韓國 컴퓨터情報學會 論文誌
第10卷 第5號, 2005. 11.

2005-10-5-1-6

멀티 데이터베이스 시스템에서 트랜잭션 관리를 위한 지연완료 기법 연구

이 상 희*

A Study of Deferred Commitment for Transaction Management in Multidatabase Systems

Sang-Hee Lee*

요 약

본 논문은 MDBS에서의 전역트랜잭션을 처리하기 위한 동시성 제어 기법을 제안한다. 전역 트랜잭션이 실행을 마치면 곧바로 완료시키는 대부분의 다른 기법과는 달리, 본 연구에서 제안한 기법은 기 결정된 지역 직렬화 가능 순서상 먼저 완료해야할 트랜잭션이 있을 경우에 그 트랜잭션이 완료할 때까지 트랜잭션 완료를 지연시킨다. 이러한 트랜잭션완료의 지연 덕분에 재실행되는 트랜잭션의 수를 줄일 수 있다. 제안된 기법의 성능은 시뮬레이션을 통해 참가하고 있는 LDBS중에 timestamp ordering(TO)방법을 지역 동시성 제어 기법으로 사용하고 있는 LDBS가 많을수록 좋은 성능을 보이는 것으로 측정 되었다.

Abstract

In this thesis, we propose a concurrency control scheme which schedules global transactions in multidatabase systems. Unlike other scheme which commit global transactions as soon as they have finished their execution, this scheme has attempted to delay them if their exist another global transactions that should be preceeded. Owing to the deferred commitment, the number of global restarts could be greatly reduced. This scheme's performance gain increased through simulations in case more than one LDBS employ timestamp ordering(TO) as a local concurrency control scheme.

▸ Keyword : 데이터베이스 시스템(Database System), 동시성 제어 기법(concurrency control scheme), 다중 데이터베이스 시스템(Multidatabase Systems), 트랜잭션(Transaction)

• 제1저자 : 이상희
• 접수일 : 2005.10.10, 심사완료일 : 2005.10.26
* 청강문화산업대학 컴퓨터소프트웨어과 교수

Ⅰ. 서 론

다중 데이터베이스 시스템(Multidatabase System: MDBS)에서는 하나이상의 로컬 데이터베이스 시스템(Local Database System: LDBS)에 산재된 자료에 접근하기 위해서 전역트랜잭션(Global Transaction)이 제기된다. 전역트랜잭션이 MDBS에 제기될 때, 그것은 결국 하나의 LDBS에 있는 자료에 접근하는 전역 부트랜잭션(Global Subtransaction)들로 나뉘어 진다. 한 로컬에 있는 자료만 접근할 수 있다는 면에서 본다면, 전역 부트랜잭션은 LDBS에 바로 제기된 로컬트랜잭션(Local Transaction)과 같은 것처럼 보인다. 그러나, 트랜잭션처리 관점에서 보면 전역 부트랜잭션은 로컬트랜잭션과 크게 다르다. 로컬트랜잭션은 한 LDBS에서만 직렬화가 가능하게 실행되면 되지만, 전역 부트랜잭션은 모든 LDBS에서 다른 전역트랜잭션의 부트랜잭션과 비교해서 같은 순서로 실행될 수 있어야 하기 때문이다.

로컬트랜잭션과 전역트랜잭션이 함께 실행될 때, LDBS의 자치성 때문에 전역트랜잭션의 직렬화가능성을 보장하는 일은 어렵다. 로컬 트랜잭션 자치성(Local Autonomy)이 완전히 보장될 때, 전역 트랜잭션처리기는 각 LDBS에서 실행된 전역 부트랜잭션에 대한 지역 직렬화 가능 순서에 관련된 어떤 정보도 얻을 수 없기 때문이다.

많은 동시성 제어 기법[1,2,4]들이 제안되었다. 전역 비직렬화 가능성(Global Nonserializability)을 피하기 위해서 전역트랜잭션의 실행을 지연시키는지 여부에 따라, 이 기법들은 두 부류로 나뉘어 진다: 비관적인 기법[Brei88]과 낙관적인 기법[2, 1]. 비관적인 동시성 제어 기법에서는 같은 LDBS에서 동시에 전역 부트랜잭션들이 실행될 때, 늘 충돌이 일어날 수 있다는 비관적인 가정으로 전역 비직렬화 가능성이 근원적으로 발생하지 못하도록 전역트랜잭션을 지연시킨다. 그러나, MDBS가 각 LDBS에서 실행된 전역 부트랜잭션들에 대한 정확한 충돌 관계를 알지 못하고 무조건 조금이라도 충돌 가능성이 있으면 지연시키기 때문에, 비관적인 트랜잭션 처리는 보수적일 수 있다. 이러한 보수적인 트랜잭션처리는 전역 트랜잭션들의 동시성을 크게 떨어뜨리게 된다.

동시성 저하를 막기 위해서, 전역 부트랜잭션들이 동시에 실행되더라도 그들간에 충돌이 거의 없을 것이란 낙관적인 생각을 가지고, 전역 부트랜잭션들이 각 LDBS에 지연없이 자신의 연산들을 제기하도록 하는 낙관적인 트랜잭션처리 기법들이 제안되었다. 이 기법에서는 전역트랜잭션의 모든 연산이 LDBS에서 성공적으로 실행된 후에, 각 LDBS에서 그들의 직렬화 가능 순서를 획득하여 모든 LDBS에서 동일한 실행순서를 갖는지를 검증한다. 만일, 어떤 전역트랜잭션이 모든 LDBS에서 동일한 실행순서를 가질 수 있다면 그 트랜잭션은 정상적으로 완료될 수 있지만, 그렇지 못하면 재실행된다. 그러나, 낙관적인 방법은 비관적인 방법과는 달리 각 LDBS에서 이미 실행을 완료한 후에 전역 직렬화 가능성을 검증하여 문제가 발생하면 트랜잭션을 재실행시키기 때문에 시간과 공간적인 측면에서 재실행 부담이 클 수 있다. 따라서, 낙관적인 방법의 성능을 증진시키기 위해서는 전역 직렬화 가능성의 위반으로 인해 재실행되는 전역트랜잭션의 수를 줄이는 것이 관건이라 할 수 있다.

Ⅱ 관련연구

낙관적인 동시성 제어 기법에서 전역트랜잭션의 직렬화 가능성을 검증하기 위해서 지역 직렬화 가능 순서가 필요하다. 널리 이용되고 있는 지역 직렬화 가능 순서로는 LDBS에 의도적으로 만들어 둔 티켓자료로부터 읽어들인 값(Ticket Data Value)[2]과 각 LDBS의 트랜잭션처리기의 특성으로부터 추측할 수 있는 직렬화 순서 결정 시점(Serialization Point) 등이 있다. OTM(Optimistic Ticket Method)은 각 전역트랜잭션의 지역 직렬화 가능 순서가 자신이 읽은 티켓값과 일치하도록 만든다. 지역 직렬화 가능 순서를 얻기 위해서 OTM은 LDBS에 어떠한 제약도 가하지 않기 때문에 전역 트랜잭션을 처리하기 위한 실용적인 트랜잭션처리 기법으로 인정받고 있다. 그러나, 트랜잭션들간에 지역 직렬화 가능 순서가 서로 엉키게 되면 지역에서 성공적으로 실행을 마친 트랜잭션이 전역 직렬화 가능성의 위반으로 재실행되어야 하는 단점이 있을 수 있다.

이러한 문제점을 완화하기 위해서, 전역트랜잭션이 각 LDBS에서의 지역 직렬화 가능 순서가 결정될 때마다 전역 직렬화 가능성의 위반 여부를 검증하여 문제가 발생하면 조

기에 재실행시킬 수 있는 DAGSO(Dynamic Adjustment of Serialization Order)[1] 기법이 제안되었다. 이 기법은 전역 직렬화 가능성을 검증하기 위해서 트랜잭션들의 지역 직렬화 가능 순서가 결정될 때마다 같은 LDBS에서 실행 중이면서 아직 지역 직렬화 가능 순서가 결정되지 않은 모든 전역트랜잭션에 대하여 에지를 첨가하여 고리의 발생 여부를 검사한다. 만일 고리가 발생하면, 그래프를 재조정하여 고리를 없애려는 노력을 한다. 도저히 고리를 제거할 수 없는 경우에는 고리에 포함된 일부 전역트랜잭션을 재실행하여 문제를 해결한다. 이 방법은 OTM에 비하여 전역 직렬화 가능성의 위반 여부를 미리 파악하여 문제가 될 가능성이 있는 전역트랜잭션을 재실행시킴으로써 재실행 비용을 줄일 수 있다. 뿐만 아니라, 때로는 동적으로 전역트랜잭션의 직렬화 가능 순서를 조정할 수 있기 때문에 재실행되는 트랜잭션의 수도 줄일 수 있는 장점이 있다.

그러나, 이들 두 방법의 경우에는 각 LDBS의 동시성 제어 기법이 서로 다른 경우에 불필요하게 전역트랜잭션을 재실행해야 하는 경우가 발생할 수 있다. LDBS의 동시성 제어 기법의 차이점이 전역트랜잭션 처리에 미치는 영향을 파악하기 위하여, LDBS에서 널리 사용되고 있는 동시성 제어 기법을 살펴보자. 데이타베이스 시스템의 동시성 제어 기법으로는 엄격한 두 단계 로킹(Strict Two-Phase Locking: S2PL), 시간 순서화(Timestamp Ordering: TO) 및 낙관적인 기법(Optimistic Scheme: OPT)이 있다. S2PL의 경우 한번 획득한 쓰기록(Write Lock)을 그 트랜잭션이 완료할 때까지 유지하도록 하기 때문에, 각 트랜잭션의 직렬화 가능 순서는 해당 트랜잭션이 필요로 하는 모든 록을 획득하는 순서로 결정된다. 즉, 직렬화 가능 순서가 트랜잭션 처리의 마지막 부분에서 결정된다는 것을 알 수 있다. 직렬화 가능 순서가 늦게 결정되는 것은 OPT에서도 마찬가지다. 실행을 마친 트랜잭션들에 대하여 타 트랜잭션들과의 충돌여부를 검증(Validation)하는 단계가 있는데 이 검증을 마친 순서로 직렬화 가능 순서가 결정될 수 있기 때문이다. 그러나, TO기법의 경우 트랜잭션들이 시간 순서로 실행되기 때문에 시간을 부여받는 순서가 해당 트랜잭션의 직렬화 가능 순서가 되고, 이는 트랜잭션이 시작되는 단계에서 결정된다는 점이 앞의 두 기법과 크게 다르다.

앞에서 살펴본 것처럼 동시성 제어 기법이 다를 경우 트랜잭션의 직렬화 가능 순서가 결정되는 시점이 다르다는 점을 알았다. 그러나, 각 지역에서 결정된 지역 직렬화 가능 순서들을 잘 살펴보면, TO방법에서 시간은 한번 결정되면 더 이상 조정이 불가능한 것이지만 S2PL의 경우 의도적으로 그 순서를 조정할 수 있다는 점을 관찰할 수 있다. 따라서, 서로 다른 동시성 제어 기법을 가진 LDBS를 통합할 때, 각 LDBS의 지역 직렬화 가능 순서가 결정되는 시점을 잘 조정한다면 지역실행을 완료하고 전역 직렬화 가능성을 위반하여 재실행되어야 하는 전역트랜잭션의 수를 줄일 수 있음을 짐작할 수 있다. 불행히도, 기존의 OTM과 DAGSO 기법은 전역 질렬화 가능성을 검증할 때 각 지역에서 결정된 지역 직렬화 가능 순서에 대한 의미를 충분히 분석하여 전역 직렬화 가능순서 결정에 활용하지 못하였기 때문에 불필요하게 전역트랜잭션을 재실행시킬 수 있다. 예제 1에서 이것을 보여 준다.

[예제 1] (지역 직렬화 가능 순서의 조정 부족으로 인한 불필요한 전역트랜잭션의 재실행): 두개의 서로 다른 지역 데이타베이스 시스템, LDBS1과 LDBS2를 생각해 보자. 동시성 제어를 위하여 LDBS1은 TO 기법을 사용하고, LDBS2는 S2PL 기법을 사용한다고 가정한다.

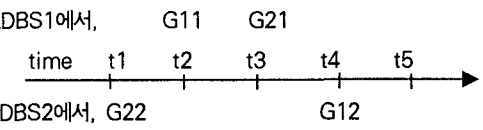


그림 1 불필요한 전역트랜잭션 재실행 예
Fig 1 example of unnecessary REDO processing for global transaction

전역트랜잭션 G1이 LDBS1과 LDBS2에 각각 부트랜잭션 G11과 G12를, G2가 LDBS1과 LDBS2에 부트랜잭션 G21과 G22를 다음과 같은 순서로 제기한다고 하자. S2PL의 경우 엄격하게(Strictly) 트랜잭션을 처리하기 때문에 로컬 트랜잭션으로 인한 간접 충돌에 상관없이 필요한 록을 모두 얻는 순서로 지역 직렬화 가능 순서가 결정될 수 있기 때문에, 문제를 단순화하기 위해서 로컬 트랜잭션은 도입하지 않는다.

LDBS1은 TO 기법을 동시성 제어 기법으로 채택하고 있기 때문에, LDBS1에 제기된 두 전역 부트랜잭션 G11과 G21의 지역 직렬화 순서는 그들이 부여받은 시간 순서와 일치할 것이다. 시간 t2에서 G11이 먼저 제기되었고 t3에서 G21이 도착하였으므로 G11의 시간이 G21의 시간 보다 작을 것이다. 따라서, G11이 G21보다 직렬화 가능 순서가 앞선다. 반면에, LDBS2는 S2PL을 채택하고 있기 때문에, LDBS2에 제기된 두 부트랜잭션 G22와 G12의 지역 직렬

화 가능 순서는 그들이 필요로 하는 모든 록을 다 획득한 시점으로 결정된다. 시간 t1에서 G22가 필요로 하는 록을 모두 획득하였다고 하고, 시간 t4에서 G12가 필요로 하는 록을 모두 획득하였다고 가정한다면 이들 둘간의 직렬화 가능 순서는 G22가 G12보다 앞선다. 이상에서 본 것과 같이 전역 트랜잭션 G1과 G2는 두 LDBS에서 서로 다른 직렬화 가능 순서를 가진다. 이러한 순서로 실행이 될 경우에 OTM과 DAGSO 두 방법 모두에서 시간 t3에서 전역 트랜잭션 G2는 정상적으로 완료할 수 있지만, 시간 t3에서 G1은 그대로 완료될 경우 전역 직렬화 가능성이 보장될 수 없기 때문에 재실행시킨다. 그러나, LDBS2에서 G22와 G12가 직접 혹은 간접적으로 충돌이 없다고 한다면, 시간 t3에서 G21이 성공적으로 실행을 마치더라도 바로 완료하지 않고 기다리게 할 수 있다. 이때, 무조건 기다리는 것이 아니라 TO를 채택하고 있는 LDBS1에서 G21보다 시간 순서상 앞서면서 아직 완료되지 않은 G11과 같은 트랜잭션이 있을 경우에만 부트랜잭션 G11을 제기한 전역트랜잭션 G1이 완료할 때까지 G2를 종료하지 않고 기다리게 할 수 있다. 이렇게 된다면, 시간 t4에서 G1이 먼저 완료하게 되고 그 다음에 시간 t5에서 G2가 완료할 수 있기 때문에 불필요하게 하나의 전역트랜잭션이 재실행될 필요가 없다.

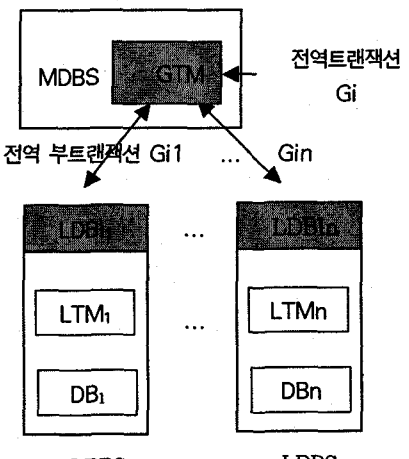
이와 같이 MDBS에 참여하고 있는 각 LDBS의 동시성 제어 기법의 직렬화 가능 순서 결정 시점이 다를 경우, 각 LDBS에서 결정된 지역 직렬화 가능 순서를 적절히 조정한다면 불필요한 전역트랜잭션의 재실행을 방지할 수 있을 것이다.

본 논문에서는 동시성 제어 기법으로 TO방법과 S2PL방법을 사용하고 있는 LDBS들이 통합된 MDBS에서 전역트랜잭션의 재실행을 줄일 수 있는 새로운 동시성 제어 기법을 제안한다.

III 트랜잭션 처리 모델

본 연구에서 채택된 트랜잭션처리 모형을 (그림 3.1)에 제시한다. 여러 개의 LDBS들이 하나의 MDBS로 통합될 때, 각 LDBS가 채택하여 사용하고 있는 동시성 제어 기법이 어떤 것인지를 알 수 있다고 가정을 한다. 또한 2PL을 사용하고 있는 LDBS의 경우 어떤 트랜잭션이 쓰기록을 획득하면 반드시 그 트랜잭션이 완료할 때까지 해당 록을 해제하지 않는 성질을 갖는다고 한다. 실세계에서 사용되고 있는 대부분의 DBMS가 이러한 방식으로 록을 관리하기 때문에, 무리한 가정은 아니다. TO방법을 사용하는 LDBS의 경우에는 각 전역트랜잭션이 첫 연산을 제기한 순서대로 시간을 부여한다고 가정한다. 이 가정 또한 해당 DBMS들이 선착순으로 작업을 처리한다고 볼 때 무리한 가정이 아니다. 만일, 선착순으로 시간을 부여하지 않는 다고 한다면 전역 트랜잭션이 첫 연산을 제기한 경우에 한하여 그 연산을 마칠 때까지 기다렸다가 다른 연산을 제기하도록 함으로써 각 트랜잭션들간의 상대적인 시간 값을 추측할 수 있다. 그 이외에 LDBS에는 어떠한 지역 자치성도 훼손하는 가정을 하지 않는다.

LDBS가 MDBS로 통합되기 전에 기 개발되어 운영되고 있던 응용들은 아무런 수정을 요구받지 않고 직접 LDBS에 트랜잭션을 제기한다. 하지만 전역트랜잭션은 먼저 MDBS에 제기되고, 한 LDBS에 있는 자료를 접근하는 연산으로 구성된 전역 부트랜잭션으로 나뉘어져 각 LDBS상부에서 실행중인 LDBI로 보내지게 된다. 이 전역 부트랜잭션은 기존 응용에서 직접 제기된 로컬 트랜잭션과 같이 LDBS에 제기되어 실행된다. 트랜잭션은 시작(Begin)후 완료(Commit) 또는 재실행(Restart)으로 종료한다. 시작과 종료사이에 자료를 접근하는 읽기(Read)와 쓰기(Write)연산을 임의의 수만큼 제기할 수 있다.



LDBI : 지역 데이타베이스 연계부
(Local Database Interface)
GTM,/LTM : 전역/지역 트랜잭션 관리기

그림 2 트랜잭션 처리 모형
Fig 2 Model of transaction processing

Ⅳ. 지연완료 동시성 제어 기법

지금부터 본 논문에서 제안하는 동시성 제어 기법을 자세히 설명한다.

4.1 지연완료의 적용 예

지연완료(Deferred Commitment: DC) 기법의 효용가치를 보이기 위해서 TO기법을 동시성 제어 기법으로 채택하고 있는 LDBS1과 S2PL을 사용하는 LDBS2로 구성된 MDBS를 가정한다. S2PL에서 모든 트랜잭션이 쓰기록을 완료할 때까지 보유하기 때문에 서로 다른 두 전역 부트랜잭션이 로컬 트랜잭션과의 간접 충돌로 인하여 직렬화 가능 순서가 바뀌는 경우는 없기 때문에 로컬 트랜잭션은 도입하지 않는다.

[예제 2] (지연완료의 적용 예): 두개의 전역트랜잭션 G1과 G2는 LDBS1과 LDBS2에 각각 다음과 같은 두 전역 부트랜잭션을 제기한다. G11은 G1이 LDBS1에 있는 X를 읽기 위해서 제기하는 전역 부트랜잭션으로서 R11(X)연산으로 구성되고, G12는 LDBS2에 있는 Z를 읽기 위해 제기하는 전역 부 트랜잭션으로서 R12(Z)연산으로 구성된다. 전역트랜잭션 G2에 대해서도 부트랜잭션 G21과 G22를 각각 W21(X)연산과 W22(Y)연산으로 구성된 것으로 정의할 수 있다. 각 연산들은 다음과 같은 순서로 제기된다. 연산을 마친 G1이 완료하기 위해서 LDBS1과 LDBS2에 각각 완료연산 C11과 C12를 제기한다.

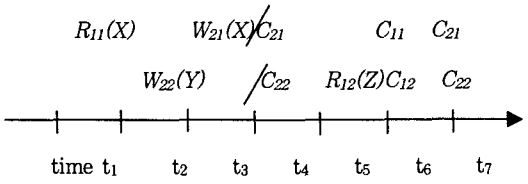


그림 3 지연완료 적용 예
Fig 3 Example of deferred commitment

전역트랜잭션 G2는 시간 t3에서 자신의 모든 연산을 성공적으로 실행하였다. 따라서, 시간 t4에서 즉시 완료연산을 제기할 수 있지만 TO방법을 사용하는 LDBS1에서 G21보다 시간 값이 앞서는 G11이 있음을 고려하여 G1이 완료할 때까지 완료연산을 제기하지 않고 기다리도록 한다. 그 후 G1은 시간 t5에서 R12(Z)의 실행을 마치고, 시간 t6에서 C11과 C12를 제기하여 정상적으로 완료한다. 그 후 시간 t7에서 G2도 C21과 C22를 제기하여 성공적으로 완료할 수 있다. 즉, G1의 재실행이 없이 G1과 G2의 순서로 두 트랜잭션이 성공적으로 실행될 수 있었다.

4.2 교착상태

낙관적인 방법에서는 두개 이상의 전역트랜잭션들 간에 교착상태가 발생할 수 있다. OTM과 DAGSO방법과는 달리 전역트랜잭션의 완료를 지연시키기 때문에 DC방법에서는 교착상태가 더 자주 발생할 수 있다. 앞의 (그림 4.1)에서 전역트랜잭션 G1이 R12(Z)대신에 R12(Y) 연산을 제기한다고 한다면 전역 트랜잭션 G1과 G2사이에는 교착상태가 발생한다. 그러나, 낙관적인 방법을 사용할 때 전역트랜잭션들 간에는 충돌이 거의 발생하지 않는 다는 기본적인 가정을 하고 있기 때문에 이러한 추가적인 교착상태는 거의 발생하지 않을 것이다.

전역 교착상태를 해결하기 위해서 일정기간 기다린 후에 응답이 없으면 재실행시키는 timeout방법이 구현의 용이성 때문에 많이 쓰이고 있다. 따라서, 본 논문에서도 교착상태를 해결하기 위해서 timeout방법을 이용한다.

4.3 시간 동기화

TO방법을 사용하는 LDBS가 두개 이상있을 경우에 한 전역트랜잭션이 해당 LDBS들에서 성공적으로 실행을 마치려고 하면 각 LDBS들에서 얻은 시간 값이 다른 트랜잭션들과 비교해서 상대적으로 같은 순서를 가져야 한다. 다시 말해서, LDBS1에서 부트랜잭션 G11이 G21보다 작은 시간 값을 얻지 못한다면 두 전역트랜잭션 G1과 G2사이에 전역 직렬화 가능 순서를 찾을 수 없기 때문에 둘 중 하나가 반드시 재실행되어야 할 것이다. 이러한 문제를 미연에 방지하기 위해서 전역트랜잭션이 어느 한 LDBS에서 시간을 부여받으면 그 MDBS에 TO방법을 쓰고 있는 다른 모든 LDBS에서 타 전역트랜잭션과 비교할 때, 상대적으로 같은 순서의 시간을 갖도록 동기화하는 일이 필요하다.

이를 위하여 TO를 사용하고 있는 모든 LDBS에 dummy 자료 항목을 두어 한 LDBS에서 시간이 결정되는 시점에서

TO를 사용하는 다른 모든 LDBS에서 이 dummy항목을 읽는 연산을 추가하여 시간을 동기화한다. 이렇게 시간을 동기화함으로써 향후에 발생할지 모르는 전역트랜잭션들간의 연산들이 시간 순서대로 제기되지 않으면 LDBS내에서 해당 연산들의 실행이 불가능해지기 때문에 지역 연산의 실행을 성공적으로 마친 전역트랜잭션이 전역 직렬화 가능 순서의 위반으로 인해 재실행될 가능성을 크게 줄여 준다. 다시 말해서, 직렬화 가능성이 위배될 수 있는 것을 조기에 발견함으로써 재실행비용을 줄일 수 있다는 것이다.

이 dummy항목에 대한 읽기연산은 추가적인 부담이 될 수 있지만, OTM과 DAGSO방법의 경우 두개 이상의 TO 방법을 사용하는 LDBS가 존재하는 경우 이들간에 지역 직렬화 가능 순서가 엉켜서 빈번하게 발생할 수 있는 트랜잭션 재실행부담에 비하면 비교가 안될 정도로 작다.

4.4 지연 완료 기법

DC기법의 중요한 트랜잭션처리 알고리즘을 기술한다.

```
/* GTM에서 */
/* 동시성 제어 기법의 종류(To, S2PL) */
string CCS_Type[1..NoofLDBS]
/* 트랜잭션의 시간 (초기값: -1) */
long TS_Trans[1..MaxTrans]
/* 완료 대기 큐 (Gi, Gj)로 구성됨, Gi는 대기하는
   트랜잭션, Gj는 대기의 대상이 되는 트랜잭션 */
waiting_queue = null
Relative_Time = 1 /* 논리적인 시간 */

/* 단계1: 트랜잭션 Gi로부터 시작 연산을 받을 때*/
  TS_Trans[i] = 0
  return 0 /* success */

/* 단계2: 트랜잭션 Gi로부터 읽기/쓰기 연산을 받을 때*/
  해당 읽기/쓰기 연산을 실행할 LDBSk를 찾음*/
  if CCS_Type[k] = 'S2PL' then
    해당 연산을 LDBSk로 보내고, timeout
    시간을 설정하고, 대기
    if success then
      return 0 /* success */
    else
      TS_Trans[i] = -1
      Gi를 재실행
      return -1 /* fail */
    endif
  else
    if TS_Trans[i] > 0 then
    /* 이미 TO방법을 사용하는 LDBS에서 시간을
       부여받은 트랜잭션임 */
      해당 연산을 LDBSk로 연산을 보내고, timeout
      시간을 설정하고, 대기
      if success then
        return 0 /* success */
      else
        TS_Trans[i] = -1
        Gi를 재실행
        return -1 /* fail */
      endif
    else

      /*해당 연산을 LDBSk로 보내고, TO를
      갖는 다른 모든 LDBS에 dummy를 읽는 연산을
      보내고, timeout시간을 설정하고, 대기 */
      if all success then
        return 0 /* success */
      else
        TS_Trans[i] = -1
        success를 받은 LDBS에 Gi를 재실행토록 함
        return -1 /* fail */
      endif
    endif
  endif
/* 단계3: 트랜잭션 Gi로부터 완료 연산을 받을 때*/
  No_Count = 0
  For k = 1 to MaxTrans
    if TS_Trans[k] > 0 and
      TS_Trans[k] < TS_Trans[i] then
      /* Gi가 완료하기 전에 Gk의 완료가
         선행되어야 함 */
      waiting_queue에 (Gi, Gk)를 추가
      No_Count ++
    endif
  Next
  if No_Count > then
    timeout시간을 설정하고, 시간이 앞선 트랜잭션들이
    종료할 때까지 대기
  else
    Gi의 부트랜잭션이 실행된 모든 LDBSk에 Cik를
    보내고, timeout시간을 설정한 후에 2단계 완료를
    시작함
    if 2단계 완료를 성공 then
      TS_Trans[i] = -1
      Gi의 완료를 기다리고 있는 모든 트랜잭션들을
      깨워 줌
      return 0 /* success */
    else
      TS_Trans[i] = -1
      Gi를 처음부터 재실행
      return -1 /* fail */
    endif
  endif
/* 단계4: timeout시간의 초과 시 */
  TS_Trans[i] = -1
  해당트랜잭션을 재실행
  return -1

/* LDBIk에서 */
  전역트랜잭션 Gi로부터 받은 전역 부트랜잭션 Gik의
  연산들을 LDBSk로 보내서 실행한 후에 결과를
  되돌림

  두단계 완료의 참여자 역할을 수행함

/* LTMk에서 */
  로컬 트랜잭션 및 전역 부트랜잭션들을 실행함
```

4.5 시뮬레이션을 통한 성능 평가 및 분석

본 논문에서 제시한 DC 기법의 성능을 시뮬레이션을 통해 측정하여 보았다. (그림 4)는 LDBS가 모두 S2PL을 사용하는 경우로 기존 기법에 비해 DC 기법이 우월한 성능을 나타내지 못하고 있었다.

그러나 (그림 5)의 결과와 같이 일부 LDBS가 TO기법을 사용하는 경우 우월한 성능을 보였다. 본 시뮬레이션은 5개의 LDBS중 2개의 LDBS가 TO기법을 사용하는 것을 가정하여 실험하였다.

만일 TO 기법을 사용하는 LDBS가 더 많을 경우 좀더 많은 성능 차이를 보일 것이다.

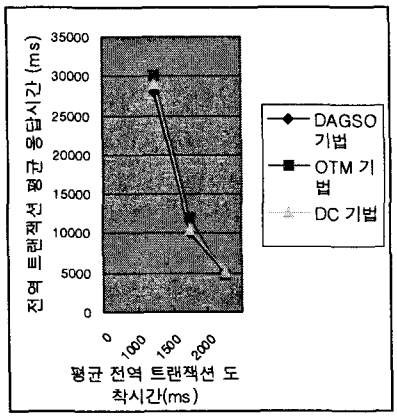


그림 4 LDBS가 모두 S2PL을 사용하는 경우
Fig 4 Case of all LDBS using S2PL

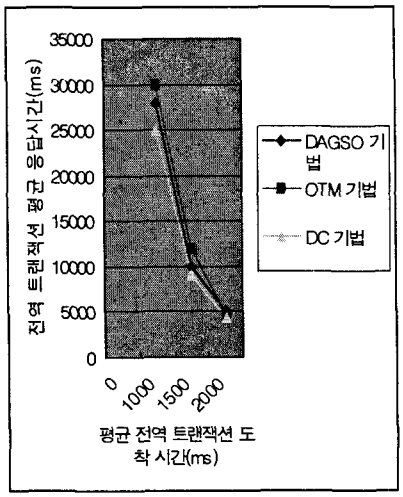


그림 5 LDBS가 TO기법과 S2PL이 함께 통합된 경우
Fig 5 Case of LDBS using S2PL and TO

결론적으로 시뮬레이션 결과는 DC기법은 LDBS가 TO 기법을 채택하고 있는 경우가 있을 경우 타 기법에 비해 우월한 성능을 보이는 것으로 나타났다.

V. 결 론

본 논문에서는 MDBS에서의 전역트랜잭션을 처리하기 위한 동시성 제어 기법 DC를 제안하였다. 기존의 OTM과 DAGSO방법이 전역트랜잭션을 처리할 때, MDBS에 참가하고 있는 LDBS들의 동시성 제어 기법들의 지역 직렬화 가능 순서 결정에 관련된 정보를 충분히 활용하지 못함으로써 불필요하게 전역트랜잭션을 재실행시키는 문제점을 트랜잭션의 완료를 지연하는 방법으로 해결하였다.

DC 기법의 성능은 참가하고 있는 LDBS가 모두 S2PL을 사용하는 경우와 TO기법이 S2PL과 함께 통합된 경우로 나눠 볼 수 있다. 전자의 경우에는 모든 LDBS에서 전역부 트랜잭션의 지역 직렬화 가능 순서가 자신이 필요로 하는 록을 모두 획득하는 시점에서 결정이 되기 때문에 DC 고유의 지역 직렬화 가능 순서에 대한 추가조정 기능이 효과를 발휘하지 못하기 때문에 DAGSO와 같은 성능을 나타낼 수 있을 것으로 추정되었다. 반면에 후자의 경우에는 각 전역 트랜잭션에 대하여 TO기법을 사용하는 LDBS에서 미리 결정된 시간 값을 존중하여 트랜잭션을 처리할 수 있기 때문에 DAGSO기법에 비하여 더 좋은 성능을 나타낼 수 있을 것으로 추정 되었다. 만일, TO기법을 채택하고 있는 LDBS들이 둘 이상 존재하는 경우에는 한 LDBS에서 결정된 시간을 다른 LDBS에서도 그대로 유지될 수 있도록 처리함으로써 그 성능의 차이가 훨씬 더 커질 수 있을 것으로 예상 되었다.

이러한 추정은 시뮬레이션을 통해 DC기법의 우월적 성능으로 입증 되었다.

참고문헌

[1] S. Hwang, J. Huang, and J. Srivastava, "Concurrency Control in Federated Databases: A Dynamic Approach," Proceedings of The 2nd International Conference on Information and Knowledge Management, Washington, U.S.A., Nov. 1999, pp. 694-703.

[2] D. Georgakopoulos and M. Rusinkiewicz, "On Serializability of Multidatabase Transactions Through Forced Local Conflicts," Proceedings of The 7th International Conference on Data Engineering, Kobe, Japan, April 1991, pp. 314-323.

[3] P. Bernstein, V. Hadzilacos, and N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987.

[4] Y. Breitbart, P. Olson, and G. Thompson, "Database Integration in a Distributed Database System," Proceedings of The 2nd International Conference on Data Engineering, California, U.S.A., Feb. 1986, pp. 301-310.

[5] Y. Breitbart, H. Garcia-Molina, and A. Silberschatz, "Overview of Multidatabase Transaction Management." VLDB Journal, 1992, pp. 181-239.

[6] Paul W. P. J. Grefen, Jochem Vonk, Peter M. G. Apers, "Global transaction support for workflow management systems: from formal specification to practical implementation." VLDB Journal, 2001, pp. 316-333

[7] 장연세, "이형분산 컴포넌트 플랫폼간 상호 운영성 보장에 대한 연구." 한국 컴퓨터 정보학회, 7권. 4호. 2002

[8] 김홍수, "참조 스키마 생성을 위한 개념적 스키마 분석." 한국 컴퓨터 정보학회, 7권. 4호. 2002

저 자 소 개

이 상 희

1996년~현재 청강문화산업대학 교수

〈관심분야〉 데이터베이스, 멀티데이타베이스, 모바일 시스템