDB시스템 모델을 구축하고 다양한 모의실험을 실시하였다. 모의 실험은 폐쇄 큐잉 모델 방식이며, 다음은 구축한 MMDB시스템의 가정사항들이다.

- 트랜잭션은 주어진 다중 프로그래밍 레벨(Multi-Programming Level : MPL)에 따라 여러 클라이언트에서 동시에 생성된다.
- 선-완료 프로토콜의 적용을 위해 엄격한 2단계 로킹을 동시성 제어방식으로 채택한다.
- 데이타베이스는 여러 개의 리레이션으로 구성된다.
- 트랜잭션은 사용자나 시스템의 고장에 의한 철회는 없고 단지 교착상태 해결을 위한 철회만이 있다고 가정한다.
- 교착 상태는 구현된 교착 상태 검출 알고리즘에 의해 로크 요청시 발견되며, 발견된 교착 상태는 이를 유발한 트랜잭션을 철회시킴으로써 해결한다.
- 트랜잭션은 SELECT, INSERT, UPDATE와 같은 자료에 대한 연산과 BEGIN, END, COMMIT, ABORT와 같은 트랜잭션 관리연산들로만 구성된다.
- 트랜잭션에 속하는 연산은 레코드 단위이며, 로크는 레코드가 포함된 리레이션 단위로 하며, 로크 관리 비용은 따로 계산하지 않는 것으로 한다.
- 한 레코드를 읽고 쓰는 경우 트랜잭션에는 WRITE연산만 있고, 한번의 WRITE 연산은 "해당 레코드 찾기+읽기+쓰기"에 해당하는 연산 수행 시간이 걸리는 것으로 한다.
- 생성된 트랜잭션은 트랜잭션 관리기에 의해 수행되며, 교착 상태에 의해 수행이 철회되면 실제의 그 트랜잭션이 완료될 때까지 트랜잭션 관리기에 의해 반복 수행된다.
- 이 모의실험은 완료 프로토콜에 따른 성능 분석을 하는 것이 목적이므로 체크포인팅 작업을 포함시키지 않는다.
- 모의 실험중 시스템 고장은 발생하지 않는 것으로 가정하므로 시스템 고장으로 인한 MMDB시스템의 재적재(reload)나 회복을 위한 다른 부가적인 알고리즘은 가정하지 않는다.
- 로깅 작업의 정확한 성능 측정을 위해 로그 전용 디스크를 가정한다

6.2 모의 실험 수행 횟수 선정

모의 실험은 CSIM 모의 실험기[31]를 사용하며, 성능 평가 기준치는 트랜잭션 평균 수행시간(avg_res_time)으로 한다. 사전 모의실험을 걸쳐 각 모의실험은 800,000개의 트랜잭션들이 수행될 때까지의 평균치를 사용하고, 모의실험 초기 값들은 정확한 자료 경쟁 수준에 도달하지 못한 값들이므로 실제로는 810,000개의 트랜잭션을 수행시켜 최초 10,000개의 트랜잭션 수행 결과는 모의 실험 측정치에서 제외시켰다.

6.3 모의 실험 매개 변수

매개 변수는 시스템 변수, 데이타베이스 변수, 트랜잭션 변수로 나뉘며, 표 1, 표 2, 표 3에 그에 대한 의미와 값이 나타나 있다. 시스템 변수는 시스템 하드웨어를 대표하는 고정값을 가지며, 트랜잭션간의 다양한 자료 경쟁 수준을 나타내기 위해 트랜잭션 변수중 MPL은 여러 값들을 가진다. 그룹 완료 프로토콜의 성능 측정을

표 1 시스템 변수

변수명	의미	값
MM_Access	주기억 장치에 있는 하나의 레코드를 접근하는 데 걸리는 시간	0.0001 ms
MM_Search	주기억 장치에 있는 하나의 레코드를 찾는 데 걸리는 시간	0.0003 ms
Lock_Time	로크 획득 시간	0.025 ms
Unlock_Time	로크 해제 시간	0.025 ms
LogAccessTime	전역 로그버퍼에 로그 레코드를 기록하는 데 걸리는 시간	0.00011 ms
DiskBandWidth	디스크 입출력 속도	30Mbytes/sec
Latency	평균 회전 지연 시간	4.17 ms(7200 RPM)
DiskBlock	디스크 출력 단위	512 bytes

표 2 데이타베이스 변수

변수명	의미	값
DataBase	주기억 장치 데이타베이스 크기	2000 relations = 1M records
RecordSize	레코드 크기	64 bytes
RedoSize	재 수행 로그 레코드 크기	128 bytes
GlobalBuffer	전역 로그 버퍼 크기	1K, 2K, 3K, 4K, 5K ,6K bytes

표 3 트랜잭션 변수

변수명	의미	값
MPL	다중프로그래밍 레벨	10, 20, 30, 40, 50
TranSize	트랜잭션의 길이(연산 수)	10 operations
WriteProb	쓰기 연산의 비율	0.6

표 4 모의실험 분석 변수들의 의미

실험 결과 변수	의미
res%	완료개선책을 사용하지 않는 경우(None)의 트랜잭션 평균 수행시간을 100으로 하고, 완료개선책을 사용한 경우의 트랜잭션 평균 수행 시간을 퍼센트 비율로 나타낸 값
avg_res_time	트랜잭션 평균 수행 시간(ms). op_done_time + commit_time
throughput	단위시간 생산량 (committed transactions/ms). avg_res_time과 같은 성격을 가진다.
op_done_time	트랜잭션이 모든 연산 작업을 수행하는 데 걸리는 평균 시간(ms).
commit_time	트랜잭션이 완료요청을 한 후 실제 완료될 때까지 걸리는 평균시간(ms)

위해 데이타베이스 변수중 GlobalBuffer역시 다양한 크기를 가지게 한다[15,32,33].

6.4 모의 실험 결과 분석

모의 실험 결과를 분석하는 데 쓰인 변수들의 의미가 표 4에 정리되어 있으며, MPL별로 완료 개선책을 사용한 경우의 시스템 성능 변화를 그림 2에 도식하였다. 모든 그래프에서는 일반적인 그룹 완료의 수행 주기인 전역 로그 버퍼 크기를 기준으로 그룹 완료를 사용하는 경우(GC), 본 논문에서 제안한 디스크 그룹 완료 프로토콜을 사용한 경우(DC), 전역 로그 버퍼 기준 그룹 완료를 수행하면서 선-완료를 병행하는 프로토콜(P+GC), 디스크 그룹 완료 프로토콜에 선-완료 프로토콜을 병행하는 경우(P+DC), 총 4가지 경우의 성능을 비교하여 분석하였다. 각 그래프의 제목에는 MPL수치와 함께 완료 개선책을 전혀 사용하지 않은 경우(None)의 avg_ res_time값이 함께 나타나 있다. 그래프의 y값인 res% 는 None경우의 avg_res_time값을 100으로 가정하고 완료 개선책을 적용한 경우의 avg_ res_time을 상대적인 비율로 나타낸 값이며, x값은 전역 로그 버퍼 크기를 나타낸다.

GC경우는 GlobalBuffer를 기준으로 그룹 완료를 수행하므로 GlobalBuffer 크기에 따라 다른 시스템 성능을 나타내나, DC경우는 아주 큰 크기의 GlobalBuffer를 가정하고 디스크 컨트롤러가 그룹 완료를 주도하므로 그래프상의 주어진 GlobalBuffer와 무관하게 하나의 값을 가지는 특징이 있다.

또한 본 논문에서 사용한 GC경우는 전역 로그 버퍼 크기를 기준으로 그룹 완료를 수행하지만 4장에서 기술한 교착 상태가 발생하는 경우 전역 로그 버퍼가 가득 차지 않았더라도 강제로 그룹 완료를 수행하도록 개선한 프로토콜을 적용하였다.

• 그룹 완료 프로토콜의 성능 평가

그림 2를 보면 기본적으로 그룹 완료 프로토콜은 어떠한 자료 경쟁 수준이나, 그룹 완료 수행 시기에 상관없이 그룹 완료를 수행하지 않는 경우보다 시스템 성능에 도움을 주는 완료 개선책임을 알 수 있다. 완료 개선책을 사용하지 않는 경우 디스크가 한 트랜잭션의 로그를 디스크에 출력하는 동안 다른 트랜잭션에 속한 연산들은 사용하는 데이타가 모두 주기억 장치에 상주해 있으므로 매우 신속하게 진행되어 결국 DRDB시스템에 비해 완료를 위한 로그 자료가 디스크 큐에 급속하게 쌓이게 된다. 디스크는 이를 트랜잭션 단위로 하나씩 출력하기 때문에 commit_time이 급증하게 되고, 더불어 완료될 때까지 로크를 가지고 있는 시간도 증가하므로 결과적으로 op_done_time또한 증가하게 된다. 이로써 avg_res_time이 형편없이 길어지게 되는 결과를 초래한다. 예를 들어 MPL이 30인 경우 완료 개선책을 사용하지 않는 경우에 op_done_time이 61.55ms이고, commit_time은 64.44ms로 avg_res_time이 125.99ms가 된다. 그러나, 5K 크기의 전역 로그 버퍼를 가지고 그룹 완료를 적용한 경우에는 op_done_time이 12.4ms, commit_time이 8.92ms가 되어 None 경우의 avg_res_

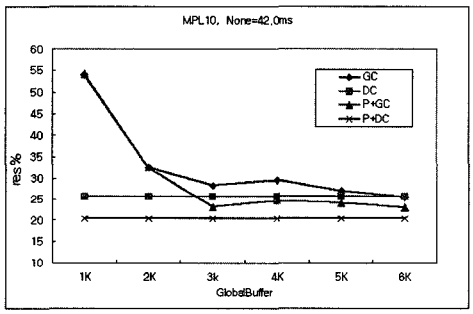
time 125.99ms의 16.92%에 해당하는 21.32ms를 가져 트랜잭션의 수행 시간이 효과적으로 개선됨을 알 수 있으며, 전체 시스템 성능 역시 throuthput이 0.24에서 1.41로 증가한다.

그러나, 적절치 못한 크기인 1K크기의 전역 로그 버퍼를 가지고 그룹 완료를 수행하는 경우는 avg_res_time이 68.26ms가 되어 5K인 경우에 비해 ⅓수준으로 성능 저하를 가져옴을 알 수 있다. 그러므로, 최적의 성능을 가지기 위해서는 적절한 주기를 가지는 그룹 완료프로토콜의 적용이 필요함을 알 수 있다.

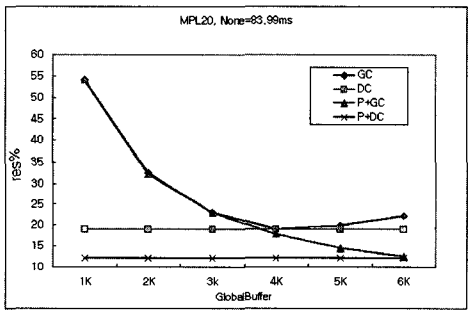
• 디스크 그룹 프로토콜 방식의 성능 평가

이 실험은 기존의 그룹 완료 프로토콜의 개선책으로 본 논문에서 제안한 디스크 그룹 완료 프로토콜의 성능 향상 여부를 알아보기 위한 것이다. 디스크 출력이 병목인 경우에는 트랜잭션들간의 자료 경쟁 수준에 맞추어 CPU에서 그룹 완료를 수행하는 것보다 디스크가 대기 상태가 되는 경우가 없도록하면서 한 번의 디스크 출력 시 가능한 최대량의 로그를 출력하게 하는 것이 더 나은 성능을 나타낼 것이라는 생각에서 제안한 프로토콜이다.

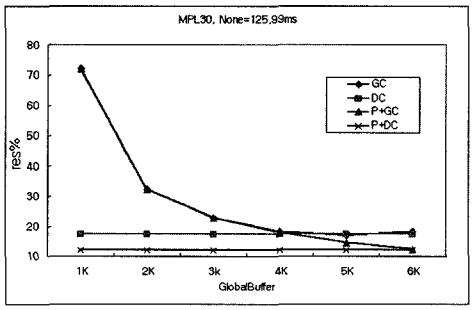
그림 2에서 확인한 것처럼 디스크 그룹 완료 프로토콜은 자료 경쟁 수준에 맞는 전역 로그 버퍼 크기를 구할 필요가 없으며, 전역 로그 버퍼 크기에 따라 다른 성능을 가지는 그룹 완료 프로토콜의 최적의 경우와 같거



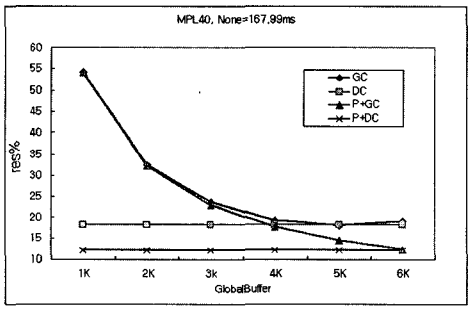
(a) MPL 10



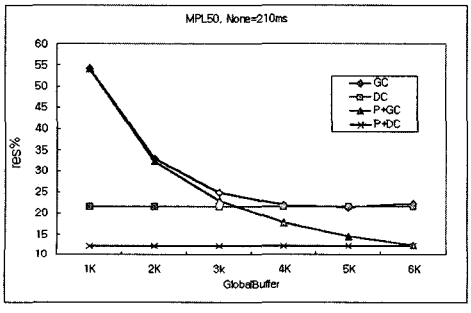
(b) MPL 20



(c) MPL 30



(d) MPL 40



(e) MPL 50

그림 2 MPL별 완료 개선 프로토콜에 따른 시스템 성능 비교

나 좀 더 나은 시스템 성능을 가짐을 알 수 있다. 이로써 다양한 자료 경쟁 수준을 가지는 MMDB시스템 환경에서 디스크 그룹 완료 프로토콜은 아무런 선결 작업이 없어도 최적의 시스템 성능을 가지는 완료 개선책임을 알 수 있다.

• 선-완료 프로토콜의 효과

이 실험은 기존 MMDB시스템에서 많이 채택한 선-완료프로토콜이 실제로는 시스템 성능에 어떤 영향을 미치는 지 분석한 것이다. 선-완료프로토콜만을 채택한 경우 트랜잭션 단위로 로크 출력을 하므로 이 프로토콜의 적용은 로크 해제 시간만을 앞당겨 디스크 큐에 쌓이는 트랜잭션 로그를 증가시켜 포화 상태가 된 디스크 큐의 길이만 길게 하는 역할을 하여 시스템 성능이 전혀 개선되지 않았다. 그룹 완료 프로토콜과 더불어 선-완료 프로토콜을 적용하면서 전역 로그 버퍼 크기가 1~2K인 경우에도 이런 현상이 남아 있어 시스템 성능이 선-완료 프로토콜을 적용하기 전과 비슷하였다.

그러나, 그룹 완료프로토콜과 선-완료프로토콜을 병행하고 전역 로그 버퍼 크기를 크게 하면 선-완료프로토콜의 효과인 로크 대기 시간을 줄이는 것이 긍정적으로 작용하여 어떤 MPL에서든 그룹 완료프로토콜만을 적용하는 것보다 향상된 시스템 성능을 가진다. 디스크 그룹 완료 프로토콜은 기존의 그룹 완료프로토콜보다 항상 우월한 성능을 가지며, 여기에 선-완료 프로토콜을 병행하면 다른 트랜잭션들의 로크 대기 시간을 줄이고 그룹 완료에 빨리 참여하게 만들어 전체 시스템 성능이 최상이 된다.

7. 결 론

모든 자료가 주기억 장치에 상주하여 획기적인 트랜잭션 성능을 가져다 주는 MMDB시스템은 최근 빠른 추세로 상용화되면서 성능 향상을 위한 여러 방안들이 제시되고 있으며, 이중에서 평상시 유일한 디스크 출력 작업인 로깅 작업의 부하를 감소시키기 위한 방안이 주요 연구과제로 등장하고 있다[34]. 완료 과정의 개선 방안중 하드웨어의 부착이나 시스템 디자인의 큰 변경없이 적용할 수 있는 방안으로 선-완료, 그룹 완료 등의 프로토콜이 있으며 기존의 DRDB시스템에서는 자료 경쟁 수준을 반영한 동적 그룹 완료 수행 주기를 산출하는 방안이 제시되기도 하였다. 그러나, 로깅 작업이 트랜잭션 수행 시간의 많은 부분을 차지하는 MMDB시스템에서는 그룹 완료프로토콜을 적용한 결과에 대해 구체적인 분석이 없었다.

본 논문에서는 MMDB시스템에 그룹 완료 프로토콜 적용시의 문제점을 제시하고, MMDB시스템에 적합한 새로운 그룹 완료 프로토콜인 디스크 그룹 완료를 제안했으며 완료 개선책의 정확한 성능 향상을 보이기 위해 다양한 자료 경쟁 수준에서의 다각적인 모의실험을 실시하였다. 그 결과 그룹 완료 프로토콜은 시스템 자료 경쟁 상황을 정확하게 반영하지 못한 수행 주기를 사용하더라도 그룹 완료를 적용하지 않을 때와 비교하여 시스템 성능 향상뿐만 아니라 트랜잭션의 수행 시간 또한 감소시키는 아주 효과적인 방안임이 밝혀졌다. 그러나, 기존의 그룹 완료 프로토콜을 적용하여 최대의 성능 향상을 얻기 위해서는 여러 자료 경쟁 수준에 맞는 그룹 완료 주기를 산출하여야 하고, 또한 잘못된 그룹 완료 주기는 교착 상태가 발생할 수도 있다. 위의 문제점을 해결하는 방안으로 우리가 제시한 디스크 그룹 완료 프로토콜은 디스크 컨트롤러가 끊임없이 작업을 하면서 또한 한번의 디스크 출력시 최대한으로 많은 양의 출력을 하게 하므로 시스템이 최고의 성능을 가진다는 것을 모의실험을 통해 확인하였다. 디스크 그룹 완료 프로토콜은 기존의 디스크 컨트롤러의 동작 과정에 수정을 가해야 하는 단점이 있으나, 실시간으로 다양하게 변하는 자료 경쟁 수준을 가지는 MMDB시스템에 그룹 완료를 적용할 때 그룹 완료 주기를 정해야 하는 선행 작업이 없이도 가장 최적의 시스템 성능을 가져올 수 있다는 장점이 있다. 한편 선-완료 프로토콜은 일반적인 데이타베이스 시스템에서는 성능 향상이 있을 수 있으나, MMDB시스템에서는 신속하게 연산 작업을 수행한 트랜잭션들을 이미 서비스 포화 상태인 디스크로 더욱 빨리 몰리게 하여 성능 향상에 도움이 되지 못하며, 그룹 완료나 디스크 그룹 완료프로토콜의 보조 수단으로 사용될 때만 시스템 성능에 보탬이 됨을 밝혔다.

향후에는 객체지향 주기억 장치 데이타베이스 시스템을 구축하고 관리할 수 있는 공개 소프트웨어인 Fast-DB관리도구[35]에 디스크 그룹 완료 프로토콜을 추가로 구현하여 실제 구현된 시스템에서의 성능 측정을 할 계획이다.

참 고 문 헌

[1] Margaret H. Eich, "Forward Main Memory Databases : Current and Future Research Issues," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 4, No. 6, pp.507-508, December 1992.

[2] Altibase Development Team, "ALTIBASE System Architecture," From Altibase mmdb system Homepage.

[3] H.V.Jagadish, D.Lieuwen, R.Rastogi, A.Silberschatz. "Dali: A High Performance Main Memory Storage Manager," Proceedings of the 20th International Conference on Very Large Data Bases, pp.12-15, September 1994.

[4] The TimesTen Team, "In-Memory Data Manage-

ment for Consumer Transactions The TimesTen Approach," Proceedings of the ACM SIGMOD/ PIDS International Conference on Management of Data, June 1999.

[5] P.A. Bernstein, V.Hadzilacos, and N.Goodman, "Concurrency Control and Recovery in Database Systems," Addison-Wesley (Reading MA), 1987.

[6] Rosennblum M, Bugnion E, Herrod SA, Witchel E, Gupta A, "The Impact of Architectural Trends on Operating System Performance," Proceedings of the 1995 Symposium on Operating Systems Principles, ACM Press, pp.285-298, 1995.

[7] Wu M, Zwaenepoel W(1994) "eNVy: A Non-Volatile, Main Memory Storage System," Proceedings of the 1994 International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS), October 1994, ACM Press, pp. 86-97.

[8] American Power Conversion(1996), "The Power Protection Handbook," Technical report. American Power Conversion, West Kingston, R.I.

[9] Chen PM, Ng WT, Chandra S, Aycock CM, Rajamani G, Lowell D, "The Rio File Cache: Surviving Operating System Crashes," Proceedings of the 1996 International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS), ACM Press, pp. 74-83.

[10] Wee Teck Ng, Peter M. Chen, "Integrating reliable memory in databases," International Journal on VLDB, Auguest 1998.

[11] Margaret H. Eich, "A classification and comparison of main memory database recovery techniques," Proceedings of International Conference on Data Engineering, pp.417-424, 1987.

[12] H. Garcia-Molina, K. Salem, "Main Memory Database Systems: An Overview," IEEE Trans. on Knowledge and Data engineering, Vol. 4, No. 6, pp.509-516, December, 1992.

[13] Hector Garcia-Molina, Richard J.Lipton, and Jacobo valdes, "A Massive Memory Machine," IEEE Transactions on Computers, Vol. c-33, No.5, pp. 391-399, 1984.

[14] DeWitt. D.J.. Katz, R.H., Olken. F., Shapiro, L.D., Stonebraker, M. R., and Wood, D., "Implementation Techniques for Main Memory Database Systems," Proceedings of SIGMOD '84, pp.1-8, June 1984.

[15] Le Gruenwald and Margaret H. Eich, "MMDB Reload Algorithms," Proceedings of ACM SIGMOD, pp.397-405, 1991.

[16] Lehman, T.J. et al., "An Evaluation of Starburst's Memory Resident Storage Component," IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 6, pp.555-566, Dec. 1992.

[17] Eliezer Levy and Avi Silberschatz, "Incremental Recovery in Main memory Database System," dept. of Computer Sciences University of Texas at Austin, TR-92-01.

[18] Xi Li, Margaret H. Eich, "Post-crash Log Processing for Fuzzy Checkpoing Main Memory Database," International Conference on Data Engineering, IEEE, pp.117-124, 1993.

[19] H. V. Jagadish, Avi Siberschatz, and S. Sudarshan, "Recovering from Main-Memory Lapses," Proceedings of 19th VLDB conference, pp.391-404, 1993.

[20] Tobin J.Lehman, "Design and performance evaluation of a Main Memory Relational Database Systems," Ph.D dissertation of computer sciences dept. Univ. of Wisconsin-madison, August 1986.

[21] Kenneth Salm, Hector Garcia-Molina, "System M : A transaction Processing Testbed for Memory Resident Data," IEEE Trans. on Knowledge and Data Engineering, Vol. 2, No. 1, pp.161-172, March 1990.

[22] Pat Helland, Harald Sammer, Jim Lyon, Richard Carr, Phill Garrett, Andres Reuter, "Group Commit Timers and High Volume Transaction Systems," Tandem Technical Report 88.1, March 1988.

[23] 우승균, 이윤준, 김명호, "주기억 장치 데이타베이스 시스템에서의 회복 기법에 대한 고찰", 정보화학회지 제14권 제2호, pp.38~46, 1996.2.

[24] Ramesh Gupta, Jayant Haritsa, Krithi Ramamritham, "Revisiting Commit Processing in Distributed Database Systems," Proceedings of the 1997 ACM SIGMOD, pp.486-497, 1997.

[25] Stephen Blott, Henry F. Korth, "Almost-Serial Protocol for Transaction Execution in Main-Memory Database Systems," Proceedings of the International Conference on VLDB, August 2002.

[26] Gawlick, Dieter, and David. Kinkade, "Varieties of Concurrency Control in IMS/VS FastPath," IEEE Database Engineering, Vol.8, No.2, pp.3-10, June 1985.

[27] Robert Hagmann, "Reimplementing the Cedar File system Using Logging and Group Commit," Proceedings of 11th Symposium on Operating System Principles, pp.155-162, November 1987.

[28] J.Baulier, P.Bohannon, S.Gogate, C.Gupta, S.Haldar, S.Joshi, A.Khivesera, H.F.Korth, P.McIlroy, J.Miller, P.P.S.Narayan, M.Nemeth, R.Rastogi, S.Seshadri, A.Silberschatz, S.Sudarshan, M.Wilder, C.Wei, "Datablitz storage manager : Main memory database performance for critical applications," In Proceedings of the ACM SIGMOD/PIDS International Conference on the Management of Data, June 1999.

[29] Jerry Baulier, Stephen Blott, Henry F.Korth, Avi Silberschatz, "A Database System for Real-Time Event Aggregation in Telecommunication," In Proceedings of the International Conference on

VLDB Industrial track paper, August 1998.
[30] Bruce L. Worthington, Gregory R. Ganger, Yale N. Patt, John Wilkes, "On-Line Extraction of SCSI Disk Drive Parameters," Proceedings of the ACM Sigmetrics, 1995.
[31] H.Schewetman, "CSIM user's guide for use with CSIM revision16," MCC, June, 1992.
[32] Margaret H. Dunham, Jun-Lin Lin, and Xi Li, "Fuzzy Checkpointing Alternatives for Main Memory Databases," Recovery Mechanism in Database Systems, Prentice-Hall, 1997.
[33] Le Gruenwald and Sichen Liu, "A performance Study of Concurrency Control in a Real-Time Main Memory Database System," SIGMOD Record, Vol.22, No.4, pp.38-44, December 1993.
[34] Juchang Lee, Kihong Kim, Sang K.Cha, "Differential Logging: A Commutative and Associative Logging Scheme for Highly Parallel Main Memory Database," Proceedings of International. Conference on Data Engineering, pp.73-183, 2001.
[35] Konstantin Knizhnik, "FastDB Main Memory Database Management System," From FastDB Homepage.

이 인 선

1983년 3월~1987년 2월 서울대학교 자연과학대학 계산통계학과 이학사. 1994년 3월~1996년 8월 한국과학기술원 정보 및 통신공학과 공학석사. 1997년 3월~2003년 8월 서울대학교대학원 공과대학 전기·컴퓨터공학부 공학박사. 1988년 1월~1991년 10월 (주) 데이콤 빌링시스템개발그룹, 사원 1992년 7월~1996년 11월 한국정보문화진흥원 교육훈련본부, 주임연구원. 1998년 3월~현재 신구대학 컴퓨터정보처리과, 전임강사. 관심분야는 분산시스템, 주기억장치 데이타베이스 시스템, 그리드 시스템

염 헌 영

1980년 3월~1984년 2월 서울대학교 자연과학대학 계산통계학과 이학사. 1984년 3월~1986년 2월 Texas A&M 전산학과 공학석사. 1986년 3월~1992년 2월 Texas A&M 전산학과 공학박사. 1986년~1990년 Texas Transportation Institute, Research Scientist. 1992년~1993년 삼성데이타시스템, 선임연구원. 1993년~1998년 서울대학교 자연과학대학 전산과학과, 조교수. 1998년~현재 서울대학교 공과대학 컴퓨터공학부, 부교수. 관심분야는 분산시스템, 분산 데이타베이스 시스템, 멀티미디어, 그리드 시스템 등