이동 트랜잭션의 완료율 향상을 위한 다중버전 타임스탬프 순서화 스케줄링 기법

김 치 연† · 황 부 현†

요 약

이동 컴퓨팅 환경은 사용자의 위치에 상관없이 정보를 접근할 수 있도록 해 주는 새로운 패러다임이다. 이동 컴퓨팅 환경에서 사용되는 트랜잭션 스케줄링 방법은 데이터베이스 일관성을 위해 이동 호스트의 제한된 성능과 무선 통신망의 좁은 대역폭을 고려하여 설계되어야 한다. 지금까지 제안된 트랜잭션의 스케줄링 방법 중에서 록을 이용한 방법에서는 록 관리를 위한 부가적인 메시지 교환이 발생하고, 캐쉬를 사용할 때 즉각적인 캐쉬으로 트랜잭션의 직렬성이 위배되어 이동 트랜잭션의 철회가 많이 발생한다는 문제점이 있다. 따라서 본 논문에서는 이러한 문제점들을 해결하기 위하여 타임스탬프를 사용한 트랜잭션 관리 방법을 제안하고자 한다. 제안하는 방법은 이동 호스트가 캐쉬를 사용할 때 적용되는 방법으로 서버에 다중버전을 유지하고, 이동 호스트의 캐쉬에 두 개의 버전을 유지하여, 이동 트랜잭션이 여러 방송 구간에 걸쳐 수행되더라도 완료될 수 있는 방법이다. 따라서 이동 트랜잭션의 성능을 향상시킬 수 있으며, 타임스탬프를 이용하여 트랜잭션을 스케줄함으로써 부가적인 메시지 교환을 줄일 수 있다.

A Multiversion Timestamp Order Scheduling Method for Improving Commit Ratio of Mobile Transactions

Chi-yeon Kim† · Bu-hyun Hwang†

ABSTRACT

A Mobile computing environment is a new paradigm which provides users with the access to information irrespective of users' location. A transaction scheduling method for the mobile computing environment must be designed so that database consistency is maintained with considering the limited performance of mobile hosts and the narrow bandwidth of a wireless network. The scheduling method using a lock has some problems : the high message overhead between a server and a mobile host for maintaining a lock and the high abort ratio of the mobile transactions owing to violating the serializability when a mobile host uses a cache. So, in this paper, we propose an efficient transaction management method using timestamp to resolve these problems. The proposed method is used in the environment under which a mobile host uses a cache having two versions for each cached data item and a server maintains several versions for each data item. So, even though a mobile transaction is executed during several broadcasting interval, can be committed. As a result, the proposed method can improve the commit ratio of the mobile transactions by maintaining multiversion for each data item and does not require the additional message exchange to schedule transactions by using timestamp.

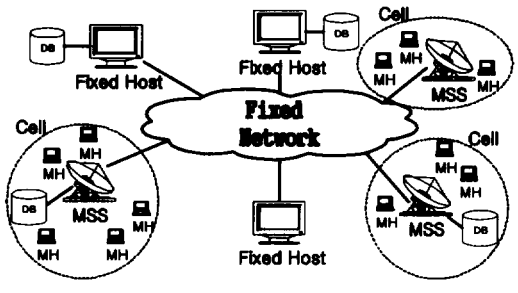
※ 본 논문은 정보통신부의 정보통신분야 우수학교 지원사업에 의하여 수행된 결과임.
† 정 회 원 : 전남대학교 대학원 전산학과
†† 정 회 원 : 전남대학교 전산학과 교수
논문접수 : 1998년 6월 16일, 심사완료 : 1999년 3월 9일

1. 서 론

최근에 급속히 확산되고 있는 이동 컴퓨팅 환경은 무선 통신 기술의 발전이 가져다 준 편리한 환경이다. 사용자들은 이동 가능한 컴퓨터나 PDA(Personal Digital Assistance)를 사용하여 자신의 위치나 장소에 상관없이 데이터베이스에 접근한다. 이동 사용자들은 전자 메일이나 주식 정보, 또는 여행 중에 필요한 정보들을 이동 컴퓨터를 사용하여 얻고자 한다[1].

이동 컴퓨팅 환경은 사용자들의 새로운 요구를 반영하기 위해 대두된 시스템이기 때문에 다음과 같은 점에서 기존의 시스템과 다르다. 첫 번째는 무선 통신망을 사용한다는 것인데[2], 무선 통신망은 유선 통신망에 비하여 신뢰성이 낮고 대역폭이 좁다는 단점을 갖고 있다[3]. 두 번째는 사용자의 이동성이다. 사용자의 이동성은 이동 컴퓨팅 환경의 가장 큰 특징이라 할 수 있는데, 이동 사용자들은 이동 중에도 서버와 연결을 유지하면서 작업을 수행하고자 한다. 하지만 여기에서의 한계는 이동 컴퓨터가 갖는 전력이다[4]. 따라서 이동 컴퓨터는 자신의 배터리 소모를 최대한 줄이기 위한 방법을 모색하게 되었는데, 이 중 하나가 연산하지 않을 때는 서버와의 연결을 끊고 단절하는 것이다[5,6,7].

이동 컴퓨팅 환경의 구성은 (그림 1)과 같다. 이동 컴퓨팅 환경은 고정 호스트(FH : Fixed Host)와 이동 호스트(MH : Mobile Host)로 구성되어 있다. 고정 호스트 중 이동 호스트와 통신할 수 있는 무선 인터페이스를 갖춘 호스트를 서버 또는 이동 기지국(MSS : Mobile Support Station)이라고 하는데[8], 이동 호스트(이동 컴퓨터)들이 트랜잭션을 처리하는데 필요한 데이터를 제공하거나 이동 호스트가 다른 호스트들과 통신할 수 있는 통로를 제공한다. 또한 서버는 유선 통신망에 연결되어 있는 고정 호스트들이 제출한 작업들을 처리하기도 한다. 그리고 하나의 서버에 의해 제어되는 지리적인 영역을 셀이라 한다.



(그림 1) 이동 컴퓨팅 환경 모델

이동 호스트를 사용하는 사용자를 이동 사용자라 하는데, 이동 호스트가 다른 호스트와 통신하기 위해서는 서버를 경유해야 한다. 이동 컴퓨팅 환경에서 이동 사용자는 데이터베이스 검색이나 원하는 작업을 수행하기 위해 트랜잭션을 제출한다. 이동 호스트가 제출하는 트랜잭션을 이동 트랜잭션(Mobile Transaction)이라고 하는데[9], 이동 트랜잭션은 주로 판독 전용 트랜잭션(Read-only Transaction)이다. 그리고 기록 연산을 포함하는 트랜잭션을 갱신 트랜잭션(Update Transaction)이라고 하며, 갱신 트랜잭션은 서버에서 수행된다.

이동 컴퓨팅 환경이 갖는 여러 가지 제약 중 무선 통신망이 갖는 대역폭의 한계는 이동 호스트가 캐쉬를 사용함으로써 다소 완화될 수 있다. 이동 호스트는 자주 접근하는 데이터를 캐쉬에 저장해두고 연산하기 때문에, 이동 트랜잭션의 정확한 수행을 보장하기 위해 캐쉬 일관성은 중요하다.

이동 트랜잭션이 접근하려는 데이터가 캐쉬에 없을 때 이동 호스트는 서버에 데이터를 즉각적으로 요청하게 되는데, 이 때 데이터의 즉각적인 캐싱으로 인하여 이동 트랜잭션의 직렬성이 위배될 수 있다[10]. 캐쉬 관리 기법을 제안한 기존의 연구들[9,11]에서는 주로 서버와 캐쉬의 일관성 유지에 초점을 두고 있는데, 이동 호스트가 서버로부터 필요한 데이터를 즉각적으로 캐싱한다면 트랜잭션의 직렬가능한 수행 여부를 검사할 수 있는 방법을 필요로 한다.

이동 컴퓨팅 환경에서의 트랜잭션의 스케줄링 방법은 충돌 관계에 있는 연산들의 수행 순서를 결정하여 데이터베이스의 상태를 일관성 있게 유지하고, 잘못된 결과를 사용자가 접근하지 않도록 하기 위해 필요하다. 기존의 분산 환경에 비하여 이동 컴퓨팅 환경은 대역폭이나 이동 호스트의 성능에 제약을 가지므로 메시지 교환 비용과 트랜잭션의 성능 측면에서 효율적인 스케줄링 방법을 설계하는 것이 중요하다. 아직까지는 이동 컴퓨팅 환경에서 수행되는 트랜잭션의 스케줄링을 위해 제안된 방법들이 그리 많지 않다. 제안된 연구 중에서 록을 사용한 방법에서는 이동 호스트가 록을 설정하거나 해제하기 위해 서버와 자주 메시지를 교환해야 하며, 교착상태로 인한 일방적인 철회가 발

생할 수 있다는 문제점이 있다.

따라서 본 논문에서는 이동 호스트가 캐쉬를 사용할 때 발생할 수 있는 즉각적인 캐싱으로 인한 트랜잭션 직렬성 위배의 문제와 트랜잭션을 스케줄할 때 발생하는 부가적인 메시지 교환의 문제점들을 해결할 수 있는 새로운 트랜잭션 관리 방법을 제안하고자 한다. 제안하는 방법에서는 타임스탬프를 사용하여 트랜잭션을 스케줄함으로써 이동 호스트와 서버 사이에서 부가적인 메시지 교환이 발생하지 않는다. 또한 다중버전을 제공하여 이동 트랜잭션이 자신의 타임스탬프에 따라 접근할 수 있는 버전을 선택함으로써 필요한 데이터를 즉시 캐쉬하더라도 이동 트랜잭션의 직렬가능한 수행을 보장할 수 있다. 특히, 다중버전은 시스템의 고장시 회복에도 사용될 수 있으며, 늦게 도착한 이동 트랜잭션의 수행이나 오랜 기간 동안 수행되는 이동 트랜잭션의 수행 성능을 높이는데 효과적으로 사용될 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 트랜잭션의 스케줄링 기법을 다룬 기존의 연구들과 각 연구들에서 발생하는 문제점들을 지적하고, 3장에서는 구체적인 예를 통하여 문제를 정의한다. 4장에서는 본 논문의 시스템 모델과 타임스탬프를 이용한 새로운 트랜잭션 관리 방법을 제안한다. 마지막으로 5장에서는 제안하는 방법에 대한 논의와 결론을 기술한다.

2. 관련 연구

본 장에서는 이동 컴퓨팅 환경에서 트랜잭션 스케줄링을 위해 제안된 방법들에 대하여 살펴보고, 각각의 연구에서 발생하는 문제점들에 대하여 기술한다. 먼저, 기존의 분산 환경에서 사용되는 2단계 록킹(2PL : 2-Phase Locking) 방법과 타임스탬프 순서화(TO : Timestamp Ordering) 방법[12]에 대하여 살펴보고, 이동 컴퓨팅 환경에서 제안된 연구들에 대하여 살펴본다.

트랜잭션의 스케줄링을 위해 2단계 록킹 프로토콜을 사용할 때 각 트랜잭션은 접근하려는 데이터 항목들에 해당 록을 얻어야 하며, 만일 충돌 관계에 있는 록이 이미 설정되어 있으면 그 록이 해제되기까지 기다려야 한다. 트랜잭션의 수행 중 일단 록을 해제하기 시작하면 다시 얻을 수 없다. 이 방법을 이동 컴퓨팅 환경에 적용하면 다음과 같은 문제점들이 발생한다. 이동 컴퓨팅 환경에서 2단계 록킹 프로토콜을 사용하면 이동 호스트에서 수행되는 트랜잭션이 접근하는 데이터에 대한 록을 얻거나 해제하기 위해서 서버와 잦은 메시지 교환을 필요로 한다. 하지만 일반적으로 이동 호스트는 유선 망에 연결되어 있는 고정 호스트에 비하여 기억 장치 용량이나 처리 속도, 전력의 용량 등에 한계를 갖고 있기 때문에 서버에 메시지를 자주 전송하는 것은 이동 호스트 입장에서 상당한 부담이 된다. 또한 무선 통신망의 좁은 대역폭에 전송량을 증가시키는 원인이 되기 때문에 바람직하지 못하다. 그리고 이동 컴퓨팅 환경에서 이동 호스트는 언제 어디서나 이동할 수 있는데, 록의 설정이나 해제, 트랜잭션의 결과 반환, 교착상태 발생으로 인한 일방적인 철회가 발생하는 경우에 이것 또한 부가적인 메시지 교환을 발생시킨다.

타임스탬프 순서화 방법에서 수행되는 각 트랜잭션들은 유일한 타임스탬프를 할당받아 타임스탬프 순서대로 스케줄된다. 이 방법은 록에 비하여 단순하지만 이동 컴퓨팅 환경에 그대로 적용하기에는 어려운 점들이 있다. 첫째, 이동 호스트나 서버에서 수행되는 트랜잭션에 유일한 타임스탬프를 할당하기 위해서는 서버와 이동 호스트 사이에 동기화된 클럭을 유지할 수 있어야 하는데, 이동 호스트의 이동이나 단절 등으로 서버와 이동 호스트의 클럭을 동기화하는 것은 쉽지 않다. 둘째, 이동 호스트에서 수행되는 트랜잭션이 장기간동안 수행된다면, 자신의 타임스탬프로 접근할 수 있는 데이터가 갱신되어 철회될 가능성이 높아진다. 이동 트랜잭션은 통신망 단절이나 이동으로 인하여 장기간 수행될 가능성이 높기 때문에 이동 트랜잭션의 성능을 향상시킬 수 있는 방법이 필요하다.

지금까지는 기존의 분산 환경에서 사용되는 2단계 록킹 방법과 타임스탬프 순서화 방법을 이동 컴퓨팅 환경에 적용할 경우에 발생하는 문제점들을 살펴보았다. 다음으로는 이동 컴퓨팅 환경을 기반으로 하여 록을 이용한 O2PL-MT(Optimistic 2-Phase Locking for Mobile Transactions) 방법[13]과 CMC-MT (Concurrency control Method using Cache-lock for Mobile Transactions) 방법[14]에 대하여 살펴본다.

O2PL-MT 방법은 낙관적 방법을 기반으로 하여, 부분 중복 데이터베이스 환경에서 이동 호스트가 이동하면서 트랜잭션을 수행할 때 반드시 록을 설정한 사이트에서 록을 해제하지 않아도 현재의 위치에서 록

해제 메시지를 기록함으로써 판독 록 해제를 위한 메시지 교환을 줄일 수 있는 방법을 제안하였다. 갱신 연산을 위해 "w-intend"라는 새로운 록 모드를 도입하였고, 갱신 연산을 2단계 완료 프로토콜의 첫 번째 단계에서 수행하게 하여 록 설정과 투표(voting)를 통합하였다. O2PL-MT 방법에서 발생하는 문제점은 록을 기반으로 하고 있기 때문에 교착상태가 발생한다는 것과, 교착상태가 발생하였을 때 트랜잭션의 일방적인 철회를 위한 부가적인 메시지 교환이 발생한다는 점이다.

CMC-MT 방법에서는 비중복 데이터베이스 환경에서 캐쉬 록이라는 새로운 록 모드를 도입한 스케줄링 방법을 제안하였다. O2PL-MT 방법이 캐쉬를 사용하지 않는데 비하여 이 방법에서는 캐쉬를 사용하였다. 이 방법에서는 이동 호스트가 서버로부터 데이터를 캐쉬할 때 갱신 트랜잭션과의 직렬성을 유지하기 위해 새로운 록 모드인 캐쉬 록 모드를 도입하였다. 캐쉬 록은 다른 록 모드와 양립가능하여 캐쉬 록이 설정되었을 때 다른 모드의 록 설정을 허용한다. 캐쉬 록이 설정되어 있는 데이터 항목을 갱신 트랜잭션이 갱신하고자 할 때에는 직렬성 검사를 위해 Pre_set을 유지한다. 임의의 데이터 항목에 유지되는 Pre_set은 갱신 트랜잭션이 갱신 록을 요구할 때 그 데이터에 이미 캐쉬 록을 갖고 있는 트랜잭션이 있는 경우나 Pre_set이 공집합이 아닌 트랜잭션에 의해 해제된 록을 얻었을 때 트랜잭션 식별자로 구성된다. Pre_set은 Pre_set안에 있는 트랜잭션이 모두 완료될 때 삭제된다. 또한 Pre_set은 록킹 방법에서 발생할 수 있는 간접 충돌의 탐지 및 해결에도 사용된다. 하지만 이 방법에서의 문제점은 록을 얻고 해제하기 위한 메시지 교환과 트랜잭션의 직렬가능한 수행 여부를 검사하는데 필요한 자료구조를 유지하기 위한 메시지 교환이 많다는 점이다.

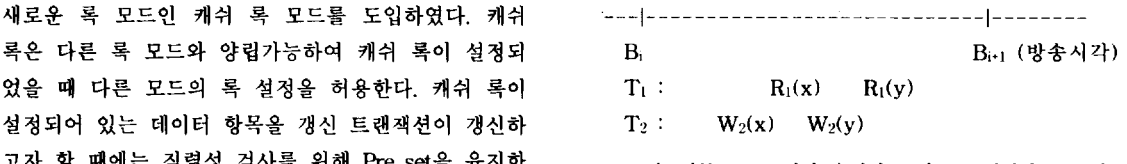
위와 같은 결과로서 이동 컴퓨팅 환경에서 사용될 트랜잭션의 스케줄링 방법은 이동 트랜잭션의 수행 성능을 저하하지 않고, 무선 통신망의 좁은 대역폭에서 부가적인 메시지를 발생시키지 않는 방법으로 설계되어야 한다는 것을 알 수 있다. 또한 이동 호스트에서는 데이터의 효율적인 접근을 위하여 캐쉬를 사용하는 경우가 많은데, 만약 이동 호스트가 캐쉬를 사용한다면 캐쉬내의 데이터와 서버에 유지되는 데이터 사이의 일관성 유지 및 즉각적인 캐싱으로 인한 트랜잭션의 직렬성 위배가 발생하지 않아야 한다. 본 논문에서는 캐쉬를 사용하는 환경에서 이러한 문제들을 해결할 수 있는 효율적인 스케줄링 방법을 제안하고자 한다.

3. 문제 정의

본 논문에서 제안하는 방법은 이동 호스트가 캐쉬를 사용하는 환경에서 트랜잭션의 스케줄링 방법을 다룬 것이다. 따라서 이 장에서는 이동 호스트가 캐쉬를 사용할 때 즉각적인 캐싱으로 발생하는 트랜잭션의 직렬성 위배와 록을 이용한 방법에서 발생하는 부가적인 메시지 교환의 문제를 구체적인 예를 통하여 기술한다.

먼저, 즉각적인 캐싱으로 인하여 발생하는 트랜잭션 직렬성 위배의 문제를 살펴본다.

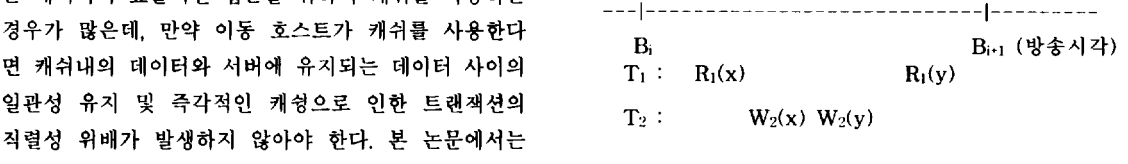
[예 1] 즉각적인 캐싱으로 인하여 발생하는 트랜잭션 직렬성 위배



T_1은 이동 호스트에서 수행되는 이동 트랜잭션으로, 이동 호스트의 캐쉬에는 x만 캐쉬 되어 있고, y는 연산에 의해 접근될 때 즉시 캐쉬된다고 가정하자. T_2는 서버에서 수행되는 갱신 트랜잭션이다. 연산들이 $W_2(x)$ $R_1(x)$ $W_2(y)$ $R_1(y)$의 순서로 수행된다면, $R_1(x)$는 이미 캐쉬되어 있는 데이타를 읽기 때문에 데이타 항목 x에 대한 직렬화 순서는 $T_1 \rightarrow T_2$, $R_1(y)$는 T_2가 갱신한 값을 판독하므로 데이타 y에 대한 직렬화 순서는 $T_2 \rightarrow T_1$이 되어 트랜잭션 직렬성이 파괴됨을 알 수 있다.

다음은 록을 이용한 스케줄링 방법 중 CMC-MT 방법에서 발생하는 문제점에 대하여 살펴본다. CMC-MT 방법에서 발생하는 문제점은 록 설정과 해제를 위해 이동 호스트가 서버에 접근해야 한다는 점과 이동 트랜잭션과 갱신 트랜잭션의 직렬화 순서를 유지하기 위해 부가적인 메시지 교환이 발생한다는 점이다.

[예 2] CMC-MT 방법에서 발생하는 부가적인 메시지 교환



CMC-MT 방법에서 이동 트랜잭션 T_1을 수행하기 위해서

는 x와 y가 캐쉬에 있더라도 캐쉬 록이 설정될 수 있는지의 여부를 서버에 문의하기 위해 메시지를 전송해야 한다. 서버에서는 캐쉬 록 요청에 대하여 갱신 록이 이미 설정되어 있지 않으면 설정 요구를 받아들인다. 따라서 CMC-MT 방법에서는 이동 트랜잭션이 접근하는 데이터의 수만큼 서버에 록 설정 요구를 보내야 한다. 위의 예에서는 트랜잭션의 수행 순서에 사이클이 발생하여 T_1은 y를 접근하는 시점에서 Pre_set 검사에 의해 철회되는데, 이동 트랜잭션 이 성공적으로 완료되면 록 해제를 위한 요구 메시지를 서버에 전송해야 한다. 이와 같이 록을 이용한 CMC-MT 방법에서는 록의 설정이나 해제를 위해 이동 호스트와 서버 사이에서 잦은 메시지 교환이 발생한다.

예 1과 예 2에서 지적한 바와 같이 이동 컴퓨팅 환경에서 트랜잭션의 스케줄링 방법은 즉각적인 캐싱으로 인한 트랜잭션의 직렬성 위배와 무선 통신망의 좁은 대역폭에서 부가적인 메시지 발생 문제들을 해결할 수 있어야 한다. 따라서 본 논문에서는 이러한 문제점들을 해결할 수 있는 타임스탬프를 이용한 스케줄링 방법을 제안하고자 한다.

4. TO/MC 알고리즘

이 장에서는 타임스탬프를 이용한 새로운 스케줄링 방법인 TO/MC(Timestamp Ordering Protocol in Mobile Computing Environments) 방법에 대하여 기술한다.

4.1 시스템 모델

먼저, 본 논문에서 사용하는 시스템 모델에 대하여 기술한다.

서로 다른 사이트에 존재하는 서버의 데이터베이스는 중복된 데이터를 갖지 않으며, 각 데이터 항목마다 여러 개의 버전이 유지된다. 이동 호스트의 캐쉬에는 저장된 데이터에 대하여 각각 두 개의 버전이 유지된다. 하나의 서버에 의해 제어되는 지리적인 영역인 각 셀들은 중첩되어 있으며, 이동 호스트는 임의의 순간에 하나의 서버와 연결을 갖는다. 본 논문에서는 1장의 (그림 1)과 같은 시스템 모델을 사용하고 있다.

4.2 가정 및 자료구조

본 논문에서 사용하고 있는 가정은 다음과 같다.

- 이동 트랜잭션은 판독 전용 트랜잭션이다.
- 갱신 트랜잭션은 서버에서 수행된다.
- 각 서버의 클럭들은 동기화된다.
- 각 이동 호스트들은 효율적인 데이터 접근을 위하여 캐쉬를 사용한다.
- 각 이동 호스트의 캐쉬에 유지되는 각 데이터 항목에 대하여 두 개의 버전을 유지한 다.
- 이동 호스트에서 수행되는 트랜잭션은 한 번에 하나씩 순차적으로 수행된다.
- 데이터의 최신 버전은 그 데이터를 갱신한 트랜잭션이 완료할 때 갱신된다.

TO/MC 알고리즘을 위해 이동 호스트와 서버에 다음과 같은 자료구조를 유지한다.

먼저, 이동 호스트에 유지되는 자료구조와 의미는 <표 1>과 같다.

〈표 1〉 이동 호스트에 유지되는 자료구조

자 료 구 조	의 미
LBT(Latest Broadcasting Time)	가장 최근의 방송 시각
V_old(x)	이동 호스트의 캐쉬에 유지되는 x의 구버전
V_new(x)	이동 호스트의 캐쉬에 유지되는 x의 신버전
wts(x)	데이터 항목 x를 갱신한 트랜잭션의 완료 시각

서버에 유지되는 자료구조와 의미는 <표 2>와 같다.

〈표 2〉 서버에 유지되는 자료구조

자 료 구 조	의 미
Bcrt(Currect Broadcasting Time)	현재의 방송 시각
LV(x) (Latest Version of x)	데이터 항목 x의 가장 최신 버전
wts(x)	데이터 항목 x를 갱신한 트랜잭션의 완료 시각
ts(x)	데이터 항목 x를 갱신한 트랜잭션의 타임스탬프
mutiversion of each data item	각 데이터 항목에 유지되는 다중 버전

4.3 타임스탬프 할당

TO/MC 방법은 타임스탬프를 이용한 방법이므로

트랜잭션의 스케줄링을 위해 이동 호스트와 서버에서 수행되는 트랜잭션에 타임스탬프를 할당하고 타임스탬프 규칙에 따라 트랜잭션들을 스케줄한다. 먼저, 이동 트랜잭션에 타임스탬프를 할당하는 방법에 대해 기술한다.

이동 트랜잭션의 타임스탬프는 트랜잭션이 제출된 시점에서 가장 가까운 이전 방송 시각으로 결정된다. 이동 호스트가 서버와 연결을 유지하고 있었다면 가장 최근의 방송 메시지를 수신할 수 있었을 것이다. 방송 메시지는 하나의 방송 구간동안 갱신된 데이터를 이동 호스트에게 알리기 위해 서버에서 주기적으로 방송하는 메시지로, 방송 메시지에는 갱신된 데이터에 대한 정보와 현재의 방송 시각이 포함되어 있다. 만약 이동 호스트가 서버와 계속적인 연결이 유지되지 않았다면 연결을 시도하여 서버로부터 가장 최신의 방송 시각을 읽어와서 이동 트랜잭션의 타임스탬프로 할당한다. 다음의 알고리즘 1은 이동 트랜잭션에 타임스탬프를 할당하는 알고리즘이다.

Algorithm for Assigning a Timestamp to MT

```
1. if a MH has been connected to its server
       then the latest broadcasting time is assigned to MT;
       else {
            try to connect to its server;
            get the latest broadcasting time from its server;
            assign it to MT;
```

(알고리즘 1) 이동 트랜잭션의 타임스탬프 할당 알고리즘

갱신 트랜잭션에 타임스탬프를 할당하는 방법은 기존의 유선 환경에서 사용하는 방법을 사용한다. 즉, 트랜잭션이 서버에 도착하는 시간에 사이트 식별자 (식별자 > 1)를 소수 부분으로 추가하여 갱신 트랜잭션의 타임스탬프로 할당한다. 예를 들어, 사이트 1에서 10이라는 시간에 도착한 갱신 트랜잭션의 타임스탬프는 10.1이 된다. 이러한 방법에 의하여 트랜잭션의 타임스탬프를 부여하면, 트랜잭션의 식별자가 10, 100, 1000과 같은 경우에 동일한 타임스탬프가 생성될 수 있다. 이 문제는 트랜잭션의 식별자를 부여할 때 검사해서 10이나 100이 아닌 11이나 111이 되도록 함으로써 피할 수 있다.

Algorithm for Assigning a Timestamp to UT

```
1. ts(UT) = attach the site_id as a fraction part to arrival time of UT;
   /* ts(UT)는 갱신 트랜잭션의 타임스탬프 */
```

(알고리즘 2) 갱신 트랜잭션의 타임스탬프 할당 알고리즘

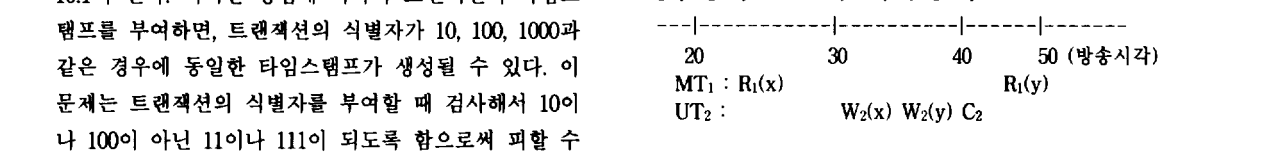
4.4 이동 호스트 알고리즘

이동 호스트에서 수행되는 이동 트랜잭션의 스케줄링을 위한 알고리즘은 다음과 같다. 이동 호스트가 서버와 계속적인 연결을 유지하고 있었을 때 이동 호스트에서 수행되는 작업은 크게 두 가지로 구분할 수 있다. 하나는 서버에서 방송된 메시지를 수신하였을 때 방송 메시지에 따라 캐쉬에 유지되는 버전을 관리하는 일이고, 다른 하나는 이동 트랜잭션을 수행하기 위해 캐쉬에 접근할 데이터가 있는지 점검하여, 접근하려는 데이터가 캐쉬에 없으면 서버에 요구하는 일이다.

TO/MC 방법에서 이동 호스트의 캐쉬에는 캐쉬된 각 데이터 항목 x에 대하여 두 개의 버전이 유지된다. 하나는 신버전(V_new(x))이고, 다른 하나는 구버전(V_old(x))이다. 이동 트랜잭션이 캐쉬에 있는 데이터를 접근하여 수행될 때 캐쉬내의 두 개의 버전 중에서 접근할 수 있는 버전은 자신의 타임스탬프보다 작거나 같은 타임스탬프의 버전 중에서 가장 큰 버전이다.

임의의 이동 트랜잭션이 시작되기 전 캐쉬에 저장된 각 두 개의 버전은 동일한 값으로 복사된다. 즉, 현재 유지하고 있는 신버전을 구버전으로 복사한다. 트랜잭션이 시작되면 구버전을 접근하여 연산을 수행하고, 트랜잭션이 종료되기 전 새로운 방송 메시지를 수신하면 신버전만을 갱신한다. 이렇게 하면 이동 트랜잭션이 많은 방송 구간에 걸쳐 장기간 수행된다 하더라도 접근할 수 있는 버전이 구버전에 유지된다. 그리고나서 트랜잭션이 완료한 후 새로운 트랜잭션이 시작되기 전 신버전을 구버전으로 복사한다.

[예 3] 이동 트랜잭션의 수행 예



각각 두 개의 연산으로 구성된 이동 트랜잭션 MT_1과 사이트 1에서 수행되는 갱신 트랜 잭션 UT_2가 위와 같을 때, 타임스탬프 할당 규칙에 의하여 $ts(MT_1)=20$, $ts(UT_2)=32.1$가 된다. MT_1이 수행되는 이동 호스트의 캐쉬에 x와 y가

캐쉬되어 있다고 가정하고, 유지되어 있는 두 버전의 타임스탬프가 V_new(x)=18, V_old(x)=7, V_new(y)=18, V_old(y)=5라고 하자. 이동 트랜잭션의 수행을 시작하기 전 두 개의 버전은 복사를 통 해 동일한 값을 갖고 있다. $R_1(x)$는 자신의 타임스탬프가 20이므로 x의 구버전을 접근한다(신버전을 접근할 수도 있다). UT_2의 두 개의 갱신 연산이 수행된 후 완료 시점을 37 이라 하면, wts(x)= wts(y)=37이 된다. 방송 시각 40에서 x와 y의 갱신이 포함된 방송 메시지를 이동 호스트가 받으면 캐쉬의 내용을 V_new(x)=37, V_old(x)=18, V_new(y)=37, V_old(y)=18로 갱신한다. 이동 트랜잭션의 두 번째 연산 $R_1(y)$가 수행될 때는 캐쉬에 유지된 두 개의 y에 대한 버전 중 구버전을 접근함으로써 수행가능하다. MT_1이 종료한 후 MT_2가 수행을 시작한다면 신버전을 구버전으로 복사하는 단계부터 시작하여 반복한다. 이동 호스트의 캐쉬에 유지되는 두 개의 버전으로 이동 트랜잭션이 여러 개의 방송 주기동안 수행되어도 수행을 완료할 수 있다.

다음의 알고리즘 3은 이동 트랜잭션을 스케줄하기 위한 알고리즘이다.

Algorithm for Mobile Host (MH) to Schedule Operations of a Mobile Transaction (MT) and to Cache a Data from a Server

```
1. when a MH receives a message BM from a server,
        /* it reflects the content of the message into its cache */
   for each data item x in the message {
      if wts(V_new(x)) > ts(MT)
         V_new(x) = value of x in the BM;
   }
2. before start a mobile transaction      /* initialize a cache */
   for each data item x in the cache
         V_old(x) = V_new(x);
3. when an operation of a MT access a data item
      it first accesses the cache for reading a required data item
      if the required data item is in the cache
        then read the latest version satisfying a condition, wts(x) < ts(MT);
        else send to its server a message such as RQ(x, ts(MT)) to request a data
        satisfying a condition, wts(x) < ts(MT);
/* The message from a server is a response for the request or
   a broadcasted message */
```

(알고리즘 3) 이동 호스트 알고리즘

4.5 서버 알고리즘

서버에서 수행되는 갱신 트랜잭션의 스케줄링을 위한 알고리즘은 다음과 같다. 서버에서 수행되는 작업은 이동 호스트에서 데이터 캐싱 요구에 응답하는 일과 방송 시각에서 방송 메시지를 구성하여 방송하는 일, 그리고 갱신 트랜잭션을 스케줄하는 일이다. 서버가 이동 호스트로부터 캐싱 요구를 받으면 이동 트랜잭션의 타임스탬프로 접근할 수 있는 버전을 선택하여 전송한다. 주기적인 방송 시각에서 서버는 방송 주기 동안 갱신된 데이터에 대한 정보와 방송 시각을 방송 메시지로 구성하여 방송한다. 그리고 갱신 연산들을 수행할 때 데이터에 대한 새로운 버전을 생성할 것인지를 결정한다. 새 버전이 생성되는 경우는 마지막 갱신 완료 이후 처음으로 데이터가 갱신되는 경우이다. 일단 새 버전이 생성되면 그 후에 발생되는 동일한 데이터 항목에서의 갱신은 하나의 새 버전에서 계속 수행된다. 다음의 예 4에서 갱신 트랜잭션들이 동일 데이터 항목을 반복 갱신하는 예를 기술한다.

[예 4] 동일 데이터 항목의 반복 갱신의 예

```
---|------------------------------|------
   20                                30 (방송시각)
UT1 : W1(x) W1(y) C1
UT2 :                    W2(x) C2
```

사이트 1과 2에서 각각 수행되는 두 개의 갱신 트랜잭션 UT_1과 UT_2가 위와 같이 수행되며, $ts(UT_1)$=22.1, $ts(UT_2)$=27.2라 가정하자. $W_1(x)$와 $W_1(y)$에 의해 데이터 항목 x와 y는 새로운 방송 시점에서 처음 갱신되므로 새로운 버전을 하나 만들어 갱신한 후 완료한다. 그 후 $W_2(x)$에 의해 x는 하나의 방송 구간에서 두 번 갱신되는데, 이 때 $W_2(x)$는 새로운 버전을 생성하는 것이 아니라 $W_1(x)$에 의해 이미 만들어진 새 버전을 갱신한다. 따라서 새로운 버전은 $W_1(x)$에 의해 한 번만 만들어지며, UT_2가 완료되었을 때 x의 새 버전에는 $W_2(x)$에 의해 갱신된 값이 남아있게 된다. 이렇게 한 방송 구간에서 데이터 항목이 중복 갱신될 때는 처음 만들어진 새 버전을 반복 갱신하게 된다.

다음의 알고리즘 4는 갱신 트랜잭션을 스케줄하기 위한 알고리즘이다.

Algorithm for Server to Respond a Request from a MH and Schedule Update Transactions(UT)

```
1. when it receives a request from a MH, RQ(x, ts(MT))
     it select the latest version satisfying a condition
          such as wts(x) < ts(MT) and send the version to the MH;
2. when it becomes a broadcasting time
     it send a message such as
          BM(Bcrt, a list of data items updated in the broad-
             casting time interval);
```

```
        /* BM : Broadcasting Message */
3. when it schedules a write operation of a UT according to
   its TO protocol
   if wts(LV(x)) < ts(Bcrt)
     then {  create a new version for x and the new version
          becomes LV(x);
            update LV(x);
     }
     else udpate LV(x) according to a timestamp ordering
          rule;  /* wts(LV(x)) > ts(Bcrt) */
```

(알고리즘 4) 서버 알고리즘

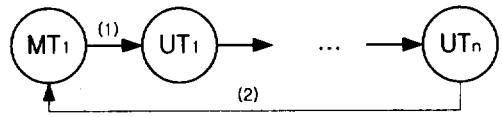
다음은 제안하는 TO/MC 알고리즘의 정확성을 증명하기 위해 TO/MC 알고리즘이 트랜잭션의 직렬가능한 수행을 보장함을 보인다. 직렬성(serializability)은 동시성 제어 방법의 정확성을 증명하기 위한 기준이다 [12]. 따라서 본 논문에서도 TO/MC 알고리즘이 트랜잭션의 직렬가능한 수행을 보장함을 보임으로써 정확성을 증명한다.

[정리] TO/MC 방법은 트랜잭션의 직렬성을 보장한다.

(증명) TO/MC 방법이 트랜잭션의 직렬성을 보장하지 못한다면 TO/MC 방법에 의하여 만들어진 직렬화 그래프에 사이클이 발생하게 된다. 증명을 위해 TO/MC에 의해 만들어진 직렬화 그래프에는 사이클이 발생하지 않음을 보인다.

TO/MC에 의해 만들어진 직렬화 그래프내의 에지 $MT_1 \rightarrow UT_2$은 $ts(MT_1) < wts(UT_2)$ 임을 의미한다. 또한 $UT_1 \rightarrow MT_2$의 에지는 $wts(UT_1) < ts(MT_2)$이면서 $ts(UT_1) < ts(MT_2)$임을 의미한다. 그리고 $UT_1 \rightarrow UT_2$의 에지는 $ts(UT_1) < ts(UT_2)$임을 의미한다.

증명을 위해 TO/MC에 의해 만들어진 직렬화 그래프 SG에 (그림 2)와 같이 $MT_1 \rightarrow UT_1 \rightarrow UT_2 \rightarrow \ldots \rightarrow UT_n \rightarrow MT_1$과 같은 사이클이 존재한다고 가정하자. SG에 있는 에지 중 이동 트랜잭션 MT로부터 갱신 트랜잭션 UT로의 에지(1)MT → ... → UT는 ts(MT) < wts(UT)임을 의미한다. 또한 갱신 트랜잭션 UT로부터 이동 트랜잭션 MT로의 에지(2) UT → ... → MT는 wts(UT) < ts(MT)임을 의미한다. SG의 사이클에 포함되어 있는 임의의 갱신 트랜잭션 UT_i를 고려하여 보자. $MT_1 \rightarrow \ldots \rightarrow UT_i$의 에지에 의하여 $ts(MT_1) < wts(UT_i)$이며, $UT_i \rightarrow \ldots \rightarrow MT_1$의 에지에 의하여 $wts(UT_i) < ts(MT_1)$이다. 두 관계로부터 $ts(MT_1) < wts(UT_i) < ts(MT_1)$의 관계가 성립함을 알 수 있다. 이것은 하나의 이동 트랜잭션의 타임스탬프가 유일하다는 점에 모순이다. 따라서 TO/MC 방법은 직렬화 그래프에 사이클이 발생하지 않도록 트랜잭션들을 스케줄할 수 있으며, 트랜잭션의 직렬가능한 수행을 보장한다는 것을 알 수 있다.



(그림 2) 직렬화 그래프 SG

5. 논의 및 결론

본 논문에서는 이동 컴퓨팅 환경에서 타임스탬프를 이용한 트랜잭션 스케줄링 방법을 제안하였다. 이동 호스트는 대역폭의 한계를 극복하고, 서버와 교환되는 메시지를 줄이기 위하여 캐쉬를 사용하며, 필요한 데이터가 캐쉬에 없었을 때는 필요한 데이터를 즉각적으로 캐쉬한다.

이동 컴퓨팅 환경에서 수행되는 트랜잭션의 스케줄링을 위해 기존의 분산 환경에서 제안된 록이나 타임스탬프 방법을 적용할 때는 서버와 교환되는 메시지 비용의 증가와 이동 트랜잭션의 수행 성능 저하, 그리고 이동 호스트의 이동성을 지원하지 못하는 문제점이 있다. 따라서 본 논문에서는 타임스탬프를 이용하여 이동 컴퓨팅 환경의 특성을 지원할 수 있는 TO/MC 알고리즘을 제안하였다.

TO/MC 방법에서 이동 트랜잭션의 타임스탬프는 트랜잭션이 제출된 시점에서 가장 가까운 이전 방송 시각으로 결정된다. 이 방송 시각은 이동 호스트가 서버와 계속적인 연결 상태를 유지하고 있었다면 방송 메시지를 수신함으로써 알 수 있고, 이동 호스트가 서버와 단절 상태였다면 서버와 연결을 시도함으로써 알 수 있다. 또한 갱신 트랜잭션의 타임스탬프는 트랜잭션이 서버에 도착하는 시간으로 결정하였다. 이와 같은 방법으로 트랜잭션의 타임스탬프를 결정하면 서버와 이동 호스트 사이의 클럭의 동기화가 요구되지 않는다.

이동 호스트의 캐쉬에는 캐쉬된 각 데이터에 대하여 두 개의 버전을 유지하고 있다. 캐쉬에 두 개의 버전을 유지함으로써 이동 트랜잭션이 여러 개의 방송 주기동안 수행되어도 접근할 수 있는 데이터를 캐쉬에 유지할 수 있다. 또한 하나의 이동 트랜잭션이 끝나고 새로 시작되는 이동 트랜잭션도 데이터가 캐쉬에 있는 한 서버에 접근하지 않아도 된다. 서버에는 각 데이터

에 대하여 여러 개의 버전을 유지한다. 버전은 완료된 갱신 트랜잭션에 의해 갱신된 데이터가 처음 갱신될 때 생성되며 더 이상 접근되지 않을 때 삭제될 수 있다. 본 논문에서는 구체적인 버전 삭제 방법을 기술하지 않았으나 임의 버전을 접근하는 가장 작은 타임스탬프의 트랜잭션이 완료되면 버전을 삭제할 수 있다고 가정하였다. 하나의 방송 구간동안 하나의 데이터에 반복되는 갱신이 발생할 때는 하나의 새 버전에서 연산이 수행된다.

이동 트랜잭션은 접근하려는 데이터가 캐쉬에 있으면 캐쉬를 접근하여 수행되고, 캐쉬에 없으면 자신의 타임스탬프로 접근가능한 버전을 서버에 요구하여 접근한다. 이동 호스트가 방송 시점에서 방송 메시지를 받으면 조건에 따라 신버전과 구버전을 갱신한다. 갱신 트랜잭션은 타임스탬프 규칙에 따라 서버에서 수행되며, 서버에는 각 데이터에 대하여 여러 개의 버전을 유지한다. 서버는 이동 호스트의 캐쉬 요청에 대하여 적절한 데이터를 전송함으로써 응답하고, 갱신 트랜잭션을 스케줄한다. 또한 방송 시점에서 방송 메시지를 구성하여 여러 호스트에게 방송한다.

TO/MC 방법은 기존의 트랜잭션 스케줄링 방법에서 지적되었던 문제점들을 갖지 않는다. 즉, 트랜잭션의 스케줄링을 위해 서버와 부가적으로 교환되는 메시지는 발생하지 않으며, 캐쉬와 다중버전을 사용함으로써 이동 트랜잭션의 수행 성능을 향상시킬 수 있었다.

지금까지 본 논문에서 기술한 내용에 추가하여 앞으로 이동 호스트에서 갱신 트랜잭션이 수행되는 경우와 여러 개의 트랜잭션이 동시에 수행되는 경우에 대한 스케줄링 방법에 대한 연구도 진행되어야 할 것이다.

참 고 문 헌

[1] Brian Marsh, Fred Douglis, and Ramon Caceres, "System Issues in Mobile Computing," Technical Report Aachen University at Carnegie Mellon University Matsushita Information Technology Laboratory MITL-TR-50-93, Feb. 1993.

[2] Tomasz Imielinski, and B. R. Badrinath, "Wireless Mobile Computing : Challenges in Data Management," *COMMUNICATIONS of the ACM*, October 1994, pp.19-27.

[3] Pictoura E., and B. Bhargava, "Maintaining Consistency of Data in Mobile Distributed Environments," *In Proceedings of 15th International Conference on Distributed Computing System(ICDCS95)*, May, 1995, pp.404-413.

[4] Rafael Alonso, and Henry F. Korth, "Database System issues in Nomadic Computing," *ACM SIGMOD*, 1993, pp.388-392.

[5] Tomasz Imielinski, "Data Management for Mobile Computing," *SIGMOD Record*, 1993, pp. 34-39.

[6] M. Satyanarayanan, James J. Kistler, and Lily B. Mummert, "Experience with Disconnected Operation in a Mobile Computing Environment," *CMU-CS-93-168*, 1993.

[7] James J. Kistler, and M. Satyanarayanan, "Disconnected Operation in the Coda File System," TOCS10(1), 1992, pp.3-25.

[8] Margaret H. "Mobile Computing and Databases : Anything new?," *SIGMOD Record*, 1995, pp. 5-9.

[9] Daniel Barbara, and Tomasz Imielinski, "Sleepers and Workaholics : Caching Strategies in Mobile Environments," *VLDB Journal*, Dec., 1995.

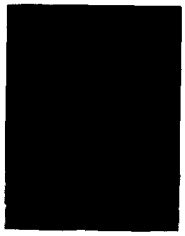
[10] 김대인, 황부현, "이동 컴퓨팅 환경에서 사이클 탐지를 이용한 트랜잭션 직렬성 유지 기법", *한국정보과학회 학술발표 논문집, Vol.24, No.1*, April 25, 1997.

[11] Man Hon Wong, and Wing Man Leungm, "A Caching Policy to Support Read-only Transactions in a Mobile Computing Environment," *CS-TR-95-07*, May 1995.

[12] Bernstein P. A, and V Hadzilacos, *Concurrency Control and Recovery for Database Systems*, Reading, Mass., Addison-Wesley, pp.11-17, 1987.

[13] Jin Jing, Omran Bukhres, and Ahmed Elmagarmid, "Distributed Lock Management for Mobile Transactions," *Proceedings of the 15th IEEE International Conference on Distributed Computing Systems(ICDCS '95)*, pp.118-125, May 1995.

[14] 김치연, 황부현, "이동 컴퓨팅 시스템에서 캐쉬 록을 이용한 동시성 제어 방법", *한국정보처리학회 논문지, Vol.5, No.2*, February 1998.



김 치 연

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

1992년 2월 전남대학교 전산통계학과(학사)

1994년 2월 전남대학교 대학원 전산통계학과(이학석사)

1997년 2월 전남대학교 대학원 전산통계학과(박사과정수료)

1996년 3월~1997년 2월 전남대학교 전산학과 조교

1999년 2월~현재 전남대학교 전산학과 시간강사

관심분야 : 분산 시스템, 이동 컴퓨팅, 실시간 시스템

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