완료 트랜잭션 우선의 이동 실시간 트랜잭션 동시성 제어 기법

김 경 배[†] · 조 숙 경[††] · 배 해 영[†††]

요 약

이동 컴퓨팅 기술이 발전함에 따라 실시간으로 트랜잭션을 처리해야 하는 다양한 이동 애플리케이션에 대한 요구가 증가되고 있다. 원격 데이터에 대한 접근을 해야 하는 이동 환경에서 데이터에 대한 접근지연은 실시간 트랜잭션의 시간적인 제약조건 준수에 대한 가장 심각한 문제가 된다. 이동 실시간 트랜잭션은 트랜잭션의 결과의 정확성뿐만 아니라 트랜잭션의 완료 시간을 보장해야 한다. 본 논문에서는 이동 실시간 트랜잭션들 간의 충돌을 해결하기 위한 낙관적인 동시성제어 기법을 제안한다. 제한된 기법은 이동 환경에서의 핸드오버나 단절에 의해 발생되는 트랜잭션의 연속적인 철회와 지연의 영향을 최소화 하였다.

A Concurrency Control Method of Mobile Real-time Transactions Using Committed Transaction Precedence

Gyoung-Bae Kim[†] · Sook-Kyoung Cho[††] · Hae-Young Bae[†††]

ABSTRACT

With the significant advances in mobile computing technology, there is an increasing demand for various mobile applications to process transactions in a real-time. When remote data access is considered in a mobile environment, data access delay becomes one of the most serious problems in meeting the deadline of real-time transaction. The mobile real-time transaction should be assured not only correctness of result of transaction but also completion time of transaction. In this paper, we propose an optimistic concurrency control method to solve conflict among mobile real-time transactions. It minimizes influence on the cascade abort and delay of transactions that occur by disconnection and hand over in a mobile environment.

키워드 : 이동 컴퓨팅(Mobile Computing), 이동 실시간 트랜잭션(Mobile Real-time Transaction), 동시성 제어(Concurrency Control)

1. 서 론

최근 들어 컴퓨팅 환경에 커다란 진화적인 변화가 이루어지고 있다. 그 하나는 기존의 데스크 탑 컴퓨터와 비교해서 기능이나 성능이 거의 동일하고 휴대하기에 충분히 작은 휴대용 컴퓨터의 대중화이며, 다른 하나는 컴퓨터 통신을 위한 구조로 이동전화, 무선 랜, 그리고 위성서비스로 대별되는 무선 매체를 이용한 영역의 확대이다. 이들의 급속한 발전과 더불어 고유의 특성에 따른 상호 요구가 자연스럽게 서로 결합하는 형태로 나타나게 되었으며, 그 결과로 휴대용 컴퓨터를 이용하는 사용자들의 네트워크 접속 형태는 기존의 유선 네트워크의 장소 제한성을 탈피할 수 있게 되었다. 즉, 사용자는 네트워크의 접속을 유지하면서 원하는 장소로의 자유로운 이동이 가능하게 되었다. 이러한 새로운 컴퓨팅 패러다임을 이동 컴퓨팅(mobile computing)[1, 2]이라 한다.

무선 통신망의 발전과 컴퓨터 하드웨어의 크기와 성능 개선으로 인한 이동 컴퓨팅 환경의 등장은 실시간 데이터베이스 시스템 사용자들이 고정된 망에서뿐만 아니라 이동 컴퓨팅 환경에서도 고정망 수준의 실시간 처리 능력을 요구하게 됨으로써 이동 컴퓨팅 환경에서의 이동 실시간 데이터베이스 시스템(MRTDBMS:mobile real-time database systems)[3, 4]에 대한 연구 필요성이 크게 대두되었다. 이동 컴퓨팅 환경을 위한 실시간 데이터베이스 시스템에서 발생되는 트랜잭션은 이동 트랜잭션(mobile transaction)과 실시간 트랜잭션(real-time transaction)이 결합되어 있으며, 이를 이동 실시간 트랜잭션(real-time mobile transaction)[5-7]이라 한다.

※ 본 연구는 대학 IT연구센터 육성·지원사업의 연구 결과로 수행되었음.
† 종신회원 : 서원대학교 컴퓨터교육과 교수
†† 정 회 원 : 인천대학교 컴퓨터공학과 교수
††† 종신회원 : 인하대학교 컴퓨터공학부 교수
논문접수 : 2004년 7월 12일, 심사완료 : 2004년 10월 12일

기존의 이동 트랜잭션은 트랜잭션의 일부가 고정 호스트에서뿐만 아니라 이동 호스트에서 실행되는 분산 트랜잭션이라는 특성을 지니고 있고, 실시간 트랜잭션은 이동 트랜잭션 트랜잭션이 시간제약조건 내에 처리되어야 한다는 특성을 지닌다. 따라서 이동 실시간 트랜잭션은 이동 컴퓨팅 환경에서 실시간 서비스를 지원하기 위한 트랜잭션으로 이동 트랜잭션의 특성을 지니고 있으며, 실시간적으로 발생되는 동적 데이터를 정해진 제약조건 내에 처리해야 하는 트랜잭션을 의미한다.

본 논문에서는 이동 컴퓨팅 환경에서 이동 실시간 트랜잭션을 효율적으로 처리하기 위해 이동 호스트에 접속 단절 현상이나 핸드오버 등으로 인한 트랜잭션의 지연이나 연속적인 철회로 다른 트랜잭션의 실행에 미치는 영향을 최소화하기 위한 완료 트랜잭션 우선의 낙관적인 동시성 제어 기법을 제안한다. 제안된 기법은 트랜잭션의 시간적인 제약조건을 최대한 만족시키며, 트랜잭션의 지연이 빈번하게 발생하는 이동 컴퓨팅 환경에서도 기존의 기법에 비해 효율적으로 트랜잭션을 처리할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구로 실시간 이동 트랜잭션에서 사용되는 동시성제어 기법에 대하여 기술하고, 3장에서는 본 논문에서 제안하는 기법에 대하여 설명하고, 4장에서 제안 기법에 대한 성능평가를 수행하고, 5장에서 논문의 결론을 맺는다.

2. 관련 연구

실시간 데이터베이스 시스템의 성능을 향상시키기 위해 현재까지 연구된 동시성 제어 기법은 데이터의 로크를 이용하는 2단계 로킹 기법과 데이터 충돌의 발생을 낙관적으로 생각하여 데이터 로크를 사용하지 않고 트랜잭션을 수행하는 낙관적인 동시성 제어 기법을 실시간 트랜잭션의 처리에 적합하게 변형한 기법들이다. [8-10]에서 수행한 성능평가의 결과에 의하면, 충돌이 적게 발생하고 트랜잭션의 길이가 짧거나 무한대의 자원이 존재하는 경우에는 낙관적인 기법이 우수하지만, 반대로 트랜잭션의 충돌이 빈번하게 발생하고 제한된 자원 하에서 트랜잭션들이 경쟁하는 경우에는 로킹 기법이 더 우수하다.

2단계 로킹 기법을 이용한 동시성 제어 기법으로는 [8]에서 제안한 2PL-HP(2 Phase Locking-High Priority) 알고리즘과 2PL-CR(2 Phase Locking-Conditional Restart) 알고리즘 그리고 2단계 로킹 기법에 순서 공유(ordered sharing)를 적용시킨 [9]의 2PL-OS(2 Phase Locking-Ordered Sharing) 알고리즘 등이 있다. 그러나, 2단계 로킹 기법을 사용한 동시성 제어 기법은 교착상태를 고려해야 하며, 높은 우선순위의 트랜잭션이 낮은 우선순위의 트랜잭션을 기다리는 우선순위 역행 현상 또한 고려해야 한다. 2PL-HP는 충돌이 발생한 경우 높은 우선순위의 트랜잭션이 데이터에 대한 로크를 획득하는 방법이다. 만약 현재 데이터에 대한 로크를 요청한 트랜잭션의 우선순위가 요청된 데이터에 대한 우선순위보다 높다면 로크를 소유하고 있는 트랜잭션을 취소시키고 새로운 트랜잭션이 데이터에 대한 로크를 얻도록 한다. 반대로 요청한 트랜잭션의 우선순위가 요청된 데이터 객체의 우선순위보다 낮거나 같으면 트랜잭션은 데이터 객체에 대한 로크가 해제될 때까지 대기를 한다. 기존의 공유 로크와 비공유 로크에 순서 공유라는 새로운 로크를 첨가하여 트랜잭션의 블로킹 수를 줄이는 동시성 제어 기법이다. 순서 공유 로크는 트랜잭션들 간에 순서 공유를 소유한 순서대로 순서 공유 로크에 대응하는 연산을 수행해야 하는 제약조건을 가진다. 실시간 동시성 제어를 위한 2PL-OS은 순서 공유를 사용하기 위한 순서 공유 규칙에 트랜잭션의 시간적인 제약조건을 첨가시켜 2PL-OS를 수정한 알고리즘을 사용한다. 성능 비교를 통하여 2PL-OS가 2PL보다 우수하다는 것을 보였다.

[10]에서는 기존의 낙관적인 동시성 제어 기법에 시간적인 제약 조건을 포함하는 동시성 제어 기법을 제안하였고, 실험을 통해서 소프트 실시간 트랜잭션의 환경 하에서는 2단계 로킹 기법보다 더 우수하다는 것을 증명하였다. OPT-WAIT[11]은 타당성 검증단계에서 데이터 충돌이 포함된 상위의 우선순위 트랜잭션이 완료될 때까지 기다리게 함으로써 불필요한 시스템의 낭비를 줄인 것이다. 따라서 상위의 트랜잭션에 종료시한을 만족할 수 있는 기회를 주는 것이다. 이 알고리즘은 실제적으로 데이터의 충돌을 감소시킬 수 있는 가능성을 지니고 있으나, 트랜잭션이 종료시한까지 대기한 후 종료 시에 CLP(Conflict Low Priority)에 포함된 트랜잭션의 철회로 인한 자원의 낭비와 CHP(Conflict High Priority)에 포함된 트랜잭션의 완료시간을 기다리므로 충돌하는 트랜잭션의 수가 증가하는 단점이 있다. WAIT-50은 OPT-WAIT에 우선순위 대기 기법을 첨가하여 확장시킨 기법으로 데이터의 충돌에 포함된 트랜잭션들 중에 50% 이상이 CHP에 포함된 트랜잭션이라면 OPT-WAIT처럼 대기를 하고, 전체 충돌 집합의 크기의 비율인 HPpercent를 이용하여 대기 기법의 조건 값으로 사용한다. WAIT-50에서 타당성 단계의 트랜잭션은 HPpercent $\geq$ 50인 경우에만 대기한다. 즉, 높은 우선순위로 구성된 트랜잭션의 절반 이상인 경우 대기를 한다. 이 알고리즘은 CHP에 포함된 트랜잭션의 수가 적으면 하위의 우선순위를 갖는 트랜잭션이라도 CHP에 포함된 트랜잭션을 철회할 수 있으므로 하위의 우선순위를 갖는 트랜잭션이라도 항상 취소되지 않는다는 특징을 지닌다.

[11]에서는 낙관적인 동시성 제어 기법에 대한 구현의 정확성을 보장하기 위하여 로크 기법을 사용하였다. 이 기법은 R-Lock(Read Phase-Lock)와 V-Lock(Validation Phase-Lock) 두

개의 로크를 사용하며 타당성 검사 트랜잭션이 임계 구역에서 V-Lock를 얻게 함으로써, 교착상태가 없고 병행성을 높일 수 있지만 분산환경에서는 한 트랜잭션이 여러 개의 사이트를 접근하는 부 트랜잭션으로 나누어져 각 사이트에서 연산을 진행한다. 이러한 전역 트랜잭션의 직렬성을 보장하기 위해 사용되는 가장 보편적인 규약은 2단계 완료 규약(2PC : 2 Phase Commit)을 사용하며, 이러한 지역 직렬성을 보장하기 위해 2단계 로크(2PL : 2 Phase Lock)가 함께 사용된다. 2단계 완료 규약과 2단계 로크가 결합되어 사용되는 경우에는 2단계 완료 규약은 참여자의 불확실성으로 인한 블로킹 현상이 발생한다. 즉, 참여자 사이트에서 실행된 트랜잭션은 조정자 사이트로부터 최종적인 결정 메시지를 수신할 때까지 자신이 소유한 데이터에 대한 로크를 소유하게 되어 해당 데이터를 필요로 하는 다른 트랜잭션의 실행을 지연한다.

[12]에서는 이동 환경에서 실시간 트랜잭션을 위한 동시성 제어 기법으로 다중 버전 데이터 모델(multi-version data model)을 제안하였다. 이 기법은 실시간 특성을 지니는 이동 데이터베이스 시스템 환경에서 상대적인 일관성 요구조건(relative consistency requirement)과 시간적인 제약 조건을 만족시키기 위한 방법으로 다중 버전 기법을 적용하였다. 실시간 트랜잭션을 위한 다중 버전 기법은 응용프로그램에서 미리 정의된 트랜잭션의 특성을 이용하여 각 서브 트랜잭션이 상대적인 일관성 조건을 만족하는 데이터들을 미리 읽어서(pre-fetching) 트랜잭션을 수행하여, 데이터의 지연을 제거한 기법이다.

3. 완료 트랜잭션 우선의 동시성 제어 기법

이동 컴퓨팅 환경에서 동시성제어는 이동 호스트와 이동 지원국간의 무선 통신으로 연결되어 데이터를 처리하기 때문에 접속지연, 접속단절, 그리고 이동 호스트의 이동성에 의한 새로운 문제들이 발생한다[13]. 각 이동 호스트들이 객체 X에 대한 복사본을 유지하고 있는 경우 임의의 이동 호스트에서 갱신연산을 수행하는 경우, 갱신을 수행하기 위한 이동 호스트는 X를 변경하기 전에 데이터에 대한 로크를 이동 지원국에 요청하고, 이동 지원국은 요청된 데이터에 대한 로크를 허용하기 위해 복사본을 유지하고 있는 이동 호스트들을 조사하고, 해당 사이트로 데이터에 대한 로크를 요청하게 된다. 그러나 이동 컴퓨팅 환경에서는 연결의 불안전성으로 인해 이동 호스트와 접속 단절 현상이 발생하게 되면, 이동 호스트로부터 데이터의 갱신에 대한 동의를 즉시 얻지 못하고 재접속이 이루어지기까지 기다려야 하는 지연의 문제가 발생한다. 이러한 지연의 문제는 트랜잭션의 수행 시간을 지연하게 되어 결과적으로 실시간 트랜잭션이 지니는 제약조건을 만족시키지 못하는 문제점을 야기한다. 또한, X에 대한 복사본을 지닌 호스트와의 접속 단절 현상이 발생하면 단절된 이동 호스트로부터 데이터의 갱신에 대한 동의를 얻지 못하게 되고, 해당 트랜잭션은 재접속이 이루어지기까지 무한정 기다려야 하므로 트랜잭션의 수행 시간을 지연하게 되어 결과적으로 실시간 트랜잭션이 지니는 제약조건을 만족시키지 못하는 문제점을 야기한다.

이동 지원국 A에 의해 고정망에 접속되어 있던 이동 호스트가 기존의 셀 영역을 벗어나 이동 지원국 B가 관할하는 새로운 셀로 핸드 오버한 경우 이동 호스트가 사용 중인 데이터에 대한 정보는 이동 기지국 A에서 관리하고 있으나, 현재 이동 호스트는 이동 지원국 B에 접속되어 있다. 따라서 이동 호스트에서 사용하는 데이터에 대한 정보를 어디에서 관리할 것인가 하는 문제가 발생한다. 만약 트랜잭션이 발생되어 이동 지원국 A에서 계속적인 관리를 하는 경우에는 이동 호스트에서 사용하는 데이터에 대한 로크 정보를 새롭게 접속한 이동 지원국 B에 전송해야 하고, 수행중인 트랜잭션이 종료되면 해당 트랜잭션에서 사용하던 데이터에 대한 로크의 해제를 이동 지원국 B가 이동 지원국 A에 요구해야 한다. 따라서 이러한 핸드오버가 발생한 경우에 트랜잭션 수행시간의 지연을 초래하게 되고 해당 트랜잭션은 종료시한을 초과하게 될 가능성이 크게 증가하게 된다.

로킹 기반 기법은 유선 망에서 트랜잭션의 직렬화 가능 스케줄을 생성하며 일반적으로 정확성과 데이터의 충돌이 빈번한 경우에도 좋은 성능을 제공한다. 고정망 환경에서 넓은 대역폭을 통해서 고속의 데이터 전송이 가능하고 높은 신뢰성을 보장하는 경우에는 이러한 로킹 기반의 동시성 제어 기법이 좋은 성능을 나타낸다. 그러나, 기존 기법을 이동 컴퓨팅 환경에 그대로 적용하는 경우에는 낮은 대역폭과 이동 호스트와 연결의 잦은 접속 단절 등으로 인해 메시지의 손실 및 전송의 지연이 발생한다. 따라서 해당 트랜잭션의 철회와 재시작을 유발하거나 해당 자원에 대한 로크의 시간이 증가하게 되어 시간과 비용 면에서의 낭비를 초래한다.

실시간 특성을 갖는 이동 컴퓨팅 환경에서 갱신은 실시간 데이터를 중심으로 주로 고정된 망에 연결된 실시간 데이터베이스 시스템에서 발생되고, 이동 호스트를 중심으로 한 무선망에서는 이러한 데이터에 대한 판독연산을 주로 하고 있다. 따라서 트랜잭션간의 충돌이 적게 발생하게 됨으로 로킹을 기반으로 한 기법보다 낙관적인 기법이 효율적이다. 또한, 이동 호스트의 접속 단절과 핸드오버의 특성으로 인해 본 논문에서는 낙관적인 기법을 기반으로 한 동시성 제어 기법을 제안하며, 직렬성의 보장을 위해서 로크를 이용한다.

3.1 로크 모드와 양립성 함수

각 트랜잭션 Ti는 트랜잭션이 판독연산을 수행하는 데이터에 대한 판독 집합인 RS(Ti)와 기록연산을 수행하는 데이터에 대한 기록 집합인 WS(Ti)를 유지한다. 동시성제어를

위해 트랜잭션은 <표 1>의 네 가지 중 하나를 설정한다.

N-Lock는 트랜잭션이 정확한 결과의 도출을 요구하지 않는 판독 전용 트랜잭션인 경우에 데이터의 충돌을 전혀 고려하지 않고, 트랜잭션을 수행하는 경우 사용하기 위한 로크이다. 따라서 N-Lock은 다른 트랜잭션과의 충돌 검사를 하지 않고 트랜잭션을 수행하고 수행이 완료되면 트랜잭션을 종료한다.

〈표 1〉 로크모드와 양립성 함수

Request \ Hold	N-Lock	RP-Lock	VP-Lock	F-Lock
N-Lock	○	○	○	○
RP-Lock	○	○	restart	restart
VP-Lock	○	C-wait	C-wait	restart
F-Lock	○	○ (restart)	○ (restart)	○ (restart)

N-Lock : No Lock
RP-Lock : Read Phase Lock
VP-Lock : Validation Phase Lock
F-Lock : Force Lock
C-wait : conditional wait

RP-Lock(Read Phase Lock)은 판독 단계에 도달한 트랜잭션을 위한 로크로 트랜잭션은 자신이 참조할 데이터에 대한 로크를 이동 지원의 로크 관리자에게 요청한다. VP-Lock(Validation Phase Lock)는 검증 단계에 도달한 트랜잭션이 갱신을 수행하기 위해 요청하는 로크로 트랜잭션의 수행을 완료한 후에 갱신이 발생된 데이터에 대한 충돌 여부를 검사하기 위한 로크이다. F-Lock(Force Lock)은 강제적인 갱신이 필요한 트랜잭션을 위한 로크이다. 이동 컴퓨팅 환경에서 센서 트랜잭션과 같이 실시간 데이터에 대한 갱신 연산이 발생한 경우에 F-Lock을 사용하며, 해당 데이터에 대한 복사본을 사용하는 모든 중복 서버에 저장 중인 값도 변경해 주어야 한다.

트랜잭션은 트랜잭션을 시작시점에서 사용하는 데이터에 대한 RP-Lock을 설정한다. 만약 RP-Lock을 설정하고자 하는 데이터 객체에 대하여 이 VP-Lock이 설정된 경우에는 기존에 다른 트랜잭션에서 해당 객체에 대한 갱신연산을 수행하므로 로크가 해제될 때까지 대기한다. 반대로 현재 VP-Lock나 RP-Lock이 설정된 데이터 객체에 대해 VP-Lock가 요청되면 트랜잭션간의 충돌이 발생한 것이므로 트랜잭션간의 충돌을 해결하기 위한 기법을 적용하여 트랜잭션을 완료할 것인지 철회할 것인지를 결정한다. F-Lock은 실시간 데이터의 갱신과 같이 기존에 수행 중인 트랜잭션을 취소시키고 데이터베이스에 최신의 데이터 값을 반영해야 하는 경우에 사용한다. 따라서 F-Lock이 요청되면 N-Lock을 제외한 각 트랜잭션은 중단하고 재수행을 해야 하며, 설정된 각 로크는 트랜재션의 완료 시점에 해지한다.

3.2 판독 단계

판독 단계에서는 트랜잭션의 연산을 수행하는데 객체를 데이터베이스로부터 판독해서 트랜잭션의 작업영역에 지역 사본을 만들어 관리한다. 만약 요구된 객체가 이동 호스트에 존재하는 경우 이를 이용하지만 이동 호스트의 캐쉬에 존재하지 않는 경우에는 이동 지원국에 해당 데이터의 전송을 요구한다. 트랜잭션에서 사용하는 데이터 객체에 대해서는 이동 지원국에 데이터에 대해 판독 집합 RS(Ti)와 갱신 집합 WS(Ti)를 기록하고, 트랜잭션을 시작하는 시점의 판독 단계에서 RP-Lock을 설정한다. RP-Lock의 설정에서 다른 트랜잭션과 VP-Lock이나 RP-Lock가 충돌하면 우선순위와 트랜잭션의 타임스탬프 값을 비교하여 재시작 여부를 결정한다. 만약 트랜잭션이 재 시작되면 해당 트랜잭션이 사용하는 데이터에 대한 갱신이 발생한 것이므로 최신의 데이터를 이동 호스트로 복사하여 트랜잭션을 다시 시작한다.

3.3 검증 단계

검증 단계는 판독 단계에서 사본에 반영된 트랜잭션의 실행 결과를 데이터베이스에 반영할 때 직렬 가능성의 위반 여부를 검사한다. 즉, 판독 단계를 지난 트랜잭션들은 서로 간섭이 거의 없이 트랜잭션을 먼저 실행하고 검증 단계에서 충돌 여부를 검사한다.

검증 단계에 도달한 트랜잭션은 이동 기지국에 해당 데이터 객체에 대한 VP-Lock을 요청한다. 요청된 VP-Lock은 로크의 양립성 함수를 이용하여 허용 여부를 검사한다. 만약 VP-Lock가 N-Lock과 충돌이 발생하면 무시하고 갱신 단계로 진행하지만 R-Lock, VP-Lock, F-Lock와 충돌이 발생하면 충돌 해결을 해야 한다.

낙관적인 기법에서 검증단계에 도달한 트랜잭션 Ti와 트랜잭션 Tj 간의 다음 조건 가운데 하나를 만족하면 데이터 간에 충돌이 발생하지 않았으므로 다음 단계로 진행한다.

① 트랜잭션 Ti의 기록 단계가 Tj의 판독 단계가 시작하기 전에 완료됨.

$$TransFinish(Ti) < TransStart(Tj)$$

② Tj가 검증 단계를 시작하기 전에 Ti는 기록 단계를 완료하였으며, Tj의 판독 집합과 Ti의 기록 집합 사이에는 공통 데이터가 없다.

$$TransStart(Tj) < TransFinish(Ti) < TransValidation(Tj) \text{ and } W\text{-}Set(Ti) \cap R\text{-}Set(Tj) = \phi$$

③ Tj의 판독 단계가 끝나기 전에 Ti는 판독 단계를 완료하였으며 Tj의 기록 집합과 판독 집합이 Ti의 기록 집합과 아무런 공통 데이터가 없다.

$$TransValidation(Ti) < TransValidation(Tj) \text{ and } W\text{-}Set(Ti) \cap R\text{-}Set(Tj) = \phi \text{ and } W\text{-}Set(Ti) \cap W\text{-}Set(Tj) = \phi$$

3.4 임계값 α를 이용한 충돌 해결 방법

3.4.1 트랜잭션 충돌의 발견

낙관적인 동시성 제어 기법에서는 검증 단계에서 트랜잭션의 충돌 여부를 검사한다. 즉, 트랜잭션을 수행에서 판독 단계를 거쳐 연산을 수행하고 검증 단계에 도달하게 되면, 충돌의 발생 여부를 검사하고 충돌을 해결하기 위한 방법을 사용한다. 충돌을 발견하기 위해서는 트랜잭션 Ti와 Tj에서 사용하는 데이터 중에서 서로 간에 데이터의 충돌이 있는지를 검사한다. 이러한 충돌 검사 기법으로는 전진 검증(forward validation)과 후진 검증(backward validation)이 있다.

제안된 기법에서는 전진 검증 기법을 사용한다. 전진 검증 방법은 현재 수행중인 트랜잭션을 대상으로 충돌을 검사한다. 검증하고 있는 트랜잭션이 직렬화 순서상 읽기 단계에서 동시 수행중인 모든 트랜잭션들보다 앞선다는 가정 하에 데이터 충돌의 검출은 검증하고 있는 트랜잭션의 갱신 집합과 활동 중인 트랜잭션의 판독 집합을 비교하여 수행된다. 즉, 검증할 트랜잭션 Tj의 갱신 집합과 동시에 수행중인 판독 단계의 트랜잭션의 판독 집합을 가지고 검사하게 된다. 만약, 검증하고 있는 트랜잭션이 쓰기를 한 데이터 객체를 다른 트랜잭션이 읽으면, 그 트랜잭션에 의해 사용된 객체의 값은 일치성을 잃어버리게 된다. 이러한 경우 데이터 충돌은 검증하고 있는 트랜잭션이 나 혹은 기록 단계에서 충돌한 트랜잭션들을 재 시작함으로써 해결할 수 있다.

3.4.2 충돌 해결 기법

낙관적인 제어 기법에서 트랜잭션의 충돌을 해결하기 위해서는 충돌이 발생한 트랜잭션을 재 시작해야 한다. 따라서 실시간 데이터베이스 시스템에서 재 시작할 트랜잭션의 선택은 매우 중요하며 트랜잭션의 시간 정보뿐만 아니라 트랜잭션의 중요도와 트랜잭션의 유형을 고려하여 선택해야 한다. 본 논문에서 제안하는 기법은 트랜잭션의 충돌해결을 위한 기법으로 트랜잭션의 중요도와 종료시한뿐만 아니라 접속 단절 현상 등에 의한 트랜잭션의 재시작의 문제를 해결하기 위해 충돌이 발생한 트랜잭션을 무조건 철회하지 않고 트랜잭션의 종료 시한까지 철회를 지연한다. 즉, 충돌된 트랜잭션의 우선순위가 높은 경우에는 충돌한 낮은 우선순위 트랜잭션을 취소하지만, 만약 높은 우선순위의 트랜잭션의 통신상의 문제로 인해 지연되어 철회되면, 검증 단계에 도달한 낮은 우선순위의 트랜잭션의 철회로 인해 자원의 낭비가 발생한다. 따라서 충돌한 트랜잭션의 비율을 나타내는 임계 값 α와 연결 상태를 이용하여 제안 기법에서는 낮은 우선순위의 트랜잭션을 무조건 철회하지 않고, 종료시한까지 트랜잭션의 철회를 연기하여 우선순위가 낮은 트랜잭션이 완료될 수 있는 기회를 준다. 지연된 트랜잭션의 종료시한이 도달하게 되면 트랜잭션을 종료시킬 것인지 재 시작할 것인지를 결정한다. α값은 트랜잭션의 수행 시에 임의로 설정할 수 있다. α의 실제 설정값은 적용되는 응용 프로그램의 특성에 크게 좌우되며, 트랜잭션의 처리에서 사용하는 우선순위의 범위와 트랜잭션의 수행시간 등에 영향을 받는다. 즉, 트랜잭션의 우선순위의 범위가 큰 경우에 α값은 커지고, 수행시간이 긴 경우에는 작은 값을 갖는다.

3.5 트랜잭션 지연에 의한 교착상태 방지

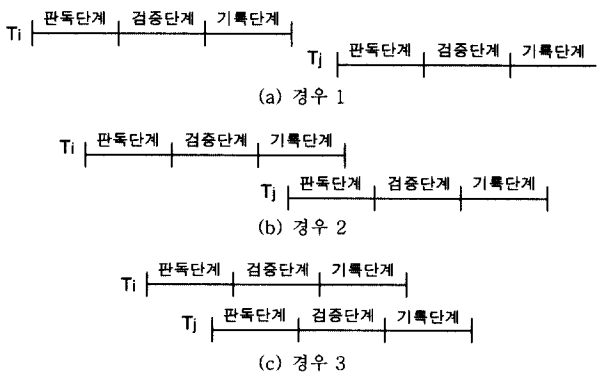
이동 컴퓨팅 환경에서 이동 호스트의 접속 단절이 발생하는 경우 트랜잭션의 지연 내지는 교착상태가 발생한다. 예를 들면, 이동 호스트가 트랜잭션을 실행하는 도중에 배터리의 전원이 다 소모되어 접속의 단절이 발생하게 되면 수행중인 트랜잭션을 완료할 수 없고 시스템은 교착상태에 빠지게 되어 전체 시스템의 성능은 크게 저하된다. 본 논문에서 제안하는 기법은 접속 단절 상태의 이동 호스트에서 진행중인 트랜잭션이 종료시한을 초과 한 후에도 재접속이 설정되지 않으면 해당 트랜잭션에서 사용중인 데이터에 설정된 로크를 해제하고 해당 트랜잭션에 대한 정보에 트랜잭션이 취소되었음을 기록한다. 접속 단절 현상이 발생했던 이동 호스트가 이동 지원국에 재접속되면 접속된 이동 지원국에 트랜잭션의 취소 여부를 검사하고 트랜잭션이 취소된 경우에는 해당 트랜잭션을 재 수행한다.

3.6 알고리즘의 정확성 증명

제안된 동시성 제어 기법은 트랜잭션 Ti가 완료될 때 충돌이 발생한 트랜잭션들을 취소함으로써 직렬 순서가 보장된다. 따라서 알고리즘의 정확성에 대한 증명은 제안된 알고리즘에 의해 생성된 모든 수행 기록은 주기를 갖지 않는 직렬성 그래프 만을 생성함으로써 충돌한 트랜잭션의 타임스탬프 값이 TS(Ti) < TS(Tj)인 경우 실행된 스케줄은 반드시 Ti가 Tj보다 먼저 나오는 직렬 스케줄과 동등하다는 것을 증명한다[14].

제안 기법에 의해 생성된 수행 기록을 H라 하고, SG(H)를 H에 대한 직렬성 그래프라고 하자. 즉, SG(H)는 종료된 트랜잭션을 정점으로 표현하고 충돌한 트랜잭션간의 연산순서를 간선으로 나타내는 방향 그래프이다. 수행 기록 H가 직렬 가능함을 보이기 위해서 SG(H)가 비주기(acyclic)임을 증명한다.

[정의] Ti와 Tj를 제안된 알고리즘에 의해 생성된 수행기록, H에서 완료된 트랜잭션이라고 하자. 만약, SG(H)상에서 간선 Ti → Tj가 존재하면 TS(Ti) < TS(Tj)이다.



(그림 1) 제안된 알고리즘에서 생성 가능한 트랜잭션 스케줄

(그림 1)은 제안된 알고리즘에 의해서 생성 가능한 트랜잭션의 스케줄로써 각각의 경우에 트랜잭션간의 직렬성이 유지가 됨을 보인다.

[경우1] $r_i[x] \rightarrow w_j[x]$

이 경우는 (그림 1)(a)와 같이 트랜잭션이 스케줄링 되는 경우에 발생되며 Ti의 읽기 $r_i[x]$가 Tj의 $w_j[x]$에 영향을 미치지 않고, Tj가 검증 단계에 도착하기 전에 Ti가 완료되는 경우이다. 제안 기법에서 Ti가 검증 단계에 도착하였을 때 Ti가 x에 대한 쓰기 연산을 하지 않으면 갱신을 위한 데이터 집합인 WS(Ti)내에 데이터 항목 x가 존재하지 않는다. 이 경우에는 검증 단계에서 충돌이 발생하지 않는 경우이기 때문에 트랜잭션 Ti가 먼저 완료된다. 따라서, TS(Ti) < TS(Tj)가 된다.

[경우 2] $w_i[x] \rightarrow r_j[x]$

이 경우는 (그림 1)(b)와 같이 트랜잭션이 스케줄링 되는 경우에 발생되며 Tj가 판독 단계에서 rj[x]를 끝내기 전에 Ti의 기록 단계가 끝나는 경우이다. 제안 기법에서는 Ti가 검증 단계에 도달하게 되면 충돌한 트랜잭션 Tj의 읽기 단계에서 해당 데이터 x에 대한 충돌의 발생을 검사하게 되어 충돌이 발생한 트랜잭션 Tj는 반드시 취소되기 때문에 항상 TS(Ti) < TS(Tj)가 된다.

[경우 3] $w_i[x] \rightarrow w_j[x]$

이 경우는 (그림 1)(c)와 같이 트랜잭션이 스케줄링 되는 경우에 발생되며 Tj가 검증 단계에 도착하기 전에 Ti의 쓰기 단계가 끝나거나 Tj가 검증 단계에 도착한 후에 Ti의 쓰기 단계가 끝나는 경우이다. 제안 기법에서는 검증 단계와 쓰기 단계가 동일한 임계 구역에서 수행이 되기 때문에 항상 먼저 도착한 트랜잭션이 먼저 쓰기를 하고 완료된다. 따라서, TS(Ti) < TS(Tj)가 된다.

[정리] 제안된 알고리즘에 의해 생성된 모든 수행기록 H는 직렬가능(serializable)하다.

[증명] H를 제안한 알고리즘에 의해 생성된 수행기록이라고 하고, SG(H)가 주기 $T1 \rightarrow T2 \rightarrow \cdots\cdots \rightarrow Tn \rightarrow T1$을 갖는다고 가정하자. 보조 정리에 의해서 $TS(T1) < TS(T2) < \cdots\cdots < TS(Tn) < TS(T1)$을 얻게 되고, 결과적으로 $TS(T1) < TS(T1)$가 되어 모순이다. 따라서, SG(H)에는 주기가 존재하지 않기 때문에 제안 알고리즘은 직렬 가능한 수행기록만을 생성한다.

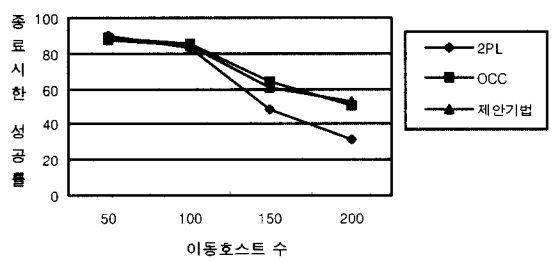
4. 성능평가

성능평가를 위해서 [4]에서 사용한 성능 평가 방법 및 인자를 사용하였으며, 기본 설정 값은 <표 2>와 같다. 이동 호스트를 지원하기 위한 이동 지원국인 고정 호스트의 개수는 10개로 설정하였다. 1페이지의 크기는 4,096바이트이고, 전체 데이터베이스의 크기는 10,000페이지로 하였으며, 이동 호스트가 가지는 지역 데이터베이스의 크기는 200페이지로 하였다. 이동 호스트와 고정망과의 데이터 전송속도는 2Mpbs로 설정하였고, 고정망에서 고정 호스트간의 데이터 전송속도는 100Mbps로 설정하였다.

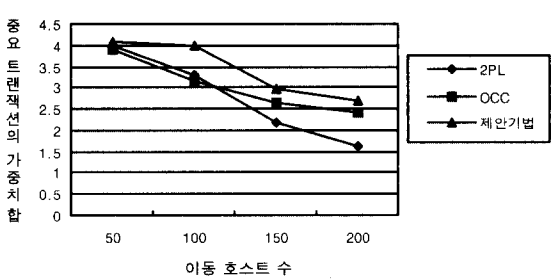
〈표 2〉 성능평가 인자의 기본 설정값

인 자	설 정 값
NumFHosts	10
TotalDBSize	10,000pages
LocalDBSize	200pages
PageSize	4096bytes
MemSize	100pages
PageCPUTime	10msec
MsgCPUTime	4msec
WiredBand	100Mbps
WirelessBand	2Mbps
HandoverInt	3sec

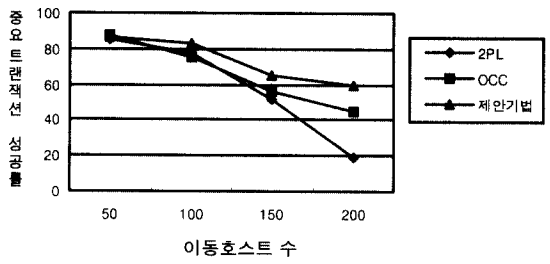
성능평가는 본 논문에서 제안하는 동시성 기법과 실시간 데이터베이스 시스템에서 제안된 2단계 로킹 기법[8], 낙관적인 동시성제어 기법[11]과의 성능평가를 수행한다. 평가 항목으로는 종료시한을 만족시키는 트랜잭션의 비율과 중요도의 합, 그리고 반드시 수행해야 하는 트랜잭션의 성공비율을 기준으로 성능평가를 수행하였다.



(그림 2) 호스트 수 증가에 따른 트랜잭션의 종료시한 성공률

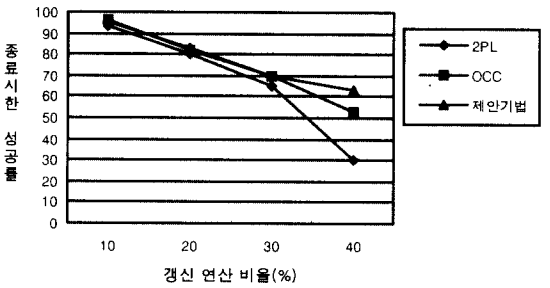


(그림 3) 이동 호스트 수 증가에 따른 중요 트랜잭션의 가중치 합

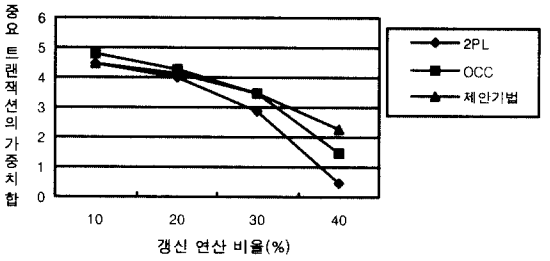


(그림 4) 이동 호스트 수 증가에 따른 중요 트랜잭션의 성공률

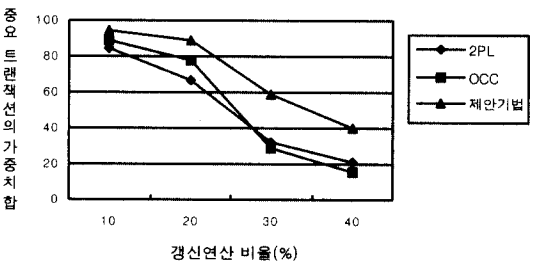
(그림 2), (그림 3), (그림 4)는 이동 호스트의 수를 증가시키고, 그에 따른 종료시한을 만족하는 트랜잭션의 비율, 중요한 트랜잭션의 성공 비율, 그리고 트랜잭션에 할당된 중요도의 만족률을 비교 평가한 것이다. 전체적으로 제안 기법이 기존의 기법에 비해 우수한 성능을 보이고 있다. 특히, 중요 트랜잭션의 성공률은 30%이상 높은 성공률을 보이고 있다.



(그림 5) 갱신 비율에 따른 트랜잭션의 종료시한 성공률



(그림 6) 갱신 비율에 따른 중요 트랜잭션의 가중치 합



(그림 7) 갱신 비율에 따른 중요 트랜잭션 성공률

(그림 5), (그림 6), (그림 7)은 판독 트랜잭션과 갱신 트랜잭션의 비율의 변화에 따른 트랜잭션의 성공률을 평가한 것이다. 기존의 기법은 기록 트랜잭션의 비율이 증가함에 따라 전체적인 트랜잭션들간의 충돌횟수가 증가되어 트랜잭션의 지연 및 취소가 발생한다. 그러나 제안 기법은 중요 트랜잭션의 경우 서버에서 수행하는 기법을 사용하므로 기존의 기법에 비해 20~30% 정도의 안정적인 처리기능을 지원하고 있다.

5. 결 론

본 논문에서는 사용자가 자유롭게 이동하는 이동 컴퓨팅 환경에서 발생되는 트랜잭션이 시간적인 제약조건을 지니는 이동 실시간 트랜잭션을 지원하기 위한 트랜잭션의 처리에 관한 연구를 수행하였다. 이동 컴퓨팅 환경에서 이동 호스트의 이동성에 의한 접속 단절과 핸드오버 등에 의해 트랜잭션의 실행에 대한 예측성이 저하됨으로 인해 실시간 특성을 지원하는 것이 매우 어렵다. 또한, 기존의 시간적인 제약조건을 갖는 실시간 트랜잭션 처리를 위한 스케줄링 기법, 동시성 제어 기법 등이 고정망을 기반으로 개발된 것으로 이를 무선통신을 사용하는 이동 컴퓨팅 환경에 적용하는 것은 부적합하다. 따라서 본 논문에서는 이동 컴퓨팅 환경에서 발생하는 이동 실시간 트랜잭션을 위한 동시성 제어 기법을 제안하였다.

제안된 동시성제어 기법은 트랜잭션의 완료율을 높이기 위해 낙관적인 기법을 기반으로 한다. 제안기법은 완료된 트랜잭션의 재 시작을 줄여 실시간 트랜잭션의 적중률을 향상시키기 위해 완료 트랜잭션을 우선으로 처리하여 이동 컴퓨팅 환경에서 동시성 제어의 문제점인 이동 호스트와 이동 지원국간에 접속 단절 현상이나 핸드오버가 발생하여 이동 호스트가 기존의 셀에서 다른 셀로 이동하는 경우 충돌된 트랜잭션들의 지연이나 충돌한 트랜잭션들의 연속적인 철회를 해결하였다. 제안된 기법은 성능 평가를 통해서 기존의 기법에 비해 중요 트랜잭션을 20~30% 더 처리됨을 보였다.

참 고 문 헌

[1] Jin Jing, "Client-Server Computing in Mobile Environments," ACM Computing Surveys, Vol.31, No.2, pp.117-157, 1999.

[2] M.H.Dunham and A. Helal, "Mobile computing and databases : Anything new?," ACM SIGMOD Record, Vol.24, No.4, pp.5-9, 1995.

[3] Kam-Yiu Lam, Tei-Wei Kuo, Wai-Hung Tsang and Gray C.K. Law, "Concurrency Control in Mobile Distributed Real-Time Database Systems," Information Systems, Vol.25, No.4, pp.261-322, 2000.

[4] R.A. Drickze and Le Gruenwald, "A Pre-serialization Transaction Management Technique for Mobile Multidata-

bases," Mobile Networks and Applications, Vol.5 Issue 4, pp.311-321, 2000.
[5] O. Ulusoy, "Real-Time data management for mobile computing," International Workshop on Issues and Applications of Database Technology, pp.223-240, 1998.
[6] Joe Chun-Hung Yuen, Edward Chan, Kam-Yiu Lam, H. W. Leung, "Cache Invalidation Scheme for Mobile Computing Systems with Real-time Data," ACM SIGMOD Record, Vol.29 Issue 4, pp.34-39, 2000.
[7] A. Prasad Sistla, Ouri Wolfson, Sam Chamberlain, and Son Dao, "Modeling and Querying Moving Objects," Proceedings of the IEEE International Conference on Data Engineering, pp.422-432, 1997.
[8] R. Abbott and H. Garcia-Molina, "Scheduling I/O Requests with Deadlines : a Performance Evaluation," Proceedings of 11th Real-Time Systems Symposium, pp.113-124, 1990.
[9] D. Agrawal, A. E. Abbadi, R. Jeffers and L. Lin, "Ordered Shared Locks for Real-Time Databases," VLDB Journal, Vol.4, No.1, pp.87-126, 1995.
[10] J. R. Haritsa, M. J. Carey and M. Livny, "Value-Based Scheduling in Real-Time Database Sysytems," VLDB Journal, Vol.2, No.2, pp.117-152, 1993.
[11] J. Huang and John A. Stankovic, "Experimentation of Real-Time Optimistic Concurrency Control Schemes," Proceedings of 17th International Conference on VLDB, pp.35-46, 1991.
[12] Kam-Yiu Lam, Guo Hui Li, Tei-Wei Kuo, "A Multi-version Data Model for Executing Real-time Transactions in a Mobile Environment," Proceedings of the 2nd ACM international workshop on Data engineering for wireless and mobile access, pp.90-97, 2001.
[13] J.Jing, O.Bukhres and A.Elmagarmid, "Distributed Lock Management for Mobile Transactions," Proceedings of the 15th IEEE International Conference on Distributed Computing Systems, pp.118-125, 1995.
[14] P.A. Bernstein, V. Hadzilacos, and N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987.

김 경 배

e-mail : gbkim@seowon.ac.kr
1992년 인하대학교 전자계산공학과 (공학사)
1994년 인하대학교 대학원 전자계산공학과 (공학석사)
2000년 인하대학교 대학원 전자계산공학과 (공학박사)
2000년~2004년 한국전자통신연구원 선임연구원
2004년~현재 서원대학교 컴퓨터교육과 전임강사
관심분야 : 이동실시간데이터베이스, 스토리지시스템, GIS, VOD등

조 숙 경

e-mail : skyoe@dreamwiz.com
1990년 인하대학교 전자계산학과(이학사)
1994년 인하대학교 대학원 전자계산 공학과 (공학석사)
2002년 인하대학교 대학원 전자계산공학과 (공학박사)
2003년~현재 인천대학교 컴퓨터공학과 강의전담교수
관심분야 : 데이타베이스, 실시간 데이타베이스 시스템, 데이터베이스 시스템의 보안등

배 해 영

e-mail : hybae@inha.ac.kr
1974년 인하대학교 응용물리학과(공학사)
1978년 연세대학교 대학원 전자계산학과 (공학석사)
1989년 숭실대학교 대학원 전자계산학과 (공학박사)
1985년 Univ. of Houston 객원교수
1992년~1994년 인하대학교 전자계산소 소장
1982년~현재 인하대학교 컴퓨터공학부 교수
1999년~현재 지능형GIS센터 센터장
2000년~현재 중국 중경우전대학교 대학원 명예교수
관심분야 : 공간데이터베이스, 지리정보시스템, 분산데이터베이스, 데이터베이스클러스터,멀티미디어데이터베이스등