무선 브로드캐스트 환경에서 트랜잭션 철회율을 최소화하기 위한 낙관적 동시성 제어 기법

서종현[O] 정성원
서강대학교 컴퓨터학과
jjong[O]@mclab.sogang.ac.kr, jungsung@sogang.ac.kr

An OCC-based Concurrency Control Method to Minimize Transaction Abort Rate in Wireless Broadcast Environments

Jong-Hyun Suh[O] Sungwon Jung
Dept. of Computer Science, Sogang University

요 약

브로드캐스트 기법은 서버에서 사용자로의 하향 대역폭을 최대한 활용하고 사용자의 수와 무관하게 데이터를 배포할 수 있기 때문에 이동 컴퓨팅 환경에서의 주요한 방식으로 쓰이고 있다. 브로드캐스트 기법에서도 기존 컴퓨팅 환경처럼 여러 트랜잭션이 같은 데이터 항목을 동시에 사용하는 경우가 발생할 수 있다. 하지만 브로드캐스트 환경에서는 모바일 클라이언트의 제한된 자원과 제한된 상향 대역폭 등의 이유로 기존의 동시성 제어 기법을 그대로 사용할 수 없다. 이러한 이유로 무선 브로드캐스트 환경을 위한 동시성 제어 기법들이 많이 연구되어 왔다. 이 논문에서는 트랜잭션들의 접근 패턴이 편향될 경우 발생하게 되는 반복적인 재실행을 문제점으로 인식하고, 동시성 제어 기법에 기반을 둔 기법을 제안한다. 동일한 데이터 항목에 대한 갱신을 수행하는 트랜잭션이 많을수록 트랜잭션이 재실행될 확률은 높아지고 성능은 저하되는데, 이는 검증을 요청하는 트랜잭션들 중 가장 먼저 서버에 도착한 트랜잭션만이 경쟁에서 살아남고 나머지는 재실행되며 경쟁을 다시하기 때문이다. 제안하는 기법에서는 브로드캐스트 사이클이 끝날 때까지 완료 여부의 결정을 유보하면서 검증을 요청한 트랜잭션들의 조합을 구성해 후보 해 리스트를 유지한다. 마지막으로 갱신되는 데이터 항목의 수와 데이터 항목의 갱신 선호도를 기준으로 최적의 해를 선택해 트랜잭션들을 완료하고 데이터베이스에 값을 반영함으로써 트랜잭션의 철회율을 최소화하고 트랜잭션의 완료율을 높일 수 있다.

1. 서 론

컴퓨팅 환경은 커다란 변화 속에 있다. 작고 편리한 PDA, 핸드폰, 스마트폰 등의 휴대용 기기들이 대중화되고, 컴퓨터 통신이 이동전화, 무선 랜, 그리고 위성서비스로 대별되는 무선 매체에까지 그 영역이 확대되고 있다. 그 결과 사용자는 네트워크의 접속을 유지하면서 원하는 장소로의 자유로운 이동이 가능하게 되었는데, 이를 이동 컴퓨팅(mobile computing)이라 한다.

이동 컴퓨팅 환경은 다음과 같은 제약을 가진다. 무선 통신망은 서버에서 사용자로의 하향(down-stream) 대역폭은 매우 크고 반대로 사용자에서 서버로의 상향(up-stream) 대역폭은 매우 작은 비대칭적인 특성을 가진다. 그리고 배터리 같은 제한된 전력 공급원을 사용하는 휴대용 기기가 갖고 있는 자원의 한계로 인해 서버로의 통신을 시도하는 것은 비용이 비싼 노력이다. 또한 모바일 환경의 특성상 많은 사용자들이 존재하게 되는데, 서버는 사용자들이 비동기적으로 보내는 요청으로 인해 과부하가 발생할 수 있다. 따라서 서버에서 사용자로의 하향 대역폭을 최대한 활용하고 사용자의 수와 무관하게 데이터를 배포할 수 있는 브로드캐스트 기법이 이동 컴퓨팅 환경에서의 주요 방식으로 쓰이고 있다. 브로드캐스트 기법에서도 여러 트랜잭션이 같은 데이터 항목을 동시에 사용하는 경우가 발생할 수 있다. 이런 경우 데이터의 일관성(data consistency)을 보장하기 위해 동시성 제어가 필요하다.

이 논문에서는 트랜잭션들의 접근 패턴이 편향될 경우 발생할 수 있는 반복적 재실행을 문제점으로 인식하고, 동시성 제어 기법에 기반을 둔 트랜잭션의 철회율을 최소화함으로써 트랜잭션의 완료율을 높이기 위한 효과적인 동시성 제어 기법을 제안한다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2장에서는 이동 컴퓨팅 환경을 위해 지금까지 제안된 동시성 제어 기법들의 내용과 특징을 간략히 요약한다. 3장에서 제안된 기법이 기반하고 있는 낙관적 동시성 제어 기법을 자세히 살펴보고, 이 후 제안된 기법의 내용을 기술한다. 4장에서는 실험을 통해 제안된 기법의 성능을 분석한다. 마지막으로 5장에서는 본 논문의 결론을 맺는다.

2. 관련연구

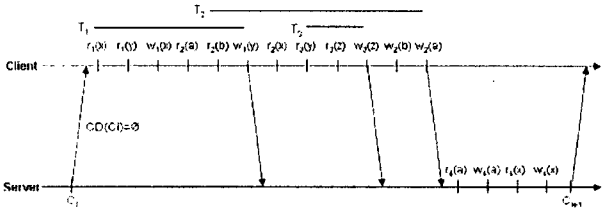
여러 트랜잭션이 같은 데이터 항목을 동시에 사용하는 경우 데이터의 일관성을 보장하기 위해 동시성 제어 문제를 해결해야 한다[2]. 널리 쓰이는 기존의 2 단계 잠금 규약(2 phase locking protocol) 같은 잠금 기반 방식은 제한된 상향 대역폭을 자주 사용해야 하기 때문에 이동 컴퓨팅 환경에 적합하지 않다[2,3,4]. 대신 검증 단계에서 데이터 충돌을 감지하여 해결하는 낙관적 동시성 제어 방법이 브로드캐스트 기법을 사용하는 이동 컴퓨팅 환경에 더 적합하다. 무선 브로드캐스트 환경을 위한 낙관적 동시성 제어 기법은 이미 많이 연구되었다. 그들 중 읽기와 쓰기가 모두 가능한 일반적인 모바일 트랜잭션을 고려한 기법으로 V. Lee는 트랜잭션의 타임스탬프를 구간으로 기록하고 이를 동적으로 조정함으로써 불필요한 트랜잭션의 중단과 재실행을 줄이는 방법을 제안했다[5]. 이 기법은 또한 트랜잭션 간의 충돌을 초기에 발견하고 중단할 수 있어 모바일 클라이언트의 불필요한 자원 낭비를 막지만 읽기전용 트랜잭션도 서버에서의 검증을 거쳐야 하고, 주고받는 제어 정보가 많다. V. Lee는 또 서버에서는 정방향 검증을 사용하고, 클라이언트에서는 역방향 검증을 사용하는 낙관적 기법을 제안하였다[1]. 이 기법에서 역방향 검증은 모든 커미트 된 트랜잭션들에 대해 이루어지고, 정방향 검증은 현재 수행중인 트랜잭션들에 대해 이루어진다. 이 기법에서 읽기 전용 트랜잭션들은 서버에 정보를 보낼 필요가 없이 모바일 클라이언트에서 자체적으로 검증하고 완료할 수 있다. 이때 트랜잭션들이 편향된 접근 패턴을 따를 경우, 즉 특정 데이터 항목들에 대한 엑세스가 상대적으로 많을 경우 계속적인 트랜잭션 재실행으로 인해 성능의 저하가 발생한다. 이는 한 브로드캐스트 사이클 내에서 동일한 항목에 대해 갱신을 수행한 트랜잭션들이 다수 존재할 경우, 최대 하나의 트랜잭션만이 완료될 수 있기 때문이다. 동일한 데이터 항목에 대한 갱신을 수행하려는 트랜잭션이 많아지면 많아질수록 모바일 트랜잭션이 재실행될 확률은 높아진다. 결국 검증을 위해 보내진 클라이언트 트랜잭션들 중에서 가장 먼저 서버에 도착한 트랜잭션만이 경쟁에서 살아남게 되고 나머지 트랜잭션은 재실행되면서 경쟁을 다시하게 된다. 이는 [1]과 [5]에게 모두 해당하는 문제로 서버에 도착하는 순서

대로 트랜잭션의 검증을 실행하여 늦게 도착하는 트랜잭션들 중 더 많은 트랜잭션을 완료시킬 수 있는 가능성을 완전히 배제한다.

3. MTAR(Minimizing Transaction Abort Rate) 기법

3.1 기본 아이디어

표 1과 같은 트랜잭션들이 그림 1의 스케줄에 따라 수행된다고 하자.



[그림 1] 트랜잭션 수행 스케줄

모바일 트랜잭션	T_1 : $r_1(x)$ $r_1(y)$ $w_1(x)$ $w_1(y)$
	T_2 : $r_2(a)$ $r_2(b)$ $r_2(x)$ $w_2(b)$ $w_2(a)$
	T_3 : $r_3(y)$ $r_3(z)$ $w_3(z)$
서버 트랜잭션	T_4 : $r_4(a)$ $w_4(a)$ $r_4(x)$ $w_4(x)$

[표 1] 트랜잭션의 구성

T_1에서 사용하는 데이터 항목 x와 y는 T_2, T_3, T_4에서도 사용된다. T_2는 데이터 항목 a, b, x를 사용하는데, 이는 T_1, T_4에서도 사용된다. T_3는 데이터 항목 y와 z를 사용하는데, 이는 T_1에서도 사용된다. 마지막으로 T_4는 데이터 항목 a와 x를 사용하는데, 이는 T_1, T_2에서도 사용된다. 모든 트랜잭션은 다른 트랜잭션과 충돌을 일으키는 데이터 항목을 하나 이상 가지고 있다.

스케줄에 따라 T_1, T_2, T_3는 클라이언트에서 부분 역방향 검증을 거치지 않고 실행을 마치고 서버로 검증을 보낸다. 기존의 V. Lee가 제안한 정방향 검증 기반의 낙관적 동시성 제어기법[1]에서는 먼저 도착하여 정방향 검증을 통과한 T_1의 완료가 결정된다. 그리고 이후 도착하는 모바일 트랜잭션 T_2, T_3와 서버 트랜잭션 T_4는 T_1이 갱신한 데이터 항목 x와 y 때문에 모두 철회된다.

T_1 : (*commit*)
T_2 : $WS(T_1) \cap RS(T_2) = \{x\}$ (*abort*)
T_3 : $WS(T_1) \cap RS(T_3) = \{y\}$ (*abort*)
T_4 : $WS(T_1) \cap RS(T_4) = \{x\}$ (*abort*)

[표 3] 트랜잭션의 검증 결과

서버는 새로운 브로드캐스트 사이클이 시작되기 전까지 트랜잭션의 완료 또는 철회를 결정하여 데이터베이스에 적용하면 된다. 따라서 검증을 요청하는 트랜잭션이 도착하자마자 즉시 완료 여부를 결정하여 데이터베이스에 적용할 필요가 없다. 제안하는 기법에서는 브로드캐스트 사이클이 끝날 때까지 완료 여부의 결정을 유보하면서 검증을 요청한 트랜잭션의 리스트를 유지한다. 검증을 요청한 트랜잭션의 리스트를 기반으로 트랜잭션들의 조합을 구성하여 후보 해의 리스트를 생성하고, 최종적으로 후보 해 중에서 최적의 해를 선택해 트랜잭션들을 완료하고 데이터베이스에 값을 반영한다. 이때 최적의 후보 해는 각 후보 해의 갱신되는 항목의 수와 데이터 항목의 갱신 선호도를 기준으로 선택된다. 갱신되는 항목의 수는 후보 해의 트랜잭션들이 갱신하는 서로 다른 데이터 항목의 수를 의미하고, 데이터 항목의 갱신 선호도 비율은 후보 해에서 갱신하고자 하는 데이터 항목들이 전체 갱신 연산에서 얼마나 비율을 차지하는지에 대한 값이다. 표 1의 트랜잭션들로부터 후보 해 리스트, 각 후보 해의 갱신 데이터 수, 갱신 선호도 비율은 다음과 같이 구할 수 있다.

데이터 항목	x	y	z	a	b	전체 갱신 연산 수
갱신 시도 횟수	2	1	1	2	1	7

[표 4] 데이터 항목에 대한 갱신 연산수

후보 해	갱신 데이터 수	갱신 선호도 비율 (갱신 시도 횟수 합/전체 갱신 연산수)
(T_1)	2 (x, y)	(2+1) / 7 [x:2, y:1]
(T_2, T_3)	3 (a, b, z)	(2+1+1) / 7 [a:2, b:1, z:1]
(T_3,T_4)	3 (a, x, z)	(2+2+1) / 7 [a:2,x:2,z:1]
…	…	…

[표 2] 후보 해 리스트

먼저 갱신해야 하는 각 데이터 항목에 대해 몇 번의 갱신이 요구되었는지 확인한다(표 3). 데이터 항목에 대한 갱신 시도 횟수는 각 트랜잭션에서 갱신하려는 모든 데이터 항목에 대한 갱신 시도 횟수 합을 구할 때 일종의 가중치(weight) 역할을 한다. 후보 해 (T_2, T_3)의 경우, 3개의 데이터 항목 a, b, z를 갱신하는데, 각 항목에 대한 갱신 시도 횟수는 표 3에서 구한 것처럼 2번, 1번, 1번이고, 갱신 시도 횟수 합은 2+1+1=4이다(표 4). 최종적으로 최적의 해를 선택할 때, 우선적으로 갱신 데이터 항목의 수가 높은 것을 찾고, 같은 수의 데이터를 갱신하는 후보 해가 존재할 경우 갱신 선호도 비율이 높은 것을 선택한다. 만약 갱신 선호도 비율까지 같다면 가장 처음 것을 선택한다. 위의 예에서는 후보 해 (T_2, T_3)와 (T_3, T_4)가 3개의 데이터 항목을 갱신하면서 가장 많은 데이터를 갱신하지만, 갱신 선호도 비율이 더 높은 후보 해 (T_3, T_4)가 최적의 해로 선택된다. 결국 트랜잭션 T_3, T_4만이 검증을 통과하여 완료되고 데이터베이스에 기록된다.

3.2 클라이언트에서의 부분 역방향 검증 알고리즘

클라이언트에서 트랜잭션이 실행되어 모든 작업이 완료되면 읽기-전용 트랜잭션의 경우 자체적으로 완료하고, 갱신 트랜잭션의 경우 전역 검증을 위해 서버로 전송한다. 이 때 트랜잭션에서 읽었던 데이터 항목의 집합(RS), 갱신한 데이터 항목의 집합(WS), 갱신된 데이터 항목의 값(New_Value) 정보가 서버로 전송된다. 그리고 트랜잭션의 수행 도중 새로운 브로드캐스트 사이클이 시작되면, 서버로부터 브로드캐스팅된 제어 정보를 바탕으로 부분 역방향 검증을 수행한다. 부분 역방향 검증의 결과에 따라 모바일 트랜잭션은 계속 진행되거나 철회된다.

```
Tpv: 부분 역방향 검증을 받는 모바일 트랜잭션
CD(Ci): 브로드캐스트 사이클 Ci동안 서버에서 갱신된 데이터의 집합
CRS(Tpv): Tpv가 현재까지 읽은 데이터 항목의 집합
Partial_Validate(Tpv) {
   if (CD(Ci) ∩ CRS(Tpv) ≠ { })    abort(Tpv);
   else { record the value of Ci;
          Tpv is allowed to continue;  }
}
```

[알고리즘 1] 클라이언트에서의 부분 역방향 검증 알고리즘

3.3 서버에서의 전역 검증 알고리즘

제안하는 방식과 낙관적 동시성 제어를 사용하는 기존의 다른 기법과의 가장 큰 차이는 서버에서의 전역 검증 과정이다. 기존 방식은 모바일 트랜잭션의 검증 요청이 도착하는 순서대로 서버에서 검증을 실행하여 완료나 철회 여부가 즉시 결정되었지만, 제안하는 방식에서는 브로드캐스트 사이클이 끝날 때까지 그 결정이 유보된다. 서버에서는 검증을 요청한 모바일 트랜잭션의 리스트를 유지하고, 그 리스트를 기반으로 데이터 충돌을 발생시키지 않는 모든 트랜잭션들의 조합으로 구성된 후보 해의 리스트를 유지한다. 이 때 각 후보 해는 직렬화 그래프의 형태로 유지된다. 이때 서버는 후보 해의 리스트를 한 브로드캐스트 사이클 동안 유지하는 작업은 서버에 과부하가 될 수 있다. 이는 트랜잭션의 조합 정보를 보다 짧은 주기로 유지함으로서 문제를 해결할 수 있다. 하지만 지나치게 짧으면 각 트랜잭션의 검증 요청마다 검증을 실행하게 되어 결국 기존의 기법과 같게 된다. 따라서 한 사이클 내 여러 번 최적 해를 선택해 데이터베이스에 기록하여 정보 유지에 따른 과부하를 최소화할 때, 얼마나 자주 허용할 것인지가 제안한 기법을 최대한 활용하기 위한 중요한 문제이다.

$$t_{cost} \times (|T_{pv}| + \alpha) > t_{remaining} \quad -(1)$$

조건 (1)이 만족될 때마다 생성된 후보 해의 리스트 중에서 최적 해를 선택해 데이터베이스에 적용한다. t_{cost}는 최근 트랜잭션의 검증 요청을 처리하는데 소요된 시간, $t_{remaining}$은 다음 브로드캐스트 사이클이 시작되기 전까지의 남은 시간, $|T_{pv}|$는 현재 큐에 대기 중인 검증을 요청하는 트랜잭션의 수를 말한다. 다음 브로드캐스트 사이클이 시작되기 전에 최적 해가 선택되어 데이터베이스에 반영되어야 하기 때문에, t_{cost}가 $t_{remaining}$보다 클 수 없다. 이 때 동안 최근 트랜잭션의 검증 요청을 처리하면서 다른 트랜잭션의 검증 요청이 대기하고 있을 수 있기 때문에 $|T_{pv}|$도 고려되어야 한다. 는 임계치(threshold)로써 접근한 데이터 항목의 수가 평균을 넘는 트랜잭션 또는 다른 어떤 이유의 가능한 부하를 고려하기 위한 상수이다. 는 서버의 성능과 검증을 요청하는 트랜잭션의 수 등을 고려하여 시스템 변수로써 조정할 수 있다.

검증을 요청하는 모바일 트랜잭션 T_v가 도착하면, 서버는 유지하고

```
Tv: 검증을 요청한 모바일 트랜잭션
Candidate_List: 후보 해의 리스트
Best_Candidate: 최종 선택된 최적 해
VTList: 현재까지 검증을 요청한 모든 모바일 트랜잭션
tremaining: 다음 브로드캐스트 사이클이 시작되기 전까지의 남은 시간
| Tpv |: 현재 큐에 대기 중인 검증을 요청하는 트랜잭션의 수
WScommitted: 현재 브로드캐스트 주기 중 갱신이 결정된 데이터의 집합
Global_Validate(Tv) {
    set tstart to current_time;
    for each candidate in Candidate_List
        if (candidate has no cycle with Tv)    Add Tv to candidate;
    new_candidate = NULL;
    Add Tv to new_candidate;
    for each Tbefore in VTList
        if (WS(Tbefore)∩RS(Tv)={ } AND (Tbefore makes no cycle))
            Add Tbefore to new_candidate;
    Add new_candidate to Candidate_List;
    tcost=current_time-tstart;
    if (tcost*( | Tpv | +    tremaining) {
        Find Best_Candidate;
        Commit transactions in Best_Candidate and abort the others;
        for each Tcommited in Best_Candidate
            WScommitted=WScommitted∪WS(Tcommitted);
        Candidate_List=NULL;
    }
}
```

[알고리즘 2] 서버에서의 전역 검증 알고리즘

있는 모든 후보 해와 비교하면서 T_v가 추가될 때 직렬화 그래프에 순환이 생성되지 않는다면 각 후보 해에 T_v를 추가한다. 만약 그래프에 순환이 생성되는 경우가 존재했다면, T_v를 갖는 새로운 후보 해를 생성한다. 새로운 후보 해에 그래프에 순환을 생성하지 않는다면 전역 검증을 위해 앞서 도착한 트랜잭션들을 하나씩 검사하여 추가한다. 이 과정에서 검증 요청을 처리하는데 소요된 시간이 결정되는데, 앞서 살펴본 조건 (1)을 검사할 때 사용한다. 만약 조건이 만족한다면 후보 해 리스트 유지하는 작업이 서버에 과부하가 되고 있음을 의미한다. 서버의 과부하를 해결하기 위해 최적의 해를 찾아 해당하는 트랜잭션을 완료한 후 데이터베이스에 값을 반영하고, 후보 해 리스트를 초기화한다. 이 과정에서 나머지 트랜잭션들은 모두 철회한다.

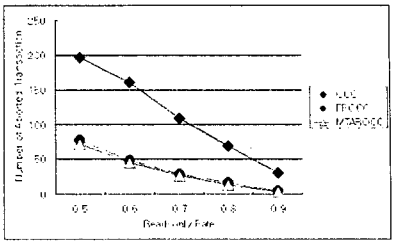
4. 성능분석

기본적인 낙관적 동시성 제어기법[6]과 V. Lee가 제안한 정방향, 역방향 검증을 이용한 낙관적 동시성 제어기법[1]과 비교를 통해서 본 논문에서 제안하는 방법의 성능을 분석한다. 편의를 위해 이 논문에서 제안하는 기법을 MTAROCC(OCC based scheme to Minimize Transaction Abort Rate), 기본적인 낙관적 동시성 제어기법을 OCC[6], V. Lee가 제안한 낙관적 동시성 제어기법을 FBOCC[1]로 쓰도록 하겠다. 그리고 성능평가에 사용된 기본적인 매개변수들은 표 5와 같다.

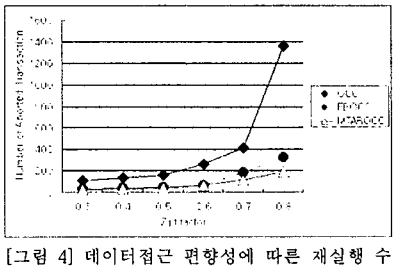
시뮬레이션 매개 변수	범위
데이터베이스 내의 데이터의 수	300
트랜잭션의 길이(연산의 수)	8
읽기 연산의 확률(갱신 트랜잭션)	0.7
읽기 전용 트랜잭션의 비율	0.5~0.9
접근 편향성(Zipf factor θ)	0.3~0.8
임계치 α	1.5

표 5 시뮬레이션 매개 변수

그림 2는 전체 트랜잭션 중 읽기-전용 트랜잭션 비율의 감소, 즉 갱신 트랜잭션 비율의 증가할 철회로 인해 재실행이 발생하는 수를 보여준다. OCC 기법의 경우 모든 읽기-전용 트랜잭션에 대해서서 서버에서의 검증을 거쳐야 하기 때문에 다른 기법 보다 훨씬 많은 재실행을 필요로 한다. 반면 FBOCC와 MTAROCC의 경우 부분 역방향 검증을 마친 읽기-전용 트랜잭션의 경우 자체적으로 서버에서의 검증 없이 완료할 수 있기 때문에 훨씬 좋은 성능을 보인다. 그림 3은 접근 패턴의 편향성을 보여주는 Zipf함수의 매개변수 θ값이 커지면서 데이터 충돌로 인한 재실행 수가 증가하는 것을 볼 수 있다. θ의 증가에 따라 재실행 수가 기하급수적으로 증가해 가는 것을 확인할 수 있는데 FBOCC와 MTAROCC의 경우 OCC에 비해 뒤늦게 나타나고, 그 정도



[그림 3] 읽기전용 트랜잭션 비율에 따른 재실행 수



[그림 4] 데이터접근 편향성에 따른 재실행 수

도 상당히 미미함을 할 수 있다. 트랜잭션의 접근 패턴이 편향될수록 제안한 MTAROCC가 FBOCC에 대해서 더 나은 성능을 보이는 것을 확인할 수 있다. 이는 MTAROCC는 검증을 요청하는 트랜잭션이 도착하는 순서대로 데이터베이스에 반영하지 않고, 철회율을 낮출 수 있는 트랜잭션의 조합을 구성해 일괄적으로 처리하는 방식을 사용했기 때문이다.

5. 결론

제안하는 OCC기반 MTAR 기법은 모바일 클라이언트에서의 부분 역방향 검증과 서버에서의 전역 검증을 사용한다. 모바일 클라이언트의 부분 역방향 검증을 통해 서버의 검증 과정의 부하를 분산시키고, 보다 일찍 데이터 충돌을 발견할 수 있기 때문에 뒤늦은 재실행을 줄일 수 있다. MTAR 기법에서 중요한 것은 서버에서의 전역 검증 과정으로 서버에서의 검증이다. 검증을 요청하는 트랜잭션의 순서대로 이루어지는 것이 아니라 여러 트랜잭션의 요청을 받아 철회율을 최소화할 수 있는 트랜잭션들의 조합을 구성하여 그 중 최적의 후보 해를 선택하여 그 후보 해의 트랜잭션을 완료시킨다. 이를 통해 먼저 도착한 트랜잭션으로 인해 이보다 늦게 도착한 여러 트랜잭션이 기회가 박탈될 수 있는 가능성이 줄임으로써 모바일 트랜잭션의 철회율을 최소화할 수 있었다.

6. 참고문헌

[1] 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.

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

[3] E. Pitoura, "Supporting Read-Only Transactions in Wireless Broadcasting," Proc. DEXA98 Int'l Workshop Mobility in Databases and Distributed Systems, pp. 428-433, 1998.

[4] Shanmugasundaram, J., Nithrakasyap, A., Padhye, J., Sivasankaran, R., Xiong, M., Ramamritham, K., 1997. Transaction processing in broadcast disk environments. In: Jajodia, S., Kerschberg, L. (Eds.), Advanced Transaction Models and Architectures. Kluwer, Boston, pp. 321-338.

[5] 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, 51, 10, pp. 1196-1211, 2002.

[6] Harder, T., Observations on optimistic concurrency control schemes. Information Systems 9 (2), 1984.