# 이동 트랜잭션을 위한 새로운 낙관적 동시성 제어 방법

김 치 연† · 배 석 찬††


## 요 약

많은 수의 클라이언트들에게 데이터를 방송하는 환경에서 가장 큰 제약은 서버와 통신하는데 이용가능한 대역폭이 낮다는 점이다. 이동 컴퓨팅 환경에서 많은 응용들이 개발되고 있으나, 무선망의 낮은 대역폭으로 인하여 전통적인 동시성 제어 방법을 그대로 적용하기 어렵다. 이 논문에서는 이동 트랜잭션을 위한 새로운 낙관적 동시성 제어 방법을 제안한다. 제안하는 방법에서 관독 전용 트랜잭션은 서버와 부가적인 메시지 교환없이 지역적으로 완료가 가능하며, 이동 갱신 트랜잭션은 서버로 보내져 전역적 검증을 수행한다. 또한 충돌 정보를 이용한 낙관적 방법에서 발생하는 트랜잭션의 불필요한 철회를 줄이고, 데이터 테이블에 유지된 정보를 이용하여 직렬가능하지 않은 수행을 찾아 직렬성 위배를 해결할 수 있다.


# A New Optimistic Concurrency Control Method for Mobile Transactions

Chi-yeon Kim† · Seok-Chan Bae††


## ABSTRACT

A crucial limitation in environments where data is broadcast to very large client populations is the low bandwidth available for clients to communicate with servers. Many advanced applications are developed in mobile computing environments, but conventional concurrency controls are not suitable because of the low bandwidth of wireless network. In this paper, we propose a new optimistic concurrency control protocol for mobile transactions. In this protocol, mobile read-only transactions can be completed locally at the clients without additional communication, only mobile update transactions are sent to the server for global validation. Our protocol reduces unnecessary aborts occurred in the previous study using only conflict information. In addition to, our algorithm can detect and resolve non-serializable execution using by data table maintained in a server.


**키워드 : 이동 컴퓨팅(Mobile Computing), 동시성 제어(Concurrency Control), 직렬성(Serializability), 이동 트랜잭션(Mobile Transactions), 무효화 레포트(Invalidation Report)**

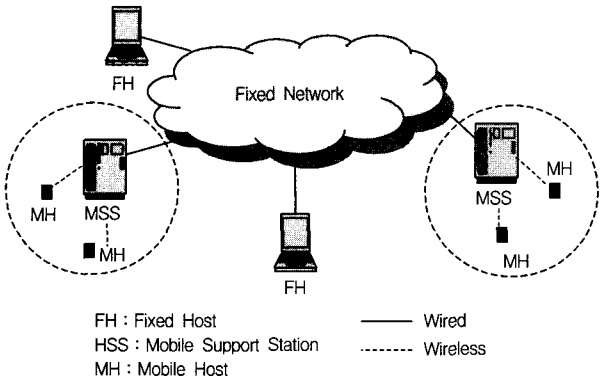
## 1. 서 론

방송은 넓은 지역의 많은 클라이언트에게 효율적으로 데이터를 전달하는 방법이다. 전통적인 클라이언트-서버 환경에서 데이터는 클라이언트의 명확한 요구에 의해 전달되었다. 하지만 이동 환경과 같이 소수의 서버와 다수의 클라이언트가 존재하는 환경에서 각각의 클라이언트가 서버에 요구를 전달하는 것은 시스템 전체에 부담이 된다. 따라서 서버는 주기적으로 데이터를 일방적으로 방송함으로써 다수의 클라이언트에게 전달한다[1, 7, 8, 11].

이동가능한 컴퓨터나 PDA 같은 장비의 개발, 그리고 무엇보다 무선망의 발전에 힘입어 발전하고 있는 이동 컴퓨팅 환경에서도 방송을 사용한다. 이동 컴퓨팅 환경은 (그림 1)과 같이 유선망에 연결된 고정 호스트(FH)와 이동 가능한 이동 호스트(MH), 그리고 고정 호스트들 중 무선 인터페이스를 통해 이동 호스트와 통신 가능한 서버(이동 기지국)로 구성된다[5]. 이동 컴퓨팅 환경의 특징이라고 하면 무엇보다도 시간과 장소에 구애받지 않고 작업을 수행하는 이동 호스트의 존재라 할 수 있다[2]. 하지만 이동 호스트는 고정 호스트에 비해 메모리 용량이나 CPU 속도, 배터리 용량 등에 한계를 갖는다[10, 12]. 이런 이동 호스트의 자체적인 한계 외에도 이동 컴퓨팅 환경은 무선망을 사용하기

† 정 회 원 : 목포해양대학교 해양전자통신공학부 전임강사
†† 종신회원 : 군산대학교 컴퓨터정보과학과 부교수
논문접수 : 2002년 11월 18일, 심사완료 : 2003년 1월 29일

때문에 신뢰성이나 대역폭에서도 한계를 갖는다.




(그림 1) 이동 컴퓨팅 환경의 구조

이동 환경에서 트랜잭션 관리 방법은 위에서 언급한 한계들을 고려하고 설계되어야 하는데, 이동 호스트의 가장 큰 특징인 이동성과 단절, 장기간 수행되는 트랜잭션, 그리고 서버와 클라이언트 사이의 비대칭적 대역폭은 시스템의 특징과 성능에 영향을 미칠 수 있는 중요한 요소들이다[3, 4, 15]. 따라서 이 논문에서는 이러한 특징과 한계들을 고려한 동시성 제어 방법을 제안하고자 한다.

이동 환경에서 제안된 동시성 제어 방법 중 [5, 9, 11, 13]에서는 판독 전용 트랜잭션의 스케줄링 방법을, [3, 6, 17]에서는 갱신 연산의 수행 방법을 다루고 있다. 기존 연구에서 판독 전용 트랜잭션의 스케줄링 방법을 다룬 이유는 초기의 시스템에서는 이동 호스트의 성능이 그리 높지 않았고, 이동 호스트에서 수행되는 대부분의 응용이 질의가 대부분이어서 판독 전용 트랜잭션을 위한 특별한 알고리즘이 필요하였기 때문이다. 그러나 이동 호스트의 성능이 많이 향상되고 수행되는 응용도 복잡해지고 있어서 이동 호스트에서 수행되는 갱신 트랜잭션을 위한 알고리즘이 필요하게 되었다. 방송 환경의 연구에서는 잠금이나 타임스탬프 프로토콜을 변형하여 적용하고 있다. 현재 낙관적 방법을 사용한 연구들이 많이 제안되고 있는데, 이동 호스트에서 수행되는 갱신의 비율이 높지 않다는 가정에 기반을 두고 있다. 이동 환경에서의 낙관적 방법은 제한된 대역폭을 효율적으로 사용하는 방법이라 생각된다. 그러나 낙관적 방법을 사용한 기존의 연구들에서는 첫째, 트랜잭션의 불필요한 철회가 발생할 수 있고 둘째, 트랜잭션의 수행 시점과 검증 시점의 차이로 인한 직렬성 위배의 가능성이 존재한다.

따라서 이 논문에서는 이러한 문제를 해결할 수 있는 새로운 낙관적 동시성 제어 방법을 제안하고자 한다. 이 논문의 구성은 다음과 같다. 2장에서는 기존 이동 환경에서 낙관적 동시성 제어 방법의 문제점들을 서술하고, 3장에서는 제안하는 동시성 제어 프로토콜을 기술하고, 제안하는 알고리즘이 트랜잭션의 직렬가능한 수행을 보장함을 증명한다. 4장에서는 결론 및 향후 연구방향을 기술한다.

## 2. 관련 연구

[6]에서는 갱신 연산을 포함하는 이동 트랜잭션에 대하여 낙관적 방법을 사용한 동시성 제어 프로토콜을 제안하였다. 충돌 정보에 기반을 둔 2PL Certifier를 사용하여 수행되는 트랜잭션에 대한 판독 집합과 기록 집합을 유지함으로써 동시성을 제어하였다. 모든 데이터를 관리하기 위한 하나의 서버가 중앙에 존재할 때 방송을 이용하여 적절한 제어 정보를 방송함으로써 클라이언트가 지역 Certifier로 작동하게 하였다. 클라이언트는 캐쉬를 사용하며, 수행도중이라도 방송 메시지에 충돌 관계에 있는 트랜잭션의 완료 메시지를 받으면 바로 철회되도록 하여 쓸모없는 수행을 계속하지 않도록 하였다. 하지만 이 연구의 문제점은 단순한 접근 집합만의 검사로 트랜잭션의 철회와 완료를 결정함으로써 직렬성이 유지되는 수행도 철회될 수 있다는 점이다. 이 문제를 (예 1)에서 다루고 있다. 또한 하나밖에 존재하지 않는 서버의 부담이 매우 크다는 문제점이 있다.

**(예 1) [6]에서 발생하는 불필요한 철회**

두 개의 트랜잭션 $T_1$, $T_2$가 다음과 같이 수행된다고 가정하자.


[6]에서 제안한 알고리즘과 시나리오에 의하면, $T_1$은 정상적으로 완료되며, $T_2$가 완료할 시점에서 $T_2$는 $T_1$과의 데이터 항목 x에 의해 발생하는 충돌 때문에 철회된다. 하지만 트랜잭션의 정확성 검증을 위해 가장 일반적으로 사용하는 직렬성의 관점에서 보면, 데이터 항목 x에 대한 직렬화 순서가 $T_2 \rightarrow T_1$, y에 대한 직렬화 순서는 $T_2$이어서 전체적으로 사이클이 발생하지 않으므로, 이 수행은 전혀 문제가 없으며 $T_1$뿐 아니라 $T_2$도 정상적으로 종료될 수 있다. 이러한 결과는 충돌 정보에 기반한 certifier를 사용하기 때문인데, 단순하게 충돌 집합에 교집합이 있는 것만으로 트랜잭션을 철회하는 것은 불필요한 철회라 할 수 있다. □

트랜잭션의 동시성 제어 방법이 정확한 수행을 보장함을 증명하기 위해 일반적으로 사용된 기준이 직렬성이다[14].

하지만 이동 환경은 무엇보다 대역폭이 충분하지 않고, 이로 인하여 서버와 클라이언트 사이에 잦은 메시지 교환이 어렵다. 따라서 [9]에서는 전통적인 직렬성을 대신하여 갱신 일관성이라 불리는 완화된 기준을 제안하였다. 기존의 직렬성이 모든 트랜잭션 상에서 직렬가능한 수행을 강요하는 반면, 갱신 일관성에서는 ① 모든 갱신 트랜잭션은 직렬가능하고, ② 판독 전용 트랜잭션은 자신이 직/간접으로 read-from 관계에 있는 트랜잭션과만 직렬가능하면 정확성이 유지된다는 기준이다. 하지만 이 방법은 많은 계산을 필요로 하고, 큰 볼륨의 제어 정보를 방송하기 위해 대역폭을 많이 소모한다는 문제점이 있다.

이동 환경에서 전통적인 직렬성의 강한 제약을 완화하려는 연구에도 불구하고 [13]에서는 직렬성의 중요성을 언급하였다. 방송을 사용하는 주식 매매와 같은 응용에서는 엄격한 정확성 기준이 필요하다. 방송 환경에서 타임스탬프 간격(Timestamp interval)을 이용한 판독 전용 트랜잭션에 대한 동시성 제어 프로토콜 BCC-TI를 제안하였다. 갱신 트랜잭션은 기존의 동시성 제어 방법으로 수행되고 완료 시점에 타임스탬프가 결정된다. 판독 전용 트랜잭션은 접근하는 데이터의 갱신 타임스탬프와 자신이 가질 수 있는 타임스탬프 간격을 비교하여 직렬화 순서가 유지되는지를 결정하였다. 즉, 판독 전용 트랜잭션이 접근하는 트랜잭션의 타임스탬프가 데이터의 마지막 갱신 타임스탬프보다 크거나 같아야만 정상적으로 수행된다. 이 연구에서는 갱신 트랜잭션의 타임스탬프가 완료 시점에 결정되기 때문에 수행 순서와 직렬화 순서가 달라지는 문제가 발생할 수 있다.

## 3. 제안하는 방법

이 절에서는 이동 환경에서 갱신 트랜잭션을 포함한 이동 트랜잭션을 위한 동시성 제어 방법인 NOC-MT(a New Optimistic Concurrency control for Mobile Transactions) 프로토콜을 제안한다. 이를 위해 사용된 가정과 시스템 모델, 제안하는 알고리즘 순으로 기술한다.

### 3.1 가정 및 시스템 모델

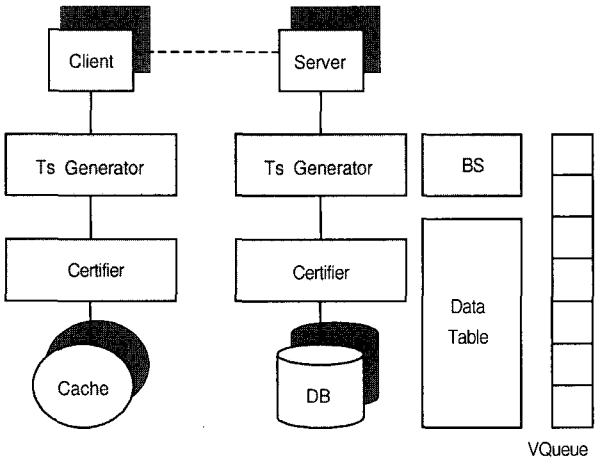
제안하는 방법에서 사용하는 가정은 다음과 같다.

- 이동 호스트에서는 한 번에 하나의 트랜잭션만 수행된다.
- 트랜잭션에 유일한 타임스탬프 할당을 위해 각 서버들과 이동 호스트의 클럭이 동기화되어 있다.

위의 두 번째 가정은 다수의 서버와 이동 호스트에서 제출되는 트랜잭션의 유일한 타임스탬프 부여를 위해 필요하다. 이동 환경에서 서버와 이동 호스트 사이의 구체적인 클럭 동기화 방법은 [19]에서 제안한 방법을 그대로 사용한다고 가정하며, 이 논문에서는 별도의 방법을 제안하지 않는다.

이동 호스트로 제출되는 트랜잭션을 이동 트랜잭션이라 하는데, 판독 연산만으로 구성된 트랜잭션을 MRT(Mobile Read Transaction)라 하고, 갱신 연산을 포함한 트랜잭션을 MUT(Mobile Update Transaction)라 정의한다. MRT와 MUT는 이동 호스트의 자발적인 단절로 장기간 수행되는 트랜잭션(Long-duration Transaction)일 수 있다. 서버에서 수행되는 트랜잭션은 이동 트랜잭션과 구별하기 위해 SRT (Server Read Transaction)와 SUT(Server Update Transaction)로 정의한다. 각 트랜잭션은 begin, read, write, commit, abort 연산의 집합으로 구성된다.

NOC-MT 프로토콜에서 사용하는 시스템 모델은 (그림 2)와 같다. 클라이언트에는 트랜잭션에 타임스탬프를 할당하는 타임스탬프 생성기와 지역 certifier, 그리고 캐쉬가 존재한다. 서버에는 방송을 담당하는 방송 스케줄러(BS)와 각 데이터를 접근하는 트랜잭션의 타임스탬프를 유지하는 데이터 테이블이 있다. 또한 검증을 위해 트랜잭션이 대기하는 검증 큐(VQueue)가 존재한다.




(그림 2) 시스템 모델

### 3.2 NOC-MT 알고리즘

이 절에서는 트랜잭션을 수행하기 위한 이동 호스트와 서버의 기능과 필요한 자료구조, 알고리즘을 기술한다.

#### 3.2.1 이동 호스트 프로토콜

이동 호스트의 기능을 요약하면 아래와 같다.

- 자주 접근하는 데이터의 최신 버전을 캐쉬에 저장한다.
- 서버에서 주기적으로 방송하는 메시지에 따라 캐쉬를

갱신한다.

- MRT와 MUT가 제출되며, 각 트랜잭션을 타임스탬프에 기반한 낙관적 방법으로 수행한다.

캐쉬는 이동 호스트가 자주 접근하는 데이터를 저장하여 서버에 요구하지 않고 접근하게 함으로써 무선 통신망의 낮은 대역폭을 완화하고 응답 시간을 단축시킬 수 있는 일종의 메모리이다. 캐쉬에 저장된 데이터는 서버에 저장된 것과 일치해야 하는데, 이를 위하여 서버는 정기적으로 최신 방송 구간에서 갱신된 데이터를 방송하고, 이동 호스트는 메시지에 따라 캐쉬를 갱신한다. 서버에서 주기적으로 방송하는 메시지를 무효화 레포트(IR : Invalidation Report)라 한다[16].

이동 호스트로 제출되는 트랜잭션은 도착한 시간에 따라 타임스탬프를 할당받는데, 동기화된 클럭을 사용하여 각 트랜잭션마다 유일한 타임스탬프를 부여받는다[18]. MRT는 캐쉬 내에 있는 데이터를 접근하여 연산을 수행한다. 만약 접근하려는 데이터가 캐쉬 내에 존재하지 않으면 데이터에 대한 요구 메시지를 서버에 보낸다. 트랜잭션이 판독할 수 있는 데이터는 자신의 타임스탬프보다 작은 갱신 타임스탬프를 갖는것 중 가장 큰 데이터이다. MRT의 마지막 연산이 수행되면 지역적인 검증을 수행하고 직렬성이 유지되는지 검사한다. MRT의 직렬성이 위배되는 경우는 MRT가 접근하는 각 데이터 x에 대하여 x를 판독하기 전에 MRT보다 큰 타임스탬프의 트랜잭션이 x를 먼저 갱신한 경우이다. 이 경우는 MRT가 수행을 마친 후 다음 방송을 듣고 판단할 수 있다. 이를 위해 서버에서는 갱신 트랜잭션이 접근한 데이터에 대한 최종 갱신 타임스탬프와 수행 시각을 방송해야 한다. 각 연산에 대한 수행 시각을 방송하는 이유는 (예 2)에서 언급한 문제를 방지하기 위해서이다. MRT는 수행이 끝나고 방송을 청취한 뒤 방송 메시지에 포함된 데이터와 자신이 판독한 데이터에 교집합이 발생하는지 검사하고 교집합이 공집합이라면 완료가 가능하다. 또한 교집합이 발생하더라도 타임스탬프와 수행 시각을 비교하여 타임스탬프 순서가 유지된다면 완료할 수 있다.

MUT는 기록 연산만으로 구성된 트랜잭션이 아니라 기록 연산을 포함하는 트랜잭션이다. 이 논문에서는 이동 트랜잭션이 갱신 연산을 포함할 비율이 높지 않다는 가정하에, 낙관적 방법을 채용하고 있다. MUT의 판독 연산은 MRT의 수행 방법과 동일하게 수행하고, 기록 연산은 지역 사본을 만들어 기록한다. 이동 호스트에서 수행된 갱신 연산은 전역적 검증을 위해 서버에 보내져야 한다. 이 때, (예 2)에서 언급한 문제를 해결하기 위해서는 각 연산의 수행 시간도 서버에 보내져야 한다. 하나의 MUT가 수행된 후 서버에 보내져야 할 정보는 (그림 3)과 같다.

이동 호스트에서 수행되는 프로토콜은 (그림 4)와 같다. 여기서 SR() 함수는 MRT가 검증을 위해 방송 메시지를 청취했을 때 IR과 판독 집합에 교집합이 발생한 경우, 직렬성이 유지되는지를 검사하는 함수이다. 직렬성이 유지되면 TRUE, 위배되면 FALSE를 반환한다.

```
/* Mobile Host Algorithm */
 이동 트랜잭션 T에 유일한 타임스탬프 할당 // T는 MRT이거나
 MUT
for (T의 각 연산 op(x)에 대하여) {
        if (op == 'r') {
           if (데이터 항목 x가 캐쉬에 존재하면) r(x) 수행 ;
           else 서버에 x 요청 ;
        if (op == 'w) 지역 사본 x' 생성하고, too-late
           transaction이 아니면 수행 ;
    if (T의 마지막 연산의 수행이 끝나면) {
        if (T가 MRT이면) {
           다음 IR 청취 ;
           common_data = (data_set(IR) ∩ read_set(T))
        if (common_data이 공집합이면) T 완료 ;
        else {
           for (IR과 판독집합에 공통인 데이터를 갱신한 모든
                트랜잭션 Tj에 대하여)
                  if (SR(IR, T)) commit T ;
                  else T 철회 ;
        }
    if (T가 MUT이면 ) {
        TID, ts와 임의 w(x)에 대하여 (x, new_value, exe_time)
        을 서버에 보낸다.
        전역 결정을 기다린다 ;
    }
}
서버로부터 방송 메시지를 받으면 {
    수행중인 이동 트랜잭션 중 IR과 직렬성이 위배되는 트랜잭션
    이 있으면 철회 ;
    IR에 따른 캐쉬 갱신 ;
    switch (전역 결정이 포함된 경우) {
           case "commit" : 갱신된 값 저장 ; break ;
           case "abort" : 지역 사본 삭제 ; break ;
    }
}
```

(그림 4) NOC-MT 이동 호스트 프로토콜

| T_id | ts(T) | x | 30 | exe_time | ... | s | 24 | exe_time |
|---|---|---|---|---|---|---|---|---|
| 식별자 | 타임스탬프 | 갱신데이터 | 갱신값 | 수행시간 | | 갱신데이터 | 갱신값 | 수행시간 |

(그림 3) MUT의 검증을 위해 서버에 보내지는 정보

3.2.2 서버 프로토콜

서버의 기능을 요약하면 아래와 같다.

- 장기간 수행되는 이동 트랜잭션을 위해 다중 버전을 유지한다.
- 완료된 갱신 트랜잭션에 의해 갱신한 데이터를 주기적으로 이동 호스트에게 방송한다.
- 이동 호스트로부터 검증이 요청된 이동 갱신 트랜잭션의 검증을 수행한다.
- 트랜잭션의 직렬가능한 수행을 보장한다.

이동 호스트에서 수행되는 트랜잭션은 잦은 이동과 단절로 인하여 수행 시간이 길다는 특징이 있다. 이러한 환경에서 판독 전용 트랜잭션의 완료율 향상을 위해 서버에 다중 버전을 유지한다. 새로운 버전은 갱신이 완료될 때마다 생성된다. 이동 호스트에서 수행중인 어떤 이동 트랜잭션의 타임스탬프보다도 작은 타임스탬프를 갖는 버전은 삭제될 수 있다.

이동 환경에서는 서버와 이동 호스트가 항상 연결을 유지할 수는 없기 때문에, 서버와 이동 호스트 사이의 데이터 일치를 위한 방법이 필요한데 가장 일반적인 방법이 방송을 이용하는 것이다. 서버에서는 갱신을 포함한 트랜잭션이 완료되면, 일정 주기로 갱신을 모아 이동 호스트들에게 방송한다.

서버의 가장 큰 기능 중 하나는 데이터에 대한 갱신이 직렬가능한 순서로 수행되도록 제어하는 것이다. 이를 위하여 서버에서 수행되는 트랜잭션과 이동 호스트들로부터 요청된 갱신 트랜잭션이 직렬가능한 순서로 수행되도록 제어해야 한다. 이를 위해 데이터 테이블을 이용한다. 데이터 테이블에는 데이터베이스에 존재하는 데이터 항목의 각 버전에 대하여 데이터 항목을 갱신한 연산의 식별자와 타임스탬프, 그리고 수행 시각을 유지한다. 갱신 트랜잭션이 검증 단계에 들어가면 데이터 테이블을 접근하여 타임스탬프 순서와 수행 순서가 직렬가능한 순서로 유지되는지를 검사하여 완료 여부를 결정한다. 이 논문에서는 다중 버전이 유지되기 때문에 1SR을 정확성 기준으로 사용한다. 구체적인 서버 알고리즘은 (그림 5)와 같다.

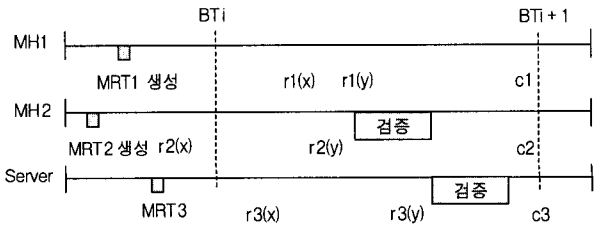
```
/* Server Algorithm */
서버 트랜잭션 T에 유일한 타임스탬프 할당 ;
 for (T의 각 연산 op(x)에 대하여) {
    if (op가 갱신 연산이면) x에 대한 새버전 x'생성 ;
    r(x)나 w(x) 수행 ;
 T의 마지막 연산이 끝나거나 검증을 요구한 이동 갱신 트랜잭션이
 있으면 검증 단계 시작 ;
 검증 단계에서 {
    트랜잭션이 접근한 모든 데이터에 대한 데이터 테이블 검사 ;
    if (수행 시각과 타임스탬프 순서에 역전이 발생하지 않으면)
       commit T ;
       else abort T ;
    검증 단계를 끝낸 T에 대하여 {
          if (완료 결정이면) {
             갱신한 값을 데이터베이스에 인스톨 ;
             T가 MUT이면 commit_set에 T 추가 ;
          }
          else { // 철회 결정이면
               생성된 사본 삭제 ;
               T가 MUT이면 abort_set에 T 추가 ;
          }
}
방송 시점에서 {
      IR 구축 ; // IR에는 갱신된 데이터와 각 연산의 수행 시각이
      포함된다.
      IR, commit_set, abort_set을 모든 이동 호스트에게 방송 ;
}
if (임의의 트랜잭션 T가 완료되었으면)
      더 이상 접근되지 않은 버전을 찾아 삭제 ;
}
```

(그림 5) NOC-MT 서버 알고리즘

다음의 (예 3)은 NOC-MT 알고리즘의 수행 예이다.

**(예 3) NOC-MT 알고리즘의 적용 예**

두 개의 이동 호스트 $MH_1$, $MH_2$와 서버에서 각각 $MRT_1$, $MUT_2$, $SUT_3$이 아래와 같이 수행된다고 가정하자. $MRT_1$은 $r_1(x)$ $r_1(y)$로 구성되며 $ts(MRT_1) = 13$이고, $MUT_2$는 $r_2(x)$ $w_2(y)$로 구성되며 $ts(MUT_2) = 11$이다. $SUT_3$은 $w_3(x)$ $w_3(y)$로 구성되며, $ts(SUT_3) = 14$라 가정하자. $MRT_1$은 연산의 수행이 끝난 후 지역적 검증을 위해 다음 방송을 기다린다. $BT_{i+1}$에서 IR에 $SUT_3$에 의해 갱신된 데이터 y가 포함되나 직렬성이 유지되므로 $MRT_1$은 완료할 수 있다. $MUT_2$도 마지막 수행을 끝낸 후 검증에 들어가는데, 직렬성이 유지되므로 완료할 수 있다. $MUT_2$는 검증이 끝나자마자 서버에서 완료된 트랜잭션으로 인식되어, 갱신된 값이 데이터베이스에 인스톨되며, $BT_{i+1}$ 시점에서 이동 호스트에게 commit_set으로 방송된다. $SUT_3$는 $MUT_2$와는 직렬성이 유지되며, $MRT_1$과 수행 순서에 사이클이 발생하는 것처럼 보이나 $r_1(x)$에 의해 접근하는 x값은 $SUT_3$에 의해 갱신된 값이 아니라 캐쉬에 저장된 그 이전 버전이다. 따라서 $SUT_3$의 수행도 직렬성을 보장하며 성공적으로 완료될 수 있다. □



다음은 NOC-MT 알고리즘의 정확성 증명을 위하여 직렬가능한 히스토리를 생성함을 보인다.

**[정리] NOC-MT 알고리즘은 트랜잭션의 직렬가능한 수행을 보장한다.**

(증명) NOC-MT 알고리즘에서 직렬성이 위배될 수 있는 경우는 동일 방송 구간에서 수행되는 ① MRT와 MUT, ② MRT와 SUT, 그리고 ③ MUT와 SUT 사이의 직렬화 순서에 사이클이 발생하는 경우이다. 증명에 앞서 다중 버전 환경에서 충돌 연산을 정의한다. 일반적으로 충돌 연산은 동일한 데이터 항목을 접근하는 두 연산 중 적어도 하나가 갱신 연산인 경우를 일컫는다. 하지만 다중 버전에서는 갱신 연산이 실행될 때마다 새로운 버전이 생성되므로, 데이터 항목 x에 대하여, $w_i[x_i] \rightarrow r_j[x_i]$의 경우만을 충돌로 정의한다[14].

***경우 1 : $MRT_i$와 $MUT_j$ 사이의 직렬성 위배***

MRT와 MUT 사이에서 직렬성이 위배될 수 있는 경우는 $MRT_i \rightarrow MUT_j \rightarrow MRT_i$이거나 $MUT_j \rightarrow MRT_i \rightarrow MUT_j$와 같은 직렬화 순서가 생성되는 경우이다. MRT는 모두 판독 연산만으로 구성되고, 판독 연산들끼리는 충돌이 발생하지 않으므로 이러한 직렬한 순서는 MRT의 판독 연산과 MUT의 갱신 연산과만 관련이 있다. 전자의 경우는 $r_i[x] \rightarrow w_j[x]$, $w_j[y] \rightarrow r_i[y]$의 경우인데, 충돌의 정의에서처럼 이러한 순서는 사이클을 발생시키지 않는다. 뿐만 아니라 MRT의 판독 연산에 의해 접근되는 데이터는 캐쉬에 있는 값이기 때문에 갱신 트랜잭션에 의해 쓰여진 값이 아니라 이전에 완료되어 데이터베이스에 반영된 값이다. 후자의 경우는, $w_j[x] \rightarrow r_i[x]$, $r_i[y] \rightarrow w_j[y]$와 같은 경우인데 마찬가지로 사이클을 발생시키지 않는다.

***경우 2 : MRT와 SUT 사이의 직렬성 위배***

경우 1의 MUT를 SUT로 대치하면 경우 1과 동일한 상황이 되므로 사이클이 발생되지 않는다.

***경우 3 : MUT와 SUT 사이의 직렬성 위배***

NOC-MT 알고리즘에 의하여 MUT와 SUT는 완료되기 전 전역적 검증을 실시한다. 연산들이 접근한 데이터에 대한 데이터 테이블을 검사하여 해당 수행이 타임스탬프 규칙을 만족하지 못할 경우는 트랜잭션이 철회되므로 직렬화 순서에 사이클이 발생하지 않는다.

위 세 가지 경우에서 모두 직렬화 순서에 사이클이 발생하지 않으므로 NOC-MT 알고리즘은 직렬가능한 히스토리만을 생성함을 보장한다. □

## 4. 결 론

이동 환경은 하드웨어와 무선 통신망 기술의 발달에 힘입어 빠른 속도로 발전하고 있다. 뿐만 아니라 이동 환경을 기반으로 한 진보된 응용들 또한 다양한 형태로 개발되고 있다. 하지만 무선 통신망의 대역폭은 쉽게 극복되기 어려운 한계 중의 하나이다.

이동 환경의 클라이언트들은 이동가능한 장비들을 사용하여 위치와 시간에 구애받지 않고 작업을 수행할 수 있으나 서버에 있는 데이터베이스를 접근해야 하는 경우, 서버와의 데이터 일치가 중요하다. 그리고 다수의 이동 클라이언트들이 존재하는 경우라면 클라이언트들 사이에서 수행되는 작업과 서버와의 관계가 효율적인 방법으로 제어되어야 한다. 왜냐하면 이동 클라이언트들끼리는 직접 연결을 갖지 않기 때문에, 충돌 연산들이 수행되는 경우 서버와 클라이언트 관계보다도 제어가 어렵기 때문이다.

이 논문에서는 이동 클라이언트의 작업 수행 능력을 제한하지 않기 위해 갱신 연산을 포함한 트랜잭션의 수행 방법에 대하여 기술하였다. 클라이언트가 노트북이나 PDA를 사용하여 영업을 하는 응용에서 이동 클라이언트에서의 갱신 연산 능력은 반드시 필요하다. 그래서 갱신 연산의 비율을 낮게 가정하여 새로운 낙관적 동시성 제어 프로토콜 NOC-MT 프로토콜을 제안하였다.

NOC-MT 프로토콜은 기존의 방법에 비하여 트랜잭션의 철회율을 감소시킬 수 있고, 수행 시점과 검증 시점이 뒤바뀔 수 있는 문제를 해결하였다. 단순히 충돌 정보에 기반한 낙관적 동시성 제어 방법에서는 직렬성이 유지되는 수행도 철회될 수 있고, 선수행 후검증인 낙관적 방법에서는 수행 시점과 검증 시점의 역전이 발생할 수 있는데, 이 논문에서는 직렬성이 위배되는 경우에만 트랜잭션을 철회하며, 연산의 수행 시점을 검증에 반영함으로써 역전 현상을 방지하였다. 그리고, 제안된 방법의 정확성 증명을 위하여 직렬가능한 히스토리만을 생성함을 보였다.

제안한 방법과 [6]의 방법의 성능을 필요한 대역폭과 트랜잭션의 완료율의 측면에서 비교하면 다음과 같다. 알고리즘을 위해 필요한 대역폭의 경우, 두 가지 방법 모두 낙관적 방법을 사용하여 갱신 연산에 대해서만 서버에서 검증을 수행하므로 검증에 필요한 정보를 서버로 전송하기 위한 대역폭이 소모된다. 각 갱신 연산마다 트랜잭션 식별자 및 타임스탬프, 갱신된 데이터 값을 전송하기 위한 대역폭이 두 방법 모두에서 공통적으로 소모되며, NOC-MT 방법은 여기에 추가하여 각 갱신 연산이 수행된 시각을 전송하기 위한 대역폭이 추가적으로 필요하다. NOC-MT 방

법에서 추가적으로 필요한 대역폭은 갱신 연산의 비율이 높아질수록 많아지나, 갱신 연산의 출현 비율이 높지 않은 환경을 가정하였으므로 일정 비율 이상 높아지지 않을 것으로 생각된다. 또한, 각 갱신 연산의 수행 시각을 서버에 전송함으로써 직렬가능하지 않은 수행을 피할 수 있기 때문에 이는 필요한 오버헤드라 생각된다. 트랜잭션의 완료율 측면에서는 NOC-MT 방법이 [6]의 방법보다 더 효율적이다. [6]에서는 무효화 메시지와 이동 트랜잭션이 접근한 데이터 집합에 교집합이 있으면 무조건 철회하지만 NOC-MT 방법에서는 직렬성이 유지되는 경우를 검사하여 위배되는 경우에만 철회하기 때문이다. 여기에 추가하여 NOC-MT 방법에서는 서버에 데이터 테이블을 유지하기 위한 공간이 필요하다. 하지만 트랜잭션의 갱신율이 높지 않고, 서버는 이동 호스트에 비해 메모리 공간을 포함한 자원이 풍부하기 때문에 서버에 데이터 테이블을 유지하는 것이 큰 부담이 되지는 않을 것으로 생각된다. 제안한 방법의 모의 실험을 통한 정량적 성능 분석은 다음 연구에서 다루고자 한다.

앞으로 모의 실험을 통한 제안한 방법의 성능 평가와 중첩 트랜잭션의 개념을 도입하여 이동 트랜잭션의 각 연산을 독립적인 트랜잭션으로 처리하는 방법, 이동 에이전트를 사용한 서버와 이동 클라이언트 사이의 정보 교환 방법에 대하여 연구할 예정이다.

## 참 고 문 헌


[1] E. Pictoura, P. K. Chrysanthis, "Exploiting Versions for Handling Updates in Broadcast Disk," Proceedings of the 15th VLDB Conference, Edinburgh, Scotland, 1999.

[2] M. H. Dunham, V. Kumar, "Impact of Mobility on Transaction Management," Proceedings of the International Workshop on Data Engineering for Wireless and Mobile Access, MobiDE '99, Seattle, WA, USA, pp.14-21, Auguest, 1999.

[3] Q. Lu, M. Satyanarayanan, "Improving Data Consistency in Mobile Computing Using Isolation-Only Transaction," Proceedings of the Fifth IEEE HotOS Topics Workshop, Orcs Island, May, 1995.

[4] J. Jing, A. Helal, A. Elmagarmid, "Client-Server Computing in Mobile Environments," ACM Computing Surveys, Vol. 31, No.2, pp.117-157, June, 1999.

[5] A. Elmagarmid, J. Jing, O. Bukhres, "An Efficient and Reliable Reservation Algorithm for Mobile Transactions," Proceedings of the CIKM 95, Baltimore, MD, USA, pp.90-95, 1995.

[6] D. Barbara, "Certification Reports : Supporting Transactions in Wireless Systems," Proceedings of the 17th International Conference Distributed Computing Systems, Vienna, 1992.

[7] S. Acharya, M. Franklin, S. Zdonik, "Disseminating Updates on Broadcast Disks," In Proceedings of the 22nd VLDB Conference Mumbai(Bombay), India, 1996.

[8] S. Acharya, M. Franklin, S. Zdonik, "Balancing Push and Pull for Data Broadcast," Proceedings of ACM SIGMOD International Conference Management of Data, Phoenix, Arizona, pp.183-194, May, 1997.

[9] J. Shanmugasundaram, A. Nithrakashyap, R. Sivasankaran, K. Ramamritham, "Efficient Concurrency Control for Broadcast Environments," ACM SIGMOD International Conference on Management of Data, 1999.

[10] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing, PODC '96, Philadelphia, PA, USA, pp.1-7, May, 1996.

[11] E. Pitoura, P. K. Chrysanthis, "Scalable Processing of Read-Only Transactions in Broadcast Push," Technical Report 98-026, Depart. of Computer Science, University of Ioannina, 1998.

[12] D. Barbara, "Mobile Computing and Databases - A Survey," IEEE Transactions on Knowledge and Data Engineering, Vol.11, No.1, Jan./Feb., 1999.

[13] Lee V., Son S. H., Lam K., On the Performance of Transaction Processing in Broadcast Environments, Int Conf on Mobile Data Access (MDA'99), Hong Kong, Dec., 1999.

[14] P. A Bernstein, V. Hadzilacos, N. Goodman, Concurrency Control and Recovery in Database Systems, Addison Wesley, Reading, Massachusetts, 1987.

[15] G. H. Forman, J. Zahorjan, "The Challenges of Mobile Computing," IEEE Computer, pp.38-47, April, 1994.

[16] D. Barbara, T. Imielinski, "Sleepers and Workaholics : Caching Strategies in Mobile Computing," Proceedings of SIGMOD, May, 1994.

[17] E. Pictoura, B. Bhargava, "Data Consistency in Intermittently Connected Distributed Systems," Transactions on Knowledge and Data Engineering, Nov., 1999.

[18] 김치연, 황부현, "이동 트랜잭션의 완료율 향상을 위한 다중 버전 타임스탬프 순서화 스케줄링 기법", 정보처리학회논문지, 제6권 제5호, pp.1143-1152, 1999.

[19] 김치연, 황부현, "이동 컴퓨팅 환경에서 타임스탬프를 이용한 트랜잭션 스케줄링", 한국정보과학회논문지, 제26권 제1호, pp.40-51, 1999.

김 치 연

e-mail : gegujang2@mail.mmu.ac.kr

1992년 전남대학교 전산통계학과(이학사)

1994년 전남대학교 대학원 전산통계학과 (이학석사)

1999년 전남대학교 대학원 전산통계학과 (이학박사)

1996년~1997년 전남대학교 전산학과 조교

1999년~2002년 전남대학교 전산학과 시간강사

2002년~현재 목포해양대학교 해양전자통신공학부 전임강사

관심분야 : 이동 컴퓨팅, 트랜잭션 관리, 전자상거래

배 석 찬

e-mail : scbae@mail.kunsan.ac.kr

1983년 전남대학교 계산통계학과(이학사)

1988년 전남대학교 대학원 전산통계학과 (이학석사)

1995년 전남대학교 대학원 전산통계학과 (박사)

1983년~1985년 ROTC

1993년~1994년 서남대학교 전산통계학과 학과장

1995년~현재 군산대학교 컴퓨터정보과학과 부교수

관심분야 : 트랜잭션 관리, 데이터베이스 보안, 객체지향 시스템