이동 데이터베이스 시스템에서 타임스탬프를 이용한 낙관적 동시성 제어 기법☆

Optimistic Concurrency Control Using Time-stamp Ordering in Mobile Databases

김 대 호* 정 병 수** 이 영 구***
Dae-Ho Kim Byeong-Soo Jeong YoungKoo Lee

요 약

이동 컴퓨팅 환경에서는 서버와 이동 클라이언트 사이의 비대칭적인 통신 대역폭의 특성 때문에 데이터 방송 기법이 효과적인 데이터 전달 방식으로 널리 사용되고 있다. 방송 기법을 사용하는 이동 데이터베이스 시스템에서는 무선 네트워크 환경과 이동 컴퓨팅 환경을 고려한 새로운 동시성 제어 기술이 요구된다. 본 논문에서는 낙관적 동시성 제어 기법을 기반으로 이동 컴퓨팅 환경에 적합한 OCC/DTA(Optimistic Concurrency Control with Dynamic Time-stamp Adjustment)기법을 제안한다. 제안하는 기법은 이동 클라이언트에서의 부분적 검증 작업을 통하여 비대칭적인 무선 네트워크 환경의 통신 부하를 줄일 수 있으며, 이동 트랜잭션들의 직렬화 순서를 동적으로 재조정함으로써 이동 트랜잭션의 처리율을 향상 시킬 수 있다. 제안한 기법이 데이터 일관성을 만족할 수 있음을 이론적으로 증명하고, 모의실험을 통하여 동시성 제어 기법의 성능을 분석한다.

Abstract

Data broadcasting is an efficient method for disseminating data, and is widely accepted in the database applications of mobile computing environments because of its asymmetric communication bandwidth between a server and mobile clients. This requires new types of concurrency control mechanism to support mobile transactions executed in the mobile clients, which have low-bandwidths toward the server. In this paper, we propose an OCC/DTA (Optimistic Concurrency Control with Dynamic Time-stamp Adjustment) protocol that can be efficiently adapted to mobile computing environments. The protocol reduces communication overhead by using client-side validation procedure and enhances transaction throughput by adjusting serialization order without violating transaction semantics. We show that the proposed protocol satisfies data consistency requirements, and simulate that this protocol can improve the performance of mobile transactions in data broadcasting environments.

☞ Keyword : 이동 컴퓨팅(Mobile Computing), 동시성제어(Concurrency Control), 방송환경(Broadcast Environment), 데이터 일관성(Data Consistency), 타임스탬프(Time-stamp)

1. 서 론

휴대폰, 노트북, PDA 등과 같은 이동 기기의 보급과 무선 네트워크 기술의 발전으로 이동 컴퓨팅 환경에서 여러 분야의 정보 서비스, 증권정보, 교통정보, 기상정보, 등의 정보 서비스가 광범위하게 있다[1,2]. 데이터베이스 서버로부터 무선 채널을 통해 이동 클라이언트에 데이터를 제공하는 방식에는 요구에 의한 방법(On-Demand)과 방송기법(Broadcast-based)이 있다[3].

무선 채널은 서버에서 클라이언트로의 하향(Down-Stream) 대역폭은 매우 크고 반대 방향인 상향(Up-Stream) 대역폭은 매우 작은 비대칭적인 특징을 갖는다. 따라서 클라이언트의 요구에

* 정 회 원 : 경희대학교 일반대학원 컴퓨터공학과 박사
techwing@khu.ac.kr
** 정 회 원 : 경희대학교 전자정보대학(컴퓨터공학전공) 교수
jeong@khu.ac.kr
*** 종신회원 : 경희대학교 전자정보대학 조교수
yklee@khu.ac.kr
[2005/11/24 투고 - 2005/12/08 심사 - 2006/06/07 심사완료]
☆ "이 연구는 2005년도 경희대학교 지원에 의한 결과임" (KHU-20050354)

의해 데이터가 전송되는 On-Demand 방식은 많은 이동 클라이언트가 각기 다른 데이터 항목을 원하는 경우 상향 링크의 혼잡과 병목현상에 의해 데이터 항목에 대한 대기 시간이 길어진다. 또한, 동일한 데이터 항목을 원하는 경우 중복 전송이 불가피하다. 반면에 방송 기법은 서버가 클라이언트의 요청과 무관하게 주기적으로 데이터 항목을 배포하는 방법으로 이동 클라이언트는 방송 채널을 통해 필요한 데이터 항목을 선택적으로 접근한다. 방송 기법은 이동 클라이언트의 수와 무관하게 제한된 대역폭을 통해 데이터를 배포할 수 있기 때문에 이동 데이터베이스 환경에서는 요구에 의한 방법보다 효율적일 수 있다.

일반적인 트랜잭션 처리 환경과 마찬가지로 방송 기법에서도 서로 다른 이동 트랜잭션들이 동시에 같은 데이터 항목을 사용하는 경우가 발생할 수 있으므로, 데이터의 일관성(Data Consistency)을 유지하기 위해 동시성 제어 기술이 요구된다[4]. 일반적인 트랜잭션 처리 환경에서 활용되는 2단계 잠금 규약(Two Phase Locking Protocol) 방식은 필요한 모든 데이터 항목에 대한 잠금을 서버에 요청하게 되는 데, 이는 통신 제약이 많은 이동 컴퓨팅 환경에서는 효과적으로 구현하기가 매우 어렵게 된다[4,9,15]. 따라서 대부분의 이동 컴퓨팅 환경에서는 데이터 일관성 유지를 위한 동시성 제어 기법으로 검증 단계에서 데이터 충돌을 감지하여 데이터 일관성을 유지하는 낙관적 동시성 제어 방법을 사용한다.

이동 컴퓨팅 환경에서 낙관적 동시성 제어 방법을 사용하는 경우 트랜잭션의 검증 단계 작업이 데이터베이스 서버에서 이루어지게 된다. 따라서 트랜잭션의 상당 부분을 서버에 의존하게 되고, 데이터 충돌로 인하여 취소되는 트랜잭션들은 불필요하게 통신 대역을 사용하게 된다. 이는 상대적으로 통신 대역폭이 작은 이동 컴퓨팅 환경에서는 실행이 지연되고 그에 따른 응답 시간이 증가함을 의미한다. 그러므로 취소가 예상되는 트랜잭션들을 이동 클라이언트에서 탐지하여 미리 취소할 수 있다면 서버로의 통신 대역폭을 효율적으로 사용하게 되어 응답 시간을 개선할 수 있는 방법이 될 수 있다.

이동 컴퓨팅 환경에서 낙관적 동시성 제어 기법을 사용하는 기존의 기법[15,17]들은 대부분 읽기 연산으로만 이루어진 이동 트랜잭션을 고려하고 있다. 실제로 기상정보, 교통정보, 주식정보 등과 같은 정보 서비스에서는 쓰기 연산은 서버에서만 이루어지고 이동 클라이언트에서는 읽기 연산만이 수행된다. 하지만, 이동 기기를 이용한 경매나 주식 거래, 전자 상거래 등과 같이 쓰기 연산이 필요한 응용 분야로 확장되기 위해서는 이동 클라이언트에서 수행되는 쓰기 연산을 고려하여 데이터 일관성을 유지할 수 있는 동시성 제어 기법이 연구되어야 한다.

본 논문은 방송 기법을 사용하는 이동 데이터베이스 시스템에서 쓰기 연산이 포함된 이동 트랜잭션을 위한 동시성 제어 기법인 OCC/DTA (Optimistic Concurrency Control with Dynamic Timestamp Adjustment)를 제안한다. 제안하는 기법은 데이터 충돌이 발생한 경우 서버에서 전송된 검증 정보를 통해 트랜잭션의 타임스탬프를 동적으로 재조정하고 검증 작업의 일부를 이동 클라이언트에서 수행한다. 이와 같은 방법으로 제안한 기법은 상향 링크의 통신 요구를 줄이고 트랜잭션의 완료율을 높일 수 있다. 제안하는 기법이 데이터 일관성을 만족할 수 있음을 이론적으로 증명하고, 모의실험을 통하여 제안한 기법의 성능을 분석한다.

본 논문의 구성은 2절에서 이동 데이터베이스 환경을 위해 지금까지 제안된 동시성 제어 기법들의 내용과 특징을 간략히 요약하고 3절에서는 제안된 기법의 내용을 기술하고 논리적 정확성을 증명한다. 4절에서는 모의실험을 통해 제안된 기법의 성능을 분석하며 5절에서 결론을 맺는다.

2. 관련연구

이동 컴퓨팅 환경에서 데이터 전달 방식에 대한 연구는 국내외에서 활발하게 연구되고 있다[3,5-7]. 특히 무선 네트워크의 비대칭적 통신 특성으로 인하여 방송 기법을 사용하는 다양한 형태의 데이터 배포 방식이 많이 연구되어 왔다[3,5,7]. 참고문헌 [6]에서는 Push 기법과 Pull 기법 그리고 이 두 가지를 혼합한 기법에 대한 성능 특성을 분석하였다. 참고 문헌 [3]에서는 방송디스크(Broadcast Disk)에 대한 개념을 소개하였다. 이는 방송 대역을 이동 데이터베이스 시스템의 저장 장치와 유사하게 사용할 수 있도록 하는 무선 데이터 전송 방법이다.

최근에는 방송 환경에서의 트랜잭션 처리를 위한 여러 연구들이 발표되고 있다[8-13]. 이들 중 대부분은 이동 데이터베이스 환경에서의 데이터 일관성 유지를 위한 동시성 제어 기법을 제안하고 있다. 이들 동시성 제어 기법들은 주로 이동 클라이언트에서 수행되는 읽기-전용 트랜잭션의 처리를 고려하고 있다. 참고문헌 [9]에서는 데이터 일관성 유지를 위해 약한 일관성(Weak Consistency)을 제안하고 있다. 약한 일관성은 서버의 개입 없이 제어 행렬을 통해 데이터 항목을 읽기 전에 데이터 일관성 검사를 수행함으로써 읽기-전용 트랜잭션의 데이터 일관성을 보장해준다. 그러나 이 기법은 데이터 항목의 수가 많아지면 제어 정보의 크기가 너무 커지는 단점을 가지고 있어 데이터베이스의 크기가 클 경우에는 활용되기 어려운 문제점이 있다.

참고문헌 [10]에서는 읽기-전용 트랜잭션의 일관성을 보장하기위해 다중 버전 방송(Multi-version Broadcast)기법과 무효화 보고(Invalidation Report)를 이용하는 기법을 제안하고 있다. 다중 버전 방송기법은 각 데이터 항목에 대해 여러 버전을 방송하는 기법으로, 무효화 보고를 이용하여 서버에서 갱신된 데이터 항목을 취소시킴으로써 읽기-전용 트랜잭션의 완료율을 높이고 있다. 이 기법은 데이터 일관성 검증을 위해 직렬화 그래프(Serialization Graph)를 서버와 이동 클라이언트에서 모두 가지고 있어야 하므로 방송 주기가 길어지거나 동시성이 낮아지는 문제점을 가지고 있다.

참고문헌 [17]에서는 직렬화(Serialization) 순서 유지를 위한 검증 작업의 일정 부분을 이동 클라이언트에서 수행하는 FBOCC 기법을 제안하였다. FBOCC 기법은 이동 클라이언트에서의 검증 작업을 위하여 주기적으로 서버에서 실행이 완료된 트랜잭션들의 수행 정보를 모든 이동 클라이언트들에게 방송한다. 이동 클라이언트에서는 방송된 수행 정보를 이용하여 충돌이 발생할 트랜잭션들을 미리 탐지하여 취소시킴으로써 서버와 이동 클라이언트들 간의 불필요한 통신을 줄이고 있다. 하지만, 직렬화 순서의 고려 없이 이미 완료된 트랜잭션들과 충돌이 일어나는 실행중인 모든 트랜잭션들을 취소하게 되므로 트랜잭션 완료율이 현격히 낮아질 수 있는 문제점이 있다.

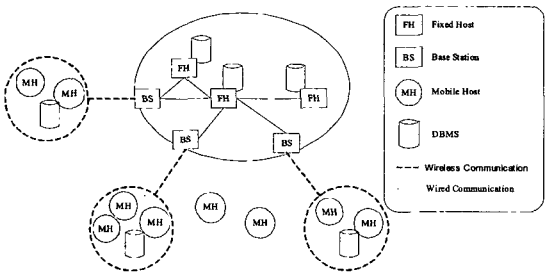
참고문헌 [14]에서는 타임스탬프 구간을 동적으로 재조정하여 직렬화 순서를 만족시키는 기법을 소개하고 있다. 이 기법은 데이터 충돌의 종류에 따라 직렬화 순서를 재조정하여 불필요한 재시작을 줄일 수 있다. 하지만, 이동환경이 아닌 중앙 집중형 실시간 환경을 고려하여 제안된 것이므로 방송 환경의 특성을 충분히 반영하지 못하고 있다.

이동 데이터베이스 시스템을 위한 동시성 제어 방법으로 제안하는 기법의 주요 특징은 참고문헌 [14]에서 사용된 직렬화 순서 재조정 방법을 이동 데이터베이스 시스템 환경에 적용하여 기존 방법의 문제점을 개선하고자 하는 것이다. 전체적인 제어 절차는 참고문헌 [17]에서의 방법을 근간으로 하고 있으며 직렬화 순서의 동적 재조정을 통하여 불필요한 이동 트랜잭션의 취소가 발생하지 않도록 하고 있다.

3. 이동 데이터베이스를 위한 동시성 제어 기법

3.1 이동 데이터베이스 시스템 모델

본 논문에서 제안하는 동시성 제어 기법은 그림 1과 같은 이동 컴퓨팅 환경을 기반으로 구축된 이동 데이터베이스 시스템을 고려한다. 데이터베이스는 유선 네트워크에 연결된 고정 호스트(Fixed Host)에서 관리되며 각 이동 클라이언트(Mobile Host)에서 요구하는 데이터 항목들을 방송 기법을 사용하여 무선 네트워크를 통하여 주기적으로 이동 클라이언트로 전달한다[1,6]. 이때 데이터베이스 서버(Fixed Host)에서 수행 완료된 트랜잭션들의 정보들도 함께 전달되어 이동 클라이언트에서의 부분 검증 작업에 사용된다. 제안하는 동시성 제어 기법은 이동 클라이언트에서 수행되는 이동 트랜잭션들이 쓰기 연산이 포함되는 것을 고려한다.



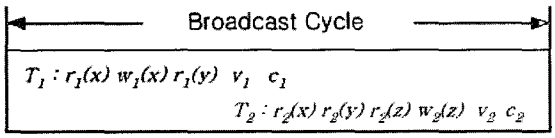
〈그림 1〉 이동 컴퓨팅 환경

낙관적 동시성 제어 기법에서는 트랜잭션의 수행이 읽기 단계 → 검증 단계 → 완료 단계의 세 단계 작업을 통하여 이루어진다. 트랜잭션의 읽기 단계에서는 각 트랜잭션들이 지역 버퍼(Local Buffer)를 이용하여 아무런 제약 없이 독립적으로 읽기/쓰기 연산을 수행하게 된다. 트랜잭션의 읽기 단계 작업이 완료되면 트랜잭션의 검증 작업을 거쳐 직렬화 순서에 위배되지 않으면 트랜잭션을 완료(Commit)하게 된다. 완료된 트랜잭션의 데이터 변경 내용은 이때 데이터베이스에 반영된다.

데이터 일관성 유지를 위한 검증 작업의 방법으로는 최근 완료된 트랜잭션들이 읽기/쓰기 작업을 한 데이터 항목들과의 충돌을 고려하는 역방향 검증(Backward Validation) 방법과 현재 활성중인(Active) 쓰기 단계의 트랜잭션들이 읽기/쓰기 작업을 한 데이터 항목들과의 충돌을 고려하는 순방향 검증(Forward Validation) 방법이 있다. 이동 데이터베이스 시스템 환경에서는 데이터베이스 서버가 검증 작업 과정에서 각 이동 클라이언트들의 트랜잭션 수행 상태를 알 수 없으므로 실행중인 트랜잭션들과의 충돌 여부를 검증하는 순방향 검증은 사용될 수 없고, 이미 완료된 트랜잭션들의 읽기/쓰기 작업을 한 데이터 항목들과의 충돌 해결을 수행하는 역방향 검증이 보다 적절하게 사용될 수 있다.

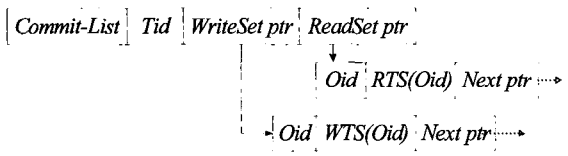
3.2 OCC/DTA 프로토콜

제안하는 OCC/DTA 프로토콜에서는 낙관적 동시성 제어 기법을 기반으로 이동 트랜잭션의 읽기 단계의 연산은 각 이동 클라이언트에서 수행되고 검증 단계와 완료 단계의 작업은 중앙의 데이터베이스 서버에서 수행된다. 검증 작업을 거쳐 완료 단계의 작업이 수행되면 갱신된 데이터는 최종적으로 서버의 데이터베이스에 반영된다. 하지만 이 과정에서 충돌이 예상되는 이동 트랜잭션의 수행 정보들이 서버로 전송된다면 서버로의 통신량이 불필요하게 증가하게 되고 또한 서버의 부하도 증가하게 된다. 따라서 제안하는 기법에서는 데이터베이스 서버로부터 매 방송 주기마다 전송되는 완료된 트랜잭션들의 정보를 이용하여 이동 클라이언트에서 트랜잭션의 부분 검증 작업을 수행하고 취소가 될 트랜잭션들은 서버로의 전송 없이 이동 클라이언트에서 독자적으로 취소함으로써 불필요한 통신 부하를 제거한다.



〈그림 2〉 수행 기록의 예

또한 트랜잭션들의 직렬화 순서를 트랜잭션의 검증 단계 작업의 시점에 따라 정하지 않고 데이터 일관성을 유지하면서 직렬화 순서를 동적으로 재조정함으로써 트랜잭션의 완료율을 높인다. 그림 2와 같은 이동 트랜잭션의 수행 기록에서 T_2가 T_1이후에 검증 작업을 수행하더라도 T_2는 이전 방송 주기에서 전송된 데이터 항목 x를 사용하였으므로 직렬화 순서는 T_2 -> T_1 가 될 것이다. 이와 같이 이동 트랜잭션들의 직렬화 순서는 각 이동 트랜잭션들의 사용한 데이터 항목들에 따라 검증 작업의 순서와 관계없이 직렬화 순서를 조정할 수 있다.



〈그림 3〉 Commit-List 자료 구조

이를 위하여 서버는 그림 3과 같은 구조로 이전 방송 주기 동안 성공적으로 완료한 트랜잭션들이 사용한 데이터 항목들을 기록한다. 이 리스트는 완료된 트랜잭션 번호(T_{id})와 완료된 트랜잭션에 의해 갱신된 모든 데이터 항목의 고유번호(O_{id})와 쓰기 타임스탬프($WTS(O_{id})$) 값을 포함하는 *WriteSet* 그리고 트랜잭션에 의해 읽기 연산을 수행한 데이터 항목의 고유번호(O_{id})와 읽기 타임스탬프($RTS(O_{id})$) 값을 포함하는 *ReadSet*으로 구성된다. 데이터 충돌이 발생하여 취소되어야 하는 트랜잭션은 *Abort-List*에 트랜잭션 번호(T_{id})를 기입하여 방송 시점에 *Commit-List*와 함께 전송된다. 이동 클라이언트들은 이동 트랜잭션의 읽기/쓰기 연산의 정보와 서버로부터 전송된 *Commit-List*의 *WriteSet* 및 *ReadSet*을 이용하여 이동 트랜잭션들의 타임스탬프 구간을 재조정하게 된다.

읽기 전용 트랜잭션을 고려하는 기법이나 [17]에서 소개한 FBOCC 기법의 경우에는 갱신된 데이터 항목에 대한 검증 정보만을 전송한다. 이에 비해 제안한 기법에서는 직렬화 순서의 재조정을 위하여 읽기 연산을 수행한 데이터 항목에 대한 정보도 추가로 필요로 한다. 그림 3에서 보듯이 검증 정보에 추가되는 데이터 항목의 정보는 해당 데이터 항목의 읽기 타임스탬프이다. 따라서 제안한 기법의 검증 정보는 다른 기법에 비해 *(완료된 트랜잭션이 읽기 연산을 수행한 데이터 항목의 수) * (타임스탬프 크기)* 만큼의 추가 정보를 요구한다.

3.3 서버에서의 전역 검증

서버는 기본적으로 두 가지 기능을 수행한다. 첫 번째, 모든 데이터 항목의 가장 최근 값과 이전 주기에 완료된 트랜잭션의 검증 정보를 방송한다. 두 번째, 각 클라이언트에서 전송되어온 검증 요청에 대해 전역 검증을 실시한다. 전역 검증은 이동 클라이언트에서 요구한 트랜잭션(T_v)의 검증 단계(Validation Phase) 작업을 최종적으로 수행하여 해당 트랜잭션이 직렬 가능한지 여부를 판단한다. 서버에서 전역 검증을 하는 이유는 아직 방송되지 않은 완료된 트랜잭션들과의 데이터 충돌을 검사하여 최종적인 검증을 하기 위함이다.

이동 트랜잭션이 읽기 연산으로만 이루어져 있으면 전역 검증 없이 이동 클라이언트에서 완료시킬 수 있지만, 하지만, 쓰기 연산이 포함되어 있으면 다른 이동 클라이언트에서 수행된 이동 트랜잭션들과의 검증이 필요하므로 서버에서의 전역 검증이 필요하게 된다. 최종적으로 전역 검증이 성공적으로 수행되면 이동 트랜잭션의 최종 타임스탬프는 $LowerBound(T_v)$ 값으로

부여하고 부여된 타임스탬프 값을 이용하여 이동 트랜잭션이 사용한 데이터 항목들의 읽기 타임스탬프와 쓰기 타임스탬프를 변경한다.

전역 검증을 위한 알고리즘은 아래와 같다. *UB(Tv)* 와 *LB(Tv)* 는 각각 검증되는 트랜잭션의 최대/최소 타임스탬프 값을 의미하고, *ReadSet(Tc)* 과 *WruteSet(Tc)*은 최근 완료된 트랜잭션들이 읽기/쓰기 연산을 한 데이터 항목들의 집합이다. *RTS(Di)* 와 *WTS(Di)* 는 해당 데이터 항목들의 읽기/쓰기 타임스탬프 값이고, *RS(Tv)* 와 *WS(Tv)* 는 검증되는 트랜잭션이 읽기/쓰기 연산을 한 데이터 항목들의 집합이다.

```
Global-Validation(Tv)
    foreach Di (i=1,2,3, ··· , m) in RS(Tv) {
        if Di in WruteSet(Tc) then set LB(Tv)
        = max(LB(Tv), WTS(Di) + 1)
                        if LB(Tv) = UB(Tv)
                        then abort Tv }
    foreach Di (i=1,2,3, ··· , m) in WS(Tv) {
        if Di in ReadSet(Tc) then set UB(Tv)
        = min(UB(Tv), RTS(Di) + 1)
                        if LB(Tv) = UB(Tv)
                        then abort Tv }
    commit Tv
```

3.4 클라이언트의 부분검증

제안하는 기법에서는 이동 트랜잭션의 수행 초기 단계에 [14]에서와 같이 연산을 시작하는 모든 이동 트랜잭션에게 $[0, \infty]$의 타임스탬프 구간을 부여하고 읽기 단계에서 읽기/쓰기 연산을 수행하는 과정에서 부분 검증을 하여 타임스탬프 구간을 동적으로 재조정한다. 타임스탬프 구간의 조정은 이동 트랜잭션의 읽기/쓰기 연산 과정에서 아래와 같은 세 가지 종류의 데이터 충돌에 따른 직렬화 순서의 선행 관계를 반영하게 된다. 세 가지 종류의 데이터 충돌은 다음과 같다. T_c는 최근 완료된 트랜잭션을 의미하고 T_v는 부분 검증이 이루어지는 이동 트랜잭션을 의미한다.

읽기-쓰기 충돌 ($ReadSet(T_c) \cap Active\ Write\ Set(T_v) \neq Null$)

현재 실행 트랜잭션에 의해 완료 트랜잭션의 연산이 영향을 받지 않아야 하므로 완료 트랜잭션의 직렬화 순서가 실행 트랜잭션에 선행하도록 $T_c \rightarrow T_v$로 타임스탬프 구간을 조정한다.

쓰기-읽기 충돌 ($UpdateSet(T_c) \cap ActiveReadSet(T_v) \neq Null$)

이 경우는 완료 트랜잭션에 의해 수행 중인 트랜잭션이 영향을 받지 않도록 $T_v \rightarrow T_c$로 타임스탬프 구간을 조정한다.

쓰기-쓰기 충돌 ($UpdateSet(T_c) \cap Active\ Write\ Set(T_v) \neq Null$)

이 경우는 수행 중인 트랜잭션에 의해 갱신된 데이터 항목을 완료 트랜잭션이 다시 쓰는 직렬화 순서를 갖지 않도록 $T_c \rightarrow T_v$로 타임스탬프 구간을 조정한다.

타임스탬프 구간의 조정은 상한 값(*UpperBound*)과 하한 값(*LowerBound*)을 이용한다. 완료 트랜잭션이 선행해야 하는 경우에는 구간의 하한 값을 데이터 항목의 타임스탬프 값보다 크게 조정하고 실행 트랜잭션이 선행해야 하는 경우에는 구간의 상한 값을 해당 항목의 타임스탬프 값보다 작게 조정한다. 상한 값과 하한 값이 같거나 하한 값이 커지는 경우에는 더 이상 타임스탬프 구간을 조정할 수 없는 경우이므로 트랜잭션을 취소한다.

이러한 부분 검증을 통해서 직렬화 순서를 조정함으로써 트랜잭션의 처리율을 높이고 직렬화 순서를 재조정하지 못하는 트랜잭션을 클라이언트에서 취소시킴으로써 검증을 위해 서

버로의 전송을 최소화할 수 있다. 또한, 전역 검증을 위해 클라이언트는 트랜잭션이 성공적으로 완료되면 실행 중에 읽거나 갱신한 데이터 항목과 타임스탬프 구간을 서버로 전송한다. 이동 클라이언트에서 수행되는 이동 트랜잭션의 읽기/쓰기 연산은 아래와 같은 알고리즘에 따라 부분 검증 작업을 수행하게 된다.

```
Read (d)
  foreach d in ReadSet of commit-list {
      set LB(T) = max (LB(T), WTS(d) + 1)
      if LB(T) = UB(T) then abort T }

Write (d)
  foreach d in WriteSet of commit-list {
      set UB(T) = min (UB(T), RTS(d) + 1)
      if LB(T) = UB(T) then abort T }
```

예제. 표 1의 수행 기록에 대하여 제안된 기법을 적용한 트랜잭션의 동작을 기술하면 아래와 같다. 각 이동 클라이언트는 한 번에 하나의 트랜잭션을 수행하는 것으로 가정한다. *T1, T2* 트랜잭션은 서버에서 완료되는 트랜잭션을 의미하고, *T3, T4* 트랜잭션은 서로 다른 이동 클라이언트에서 수행되는 트랜잭션들을 의미한다. $w_4(z)$와 $r_4(q)$ 사이 시점에 검증 정보를 방송한다고 가정하고, 각 데이터 항목의 초기 타임스탬프 값은 *RTS*(*x*)=6, *WTS*(*x*)=4, *RTS*(*y*)=7, *WTS*(*y*)=7, *RTS*(*z*)=9, *WTS*(*z*)=8, *RTS*(*q*)=10, *WTS*(*q*)=10로 가정한다.

〈표 1〉 수행 기록의 예

	Time
Client	··· r3(x) r3(y) w3(x) r4(z) w4(z) || r4(q) w3(y) ···
Server	··· r1(x) w1(q) r2(z) w2(z) broadcast

이동 클라이언트 트랜잭션의 읽기 단계의 수행의 결과 트랜잭션 *T3, T4*의 타임스탬프는 3.4절의 알고리즘에 따라 각각 *LowerBound(T3)* = *7, LowerBound(T4)* = *9*로 조정된다. 타임스탬프의 *UpperBound*는 읽기 단계에서는 조정되지 않는다. 각 이동 클라이언트의 트랜잭션들이 수행되는 동안 서버에서는 트랜잭션 *T1, T2*의 전역 검증을 수행하고, 완료된 T_1, T_2에게 각각 *14, 15*의 타임스탬프를 부여하고 변경된 정보를 방송한다. $w_4(z)$의 수행 후 방송된 정보를 이용하여 트랜잭션은 부분 검증을 수행한다. *UpdateSet*의 *z*에 의해 *T4*의 *Upperbound*와 *LowerBound*는 15로 같아지므로 *T4*는 취소된다. *ReadSet*의 *x*에 의해 *T3*의 *LowerBound*는 *14*로 조정되고 수행을 계속한다.

3.5 정확성 검증

제안한 기법의 정확성을 증명하기 위해서는 제안 기법에 의해 생성되는 모든 수행 기록(*H*)의 직렬화 그래프(*SG*(*H*)가 항상 비순환적(Acyclic)임을 증명하면 된다.

보조정리 1. *Ti*와 *Tj*를 제안된 기법에 의해 생성된 수행 기록 *H*에서 완료된 트랜잭션들이고 *SG*(*H*)에 *Ti* -> *Tj* 가 존재하면 항상 *TS*(*Ti*) < *TS*(*Tj*) 이다.

증명 : *SG*(*H*)에 *Ti* -> *Tj* 가 존재한다는 것은 데이터 항목 *x* 에 대해 다음과 같은 경우에 해당하는 데이터 충돌이 발생한 것이다.

case1) *Ti*가 *Tj*의 검증 단계 작업 이전에 완료된 경우

a) $r_i(x) \rightarrow w_j(x)$

T_j가 검증 단계에 들어서면 $w_j(x)$에 의해 트랜잭션의 타임스탬프 구간 하한 값이 *Lower Bound*(*Tj*) = *Max*(*LowerBound*(*Tj*), *RTS*(*x*))에 의해 *RTS*(*x*)보다 크거나 같은 값을 갖게 된다. 따라서 *TS*(*Ti*) ≤ *RTS*(*x*) ≤ *TS*(T_j)에 의해 *TS*(*Ti*) <

*TS(Tj)*가 된다.

b) $w_i(x) \rightarrow r_j(x)$

*Tj*가 데이터 항목 *x*를 읽기 전에 *Ti*가 완료된 경우이다. 이는 트랜잭션의 타임스탬프 구간 하한 값이 *LowerBound(Tj) = Max(LowerBound(Tj), WTS(x))*에 의해 *WTS(x)*보다 큰 값을 갖는 경우이다. 또한 *TS(Ti)*는 *WTS(x)*보다 큰 값을 가질 수 없다. 따라서 *TS(Ti) < TS(Tj)*이다.

case2) *Tj*가 *Ti*의 검증 단계 작업 이전에 완료된 경우

a) $r_i(x) \rightarrow w_j(x)$

*Ti*가 검증 단계에 들어서면 *UB(Ti) = Min(UB(Ti), WTS(x))*에 의해 *TS(Ti)*는 *WTS(x)*에 의해 *WTS(x)*보다 작은 값을 갖는다. 따라서 항상 *TS(Ti) < TS(Tj)*
가 된다.

b) $w_i(x) \rightarrow r_j(x)$

*LowerBound(Tj)*가 *RTS(x)*보다 작은 경우에는 트랜잭션 *Ti*가 취소되므로 직렬화 순서 *Ti -> Tj* 는 *SG(H)*에 존재하지 않는다. ■

정리 1. 제안된 기법에 의해 생성된 완료된 수행기록 *H*는 직렬가능(Serializable)하다.

증명 : *H*를 제안된 기법에 의해 생성된 수행기록이라고 하고, *SG(H)*가 *T1 -> T2 -> T3 ⋯ ->* *T1*의 순환 구조를 갖는다고 가정하자. 이는 보조정리 1에 의해 *TS(T1) < TS(T2) < TS(T3) < ⋯ < TS(T1)*이 되고 *TS(T1) < TS(T1)*의 결과를 생성하므로 모순이 된다. 따라서, *SG(H)*에는 순환(Cycle)이 존재할 수 없기 때문에 제안된 기법에 의해 생성된 수행기록은 항상 직렬가능하다. ■

4. 성능평가

이 장에서는 제안한 기법의 성능 평가를 위하여 참고문헌 [17]의 FBOCC기법과 트랜잭션 처리율을 비교 분석한 모의 실험한 결과를 기술한다. FBOCC는 제안한 기법과 마찬가지로 낙관적 동시성 제어 기법에 기반을 두어 방송 환경의 요구사항을 만족시키도록 설계된 것으로 데이터 충돌을 감지하는 즉시 실행중인 트랜잭션을 취소시키는 것이 다른 점이다.

성능평가에 사용된 인수들은 다양한 충돌 환경을 나타내기 위하여 표 2와 같이 설정하였다. 읽기 연산 빈도는 트랜잭션내의 연산 중 읽기 연산의 비율을 나타낸 것이다. 이는 쓰기 연산이 시스템 성능에 미치는 영향을 고려하기 위해 사용하였다. 데이터 접근 분포는 트랜잭션이 데이터에 접근하는 패턴을 나타낸 것으로 데이터 충돌 발생 빈도에 따른 성능차이를 고려하기 위해 사용하였다. 트랜잭션의 크기는 하나의 트랜잭션이 가지는 평균 연산(읽기 혹은 쓰기)의 수를 나타내고, 크기를 변화시킴으로서 다양한 시스템 부하를 가진 환경을 실험하였다.

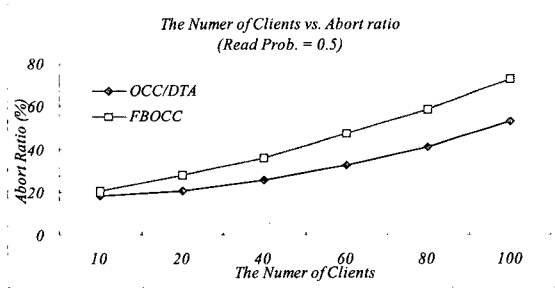
〈표 2〉 시뮬레이션 인수

실험 파라미터	값	설 명
데이터베이스 크기	2000 pages	데이터베이스의 페이지 수
트랜잭션의 읽기 연산 빈도	0.5~1.0	트랜잭션의 연산중에 읽기 연산이 차지하는 비율
트랜잭션의 데이터 접근 분포	0.2~1.0	트랜잭션이 접근하는 데이터의 분포(1.0이면 균등 분포)
트랜잭션의 평균 연산 수	4~15	하나의 트랜잭션이 가지는 평균 연산의 개수
이동 클라이언트의 수	10~100	시스템에 존재하는 이동 클라이언트의 수

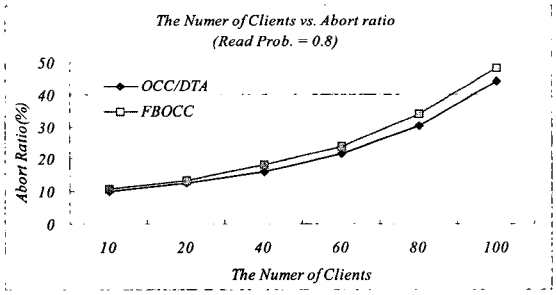
본 실험은 Pentium4 3GHz 512MB RAM환경에서 CSIM 라이브러리를 이용하여 수행하였다. 파라미터 변경을 통해 데이터 충돌 발생 빈도를 다양하게 조정하여 그것이 성능에 미치는 영향을 측정하였다. 성능 비교에 사용되는 척도로는 데이터 충돌 해결에 결과로 나타나는 취소된 트랜잭션의 비율을 사용하였다. 제안한 기법의 성능적인 특성을 알아보기 위하여 실행시간이나 네트워크 대역폭과 같은 특성들은 고려하지 않았다.

그림 4와 그림 5는 읽기/쓰기 비율에 따른 트랜잭션의 취소 비율을 보여준다(그래프에서 OCC/DTA는 제안한 프로토콜을, FBOCC는 [17]의 프로토콜을 의미한다.). 모의실험에서 트랜잭션 수행에 따른 데이터 충돌의 빈도를 조절하기 위하여 읽기 연산의 확률이 0.5와 0.8의 경우를 설정하여 수행하였다. 이동 클라이언트의 수가 많아지면 트랜잭션의 수가 많아져 데이터 충돌의 가능성이 높아진다. 또한, 쓰기 연산의 비율이 높기 때문에 충돌의 가능성은 더욱 높아진다. 이렇게 충돌이 빈번히 일어나는 환경에서 제안한 기법은 그림 4에서와 같이 최대 20%이상의 우수한 성능을 보이고 있다. 그림 5에서와 같이 읽기 확률이 높아서 데이터 충돌이 적은 경우에는 성능의 차가 거의 나타나지 않는다. 반면에 읽기 확률이 적은 경우에는 제안한 기법이 좋은 성능을 보인다. 이는 기존의 기법에서는 각 이동 클라이언트들의 트랜잭션들 또는 서버에서 수행되는 트랜잭션들 사이에서 발생하는 모든 데이터의 충돌에 대해 실행중인 트랜잭션을 취소시키는데 반하여 제안한 기법은 트랜잭션의 타임스탬프를 동적으로 재구성하여 처리율을 높이기 때문이다.

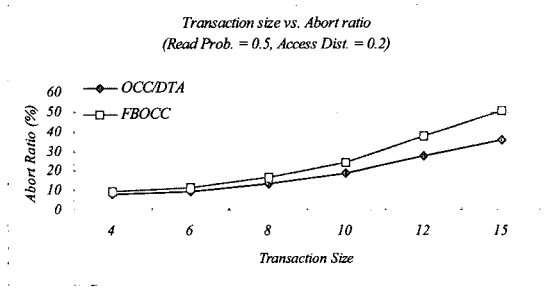
그림 6과 그림 7도 비슷한 이유로 설명할 수



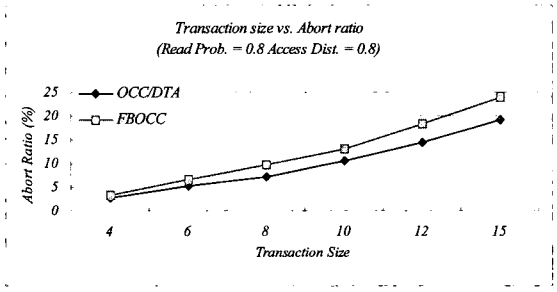
〈그림 4〉 클라이언트 수에 따른 Abort Ratio
(읽기 비율 = 0.5)



〈그림 5〉 클라이언트 수에 따른 Abort Ratio
(읽기 비율 = 0.8)



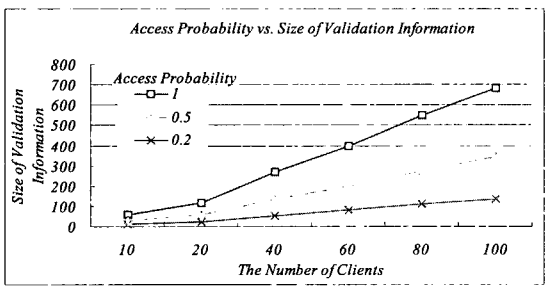
〈그림 6〉 트랜잭션 크기에 따른 Abort Ratio
(읽기 비율 = 0.5)



〈그림 7〉 트랜잭션 크기에 따른 Abort Ratio
(읽기 비율 = 0.5)

있다. 트랜잭션의 크기가 커질수록 트랜잭션이 가지는 연산의 수가 많아진다. 그림 6과 같이 쓰기 연산의 빈도가 높고 특정 데이터를 집중적으로 요구할 때 데이터 충돌 가능성은 더욱 높아진다. 따라서 직렬화 순서를 재조정하여 충돌을 해결하는 제안한 기법이 최대 20% 이상의 우수한 성능을 보이는 반면 그림 7과 같이 읽기 확률이 높고 고른 데이터 항목에 대해 접근 요구가 발생하면 데이터 충돌 가능성이 상대적으로 낮아 두 기법이 비슷한 성능을 보인다.

그림 8에서는 이동 클라이언트의 수와 데이터 접근 빈도에 따른 검증정보의 크기를 나타내고 있다. FBOCC에서는 클라이언트 트랜잭션을 취소하여 충돌을 해결하는 정책을 사용하고 있기 때문에 검증에 필요한 정보는 이전 주기에 완료된 트랜잭션에 의해 접근되거나 갱신된 데이터 항목의 *id*만을 필요로 한다. 제안한 기법에서는 데이터 항목의 타임스탬프를 이용하여 트랜잭션의 타임스탬프 구간을 조정해야 하기 때문에 데이터 항목의 *id*이외에 읽기 타임스탬프와 쓰기 타임스탬프 값을 필요로 한다. 실험 결과는 트랜잭션의 데이터 접근 빈도에 따른 검증정보의 크기를 나타낸다. 접근 빈도가 1.0인 경우에는 트랜잭션이 거의 모든 데이터 항목에 대해 읽기나 쓰기 요청을 하기 때문에 검증정보의 크기가 커짐을 보이고 있다. 하지만, 검증 정보의 크기는 최대 1KB 정도이므로 검증 정보의 크기는 통신 부하에 큰 영향을 주지 않을 것이다.



〈그림 8〉 클라이언트 수에 따른 검증 정보의 크기

5. 결 론

본 논문에서는 낙관적 기법에 근거하고 타임스탬프 구간을 이용한 이동 컴퓨팅 환경에서의 새로운 동시성 제어 기법을 제안하였다. 제안한 프로토콜은 데이터 충돌이 발생한 트랜잭션을 무조건 취소시키는 대신 타임스탬프 구간을 적절히 조정해 트랜잭션을 계속 진행시킴으로써 트랜잭션의 직렬성을 유지함과 동시에 트랜잭션의 완료율을 높인다. 또한, 이동 클라이언트에서 충돌 해결이 불가능한 트랜잭션을 조기에 취소시킴으로써 클라이언트의 자원의 낭비를 막고 클라이언트에서 서버로의 데이터 전송을 줄이고 있다. 성능 평가를 위해 낙관적 방법에 기반을 두고 전통적인 충돌 해결 기법을 사용하는 FBOCC기법과 제안한 기법에 대하여 트랜잭션의 크기, 이동 클라이언트의 수, 데이터 집중도, 읽기 연산 비율 등을 비교 실험하였다. 실험결과는 직렬화 순서를 조정하여 불필요한 재시작을 감소시키기 때문에 직렬화 재조정이 없는 전통적 기법보다 제안한 기법이 우수한 성능을 나타냄을 보여주었다.

참 고 문 헌

[1] Imielinski, T. and Badrinath, B. R., "Mobile Wireless Computing: Challenges in Datamanagement.", Communications of the ACM 37 (10), pp.18-28, 1994

[2] Barbara, D., "Mobile Computing and Databases-A Survey", IEEE Trnascation of Knowledge Data Engineering, volume 11 issue 1, pp.108-117, 1999

[3] Acharya, S., Alonso, R., Franklin, M. and Zdonik, S., "Broadcast Disks: Data Management for Asymmetric Communication Environments.", Proc. ACM SIGMOD 1993 Int. Conf. on Management of Data, pp.199-210, 1993

[4] Bernstein, P.A, Hadzilacos, V. and Goodman, N., "Concurrency Control and Recovery in Database System", Addison-Wesley 1987

[5] Acharya, S., Franklin, M. and Zdonik, S., "Disseminating Updates on Broadcast Disk" Proc. 22nd VLDB Conference, pp.354-365, 1996

[6] Acharya, S., Franklin, M. and Zdonik, S., "Balancing Push and Pull for Data Broadcast" Proc. ACM SIGMOD, pp.183-194, 1997

[7] Acharya, S., Franklin, M. and Zdonik, S., "Prefetching from a Broadcast Disk" Proc. IEEE Conference on Data Engineering, pp.276-285, 1996

[8] Das, A. and Kai, K. Y., "Tradeoff between Client and Server Transaction Validation in Mobile Environment", International Symposium on Database Engineering and Application, pp.265-272, 2001

[9] Shanmugasundaram, J., Nithrakashyap, A., Sivasankaran, R. and Romamritham K., "Efficient Concurrency Control for Bdisk Environments", ACM SIGMOD International Conf. on Management of data, pp.85-86, 1999

[10] Pitorura, E. and Chrysanthis, P. K., "Exploiting Version for Handling Updates in Broadcast Disk", Proc. 25th VLDB Conference, pp.114-125, 1999

[11] Herman, G., Gopal, G., Lee, K. C. and Weinreb, A., "The Datacycle Architecture for Very High Throughput Database System", Proc. ACM SIGMOD Conference, pp.97-103, 1987

[12] Barbara, D., "Certification Reports : Supporting Transaction in Wireless System", Proc. 17th International Conference on Distributed Computing Systems, pp.466-473, 1997

[13] Victor, C. S., Kwok, wa Lam and Son, S. H., "Concurrency Control Using Timestamp Ordering in Broadcast Environments", The Computer Journal, vol.45 no.4 pp.410-422, 2002

[14] Lee, J. and Son, S. H., "Using Dynamic Adjustment of Serialization Order for Real-Time Database Systems", Proc. 14th Real-Time Systems Symp., pp.66-75, 1993

[15] Pitoura, E., "Supporting Read-Only Transactions in Wireless Broadcasting", Proc. DEXA Workshop on Mobility in Database and Distributed Systems, pp.428-433, 1998

[16] Mei-Wai, Au, et. al., "Concurrency Control for Mobile Systems with Data Broadcast", Journal of Interconnection Networks, pp. 253-267, 2001

[17] Victor, C. S., et. al., "Efficient Validation of Mobile Transactions in Wireless Environments", The Journal of Systems and Software, pp.183-194, 2004

◐ 저 자 소 개 ◑

김 대 호 (Dae-Ho Kim)

1997년 경희대학교 전자계산공학과 졸업(학사)
1999년 경희대학교 일반대학원 전자계산공학과(공학석사)
2005년 경희대학교 일반대학원 컴퓨터공학과 박사
관심분야 : 실시간 데이터베이스, 이동 데이터베이스, 동시성 제어 등
E-mail : techwing@khu.ac.kr

정 병 수 (Byeong-Soo Jeong)

1983년 서울대학교 전자계산기공학과 학사
1985년 한국과학기술원 전산학과 석사
1995년 Georgia Institute of Technology, College of Computing 박사
1985년 ~ 1989년 8월 한국데이터통신(주) 정보통신연구소 선임연구원
1996년 ~ 현재 경희대학교 전자정보대학(컴퓨터공학전공) 교수
관심분야 : 이동 데이터베이스, 실시간 데이터베이스, 데이터 마이닝 등
E-mail : jeong@khu.ac.kr

이 영 구 (YoungKoo Lee)

1992년 한국과학기술원 전산학과 학사
1994년 한국과학기술원 전산학과 석사
2002년 한국과학기술원 전산학과 박사
2002. 3 ~ 2004. 2 한국과학기술원 박사 후 연구원
2002. 9 ~ 2004. 2 미국 University of Illinois at Urbana-Champaign, Postdoctoral Research Associate,
2004 ~ 현재 경희대학교 전자정보대학 조교수
관심분야: 유비쿼터스 데이터베이스, 데이터 웨어하우징, 데이터 마이닝, Bioinformatics
E-mail : yklee@khu.ac.kr