재순서화 기반 이동 트랜잭션 스케쥴링 알고리즘
(Reordering-based Mobile Transaction Scheduling Algorithms)

김 성 석 † 이 상 근 †† 황 종 선 †††
(SungSuk Kim) (SangKeun Lee) (Chong-Sun Hwang)

요 약 무선 통신 기술 및 이동 컴퓨터의 성능이 발전함에 따라, 이동 컴퓨팅 환경을 단순한 단말기로 사용하기보다는 독립된 수행 단위로 이용하려는 연구가 활발히 진행되고 있다. 즉 모든 작업을 서버에게 보내어 결과를 기다리는 대신 이동 컴퓨터에서 직접 수행할 수 있는 알고리즘이 개발되고 있다. 본 논문에서는 이동 컴퓨터에서의 트랜잭션 수행 알고리즘을 제안한다. 우리는 낙관적인 기법을 채택하였는데, 이 기법은 (a) 동기화에 필요한 메시지를 적게 필요로 하며, (b) 브로드캐스트 기법의 장점을 얻을 수 있다. 브로드캐스트 기법은 최근 많은 수의 클라이언트에게 정보를 전송할 수 있는 수단으로써 많이 연구되고 있다. 그러나 접근하는 데이타간에 충돌이 빈번하게 발생한다면, 낙관적 기법은 결국 높은 철회율을 보이게 되며, 이는 다시 이동 컴퓨터 자원을 효율적으로 사용하려는 목적과 반대 결과가 된다. 본 논문에서는 이러한 높은 철회율을 감소시키기 위해 재순서화 개념을 도입하였다. 즉, 충돌이 발견될 경우 데이타의 일관성을 침해하지 않는 범위내에서 충돌 순서를 결정하도록 하는 것이다. 특히 읽기-쓰기 트랜잭션에 대해서는 *후위-재순서화 개념(O-Post)*을, 읽기-전용 트랜잭션에 대해서는 *전위-재순서화 개념(O-Pre)*을 제안하였으며, 실험에 의하여 성능 향상을 보였다.

Abstract With the advances in wireless communication and its related technologies, there are a lot of research efforts which intend to utilize mobile computers as a processing unit, rather than just a terminal. That is, the algorithms with which mobile users can execute their jobs have been developed. In this paper, we propose efficient transaction processing algorithms in the mobile computing environment. In particular, we take an optimistic approach because (a) it needs less number of messages for maintaining transactional consistency, and (b) it makes use of broadcasting facilities from the servers. The more the data conflicts occur in the optimistic approach however, the more mobile transactions are aborted, thereby resulting in the inefficient use of limited resources of mobile computing environments. Therefore, we also accept a reordering technique to reduce the possibility of aborts. When a mobile computer finds conflicts, it determines the operation orders semantically so that the order may not violate the data consistency. Considering the type of conflicts, we devise both *Post-Reordering* algorithm for update transactions and *Pre-Reordering* algorithm for read-only transactions. We also evaluate the performance behavior through simulation study.

1. 서 론

트랜잭션 처리 시스템의 성능은 동시성 제어 기법에 큰 영향을 받는다. 잠금 (locking) 기법은 트랜잭션이 연산을 수행하기 전에 필요한 잠금을 얻어야 하며, 모든 연산이 종료하게 되면 얻었던 모든 잠금을 해제하도록 하는 방식이다. 낙관적 동시성 제어(Optimistic Concurrency Control, *OCC*) 기법은 낙관적이라는 이름에서 알 수 있듯이 연산을 수행할 때 어떠한 조건 검사도 하지 않는다. 대신 모든 연산을 수행한 종료시점에서 그 수행이 올바른지를 검사하도록 하는 것이다. 이와 같은 잠금 기법 및 낙관적 기법 외에도 다양한 방법-에를 들면 타임스탬프 기법, 직렬화 그래프 검사 기법 등-이

† 학생회원 : 고려대학교 컴퓨터학과
sskim@disys.korea.ac.kr
†† 비 회 원 : 고려대학교 기초과학연구소 연구원
lsk@disys.korea.ac.kr
††† 종신회원 : 고려대학교 컴퓨터학과 교수
hwang@disys.korea.ac.kr
논문접수 : 2000년 1월 27일
심사완료 : 2001년 4월 12일

제안되었다[1]. 현재 개발되어서 사용중인 상당수의 시스템은 동시성 제어 기법으로서 잠금 기법을 이용하고 있다. 이는 데이타에 대한 다양한 접근 경향에 대하여, 다른 기법보다 비교적 일정한 성능을 보여주기 때문이다. 하지만 이동 컴퓨팅 환경으로 변환한다면 잠금 기법에서는 여러 가지 고려사항 및 문제점을 지닐 수 있다. 즉 매 연산을 수행할 때마다 잠금을 얻기 위해 많은 수의 메시지를 교환해야 하며, 교착상태(deadlock)의 발생으로 인한 부하가 커진다. 하지만 이동 컴퓨팅 환경의 제약조건은 그와 같은 부하를 허용할 수 없다.

본 논문에서는 이동 분산 컴퓨팅 환경에서의 트랜잭션 스케쥴링 알고리즘을 개발하고자 한다. 이동 컴퓨팅 환경의 특징은 이전의 분산 환경과는 달리 다음과 같은 새로운 특징 및 제약조건을 가지고 있다 [2]:

- 무제한적인 이동성
- 제한된 자원 (대역폭, 전력 등)
- 빈번한 접속 단절
- 안정성 부족

이동 컴퓨팅 환경은 특히 제한된 자원을 효율적으로 이용할 수 있는 알고리즘을 필요로 하며, 따라서 트랜잭션 수행동안 많은 수의 메시지를 필요로 하지 않은 낙관적 처리 기법이 이동 컴퓨팅 환경에 적합해 보인다. 특히 최근에는 서버에서 전송하는 브로드캐스팅 정보를 이용하여 수행중인 이동 트랜잭션의 일관성 검사를 수행하는 알고리즘들이 많이 제안되었다[3, 4]. 즉, 낙관적 동시성 제어 기법과 주기적인 브로드캐스트 정보를 함께 이용할 경우, 트랜잭션이 수행하는 중에도 지금까지 트랜잭션 수행의 정당성을 검증할 수 있다. 하지만 만약 트랜잭션들 간에 충돌이 빈번하게 발생한다면, 수행 중 혹은 완료를 요청한 후에 철회되는 회수가 증가하게 된다. 이는 이동 컴퓨팅 환경의 제한된 자원의 효율적인 이용에 반하는 결과가 된다. 이에 대하여 본 논문에서는 순수한 낙관적 동시성 제어 기법의 높은 철회율을 줄여서, 낙관적 수행의 장점을 얻을 수 있는 알고리즘을 개발하고자 한다. 즉 이동 트랜잭션의 연산을 수행하는 중에 충돌이 발생할 경우 무조건 철회시키는 대신 데이타의 일관성을 침해하지 않은 범위에서 충돌 순서를 결정한 후 이와 같이 결정된 직렬화 순서에 의해 계속 트랜잭션을 수행시키도록 하는 것이다. 트랜잭션이 읽기 및 쓰기 연산을 모두 가지고 있는 경우에는 *후위-재순서화(post-reordering)* 개념을 적용하며, 읽기 전용 트랜잭션인 경우에는 *전위-재순서화(pre-reordering)* 개념을 적용하여 이동 트랜잭션의 충돌 순서를 결정한다. 이러한 재순서화 기법을 이용한다면 다음과 같은 장점을 얻을 수 있다 :

- 이동 트랜잭션은 낙관적인 방식으로 수행하면서 철회율을 감소시킬 수 있으므로, 이동 컴퓨터의 자원을 효율적으로 이용할 수 있다.
- 이동 트랜잭션의 연산 수행은 완전히 이동 컴퓨터 내에서만 수행되며 서버는 검증 및 연산 수행을 위한 브로드캐스트 전송 역할만 한다. 따라서 서버의 이동 트랜잭션 수행에 대한 부하를 줄일 수 있어서 확장성이 좋다.

본 논문의 구성은 다음과 같다 : 2장에서는 대상이 되는 시스템 모델 및 재순서화 개념에 대하여 설명한다. 3장에서 재순서화 개념을 이용한 두 가지 알고리즘을 설명한다. 알고리즘은 후위-재순서화를 이용한 경우와 전위-재순서화를 이용한 경우로 나뉘어 진다. 4장에서는 제안한 알고리즘에 대한 성능 평가 결과를 보여준다. 5장에서는 관련 연구를 설명하며, 6장에서는 결론 및 향후 과제를 정의한다.

2. 시스템 모델 및 재순서화

2.1 시스템 모델

본 논문에서 가정하는 시스템 모델은 다음 그림 1과 같다:

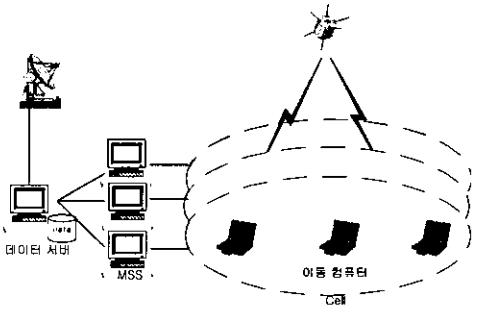


그림 1 시스템 모델

이동 컴퓨팅 환경은 크게 유선망과 무선망으로 구성되어 있다. 유선망에는 데이타 서버 및 이동 컴퓨터의 업무를 지원하는 기능을 가진 *이동기지국(MSS, Mobile Support System)*이 있다. 데이타 서버는 기본적으로 데이타(*D*)를 관리하면서, 사용자의 데이타 요구를 처리한다. 또한 이동 컴퓨터 사용자들에게 필요한 제어 정보를 브로드캐스트하는 기능도 수행한다. 본 논문에서는 풀-기반 주기적 브로드캐스트 모델을 가정한다. 데이타 서버는 이전 주기동안 서버에서 완료한 트랜잭션에 의해 수정된 데이타 식별자 및 서버에게 완료를 요청한 트랜

잭션의 검증 결과를 주기적으로 전송한다. 제안하는 시스템 모델에서는 매 L초마다 한번씩 브로드캐스트 BC_i를 전송한다. 이때 i는 브로드캐스트의 인식자라고 하며, 이는 시작부터 경과 시간($t_i = i \times L$)을 의미한다(단 i는 정수이며, 1씩 증가한다). 따라서 t_i는 전송되는 브로드캐스트에 추가되는 정보가 수행된 시각이 된다.

BC_i = { *Update, Commit, Abort* }

Update : 시각 t_{i-1}과 t_i 동안 수정된 데이타 식별자

Commit : 이전 주기동안 서버에서 완료한 트랜잭션의 인식자

Abort : 이전 주기동안 서버에게 완료를 요청하였으나 철회된 트랜잭션의 인식자

이동 기지국은 이동 클라이언트들과 데이타 서버 사이에 위치하여 중계 역할을 담당한다. 즉 이동 기지국이 담당하는 각 셀에 있는 이동 컴퓨터에게 필요한 정보를 전송하거나, 이동 컴퓨터가 서버에게 보내는 요청 메시지를 전송하는 역할을 수행한다. 따라서 이동 기지국의 역할은 알고리즘 수행에서 특별한 의미를 가지지 않는다고 가정한다.

무선망을 이용하는 이동 컴퓨터의 사용자들은 제한된 자원의 시스템을 이용하여 *이동 트랜잭션(mobile transaction)*을 수행한다. 즉 트랜잭션 수행에 필요한 데이타들을 서버로부터 받아서 연산들을 지역적으로 수행한다. 모든 연산이 수행되면 수행 결과와 함께 완료 연산을 서버에게 보내어 검증 받도록 한다. 서버에서의 검증 결과의 전송은 주기적인 브로드캐스트를 이용한다.

본 논문에서는 시스템 모델을 간단히 하기 위해 다음과 같은 가정을 한다:

- 단일 서버를 가진다. 즉 서버간에 데이타 복제에 관련된 문제는 없다.
- 이동 컴퓨터의 접속 단절이나 이동성으로 인한 문제는 발생하지 않는다.
- 이동 컴퓨터는 캐쉬 데이타를 유지하지 않는다. 따라서 필요한 모든 데이타는 서버에게 요청하도록 한다.

이와 같은 시스템 모델에 대한 기본적인 가정을 바탕으로 하여, 본 논문에서는 이동 컴퓨터의 제한된 자원을 효율적으로 이용하여 트랜잭션을 낙관적으로 수행할 수 있는 알고리즘을 개발한다.

2.2 재순서화(Reordering)

일반적으로 트랜잭션 처리 시스템에서 독립적으로 수행되는 트랜잭션들 간에는 임의의 선후 순서관계가 없다고 가정한다. 즉 여러 사용자로부터 제기되는 트랜잭션들은 공유 데이타를 이용하여 완전히 독립적인 작업을 수행하는 것이다. 이와 같이 트랜잭션들이 병행 수행되더라도 일관성을 유지하기 위한 많은 기준이 있으며, 충돌 관계를 고려한 *직렬화가능성(serializability)*이 정당성의 기준으로서 일반적으로 채택된다[1].

직렬화가능성을 유지하기 위한 여러 기법들이 제안되었는데, 본 논문에서는 *낙관적 동시성 제어 기법(OCC)*을 이용한다. 사용자들은 어떠한 지연이나 제약조건 없이 트랜잭션의 연산들을 지역적으로 수행한 후, 그 수행의 정당성을 서버에게 검증받도록 하는 것이다. 이 기법은 다른 기법에 비하여 수행 중에 필요한 동기화 메시지의 수가 적으므로, 이동 컴퓨팅 환경에 적합해 보인다.

그러나, 시스템에서 접근 데이타 간에 충돌 비율이 높아질 경우 트랜잭션의 철회율(abort ratio) 역시 높아지게 된다. 따라서 철회율을 감소시키기 위해 우리는 *재순서화(reordering)* 개념을 이용한다[5].

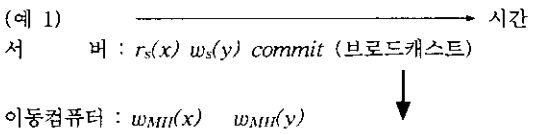
트랜잭션 T_i의 연산 $p(x)$와 트랜잭션 T_j의 연산 $q(y)$에 대하여($i \neq j$) x와 y가 동일한 데이타이며, p와 q중 적어도 하나가 쓰기 연산(w)일 경우 충돌이 발생하였다고 한다. 충돌의 종류는 w-w, w-r, r-w가 있으며, 본 논문에서 표현하는 a-b 충돌이란 서버에서 수행되고 완료된 트랜잭션이 a 연산을 수행하였으며, 이동 컴퓨터에서 수행중인 트랜잭션이 b 연산을 수행하였고, a와 b는 충돌관계인 연산임을 의미한다. 이동 컴퓨터에서 수행중인 트랜잭션과 서버에서 완료한 트랜잭션들 간에 충돌이 발견될 경우, 연산 수행 시간에 의하여 직렬화 순서를 결정해야 한다. 하지만 분산된 위치에서 수행되는 트랜잭션의 연산 수행에 대하여 시간적인 선후관계를 결정하는 것은 곤란하다[8]. 따라서 충돌이 발견될 경우, 접근한 데이타의 일관성을 유지할 수 있도록 상대적인 연산 수행 순서를 결정해야 한다.

만약 주기적인 브로드캐스트 정보를 이용한다면, 충돌에 대한 상대적인 수행 순서를 결정할 수 있다. 즉 브로드캐스트 정보에는 이전 주기동안 서버에서 완료된 트랜잭션이 수정한 데이타 식별자-브로드캐스트의 *Update*-를 포함되어 있다. 따라서 이동 컴퓨터가 이 정보를 받게 되면, 지역적으로 수행중인 트랜잭션이 접근한 데이타와 *Update*를 비교하여 공집합이 아니라는 사실로부터 충돌 발생을 알 수 있으며, 재순서화 개념을 이용하여 충돌 순서를 결정한다.

2.2.1 후위-재순서화 (Post-Reordering)

서버에서 수행되고 완료한 트랜잭션(Ts)들이 접근(읽기 혹은 쓰기 연산을 수행)한 데이타와 이동 컴퓨터에

서 수행중인 트랜잭션(TMH)의 쓰기 연산간에 충돌이 발생할 경우, 철회율을 감소시키기 위해 후위-재순서화 개념을 적용한다. 다음의 예 1을 살펴보자.



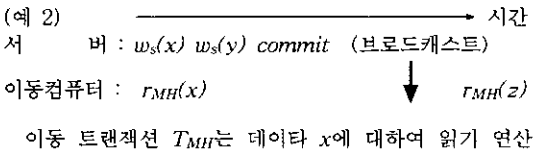
이동 트랜잭션 T_{MH}는 쓰기 연산을 수행하였고, 이는 서버에서 수행하고 완료한 트랜잭션 T_S과 충돌을 발생시킨다. 하지만 이러한 r-w 혹은 w-w 충돌을 발견할 경우, 데이타의 의미를 고려할 때 이동 컴퓨터는 $T_S \rightarrow T_{MH}$와 같이 충돌 순서를 결정할 수 있다. 이와 같이 결정된 충돌 순서는 다음과 같은 의미를 가진다: 이동 트랜잭션에서 r-w 혹은 w-w 충돌은 고려하지 않아도 된다. 왜냐하면 수행중인 이동 트랜잭션의 직렬화 순서는 결국 서버에서 완료한 트랜잭션(T_S)보다 늦어지기 때문이다.

(정의 1) 이동 컴퓨터가 주기적인 브로드캐스트 정보를 받은 후, r-w 혹은 w-w 충돌을 발견한다면, 이동 트랜잭션이 서버에서 수행하고 완료한 트랜잭션보다 이후에 수행되었다고 결정한다. 이를 *후위-재순서화(post-reordering)*라고 한다.

하지만 w-r 충돌에 대하여 이러한 후위-재순서화 개념을 적용한다면 데이타의 일관성을 침해할 수 있다. 즉 이동 트랜잭션이 읽은 데이타가 서버에서 완료한 트랜잭션이 수정한 데이타보다 이전 버전일 수 있기 때문이다. 따라서 w-r 충돌이 발견된다면 이동 트랜잭션을 철회시킨다.

2.2.2 전위-재순서화 (Pre-Reordering)

후위-재순서화에서는 이동 트랜잭션의 읽기 연산에 대하여 w-r 충돌이 발생할 경우 무조건 철회시켰다. 그러나 이동 컴퓨터에서 수행되는 응용프로그램들은 읽기-전용인 경우-- 예를 들면, 주식, 날씨, 교통 정보 등 -가 많으며, 따라서 빈번하게 w-r 충돌이 발생한다면 철회율 역시 높아진다. 따라서 읽기-전용 트랜잭션인 경우에는 전위-재순서화 개념을 적용하여 철회율을 줄이도록 한다. 다음의 예 2를 살펴보자:



이동 트랜잭션 T_{MH}는 데이타 x에 대하여 읽기 연산을 수행하였다. 하지만 이 데이타는 서버에서 수행하고 완료한 트랜잭션 T_S가 수정하였으므로, 브로드캐스트 정보(BC_i)로 부터 w-r 충돌이 발생하였음을 알 수 있다. 하지만 데이타의 의미를 고려해볼 때, $r_{MH}(x)$가 서버의 쓰기 연산보다 먼저 수행되었다고 결정할 수 있는 경우가 있다. 즉 서버에서 가장 최근의 브로드캐스트(BC_{i-1}) 이후 완료한 트랜잭션의 쓰기 결과를 이동 트랜잭션이 읽지 않았다면, $T_{MH} \rightarrow T_S$와 같이 충돌 순서를 결정할 수 있다. 이를 위하여 서버는 이동 트랜잭션이 읽기 연산을 위해 데이타 x를 요구할 경우, 데이타 x가 가장 최근의 브로드캐스트 이후 완료한 트랜잭션에 의해 수정된 - 따라서 다음 주기의 브로드캐스트 정보에 포함될 - 데이타인지를 검사한다; 그런 경우에는 데이타 값과 함께 *msgAbortable* 메시지를 함께 보내준다. 이 메시지는 이동 트랜잭션에서 w-r 충돌이 발생할 경우 전위-재순서화할 수 없음을 의미한다. 이후의 읽기 연산이 결정되어진 충돌 순서를 위배하지 않도록 수행된다면 직렬화가능한 수행 결과를 얻을 수 있다.

(정의 2) 이동 컴퓨터가 주기적인 브로드캐스트 정보를 받은 후 읽기 전용 트랜잭션에서 w-r 충돌이 발생한다면, 이동 트랜잭션이 서버에서 수행하고 완료한 트랜잭션보다 충돌 순서가 앞선다고 결정한다. 이를 *전위-재순서화(pre-reordering)*라고 한다.

물론 정의 2에서 이동 트랜잭션이 *msgAbortable* 메시지를 받지 않은 경우에만 가능하다.

충돌에 의하여 전위-재순서화된 트랜잭션은 이후 연산 수행에 대하여 다소 비관적으로 수행되어야 한다. 즉 서버에서 이미 완료한 트랜잭션보다 직렬화 순서가 앞서도록 재순서화 되었으므로, 이후의 읽기 연산이 그러한 재순서화된 직렬화 순서를 위배하지 않도록 검사를 수행하여야 한다.

3. 낙관적 스케쥴링 알고리즘

3.1 후위-재순서화 기반 낙관적 스케쥴링 (O-Post : Optimistic Scheduling based on Post- Reordering)

본 논문의 재순서화 개념은 서버에서 주기적으로 브로드캐스트하는 정보를 이용한다. 하지만 브로드캐스트 정보는 무선 채널을 통하여 전송되므로, 전송되는 정보의 양을 가능한 줄이는 것이 좋은 성능을 보일 수 있다. 만약 후위-재순서화 개념을 적용한다면, 이동 트랜잭션에 대하여 r-w 및 w-w 충돌은 검사하지 않아도 된다(2.2.1절 참조). 왜냐하면 재순서화하더라도 그 순간에는 직렬화가능성을 침해하지 않기 때문이다(보조정리 1). 따라서

이동 컴퓨터는 w-r 충돌만 검사하면 된다. 이동 컴퓨터에서 연산 수행에 필요한 데이타는 매번 서버에게 요청하여 얻는다고 가정한다. 캐쉬 데이타를 유지한다면 메시지의 수를 줄일 수 있지만, 캐쉬 데이타의 일관성을 유지하기 위한 별도의 메시지가 필요하다. 이와 관련된 주제는 본 논문의 범위를 벗어나므로 고려하지 않겠다.

보조정리 1 후위-재순서화 개념에서 r-w 및 w-w 충돌은 검사하지 않아도 직렬화가능성을 위배하지 않는다.

증명. r-w 및 w-w 충돌은 이동 트랜잭션 T_{MH}의 쓰기 연산과 관계된다. 만약 이동 트랜잭션 T_{MH}가 수행중 (혹은 완료 연산을 요청한 후) 주기적인 브로드캐스트 정보를 받은 후, 이전 주기동안 서버에서 완료한 트랜잭션 T_{si} ($i \geq 1$)과 r-w 혹은 w-w 충돌이 발생하였다고 가정하자. 트랜잭션 T_{si} ($i \geq 1$)는 이미 완료한 트랜잭션들이며, 이러한 충돌에 대하여 데이타의 의미를 고려할 때 $T_{si} \rightarrow T_{MH}$와 같은 충돌 순서를 결정할 수 있다. 그리고 이동 트랜잭션의 쓰기 결과는 결코 T_{si}에 영향을 주지 않는다. 따라서 후위-재순서화한 결과는 데이타의 일관성을 위배하지 않는다. □

3.1.1 이동 컴퓨터의 알고리즘

이동 컴퓨터는 트랜잭션의 연산 $o(x)$을 낙관적으로 수행한다. 수행 도중 주기적인 브로드캐스트 정보를 받는다면, 단지 w-r 충돌만을 검사한다. w-r 충돌이 발생한 경우에는 철회 후 재시작을 시키고, 그렇지 않은 경우에는 계속 연산을 수행한다. 모든 데이타 연산이 다 수행되면 다음의 정보와 함께 서버에게 완료 요청을 보낸 후 검증 결과를 기다린다:

MobileData={*T_id, ReadSet, WriteSet, BCindex*}

T_id : 트랜잭션의 인식자

ReadSet : 트랜잭션이 읽은 데이타

WriteSet : 트랜잭션이 수정한 데이타 및 새로운 값

BCindex : 수행중 받은 가장 최근의 브로드캐스트 정보의 인식자

완료 결과는 다음 주기의 브로드캐스트에서 알려진다. 구체적인 알고리즘은 표 1과 같다.

3.1.2 서버의 알고리즘

서버는 기본적으로 다음과 같은 역할을 수행한다:

- 이동 트랜잭션이 필요로 하는 데이타 전송
- 이동 트랜잭션의 완료 요청 처리 (검증)
- 주기적인 브로드캐스트 정보 전송

3.1.1절에서 언급하였듯이 주기적인 브로드캐스트 정보에는 재순서화에 필요한 정보가 포함되어야 한다. 이 정보에는 이전 주기동안 완료 요청한 트랜잭션의 결과 및 그 완료한 트랜잭션이 수정한 데이타에 대한 정보로 구성된다(2절 참조).

이동 컴퓨터는 낙관적으로 연산을 수행하므로, 서버는 이동 컴퓨터의 데이타 요구에 대하여 아무런 조건 검사도 수행하지 않는다. 또한 이동 트랜잭션의 완료 연산을 받을 경우에도, 다음 주기의 브로드캐스트 정보에 포함될 트랜잭션의 쓰기 집합과의 w-r 충돌 여부만 조사하면 된다. 즉 이동 트랜잭션이 서버에게 완료를 요청한 이후 서버가 전송한(혹은 전송할) 브로드캐스트 정보(즉 *MobileData.BCindex* 이후 이동 트랜잭션이 받지 못한 모든 브로드캐스트 정보)의 *Update*와 *ReadSet*을 비교한다. 만약 교집합이 *NULL*이 아니라면 철회된다; *NULL*인 경우에는 완료된다.

표 1 *O-Post*의 이동 컴퓨터 알고리즘

```
이동 트랜잭션 T_MH에 대하여
※ 연산 r(x)를(혹은 w(x, new)) 받은 경우
   - 서버에게 데이타 x 값 요청후 연산 수행
   - ReadSet (혹은 WriteSet) ← x (혹은 (x, new))
※ 완료 연산을 받은 경우
   - MobileData와 함께 서버에게 완료 요청
※ 주기적인 브로드캐스트 정보 BC_i를 받은 경우 (3.1.2절 참조)
   if (이동 트랜잭션이 수행중인 경우)
      S ← MobileData.ReadSet ∩ BC_i.Update
      if (S ≠ NULL) abort T_MH ;
      else MobileData.BCindex = i ;
   else // 서버에게 완료를 요청한 경우
      if (BC_i.Commit ∋ T_MH) commit T_MH ;
      else if (BC_i.Abort ∋ T_MH) abort T_MH ;
      else
         다음 주기의 브로드캐스트 정보를 기다림;
```

정리 1 *O-Post*에 의한 수행은 항상 직렬화가능하다.

증명. 이동 트랜잭션에서 w-r 충돌이 발생할 경우 철회된다. r-w 및 w-w 충돌이 발생한다면 충돌 순서는 항상 $T_{si} \rightarrow T_{MH}$가 된다. 또한 이 충돌은 직렬화가능성을 침해하지 않는다 (보조정리 1). 따라서 T_{si}와 T_{MH}를 포함하는 직렬화그래프에서는 $T_{MH} \rightarrow T_{si}$ 충돌 관계가 발생할 수 없으므로, *O-Post*에 의한 수행은 항상 직렬화가능하다. □

3.2 전위-재순서화 기반 낙관적 스케쥴링 (O-Pre : Optimistic Scheduling based on Pre-Reordering)

전위-재순서화는 후위-재순서화에서 w-r 충돌에 의한 철회를 감소시키기 위하여, 읽기-전용 트랜잭션에 적용할 수 있는 기법이다. 2.2.2 절에서 설명하였듯이 읽기

-전용 트랜잭션에서 w-r 충돌이 발생하면, 데이타의 의미를 고려할 때 이동 트랜잭션의 읽기 연산이 먼저 수행되었다고 결정할 수 있는 경우가 있다. 이에 초점을 맞추어서 데이타의 일관성을 지킬 수 있도록 전위-재순서화 기반 스케쥴링 기법을 제안한다.

3.2.1 이동 컴퓨터의 알고리즘

우선 이동 트랜잭션이 읽기 연산을 수행하기 위해 서버에게 데이타를 요청한 후 데이타 값과 함께 *msgAbortable* 메시지를 받는다면, 이는 이후 w-r 충돌이 발생할 경우 전위-재순서화를 할 수 없음을 의미한다. 왜냐하면 그 메시지는 가장 최근의 브로드캐스트 전송 이후 서버에서 완료한 트랜잭션 T_{Si}에 의해 수정된 데이타 값을 이동 트랜잭션이 읽었음을 의미하므로, 충돌순서는 $T_{Si} \rightarrow T_{MH}$가 된다. 따라서 이 메시지를 받게되면 전위 재순서화를 할 수 없게 된다. 수행중인 이동 트랜잭션 T_{MH}가 브로드캐스트 정보로부터 w-r 충돌을 발견한다면 다음과 같이 수행한다:

- -이미 *msgAbortable* 메시지를 받은 경우 : 이동 트랜잭션은 철회된다
- -*msgAbortable* 메시지를 받지 않은 경우 : 전위-재순서화한다. 즉 w-r 충돌에 대하여 $T_{MH} \rightarrow T_{Si}$로 충돌 순서를 결정한다.

철회되는 이유는 다음과 같다: 이동 트랜잭션이 전위-재순서화 한다면 $T_{MH} \rightarrow T_{Sj}$ 충돌관계가 되며, *msgAbortable* 메시지에서 $T_{Si} \rightarrow T_{MH}$가 된다. 이때 T_{Si}와 T_{Sj}가 동일한 트랜잭션이거나 직,간접 충돌관계를 가진다면 직렬화가능성을 위배할 수 있다. 따라서 잘못된 수행을 막기 위해 이동 트랜잭션은 철회된다.

w-r 충돌에 대하여 전위-재순서화 되었다면, 앞으로의 읽기 연산은 비관적으로 수행된다. 즉 매 읽기 연산 $r_{MH}(x)$에 대하여 다음의 조건을 검사하여야 한다:

(1) 데이타 x가 w-r 충돌을 발생시킨 브로드캐스트 정보부터 최근에 받은 브로드캐스트 정보에서 수정되었다고 알려졌는가?

(2) 그렇지 않은 경우, 서버에게 데이타를 요청한다. 이후 *msgAbortable* 메시지가 함께 오는가?

위의 두 조건 중 하나라도 성립하면 철회되는데, 이 역시 위와 동일한 이유이다. 이러한 조건 검사를 수행함으로써, *O-Pre* 알고리즘에 의해 전위-재순서화하더라도 직렬화가능성을 침해하지 않는다(정리 2). *O-Pre* 알고리즘에서 이동 트랜잭션이 가져야 할 정보는 다음과 같다:

MobileData={*T_id, ReadSet, BCindex, ConflictData* }

T_id : 트랜잭션의 인식자

ReadSet : 트랜잭션이 읽은 데이타

BCindex : w-r 충돌을 발생시킨 브로드캐스트 정보의 인식자(0인 경우에는 충돌이 발생하지 않은 경우)

ConflictData : *msgAbortable* 메시지를 발생시킨 데이타

*ConflictData*는 다음 주기의 브로드캐스트 정보를 받았을 때 *msgAbortable* 메시지를 발생시킨 데이타는 배제하기 위해 저장한다. 이러한 *MobileData* 외에도 이동 컴퓨터는 전위-재순서화된 경우, 위의 (1) 조건을 검사하기 위해서 부수적인 정보를 저장하여야 한다; 즉 w-r 충돌을 발생시킨 브로드캐스트 정보(BC_i) 및 이후 완료하기 전까지의 모든 브로드캐스트 정보(BC_{i+1}, BC_{i+2}, ...)의 *Update* 정보를 저장하여야 한다. 이러한 정보를 WR_i, WR_{i+1}, ...라고 표시한다. 이때 이동 트랜잭션이 시작한 이후부터 w-r 충돌을 발생시킨 브로드캐스트 이전의 정보와는 검사하지 않은 이유는, 그 전 브로드캐스트 정보와 충돌을 발생시키더라도 전위-재순서화 개념을 침해하지 않기 때문이다.

표 2 *O-Pre*의 이동 컴퓨터 알고리즘

```
이동 트랜잭션 T_MH에 대하여
※ 연산 r(x)를 받은 경우
  if (전위-재순서화되지 않은 경우)
    // 즉 w-r 충돌이 발생하지 않은 경우
    - 서버에게 데이타 요청
    - if (msgAbortable 메시지) ConflictData ← x
      // 전위-재순서화 불가능
    - 연산 수행
  else // w-r 충돌이 발생하여 전위-재순서화된 경우
    if (x ∈ ⋃_i WR_i) abort T_MH  // 조건 (1) 검사
    else 서버에게 데이타 요청
      if (Abortable 메시지) abort T_MH  // 조건 (2) 검사
      else MobileData.ReadSet ← x
※ 완료 연산을 받은 경우
  - 완료 수행
※ 주기적인 브로드캐스트 정보 BC_i를 받은 경우
  if (이동 트랜잭션이 수행중인 경우)
    if (전위-재순서화되지 않은 경우)
      S←MobileData.ReadSet ∩ (BC_i.Update - ConflictData)
      if (S ≠ NULL)
          MobileData.BCindex ← i
          WR_i ← BC_i.Update
          ConflictData ← NULL
    else  // 전위-재순서화된 경우, 즉 w-r충돌이 발생한 경우
      WR_i ← BC_i.Update
      BCindex ← i
```

다음의 표 2는 이동 컴퓨터가 수행하는 전체적인 알고리즘이다. 표에서 알 수 있듯이 w-r 충돌 발생한 이후에는 더 이상 w-r 충돌 검사를 하지 않는다. 이는 이미 발생한 w-r 충돌에 의하여 이동 트랜잭션은 전위-재순서화되었으며, 따라서 이후의 연산은 비관적으로 수행하였기 때문이다. 표 2에서 알 수 있듯이, 완료 요청을 받게 되면 어떠한 조건 검사 없이 완료할 수 있음을 알 수 있다. 이는 전위-재순서화 결정이 올바르고(보조정리 2), 이후의 연산이 데이타의 일관성을 침해하지 않으므로 항상 직렬화가능한 수행이라고 결정되었기 때문이다(정리 2).

3.2.2 서버의 알고리즘

서버의 알고리즘을 *O-Post*와 비교할 때, 이동 트랜잭션의 데이타 x 값 요구에 대한 처리 방식에서 차이가 난다. 즉 *O-Pre*에서는 요구한 데이타 x가 다음 주기에 전송하게 될 브로드캐스트 정보 $BC_i.Update$에 포함되었는지를 매번 검사해야 한다. 만약 그렇다면, 데이타 값과 함께 *msgAbortable* 메시지를 보낸다.

*O_Pre*와 관련된 읽기 전용 트랜잭션의 경우에는 부수적인 완료 연산 요청이 없으므로, 서버는 고려할 필요가 없다.

보조정리 2 주기적인 브로드캐스트 BC_i를 받은 후 w-r 충돌이 발견된 후, 전위-재순서화한 결과는 데이타의 일관성을 침해하지 않는다.

증명. w-r 충돌을 발생시킨 서버의 트랜잭션이 하나인 경우와 둘 이상인 경우로 나누어서 생각하자. 먼저 하나인 경우, 즉 T_{si}은 서버에서 완료한 트랜잭션으로서 이동 트랜잭션 T_{MH}와 충돌 관계를 갖는다고 가정하자. 이때 충돌이 발생한 T_{si}의 쓰기 연산과 T_{MH}의 읽기 연산에 대하여 데이타의 의미를 고려한다면, 읽기 연산이 수행된 후 쓰기 연산이 수행된 경우와 동일하다(전위-재순서화). 또한 T_{MH}의 다른 읽기 연산은 T_{si}외의 다른 트랜잭션과 충돌 관계가 없으므로, 전위-재순서화의 결과는 직렬화가능성을 위배하지 않는다.

다음, 이동 트랜잭션이 두 개 이상의 트랜잭션과 충돌 관계를 갖는 경우를 생각해보자. 즉 T_{si} ($i \geq 2$)와 T_{MH}는 각각 충돌 관계를 가진다. 그러나 충돌이 발생한 모든 T_{si}의 쓰기 연산에 대하여 T_{MH}의 읽기 연산은 전위-재순서화되므로, 항상 (모든 i에 대하여) $T_{MH} \rightarrow T_{si}$ 관계가 성립한다. 이때 전위-재순서화의 결과가 직렬화가능성을 위배하려면 $T_{sk} \rightarrow T_{MH}$의 충돌이 발생하여야 한다(단 T_{sk}는 T_{si}중 하나이거나, 혹은 T_{si}와 직,간접 충돌 관계의 트랜잭션). 그러나 그러한 충돌은 T_{si}의 쓰기 연산의 결과를 T_{MH}가 읽은 경우가 되며, 이 경우 T_{MH}는 철회된다. 따라서 전위-재순서화에 의한 충돌 순서 결정은 직렬화가능하다. □

w-r 충돌이 발견되어 전위-재순서화하더라도 철회되지 않았다면, 이후의 연산은 비관적으로 수행한다. 즉 직렬화가능성을 위배할 수 있는 경우는 미리 철회한다.

정리 2 *O-Pre*에 의한 이동 트랜잭션의 수행은 항상 직렬화가능하다.

증명. 이동 컴퓨터에서 수행되는 트랜잭션 T_{MH}가 수행도중 주기적인 브로드캐스트 정보를 받은 후 충돌이 발견되지 않았다면, 그 수행결과는 직렬화가능하다. 만약 w-r 충돌이 발생하였다면, 충돌순서를 전위-재순서화한다. 즉 서버에서 완료한 트랜잭션 T_{si}의 쓰기 연산과의 충돌 순서를 $T_{MH} \rightarrow T_{si}$로 결정한다. 이후의 연산을 수행한 결과가 w-r 충돌을 발생시키게 되며, 그 충돌이 $T_{sj} \rightarrow T_{MH}$가 된다면 직렬화가능성을 침해할 수 있다. 이와 같은 경우는 $T_{MH} \rightarrow T_{si}$ 충돌 발견이후 서버에서 완료한 트랜잭션의 쓰기 결과를 읽은 경우이다. 하지만 *O-Pre* 알고리즘은 이러한 경우는 모두 철회시키므로 직렬화가능성을 침해하지 않는다. 따라서 철회되지 않고 모든 연산을 수행한 T_{MH}의 수행결과는 직렬화가능하다. □

4. 실험 환경 및 결과

본 논문에서는 제안하는 스케줄링 알고리즘의 성능을 평가하기 위해, CSIM을 기반으로 한 시뮬레이터를 만들어서 실험을 수행하였다. 제안하는 알고리즘의 성능을 평가하기 위해 *WoundCertifier* 기법[3]과 비교를 하였다. *WoundCertifier* 기법은 제안하는 알고리즘과 동일한 수행 환경을 가지고 있다. 즉 서버로부터 주기적인 브로드캐스트 정보를 이용하여 낙관적으로 지역 트랜잭션을 수행한다. 하지만 *WoundCertifier* 기법은 순수한 낙관적 기법을 사용하므로, 충돌이 발견되면 무조건 이동 트랜잭션을 철회시키며, 읽기-전용 트랜잭션과 수정을 포함한 트랜잭션간의 구별없이 동일한 검사 기법을 가진다. 이 절은 실험에 대한 환경과 그 결과를 비교한다.

4.1 실험환경

제안하는 알고리즘은 낙관적 수행을 기반으로 한다. 따라서 여러 이동 클라이언트들이 지역적으로 자신의 트랜잭션을 수행한다. 각 트랜잭션은 평균 Lamda(λ=5)를 가지는 지수적(exponential) 도착 시간 분포를 보인다. 각 트랜잭션의 연산에 필요한 데이타는 서버에게 요청한다. 또한 완료를 위한 검사도 서버에서 수행되므로, 서버에게 요청해야 한다. 이때 서버에서 클라이언트 방향의 대역폭(= 10^5 bit/sec)은 그 반대 방향에 비하여

10배라고 설정함으로써 비동기적 통신 환경을 고려하였다. 이러한 서비스 요청에 대하여 서버는 FCFS(First Come First Serve) 방식으로 처리한다.

서버는 이동 클라이언트의 데이타 요구 및 완료 요청 외에도, 완료한 트랜잭션의 수행 결과를 모든 이동 클라이언트들에게 주기적으로 브로드캐스트 한다. 브로드캐스트 정보를 이동 클라이언트가 받게 되면, 지금까지의 수행 결과에 대한 일관성 검사를 수행한다. 이러한 일관성 검사 및 이동 트랜잭션 완료 요청에 대한 일관성 검사는 제안하는 알고리즘과 *WoundCertifier* 기법에서 차이가 있다; 즉 제안하는 *O-Post*와 *O-Pre* 알고리즘은 기본적으로 *w-r* 충돌을 기반으로 정당성 검사를 수행하므로, 검사를 위해서 오직 이동 트랜잭션의 읽은 데이타 집합과 서버에서 완료한 트랜잭션이 수정한 데이타 집합만을 필요로 한다. 하지만 *WoundCertifier* 기법은 이동 클라이언트와 브로드캐스트 정보 모두 읽은 데이타와 수정한 데이타를 이용하여 검사를 수행한다.

실험에서 알고리즘의 성능 비교를 위해 트랜잭션의 평균 수행시간과 철회율을 기준으로 선택하였다. 철회율의 경우, 두 알고리즘이 모두 낙관적 수행을 기반으로 하므로 실험 인자가 변함에 따라 크게 영향을 받을 수 있다. 성능에 영향을 주는 요소는 다음과 같다: (1) 동시 수행되는 최대 이동 클라이언트의 수 (2) 트랜잭션 당 쓰기 연산의 비율 (3) 서버가 관리하는 데이타 중 hot spot의 비율 (4) hot spot 대 cold spot의 접근 비율.

다음의 표 3은 이 실험을 위해 정의한 인자로써, 비교할 알고리즘[3]과 유사하게 설정하였다. 단 수행 시간은 서버에서 읽기 연산을 수행하기 위해 소요되는 시간(10)을 기준으로 상대 시간으로 설정하였다.

4.2 실험 결과

이 절에서는 제안한 알고리즘에 대한 실험을 통해 성능을 평가하였다. 실험은 먼저 *O-Post* 알고리즘과 *WoundCertifier* 알고리즘에 대하여 평균 수행 시간 및 철회 회수를 비교하였다. 다음으로 *O-Pre* 알고리즘과 *WoundCertifier* 알고리즘에 대하여 동일한 실험을 수행하였다. 각 실험은 다시 서버가 관리하는 모든 데이타에 대하여 동일한 접근 확률을 가지는 경우와 일부 데이타에 대하여 높은 접근 확률을 가지는 데이타(*Hot Data*)가 있는 경우로 나뉘었다. 먼저 그림 2에서는 이동 클라이언트의 수가 변할 때, 평균 수행 시간과 철회 회수에 미치는 영향을 보여주고 있다. 각 클라이언트당 30개의 트랜잭션을 수행하여 평균값을 실험값으로 내었다. 실험에서 각 트랜잭션은 전체 연산 중 평균 20%의 쓰기 연산을 가지고 있다. 그림에서는 클라이언트의 수가 증가하더라도 특히 철회 회수가 큰 폭으로 증가하지 않음을 보여준다. 클라이언트의 수가 증가하는 것은 병행적으로 수행되는 트랜잭션의 수가 증가함을 의미하며, 이는 다시 트랜잭션들 간에 충돌이 발생할 확률이 높아짐을 의미한다. 따라서 충돌이 발생하면 무조건 철회시키는 *WoundCertifier* 알고리즘에 비해, 특별한 작업 수행 없이 데이타의 일관성을 유지할 수 있는 범위내에서 논리적으로 재순서화시키는 *O-Post* 알고리즘이 우수한 성능을 보여주고 있다. 트랜잭션의 수행 시간의 경우, 전체적으로 7 - 11% 만큼 단축되었음을 알 수 있다.

표 3 실험 환경 변수

DB_Size	10^3 개	서버가 관리하는 데이타의 개수
Bandwidth	10^3 bits/sec	서버에서의 브로드캐스트 채널의 대역폭
Period	10000	브로드캐스트 주기
Lamda(λ)	5	트랜잭션 도착 비율 (지수 분포)
OPNumber	8 - 12 개	이동 트랜잭션 당 수행하는 연산의 개수
Read_Time	10	서버에서 읽기 연산을 수행하는데 걸리는 시간
Write_Time	15	서버에서 쓰기 연산을 수행하는데 걸리는 시간
MSG	400	이동 클라이언트와 서버간에 메시지 전송 시간
Restart	100	철회 후 재시작되기까지의 시간
CommitRQ	100	서버가 완료 요청을 처리하는데 걸리는 시간
Validation	200	이동 트랜잭션이 주기적인 브로드캐스트를 처리하는 시간

그림 3에서는 서버가 관리하는 데이타 중 Hot Data Ratio만큼의 그렇지 않은 데이타에 비하여 높은 접근 확률을 보이는 경우에서 실험한 결과를 보여준다. 실험에서는 Hot Data와 Cold Data의 접근 확률은 4:1로 가정하였다. 만약 Hot Data Ratio가 10%라면 전체 데이타 중 10%가 Hot 데이타이며, 이 10%의 데이타 각각에 접근할 확률은 나머지 90%의 데이타 각각에 접근할 확률의 4배임을 의미한다. 이 경우, Hot 데이타가 없는 경우에 비하여 충돌이 발생할 확률이 높아진다. 왜냐하면 각 트랜잭션들이 Cold 데이타에 비하여 Hot 데이타에 접근할 확률이 높아지며 이는 충돌을 발생시키기 때문이다. 특히 Hot Data Ratio가 높다는 것은 트랜잭션이 접근하는 데이타가 Hot 데이타일 확률이 높다는 것

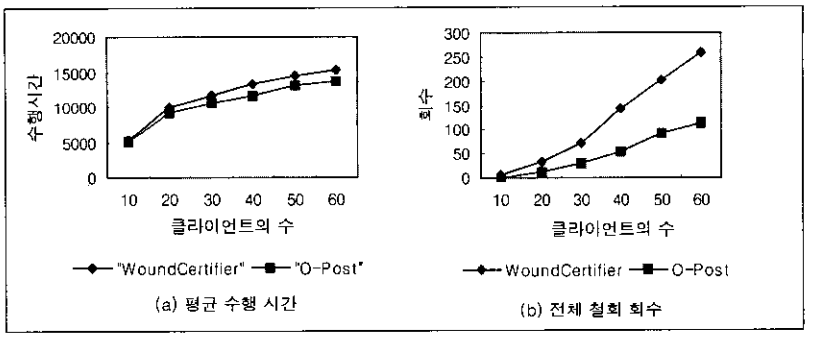


그림 2 *Hot Data*가 없는 경우

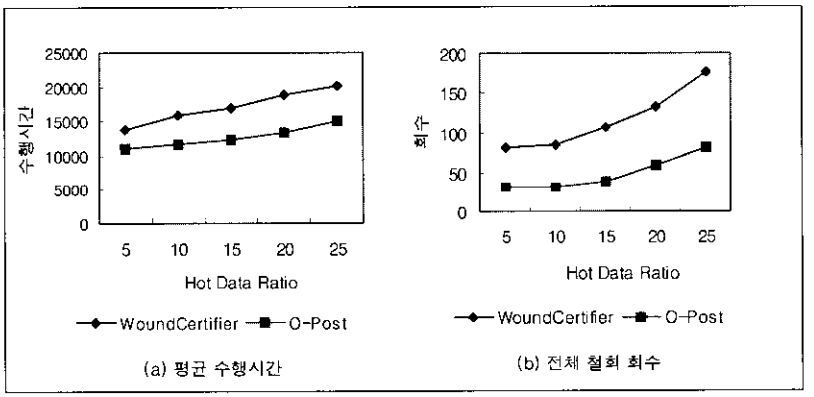


그림 3 *Hot Data*가 있는 경우

을 의미하며, 따라서 충돌 회수 역시 많아진다. 트랜잭션의 수행 시간의 경우, 19 - 25% 만큼 단축됨을 알 수 있다.

그림 3에서 알 수 있듯이 *O-Post* 알고리즘의 성능이 *WoundCertifier* 알고리즘에 비하여 상당히 향상 되었음을 알 수 있다. 하지만 제안하는 알고리즘 역시 낙관적 수행을 기반으로 하므로, 충돌이 발생하여 철회되는 회수가 Hot Data Ratio가 증가함에 따라 커짐을 보여준다.

그림 4와 5는 읽기 전용 트랜잭션에 대하여 제안한 *O-Pre* 알고리즘과 *WoundCertifier* 알고리즘의 성능을 비교한 것이다. 읽기-전용 트랜잭션간에는 어떠한 충돌도 발생하지 않는다. 따라서 실험을 위하여 전체 클라이언트 중 30%는 읽기-전용 트랜잭션만을 수행한다고 가정하였다. 가령 클라이언트의 수가 30개일 경우 그 중 9개가 읽기-전용 트랜잭션만을 수행시킨다. 나머지 70%의 클라이언트는 읽기-쓰기 연산을 모두 수행하며, 그 수행은 각각 *O-Post*와 *WoundCertifier* 알고리즘에 의해 수행되도록 하였다. 실험수치를 이전 그림 2, 3과 비교하였을 때 평균 수행 시간 및 철회 회수가 줄어들었음을 알 수 있다. 이는 읽기-전용 트랜잭션의 경우 발생할 수 있는 충돌의 종류가 적으므로, 철회 회수가 줄어들게 되며 따라서 평균 수행시간도 단축되었다고 볼 수 있다.

그림 4는 서버가 관리하는 모든 데이타에 대한 접근 확률이 같은 경우이며, 그림 5는 각각 다를 경우이다. 그림 5에서 알 수 있듯이 Hot 데이타가 많을 수록 수행 시간 및 철회율이 증가한다. 트랜잭션의 수행 시간의 경우, Hot Data가 없는 경우에는 4 - 19%, Hot data가 있는 경우에는 12 - 25% 단축됨을 알 수 있다.

5. 관련 연구

이동 컴퓨팅 환경의 특징 및 제약조건을 고려하면서 정보 서비스를 제공하기 위한 여러 연구논문들이 발표

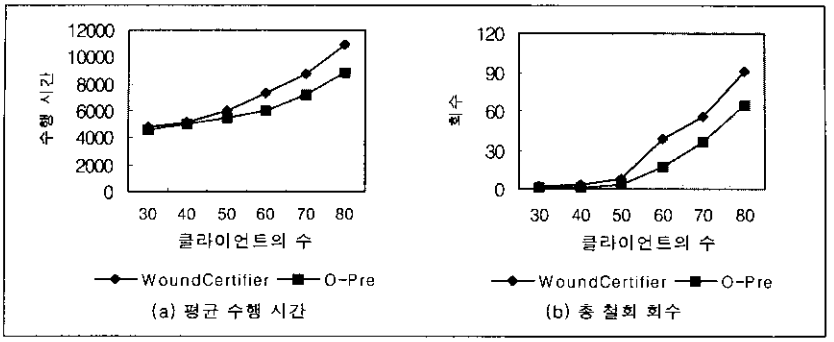


그림 4 *Hot Data*가 없는 경우

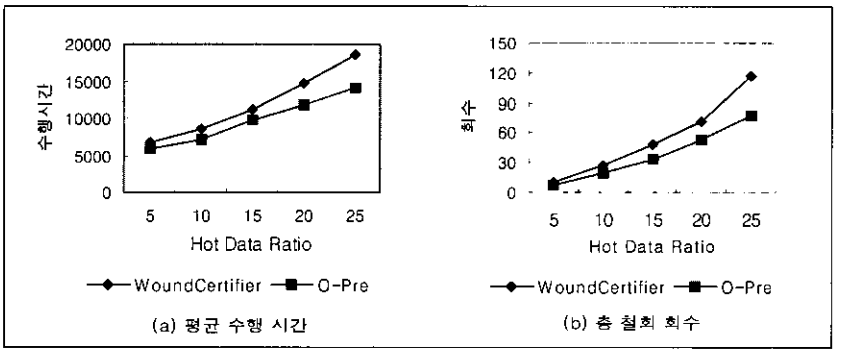


그림 5 *Hot Data*의 비율이 변하는 경우

되었으며, 특히 그 속성 중 하나인 비대칭적인 통신 환경에 적합한 알고리즘 개발과 관련된 연구가 활발히 진행되고 있다. 즉 서버로부터 이동 컴퓨터 방향으로 전송 가능한 대역폭과 그 반대 방향의 대역폭은 큰 차이가 나타나므로, 클라이언트는 서버로의 메시지의 수를 줄이는 것이 중요하다. 이를 위한 방법 중 하나로써 서버로부터 브로드캐스트 되는 정보를 이용할 수 있다.

먼저 이동 클라이언트는 자신의 지역 캐쉬를 이용하여 연산을 신속하게 수행할 수 있다 [7, 8]. 캐쉬 데이타의 일관성을 유지하기 위하여 서버는 주기적으로 필요한 정보(무효화 보고, *IR*)를 브로드캐스트 해준다. 이 정보를 받으면 클라이언트는 캐쉬 데이타의 일관성 검사를 수행한다. *IR*에 다른 부수적인 정보를 전송한다면 보다 복잡한 응용, 예를 들면 트랜잭션 처리도 지원할 수 있다.

이러한 환경에서 대개 이동 트랜잭션들은 낙관적으로 수행한다. 주기적인 브로드캐스트 정보를 받게되면, 수행중인 트랜잭션은 이전 주기동안 서버에서 수행되고 완료한 트랜잭션들과 일관성 여부를 검사하게 된다. 따라서 서버는 이전 주기동안 완료된 트랜잭션에 의해 수정된 데이타와 관련된 제어 정보를 전송해 주어야 한다. 하지만 순수한 낙관적인 수행에서 발생할 수 있는 높은 철회율로 인한 성능 저하가 매우 커질 수 있다. 다중 버전으로 이루어진 무효화 보고를 서버가 주기적으로 전송하게 하여, 그러한 문제점을 해결하려는 시도가 있었다[4]. 즉 이동 컴퓨터에서 수행되는 읽기-전용 트랜잭션이 읽은 데이타가 수정된 경우 무조건 철회시키는 대신 여러 버전과 관련된 정보를 전송하여 철회율을 감소시키도록 하려고 하였다. 즉, 이 기법처럼 필요한 데이타를 서버에게 요구하는 요구-기반(on-demand) 방식의 브로드캐스트가 아니라, 서버가 관리하는 모든 데이타를 계속적으로 전송하는 푸쉬-기반(push-based) 브로드캐스트 모델이 최근에 많이 개발되고 있다. 전송되는 브로드캐스트 정보들의 구조는 그 목적에 따라 차이가 날 수 있다. 가령 이동 컴퓨터에서 데이타에 대한 평균 대기 시간을 줄이기 위하여, [9]의 저자들은 Broadcast Disk를 제안하였다. 이와 달리 브로드캐스트 되는 데이타 정보에 대한 인덱스 정보를 추가함으로써 이동 컴퓨터의 제한된 전력 자원을 효율적으로 이용하기 위한 연

구도 제안되었다[10]. [10] 연구를 기반으로 하여 브로드캐스트 정보 전송과정에서 에러가 발생할 경우에 효율적으로 처리할 수 있는 기법도 제안되었다[11].

6. 결 론

본 논문에서는 이동 컴퓨팅 환경에서 트랜잭션을 수행하기 위한 낙관적 스케쥴링 알고리즘(*O-Post*와 *O-Pre*)을 제안하였다. 이전에 제안되었던 순수한 낙관적 알고리즘은 병행적으로 수행되는 트랜잭션의 상태 및 트랜잭션의 속성에 의해 높은 철회율을 보일 수 있었다. 본 논문에서는 충돌이 발견되면 재순서화 개념을 적용하여 철회율을 줄일 수 있었다. 이러한 결과는 실험에서 보여주고 있다. 이외에도 제안하는 알고리즘에서 얻을 수 있는 장점은 다음과 같다 :

- 낙관적으로 트랜잭션을 수행하므로, 동기화에 필요한 메시지의 수를 줄일 수 있으며, 따라서 이동 컴퓨터의 제한된 자원을 효율적으로 이용할 수 있다.
- 이동 트랜잭션들은 낙관적으로 수행을 하며 서버는 이에 대하여 특별한 정보를 유지하지 않으므로 부하가 감소한다. 또한 서버는 브로드캐스트 기법을 사용함으로써 높은 확장성까지 얻을 수 있다.
- 읽기 전용 트랜잭션의 경우, 서버에서의 완료 검사 없이 지역적으로 완료 가능하다.

그러나 본 논문 역시 낙관적 기법에서 오는 단점을 완전히 극복할 수는 없다. 즉 접근 빈도가 높은 데이타(hot spot)를 이용해야 할 경우 높은 철회율을 보일 수 있으며, 이는 다시 이동 컴퓨팅 환경의 자원을 효율적으로 이용하려는 취지와 멀어질 수 있다. 그리고 본 논문에서는 이동 트랜잭션에 필요한 모든 데이타를 서버에게 요청하도록 하고 있다. 이를 위해 타임스탬프를 비롯한 다양한 기법을 혼합시키는 것을 고려중이며, 캐쉬 데이타를 도입하는 것도 앞으로의 과제가 될 수 있다.

또한 최근에는 비동기적인 이동 컴퓨팅 환경의 통신 환경에서 효율적인 정보 교환을 위해 푸쉬-기반 브로드캐스트 모델이 많이 연구되고 있다. 이에 따라 제안한 알고리즘을 이러한 환경에 적용하기 위해 연구를 수행하고 있다.

참 고 문 헌

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

[2] M.H. Dunham and A. Helal, "Mobile Computing and Databases: Anything New?," *ACM SIGMOD Record*, vol 24(4), pp.5-9, 1995.

[3] D. Barbara, "Certification Reports: Supporting Transactions in Wireless Systems," *in Proceedings of International Conference on Distributed Computing Systems*, pp.466-473, 1997.

[4] E. Pitoura and P.K. Chrysanthis, "Exploiting Versions for Handling Updates in Broadcast Disks," *in Proceedings of the 25th Very Large Data Bases Conference*, pp.114-125, 1999

[5] Y. Lee and S. Moon, "Commit-Reordering Validation Scheme for Transaction Scheduling in Client-Server Based Teleputing Systems: COREV," *in Proceedings of International Conference on Information and Knowledge Management*, pp.59-66, 1997.

[6] L.Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System", *Communications of the ACM*, vol. 21(7), pp.558-565, 1978.

[7] K.L. Wu, P.S. Yu and M.S. Chen, "Energy-Efficient Caching for Wireless Mobile Computing," *In Proceedings of the International Conference on Data Engineering*, pp.336-343, 1996.

[8] D. Barbara and T. Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Environments," *in Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp.1-12, 1994.

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

[10] T.Imielinski, S.Viswanthan, and B.R.Badrinath, "Energy Efficient Indexing on Air," *in Proceedings of ACM SIGMOD Conference on Management of Data*, pp.25-36, 1994

[11] S-C.LO and Arbee L.P.Chen, "An Adaptive Access Method for Broadcast Data under an Error-Prone Mobile Environment," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 4, pp.609-620, 2000

김 성 석

1997년 고려대학교 이과대학 전산과학과 졸업. 1999년 고려대학교 대학원 전산과학과(이학석사). 2001년 고려대학교 대학원 컴퓨터학과 박사과정 수료. 관심분야는 분산시스템에서 캐싱 및 동시성 제어 기법, 이동 컴퓨팅 시스템에서 정보 서비스 등임.

이 상 근

1994년 고려대학교 이과대학 전산과학과 졸업. 1996년 고려대학교 대학원 전산과학과(이학석사). 1999년 고려대학교 대학원 컴퓨터학과(이학박사). 2000년 4월 ~ 2001년 3월 동경대학교 생산기술연구소 Visiting Postdoc 연구원. 1996년 ~ 현재 고려대학교 기초과학연구소 연구원. 관심분야는 이동 시스템에서의 정보 접근, 무선 이동 응용서비스, 웹 캐싱 및 응용, 분산 시스템 등임.

황 종 선

1978년 Univ. of Georgia, Statistics and Computer Science 박사. 1978년 South Carolina Lander 주립대학교 조교수. 1981년 한국표준연구소 전자계산실 실장. 1995년 한국정보과학회 회장. 1982년 ~ 현재 고려대학교 컴퓨터학과 교수. 1996년 ~ 현재 고려대학교 컴퓨터과학기술원 원장. 관심분야는 알고리즘, 분산 시스템, 결함포용시스템, 데이타베이스