중복 데이터베이스 시스템에서 낙관적인 원자적 방송을 이용한 동시성제어 기법

최 희 영† · 황 부 현††

요 약

중복 데이터베이스 시스템에서 트랜잭션을 처리하기 위해서 원자적 방송이 주로 사용된다. 그런데 원자적 방송을 사용할 경우에는 트랜잭션을 처리하기 전에 먼저 서버들 사이에 조정단계가 선행되어야 하므로 트랜잭션 지연과 같은 문제점이 있다. 이 논문에서는 원자적 방송을 사용하여 트랜잭션을 처리할 경우에 발생되는 트랜잭션 지연문제를 해결할 수 있는 알고리즘을 제안한다. 이를 위해서 제안된 알고리즘에서 트랜잭션은 낙관적인 방법을 이용하여 처리하고, 판독연산은 트랜잭션이 제출된 사이트에서 수행된다. 그리고, 기록연산은 중복된 모든 사이트에서 원자적으로 갱신이 이루어지도록 한다. 이렇게 함으로써 각 사이트의 클라이언트가 지역 데이터베이스에 제출한 연산을 모든 사이트에서 독립적으로 수행할 수 있게 되어 병행성이 향상되고 트랜잭션의 지연이 방지된다. 또한 트랜잭션의 직렬가능성은 완료검사 단계에서 트랜잭션의 순서번호를 검사함으로써 보장되도록 한다.

A Concurrency Control Technique Using Optimistic Atomic Broadcast in Replicated Database Systems

Hee-Young Choi† · Bu-Hyun Hwang††

ABSTRACT

To process transactions in fully replicated databases, an atomic broadcast is mainly used. In the case of using atomic broadcast, transactions can be delayed because of the coordinating step among servers before processing the transaction. In this paper, we propose an algorithm to resolve the problem of transaction delay. In the proposed algorithm, the transactions are processed by using the optimistic method. The read operations of a transaction are performed in the site that it is submitted and its write operations execute its updates atomically in all replicated sites. Since the transactions submitted to each site are performed independently in that site, their parallelism is enhanced and the transaction delay is prevented. The serializability of transactions is ensured by checking the sequence number of transactions in the completion-inspection step.

키워드 : 중복 데이터베이스(Replicated Database), 동시성제어기법(Concurrency Control Method), 낙관적동시성제어(Optimistic Concurrency Control), 원자적 방송(Atomic Broadcase), 갱신트랜잭션(Update Transaction)

1. 서 론

분산 컴퓨팅 시스템에서는 분산된 여러 사이트에 데이터를 중복시킴에 따라 가용성(availability), 신뢰성(reliability), 성능(performance)을 향상시킬 수 있다. 즉 사용자가 데이터를 원격(remote)으로 접근하지 않고 인접된 지역 데이터베이스에 접근하여 트랜잭션을 처리함에 따라 트랜잭션 처리시간을 감소시키고 성능을 향상시킬 수 있다. 중복 데이터베이스는 신뢰성 즉, 시스템의 견고성을 증가시키기 위한 메커니즘으로도 오랫동안 사용되어 왔다. 그러나, 모든 사이트에서 데이터의 일관성이 보장되어야 하는 문제점을 내포하고 있다[1, 3-5, 7].

중복 데이터베이스에서 사용되고 있는 갱신전파 기법으로는 Eager 기법과 Lazy 기법이 있다. Eager 기법은 트랜잭션 수행 중에 사본을 가진 모든 사이트가 동기화 되어 갱신이 이루어지도록 하는 기법이다. 이 기법은 직렬가능한 수행을 보장하나, 모든 사이트에서 갱신이 이루어져야만 완료가 가능하므로 성능과 응답시간(response time)이 좋지 않으며 많은 비용을 필요로 한다[7, 9].

Lazy 기법은 Eager 기법과는 달리 갱신전파가 트랜잭션 완료 후에 모든 사이트에서 비동기적으로 수행되는데 중복된 사이트의 수만큼 갱신 전파 트랜잭션이 수행되게 된다. 이 기법의 가장 큰 문제는 각 지역에서 수행되는 트랜잭션과 갱신전파 트랜잭션이 동시에 수행되었을 때 데이터의

※ 본 논문은 한국관학 재단 2000년도 목적기초연구 지역대학 우수 과학자 지원연구(R02-2000-00401)비에 의하여 연구되었음.
† 정 회 원 : 전남대학교 전산학과 시간강사
†† 정 회 원 : 전남대학교 전산학과 교수 · 전남대 정보통신연구소
논문접수 : 2001년 7월 19일, 심사완료 : 2001년 8월 4일

일관성이 파괴될 수 있다는 것이다. 그러나 Lazy 기법은 각 사이트에서 트랜잭션을 수행하는 동안 다른 중복 사이트와의 원격 통신을 요구하지 않고 지역적으로 트랜잭션 처리가 가능하기 때문에 통신비용을 줄일 수 있다. Lazy 기법은 트랜잭션의 철회율을 줄이고, 데이터의 일관된 값을 유지하기 위하여 갱신전파 트랜잭션의 재 조정절차가 필요하다[1, 5, 6].

중복 데이터베이스의 동시성 제어 기법에 의해서 트랜잭션을 수행할 때 중복 데이터의 일관성을 유지하기 위해서는 트랜잭션의 갱신에 대한 모든 정보가 모든 사이트에 전달되어야 하는데 이를 위한 메커니즘으로 원자적 방송 프리미티브기법이 있다. 그러나 원자적 방송은 먼저 서버들 사이에 조정 단계를 거친 후에 갱신 정보를 모든 사이트에 전달하므로 트랜잭션의 수행이 지연되는 문제가 발생하게 된다. 이에 따른 문제를 해결하기 위해 트랜잭션을 일단 수행시키고 트랜잭션이 완료되기 전에 유효성 검증을 행하는 낙관적인 동시성 제어 기법이 주로 사용되고 있다.

본 논문에서는 중복 분산 환경에서 낙관적인 방법을 사용함으로써 원자적 방송 프리미티브를 사용할 때 발생되는 트랜잭션들의 지연문제를 보완할 수 있는 방법을 제안한다. 이러한 지연은 트랜잭션이 제출된 사이트에서 트랜잭션에 대한 연산을 수행하고 수행된 결과 메시지를 모든 사이트에 방송하는데, 모든 사이트에서 방송 받은 트랜잭션 순서가 일치할 때까지 메시지의 전달을 지연시키는 문제가 발생한다. 메시지가 불일치한 순서로 처리된다면 지금까지 처리했던 연산들을 원래의 상태로 되돌려야(undoing) 하는 처리비용을 지불해야 하는 문제가 발생한다.

따라서 본 논문은 낙관적인 원자적 방송을 이용한 중복 데이터베이스 환경에서 각 사이트의 클라이언트가 지역 데이터베이스로 재출된 연산들을 그 사이트에서 독립적으로 수행하면서 병행성을 향상시킬 수 있는 동시성 제어 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 연구에서 제시된 중복 데이터베이스 시스템에서 트랜잭션 관리 방법과 문제점을 제시하고, 3장에서는 본 논문에서 사용하는 중복 데이터베이스 시스템구조와 트랜잭션 처리 모델을 제시한다. 4장에서는 중복 데이터베이스에서 동시성 제어를 위한 새로운 알고리즘을 제안하고, 이 알고리즘의 정확성 검증을 기술한다. 5장에서는 결론 및 향후 연구방향을 제시한다.

2. 관련 연구

이 장에서는 중복 데이터베이스 시스템에서 거래 관리를 위한 기법을 분류하고, 낙관적 동시성 제어 방법에 대하여 기술한다.

2.1 중복 데이터베이스에서 트랜잭션 관리

데이터베이스 중복 기법은 갱신 결과를 각 사이트에 전파하는 방법과 갱신하는 시점에 따라 Eager 기법과 Lazy 기법인 두 가지로 분류한다[7]. 먼저 Eager 기법은 클라이언트가 갱신을 요구했을 때 트랜잭션을 완료하기 전에 중복된 모든 사이트에서 동기화 시켜서 트랜잭션 연산들을 수행시키는 방법이다. 즉 모든 올바른 데이터베이스 사이트에서 갱신이 실행되어 완료하던가 아니면 하나도 갱신이 이루어지지 않게 하는 방법이다. 그러나 통신 부담과 응답시간이 너무 길어 트랜잭션이 지연되는 문제가 발생한다. Lazy 기법은 Eager 기법에서 발생되는 동기화에 대한 통신부담을 피하기 위하여 제안된 기법이다. 트랜잭션의 수행은 중복된 데이터를 제어하는 서버에서만 수정이 일단 완료되면 그 트랜잭션의 완료를 허용하며 이후 통신 트래픽이 발생되지 않는 적당한 시기에 서버의 책임 하에 갱신 결과를 전파하여 연산을 수행하는 방법[6]으로 약한 일관성을 보장하고, 각 사이트에서 수행되는 트랜잭션의 완료하는 시점이 다르므로 전역인 일관성을 유지하지 못한다는 문제가 발생하게 된다. 즉 이 방법은 1-사본 직렬가능성을 보장하지 못하기 때문에 강한 일관성을 필요로 하는 응용에는 적합하지 않다.

Eager 기법은 두 가지 변형을 제공한다[2, 6]. 첫 번째, 클라이언트가 트랜잭션을 제출하면 그 트랜잭션이 실행하는 동안 제출된 트랜잭션을 모든 데이터베이스 사이트로 즉시 전파하여 클라이언트 요구를 방송하여 갱신하도록 한다. 두 번째, 트랜잭션이 제출된 사이트에서 클라이언트 요구 전부를 처리한다. 그리고 클라이언트가 완료연산을 요구했을 때 갱신연산을 다른 데이터베이스 사이트로 전파하여 갱신하도록 한다[19].

또한 Eager 기법에 따라 수행되는 갱신 요구를 어떤 사이트에서 갱신하느냐에 따라 다음 두가지로 분류할 수 있다. 첫 번째는 데이터의 주사본을 가지고 있는 사이트를 마스터라 했을 때 마스터 사이트에서만 갱신을 수행하는 기법이 있고, 두 번째는, 모든 사이트를 그룹으로 묶어서 마스터 사이트가 없이 어떤 사이트에서도 갱신 요구를 클라이언트로부터 수신 받아 갱신할 수 있는 기법이 있다. 즉 마스터사이트를 두는 갱신은 주 사본에 해당되는 마스터 사이트에서만 수정이 이루어지므로 통신부담과 병목현상이 발생하게 된다[7].

2.2 낙관적 동시성 제어

대부분의 상용 데이터베이스 시스템에서는 2PL(Two Phase Locking) 규약을 트랜잭션들의 동시성 제어를 위하여 사용되고 있음에도 불구하고 낙관적 동시성 제어 기법에 많은 관심이 모이고 있다[15, 16]. 그 이유는 충분한 하드웨어 자원을 이용할 수 있다면 낙관적 동시성 제어는 2PL보다 더욱 향상된 트랜잭션의 처리 성능을 제공하기 때문이다[17]. 이러한 결과 트랜잭션의 높은 처리량을 보여줌에 따라 다중 프로그래밍 수준을 증가시킨다. 중복 데이터베이

스 환경에서 고려해야 하는 점은 메시지 전달에 의한 중복된 데이터베이스의 접근 비용을 고려해야 한다. 사실 [7]에서의 연구는 중복된 데이터베이스를 완전히 동기화 하여 접근하게 되면 시스템에서 많은 데이터베이스 사이트가 교착상태를 발생시킬 수 있고, 데이터가 중복된 모든 사이트가 잠금으로 인하여 트랜잭션의 대기 시간이 너무 길어질 수 있다는 문제점을 내포하고 있다는 것을 보여주었다.

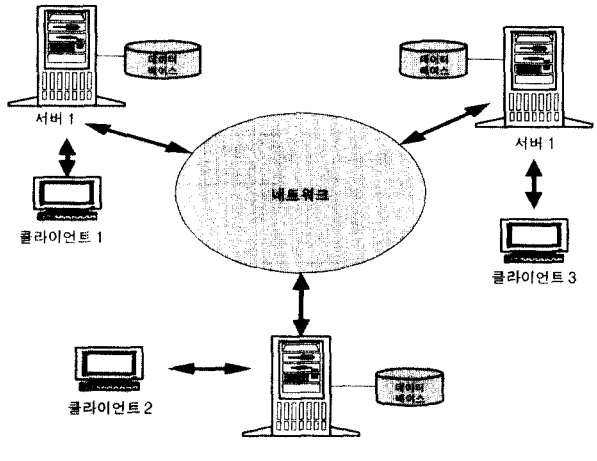
Eager 기법은 read-one/write-all 기법을 사용하여 판독은 트랜잭션이 발생된 사이트에서 지역적으로 수행하고 갱신은 모든 사이트를 동기화 시켜서 수행한다. 이 방법중 하나인 원자적 방송(Atomic Broadcast : ABCAST)을 기본으로 한 중복 데이터베이스 관리방법은 그룹 통신 프리미티브(group communication primitive)의 사용을 제안하였다. 그러나 실제적인 솔루션[4, 11, 12]에서 그룹 통신 프리미티브를 사용하고 있지만 충분한 깊이를 제공하지 못하는 문제가 있다. 이 방법을 사용한 [11]에서는 분산 시스템의 견고성을 증가시키기 위해서 근거리 통신망으로 국한된 그룹 통신 프리미티브(group communication primitive)를 제안하였다. 이 프리미티브는 원자적 방송 프리미티브의 의미를 사용하였으나 성능을 향상시키기 위한 구현 전략과 기본적인 병목현상(bottleneck)으로 인하여 메시지를 모든 사이트에 전달하기 전에 조정단계가 필요하고, 이 조정단계가 끝나기 전까지는 메시지를 전달할 수 없으므로 메시지가 지연되는 문제가 발생하게 된다. 이를 해결하기 위해서 [5]에서는 갱신 트랜잭션을 중복 사이트에서 즉시 전달하고, 수행한 후에 전달된 순서대로 수행을 하였는가 즉 그 트랜잭션들의 전역 순서가 존재하는지를 검사하는 낙관적인 방법을 제안하였다. 이 방법을 확장하고 발전시킨 방법으로 [10]에서는 원자적 방송의 조정단계가 트랜잭션의 수행과 병행하여 이루어지게 함으로써 성능을 향상시킬 수 있는 방법을 제안하였다. 방송한 순서로 각 사이트에 정보가 전달된다는 가정 하에 해당사이트에서 갱신 트랜잭션들을 수행시키고 방송순서대로 갱신 결과가 각 사이트에 전달되어 수행되었는가를 검사하는 낙관적인 방법이다. 이 방법은 제출된 사이트에서 판독 연산을 수행하고 기록 연산을 모든 사이트에서 수행하게 하는 방법보다 판독과 기록 연산 모두가 중복 사이트에서 수행되어야 하기 때문에 중복데이터베이스의 특징중의 하나인 병행성이 저하된다. 또한, 각 사이트에서 동시에 수행되는 갱신 트랜잭션의 수가 많을 경우에 트랜잭션들의 잠정수행순서(tentative order)와 확정수행순서(definitive order)가 달라질 가능성이 많아 철회되는 트랜잭션의 수가 많아지게 된다.

따라서 본 논문에서는 Eager 중복기법을 기반으로 한 완전 중복 데이터베이스 시스템 환경에서 각 사이트에 제출된 트랜잭션을 독립적으로 수행하면서 병행성을 향상시키기 위한 낙관적인 동시성 제어 기법을 제안하고 모든 사이트에서 트랜잭션이 일관되게 처리될 수 있도록 원자적 방송을 기반으로 한 새로운 OCCRD(Optimistic Concurrency Control for Replicated Database)알고리즘을 제안한다. 제안한 알고리즘은 원자적 방송 프리미티브에서 발생되는 조정단계에 의해 지연되는 문제점을 해결하고 직렬성을 보장한다.

3. 중복 데이터베이스 시스템 모델

이 장은 제안하고 있는 알고리즘이 토대를 두고 있는 중복 데이터베이스 시스템 구조와 각 지역 사이트에서 메시지를 전달하기 위하여 사용되는 원자적 방송에 대하여 기술한다. (그림 1)과 같이 시스템 구조는 분산 시스템을 기반으로 하여 지역 데이터베이스 시스템들이 네트워크 망에 연결되어 있다. 각 클라이언트는 그의 지역 데이터베이스의 서버에게 트랜잭션의 처리를 요구할 수 있다. 각 서버에 연결되어 있는 데이터베이스는 데이터를 완전 중복하고 있고, 지역 데이터베이스 시스템은 중복된 모든 데이터베이스들과 협력하여 클라이언트가 그 지역 데이터베이스 시스템에 제출한 트랜잭션을 관리한다.

중복 데이터베이스에서 트랜잭션은 트랜잭션이 생성된 원래의 사이트에서 수행되는 주프로세서와 트랜잭션이 요구하는 데이터가 중복되어 있는 다른 사이트에 있는 부 프로세스들로 구성되고[9], 트랜잭션의 연산들은 순차적으로 실행된다. 중복 데이터베이스 시스템은 정확성 검증을 위해 1-사본 직렬성을 만족하도록 한다. 1-사본 직렬성은 트랜잭션들에 대한 동시성 제어와 데이터의 복사본들에 대한 상호 일관성 유지를 위한 정확성 기준을 사용된다.



(그림 1) 시스템 구조

각 서버는 제출된 트랜잭션을 지역적으로 처리하고, 완료 시점에서 모든 다른 서버에게 갱신메시지를 방송한다. 이것은 원자적 방송 프리미티브를 적용해서 실행한다. 원자적 방송은 방송한 순서로 모든 수신서버는 메시지를 받았다는 것을 의미한다. 이를 위하여 송신서버는 먼저 메시지를 받아서 다른 서버와 순서를 조정한 후 그 메시지를 전달한다.

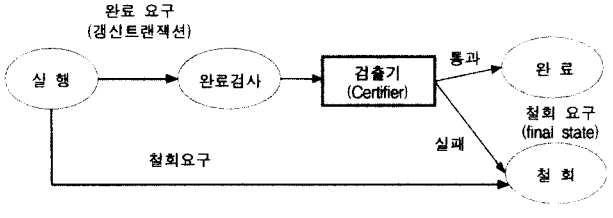
ABCAST 프리미티브를 이용하여 메시지 m을 전송하는 이벤트를 ABCAST(m)이라 표기하고, ABCAST(m)은 다음과 같은 3가지 특성을 만족한다. 첫 번째, 만약 서버 s_i와 s_j에서 각각 ABCAST(m1), ABCAST(m2)를 방송한다면 그들은 동일한 순서로 각 서버에 전달되는 것을 보장한다. 즉, 모든 서버에 m2전에 m1이 전달되거나 m1전에 m2가 전달되어진다. 두 번째, 원자성(atomicity)이다. 만약 한 서버 s에서 메시지 m을 전달한다면 모든 정당(correct)한 서버는 메시지 m을 수신한다. 중복 데이터베이스에서 트랜잭션은 모든 사이트에서 전부 또는 전무(all-or-nothing) 형태로 수행되어 트랜잭션의 원자적 종료 특성을 보장한다.

4. 중복 데이터베이스에서 낙관적 동시성 제어 알고리즘

이 장에서는 Eager 기법을 기반으로 한 중복 데이터베이스 환경에서 각 사이트에 제출된 트랜잭션을 독립적으로 수행하여 병행성을 향상시키기 위한 동시성 제어 기법을 제안한다. 제안하는 기법은 중복 데이터베이스에서 원자적 방송을 기반으로 한 낙관적 동시성 제어 알고리즘인 OCCRD(Optimistic Concurrency Control Algorithm for Replicated Databases)이다. 갱신트랜잭션이 처리되는 진행 과정을 나타내는 트랜잭션의 상태 전이도와 OCCRD 알고리즘을 기술한다.

4.1 낙관적인 방법에서 갱신 트랜잭션의 상태 전이도

중복 데이터베이스 환경에서 낙관적인 방법을 이용한 갱신 트랜잭션의 처리는 실행단계, 완료검사 단계, 완료 또는 철회 단계를 거쳐서 수행한다. 이에 대한 트랜잭션의 상태 전이도는 (그림 2)와 같다.



(그림 2) 트랜잭션 상태 전이도

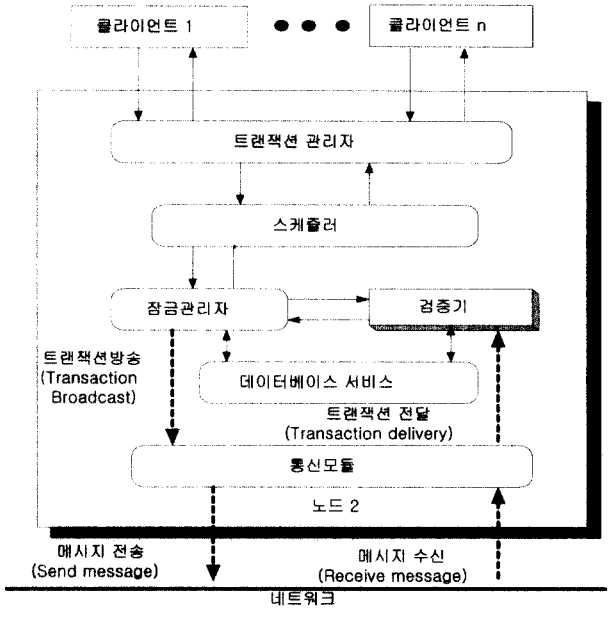
각 사이트의 서버는 제출된 트랜잭션에 유일한 타임스탬프를 부여하고 실행 상태에 들어간다. 실행 상태는 클라이언트가 요구한 판독, 기록 연산들을 지역적으로 수행하는 상태로서 클라이언트가 트랜잭션의 판독/기록을 요구하면 데이터베이스에 요구한 데이터에 접근하여 연산들을 수행한다. 트랜잭션이 완료를 요구할 때까지가 실행상태이다.

완료검사 상태는 클라이언트가 완료를 요구하면 갱신메시지를 복사본이 있는 모든 사이트로 방송하고, 동시에 수행되는 갱신 트랜잭션의 일관성을 보장하기 위하여 트랜잭션들의 직렬성 검증을 통해 트랜잭션의 완료와 철회를 결정한다.

최종상태는 직렬성 검사를 마친 트랜잭션이 완료로 결정되어 갱신연산들을 수행시키는 완료상태와 직렬성이 위배된 트랜잭션이 철회되는 철회상태로 구분된다. 만약 트랜잭션은 실행 상태에서 클라이언트가 철회를 요구하면 바로 철회상태로 들어갈 수 있다.

4.2 중복 데이터베이스 시스템에서 각 사이트의 노드 구조

이 절에서는 중복 데이터베이스 환경에서 트랜잭션의 상태 전이도에 따라 수행되는 갱신 트랜잭션을 수행하기 위해서 필요한 각 사이트의 노드 구조에 관하여 기술한다. 각 노드의 구성요소는 (그림 3)과 같이 트랜잭션관리자, 스케쥴러, 잠금관리자, 검증기, 데이터베이스 서비스, 원자적 방송모듈로 구성된다.



(그림 3) 중복데이터베이스 시스템에서 노드의 구조

트랜잭션 관리자는 클라이언트가 제출한 트랜잭션을 받아서 스케쥴러에게 전달하고 트랜잭션의 수행결과를 클라이언트에게 보내는 역할을 수행한다. 트랜잭션의 입장에서 스케쥴러는 기점(originating) 스케쥴러와 원격(remote) 스케쥴러가 있는데, 기점 스케쥴러는 트랜잭션이 제출된 사이트의 스케쥴러이고 원격 스케쥴러는 트랜잭션의 수행에 참여하는 다른 사이트이다. 즉, 각 사이트의 스캐쥴러는 기점 스케쥴러가 되기도 하고 원격 스케쥴러가 되기도 한다. 따라서 스캐쥴러는 지역적으로 갱신 트랜잭션을 수행하기 위하여 잠금 관리자에게 잠금을 요구하고 클라이언트가 트랜잭션 관리자에게 완료를 요구하면 스케쥴러는 통신모듈을 통하여 중복 된 모든 사이트로 갱신 메시지를 방송한다. 이때 갱신 메시지의 정보는 (Tr-id, RS(T_i), WS(T_i, value), TS(T_i), last_SN)값을 포함하고 있다. RS와 WS는 지역 사이트에서 동시에 수행되는 트랜잭션들의 충돌 관계를 알기

위해서 필요한 자료로써, $RS(T_i)$는 트랜잭션 Ti의 판독집합이고, $WS(T_i,value)$는 기록과 기록될 값을 나타내는 기록집합이다. 그리고 타임스탬프 값 $TS(T_i)$은 트랜잭션의 순서를 결정하기 위해서 필요하고, last_SN (Sequence Number)는 현재 트랜잭션이 제출된 사이트에서 가장 최근에 검증단계를 통과한 트랜잭션의 순서번호이다. 만약 검증기를 통과한 트랜잭션이 직렬가능 한 순서가 있다면 완료 상태로 전이된다. 그러나 검증단계가 실패되면 그 트랜잭션은 철회 상태로 전이되면서 기점 스케쥴러에 의해서 철회된다. 또한 판독연산은 제출된 사이트에서 수행하고, 기록 연산들은 모든 사이트에서 기록이 이루어지도록 한다. 이들 구성요소들 사이에서 트랜잭션의 상태 전이도에 따라 낙관적인 방법에 의해서 수행되는 갱신트랜잭션의 처리 절차는 다음 (그림 4)와 같다.

(그림 4)에서 처럼 각 클라이언트는 트랜잭션을 트랜잭션 관리자에게 제출하면 트랜잭션 관리자는 그 갱신 트랜잭션의 연산들을 기점 스케쥴러에게 스케쥴하도록 한다. 기점 스케쥴러는 잠금 관리자에게 연산들을 수행하기 위한 잠금을 요구하여 판독연산을 수행하고 기록연산은 트랜잭션의 독자적인 작업공간으로 수행한다. 만약 클라이언트가 완료를 요구하면 모든 복사본 사이트에서 갱신될 수 있도록 갱신 메시지를 원자적 방송 성질을 만족하는 통신 모듈에 의해서 방송한다. 각 갱신 메시지를 전달받은 사이트는 갱신 트랜잭션이 지역 사이트에서 갱신 가능한가를 검사하여 지역적으로 기록 잠금(write lock)을 획득하여 수행을 시작하게 한다. 즉 원자적인 작업단위로 기록연산을 처리한다. 지역 데이터베이스에서 동시성제어는 2PL잠금 방법[7]에 따라 트랜잭션을 관리한다. 중복 데이터베이스의 정확성 기준은 1-사본 직렬성을 기준으로 하며 이는 트랜잭션의 일관성을 보장한다.

4.3 OCCRD 알고리즘

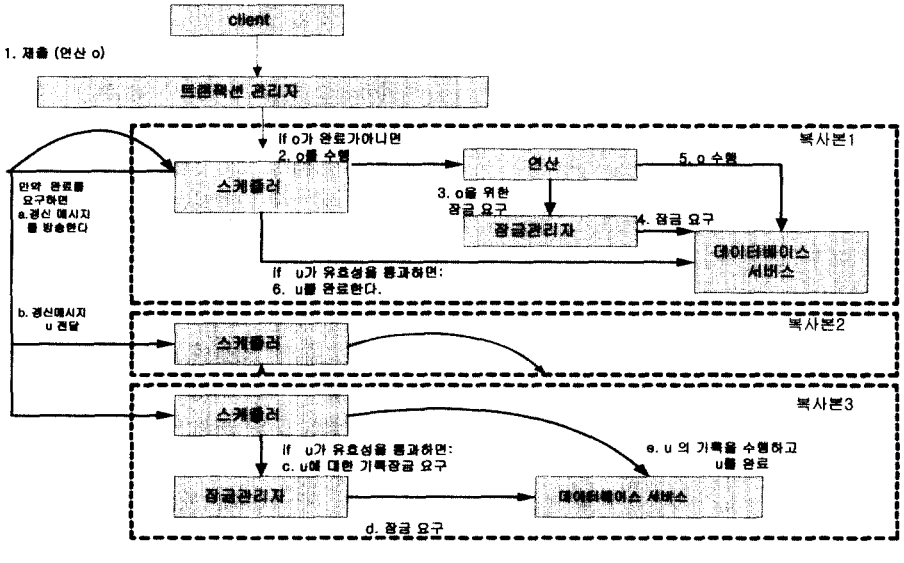
이 절에서는 데이터가 완전 중복된 시스템에서 수행되는 갱신 트랜잭션들 사이의 일관성을 유지할 수 있는 중복 데이터베이스를 위한 낙관적 동시성 제어 알고리즘인 OCCRD을 제안한다.

제안하는 OCCRD에서 클라이언트는 모든 사이트로 트랜잭션을 제출할 수 있으며 모든 복사본 사이트의 데이터 항목에 동일한 값을 갱신하도록 한다. 이를 위해 트랜잭션의 검증은 모든 사이트에서 동시에 수행되는 갱신 트랜잭션의 직렬성을 유지하면서 기록연산을 수행할 수 있도록 연산들 사이의 충돌관계를 검사한다. 두 트랜잭션이 충돌이 발생된다는 것은 동일한 데이터에 대해서 수행되는 연산중에서 하나가 기록연산인 경우이다. 만약 두 개의 서로 다른 사이트에서 동시에 수행되는 트랜잭션이 충돌을 발생시킨다면 그중 한 개의 트랜잭션은 철회시켜야 한다.

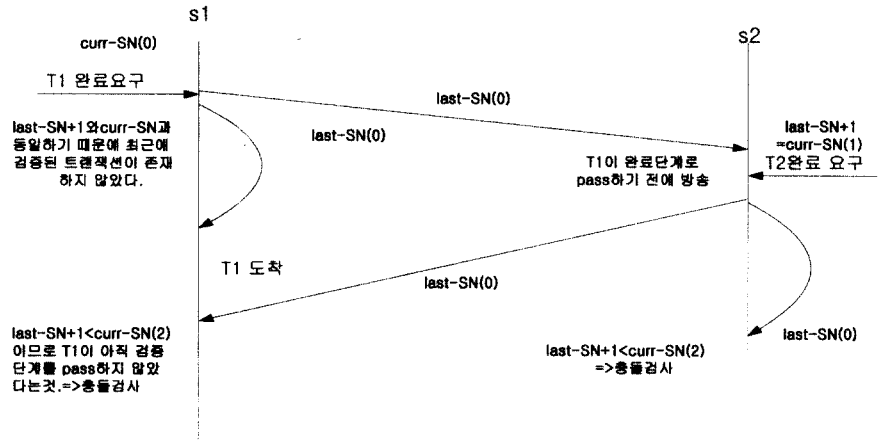
트랜잭션 T1은 사이트 S1으로 제출되고, 트랜잭션 T2는 사이트 S2로 제출되었다고 가정하자. 사이트 S1에서는 트랜잭션 T1과 T2가 연산들 사이에 충돌이 발생하는가를 검사하여 트랜잭션의 순서를 결정해야 한다. 트랜잭션의 순서는 현재 수행되고 있는 갱신 트랜잭션들의 상태에 따라 트랜잭션의 순서가 정해지므로 이를 위한 조건은 다음과 같다.

경우1. S1에서 트랜잭션 T1이 실행 상태에 들어가기 전에 트랜잭션 T2가 완료 상태를 통과한다면, 트랜잭션 T1은 T2와 인터리빙이 존재하지 않기 때문에 검증 단계를 통과하는 경우이다. 즉, 트랜잭션 T2의 WS(Write Set)와 T1의 RS(Read Set) 사이에 교집합이 존재하지 않는 경우이다.

경우2. S1에서 트랜잭션 T1이 실행 상태 일 때 T2가 완료 상태를 통과한다면 S1에서의 지역 동시성 제어메커니즘에 의해서 직렬성에 위배되지 않으면 T1은 직렬성 검증단계를 통과한다.



(그림 4) 낙관적인 방법을 적용한 갱신 트랜잭션 처리단계



(그림 5) 순서번호에 의한 직렬성 검사

경우3. S1에서 트랜잭션 T1이 실행상태이고, T2가 완료검사 상태에 있을때 T2가 T1의 트랜잭션의 연산들과 충돌이 발생한다면 지역 스케줄러에 의해서 직렬성이 보장된다. 즉 T1이 판독한 데이터의 값은 트랜잭션 T2가 갱신하기 전의 값을 판독하고 만약 T2가 갱신한다면 지역 스케줄러에 의해서 트랜잭션 T1은 철회된다.

경우4. 트랜잭션 T1이 완료검사 상태일 때 T2가 S1에서 완료 상태를 통과한다면 즉, T1이 T2 이후에 전달되었다는 의미와 같다. 따라서 WS(T2)와 RS(T1)사이에 교집합이 존재하지 않으면 T1은 완료되고, 교집합이 존재하면 T1의 직렬성 검증은 실패되며, 그 트랜잭션을 철회시켜야 한다. 그러나 T2가 판독한 데이터를 T1이 갱신하는 판독-기록 충돌만을 발생시킨다면 T2와 T1의 트랜잭션의 순서를 재조정시킨다. 또한 판독-기록충돌이외에 기록-기록 충돌이 발견되면 이러한 충돌은 재조정시킬 수 없으므로 즉시 철회되거나 재 시작시킨다. 그러나 동일한 데이터에 대하여 기록-기록 충돌만을 발생한다면 이러한 트랜잭션은 철회시키지 않고 타임스탬프 값을 비교해서 타임스탬프 값이 가장 큰 트랜잭션의 연산이 수행되도록 한다. 따라서 철회될 트랜잭션을 완료시킴에 따라 트랜잭션의 완료율을 향상시킬 수 있다. 이에 대한 예는 예제1과 같다.

【예제 1】 트랜잭션들의 직렬화 순서 재조정의 예

갱신트랜잭션 T_i와 T_j가 각각 사이트S1과 S2에서 다음과 같이 수행한다고 하자.

S1 : T_i : r (x)r (y)w(z)c	RS(T_i) = {x, y}
	WS(T_i) = {z}
S2 : T_j : r (x)w(y)w(z)c	RS(T_j) = {x}
	WS(T_j) = {y, z}

사이트 S2에서 T_j가 완료상태로 전이하기 전에 T_i가 완료검사 상태를 시작한다고 하면 T_i의 판독 데이터 아이템 y를 T_j에서 갱신하였으므로 T_i 트랜잭션은 철회(abort)된다. 그러나 두 트랜잭션의 수행순서가 바뀐다면 이 두 트랜잭션들은 판독 의존관계(read-from)가 성립되지 않으므로 철회되지 않고 완료할 수 있다. 즉 T_i가 기록한 어떤 데이터도 T_j에서 판독하지 않았기 때문이다. 이때 z에 대한 기록연산은 타임스탬프 값이 비교되어 타임스탬프 값이 더 큰 T_j의 연산이 실행된다. □

예제 1을 통하여 알 수 있듯이 철회될 트랜잭션의 완료순서를 재조정함에 따라 트랜잭션의 완료율을 향상시킬 수 있다. 그리고 각 사이트에 제출된 갱신트랜잭션은 실행 상태에서 수행하고 수행된 결과를 모든 사이트에 반영하기 때문에 서로 다른 사이트에서 수행되는 트랜잭션 사이에 충돌연산이 발생할 수 있으므로 모든 사이트에서 트랜잭션의 수행을 동일하게 수행토록 하기 위하여 직렬성을 검증하는 단계에서 다른 복사본에서 수행되는 트랜잭션 수행상태에 대한 정보를 필요로 한다. 즉 사이트 S2에서 검증알고리즘은 만약 T1이 완료 검사 상태로 전이했을 때 사이트 S1에서 T2가 이미 검증이 끝났다는 것을 알아야 한다. 이러한 정보는 순서번호(Sequence Number : SN)를 사용하여 처리한다. 각 사이트는 두 개의 순서번호를 나타내는 last_SN과 curr_SN의 값을 유지한다. last_SN은 갱신하는 트랜잭션을 모든 사이트로 방송하기 전에 가장 최근에 검증단계를 통과 한 트랜잭션의 순서번호이다. 이 순서번호는 갱신 메시지 m에 포함시켜 방송(ABCAST(m))하고, Curr_SN은 각 사이트에 도착하는 갱신메시지의 순서번호인데, 모든 사이트에서 동일한 순서번호를 유지한다. 이러한 성질은 원자적 방송이 가지는 성질에 의해서 만족한다. 즉 원자적 방송은 서로 다른 사이트 S1과 S2에서 각각 메시지 m과 m'를 보내었을 때 m전에 m'을 보내면 m전에 m'이 도착한다는 성질을 가지므로 갱신 메시지에 대해 모든 복사본은 동일한 SN의 값을 가지며 갱신 메시지가 모든 복사본으로 방송했을 때 가장 최근에 검증단계를 통과한 갱신 메시지에 대한 순서번호를 함께 피기백(piggy-backed)하여 검사하도록 한다.

위의 예제 1에서 순서번호를 적용하면 사이트 S1에서 갱

신 트랜잭션 T1에 대하여 갱신 메시지를 방송하는 시점에서 가장 최근에 검증된 트랜잭션이 존재하지 않으면 SN의 값은 '0' 이 피기백된다. 각 사이트에 순서번호(last_SN)를 포함한 갱신 메시지가 도착했을 때 last_SN+1과 curr_SN의 값과 비교하여 이 두 값이 일치하지 않는다면 갱신 메시지를 보낸 사이트에서는 다른 트랜잭션이 완료 검사 상태이거나 완료 상태에 있다는 것이다. 그러므로 이들 사이에 발생할 수 있는 연산들 사이에 충돌검사를 하여 트랜잭션의 순서를 결정하여야 한다. 그러나 이 두 값이 동일하면 완료 검사 상태를 통과한 트랜잭션이 없다는 것을 나타내므로 충돌검사를 하지 않고 트랜잭션의 상태를 완료상태로 바로 전이하여 각 지역 스케줄러에게 제출하여 갱신 메시지를 처리하도록 한다. 위의 예제 1을 순서번호를 적용하여 나타내면 (그림 5)와 같고 이에 대한 직렬성 검사와 완료 순서를 재조정시키는 알고리즘은 (알고리즘 1)과 같다.

```
/* serializability_procedure */

/* last_SN(사이트_id, Ti) : 가장 최근에 완료 검증 단계를 통과한
                            트랜잭션의 SN
   curr_SN(사이트_id, Tj) : 각 사이트의 통신 모듈에서 검증기로
                            전달한 트랜잭션의 순서번호이며, 초기
                            값은 0이다.
   WS, RS : 트랜잭션 연산의 판독과 기록연산들의 집합
   T : 현재 제출된 트랜잭션
   Ti : T보다 먼저 방송된 트랜잭션  */

     curr_SN = curr_SN + 1
     if (last_SN + 1 == curr_SN)
     then return 'serializability passed'
     else if (WS(Ti) ∩ RS(T) ≠ 0)
          then {
               if(RS(Ti) ∩ WS(T) ≠ 0)
               then abort T   /* 사이클이 발생  */
               else if (WS(Ti) ∩ WS(T) ≠ 0)
                    then {
                         if (Ts(Ti) < Ts(T))
                         then Ti ignore
                          else execution ordering of the
                              transaction is T → Ti
                         }          /* 트랜잭션의 수행 순서가 T
                                       →Ti의 순서를 가짐
               }
          else if (RS(Ti) ∩ WS(T) ≠ 0)
               then abort T
               else if (WS(Ti) ∩ WS(T) ≠ 0)
                    then if (Ts(Ti) < Ts(T))
                         then Ti ignore
                         else if (Ti has been committed)
                              then T is aborted
                              else execution ordering of the
                              transaction is T→ Ti
                    else return 'serializability passed'
```

(알고리즘 1) 갱신 트랜잭션의 직렬성 검사를 위한 검출기

(알고리즘 1)을 통하여 직렬성 검증을 마친 갱신메시지는 실제 데이터 값을 기록하기 위하여 완료 상태로 전이된다. 완료 상태로 전이되는 트랜잭션은 다음 두 가지 방법 중 하나로 처리한다. 먼저 트랜잭션 T_i에 대한 갱신 메시지가 만들어진 사이트라면 그 트랜잭션은 이미 기록 연산을 수행했기 때문에 기록연산은 데이터베이스에 반영되고 완료시킨다. 아니면 시작 연산은 데이터베이스 서비스에서 기록 트랜잭션이 시작되고 갱신 메시지에 유지된 모든 기록연산들에 대한 기록 잠금을 잠금 관리자에게 요구한다. 이러한 동작은 중복된 모든 사이트에서 원자적 방송 프리미티브의 성질에 의해서 동일하게 수행되고, 그 잠금 관리자는 그 잠금을 승인할 수 있으면 즉시 잠금을 부여한다. 그러나 트랜잭션 T_i가 요구한 데이터 항목에 잠금이 설정되어 있다면 트랜잭션 T_j는 대기 큐에 유지된다. 지역 사이트에서 수행되는 트랜잭션과 갱신 메시지에 의한 연산들과의 충돌이 발생하면 각 지역의 잠금 관리자는 갱신 트랜잭션에게 잠금을 승인할 수 있도록 하기 위하여 지역 트랜잭션을 철회시킨다. 만약 지역적으로 수행되는 트랜잭션 T_i가 완료검사 상태에 있고 다른 갱신 트랜잭션 T_j의 연산들과 충돌되는 기록 잠금을 유지하고 있다면 이 트랜잭션 T_i는 현재 수행되는 T_j 트랜잭션의 갱신메시지 후에 방송되었다는 것을 알 수 있다. 따라서 트랜잭션 T_i가 검증이 끝날 때까지 갱신 메시지는 지연된다. 그러나 트랜잭션 T_j가 철회된다면 즉시 기록 연산을 적용하고, T_i가 완료 검사 상태를 통과하면 그 기록 연산은 수행될 수 있으나 트랜잭션 T_j가 기록한 값에 중복 기록되므로 이러한 경우는 타임스탬프가 가장 큰 값의 연산 값으로 갱신된다.

지금까지 위 절에서 논의된 갱신 트랜잭션 관리를 위한 OCCRD 알고리즘은 (알고리즘 2)와 같다.

```
/*  지역트랜잭션 관리 알고리즘  */
   T : 지역 사이트로 제출된 트랜잭션
   UM(Tr-id, RS(Ti), WS(Ti value), TS(Ti), last_SN) : 갱신 메시지
   Ti : Ti의 갱신 메시지를 방송하기 전의 지역 트랜잭션
The lock manager of each node N controls and coordinates the
opertion requests of a transaction T in the following manager  */

1. Executing state : /* 지역 사이트에서 실행상태  */
   (a) When a transaction T is submitted to local server
       Assign time_stamp TS to the transaction T
   (b) When a read_only transaction T or update transaction T
       requests a new lock
       if (there is no conflict with setting lock)
       then {
            get a lock
            perform the operations of transaction T
            Update RS and WS
            }
       else the T is blocked

2. Broadcast state : /*  방송상태  */
   If (T is not read-only transaction)
```

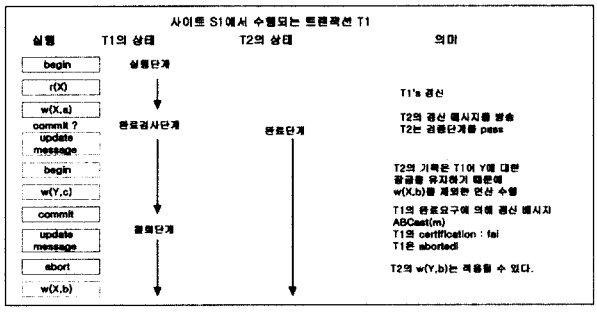
```
then {
    broadcast update message and last_SN to the all sites
                /* RS, WS, TS, last_SN */
    change the executing state of transaction to committing state
    }
else  commit.

3. Committing State : /*  완료검사상태  */
when a update massage received
    inwoke serializability_procedure

4. Commit State : /*  완료상태 : 지역 스케줄러에게 갱신 메시지
                    의한 기록연산을 수행  */
if(serializability passed)
then {
    change the committing state of transaction to commit state
    Upon delivery of WS request, locks for all data item in
        WS in an atomic step
    (a) For each operation W(X) in WS
        i. If there is no lock on X, grant the lock.
        ii. If there is a write lock on X or all read locks
            on X are from transaction that have already
            processed their commit state, then wait
            until all other locks are release and the new
            lock can granted.
        iii. If there is a granted read lock r(X) and the
            write set WS of T has not yet been delivered,
            abort Ti and grant w(X).
            if T has already been sent, then broadcast
            abort message
    (b) Whenever a write lock is granted, perform the
        corresponding operation.
}
else abort
```

(알고리즘 2) 지역 트랜잭션을 관리하는 OCCRD 알고리즘

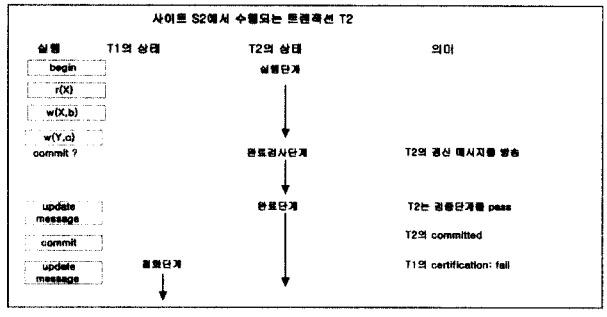
【예제 2】 사이트 S1과 S2(S1 ≠ S2)에서 각 갱신 트랜잭션 T1과 T2가 동시에 수행한다고 하자. 지역적으로 수행되는 갱신 트랜잭션은 다음과 같다.

S1 : T1 : r(x)w(x)c	RS(T1) = {X}
	WS(T1) = {X}
S2 : T 2 : r(x)w(x)w(y)c	RS(T2) = {X}
	WS(T2) = {X, Y}

갱신 트랜잭션 T1과 T2가 각각 사이트 S1과 S2에서 수행되는 과정은 (그림 6)과 (그림 7)에서 보여주고 있다.



(그림 6) 사이트 S1에서 수행되는 트랜잭션 T1



(그림 7) 사이트 S2에서 수행되는 갱신 트랜잭션 T2

위의 (그림 6)과 (그림 7)에서와 같이 갱신 트랜잭션 T1과 T2가 동시에 수행했을 때 T2의 갱신 메시지가 T1전에 전달되었다면 아직 검증 단계를 통과한 트랜잭션이 없으므로 T2가 검증 단계를 통과하도록 하고 사이트 S2에서 T2를 완료시킨다. 그러나 S1에서는 T2의 갱신 연산들을 집합을 나타내는 WS의 요소를 적용하려고 하지만 T1이 아직 X에 대한 잠금을 유지하고 있기 때문에 T1의 검증의 결과를 알 때까지 T2의 기록연산 W(x,b)는 지연된다.

사이트 S2에 T1의 갱신 메시지가 전달되었을 때 T1은 양쪽의 복사본 상에 검증시험을 통과할 수 없다. S1에서 트랜잭션의 상태는 T1이 완료검사 상태 안에 있을 때 T2는 완료상태를 통과하므로 사이트 S1에서 갱신 트랜잭션 T1의 RS와 T2의 WS사이에 X 항목의 교집합이 발생하게 되므로 갱신 트랜잭션 T1은 철회된다. 따라서 T2는 데이터 항목 X에 대한 잠금을 얻어서 지연된 연산 w(X, b)는 실행되고, 사이트 S2에서도 T1에 대한 갱신메시지의 실행이 실패되므로 T1을 철회시킨다.

따라서 모든 사이트가 같은 결정을 내리기 위해서는 직렬성 검증을 하기 위해서 다른 사이트 상에 발생된 사건에 대한 정보를 필요로 한다. 즉 사이트 S2에서 검증 알고리즘은 만약 T1이 완료 검사 상태로 전이했을 때 사이트 S1에서 T2가 이미 검증이 끝났다는 것을 알아야 한다. 이러한 정보는 순서번호를 사용하여 처리한다. 사이트 S1에서 갱신 트랜잭션 T1에 대하여 갱신 메시지를 방송했을 때 가장 최근에 검증된 트랜잭션이 존재하지 않기 때문에 그 값은 '0' 이 피기백된다. 갱신 트랜잭션 T2에 대한 갱신메시지가 사이트 S2에 전달되었을 때 SN은 '1' 을 부착시키고, T1에 대한 갱신 메시지는 '2' 를 부착시킨다. 이러한 정보를 이용하여 사이트 S2에서의 검증시험은 S1에서 T2가 검증이 완료되기 전에 T1이 완료검사 상태를 통과하였다는 것을 알 수 있다. 왜냐하면 T2의 순서번호는 piggy-backed(0)의 값보다 더 큰 값인 '1' 을 가지기 때문이다. □

다음으로 제안하는 알고리즘의 정확성을 위해 알고리즘이 전역 교착상태가 발생하지 않는다는 것과 트랜잭션의 직렬성을 보장하는지에 대하여 살펴본다.

【정리 1】 제안한 OCCRD 알고리즘은 전역 교착상태를 일으키지 않는다.

<증명> 갱신 트랜잭션 T_j가 잠금을 요구한다고 하자. 요구하는 잠금과 다른 트랜잭션 T_i에 의해서 설정된 잠금과의 충돌이 발생하는지를 먼저 조사하여야 한다. 충돌이 발생할 수 있는 경우는 다음과 같다.

경우 1. 갱신 트랜잭션 T_i는 실행상태에 있는 트랜잭션으로서 X에 대해서 판독 잠금을 가지고 있고 아직 기록연산을 수행하기 전이라면 트랜잭션 T_i는 알고리즘 2에서의 실행상태에 있다고 할 수 있다. 이때 T_j가 기록 잠금을 요구하면 알고리즘 2에서 T_i는 아직 다른 사이트로 전달되기 전이므로 철회시키고, T_j는 기록 잠금을 얻어 기록연산을 수행한다.

경우 2. T_i는 X에 대해서 판독 잠금과 기록 잠금을 가지고 있고 이미 다른 사이트로 갱신 메시지를 방송하여 전달하였을 때 T_j가 기록 잠금을 요구한다면 T_j는 T_i의 수행이 끝날 때까지 기다린다. 이때 $T_i \rightarrow T_j$로 가는 에지가 생성된다.

경우 3. T_i가 $W_i(X)$에 대해서 기록 잠금을 가지고 있을 때, T_j가 $R_j(X)$을 요구한다면 트랜잭션 T_i의 갱신연산의 집합인 WS_i의 요소(element)를 처리한 다음 그 트랜잭션은 종료하고 데이터에 대한 잠금을 해제한 후에 $R_j(X)$에 대한 판독 잠금을 승인한다.

교착상태가 발생한다고 하자. 교착상태가 발생하면 대기 그래프(wait-for graph)안에 $T_j \rightarrow T_k \rightarrow T_l \rightarrow \cdots \rightarrow T_i \rightarrow T_j$와 같은 사이클이 존재하게 된다. 그래프에서 에지가 생성된다는 것은 경우2와 같은 경우에만 발생하게 된다. 즉, $T_j \rightarrow T_k$사이의 에지는 T_j가 기록 잠금을 가지고 있고 T_k가 판독 잠금을 요구했을 때 발생할 수 있고, $T_i \rightarrow T_j$의 에지는 T_j가 기록 잠금을 가지고 있고 T_i가 판독 잠금을 요구했을 때 발생할 수 있다. 그러나 트랜잭션 T_k가 기록하고자 하는 갱신 연산들을 트랜잭션 T_j의 갱신 연산전에 처리된다는 것을 의미하므로 이러한 수행은 알고리즘2의 4.a.iii에 따라 T_j는 철회된다. 그러므로 사이클은 존재하지 않으며, 교착상태는 발생하지 않는다. □

【정리2】 제안한 OCCRD 알고리즘은 직렬성을 보장한다.

<증명> 제안하는 방법이 트랜잭션 직렬성을 보장하지 못한다면 SG(H)에 사이클이 발생하게 된다. 증명을 위해 SG(H)에는 사이클이 발생하지 않음을 보이면 된다.

증명을 위해 완료된 트랜잭션들 사이의 SG(H)에 $T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_{n-1} \rightarrow T_n \rightarrow T_1$과 같은 사이클이 발생한다고 하자. SG(H)에 에지 $T_1 \rightarrow T_2$가 생성된다는 의미는 다음과 같은 의미를 가진다.

- $RS(T_1) \cap WS(T_2) \neq \{\}$: T_2가 기록하기 전에 T1이 먼저 읽었다.
- $WS(T_1) \cap WS(T_2) \neq \{\}$: T_2가 기록하기 전에 T_1이 먼저 기록하였다.

그러나 $T_n \rightarrow T_i$ 사이에 에지가 생성되었다는 것은 3가지 다른 종류의 충돌로 인해 발생할 수 있다. 이들 충돌의 경우를 살펴보면 다음과 같다.

- (R_n, W_1) : 이러한 충돌의 발생은 T_n의 갱신 메시지가 T_1의 갱신 메시지 전에 모든 사이트에 전달되어야 한다는 것을 의미한다. 그렇지 않으면 T_n은 아직 다른 사이트로 갱신 메시지를 전달하기 전이므로 알고리즘 5에 의해서 철회된다. 따라서 SG(H)에는 에지가 만들어지지 않는다.
- (Wn, W_1) : 기록연산과 기록연산의 충돌의 발생은 T_n의 갱신 메시지가 T_1의 갱신메시지 보다 먼저 다른 사이트에 도착되었을 때 만 가능하다. 이러한 충돌은 가장 큰 타임스탬프 값을 가지는 기록연산이 최종적으로 데이터베이스에 기록하도록 한다.
- (Wn, R_1) : T_1은 T_n가 완료했을 때 T_n로부터 판독할 수 있다. 그러므로 T_n의 갱신메시지의 갱신 연산의 집합 WS_n이 전달된 후에만 가능하므로 이러한 충돌은 발생하지 않는다.

따라서 본 논문에서 제안한 알고리즘은 사이클을 발생시키지 않으며 트랜잭션의 직렬성을 보장한다. □

5. 결론 및 향후 방향

본 논문은 중복 데이터베이스에서 가용성과 병행성 및 성능을 향상시키기 위하여 원자적 방송 프리미티브를 사용한 낙관적 동시성 제어 방법을 제안하였다. 낙관적 동시성 제어 기법을 사용함에 따라 원자적 방송에서 조정하는데 필요한 시간만큼의 트랜잭션이 지연되는 문제를 해결할 수 있었다. 또한 트랜잭션의 병행성을 향상시키기 위하여 지역 사이트로 제출된 트랜잭션을 지역적으로 수행하고 수행된 결과만을 모든 사이트에 방송하여 직렬성을 파괴시키지 않고 독립적으로 수행되도록 하였다. 각 사이트에서는 동시에 수행되는 연산들의 순서를 보장하고 갱신트랜잭션의 전부 또는 전무(all-or-nothing)을 보장함에 따라 트랜잭션의 일관성을 보장할 수 있었다. 트랜잭션이 일관되게 갱신한다는 것은 1-사본 직렬성을 보장되어야 한다. 즉 모든 사이트가 서로 일관된 값을 유지하면서 동일한 순서에 의해 갱신거래를 수행한다는 것이다. 이를 위해 모든 사이트에서 트랜잭션이 유지되는 동일한 순서번호를 사용하여 OCCRD알고리즘에 의해 직렬성을 만족하도록 하였다. 또한 이미 완료검사 상태를 통과한 트랜잭션들과 충돌관계가 있는 트랜잭션들의 직렬화 순서를 재조정시킴으로서 트랜잭션의 철회율을 감소시킬 수 있었다.

제안한 OCCRD 알고리즘은 갱신 트랜잭션들 사이에 충돌 발생률이 적다면 좋은 성능을 보일 수 있다. 즉, 판독연산의 수가 기록 연산의 수보다 훨씬 많은 응용에서 판독연산의 국지성을 증가시킬 수 있기 때문이다.

향후 연구되어야 할 방향은 본 논문에서 제안한 OCCRD 알고리즘에의 통신비용을 줄이는 문제와 보다 효율적인 직렬가능 순서를 보장할 수 있는 방법에 대한 연구가 더 이루어져야 할 것이다.

참 고 문 헌

[1] D. Agrawal, G. Alonso, A. E. Abbadi, "Exploiting Atomic Broadcast in Replicated Databases," Inproccdings of EuroPar(EuroPar'97), Passau(Germany), 1997.

[2] P. Bernstein, V. Hadzilacos, N. Goodman, Concurrency Control and Recovery in Database System, Addison Weslay, 1987.

[3] B. Kemme, Gustavo Alonso, "A Suite of Database Replication Protocols Based on Group Communication Primitive," In Proceedings of the 18th International Conference on Distributed Computing Systems(ICDCS), Amsterdam, The Netherlands, May, 1998.

[4] B. Kemme, Fernando Pedone, "Processing Transactions over Optimistic Atomic Broadcast Protocols," In Proceedings of the International Conference on Distributed Computing Systems, Austin Texas, June, 1999.

[5] Fernando Pedone, A.Schiper, "Optimistic Atomic Broadcast," In Proceedings of the 16th International Symposium on Distributed Computing, DISC '98, WDAG, September, 1998.

[6] M. Wiesmann, Fernando Pedone, A. Schiper, B. Kemme, and G.Alonso, "Understanding Replication in Databases and Distributed Systems," In Proceedings of 20th International Conference on Distributed Computing Systems(ICDCS '2000), pp.264-274, 2000.

[7] J. Gray, P. Helland, D. Shasha, "The Dangers of Replication and a Solution," in Proc. of the ACM SIGMOD, pp.568-574, 1996.

[8] P. A. Benstein, N. Goodman, "An Algorithm for Concurrency Control and Recovery in Replicated Distributed Database," ACM Trans. on Database Systems, Vol.9, pp.596-615, 1984.

[9] J. Gray, P. Homan, H. Korth, and R.Obermark, "A Strawman Analysis of the Probability of Wait and Deadlock," Technical Report RJ2131, IBM San Jose Research Laboratory, 1981.

[10] B. Kemme, Fernando Pedone, Gustavo Alonso, Andre Schoper, "Using Optimistic Atomic Broadcast in Transaction Processing System," Technical Report, Department of Computer Science, ETH Zurich, March, 1999.

[11] B. Kemme, and G. Alonso. "A Suite of Database Replication Protocols Based on Group Communication Primitives," In Proceedings of the 18th International Conference on Distributed Computing System(ICDCS), Amsterdam, The Netherlands, May, 1998.

[12] A. Schiper and M. Raynal. "From Group Communication to Transaction in Distributed Systems," Communications of the ACM, pp.84-87, April, 1996.

[13] Yuri Breitbart and Henry F.Korth, "Replication and Consistency : Being Lazy Helps Sometimes," In Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Tucson Arizona, 1997.

[14] P. Chundi, D. J. Rosenkratz, and S. S. Ravi, "Deferred Updates and Data Placement in Distributed Databases," In Proceedings of the Twelveth International Conference on Data Engineering, New Orleans Louisiana, 1996.

[15] J. N. Gray and A. Reuter, "Transaction Processing : Concepts and Techniques," Morgan Kaufmann, 1993.

[16] H. T. Kung and J. T. Robinson. "On Optimistic Methods for Concurrency Control, ACM Transactions on Database Systems," Vol.6, No.2, pp.213-226, June, 1981.

[17] R. Agrawal, M. Carey, and M. Livny. "Concurrency Control Performance Modelling : Alternatives and Implications," ACM Transactions on Database Systems, Vol.12, No.4, pp. 609-954, December, 1987.

[18] D. Agrawal, A. El Abbadi, and R. Steinke. "Epidemic Algorithms in Replicated Database," In Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Tucson(USA), May, 1997.

[19] Fernando Pedone, Rachid Guerraoui, Andre Schiper, "The Database State Machine Approach," Technical Reports SSC/1999/008, Ecole Polytechnique Federal de Lausanne, Switzerland, March, 1999.

최 희 영

e-mail : hychoi@sunny.chonnam.ac.kr

1987년 목포대학교 전산통계학과(학사)

1989년 전남대학교 대학원 전산통계학과 (이학석사)

1999년 전남대학교 대학원 전산통계학과 (박사과정 수료)

1997년~현재 전남대학교 전산학과 시간강사

관심분야 : 중복 분산시스템, 전자상거래, 중복트랜잭션 관리, 실시간 시스템 분산처리시스템

황 부 현

e-mail : bhhwang@chonnam.chonnam.ac.kr

1978년 숭실대학교 전산학과(학사)

1980년 한국과학기술원 전산학과(공학석사)

1994년 한국과학기술원 전산학과(공학박사)

1980년~현재 전남대학교 전산학과 교수

관심분야 : 분산시스템, 분산 데이타베이스 보안, 객체지향 시스템, 전자상거래