이동 환경에서 갱신 연산을 지원하는 낙관적 동시성 제어 방법

김치연* · 배석찬**

An Optimistic Concurrency Control supports Update Operations for Mobile Transactions

Chi-yeon Kim* · Suk-chan Bae**

이 논문은 2002년도 목포해양대학교 교내 연구비에 의해 연구되었음

요 약

지금까지 이동 컴퓨팅 환경에서 수행되는 대부분의 응용들은 판독 전용 트랜잭션만을 대상으로 하였다. 하지만 이동 시스템의 발전과 확산에 따라 이동 호스트에서도 데이터를 갱신할 수 있는 새로운 메커니즘이 필요하게 되었다. 이에 따라 이 논문에서는 갱신 연산을 포함하는 이동 트랜잭션의 낙관적 동시성 제어 방법을 제안하고자 한다. 이동 호스트에서 수행되는 판독 전용 트랜잭션은 서버와의 어떤 정보도 교환하지 않고 지역적으로 종료될 수 있으며, 갱신 연산을 포함한 트랜잭션은 서버에서 검증을 통하여 종료된다. 제안하는 방법은 타임스탬프와 직렬화 그래프를 이용함으로써 충돌 정보만 사용한 기존의 연구에서 발생하는 불필요한 이동 트랜잭션의 철회를 해결하였다.

ABSTRACT

So far, the main applications of mobile client are querying data. However, we need a new mechanism to update data at a mobile client as mobile systems are advanced and extended. So, we present a optimistic concurrency control to schedule mobile transactions may include update operations. The read-only transactions can be terminated without sending information to a server, and update transactions are terminated by validating at a server. In our method, we can reduce the additional aborts by using a timestamp ordering and serialization graph test mechanism rather than using only conflict information between concurrent transactions.

키워드

이동 트랜잭션, 낙관적 방법, 동시성 제어, 직렬성, 방송

Ⅰ. 서 론

이동 컴퓨팅 환경은 휴대용 컴퓨터와 무선 통신망의 발전에 힘입어 등장하였으며, 최근 빠른 속도로 발전하고 있다. 이동 컴퓨터나 PDA같이 무선망을 통하여 사용자 요구를 처리하는 장치들의 성능은 유선 망에 연결된 장치와 거의 유사할 정도로 발전하고 있고, 가까운 미래에는 데이터베이스 시스템을 구성하는 주요한 하부 구조가 될 전망이다[1].

*목포해양대학교 해양전자통신공학부 전임강사

**군산대학교 컴퓨터정보과학과 부교수

접수일자 : 2002. 9. 30

이동 컴퓨팅 환경은 분산 시스템의 확장으로 고려할 수도 있다. 하지만 이동 컴퓨팅 환경에는 사용자의 이동성, 서버와 클라이언트 사이의 비대칭적 통신 용량, 무선망의 사용, 단절 등의 독특한 특징과 제약들이 존재한다[1, 2, 3, 4, 5, 6]. 따라서 이동 컴퓨팅 환경의 시스템 관리 방법들은 이러한 제약들을 완화할 수 있어야 한다.

이동 컴퓨팅 환경에서 다수의 사용자들은 자신들의 요구를 트랜잭션의 형태로 제출한다. 따라서 이동 환경에서도 트랜잭션의 처리 방법이 필요한데, 이동 환경의 고유한 특징으로 인하여 기존의 트랜잭션 관리 방법을 그대로 적용할 수는 없다. 또한 이동 호스트를 비롯한 하드웨어의 성능이 개선되고, 다양한 응용이 출현함에 따라 판독 전용 연산뿐 아니라 갱신 연산을 포함한 이동 트랜잭션의 관리 방법이 필요하게 되었다[7].

이동 컴퓨팅 환경의 구조는 그림 1과 같다. 유선망에 연결된 호스트를 고정 호스트(FH: Fixed Host)라 하고, 노트북과 같이 이동할 수 있는 호스트를 이동 호스트(MH: Mobile Host)라 한다. 고정 호스트 중 이동 호스트와 무선으로 통신할 수 있는 인터페이스를 갖는 호스트를 서버 또는 이동 기지국(MSS: Mobile Support Station)이라 한다. 이동 호스트에서 제출되는 트랜잭션을 이동 트랜잭션(Mobile Transaction)이라 하고, 판독 연산만으로 구성된 트랜잭션을 판독 전용 트랜잭션(Read-Only Transaction)이라 한다. 하나의 이동 기지국이 제어하는 물리적 영역을 셀이라 한다.

이 논문에서는 이동 컴퓨팅 환경에서 수행되는 트랜잭션의 동시성 제어 방법에 대하여 제안한다. 이동 호스트에서 제출되는 트랜잭션은 판독 연산뿐 아니라 갱신 연산을 포함할 수 있으며, 낙관적 동시성 제어 방법에 따라 수행된다. 제안하는 방법은 타임스탬프 메커니즘을 사용함으로써 충돌 트랜잭션들의 직렬가능한 수행을 보장할 수 있고, 충돌 정보만으로 검증한 방법에서 발생하는 트랜잭션의 불필요한 철회를 해결할 수 있다.

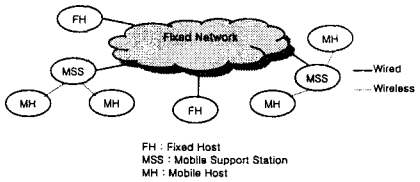

그림 1. 이동 컴퓨팅 환경
Fig. 1 Mobile Computing Environments

이 논문의 구성은 다음과 같다. 2장에서는 이동 환경에서 동시성 제어를 다룬 연구들과 발생하는 문제점들을 살펴보고, 3장에서는 제안하는 방법에 대한 시스템 모델과 알고리즘을 기술한다. 4장에서는 제안된 방법의 특징을 분석하고, 5장에서는 결론과 향후 연구방향을 기술한다.

II. 관련 연구

이동 환경에서 동시성 제어를 위해 제안된 연구로는 [7, 8, 9, 10, 11, 12, 13] 등이 있는데, [8, 10, 13]에서는 갱신 연산의 관리 방법을 다루고 있고, 다른 연구들에서는 판독 연산만 허용하거나 부분적인 갱신 연산을 허용하고 있다. 또한 [11]을 제외하고는 트랜잭션의 정확성 기준으로 모두 직렬성(Serializability)을 사용하고 있다. 동시성 제어 방법의 관점에서 보면 [10]은 낙관적 방법(Optimistic approach)을 사용하고 있고, 다른 연구에서는 잠금이나 직렬화 그래프 검사와 같은 비관적 방법(Pessimistic approach)을 사용하고 있다.

[10]에서는 갱신 연산을 포함하는 이동 트랜잭션에 대하여 낙관적 방법을 사용한 동시성 제어 프로토콜을 제안하였다. 충돌 정보에 기반을 둔 2단계 잠금 검증자(2-Phase Locking Certifier)를 사용하여 수행되는 트랜잭션에 대한 판독 집합과 기록 집합을 유지함으로써 동시성을 제어하였다. 서버는 중앙에 하나만 존재하고 방송을 이용하여 클라이언트가 지역 검증자로 작동하게 하였다. 이 연구의 문제점은 단순한 접근 집합만의 검사로 트랜잭션의 철회와 완료를 결정함으로써 직렬성이 유지되는 수행도 철회될 수 있다는 점이다.

[11]에서는 이동 환경에서 수행되는 트랜잭션의 정확성을 검증하기 위해 직렬성보다 완화된 기준인 갱신 일관성(Update Consistency)을 제안하였다. 분산 환경에서 지금까지 트랜잭션 수행의 정확성을 검증하기 위해 가장 일반적으로 사용되어온 기준은 직렬성이다[14]. 하지만 분산 환경에 비해 여러 가지 자원이 빈약한 이동 환경에서 이 기준을 그대로 사용하기에는 성능 저하와 같은 문제점이 뒤따른다. 이러한 문제점들을 해결하기 위하여 전통적인 정확성을 상호 일관성(Mutual Consistency)과 최신성(Currency)의 관점으로 접근하였다. 하지만 이 방법은 많은 계산을 필요로 하고, 큰 볼륨의 제어 정보를 방송하기 위해 대역폭을 많이 소모한다는 문제점이 있다.

위의 연구들을 종합하면, 자원을 최소한 사용하면서 이동 호스트가 일관된 값을 접근할 수 있는 방법과 이동 트랜잭션의 효율적인 수행 방법, 낮은 대역폭을 극복하기 위한 메시지 교환의 최소화 등이 이동 컴퓨팅 환경에서 트랜잭션 처리 방법의 목표가 됨을 알 수 있다.

III. 제안하는 방법

이 장에서는 제안하는 방법인 OCUT-ME(an Optimistic Concurrency control for Update Transactions in Mobile Environments) 방법에 대하여 기술한다.

1. 시스템 모델 및 가정

제안하는 방법이 적용되는 이동 컴퓨팅 환경은 (그림 1)과 같고, OCUT-ME 알고리즘에서 사용된 가정은 다음과 같다.

- 이동 트랜잭션은 갱신 연산을 포함할 수 있으나 출현 비율은 높지 않다.
- 이동 호스트와 서버는 동기화된 클럭을 갖는다.
- 이동 호스트에서는 한 번에 하나씩의 트랜잭션만 수행된다.

2. 이동 호스트 알고리즘

이 절에서는 이동 호스트로 제출되는 트랜잭션의 스케줄링을 위한 캐쉬 관리 방법과 낙관적 방법에 근거한 트랜잭션의 동시성 제어 방법에 대하여 기술한다.

이동 호스트는 무선망의 낮은 대역폭을 극복하고, 트랜잭션의 응답 시간을 단축시키기 위해 캐쉬를 사용한다[15]. 이동 호스트의 캐쉬에는 자주 접근하는 데이터를 저장하고, 서버에서 주기적으로 방송하는 무효화 레포트(IR: Invalidation Report)를 통하여 내용을 갱신한다. 무효화 레포트는 서버에서 갱신된 데이터를 이동 호스트에게 알려주기 위해 서버가 주기적으로 방송하는 메시지이다. 캐쉬에는 각 데이터에 대하여 하나의 버전만을 유지하고, 필요한 데이터가 캐쉬에 존재하지 않을 경우에는 트랜잭션의 타임스탬프에 따라 접근가능한 버전을 서버에 요구한다.

이동 호스트에서 트랜잭션이 생성되면 가장 먼저 타임스탬프를 할당받는다. 트랜잭션의 타임스탬프는 트랜잭션이 시스템에 도착한 시간값이다. 3.1절에서 가정한 동기화된 클럭과 호스트 식별자를 이용하여 각 트랜잭션에 유일한 타임스탬프를 부여한 후 규칙에 따라 스케줄한다.

■ 판독 전용 트랜잭션

이동 호스트로 제출되는 전형적인 트랜잭션으로 판독 연산만으로 구성되는 경우이며, 판독 전용 트랜잭션은 값을 변경시키지 않으므로 가능한 한 지연없이 수행하는 것이 성능을 향상시킬 수 있는 방법이다. OCUT-ME 알고리즘에서 판독 전용 트랜잭션은 캐쉬에 저장된 값을 접근한다. 마지막 연산을 수행한 후에는 서버에서 보내는 무효화 레포트를 청취하여 완료 여부를 결정한다. 즉 타임스탬프 순서대로 수행되지 않은 충돌 연산이 탐지될 경우에 판독 전용 트랜잭션은 철회된다. 트랜잭션의 완료 여부에 상관없이 이동 호스트는 서버로부터 받은 무효화 레포트에 따라 캐쉬 내용을 갱신한다. 만약 판독 전용 트랜잭션이 여러 방송 구간 동안 실행된다면, 각 방송 시점에 방송되는 무효화 레포트를 저장해야 올바른 수행을 보장할 수 있다. 완료를 기다리는 판독 전용 트랜잭션은 다음의 조건 중 하나를 만족하면 성공적으로 완료될 수 있다.

① 무효화 레포트에 포함된 데이터 집합과 판독 집합에 교집합이 없는 경우

② 무효화 레포트와 판독 집합에 교집합이 있다

하더라도 갱신 타임스탬프가 자신의 타임스탬프보다 작은 경우

■ 이동 갱신 트랜잭션

이동 호스트에서 제출되는 이동 트랜잭션은 갱신 연산을 포함할 수 있다. 서버에서 수행되는 갱신 트랜잭션과 구별하기 위해 이동 호스트에서 제출되는 갱신 트랜잭션을 이동 갱신 트랜잭션(MUT: Mobile Update Transaction)이라 하고, 서버에서 제출되는 갱신 트랜잭션을 서버 갱신 트랜잭션(SUT: Server Update Transaction)이라 정의한다. 이동 호스트에서 수행되는 갱신 연산은 일단 지역 복사본을 만들어 수행된다. 또한 다중 버전 환경을 가정했기 때문에 모든 갱신 연산은 새로운 버전을 생성한다. 생성되는 다중 버전들 사이의 직렬성을 보장하기 위해 1-사본 직렬성을 사용한다[14]. 갱신 트랜잭션의 완료 여부는 지역적으로 결정할 수 없으므로 마지막 연산 수행 후 서버에 보내어 검증한다. 이동 호스트는 갱신 트랜잭션의 마지막 연산의 수행이 끝나면 트랜잭션의 식별자와 타임스탬프, 그리고 갱신한 각 데이터 항목에 대하여 (갱신한 데이터 항목, 새롭게 씌여진 값, 연산 수행 시각)을 서버에 보낸다. 서버에서는 이 정보를 이용하여 직렬화 그래프 검사를 하게 되고, 그래프에 사이클이 발생하지 않으면 성공적으로 완료할 수 있다. 서버는 검증을 통과한 갱신 트랜잭션에 대하여, commit_set에 첨가하고 새로 씌여진 값을 데이터베이스에 인스톨한 후 트랜잭션의 식별자를 방송하여 이동 호스트에서도 완료될 수 있게 한다. 검증을 통과하지 못한 트랜잭션들은 abort_set에 첨가하고 방송을 통하여 클라이언트들에게 전달된다. 서버에 유지되는 그래프 관리 방법은 3.3절에서 기술한다. 이동 호스트에서 수행되는 알고리즘은 그림 2와 같다.

```
/* Mobile Host Algorithm */
Assign a unique timestamp to a MT T;
for (each operation op(x) of T) {
  if (op == 'r')
    if (x exists in cache) reads x;
    else requests x to a server;
  else if (op == 'w') creates a local copy of x;
}
if (executes a last operation of T) {
 if (T is a ROT) {
  waits the next IR;
  intersection = (writeset(IR) ∩ readset(T))
  if (intersection is empty) commits T;
  else {
    for (each transaction T' that updates
common data with T)
      if (ts(T) > ts(T')) commits T;
      else aborts T;
  }
 }
 if (T is a MUT) {
   sends (TID, ts, (x, new_value, exe_time)) a
server;
   waits a global decision;
  }
}
if (a mobile client receives IR) {
  if (there is any transaction T' violates the
serializability) aborts T';
  updates the cache according to the IR;
  switch (global decision) {
    case "commit" : stores the new value;
break;
    case "abort" : deletes the local copy;
break;
  }
}
```

그림 2. 이동 호스트 알고리즘

Fig. 2. Algorithm for a mobile host

3. 서버 알고리즘

이 절에서는 서버에서 수행되는 알고리즘에 대하여 기술한다.

서버에는 다중 버전을 유지한다. 서버에 다중 버전을 유지하는 이유는 장기간 수행되는 트랜잭션(Long-duration Transaction)의 완료율을 향상시키기 위해서이다. 이동 호스트는 전력의 한계나 이동성으로 인하여 트랜잭션이 종료할 때까지 걸리는 시간이 길어질 수 있다. 이 경우에, 최신 버전 하나만을

유지하게 되면 트랜잭션이 접근해야 할 데이터가 존재하지 않아 트랜잭션은 철회되게 되고, 최악의 경우에는 기아상태가 발생할 수도 있다. 또한 다중 버전은 고장의 발생시 회복 단계에서도 사용될 수 있고, 동시성을 증가시킬 수 있다[16]. 새로운 버전은 갱신 연산이 수행될 때마다 생성되고, 완료되면 다른 서버에 전파된다. 또한 더 이상 그 버전을 접근할 트랜잭션이 존재하지 않는 경우 삭제될 수 있다. 시스템에서는 트랜잭션이 완료될 때마다 불필요한 버전이 있는지 검사하여 삭제한다.

이동 환경은 소수의 서버와 대다수의 이동 호스트로 구성된다. 하나의 서버에 의해 제어되는 셀 내에는 다수의 이동 호스트가 존재할 수 있다. 이런 환경에서 효율적으로 데이터를 전파할 수 있는 방법이 방송이다[17, 18]. 서버가 이동 호스트로 방송해야 할 중요한 정보 중 하나는 최근 방송 구간에서 완료된 데이터의 값이다. 서버와 이동 호스트의 데이터를 일치시키기 위하여 서버는 주기적인 시간 간격으로 갱신된 데이터에 대한 정보인 무효화 레포트를 방송한다. 무효화 레포트에 부가하여 서버는 각 갱신 연산이 수행된 시각을 이동 호스트에 방송한다.

서버로 제출되는 트랜잭션의 동시성 제어를 위해 직렬화 그래프를 이용한 낙관적 방법을 사용한다. 트랜잭션들은 일단 수행되고, 검증 단계에 들어가면 갱신한 각 데이터 항목에 유지되는 그래프에 타임스탬프 순서로만 에지를 첨가한다. 이 때 전체적인 그래프에 사이클이 생기지 않아야 완료될 수 있다. 하지만 이 때의 문제점은 비록 검증이 타임스탬프 순서대로 요청된다고 해도, 실제 연산의 수행 순서는 타임스탬프 순서와 다를 수 있다는 점이다. 따라서 그래프 검사시 타임스탬프 순서대로 수행되었는지에 대한 검사 뿐 아니라 실제 수행 시각에 대한 검사가 필요하다. 서버에서 수행되는 서버 알고리즘은 그림 3과 같다.

```
/* Server Algorithm */
Assign a unique timestamp to a server transaction T;
for (each operation op(x) of T) {
  if (op=='w') creates a local copy of x;
  executes op(x)
}
if (execute the last operation of T || there is a transaction T requests a validation)
in validation phase {
  serialization graph test;
  if (T is not included a cycle) commits T;
  else aborts T;
}
for (every T terminates the validation phase) {
   if (global decision == "commit") {
      installs the new value;
      if (T is a MUT) adds T to the commit_set;
   }
   else { // decision == "abort"
      deletes the local copy;
      if (T is a MUT) adds T to the abort_set;
    }
}
On broadcasting time {
  constructs the next IR;
  broadcasts IR, commit_set, and abort_set to all hosts;
}
if (T is committed)
  if (exists a version that won't access any more)
     deletes the version;
```

그림 3. 서버 알고리즘
Fig. 3 Algorithm for a server

다음은 제안하는 알고리즘인 OCUT-ME 알고리즘의 정확성을 증명하기 위해 전통적인 기준인 직렬성을 만족하는지에 관한 증명이다.

(정리 1) OCUT-ME 알고리즘은 직렬가능한 히스토리만을 생성한다.

(증명) 증명을 위해 두 가지 경우를 고려한다. 1) 판독 전용 이동 트랜잭션으로 인해 직렬성이 위배되는 경우와 2) 갱신 트랜잭션으로 인해 직렬성이 위배되는 경우이다. 첫번째 경우에 대한 증명을 위해

임의의 이동 트랜잭션 MTi와 트랜잭션 Tj, Tk, Tl에 대하여 직렬화 그래프에 MTi → Tj → Tk → .. Tl → MTi와 같은 사이클이 존재하는 경우를 가정한다. 타임스탬프 순서화 규칙에 의하여 MTi → Tj의 에지는 ts(MTi) < ts(Tj)임을 의미하고, Tl → MTi는 ts(Tl) < ts(MTi)임을 의미한다. 직렬화 그래프에 MTi로 인한 사이클이 존재한다면 ts(MTi) < ts(Tj) < ...< ts(Tl) < ts(MTi)이 되고, ts(MTi)<ts(MTi)의 의미가 되어 각 트랜잭션에 유일한 타임스탬프가 부여되어야 한다는 타임스탬프 규칙에 위배된다. 따라서 이와 같은 사이클은 발생하지 않는다.

두번째 경우의 예는 임의의 갱신 트랜잭션(MUT, SUT포함) UTi에 대하여, UTi → Tj → ... → Tk → UTi와 같은 사이클이 직렬화 그래프에 존재하는 경우이다. 모든 갱신 트랜잭션들은 종료하기 전 서버에서 직렬화 그래프 검사를 거치는데, 에지의 첨가로 사이클이 발생하는 경우는 해당 트랜잭션을 철회한다. 따라서, Tk → UTi의 에지로 인하여 사이클이 발생했다면 UTi는 철회되므로 사이클이 발생하지 않는다. 위의 두 경우로 OCUT-ME 알고리즘에서는 직렬성이 유지됨을 알 수 있다. □

Ⅳ. 분 석

이 장에서는 제안하는 방법의 성능을 분석적 방법을 사용하여 낙관적 방법을 적용한 [10]과 비교하여 설명하고자 한다. 비교 항목은 대역폭 소모량과 트랜잭션의 철회율이다. 비교를 위해 사용된 매개변수는 표 1과 같다.

단위 갱신 트랜잭션의 완료를 위해 이동 호스트에서 서버로 전송되는 대역폭 소모량은 [10]의 방법이 (식 1), OCUT-ME 알고리즘이 (식 2)와 같은 소모량을 갖는다.

표 1. 매개변수 목록

Table. 1. Parameter Description

매개변수	의 미
N_{op}	단위 트랜잭션의 연산 수
τ	판독 연산 비율
ω	갱신 연산 비율
δ	단위 데이터의 크기
υ	갱신된 값의 크기
B_{ts}	타임스탬프를 위한 용량
B_{id}	트랜잭션 식별자를 위한 용량
σ	IR과 접근 집합이 공통일 확률
ε	IR과 접근 집합이 존재하더라도 직렬성이 유지될 확률

$$BWCR = Nop \times \tau \times \delta + Nop \times \omega \times (\upsilon + \delta) + Bid \quad (식\ 1)$$

$$BWOCUT\text{-}ME = Nop \times \tau \times \delta + Nop \times \omega \times (\upsilon + \delta + Bts) + Bid \quad (식\ 2)$$

두 방법의 대역폭 소모량은 OCUT-ME 알고리즘이 각 갱신 연산에 대한 수행 시각을 전송하는 양만큼이 크나, 갱신 확률이 낮다고 가정했으므로 많은 차이를 보이지 않을 걸로 사료된다.

[10] 방법의 트랜잭션 철회율은 (식 3), OCUT-ME 알고리즘의 철회율은 (식 4)와 같다.

$$ACR = 1\text{-}prob(\sigma) \quad (식\ 3)$$

$$AOCUT\text{-}ME = 1\text{-}prob(\sigma)\text{-}prob(\varepsilon) \quad (식\ 4)$$

트랜잭션의 철회율 측면에서는 [10]의 경우 무효화 레포트와 트랜잭션의 접근 집합이 공통인 경우는 반드시 트랜잭션이 철회되나, OCUT-ME 알고리즘의 경우에는 공통 집합이 존재한다해도 직렬성이 유지되는 경우는 완료할 수 있으므로 [10]보다 더 많은 트랜잭션이 완료할 수 있다.

Ⅴ. 결 론

이동 컴퓨팅 환경은 사용자의 요구가 복잡해짐에 따라 발전하고 있고, 가까운 장래에 많은 응용들이 수행될 하부구조가 될 전망이다. 이에 따라 기존의 이동 환경에서 다루어왔던 판독 전용 연산으로 구성된 이동 트랜잭션뿐 아니라 갱신 연산을 포함하는

이동 트랜잭션의 수행 방법에 대한 연구가 필요하게 되었다.

이 논문에서 제안한 방법의 응용 분야 중 하나는 판매원이 노트북을 들고 다니며 관광 상품을 파는 경우이다. 각 판매원들은 자신이 담당한 분야를 갖고 있고, 경우에 따라서는 두 세 명이 같은 분야의 관광 상품을 판매할 수 있다. 각 사원들은 고객들을 만나 상품을 판매하고 노트북에서 바로 상품의 재고를 갱신한다. 데이터의 갱신은 수행가능하지만, 충돌은 그리 많지 않다. 또한 유선망에 연결된 사무실에도 고객들이 찾아와 관광 상품을 구매할 수 있다. 고정 호스트와 이동 호스트가 동일한 데이터를 접근할 수 있다. 이동 호스트에서 갱신된 데이터가 대역폭의 제약으로 매번 갱신된 정보를 전송하기 어렵다면 주기적인 시간 간격마다 갱신된 데이터를 전송함으로써 대역폭을 효율적으로 이용할 수 있다.

이 논문에서는 충돌이 드문 이동 환경에서 갱신 연산을 수행할 수 있는 동시성 제어 방법을 제안하였다. 판독 전용 트랜잭션은 아무런 지연 없이 지역적으로 수행되며 단순히 무효화 레포트와의 교집합이 발생한다고 해서 철회하지 않기 때문에 트랜잭션의 완료율을 향상시킬 수 있다. 갱신 트랜잭션은 수행 후 전역적 검증을 위해 서버에 보내진다. 서버에서는 직렬화 그래프를 유지하여 수행되는 트랜잭션의 직렬성을 검사하고, 다중 버전을 유지함으로써 장기간 수행되는 트랜잭션이나 회복에 대비하였다.

논문에서 제시한 내용에 추가하여 앞으로 모의실험을 통한 알고리즘의 정확한 성능 분석과 완화된 정확성 기준인 갱신 일관성을 적용한 동시성 제어 방법에 대해서도 연구할 예정이다.

참고문헌

[1] G. H. Forman, J. Zahorjan, "The Challenges of Mobile Computing", IEEE Computer, pp. 38-47, April 1994.

[2] M. H. Dunham, V. Kumar, "Impact of Mobility on Transaction Management," Proceedings of the International Workshop on Data Engineering for Wireless and Mobile Access, MobiDE '99, Seattle, WA, USA, pp. 14-22, Auguest 1999.

[3] J. Jing, A. Helal, A. Elmagarmid, "Client-Server Computing in Mobile Environments," ACM Computing Surveys, vol. 31, no. 2, pp. 117-157, June 1999.

[4] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing, PODC '96, Philadelphia, PA, USA, pp.1-7, MAY 1996.

[5] D. Barbara, "Mobile Computing and Databases-A Survey," IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 1, pp. 117-157, Jan./Feb. 1999.

[6] M. H. Dunham, A. Helal, S. Balakrishnan, "A Mobile Transaction Model that Captures Both the Data and Movement Behavior," ACM-Baltzer Journal on Mobile Networks and Applications(MONET), vol. 2, no. 2, pp. 149-161, October 1997.

[7] Lee V., Son S. H., Lam K.: On the Performance of Transaction Processing in Broadcast Environments, Int Conf on Mobile Data Access (MDA'99), Hong Kong, Dec 1999.

[8] Q. Lu, M. Satyanarayanan, "Improving Data Consistency in Mobile Computing Using Isolation-Only Transaction," Proceedings of the Fifth IEEE HotOS Topics Workshop, Orcs Island, May 1995.

[9] A. Elmagarmid, J. Jing, O. Bukhres, "An Efficient and Reliable Reservation Algorithm for Mobile Transactions," Proceedings of the CIKM 95, Baltimore, MD, USA, pp. 90-95, 1995.

[10] D. Barbara, "Certification Reports: Supporting Transactions in Wireless Systems," Proceedings of the 17th International Conference Distributed Computing Systems, Vienna, 1992.

[11] J. Shanmugasundaram, A. Nithrakashyap, R. Sivasankaran, K. Ramamritham, "Efficient Concurrency Control for Broadcast

Environments", ACM SIGMOD International Conference on Managament of Data, pp. 85-96, 1999.

[12] E. Pitoura, P. K. Chrysanthis, "Scalable Processing of Read-Only Transactions in Broadcast Push" Technical Report 98-026, Depart. of Computer Science, University of Ioannina, 1998.

[13] E. Pictoura, B. Bhargava, "Data Consistency in Intermittently Connected Distributed Systems", Transactions on Knowledge and Data Engineering, vol. 11, no. 6, pp. 896-915, Nov. 1999.

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

[15] D. Barbara, T. Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Computing," Proceedings of SIGMOD, pp. 1-12, May, 1994.

[16] E. Pictoura, P. K. Chrysanthis, "Exploiting Versions for Handling Updates in Broadcast Disk", Proceedings of the 15th VLDB Conference, Edinburgh, Scotlandm, pp. 114-125, 1999.

[17] S. Acharya, M. Franklin, S. Zdonik, "Disseminating Updates on Broadcast Disks", In Proceedings of the 22nd VLDB Conference Mumbai(Bombay), pp. 354-365, India, 1996.

[18] S. Acharya, M. Franklin, S. Zdonik, "Balancing Push and Pull for Data Broadcast," Proceedings of ACM SIGMOD International Conference Management of Data, Phoenix, Arizona, pp. 183-194, May 1997.

저자소개

김치연(Chi-yeon Kim)

1992년 2월 전남대학교 전산통계학과(이학사)

1994년 2월 전남대학교 대학원 전산통계학과(이학석사)

1999년 8월 전남대학교 대학원 전산통계학과(이학박사)

1996년 3월～1997년 2월 전남대학교 전산학과 조교

2002년 3월～현재 목포해양대학교 해양전자통신공학부 전임강사

※관심분야: 이동 컴퓨팅, 트랜잭션 관리, 전자상거래

배석찬(Seok-chan Bae)

1983년 전남대학교 계산통계학과(이학사)

1988년 전남대학교 대학원 전산통계학과(이학석사)

1995년 전남대학교 대학원 전산통계학과(박사)

1983년～1985년 ROTC

1993년～1994년 서남대학교 전산통계학과 학과장

1995년～현재 군산대학교 컴퓨터정보과학과 부교수

※관심분야: 트랜잭션 관리, 데이터베이스 보안, 객체지향 시스템