# 멀티데이터베이스 시스템에서 메시징 순서를 적용한 전역 동시성 제어 기법

문애경° 남궁한
한국전자통신연구원 분산협업기술 연구팀
{akmoon°, nghan }@etri.re.kr

# Global Concurrency Control using Messaging Ordering in Multidatabase System

Aekyung Moon° Han Namgoong
Distributed Collaboration Technology Research Team / ETRI

**요 약**

멀티데이터베이스 시스템이란 기존에 독자적으로 개발되어 사용되고 있는 서로 다른 지역 데이터베이스 시스템들을 논리적으로 통합하기 위해 제안된 시스템이다. 본 논문에서는 논리적인 통합 인터페이스로 메시징 소프트웨어를 이용하고 메시징 순서 기능을 적용한 전역 동시성 제어 기법을 제안한다. 전역 직렬성을 보장하기 위하여 전역 트랜잭션의 서브 트랜잭션이 실행되는 모든 LDBS에서 같은 순서로 직렬화되어야 한다. 메시징 순서 기능은 하나의 노드가 여러 개 메시지를 송신하는 경우, 모든 노드는 송신한 순서에 따라 메시지를 수신함을 보장하는 것으로 이를 이용하여 전역 트랜잭션의 서브 트랜잭션들을 실행 노드에 같은 순서로 전송하고 실행 LDBS는 해당 순서대로 직렬화한다면 전역 직렬성을 보장할 수 있다. 그 결과 제안된 기법은 LDBS의 실행 정보 없이 전역 트랜잭션의 상대적인 실행 순서를 결정할 수 있기 때문에 지역 자치성을 보장한다.

## 1. 서 론

최근 웹 문서 검색, 컨텐츠 서비스 등을 지원하기 위한 다양한 데이터 자원이 발생하면서, 정보 통합에 대한 많은 기술 개발이 진행되고 있다. 멀티데이터베이스 시스템(Multidatabase System: MDBS)은 여러 곳에 산재해있는 유용한 정보를 통합, 일관된 인터페이스를 통한 접근을 제공하는 이질형 분산 데이터베이스 시스템이다[1]. 즉, 통합하고자 하는 정보 자원들의 데이터를 실제로 저장하는 것이 아니라 각 정보 자원을 갖는 지역 데이터베이스 시스템들을 네트워크, 또는 인터페이스로 연결하고 이에 대해 질의를 분해, 전달 후 질의 결과를 통합하는 가상 통합 시스템(virtual integrated system)이다. 이러한 MDBS는 하나이상의 지역 데이터베이스시스템(Local Database System: LDBS)을 액세스하는 전역 응용을 처리할 수 있다. 하지만, 일반적인 분산 데이터베이스 시스템 환경과 달리 통합되는 LDBS들의 서로 다른 질의 처리 능력을 제공할 뿐만 아니라, 지역 자치성 요구 사항 때문에 전역 트랜잭션 처리를 위한 상호 협조가 불가능하여 전역 직렬성을 만족하는 전역 동시성 제어 기법(Global Concurrency Control: GCC)의 설계가 어렵다.

최근 대두된 메시징 소프트웨어는 응용 프로그램간 비동기식 통신 방법을 제공하는 것으로 기업간 응용 통합 인터페이스로 중요성이 증가하고 있다[2,3]. 본 논문에서는 LDBS와 MDBS 사이의 논리적인 통합 인터페이스로 메시징 소프트웨어를 이용하고 메시징 순서 기능을 적용한 GCC를 제안한다. GCC는 전역 직렬성을 보장하기 위하여 전역 트랜잭션의 서브 트랜잭션이 실행되는 모든 LDBS에서 같은 순서로 직렬화되어야 한다. 메시징 순서 기능은 하나의 노드가 여러 개 메시지를 송신하는 경우, 모든 노드는 송신 순서에 따라 메시지 수신을 보장하는 것으로 이를 이용하여 전역 트랜잭션의 서브 트랜잭션들을 실행 노드에 같은 순서로 전송하고 실행 LDBS는 해당 순서대로 직렬화한다면 전역 직렬성을 보장할 수 있다. 그 결과 본 논문에서 제안하는 전역 동시성 제어 기법은 LDBS의 실행 정보 없이 전역 트랜잭션의 상대적인 실행 순서를 결정할 수 있기 때문에 지역 자치성을 보장한다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구에 대해 살펴보고, 3절에서는 본 논문에서 제안하는 전역 동시성 제어 기법을 설명한다. 4절에서 분석하고 5절에서 결론을 맺는다.

## 2. 관련 연구

### 2.1 GCC 알고리즘

지역 자치성을 보장하는 GCC에는 낙관적 티켓 기법(Optimistic Ticket Method: OTM)[4]과 chain-conflicting serializability[5]과 있다. 두 기법은 전역 트랜잭션 사이에 발생하는 간접 충돌을 해결하기 위하여 강제적인 직접 충돌을 유발한다. 특히, OTM은 모든 전역 트랜잭션들이 티켓이라는 공통된 데이터를 수정하게 함으로써 직접 충돌을 유도하는 기법으로, 직접 충돌로 인하여 비직렬 스케쥴을 발견하고 지역 동시성 제어기로 하여금 그 스케쥴을 수행하지 않게 함으로써 전역 직렬성을 보장한다. 그러나, OTM의 경우 동시에 실행되는 트랜잭션의 수가 많아지면 빈번한 티켓

충돌로 인해 낮은 동시성과 높은 철회율로 성능이 떨어질 수 있고, 전역 교착상태가 발생할 수 있다는 단점을 갖는다[6].

2.2 메시징 소프트웨어

메시징 소프트웨어는 응용간 비동기식 통신 방법을 제공하는 것으로 기업간 응용 통합, 인터넷 기반 비즈니스 솔루션 그리고 J2EE의 메시징 서버 솔루션 등으로 각광 받고 있다. 최근 상용화된 메시징 소프트웨어로는 IBM'의 MQSeries, Progess사의 SonicMQ, Fiorano의 FioranoMQ 그리고 썬 마이크로 시스템즈의 JMQ 등을 들 수 있다[3,7]. 메시징 순서 기능은 메시지 순서 기능은 그룹통신 메시징 시스템이 제공하는 기능이지만 최근에는 일반적인 메시징 시스템에도 채택되는 기능이다[8].

메시징 순서 기능은 노드가 여러 개 메시지를 방송하는 경우, 모든 노드에서 송신한 순서에 따라 메시지 수신을 보장하는 것이다. 예를 들면, 두 노드 $N_1$과 $N_2$가 다른 노드로부터 메시지 m과 m'를 전송 받는다면, $N_1$과 $N_2$의 메시지 실행 순서는 모두 m → m' 혹은 m' → m임을 보장한다. 이러한 메시징 순서 기능은 트랜잭션의 순서를 미리 정하는 것과 같은 역할을 하므로 전역 트랜잭션의 실행 LDBS에 대한 상대적인 직렬순서를 제어 할 수 있고 전역 교착상태 해결에 용이하다[9,10].

## 3. 메시징 순서 기반 전역 동시성 제어 기법

본 절에서는 메시징 순서 기반 전역 동시성 제어 기법(Global Concurrency Control based on Message ordering: GCC-M)을 제안한다. GCC-M은 지역 자치성을 보장하면서 LDBS의 실행순서를 보장하기 위하여 강제 직접 충돌을 사용한다. 전역 트랜잭션은 각 LDBS에 실행을 할당하기 전에 액세스하는 데이터 집합을 알고 있고 LDBS는 2단계 로킹을 지원한다고 가정한다.

3.1 자료 구조

MDBS는 전역 트랜잭션 관리자(Global Transaction Manager: GTM), 전역 트랜잭션 에이전트(Global Transaction Agent: GTA) 그리고 메시징 순서 컴포넌트(Message Ordering Component: MOC)로 구성된다. GTM은 전역 트랜잭션을 수락하고 그들의 실행을 담당하는 역할을 한다. 전역 트랜잭션은 동시에 실행되는 여러 개의 서브 트랜잭션으로 구성되는데, 각 서브 트랜잭션은 하나의 LDBS에 저장된 데이터베이스를 액세스하게 된다. GTM은 서브 트랜잭션 사이에 직접 충돌 여부를 검사하기 위하여 트랜잭션 테이블(Transaction Table: TTBL)과 메시지 큐 (Message Queue: MQ)의 두 가지 정보를 갖는다. 전역 트랜잭션 $T_k$의 $i$번째 서브 트랜잭션이 $T_k^i$인 경우, TTBL은 $<RS_k^i, WS_k^i>$를 기록한다. $RS_k^i$는 $T_k^i$의 판독 데이터 집합이고 $WS_k^i$는 $T_k^i$의 갱신 데이터 집합이다. MQ[$i$]는 $LDBS_i$의 완료되지 않은 서브 트랜잭션의 식별자를 기록한다. 만약 MQ[$i$]에 $T_{k-1}^i$가 $T_k^i$보다 먼저 있다면 직렬화 순서가 $T_{k-1}$ → $T_k$임을 의미한다. MOC는 기존에 존재하는 LDBS에 통합된 인터페이스를 제공함으로 써 메시지 순서 기능을 담당한다. 특히 MOC의 송신자는 각 서브 트랜잭션에 단일화된 타임스탬프를 부여하고 MOC의 수신자는 자신의 지역 큐에 수신한 서브 트랜잭션을 저장하고 타임스탬프 순서에 실행한다[11]. $MOC_i$는 $LDBS_i$에 존재하는 MOC의 수신자를 의미한다.

GTM은 $T_k$의 서브 트랜잭션들을 $T_k$에 할당된 GTA에 전달한다. 그러면 해당 GTA는 MOC를 통하여 각 서브 트랜잭션을 실행 LDBS에 전송한다. $MOC_i$는 모든 LDBS에서 동일한 순서로 실행되게 하기 위하여 메시징 순서에 따라 서브 트랜잭션을 실행한다. $LDBS_i$에는 $MOC_i$로부터 실행 순서대로 전송 받은 서브트랜잭션을 실행하기 위한 지역 트랜잭션 에이전트(Local Transaction Agent: LTA)가 존재한다. $LDBS_i$에서 $LTA_k^i$는 트랜잭션 $T_k$의 서브트랜잭션을 실행하기 위한 LTA를 의미한다. LDBS는 전역 직렬성을 보장하기 위하여 현재 진행중인 전역 트랜잭션의 실행 정보를 저장하기 위한 테이블(Execution Table: ETBL)을 갖는다. ETBL은 <트랜잭션 식별자, 충돌 연산, 실행 상태>의 정보를 갖는다. 서브 트랜잭션 $T_k^i$의 충돌 연산이 $O_k^i$인 경우, $O_k^i \in (RS_k^i \cup WS_k^i)$이고 $O_k^i$의 실행 상태는 $S(O_k^i)$로 표시한다. 단, $S(O_k^i)$의 초기 상태는 '0'이며, 실행 상태가 되면 '1'로 변경된다. 즉, 트랜잭션 $T_k^i$가 현재 실행 중이고 $O_k^i$가 현재 $LTA_k^i$에 의해 실행 요청되었다면 $S(O_k^i)$는 '1'이 된다. ETBL은 전역 트랜잭션의 실행을 제어하기 위하여 사용되는데 자세한 알고리즘은 3.2절에 설명된다.

3.2 GCC-M의 알고리즘

전역 트랜잭션들을 순차적으로 실행하더라도 지역 트랜잭션에 의해 발생하는 간접 충돌에 의해 전역 직렬성이 위반될 수 있다. 따라서 GCC-M은 간접 충돌을 해결하기 위하여 OTM과 같이 강제적인 직접 충돌 유발 방법을 사용한다. 그러나 OTM과 달리 전역 트랜잭션의 상대적인 직렬 순서를 사전에 알고 있기 때문에 불필요한 강제적인 직접 충돌을 제거할 수 있다. GCC-M의 구체적인 알고리즘은 다음과 같다.

**단계 1:** GTM은 전역 트랜잭션 $T_k$를 액세스하는 LDBS에 따라 서브 트랜잭션으로 분할한다. 서브 트랜잭션 $T_k^i$는 $LDBS_i$를 액세스 한다. 각 $T_k^i$에 대해 GTM은 $<RS_k^i, WS_k^i>$를 TTBL[$k$][$i$]에 등록하고, $T_k^i$는 MQ[$i$]에 등록한다.

**단계 2:** GTM은 MQ[$i$]에 저장된 바로 이전 전역 트랜잭션과의 관계를 검사한다. MQ[$i$]에 기록된 $T_k^i$의 이전 전역 트랜잭션을 $T_{k-1}^i$라고 가정하자. 그러면 전역 트랜잭션의 직렬화 순서는 $T_{k-1}$ → $T_k$가 되어야 한다. $T_k^i$의 갱신 집합 $WS_k^i$의 데이터 중 $WS_{k-1}^i$ 혹은 $RS_{k-1}^i$가 속하는 데이터가 있으면 $T_{k-1}^i$와 $T_k^i$간에는 직접 충돌이 존재하고 충돌 정보는 $(O_{k-1}^i, O_k^i)$가 된다. 그러나 $T_{k-1}^i$와 $T_k^i$사이에 직접 충돌이 존재하지 않으면 다음과 같이 충돌을 유발하기 위한 추가적인 연산이 필요하다.

(1) $WS_{k-1}^i$ 가 Ø가 아닌 경우, $WS_{k-1}^i$에 속하는 $x$를 선택하고 $T_k^i$의 마지막 위치에 R($x$)를 추가한다.
(2) $WS_{k-1}^i$ 가 Ø인 경우, $RS_{k-1}^i$에 속하는 $x$를 선택하고 $T_k^i$의 마지막 위치에 W($x$)를 추가한다. 이 때 x는 원본과 동일한 값을 갖는다.

**단계 3:** $T_k$의 모든 서브 트랜잭션에 대하여 충돌 정보가 결정되면, GTM은 GTA에게 그들의 충돌 정보와 함께 $T_k$의 서브 트랜잭션을 전달한다. 그러면, GTA는 각 실행 LDBS의 MOC의 수신자에게 각각의 서브 트랜잭션을 전송한다.

**단계 4:** 서브 트랜잭션 $T_k^i$와 관련 충돌 정보를 전송받으면, $MOC_i$는 $T_k^i$의 충돌 연산($O_k^i$)와 $S(O_k^i)$를 ETBL에 등록한다. $S(O_k^i)$의 초기 상태는 '0'이다. $MOC_i$는 $LTA_{k-1}^i$에게 $O_{k-1}^i$의

실행 정보를 요청한다. $O_{k-1}^i$가 실행되면 ETBL에 저장된 $S(O_{k-1}^i)$의 상태는 '1'이다. $MOC_i$는 $LTA_k^i$에게 서브 트랜잭션 $T_k^i$와 $O_k^i$를 전송한다. 그러면 $LTA_k^i$는 $O_k^i$를만날 때까지 $T_k^i$의 연산을 순서대로 실행 요청한다. 만약, 요청하고자 하는 연산이 $O_k^i$인 경우, $LTA_k^i$는 먼저 $MOC_i$를 통해 $O_k^i$와 충돌되는 연산이 실행되었는지 확인한다. $MOC_i$는 $O_k^i$의 충돌 연산 $O_{k-1}^i$의 $S(O_{k-1}^i)$을 확인한다. $O_{k-1}^i$가 속한 서브 트랜잭션이 $O_{k-1}^i$의 실행을 요청하지 않았다면 즉, $S(O_{k-1}^i)$가 '0'이면 $O_k^i$는 해당 $T_{k-1}^i$의 $O_{k-1}^i$가 실행될 때까지 대기한다. 이 때 $MOC_i$와 $LDBS_i$사이에 지역 교착상태를 유발할 수 있다. $O_k^i$가 무한히 대기하는 것을 방지하기 위하여 타임아웃 기법을 사용한다.

## 4. 분석

제시된 GCC-M은 전역 직렬성을 보장하기 위하여 다음과 같은 조건을 가정하고 있다. 첫째, GTM은 전역 트랜잭션이 액세스하는 데이터 집합(data set)을 알고 있다. 둘째, LDBS는 2단계 로킹을 지원한다. 본 절에서는 각각의 가정을 만족하지 않는 경우 발생하는 문제점 및 그에 대한 해결 방안을 설명하도록 한다.

GTM에서 직접 충돌을 검사하여 이전 트랜잭션간 직접 충돌이 없는 곳에만 강제적인 충돌 연산을 추가한다. 이는 전역 트랜잭션이 액세스하는 데이터 집합을 알고 있다는 가정하에서 적용할 수 있는 것으로 만약 액세스하는 데이터 집합을 알지 못하는 경우 판독 연산과 갱신 연산 집합 $<RS_k^i, WS_k^i>$을 정의 할 수 없기 때문에 GTM에서 직접 충돌 유무의 검사가 어렵다. 따라서 모든 전역 트랜잭션에는 단계 2의 충돌 검사 단계는 필요하지 않고 OTM과의 티켓과 같은 공통된 데이터를 사용하여 강제적인 충돌을 발생시켜야만 한다. 이 경우 OTM의 문제점을 그대로 갖지만 단계 4와 같이 서브 트랜잭션은 로크를 순서대로 획득한다면 OTM에 비해 전역 교착상태 발생 가능성은 줄일 수 있을 것이다.

GCC-M에서 2PL을 지원한다고 가정하고 있다. 그 이유는 현재 상용화된 대부분의 LDBS가 지역 동시성 제어 기법으로 2PL을 채택하고 있기 때문이다. LDBS가 타임스탬프 순서 기법과 낙관적 기법을 지원하는 경우에도 적용이 가능하다. 타임스탬프 순서 기법을 지원하는 경우에는 2PL을 사용하는 경우에 비해 GCC-M 적용이 효과적이다. 타임스탬프 순서 기법은 트랜잭션과 데이터에 타임스탬프를 할당하고 트랜잭션의 타임스탬프 순서대로 데이터를 액세스하게 하여 순서를 만족하지 못하는 경우 해당 트랜잭션을 철회함으로써 직렬성을 보장하는 기법이다. 일반적으로 타임스탬프는 트랜잭션이 LDBS에 들어오는 순서를 의미하는 것으로 메시징 순서와 일치한다. 전역 트랜잭션 $T_{k-1}$, $T_k$가 있고 메시징 순서가 $T_{k-1} \rightarrow T_k$라고 가정하자. 타임스탬프 순서 기법은 전달 받은 순서대로 타임스탬프 할당한다고 보면 $ts(T_{k-1})$는 $T_{k-1}$의 서브 트랜잭션에, $ts(T_k)$는 $T_k$의 서브 트랜잭션에 할당되고 모든 노드에서 $ts(T_{k-1}) < ts(T_k)$가 된다. 또한 타임스탬프 순서 기법은 이 순서대로 트랜잭션의 실행을 보장해주기 때문에 순서대로 로킹하는 단계 4가 필요 없다. 따라서 모든 노드는 전달 받은 서브 트랜잭션의 병행 실행이 가능하여 2단계 로킹을 사용하는 경우에 비해 동시성을 증가시킬 수 있다.

트랜잭션을 먼저 실행한 후 직렬성을 검증하는 낙관적 기법의 경우는, 모든 노드에서 전달 받은 트랜잭션 순서대로 실행하였다는 것을 보장한다면 적용이 가능하다. 즉, 낙관적 기법을 사용하는 노드들은 서브 트랜잭션을 병행적으로 실행한 후 직렬성을 검증한다. 전역 트랜잭션 $T_{k-1}$, $T_k$가 있다면 낙관적 기법의 경우 실행 노드 $LDBS_i$에서 모든 충돌 연산이 $T_{k-1} \rightarrow T_k$ 순으로 실행될 수 있고, $LDBS_i$에서는 $T_k \rightarrow T_{k-1}$순으로 실행될 수 있다. 이를 해결하기 위하여 2PL 적용시와 마찬가지로 원자적으로 서브 트랜잭션을 실행하는 방법과 2PC를 지원하는 GTA에서 완료(commit) 순서를 검증하는 방법이 있을 수 있다. 만약 판독 연산이 대부분인 트랜잭션이라면 전자는 낙관적 동시성 제어 기법의 장점을 활용하지 못하게 되므로 후자의 경우가 더 적합할 것이다.

## 5. 결론 및 향후계획

MDBS는 일반적인 분산 데이터베이스 시스템 환경과 달리 통합되는 LDBS들의 서로 다른 질의 처리 능력을 제공할 뿐만 아니라, 지역 자치성 요구 사항 때문에 전역 트랜잭션 처리를 위한 상호 협조가 불가능하여 전역 직렬성을 만족하는 GCC의 설계가 어렵다. 메시징 소프트웨어는 응용 프로그램간 비동기식 통신 방법을 제공하는 것으로 기업간 응용 통합 인터페이스로 그 중요성이 증가하고 있다. 본 논문에서는 메시징 소프트웨어를 논리적 인터페이스로 활용한 GCC를 제안하였다. 제안된 기법은 메시징 순서에 따른 실행으로 LDBS의 실행 정보 없이 전역 트랜잭션의 상대적인 실행 순서를 결정할 수 있기 때문에 지역 자치성을 보장한다. 향후에는 기존의 기법과 제안된 기법을 성능평가 모형을 통하여 기존의 기법들과 성능을 비교하고자 한다.

## 참고 문헌


[1] J. Albert, "Theoretical Foundations of Scheme Restructuring in Heterogeneous Multidatabase Systems," *CKIM*, 2000, 461-470

[2] M. Wisemann, X. Defago and A. Schiper, "Group Communication based on Standard Interfaces," *IEEE intl. Symp. Network Computing and Applications*, 2003

[3] R. Monson-Haefel and D. Chappell, *Java™ Message Service*, O'Reilly, 2001

[4] D. Georgakopoulos, M. Rusinkiewicz and A. Sheth, "Using Tickets to Enforce the Serializability of Multidatabase Transactions," *IEEE Trans. on Knowledge and Data Eng.* 6(1), 1994, 166-180

[5] A. Zhang and A. Elmagarmid, "A Theory of Global Concurrency Control in Multidatabase Systems," *VLDB Journal*, 2(3), 1993, 331-360

[6] Y. Breitbart, H. Garcia-Molina and A. Silberschatz, "Overview of Multidatabase Transaction Management," *VLDB Journal*, 1(2), 1992, 72-79

[7] Fiorano, "FioranoMQ and Progress SonicMQ Highlights," *http://www.fiorano.com*, 2001

[8] P. Laumay, E. Bruneton, N. de Palam and S. Krakowiak, "Preserving Causality in a Scalable Message-oriented Middleware," *Lecture Notes in Computer Science*(2218), 311-328

[9] J. Holliday, D. Agrawal and A. Abbadi, "Using Multicast Communication to Reduce Deadlock in Replicated Databases," *IEEE Symp. on Reliable Distributed Systems*, 2000, 196-205

[10] B. Kemme and G. Alonso, "A New Approach to Developing and Implementing Eager Database Replication Protocols," *ACM Trans. Database Syst.* 25(3), 2000, 333-379

[11] A. Tanenbaum and M. Steen, *Distributed Systems - Principles and paradigms*, Prentice Hall, 2002