무선 브로드캐스트 환경에서 편향된 엑세스 패턴을 가진 모바일 트랜잭션을 위한 효과적인 동시성 제어 기법

(An Energy-Efficient Concurrency Control Method for Mobile Transactions with Skewed Data Access Patterns in Wireless Broadcast Environments)

정 성 원 [†] 박 성 근 [††] 최 근 하 [†††]
(Sungwon Jung) (Sunggeun Park) (Keunha Choi)

요 약 브로드캐스트는 하나 또는 여러 개의 채널을 이용해서 다수의 모바일 클라이언트들이 빈번하게 필요로 하는 데이타를 효과적으로 전송하기 위한 방법 중의 하나이다. 무선 브로드캐스트 환경에서는 채널의 상향 대역폭의 한계로 인해 기존의 동시성 제어 기법은 적합하지 않다. 무선 브로드캐스트 환경에서 서버는 종종 모바일 클라이언트의 접근 패턴을 고려하여 편향된 접근 빈도를 갖는 서로 다른 데이타 아이템을 브로드캐스트 하기도 한다. 무선 브로드캐스트 환경에서 모바일 트랜잭션을 위한 기존의 제안된 동시성 제어 기법들은 일정한 데이타 접근 패턴에 중점을 두고 있다. 하지만, 기존의 기법들은 데이타의 접근 패턴이 일정하지 않고 편향된 경우에는 오히려 심각한 성능 저하를 발생시킨다. 편향된 데이타 접근 패턴을 갖는 갱신 모바일 트랜잭션들은 높은 접근 빈도를 같은 데이타를 동시에 접근하고자 하는 다른 모바일 트랜잭션들 간의 충돌로 인해 실행이 취소되고 재실행될 것이다. 본 논문에서는 일정한 데이타 접근 패턴뿐만 아니라 편향된 데이타 접근 패턴을 갖는 모바일 트랜잭션을 위한 에너지 효율적인 동시성 제어 기법을 제안한다. 본 논문에서는 임의 백오프 기법을 통해 갱신 모바일 트랜잭션의 빈번한 실행 취소와 재실행을 방지한다. 우리는 기존의 동시성 제어 기법과의 비교를 통해 본 논문에서 제안하는 기법을 심층적으로 분석한다. 또한 실험을 통해 기존의 기법들에 비해 평균 접근 시간, 상향 및 하향 통신 대역폭의 사용량이 현저히 줄어드는 것을 보임으로써 제안하는 기법의 성능을 검증한다.

키워드 : 모바일 컴퓨팅, 모바일 데이타베이스, 모바일 트랜잭션 관리, 모바일 동시성 제어, 무선 브로드캐스트 환경, 브로드캐스트 디스크

Abstract Broadcast has been often used to disseminate the frequently requested data efficiently to a large volume of mobile clients over a single or multiple channels. Conventional concurrency control protocols for mobile transactions are not suitable for the wireless broadcast environments due to the limited bandwidth of the up-link communication channel. In wireless broadcast environments, the server often broadcast different data items with different frequency to incorporate the data access patterns of mobile transactions. The previously proposed concurrency control protocols for mobile transactions in wireless broadcast environments are focused on the mobile transactions with uniform data access patterns. However, these protocols perform poorly when the data access pattern of update mobile transaction are not uniform but skewed. The update mobile transactions with skewed data access patterns will be frequently aborted and restarted due to the update conflict of the same data items with a high access frequency. In this paper, we propose an energy-efficient concurrency control protocol for mobile transactions with skewed data access as well as uniform data access patterns. Our

· 본 연구는 한국과학재단 목적기초연구(R01-2003-000-10197-0)지원과 서강대학교 산업기술연구소의 지원을 받아 수행되었음
† 종신회원 : 서강대학교 컴퓨터학과 교수
jungsung@sogang.ac.kr
†† 비 회 원 : SK INNOACE Terminal solution team 연구원
sgpark@innoace.com
††† 학생회원 : 서강대학교 컴퓨터학과
purymul@sogang.ac.kr
논문접수 : 2005년 2월 18일
심사완료 : 2005년 10월 6일

protocol use a random back-off technique to avoid the frequent abort and restart of update mobile transactions. We present in-depth experimental analysis of our method by comparing it with existing concurrency control protocols. Our performance analysis show that it significantly decrease the average response time, the amount of upstream and downstream bandwidth usage over existing protocols.

Key words : Mobile Computing, Mobile Databases, Mobile Transaction Management, Mobile Concurrency Control, Wireless Broadcast Environments, Broadcast Disks

1. 서 론

최근 휴대폰이나 PDA와 같은 컴퓨터 하드웨어와 블루투스나 IEEE 802.11과 같은 무선 네트워크 기술의 급격한 발달은 모바일 컴퓨팅을 점차 가능케 하고 있다. 가까운 미래에는 수십만의 사용자들이 배터리를 전원으로 사용하는 모바일 기기들을 가지고 언제 어디서나 다양한 종류의 서비스에 접근하게 될 것이다[1]. 모바일 기기는 무선 컴퓨팅 환경에서 통신비용을 줄이고 한정된 전원을 유지하고자 종종 접속을 끊기도 한다. 즉, 무선 환경에서는 방대한 양의 데이타를 무선 통신 네트워크상에서 좀 더 빠르고 적은 전력을 사용하도록 제공하는 것이 가장 중요한 부분이다.

그러나 현존하는 기술들은 물리적 제약 조건으로 인해 좁은 통신 대역폭, 잦은 접속 단절, 배터리 부족 등의 문제를 가진다. 이러한 문제들을 해결하기 위해 무선 데이타 브로드캐스팅을 이용한 데이타 전달 기법에 대한 많은 연구가 있었다[1-3]. 일반적으로 기존 시스템 모델에서는 클라이언트가 서버에게 명시적으로 데이타를 요구하고, 서버가 그에 대해서 응답하는 방식을 취하였다. 이러한 방식에서는 클라이언트가 상향 통신 대역폭을 이용해 요청을 보내야만 하고, 클라이언트의 수가 많아지면 서버에 과부하가 발생하여 데이타 아이템을 얻을 위한 지연 시간이 길어지게 되는 단점이 있었다. 반면 무선 데이타 브로드캐스트 모델에서는 데이타를 일정한 주기를 통해 반복적으로 브로드캐스팅을 함으로써 그와 같은 문제를 해결한다.

브로드캐스팅 방식은 클라이언트가 명시적으로 데이타를 요구하지 않아도 원하는 데이타를 수신할 수 있으므로 상향 통신 대역폭을 낭비하지 않고, 서버는 클라이언트의 수에 대한 확장성을 가지는 장점이 있다. 클라이언트의 수에 제한이 없다는 특징으로 인해 무선 데이타 브로드캐스트는 경매, 전자 입찰과 같은 전자 상거래 응용 분야와 주식 거래, 기상 정보, 교통 정보 방송과 같은 다양한 응용 분야에 적용되고 있다[4]. 예를 들어 비행 티켓 예매 서비스와 같은 응용 분야에서 모바일 고객은 비행 티켓을 모바일 기기를 통해 신용 카드로 구매한다. 이러한 응용 분야에서는 많은 사용자들이 동시에 하나의 아이템을 읽거나 갱신하는 상황이 발생하기 때문에 데이타를 일관성 있게 관리할 수 있어야 한다. 즉 갱신 작업이 발생하더라도 갱신이 발생한 데이타는 신속하고 일관성 있게 다음 브로드캐스트 주기에 모바일 클라이언트에게 브로드캐스팅 되어야 한다. 이 경우 만약 모바일 클라이언트가 동시에 여러 개의 브로드캐스트 주기 동안에 실행된다면 모바일 클라이언트는 서로 일관성이 없는 데이타를 접근하여 잘못된 결과를 발생시킬 수 있다. 그러므로 모바일 트랜잭션의 일관성을 보장하기 위해서 트랜잭션 단위의 동시성 제어 기법이 필요하다. 그러나 무선 데이타 브로드캐스트 환경에서는 기존의 전통적인 동시성 제어 기법을 그대로 적용할 수 없다[5]. 기존의 동시성 제어 기법은 모바일 클라이언트와 서버 사이의 많은 양방향 메시지 교환을 요구하는데, 일정한 주기를 통해 데이타를 브로드캐스트 하는 환경에서는 양방향 메시지 교환에 상당한 시간이 소모되고 이는 트랜잭션 처리 시간의 지연으로 이어지게 된다. 또한 모바일 클라이언트가 서버에게 메시지를 보내는 것은 한정된 배터리를 전원으로 모바일 클라이언트에게는 큰 부담이 된다. 그리고 무선 데이타 브로드캐스트 환경의 많은 응용 분야는 클라이언트의 수가 매우 많은데, 기존의 동시성 제어 기법에서는 서버가 모든 것을 담당하기 때문에 과부하를 받게 된다.

이런 문제를 해결하고자 무선 데이타 브로드캐스트 환경을 위한 트랜잭션 처리 기법이 많이 연구되어왔다. 기존의 기법들은 읽기 전용 트랜잭션에만 적용할 수 있는 기법들과 읽기 전용 트랜잭션과 갱신 트랜잭션 모두에 적용이 가능한 기법들로 분류할 수 있다. 읽기 전용 트랜잭션은 서버로 데이타를 보낼 필요가 없이 모바일 클라이언트가 자체적으로 검증하고 커미트할 수 있다. 트랜잭션의 대부분이 읽기 전용 트랜잭션인 것을 감안한다면 모바일 클라이언트와 서버와의 연계 없이 자체적으로 검증하고 커미트하는 것은 제한적인 통신 대역폭을 절약할 수 있게 된다. 읽기 전용 트랜잭션에 대해서는 이미 많은 연구가 진행되었다. Shanmugasundaram은 동시성 제어 기법의 정확성 기준(correctness criterion)을 약하게 하여 읽기 전용 트랜잭션이 서버와 일관성이 있는 데이타를 읽을 수 있도록 하는 기법을 제시하였고[6], Pitoura는 데이타와 함께 제어 정보를

브로드캐스팅 함으로써 모바일 클라이언트 측에서 읽기 전용 트랜잭션을 자체적으로 검증할 수 있도록 하는 기법을 제시했다[7]. 그와 관련해 하나의 아이템에 대한 다양한 버전을 제공하거나, 무효화 보고(invalidation report), 직렬화 그래프(serialization graph)와 같은 제어 정보를 제공하는 다양한 기법이 존재한다. 하지만 모바일 주식 거래, 티켓 예매, 경매 등과 같은 많은 응용분야에서는 모바일 클라이언트 측에서 읽기 전용 트랜잭션뿐만 아니라 갱신 트랜잭션도 발생하게 된다. 따라서 갱신 트랜잭션들을 효과적으로 처리할 수 있는 동시성 제어 기법이 필요하다. 갱신 트랜잭션에 대해서는 상대적으로 연구가 많이 이루어지지 않았다. 대표적으로 Lee는 전통적인 낙관적 동시성 제어기법을 변형하여 모바일 데이타 브로드캐스트 환경에서 사용할 수 있는 기법을 제시하였다[6]. 그러나 기존의 갱신 트랜잭션을 다룬 동시성 제어기법들은 모바일 클라이언트의 엑세스 패턴이 편향될 경우 반복적인 재실행(continuous restart)으로 인해 성능 저하를 발생시키게 된다. 이는 하나의 브로드캐스트 주기 동안 다수의 모바일 트랜잭션이 같은 데이타를 갱신하고자 할 때, 하나의 모바일 트랜잭션만이 선택적으로 수행되고 나머지 트랜잭션은 다음 주기에 재실행이 되어야 하기 때문이다. 이는 제한적인 모바일 클라이언트의 자원을 반복적으로 낭비하는 문제를 초래한다.

본 논문에서는 기존 연구들의 이와 같은 문제점을 파악하고 읽기 전용 트랜잭션과 갱신 트랜잭션 모두에 적용이 가능한, 무선 데이타 브로드캐스트 환경에 적합한 효과적인 동시성 제어 기법을 제안한다. 읽기 전용 트랜잭션은 클라이언트의 상향 통신 대역폭을 사용하지 않고 지역적으로 커미트 하도록 하고, 갱신 트랜잭션은 정방향, 역방향 검증을 기본으로 사용하여 서버와의 불필요한 통신 대역폭을 가능한 적게 사용한다. 그리고 Franaszek가 제안한 접근 불변성(access invariance)[8]을 이용하여 프리패칭할 데이타 아이템의 집합을 구성하여, 재실행시 클라이언트의 버퍼 유지 효과를 높여 재실행 시 응답시간을 최소화하도록 한다. 마지막으로 트랜잭션들의 엑세스 패턴이 편향될 경우 발생하게 되는 반복적인 재실행을 문제점으로 인식하고, 임의 백오프를 이용해 반복적인 재실행 문제를 해결한다. 즉, 서버 측에서 모바일 클라이언트들이 최종 검증을 위해 보내는 데이타를 바탕으로 각 데이타에 대한 경합 정도를 계산하고 모바일 트랜잭션들에게 제어 정보로 보내준다. 재실행되는 모바일 클라이언트는 이 정보를 통해 임의 백오프를 함으로써 반복적인 재실행을 피하고 클라이언트의 엑세스 패턴의 편향성에 유연하게 적응한다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2장에서는 우리의 무선 브로드캐스트 시스템 모델과 그 특성에 대해서 설명한다. 3장에서는 기존에 제안되었던 트랜잭션 처리 기법을 설명한다. 4장에서는 우리가 제안하는 트랜잭션 처리 기법을 소개한다. 5장에서는 제안된 기법의 성능을 분석한다. 마지막으로 6장에서는 논문의 결론을 논의하였다.

2. 시스템 모델

이 장에서 우리는 무선 브로드캐스트 환경에 대해서 간략하게 설명한다. 기본적인 모델은 Acharya가 제시한 브로드캐스트 디스크(Broadcast Disks)와 같다[1]. 시스템은 그림 1과 같이 크게 서버와 모바일 클라이언트로 구성된다. 서버는 주기적으로 데이타베이스의 데이타 아이템을 하나 혹은 그 이상의 무선 채널을 통해서 모바일 클라이언트에게 브로드캐스팅 한다. 그리고 모바일 클라이언트는 필요한 데이타가 있을 때, 해당 데이타가 브로드캐스팅 될 때까지 대기한 뒤 채널을 통해 수신한다. 따라서 모바일 클라이언트의 수가 증가해도 개별 데이타에 대한 엑세스 타임의 증가, 즉 성능의 저하는 발생하지 않는다.

서버는 데이타베이스 내의 데이타의 일관성을 유지할 책임을 가진다. 또한 하나의 브로드캐스트 주기 동안 브로드캐스팅 되는 데이타는 일관성을 가진다. 다시 말해 브로드캐스팅 되는 데이타는 해당 브로드캐스트 주기의 시작 이전의 데이타베이스의 상태만을 반영하고, 브로드캐스트 주기 도중 일어나는 모든 데이타에 대한 갱신은 다음 주기에 반영된다. 또한 서버와 클라이언트는 각각 읽기 전용 트랜잭션과 갱신 트랜잭션을 수행할 수 있다. 본 논문에서는 트랜잭션 처리의 정확성의 평가 조건으로 직렬성(serializability)을 사용했다[9].

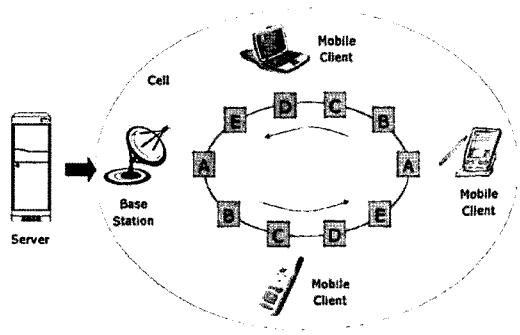

그림 1 무선 브로드캐스트 환경

또한, 브로드캐스트 사이클에는 데이타베이스의 데이타, 아이템과 함께 데이타 아이템을 위한 인덱스(index)나 무효화 보고(invalidation report)와 같은 유용한 제

어 정보가 제공된다.

3. 무선 브로드캐스트 환경에서의 기존의 동시성 제어 기법

이 장에서 우리는 무선 브로드캐스트 환경을 위해서 기존에 제안된 동시성 제어 기법들을 살펴본다. 먼저 무선 브로드캐스트 환경의 특성과 그로 인해 낙관적 동시성 제어 기법이 비관적 동시성 제어 기법보다 적합한 이유를 살펴본다. 그리고 이어서 기존에 제안된 동시성 제어 기법들과 그들의 특징을 살펴보도록 한다.

3.1 무선 브로드캐스트 환경과 동시성 제어 기법

무선 브로드캐스트 환경은 일반적으로 물리적 통신 매체의 한계와 응용 분야의 정보 흐름(information flow)의 패턴들로 인해서 비대칭적인 특징들을 가지게 된다[1]. 그러한 특징들로는 네트워크 대역폭의 비대칭, 클라이언트와 서버의 비율의 비대칭, 모바일 클라이언트의 제한된 자원 등이 존재하고, 따라서 동시성 제어 기법은 이를 고려해야 한다[2,5,8].

일반적으로 동시성 제어 기법은 낙관적 동시성 제어 기법과 비관적 동시성 제어 기법으로 나뉜다. 하지만 다양한 락(lock) 기반의 기법이나 타임스탬프 순서화(timestamp ordering) 기법과 같은 비관적 동시성 제어 기법은 무선 브로드캐스트 환경의 특성과 잘 맞지 않는다. 첫째로 클라이언트와 서버 사이의 통신을 지나치게 요구한다. 예를 들어 락 기반의 기법을 사용한다면, 모바일 트랜잭션이 데이타 아이템에 대한 락을 요구할 때마다 모바일 클라이언트와 서버 사이의 양방향 통신이 필요하다. 이는 모바일 클라이언트의 상향 통신 대역폭을 사용하고, 배터리를 소모하게 되므로 한정된 자원을 가진 클라이언트에게는 부담이 된다. 둘째로 무선 환경의 특성상 모바일 클라이언트는 자주 접속이 단절될 수 있는데, 모바일 클라이언트가 단절되면 그 클라이언트에서 실행하고 있던 트랜잭션들이 가진 락들을 기다리는 다른 모든 모바일 트랜잭션들의 처리가 지연된다. 따라서 전체적으로 시스템 내의 트랜잭션들의 평균 응답 시간이 길어질 수 있다. 셋째로 무선 브로드캐스트 환경의 다양한 응용 분야는 클라이언트의 수가 매우 많은데, 이는 지나치게 많은 락 요구로 서버에게 과부하를 가져올 수 있다[5].

그에 비해 낙관적 동시성 제어 기법은 특성이 무선 브로드캐스트 환경에 적합하다. 낙관적 동시성 제어 기법에서 모바일 트랜잭션은 검증 단계(validation phase)에 이르기 전까지는 서버에게 메시지를 보낼 필요가 없다[10]. 모바일 클라이언트는 서버가 브로드캐스팅 하는 데이타 중에서 필요한 데이타를 수신하고, 그를 바탕으로 자신의 작업 공간(local workspace)에서 모바일 트랜잭션을 수행한다. 그리고 모바일 트랜잭션은 모든 실행이 끝나면 검증에 필요한 정보를 모아서 서버에 전송하고 최종 검증 단계를 거친다. 따라서 낙관적 기법은 비관적인 기법에 비해서 모바일 클라이언트에게 더 적은 통신 비용을 요구하게 된다. 하지만 낙관적 동시성 제어 기법의 성능은 트랜잭션 재실행률에 크게 영향을 받는다. 트랜잭션의 재실행률이 높아지면 높아질수록, 모바일 클라이언트는 상향, 하향 통신 대역폭과 배터리를 소모하게 되고, 응답 시간도 길어지게 된다. 따라서 모바일 트랜잭션의 재실행률을 낮추고, 재실행시 발생하는 자원 소모를 줄이는 것이 프로토콜 디자인의 핵심이라고 할 수 있다. 우리의 방법에 대한 좀 더 쉬운 이해를 위해 기존에 제안된 무선 브로드캐스트 환경을 위한 낙관적 동시성 제어 기법들을 간략하게 살펴보겠다.

3.2 기존의 낙관적 동시성 제어 기법

무선 브로드캐스트 환경을 위한 낙관적 동시성 제어 기법은 이미 많이 연구되었다. 그러나 대다수는 읽기 전용 트랜잭션(read-only transaction)만을 위한 기법들로, 갱신 연산이 포함된 트랜잭션에는 적용할 수 없다. 그래서 우리가 제안하는 기법은 읽기와 갱신이 포함될 수 있는 일반적인 트랜잭션을 고려한다. 여기서 우리는 일반적인 트랜잭션을 위해 기존에 제안된 낙관적 동시성 제어 기법들을 살펴보겠다[5,11,12].

Chung은 모바일 트랜잭션이 지역 캐시에 있는 아이템을 엑세스할 때는 낙관적인 동시성 제어 기법을, 서버에 있는 아이템을 엑세스할 때는 비관적 동시성 제어 기법을 사용하는 방법을 제안했다[12]. 이 기법은 낙관적 동시성 제어 기법과 비관적 동시성 제어 기법을 동시에 사용하고 있다. 여기서 비관적 동시성 제어 기법은 서버가 데이타와 함께 전송하는 직렬화 그래프(serialization graph)의 부분적인 뷰(partial view)를 이용한다. 이를 통해 읽기 전용 트랜잭션은 지역적으로 트랜잭션을 커미트하고, 갱신 트랜잭션은 직렬화 될 수 없는 상황을 조기에 감지하고 트랜잭션을 중단할 수 있다. 그러나 이 기법은 비동기식 브로드캐스팅을 기반으로 하기 때문에, 모바일 클라이언트는 하향 통신 채널을 계속적으로 수신해야하고 배터리를 소모하게 되는 문제가 있다. Lee는 [11]에서 트랜잭션의 타임스탬프를 구간으로 기록하고, 이를 동적으로 조정함으로써 불필요한 트랜잭션의 중단과 재실행을 줄이는 기법을 제안했다. 이 기법은 또한 트랜잭션들 간의 충돌을 초기에 발견하고, 중단할 수 있도록 해서 모바일 클라이언트의 불필요한 자원 낭비를 막는다. 그러나 무선 브로드캐스팅의 많은 응용 분야에서 대다수의 트랜잭션은 읽기 전용 트랜잭션인데, 이 기법은 읽기 전용 트랜잭션도 최종 검증을 위해 서버로 전송해야하므로 상향 통신 대역폭과 배터

리를 소모하게 되는 문제가 있다. 또한 Lee는 [5]에서 서버에서는 정방향 검증(forward validation)을 사용하고, 클라이언트에서는 부분적 역방향 검증(partial backward validation)을 사용하는 낙관적 동시성 제어 기법의 변형을 제안하였다. 이 기법에서 모바일 클라이언트 측에서 수행되는 부분적 역방향 검증은 모든 커미트된 트랜잭션들에 대해 이루어지고, 서버 측에서 수행되는 정방향 검증은 현재 수행중인 트랜잭션들에 대해 이루어진다. 이 기법에서 읽기 전용 트랜잭션들은 서버에 데이타를 보낼 필요가 없이 모바일 클라이언트가 자체적으로 검증하고 커미트할 수 있다. 이는 서버의 부담을 덜어주고, 모바일 클라이언트의 상향 통신 대역폭의 사용을 줄여준다.

그러나 기존의 기법들은 버퍼의 영향을 고려하지 않았다. 모바일 클라이언트가 버퍼를 가지고 있으면 재실행되는 트랜잭션은 기존의 실행 때 버퍼에 가져왔던 데이타를 그대로 사용함으로써 서버로부터 데이타 아이템을 가져오는데 발생하는 지연 시간을 피할 수 있다. 또한 주기를 가지고 데이타를 브로드캐스팅 하는 시스템의 특성으로 인해 Lee가 [5]과 [11]에서 제안한 기법과 같은 경우, 한 브로드캐스트 사이클 내에 동일한 데이타 아이템에 대해 갱신 연산을 수행한 트랜잭션들이 다수개 존재할 때, 그 중 최대 하나의 트랜잭션만 커미트될 수 있다. 이는 나머지 트랜잭션에 대해 반복적인 재실행을 유발하여 성능을 저하시키는 요인으로 작용한다.

본 논문에서는 기존 연구의 이런 문제들을 해결하기 위해 Lee가 제안한 부분적 역방향 검증[5]뿐만 아니라 두 가지 추가적인 기법을 사용했다. 첫째로 Franaszek가 제안한 접근 불변성[2]을 기반으로 효과적으로 프리패칭을 한다. 둘째로 트랜잭션들의 엑세스 패턴이 편향되면 선호도가 높은 아이템에 대해 여러 트랜잭션들이 경합(contention)을 벌이게 되고 반복적인 재실행이 발생하게 되는데 이를 임의 백오프를 통해 해결한다. 지금부터 본 논문에서 제안하는 방법을 자세히 설명하도록 하겠다.

4. 제안하는 동시성 제어 기법

이 장에서 우리는 무선 브로드캐스트 환경에서 효과적인 트랜잭션 처리를 위한 동시성 제어 기법을 제안한다. 이 기법은 낙관적 동시성 제어 기법의 일종으로 접근 불변성을 바탕으로 트랜잭션이 엑세스할 데이타 아이템의 집합을 부분적으로 파악하여 프리패칭 함으로써 트랜잭션의 응답 시간을 줄인다. 또한 선호도가 높은 데이타에 접근하는 트랜잭션들에게 임의 백오프를 사용함으로써 낙관적 동시성 제어기법의 가장 큰 문제점인 모바일 트랜잭션의 중단률을 낮추고, 모바일 클라이언트의 상향, 하향 통신 대역폭 사용량을 줄인다.

4.1 가정과 용어

이제 본 논문에서 제안하는 방법을 설명하기 위해 필요한 가정들과 용어들을 살펴보도록 하겠다.

- 데이타는 브로드캐스트 디스크[5]의 방식으로 주기(cycle)를 가지고 브로드캐스팅 된다.
- 모바일 클라이언트는 모바일 트랜잭션을 실행하는데 필요한 충분한 버퍼(메모리)를 가지고 있다.
- 트랜잭션은 서버에서 수행되는 서버 트랜잭션과 모바일 클라이언트에서 수행되는 모바일 트랜잭션으로 분류된다.
- 중단(abort)된 트랜잭션은 커미트 될 때까지 재실행(restart)한다.
- 트랜잭션은 데이타에 대해서 읽지 않고 쓰기(blind write)를 하지 않는다.
- 각 주기의 시작에는 해당 주기에 포함된 데이타 아이템에 대한 인덱스가 제공된다. 그를 통해 모바일 클라이언트는 특정한 데이타가 브로드캐스팅 되는 시점을 알 수 있다. 그러므로 원하는 데이타가 브로드캐스팅 될 때까지 대기 상태(doze mode)로 유지함으로써 불필요한 에너지 소모를 하지 않는다.
- 각 주기의 시작에는 이전 주기 동안 갱신된 데이타 아이템의 집합이 무효화 보고로 제공된다. 모바일 클라이언트는 버퍼에 존재하는 데이타의 값이 서버에서 이미 갱신되었으면 일관성을 유지하기 위해 버퍼의 데이타를 무효화한다.

브로드캐스트 사이클의 시작에는 데이타 아이템에 대한 무효화 보고가 제공되어 모바일 클라이언트는 버퍼에 존재하는 데이타의 값이 서버에서 이미 갱신되었으면 더 이상 버퍼의 데이타를 사용하지 않도록 한다. 다만 본 논문에서는 인덱스와 무효화 보고가 제공된다고 가정하고 특별한 제한은 두지 않는다.

4.2 기본 아이디어

3.1장에서 살펴본 것처럼 무선 브로드캐스트 환경에서는 낙관적 동시성 제어기법을 사용하는 것이 합리적이다. 일반적으로 낙관적 동시성 제어 기법은 정방향 검증(forward validation)과 역방향 검증(backward validation)의 두 가지 방식으로 나누어진다[13].

- 역방향 검증: 검증을 수행하는 트랜잭션은 검증 과정에서 이미 기존에 커미트 된 모든 트랜잭션에 대해서 충돌(conflict) 검사를 수행한다.
- 정방향 검증: 검증을 수행하는 트랜잭션은 검증 과정에서 현재 병렬적으로 수행 중인 모든 트랜잭션에 대해서 충돌 검사를 수행한다.

역방향 검증은 모든 커미트 된 트랜잭션을 대상으로 충돌 검사를 수행하기 때문에 검증 과정에서 충돌이 발

생하면 현재 검증 중인 트랜잭션을 재실행해야한다. 반면 정방향 검증은 현재 병렬적으로 실행중인 트랜잭션들을 대상으로 충돌 검사를 수행하므로, 실행중인 트랜잭션들 중에서 재실행할 트랜잭션을 선택할 수 있다. 일반적으로 읽기 단계를 끝내고 검증을 수행 중인 트랜잭션을 재실행하는 것보다 아직 읽기 단계를 수행 중인 트랜잭션을 중단하는 것이 재실행 비용의 측면에서 저렴하다. 또한 실시간 데이타베이스 시스템의 경우 트랜잭션의 마감 기한을 고려하여 중단시킬 트랜잭션을 선택할 수 있다. 그래서 많은 데이타베이스 시스템에서 정방향 검증이 선호된다.

우리가 가정하는 무선 브로드캐스트 환경에서 정방향 검증은 다음과 같은 이유로 적합하다. 첫째, 모바일 트랜잭션이 모바일 클라이언트에서 수행이 끝나고 검증을 위해서 서버로 전송이 되면 가능한 재실행되지 않아야 한다. 이는 모바일 클라이언트가 가진 자원이 제한되어 있기 때문이다. 정방향 검증을 사용하면 중단할 트랜잭션을 선택할 수 있고 모바일 트랜잭션이 중단되는 것을 피할 수 있다. 둘째, 정방향 검증은 검증할 트랜잭션의 쓰기 집합만을 요구한다. 따라서 모바일 클라이언트는 모바일 트랜잭션의 쓰기 집합에 대한 정보만 전송하면 된다. 그리고 읽기 전용 트랜잭션은 쓰기 집합이 없으므로 지역적으로 검증하고 커미트할 수 있다. 따라서 정방향 검증은 역방향 검증에 비해 모바일 클라이언트의 통신 대역폭 사용을 줄일 수 있다. 그러므로 우리는 정방향 검증 기반의 낙관적 동시성 제어기법을 바탕으로 모바일 클라이언트들이 자체적인 수행 검증을 위해 부분적인 역방향 검증을 사용한다[5].

우리의 기법에서는 재실행 시 트랜잭션의 응답시간을 줄이기 위해서 접근 불변성을 기반으로 데이타 아이템을 프리패칭한다. 접근 불변성이란 트랜잭션이 첫 번째 실행과 재실행될 때 엑세스하는 데이타 아이템이 다르지 않다는 것을 의미하는데, Franaszek는 높은 수준의 접근 불변성이 오늘날 많은 데이타베이스 시스템에 존재한다고 주장하며 그 근거로 두 가지를 언급하고 있다. 첫째로 오늘날의 많은 데이타베이스 시스템들에서 직렬성(serializability)이 꾸준히 정확성의 척도로 받아들여지고 있다는 점, 둘째로 많은 트랜잭션들이 생성되는 방식이 일반적인 경우를 처리하기 위한 일련의 문장들과 데이타베이스 호출들, 그리고 몇몇 예외상황을 처리하기 위한 것들로 구성되는 점을 지적했다. 그래서 높은 수준의 접근 불변성을 가정한 동시성 제어 기법들이 다수 연구되었다. 따라서 우리의 기법에서 모바일 클라이언트는 모바일 트랜잭션이 실행되는 동안 엑세스하는 데이타 아이템의 집합을 기록해두었다가 재실행할 때 기록된 데이타 아이템에 대해서는 프리패칭함으로써 모바일 트랜잭션이 무선 브로드캐스트 채널을 통해서 데이타를 수신하는데 발생하는 지연 시간을 줄이고 응답 시간을 단축시킨다.

기존의 Lee가 제안한 정방향 검증 기반의 낙관적 동시성 제어기법은 트랜잭션들이 편향된 엑세스 패턴을 따를 경우, 즉 특정 데이타 아이템들에 대한 엑세스가 상대적으로 많을 경우 계속적인 트랜잭션 재실행으로 인해 성능의 저하가 발생한다. 이는 한 브로드캐스트 사이클 내에서 동일한 아이템에 대해 갱신 연산을 수행한 모바일 트랜잭션들이 다수 존재할 경우, 그들 중 최대 하나의 트랜잭션만이 커미트될 수 있기 때문이다. 동일한 데이타 아이템에 대한 쓰기를 수행하려는 트랜잭션이 많아지면 많아질수록 자연스레 모바일 트랜잭션이 재실행될 확률은 높아진다.

정리 1. 이 시스템에서 하나의 브로드캐스트 사이클 내에 동일 데이타 아이템에 대해 쓰기 연산을 수행한 모바일 트랜잭션들이 존재하면, 그 트랜잭션들 중 최대 하나의 트랜잭션만 커미트 될 수 있다.

증명. 하나의 브로드캐스트 사이클 내에서 데이타 아이템 x에 대해 쓰기 연산을 수행한 뒤 커미트된 모바일 트랜잭션들이 두 개 이상 존재한다고 가정하자. 그들 중 임의의 두 개의 트랜잭션을 T_a와 T_b라 두면 T_a에는 $r_a(x)$, $w_a(x)$가, T_b에는 $r_b(x)$, $w_b(x)$가 반드시 포함된다. 그리고 그들이 수행한 스케줄을 s라 하자. 그러면 s는 직렬화 가능하기 위해서 $r_a(x)w_a(x)r_b(x)w_b(x)$ 혹은 $r_b(x)w_b(x)r_a(x)w_a(x)$와 충돌 동등(conflict equivalent)해야 한다. 그러나 주기를 가지고 브로드캐스팅을 하는 브로드캐스트 디스크의 특성으로 인해 모바일 트랜잭션의 읽기 연산은 현재 브로드캐스트 사이클 이전에 갱신된 내용을 읽게 되므로 $w_a(x)r_b(x)$나 $w_b(x)r_a(x)$와 같은 스케줄은 존재할 수 없고, 스케줄의 모든 읽기 연산은 쓰기 연산에 우선 되어야 한다. 즉, $r_a(x)r_b(x)w_a(x)w_b(x)$나 $r_b(x)r_a(x)w_b(x)w_a(x)$와 같이 되어야한다. 따라서 가정에 모순이 된다. □

또 하나 트랜잭션의 재실행을 유발하는 요인은 버퍼이다. 기존에 제안된 Lee의 낙관적 동시성 제어기법은 모바일 클라이언트 측 버퍼의 영향을 고려하지 않았다[5,11]. 하지만 모바일 클라이언트에 버퍼가 존재할 경우, 트랜잭션의 재실행 시 버퍼에 있는 데이타를 다시 사용함으로써 트랜잭션의 응답 시간을 줄일 수 있다. Yu는 낙관적 동시성 제어기법 하에서 버퍼를 고려할 경우, 트랜잭션의 재실행 시 버퍼에 있는 데이타를 활용함으로써 불필요한 I/O를 줄일 수 있다고 지적하였다. 그리고 이로 인해 트랜잭션의 응답 시간이 단축되고, 성능이 향상되는 것을 보였다[14-16]. 이는 일반적인 데이타베이스 환경을 가정했지만, 무선 브로드캐스팅 환경에

서도 적용될 수 있다. 오히려 메모리나 디스크에서 데이타 아이템을 엑세스 하는 것보다 모바일 클라이언트가 무선 브로드캐스트 채널을 통해서 데이타를 엑세스하는데 걸리는 시간이 훨씬 길기 때문에 버퍼를 사용했을 때 모바일 트랜잭션의 성능 향상은 훨씬 크다. 그런데 때로는 버퍼로 인해서 트랜잭션이 응답시간이 짧아지는 것이 아래와 같은 상황을 야기한다.

그림 2에서 모바일 트랜잭션 T_1, T_2, T_3, T_4, T_5는 모두 하나의 아이템 x에 대한 쓰기 연산을 포함하고, 재실행 중인 트랜잭션이다. 트랜잭션은 높은 수준의 접근 불변성을 따르고, 모바일 클라이언트는 버퍼를 가지고 프리패칭을 한다고 가정하자. 이를 바탕으로 모바일 트랜잭션이 재실행될 때 필요한 데이타 아이템을 하나의 브로드캐스트 사이클 내에 모두 엑세스 할 수 있다면 하나의 사이클 내에 충분히 수행될 수 있다. 물론 모바일 트랜잭션의 길이와 브로드캐스트 사이클의 길이 등 다양한 외부 요소에 따라서 트랜잭션의 응답 시간이 결정되지만 이 예에서는 재실행되는 모바일 트랜잭션이 하나의 브로드캐스트 사이클 내에서 실행된다고 가정한다. 그러면 정리 1에 의해 하나의 브로드캐스트 사이클에 최대 하나의 트랜잭션만 커미트될 수 있으므로, 기존 낙관적 동시성 제어기법을 사용하면 트랜잭션들의 반복적인 재실행이 발생하게 된다. 그림 2의 좌측을 보면 브로드캐스트 사이클 C_{i-2}에서 4번, C_{i-1}에서 3번, C_i에서 2번, C_{i+1}에서 1번, 즉, 총 10번의 트랜잭션 재실행이 발생한다. 이러한 계속적인 재실행은 트랜잭션들의 데이타 아이템에 대한 엑세스 패턴이 편향될 경우 더 심해진다.

위와 같이 반복적인 트랜잭션의 재실행이 발생하면 모바일 클라이언트는 상향, 하향 통신 대역폭과 배터리를 낭비하게 된다. 따라서 버퍼를 사용하는 일반적인 모바일 클라이언트들에게 기존 동시성 제어 기법을 적용하는 것에 문제가 있다. 우리는 그러한 반복적인 재실행을 막을 수 있는 기법을 제안한다. 제안하는 기법에서는 서버가 모바일 클라이언트에게 데이타 아이템에 대한 경합 정도를 제어 정보로 제공하여 모바일 클라이언트가 임의 백오프(random back-off)를 하도록 함으로써 모바일 트랜잭션의 엑세스를 분산시킨다. 이 기법은 그림 2의 우측과 같이 반복적인 재실행을 피할 수 있도록 하는 기법으로 이더넷(ethernet)의 CSMA/CD에서 사용하는 충돌 회피 기법과 유사하다. 예에서 서버는 사이클 C_{i-2}의 시작에 데이타 아이템 x에 대한 경합 정도로 3이라는 값을 보내준다. 재실행되는 트랜잭션은 이 경합 정도를 바탕으로 0부터 3사이의 임의의 정수를 선택해 선택한 수의 브로드캐스트 사이클만큼 기다린 뒤 재실행한다. 따라서 동일한 데이타 아이템에 대해서 경합을 벌이는 모바일 트랜잭션의 접근이 분산되고 반복적인 재실행을 줄일 수 있다. 이 때 재실행될 모바일 클라이언트가 중단된 모바일 트랜잭션을 재실행할 브로드캐스트 사이클을 서버가 직접 지정하지 않고 모바일 클라이언트가 임의로 선택하게 하는 것은 잦은 접속 단절이 발생하는 모바일 클라이언트의 특성에 기인한다. 즉, 서버가 특정한 클라이언트에게 실행의 권한을 부여했을 때, 해당 트랜잭션이 제어 정보를 놓치거나 접속 단절 등으로 실행을 하지 못하게 되면 해당 브로드캐스트 사이클을 낭비하게 되는 결과를 가져오기 때문이다.

4.3 모바일 클라이언트 측의 부분적 역방향 검증 기법

지금부터 모바일 클라이언트 측에서 모바일 트랜잭션들을 처리하기 위한 프로토콜을 설명하겠다. 모바일 트랜잭션은 기본적으로 다음과 같은 순서로 수행된다.

1) 데이타 아이템에 대한 읽기/쓰기 연산을 포함하는 모바일 트랜잭션이 모바일 클라이언트의 버퍼를 이용해 수행된다.
2) 모바일 트랜잭션의 수행 도중 다음 사이클이 시작되면, 서버로부터 브로드캐스팅된 제어 정보를 바탕으로 부분적 역방향 검증을 수행한다. 부분적 역방향 검증의 결과에 따라 모바일 트랜잭션은 계속

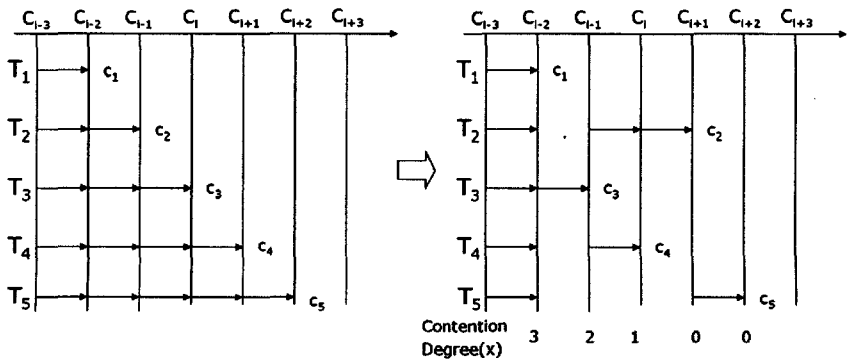

그림 2 소모적 재실행과 임의 백오프

진행되거나 재실행(restart)된다.

3) 모든 수행이 끝나면 모바일 읽기 전용 트랜잭션은 자체적으로 커미트되고, 모바일 갱신 트랜잭션은 서버로 보내 최종 검증을 받는다.

1)은 기존의 낙관적 동시성 제어 기법과 동일하다. 2)에서 모바일 클라이언트는 서버로부터 바로 전 브로드캐스트 사이클 동안 갱신된 데이타 아이템에 대한 목록을 수신하여 현재 실행 중인 모바일 트랜잭션의 부분적 역방향 검증을 수행해야 한다. 3)에서 모바일 갱신 트랜잭션은 최종 검증을 위해 자신의 쓰기 집합에 대한 정보를 서버에 전송한다. 그리고 최종 검증을 마치고 문제가 없으면 그 결과는 서버에 반영된다. 그리고 쓰기 집합이 공집합인 모바일 읽기 전용 트랜잭션은 지역적으로 검증을 마치고 커미트될 수 있다.

따라서 클라이언트 프로토콜의 핵심은 부분적 역방향 검증에 있다. 앞에서 설명한 것처럼 서버 트랜잭션이나 모바일 갱신 트랜잭션의 최종 검증은 정방향 검증을 통해서 이루어진다. 정방향 검증을 통해서 트랜잭션이 커미트 되었다면 현재 병렬적으로 읽기 단계를 수행 중인 모든 트랜잭션들 중에서 데이타 충돌이 발생하는 트랜잭션들은 중단되어야한다. 즉, 커미트된 트랜잭션의 쓰기 집합에 포함되는 데이타를 읽은 트랜잭션은 중단되어야 한다. 그러나 서버에서 트랜잭션이 커미트 될 때 그 정보를 모바일 트랜잭션이 즉시 받기 위해서는 지속적으로 브로드캐스트 채널을 수신해야 하는 문제가 있다. 이 문제는 하나의 브로드캐스트 사이클 내 커미트된 트랜잭션들의 쓰기 집합에 포함되는 데이타 아이템을 모아둔 뒤, 그 목록을 다음 브로드캐스트 사이클의 시작에 제어 정보로 일괄적으로 브로드캐스팅함으로써 해결되었다[5]. 물론 바로 중단되어야할 트랜잭션이 최대 하나의 브로드캐스트 사이클만큼 지연된 뒤 중단된다는 단점이 있지만, 모바일 클라이언트의 배터리 소모를 줄일 수 있는 장점이 있다. 따라서 우리의 기법에서도 같은 방법을 사용한다. 이 때, 본 논문의 가정과 같이 무효화 보고의 내용이 이전 사이클 동안 갱신된 데이타 아이템의 집합이라면 위에서 언급한 제어 정보와

```
Tpv: 부분적 역방향 검증을 받는 모바일 트랜잭션
UpdatedSet(Ci): 브로드캐스트 사이클 Ci 동안 서버에서 갱신된 데이터 아이템의 집합
CurrentReadSet(Tpv): Tpv가 현재까지 읽은 데이터 아이템의 집합
CurrentWriteSet(Tpv): Tpv가 현재까지 갱신한 데이터 아이템의 집합
PrefetchingSet(Tpv): 트랜잭션 Tpv가 프리패칭할 데이터 아이템의 집합
ContentionDegree[x]: 데이터 아이템 x에 대한 경합 정도

PartialBackwardValidation(Tpv)
{
    if UpdatedSet(Ci-1) ∩ CurrentReadSet(Tpv) ≠ { } {
        PrefetchingSet(Tpv) = PrefetchingSet(Tpv) ∪ CurrentReadSet(Tpv);
        BackoffAndRestart(Tpv);
    } else {
        record the value of Ci-1;
        continue;
        // Tpv의 모든 수행이 끝나면 LastPartialBackwardValidation(Tpv)를 호출
    }
}

BackoffAndRestart(Tpv)
{
    abort(Tpv);
    Wait as a doze mode until the start of next broadcast cycle;
    if CurrentWriteSet(Tpv) ≠ { } {
        MaxContentionDegree = MAX_{x ∈ CurrentWriteSet(Tpv)} (ContentionDegree[x]);
    } else {
        MaxContentionDegree = 0;
    }
    Choose a random number between 0 to MaxContentionDegree as BackoffTime;
    Wait as a doze mode during BackoffTime broadcast cycles;
    restart(Tpv);
}
```

그림 3 부분적 역방향 검증 알고리즘

같으므로 무효화 보고를 제어 정보로 사용할 수 있다. 만약 다른 방식의 무효화 보고가 제공된다면 그로부터 제어정보를 추출하거나 별도의 제어 정보가 제공되어야 한다.

모바일 클라이언트는 모든 실행 중인 모바일 트랜잭션들에 대해 제어정보를 바탕으로 부분적 역방향 검증을 수행한다. 부분적 역방향 검증 알고리즘은 다음의 그림 3과 같다. 여기서 우리는 T_{pv}가 브로드캐스트 사이클 C_i의 시작에 부분적 역방향 검증을 받는다고 가정한다.

실행 중인 모바일 트랜잭션은 브로드캐스트 사이클의 시작에 서버로부터 받은 제어 정보를 바탕으로 Partial-BackwardValidation()에 제시된 알고리즘에 따라 부분적 역방향 검증을 실행한다. 현재 검증을 받는 모바일 트랜잭션이 읽은 데이타 아이템의 집합과 서버에서 갱신된 데이타 아이템의 집합을 비교해서 교집합이 존재하면 데이타 충돌(data conflict)이 발생한 것이므로 부분적 역방향 검증을 받는 모바일 트랜잭션은 중단된다. 그리고 모바일 트랜잭션이 현재까지 읽은 데이타 아이템의 집합을 프리패칭할 데이타 아이템의 집합에 포함시킨다. 이런 방법으로 모바일 트랜잭션의 엑세스 패턴을 파악하고 재실행 시 프리패칭을 함으로써 트랜잭션의 응답시간을 줄인다. 그리고 BackoffAndRestart()에 제시된 알고리즘에 따라서 모바일 트랜잭션을 재실행한다. 부분적 역방향 검증에 성공했을 경우는 현재 브로드캐스트 사이클의 번호를 저장해둔다. 이 번호는 트랜잭션의 수행을 마치고 서버에 보내졌을 때, 정방향 검증 이전에 최종 역방향 검증을 받을 때 사용된다.

재실행할 트랜잭션은 BackoffAndRestart()에 제시된 알고리즘에 따라 재실행을 수행한다. 트랜잭션은 일단 중단된 뒤, 다음 브로드캐스트 사이클의 시작까지 대기한다. 그리고 쓰기 집합에 포함된 데이타 아이템에 대한 경합 정도(contention degree)를 서버로부터 수신하여 그 중 최대값을 선택한다. 그리고 0부터 최대값 사이의 임의의 정수를 선택하고, 선택한 브로드캐스트 사이클 동안 대기 모드로 유지한 뒤 재실행한다.

실행을 끝낸 모바일 트랜잭션은 그림 4의 LastPartial-BackwardValidation()에 제시된 알고리즘에 따라서 모바일 클라이언트 측에서 최종 검증을 하게 된다. 읽기 전용 트랜잭션은 쓰기 집합이 없으므로 서버에 전송할 필요가 없이 지역적으로 커미트하고, 갱신 트랜잭션은 쓰기 집합에 관한 정보와 마지막으로 부분적 역방향 검증을 받은 브로드캐스트 사이클의 번호를 서버에 전송하여 정방향 검증을 받는다.

그림 5의 스케줄을 따르는 아래의 일련의 트랜잭션 집합을 고려해보자.

모바일 트랜잭션:

T_1: $r_1(a)$ $r_1(b)$ $r_1(c)$ $w_1(a)$ $w_1(b)$ $w_1(c)$

T_3: $r_3(b)$ $r_3(c)$ $r_3(d)$

서버 트랜잭션:

T_2: $r_2(b)$ $r_2(d)$ $w_2(b)$ $w_2(d)$

그림 5를 보면 모바일 트랜잭션 T_1은 서버 트랜잭션 T_2가 브로드캐스트 사이클 C_{i-2}에서 아이템 b를 갱신하고 커미트 되었기 때문에 브로드캐스트 사이클 C_{i-1}에서 제어정보를 받고 부분적 역방향 검증에서 실패하게 된

```
Tpv: 최종 부분적 역방향 검증을 받는 모바일 트랜잭션
CurrentReadSet(Tpv): Tpv가 현재까지 읽은 데이터 아이템의 집합
CurrentWriteSet(Tpv): Tpv가 현재까지 갱신한 데이터 아이템의 집합
PrefetchingSet(Tpv): 트랜잭션 Tpv가 프리패칭할 데이터 아이템의 집합

LastPartialBackwardValidation(Tpv)
{
    PrefetchingSet(Tpv) = PrefetchingSet(Tpv) ∪ CurrentReadSet(Tpv);

    if Tpv is read-only transaction {
        commit(Tpv);
        exit;
    }
    summit the values of CurrentWriteSet(Tpv) and Ci-1 to server;
    if Tpv is successfully validated {
        commit(Tpv);
        exit;
    } else {
        BackoffAndRestart(Tpv);
    }
}
```

그림 4 최종 부분적 역방향 검증 알고리즘

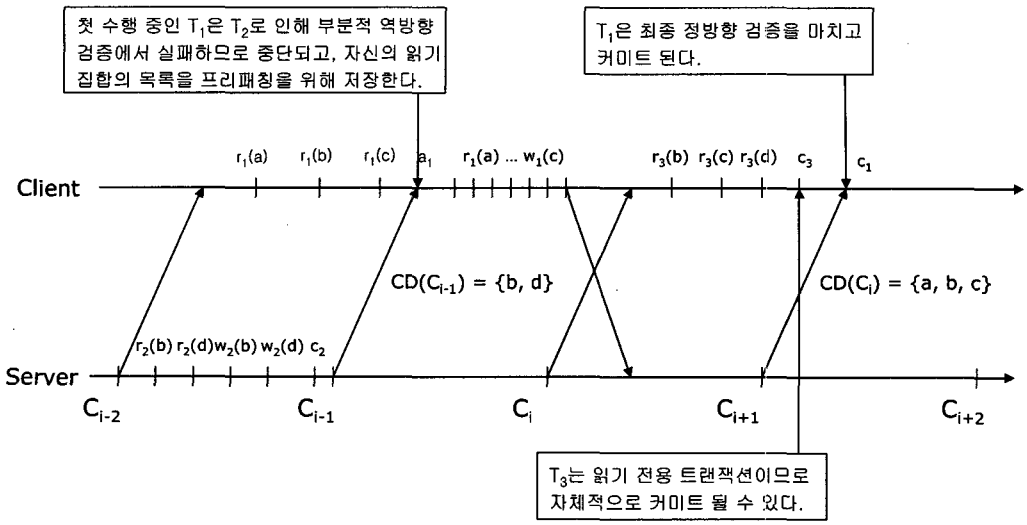

그림 5 트랜잭션 수행 스케줄

다. 따라서 T_1는 중단된다. 그리고 재실행시 프리패칭을 이용하여 버퍼 유지 효과를 높이고, 버퍼에 있는 데이타를 이용해 수행함으로써 빠르게 수행을 마치고, 브로드캐스트 사이클 C_i에서 커미트 된다. 모바일 트랜잭션 T_3은 읽기 전용 트랜잭션이므로 서버에 전송할 필요가 없이 지역적으로 검증을 마치고 브로드캐스트 사이클 C_i 내에 커미트된다.

4.4 서버 측의 정방향 검증 기법

지금부터는 서버 측에서 트랜잭션들을 처리하기 위한 프로토콜을 설명한다. 서버 측 프로토콜의 경우, 경합 정도와 관련된 부분을 제외하고는 Lee가 제안한 프로토콜과 동일하다[5]. 서버 측에서는 정방향 검증 기법을 사용한다. 하나의 트랜잭션에 대한 정방향 검증은 그 트랜잭션의 쓰기 집합과 병렬적으로 수행 중인 모든 다른 트랜잭션들의 읽기 집합들을 비교함으로써 이루어진다. 검증 중인 트랜잭션이 이미 갱신한 데이타 아이템에 대해서 읽기 연산을 수행한 실행 중인 다른 트랜잭션들은 모두 재실행된다. 이를 통해서 데이타 충돌을 해결한다. 여기서 정방향 검증은 서버 트랜잭션과 검증을 위해 전송된 모바일 트랜잭션에게 공통적으로 적용된다. 다만 모바일 트랜잭션의 경우 정방향 검증 이전에 최종 역방향 검증(last backward validation)의 단계를 거친다. 이는 모바일 트랜잭션의 최종 부분적 역방향 검증 이후로 서버에서 커미트된 트랜잭션이 존재할 수 있기 때문이다. 따라서 모바일 트랜잭션은 부분적 역방향 검증을 받을 때마다 그 시점의 브로드캐스트 사이클의 번호를 기록해두었다가 정방향 검증을 위해 쓰기 집합을 보낼 때 함께 전송한다. 정방향 검증의 알고리즘은 그림 6과 같다.

서버는 트랜잭션 T_v가 커미트되면 T_v의 쓰기 집합을 기록해둔다. 이는 하나의 브로드캐스트 사이클 동안 모아진 뒤, 다음 브로드캐스트 사이클의 시작에 모바일 클라이언트에게 제어 정보로 제공된다. 이 정보를 바탕으로 모바일 클라이언트는 부분적 역방향 검증을 수행한다.

또한 서버는 트랜잭션 T_v가 중단되면 T_v의 쓰기 집합을 경합 정도를 계산하는데 사용한다. 서버는 각 브로드캐스트 사이클 동안 데이타 아이템들에 대한 경합 정도를 계산하여, 다음 브로드캐스트 사이클의 시작에 제어정보로 보내준다. 경합 정도는 해당 아이템에 대한 쓰기 연산을 수행할 것으로 예상되는 모바일 트랜잭션의 수를 의미한다. 정리 2와 같이 우리의 시스템에서는 하나의 브로드캐스트 사이클 내에 쓰기 연산을 수행한 모바일 트랜잭션들 중 최대 하나만 커미트될 수 있으므로 그림 2의 좌측과 같은 반복적 재실행이 발생할 수 있다. 그러므로 경합 정도를 이용해서 모바일 트랜잭션의 접근을 분산시켜야 한다. 모바일 트랜잭션에 대한 검증이 실패하면, 그 트랜잭션은 재실행 될 것이므로 그림 6의 ForwardValidation()에 제시된 알고리즘과 같이 쓰기 집합에 포함되어있는 데이타들에 대한 경합 정도를 하나씩 증가시킨다.

그리고 각 브로드캐스트 사이클의 시작에는 그림 7의 AdjustContentionDegree()에 제시된 알고리즘에 따라 경합 정도를 조정한다. 각 데이타 아이템에 대한 경합 정도는 다음 브로드캐스트 사이클에 해당 데이타 아이템에 대해 쓰기 연산을 수행할 것으로 예상되는 모바일 트랜잭션의 수에 대한 기댓값을 의미한다. 경합 정도는 이전 사이클의 경합 정도가 1보다 클 경우 1로 조정되

```
Tv: 정방향 검증을 수행 중인 서버 혹은 모바일 트랜잭션
Ta: Tv와 병렬적으로 수행 중인 서버 트랜잭션들
Ck: Tv가 마지막으로 역방향 검증을 수행한 브로드캐스트 사이클
TCk: Ck 동안 커미트 된 트랜잭션들의 집합
ReadSet(T): 트랜잭션 T의 읽기 집합
WriteSet(T): 트랜잭션 T의 쓰기 집합
CurrentUpdated(Ci): 브로드캐스트 사이클 Ci에서 커미트 된 트랜잭션들에 의해 갱신된 데이터 아이템의 집합
ContentionDegree[x]: 데이터 아이템 x에 대한 경합 정도

ForwardValidation(Tv)
{
    if Tv is a mobile transaction {
        if LastBackwardValidation(Tv) is failed {
            abort(Tv);
            foreach x ∈ WriteSet(Tv) {
                ContentionDegree[x] = ContentionDegree[x] + 1;
            }
        }
    }
    foreach Ta {
        if ReadSet(Ta) ∩ WriteSet(Tv) ≠ { } {
            abort(Ta);
        }
    }
    CurrentUpdated(Ci) = CurrentUpdated(Ci) ∪ WriteSet(Tv);
    commit(Tv);
}

LastBackwardValidation(Tv)
{
    foreach Tc {
        if WriteSet(TCk) ∩ ReadSet(Tv) ≠ { }
            return false;
    }
}
```

그림 6 정방향 검증 알고리즘

```
ContentionDegree[x]: 데이터 아이템 x에 대한 경합 정도

AdjustContentionDegree()
{
    foreach data item x in database {
        if ContentionDegree[x] > 1 {
            ContentionDegree[x] = 1;
        } else {
            ContentionDegree[x] = 0;
        }
    }
}
```

그림 7 경합 정도 조정 알고리즘

는데, 이는 제안하는 기법이 BackoffAndRestart()에 제시된 알고리즘에 의해 경합을 벌이는 트랜잭션들을 분산시키기 때문이다. 예를 들어 특정 데이타 아이템에 대해 k개의 모바일 트랜잭션이 경합을 벌이고 있다면 모바일 트랜잭션들은 재실행될 때 각각 0부터 k-1 사이에서 하나의 값을 임의로 선택하게 되고, 선택한 수의 브로드캐스트 사이클만큼 기다린 뒤 재실행된다. 이 때, 다음 브로드캐스트 사이클에 해당 데이타 아이템에 대해서 쓰기 연산을 수행할 것으로 예상되는 모바일 트랜잭션의 수에 대한 기댓값은 0을 선택할 것으로 예상되는 모바일 트랜잭션의 수에 대한 기댓값인 1과 같다. 그러한 이유로 경합 정도 조정 알고리즘에서는 경합 정도가 1보다 클 경우 1로 조정한다.

4.5 프로토콜의 직렬성

모바일 클라이언트의 읽기 전용 트랜잭션은 부분적 역방향 검증을 무사히 마친 후 지역적으로 커미트된다. 커미트된 트랜잭션은 커미트 시점이 속한 브로드캐스트 사이클의 시작 이전으로, 그리고 이미 커미트된 모든 트랜잭션들 이후로 직렬화 된다.

갱신 트랜잭션은 최종 검증을 위해 트랜잭션의 쓰기 집합, 갱신된 아이템의 값, 역방향 검증을 받은 브로드캐스트 사이클의 번호 등을 서버로 전송한다. 서버는 해

당 정보를 바탕으로 최종 역방향 검증과 정방향 검증을 수행한다. 이를 무사히 통과하면 갱신 트랜잭션은 검증 시점 이전에 커미트된 트랜잭션의 이후로, 그리고 모든 현재 활성화된 트랜잭션 이전으로 직렬화 된다.

정리 2. 이 시스템에서 모든 커미트 된 트랜잭션은 직렬화 가능하다.

증명. 이 정리의 정확성은 바로 이전 단락에 논의된 내용을 그대로 따른다. □

5. 성능 분석

이 장에서는 기본적인 낙관적 동시성 제어기법과 Lee가 제안한 정방향, 역방향 검증을 이용한 낙관적 동시성 제어기법[5]과 비교를 통해서 본 논문에서 제안하는 방법의 성능을 분석한다. Chung이 [12]에서 제안한 기법이나 Lee가 [11]에서 제안한 기법은 이미 언급한 것과 같이 불필요한 통신 대역폭을 지나치게 소모하게 되므로 고려하지 않았다. 그리고 우리는 임의 백오프의 성능을 확인하기 위해, 제안한 기법을 임의 백오프를 사용할 때와 사용하지 않을 때의 두 가지로 나누어 실험하였다. 편의를 위해서 이후에는 이 논문에서 제안하는 기법을 AOCCRB(Adaptive Optimistic Concurrency Control with Random Back-off), 임의 백오프를 사용하지 않는 기법은 AOCC(Adaptive OCC), 기본적인 낙관적 동시성 제어기법을 OCC, Lee가 제안한 낙관적 동시성 제어기법을 FBOCC로 쓰도록 하겠다.

본 논문의 모든 실험은 AMD Athlon XP 1600+ 프로세서와 512MB 메모리 상에서 시뮬레이션 되었다. 시뮬레이터는 CSIM18 시뮬레이션 엔진을 이용해서 구현하였다[17,18]. 위에서 설명한 것과 같이 모든 실험에 대해서 모바일 클라이언트들의 비정규 분포의 접근 패턴을 모델링하기 위해서 매개 변수 θ를 갖는 Zipf 분포를 사용하였다. Zipf 분포는 모바일 브로드캐스트 환경에서 모바일 클라이언트의 편향된 분포의 접근 패턴을 모델링 하는 데 널리 사용된다.

기본적으로 모델은 모바일 클라이언트, 서버, 그리고 데이타와 제어 정보를 전송하기 위한 브로드캐스트 디스크로 구성된다. 모바일 클라이언트 쪽에서는 읽기 전용 트랜잭션과 갱신 트랜잭션이 모두 발생한다. 그리고 트랜잭션은 커미트 될 때까지 계속 실행되고 마감시간을 고려하지 않는다. 본 논문의 시뮬레이션에서 사용된 여러 가지 매개 변수는 아래 표와 같다.

본 논문에서는 모바일 클라이언트의 상향, 하향 통신 대역폭 사용량과 트랜잭션의 평균 응답 시간을 성능 지표로 삼았다. 하향 통신 대역폭 사용량은 모바일 클라이언트가 모바일 트랜잭션을 수행하면서 브로드캐스트 채널을 통해서 수신한 데이타의 양을, 상향 통신 대역폭 사용량은 모바일 클라이언트가 갱신 트랜잭션을 수행하면서 백 채널을 통해서 서버로 전송한 데이타의 양을 나타낸다. 따라서 상향, 하향 통신 대역폭 사용량은 에너지 효율성의 성능 지표가 된다. 실험의 결과는 10,000번의 시뮬레이션을 통해서 모바일 클라이언트의 평균을 취하였다. 측정의 단위는 비트 시간(bit-time)을 기준으로 하였다. 예를 들어, 하나의 서버가 하나의 데이타 아이템을 브로드캐스팅 하는 데는 8,000 비트 시간이 소요된다. 다음 장에서 여러 가지 조건에서 AOCCRB, AOCC, OCC, 그리고 FBOCC의 성능을 분석한다.

표 1 시뮬레이션 매개변수

시뮬레이션 매개 변수	범위
서버 일반	
브로드캐스트 디스크의 수	3
Zipf 매개 변수 θ	0.0 ~ 1.0
데이타베이스 내의 데이타의 수	300
데이타 오브젝트의 크기	8000 bits
모바일 트랜잭션(모바일 클라이언트)	
트랜잭션의 길이(연산의 수)	8
읽기 연산의 확률(갱신 트랜잭션)	0.5
읽기 전용 트랜잭션의 비율	0.7
연산 사이의 평균지연시간	65,536 bit-times(exponentially distributed)
트랜잭션 사이의 평균지연시간	131,072 bit-times(exponentially distributed)
접근 불변성의 정도	0.95, 0.0 ~ 1.0
서버 트랜잭션(서버)	
트랜잭션의 길이	8
트랜잭션의 도착 빈도	1 per 1,000,000 to per 333,333 bit-times
읽기 연산의 확률	0.5
동시성 제어 기법	OCC with forward validation

5.1 트랜잭션의 접근 패턴의 편향 정도에 따른 성능

트랜잭션의 데이타 아이템에 대한 엑세스 패턴이 편향될수록 트랜잭션간의 충돌이 발생할 확률이 커지기 때문에 엑세스 패턴의 분포가 동시성 제어 기법의 성능에 영향을 미치게 된다. 그러므로 우리는 Zipf 매개 변수 θ를 0.0에서 1.0까지 증가시키면서 세 가지 동시성 제어기법이 엑세스 패턴의 편향성 변화에 어떻게 적응하는지를 분석한다. Zipf 매개 변수의 값이 크면 클수록 편향된 엑세스 패턴을 생성하게 된다.

5.1.1 트랜잭션의 접근 패턴의 편향 정도에 따른 응답 시간의 변화

그림 8은 트랜잭션의 엑세스 패턴에 따른 평균 응답 시간의 변화를 나타낸다. θ가 0.4보다 작을 경우, AOCCRB와 AOCC는 FBOCC보다 최대 25%, OCC보다는 최대 60% 정도 짧은 응답시간을 보인다. 이는 우리가 제안한 기법들은 트랜잭션의 엑세스 패턴을 파악하여 프리패칭함으로써 기존의 기법에 비해서 재실행시 빠르게 원하는 데이타 아이템을 엑세스할 수 있기 때문이다. 그러나 θ가 0.4보다 클 경우 AOCCRB와 FBOCC 사이의 응답시간의 차이는 점차 줄어들고, θ가 0.6보다 클 경우 역전되어 최대 8% 정도 응답시간이 길어지는 것을 확인할 수 있다. 이는 편향된 엑세스 패턴으로 인해 발생하는 충돌을 해결하기 위해 AOCCRB에서 임의 백오프가 사용되었기 때문이다. 이는 그림 9, 그림 10의 상향, 하향 대역폭 사용량에서 확인할 수 있다.

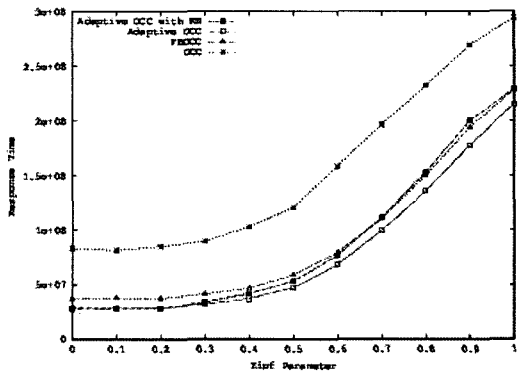

그림 8 접근 패턴의 편향 정도에 따른 평균 응답 시간의 변화

5.1.2 트랜잭션의 접근 패턴의 편향 정도에 따른 통신 대역폭 사용량의 변화

그림 9, 그림 10을 보면 θ가 0.6보다 클 경우 AOCC, OCC, 그리고 FBOCC의 경우 통신 대역폭 사용량이 급격하게 증가하는 것을 확인할 수 있다. 이는 편향된 엑세스 패턴으로 인해 반복적인 재실행이 발생하기 때문

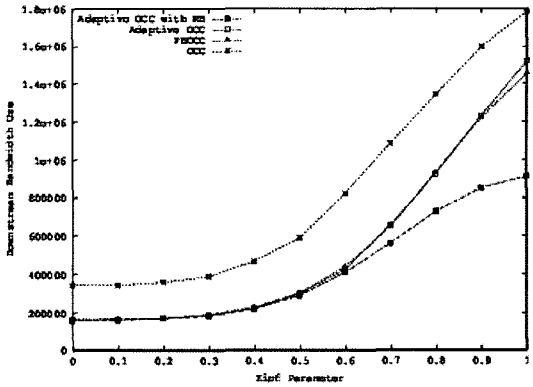

그림 9 접근 패턴의 편향 정도에 따른 평균 하향 대역폭 사용량의 변화

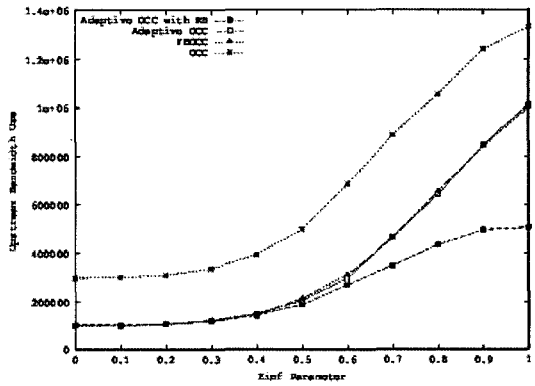

그림 10 접근 패턴의 편향 정도에 따른 평균 상향 대역폭 사용량의 변화

이다. 이러한 통신 대역폭 사용은 모바일 클라이언트의 배터리 소모로 이어지게 된다. 반면 AOCCRB의 경우 편향된 엑세스 패턴에도 통신 대역폭 사용량의 증가가 완만함을 확인할 수 있다. 그림 9에서 θ가 0.95일 경우 AOCCRB는 AOCC와 FBOCC보다 40% 정도, 그리고 OCC보다 45% 정도 하향 통신 대역폭을 적게 사용한다. 그리고 그림 10에서 θ가 0.95일 경우 AOCCRB는 AOCC와 FBOCC보다 30% 정도, 그리고 OCC보다 60% 정도 상향 통신 대역폭을 적게 사용한다. 이는 우리가 제안한 임의 백오프 기법이 데이타에 대한 경합을 효과적으로 해결하여 편향된 엑세스 패턴에도 효과적으로 적응함을 보여준다. 그림 9, 그림 10에서 θ가 0.6보다 작을 경우 AOCCRB, AOCC, FBOCC가 거의 비슷한 통신 대역폭 사용량을 가지는 것을 볼 수 있는데, 이는 균일한(flat) 엑세스 패턴으로 인해 트랜잭션 간의 충돌이 별로 발생하지 않기 때문이다.

5.2 서버 트랜잭션의 도착 빈도에 따른 성능

서버 트랜잭션의 도착 빈도가 많아질수록 모바일 트랜잭션의 충돌 가능성은 높아지게 되므로 동시성 제어 기법의 성능에 영향을 미치게 된다. 우리는 106 비트 시간 동안 도착하는 서버 트랜잭션의 수를 증가시키면서 세 동시성 제어기법의 성능을 분석한다. 균일한 엑세스 패턴과 편향된 엑세스 패턴에서 각각 비교하기 위해 θ가 0.0일 때 그리고 0.95일 때 각각에 대해서 실험을 하였다.

5.2.1 서버 트랜잭션의 도착 빈도에 따른 응답 시간의 변화

그림 11은 서버 트랜잭션의 도착 빈도에 따른 평균 응답 시간의 변화를 보여준다. AOCCRB, AOCC, FBOCC는 상대적으로 서버 트랜잭션의 증가로 인해 발생하는 충돌에 잘 적응하는 반면, OCC는 급격하게 응답시간이 길어지는 것을 볼 수 있다. θ가 0.0일 때 AOCCRB와 AOCC는 FBOCC보다 최대 25% 정도, 그리고 OCC보다는 최대 60% 정도 짧은 응답 시간을 보인다. AOCCRB와 AOCC가 비슷한 응답시간을 보이는 것은 균등한 엑세스 패턴 하에서는 임의 백오프 기법이 거의 작용하지 않기 때문이다. θ가 0.95일 때 AOCC는 AOCCRB와 FBOCC보다 최대 15% 정도, 그리고 OCC보다 최대 40% 정도 짧은 응답 시간을 보인다. 이는 우리가 제안하는 프리패칭 기법이 서버 트랜잭션의 도착 빈도가 증가하더라도 효과적으로 동작하는 것을 알 수 있다.

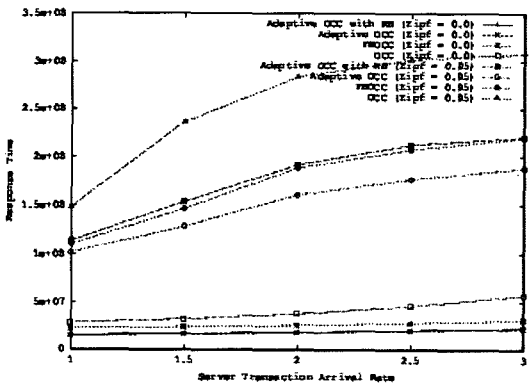

그림 11 서버 트랜잭션의 도착 빈도에 따른 평균 응답 시간의 변화

5.2.2 서버 트랜잭션의 도착 빈도에 따른 통신 대역폭 사용량의 변화

그림 12와 그림 13은 서버 트랜잭션의 도착 빈도에 따른 평균 상향, 하향 대역폭 사용량의 변화를 보여준다. θ가 0.0일 때 AOCC, AOCCRB, FBOCC는 거의 비슷한 사용량을 보이고, 그 중 FBOCC가 나머지 두

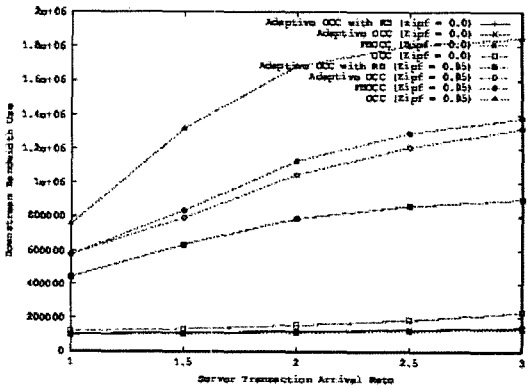

그림 12 서버 트랜잭션의 도착 빈도에 따른 평균 하향 대역폭 사용량의 변화

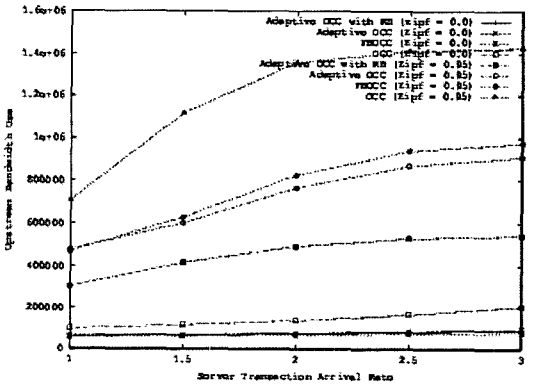

그림 13 서버 트랜잭션의 도착 빈도에 따른 평균 상향 대역폭 사용량의 변화

기법에 보다 5% 정도 적은 사용량을 보인다. AOCC와 AOCCRB가 프리패칭을 통해서 FBOCC에 비해 30% 정도 응답 시간을 감소시키므로 이는 감수할만한 수준이다. θ가 0.95일 때 서버 트랜잭션의 도착 빈도가 증가에 가장 잘 적응하는 기법은 역시 AOCCRB이다. 임의 백오프 기법을 통해서 엑세스를 분산시키므로 AOCCRB는 FBOCC나 AOCC에 비해 최대 45%, 그리고 OCC에 비해 최대 60%의 적은 대역폭 사용량을 보인다. 이러한 상향, 하향 통신 대역폭 사용의 감소는 모바일 클라이언트의 배터리 소모의 감소로 이어진다.

5.3 접근 불변성의 수준에 따른 성능

우리가 제안한 AOCC, AOCCRB는 모바일 트랜잭션이 재실행될 때, 이전 수행 때 얻은 정보를 바탕으로 프리패칭을 함으로써 응답시간을 단축시킨다. 이는 Franaszek가 제안한 접근 불변성(access invariance)의 아이디어에 기반하고 있다[8]. 그러나 Franaszek는 실제 접근 불변성의 정도는 실행 시 트랜잭션의 매개변수나 데

이타베이스의 상태에 따라서 달라질 수 있다는 점을 지적했다[8]. 따라서 그러므로 우리는 접근 불변성의 정도에 따라 우리가 제안한 기법의 성능이 어떻게 달라지는지 분석한다.

접근 불변성의 정도를 모델링하기 위해 우리는 확률적으로 트랜잭션의 엑세스 패턴을 새롭게 생성하였다. 접근 불변성의 정도는 트랜잭션이 재실행시 엑세스 하는 데이타 아이템의 집합이 기존의 데이타 아이템의 집합과 같을 확률을 의미한다.

5.3.1 접근 불변성의 수준에 따른 응답 시간의 변화

그림 14는 접근 불변성의 정도에 따른 평균 응답 시간의 변화를 보여준다. 접근 불변성의 정도가 높을수록 평균 응답시간이 짧아지고, 접근 불변성의 정도가 낮을수록 평균 응답시간이 길어지는 것을 알 수 있다. 이는 접근불변성의 정도가 낮으면 낮을수록 모바일 클라이언트 내의 버퍼에 있는 데이타를 활용하지 않을 확률이 높아지기 때문이다. 균일한 엑세스 패턴에서는 AOCCRB가, 편향된 엑세스 패턴에서는 FBOCC가 가장 짧은 응답시간을 보인다. 그리고 편향된 엑세스 패턴보다는 균일한 엑세스 패턴을 따를 때 접근 불변성의 정도가 낮아짐에 따라 평균 응답시간의 증가가 심해지는 것을 확인할 수 있다. 이는 편향된 엑세스 패턴을 따를 경우 모바일 클라이언트의 버퍼에 있는 데이타 아이템을 다시 엑세스할 확률이 높아지기 때문이다.

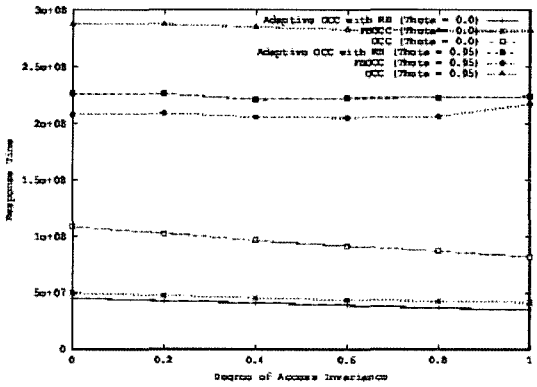

그림 14 접근 불변성의 수준에 따른 평균 응답 시간의 변화

5.3.2 접근 불변성의 수준에 따른 통신 대역폭 사용량의 변화

그림 15는 접근 불변성의 수준에 따른 평균 하향 대역폭 사용량의 변화를 보여준다. 우리가 제안한 AOCCRB는 접근 불변성을 기반으로 데이타 아이템에 대한 프리패칭을 함으로써 트랜잭션 응답 시간의 단축을 꾀

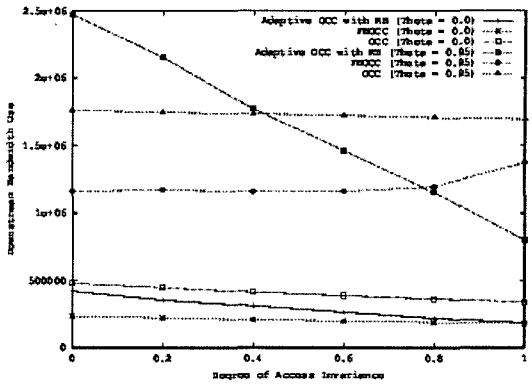

그림 15 접근 불변성의 수준에 따른 평균 하향 대역폭 사용량의 변화

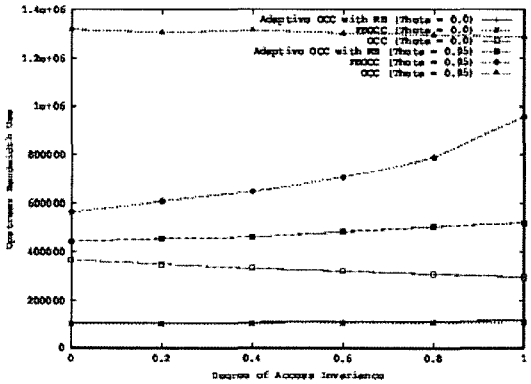

그림 16 접근 불변성의 수준에 따른 평균 상향 대역폭 사용량의 변화

한다. 따라서 접근 불변성의 정도가 낮아지면 유효하지 않은 프리패칭을 하게 되므로 평균 하향 대역폭 사용량이 증가하게 된다. 높은 수준의 접근 불변성에서는 AOCCRB가 가장 적은 사용량을 보이지만, 접근 불변성이 낮아지면 AOCCRB의 경우 하향 대역폭 사용량이 증가함을 알 수 있다. 그림 16은 접근 불변성의 수준에 따른 평균 상향 대역폭 사용량의 변화를 보여준다. 접근 불변성이 높아지면 평균 상향 대역폭 사용량이 증가하는 것을 알 수 있는데, 이는 버퍼에 이미 존재하는 데이타를 활용하여 빠르게 수행할 수 있으므로 더 자주 재실행을 시도하게 되기 때문이다. 균일한 접근 패턴에서는 기법들 간의 상향 대역폭 사용량의 차이가 별로 없지만, 편향된 접근 패턴에서는 AOCCRB가 가장 적은 사용량을 보인다. 이는 AOCCRB의 경우 백오프를 사용하여 연속적인 재실행을 피하기 때문이다.

6. 결 론

본 논문에서 우리는 무선 브로드캐스트 환경에서 모바일 트랜잭션의 효과적인 처리를 위한 동시성 제어 기법을 제안하였다. 지금까지 제안된 방법들은 읽기 전용 트랜잭션에만 적용이 가능하거나, 버퍼 유지 효과를 제대로 반영하지 못하거나, 혹은 편향된 엑세스 패턴에서 반복적인 트랜잭션 중단으로 심각한 성능 저하를 보였다. 이러한 문제들을 해결한 우리의 제안하는 기법은 읽기 전용 트랜잭션과 갱신 트랜잭션 모두에 적용이 가능하다. 읽기 전용 트랜잭션은 서버와의 통신이 없이 지역적으로 커미트 가능하고, 갱신 트랜잭션은 정방향, 역방향 검증을 기본으로 사용하여 서버와의 불필요한 통신을 줄였다. 그리고 클라이언트의 편향된 엑세스 패턴으로 인해 발생할 수 있는 반복적인 재실행을 문제점으로 인식하고 임의 백오프 기법을 이용해 이를 해결함으로써 클라이언트의 엑세스 패턴에 유연하게 적응한다. 또한 트랜잭션은 수행 시 엑세스 패턴을 파악하여, 재실행 시 프리패칭할 수 있도록 함으로써 클라이언트 측의 버퍼 유지 효과를 극대화한다.

기존의 낙관적 동시성 제어 기반 기법들과 비교를 통해서 여러 가지 조건 하에서 본 논문에서 제안한 기법의 성능을 분석하였다. 시뮬레이션 결과, 상향, 하향 통신 대역폭 사용량과 트랜잭션의 평균 응답 시간이라는 세 가지 성능 지표 모두에서 기존 기법에 비해서 좋은 성능을 보였고, 기존 기법과 달리 엑세스 패턴의 편향성에도 잘 적응하는 것을 확인하였다.

참 고 문 헌

[1] S. Acharya, M. Franklin, S. Zdonik, and R. Alonso, *"Broadcast Disks: Data Management for Asymmetric Communication Environments,"* Proc. ACM SIGMOD International Conference on Management of Data, pp. 199-210, 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik, *"Balancing Push and Pull for Data Broadcast,"* Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 183-194, 1997.

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

[4] H. Cho, "Concurrency Control for Read-Only Client Transactions in Broadcast Disks," IEICE Trans. Commun., vol.E86-B, no.10, 2003.

[5] V. Lee, K-W. Lam, T-W Kuo, *"Efficient validation of mobile transactions in wireless environments,"* The Journal of Systems and Software, pp.183-193, 2004.

[6] Shanmugasundaram, J., Nithrakashyap, A., Sivasankaran, R., Ramamritham, K., *"Efficient concurrency control for broadcast environments,"* In: ACM SIGMOD International Conference on Management of Data, 1999.

[7] E. Pitoura, P. K. Chrysanthis, *"Scalable Processing of Read-Only Transactions in Broadcast Push,"* ICDCS, Austin, TX, 432439, 1999.

[8] P. A. Franaszek, J. T. Robinson, and A. Thomasian, *"Access invariance and its use in high contention environments,"* IBM Res. Rep. RC 14704, 1989.

[9] P. A. Bernstein, Va. Hadzilacos, N. Goodman, *"Concurrency Control and Recovery in Database Systems,"* Addison Wesley, Massachusetts, 1987.

[10] Kung, H. T., Robinson, J. T., *"On optimistic methods for concurrency control,"* ACM Transactions on Database Systems 6(2), pp. 213-226, 1981.

[11] Lee, V.C.S., Kwok-Wa Lam, Son, S.H., *"On transaction processing with partial validation and timestamp ordering in mobile broadcast environments,"* IEEE Transactions on computers, 51, 10, pp. 1196-1211, 2002.

[12] Il Young Chung, Bhargava, B., Mahoui, M., Lilien, L., *"Autonomous transaction processing using data dependency in mobile environments,"* Distributed Computing Systems, 2003. FTDCS 2003. Proceedings. The Ninth IEEE Workshop on Future Trends of, 28-30, pp 138-144, 2003.

[13] Haerder, T., *"Observations on optimistic concurrency control schemes,"* Information Systems 9 (2), 1984.

[14] Yu, P. S., DIAS, D. M., *"Analysis of hybrid concurrency control schemes for a high data contention environment,"* IEEE Transactions on Software Engineering, 18, 2, pp. 118129, 1992.

[15] P. S. Yu, D. M. Dias, *"Impact of large memory on the performance of optimistic concurrency control schemes,"* in Proc. Int. Conf. on Databases, Parallel Architectures, and their applications (PARBASE-90), (Miami Beach, FL), pp. 86-90, Mar. 1990.

[16] P. S. Yu, D. M. Dias, S. S. Lavenberg, *"On modeling database concurrency control,"* IBM. Yorktown Heights, NY, Res. Rep. RC 15386, 1990.

[17] H. D. Schwetman, *"CSIM: A C-based Process Oriented Simulation Language,"* Proceeding 1986 Winter Simulation Conference, 1986.

[18] Mesquite Software, Inc, *CSIM18 Simulation Engine USER'S GUIDE*, 1987-1994.

[19] D. Knuth, *"The Art of Computer Programming Second Edition, Vol III,"* Addison Wesley, 1998.

[20] G. K. Zipf, *"Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology,"* Addison Wesley Press, Cambridge, Massachusetts, 1949.

[21] J. Gray, P. Sundaresan, S. Englert, K. Baclawski and P. Weinberger, *"Quickly Generating Billion-*

record Synthetic Databases," Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, 1994.

[22] T. Imielinski, S. Viswanathan, B. Badrinath. *"Data on air: Organization and acces,"* IEEE Transactions on Knowledge and Data Engineering, 9(3):353-372, 1997.

[23] S. K. Lee, C. S. Hwang, Kitsuregawa, M. *"Using predeclaration for efficient read-only transaction processing in wireless data broadcast,"* IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on, 15(6): 15791583, 2003.

정 성 원

1988년 서강대학교 전자계산학 학사 1990년 M.S. in Computer Science at Michigan State Univ. 1995년 Ph.D. in Computer Science at Michigan State Univ. 1997년~2000년 한국전산원 선임연구원. 2000년~현재 서강대학교 컴퓨터학과 부교수. 관심분야는 Mobile Computing Systems, Mobile Databases, Telematics, Spatial DB, Mobile Agents, Streaming Data Processing in Ubiquitous Computing Environments, Distributed Databases

박 성 근

1999년 3월~2003년 2월 서강대학교 컴퓨터학과 학사. 2003년 3월~2005년 2월 서강대학교 컴퓨터학과 석사. 2005년 3월~현재 SK INNOACE Terminal Solution Team 연구원. 관심분야는 이동통신, 모바일컴퓨팅, 모바일데이타베이스, 모바일 트랜잭션

최 근 하

1997년 3월~2004년 2월 서강대학교 컴퓨터학과 학사. 2004년 3월~현재 서강대학교 컴퓨터학과 석사과정. 관심분야는 이동통신, 모바일 컴퓨팅 시스템, 모바일 데이타베이스, 모바일 트랜잭션, 모바일 에이전트