이동 컴퓨팅 환경에서 트랜잭션의 단계분할에 의한 동시성 향상 기법

조영일[0] 이익훈 이상구
서울대학교 컴퓨터공학부
{yicho, ihlee, sglee}@dblab.snu.ac.kr

Concurrency Control Using Step-Decomposition of Transactions in Mobile Computing Environment

Young-Il Cho[0] Ig-Hoon Lee Sang-Goo Lee
School of Computer Science and Engineering, Seoul National University

요 약

이동 컴퓨팅 환경은 분산 컴퓨팅 환경과는 달리 네트웍의 낮은 신뢰성과 제한된 대역폭을 가지고, 이동 호스트 또한 제한된 저장장치와 배터리만을 사용할 수 있으며 트랜잭션(transaction)은 장시간에 걸쳐 수행되는 특성을 가진다. 이동 컴퓨팅 환경에서는 전통적인 트랜잭션의 동시성 제어 기법 대신, 트랜잭션의 시맨틱스(semantics)를 이용하여 동시성을 향상시킬 수 있다. 본 연구에서는 중첩된 트랜잭션의 구조를 단순화시킨 단계분할(step-decomposition) 기법을 사용하여, 서브트랜잭션(sub-transaction)의 시맨틱 타입(semantic type) 별로 인터리빙(interleaving)을 제어함으로써 트랜잭션의 동시성을 향상시키는 기법을 제안하고자 한다.

1. 서론

오늘날 이동 통신 기술의 급속한 발전과 무선 컴퓨팅 장치의 대중적인 보급에 힘입어, 이동 컴퓨팅 환경에서의 데이터 서비스의 수요가 점차 증가하는 추세이다. PCS나 PDA를 이용한 인터넷으로의 연결은 더 많은 사용자들이 더 많은 데이터에 접근할 수 있도록 허용할 것이고, 이는 유선 네트웍에 고정되어 있는 MSS(Mobile Support Station)이 더 많은 트랜잭션을 동시에 수행해야 한다는 점을 의미하게 된다.[7]

이동 컴퓨팅 환경이 분산 컴퓨팅 환경과는 다르게 발생시킬 수 있는 문제점으로, 첫째, 무선 네트웍을 사용하므로 대역폭이 제한되고, 둘째, 이동 호스트가 이동하기 때문에 접속단절이 빈번하게 발생하며, 전력사용에 제한이 발생할 수 있다. 이러한 문제점들로 인하여 예측하지 못하는 상황에서 작업의 지연이 발생하다보면 수행되는 총 작업 시간이 전체적으로 증가하게 될 것이다.

무선 네트웍에 연결된 PDA나 PCS의 응용프로그램들이 더욱 복잡해지는 추세를 감안하면 이동 컴퓨팅 환경에서도 트랜잭션의 지원은 점차 필수적인 것이 되고 있다. 이동 호스트를 통한 미래의 정보 시스템에 대한 접근은 이동 트랜잭션(mobile transaction)의 도움을 받아야 할 것이다. 그러나 위에서 열거한 문제점들로 인하여 이동 트랜잭션은 기존의 트랜잭션에 비해 보다 효율적이고 정교한 방식으로 제어되어야 한다.

이 논문은 2000년도 두뇌한국21사업에 의하여 지원되었음.

이동 트랜잭션은 기존의 트랜잭션과는 달리 트랜잭션을 구성하는 연산(instruction)들이 일부는 이동 호스트에서, 일부는 MSS에서 처리되어야 하며 데이터베이스의 부분적인 결과(트랜잭션의 중간 상태에서 나타나는 결과)를 공유한다는 차이점을 가진다.[3] 이것은 트랜잭션의 고립 수준을 낮추게 된다는 점을 의미하는데, 분산 또는 이동 트랜잭션들에 대한 연구들은 고립 수준을 낮추어 동시성(concurrency)을 향상시키는 방법에 초점을 두고 있다.

이에 본 논문에서는 단계분할 기법을 사용하여 트랜잭션을 여러 단계로 분할하고, 분할된 서브트랜잭션을 시맨틱 타입 별로 분류하여 그 유형에 따라 동시적으로 수행(인터리빙) 가능 여부를 판단함으로써 동시성을 효율적으로 제어하는 기법을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서 이동 트랜잭션의 분할에 대한 관련 연구를 살펴보고, 3장에서 단계분할과 시맨틱 타입의 분류와 동시성 제어 알고리즘을 설명한다. 마지막으로 5장에서는 결론과 향후과제에 대한 언급으로 끝맺음을 한다.

2. 관련 연구와 그 문제점

2.1 전통적인 트랜잭션의 동시성 제어

전통적인 트랜잭션의 동시성 제어는 트랜잭션의 ACID 특성을 보장하기 위해 일반적으로 트랜잭션들 사이의 직렬가

능성(serializability)를 이용하여 왔다. 직렬가능성이란 곧 2PL 기법을 사용하여 트랜잭션에게 데이터 항목(data item)에 대한 록(lock)을 허용하는 것에 의해 발생하는 트랜잭션들의 스케쥴링을 뜻하게 된다. 그러나 직렬가능성은 너무 강력한 정확성 평가조건(correctness criteria)이기 때문에, 분산 또는 이동 컴퓨팅 환경에서 이것을 그대로 사용하게 되면 하나의 트랜잭션이 접근하려는 모든 데이터 항목에 대해 모든 록을 얻으려 하게 되고, 이로 인해서 다른 트랜잭션들이 충돌하는 연산을 수행하기 위해서는 먼저 시작한 트랜잭션이 완료(commit)할 때까지 기다려야만 하게 된다. 이러한 이유로 성능이 과도하게 저하되고 그로 인해서 트랜잭션의 취소(abort)가 발생할 가능성이 높아져서 데이터베이스 시스템의 전체적인 성능의 저하를 초래하게 된다.

2.2 중첩된 트랜잭션 모델

전통적인 트랜잭션 모델의 문제점을 해결하기 위해 기존의 분산 또는 이동 트랜잭션 모델은 중첩된 트랜잭션의 형태로 제안되어왔다.[3] 그러나 이 모델은 응용프로그램의 시맨틱스(semantics, 기능적 의미)가 다층적이고 서브트랜잭션이 동일한 의미를 가지는 경우에 주로 적합하며, 서브트랜잭션의 수행 결과를 원래 상태로 되돌려놓는(undo) 보정 트랜잭션(compensating transaction)을 지원해야 하는 어려움이 있으며, 프로그래머가 트랜잭션을 설계하고 계층적으로 구조화하여 실행 순서를 명세화 하거나 술어(predicate)을 조건화 하는 부담을 지게 된다. 또한 다음에 나오는 단계분할된 트랜잭션 모델과 함께 중첩된 트랜잭션 모델은 직렬가능성보다 약한 정확성 평가기준을 가지게 되므로 정확한 또는 일관성 있는 트랜잭션 스케쥴링을 위해서는 여러 가지 보완책을 마련해야 한다는 문제점이 상존하게 된다.

2.3 일관성의 정도

기존의 [6]에서 제안된 일관성 정도(degree of consistency)는 트랜잭션이 데이터베이스에 대해서 지키는 일관성의 정도를 나타내며, SQL-92 표준에 따르면 고립 수준(isolation level)이라고 표현되기도 한다. [6]에서 제시된 일관성의 정도는 <표 1>과 <표 2>에 정리되어 있다.

<표 1> 일관성의 정도에 따라 나타나는 현상

P0	한 트랜잭션이 변경하려는 데이터를 완료하기 전에 다른 트랜잭션이 변경하지 않는다.
P1	한 트랜잭션이 변경하려는 데이터를 완료하기 전에 다른 트랜잭션이 읽어들이지 않는다.
P2	한 트랜잭션에 의해 읽혀지고 있는 데이터는 완료하기 전에 다른 트랜잭션에 의해 변경되지 않는다.
P3	한 트랜잭션에 의해 어떤 조건으로 읽혀지고 있는 데이터 집합 중에서 어떤 데이터 항목도 완료하기 전에 다른 트랜잭션에 의해 변경되지 않는다.

그러나 [1]에서 지적된 바로는 기존의 일관성의 정도인 Degree 2 Consistency와 Degree 3 Consistency 사이에 PL-2.99의 수준이 존재한다는 것이다. [1]에서 제시된 "일반화된 고립 수준(generalized isolation level)은 <표 3>로 정리되어 있다.

<표 2> 일관성의 정도

일관성의 정도	고립 수준	현상
Degree 0		없음
Degree 1	READ UNCOMMITTED	P0
Degree 2	READ COMMITTED	P0, P1
Degree 3	SERIALIZABLE	P0, P1, P2, P3

즉, 직렬가능성이 지켜졌을 때 보장되는 일관성의 수준인 Degree 3 Consistency와 한 트랜잭션에 의해 읽혀지는 데이터 항목이 다른 트랜잭션에 의해 변경될 수 있는 일관성의 수준인 Degree 2 Consistency 사이에, 한 트랜잭션이 특정 술어에 의해 지정된 데이터 항목들만을 읽어 들이되 다른 트랜잭션에 의해 변경될 수 있는 PL-2.99라는 일관성 수준이 존재한다. 또한 PL-2.99 수준은 select 질의의 where 절에 조건을 지정하고 이 조건에 해당하는 결과 집합에 대해서는 질의가 수행된 순간의 값만을 받아들이고, 동시에 수행된 쓰기 연산에 의해 데이터 항목이 변경되었다고 하더라도 그 질의 자체로 데이터베이스의 일관성이 깨지지 않았기 때문에 이러한 트랜잭션들의 스케쥴을 허용하는 수준을 의미한다.

<표 3> 일반화된 고립 수준

일반화된 고립수준	고립 수준	현상
PL-1	READ UNCOMMITTED	P0
PL-2	READ COMMITTED	P0, P1
PL-2.99	READ REPEATABLE	P0, P1, P2
PL-3	SERIALIZABLE	P0, P1, P2, P3

3. 트랜잭션의 단계분할

3.1 시맨틱스를 이용한 단계분할

직렬가능성 이외의 정확성 평가기준을 사용하여 트랜잭션을 스케쥴링하는 방법은 대개 트랜잭션의 시맨틱스를 사용하게 되는데, 이러한 방법은 분산 컴퓨팅 환경에서의 트랜잭션을 위해 [4]와 [5]에서 제안되었다. 시맨틱스란 트랜잭션의 시맨틱 타입, 트랜잭션의 단계, 호환성 집합, 보정 단계, 트랜잭션의 중단점(breakpoint) 등의 의미적 정보를 뜻한다. 기존의 여러 연구에서는 이러한 트랜잭션의 시맨틱스를 구성하는 요소들 중에서 몇 가지를 선택하여 트랜잭션 시맨틱스에 기반하여 일관성을 보장하려는 기법들을 제시했다.[2]

본 연구에서는 직렬가능성보다 다소 약하면서 이동 컴퓨팅 환경에 적합한 PL-2.99 이상의 고립 수준을 가지는 시맨틱스를 지정한다. 그리고 이에 기반하여 트랜잭션을 여러 단계로 분할하여 직렬가능성에 의해 발생되는 스케쥴보다 좋은 동시성을 보여주는 스케쥴을 만들어내는 기법을 제시하고자 한다.

[4]에서 제안된 단계분할된 트랜잭션은 중첩된 트랜잭션 모델의 단순화된 형태이며, 전체 트랜잭션을 여러 단계(step)로 분할하여 실행 시에 그 단계들끼리 인터리빙을 허용한다. 그러나 이 모델은 트랜잭션 스케쥴(schedule)의 정확성(correctness)을 보장하기 어렵다는 문제점을 내포하고 있다. 단계분할된 트랜잭션의 단계들끼리 인터리빙이 허용되는 경우, 부분 결과를 읽어 들이거나 변경함으로써 데이터베이스의 일관성(consistency)이 훼손될 수 있기 때문이다. 그러나 트랜잭션의 시맨틱스를 이용하여 프로그래머가 단계를 분할하고 각 단계의 시맨틱 타입을 미리 명세화 함에 따라 정확한 스케쥴을 생성할 수 있다.

3.2 시맨틱 타입

전체 트랜잭션이나 서브트랜잭션의 타입을 포함하고 있는 연산들의 기능적 의미에 따라 다음과 같이 크게 3가지로 분류할 수 있다.

- read-only transaction : 완료된 값을 읽어 들이는 트랜잭션으로 반드시 데이터베이스에 위치한 데이터 항목의 일관된(consistent) 값을 읽어 들이는 것을 보장할 때 사용될 수 있는 타입이다.
- reporting transaction : 완료되지 않은 부분 결과를

읽어 들이는 트랜잭션으로 빠른 응답과 읽어 들일 데이터 항목의 값의 순간적인 변경에 크게 영향을 받지 않을 수 있을 때 사용될 수 있는 타입이다.
update transaction : 읽기(read) 연산 뿐만 아니라 쓰기(write) 연산까지도 수행하는 트랜잭션으로 READ-ONLY 타입의 서브트랜잭션과는 동시에 수행될 수 없지만, REPORTING 타입의 서브트랜잭션과는 동시에 수행되는 것이 허용된다.

3.2 인터리빙을 허용하는 조건

어떤 트랜잭션의 단계가 수행될 때, 다른 트랜잭션의 단계가 인터리빙하는 것을 허용하거나 허용하지 않는 조건은 트랜잭션 단계의 타입에 따라 <표 4>와 같이 정의된다.

<표 4> 트랜잭션 단계의 타입 별 허용 여부

Type	READ-ONLY	REPORTING	UPDATE
READ-ONLY	O	O	X
REPORTING	O	O	O
UPDATE	X	O	X

REPORTING 타입의 트랜잭션 단계인 경우, 완료되지 않은 부분 결과를 읽어 들여도 트랜잭션의 기능적 의미를 손상시키지 않는 경우라면 UPDATE 타입의 단계와의 동시 수행을 허용하는 것이다. 그러므로 REPORTING 타입의 단계가 수행되는 도중에 다른 트랜잭션에 의해 쓰기 연산이 수행되어 데이터 항목의 값이 변경되더라도 트랜잭션의 작업에 크게 영향을 미치지 않으며, 쓰기 연산 때문에 읽기 연산이 지연되는 것을 피함으로써 빠른 응답을 원할 때 사용 가능한 타입이다. 그러나 일관성을 보장해야 하는 읽기 연산이 필요한 경우에는 READ-ONLY 타입의 트랜잭션 단계를 사용하여 UPDATE 타입의 단계와 인터리빙되지 않도록 방지할 수 있다. 그리고 물론 UPDATE 타입의 단계와 UPDATE 타입의 단계는 인터리빙되지 않도록 방지해야 한다.

3.3 알고리즘

본 연구에서는 서로 다른 트랜잭션 단계들 사이의 인터리빙을 허용하거나 허용하지 않을 것을 판단하는 알고리즘을 제시한다.

임의의 서로 다른 트랜잭션 T_i와 T_j에 대해, T_i의 시작시각이 T_j의 시작시각보다 빠르다고 가정하는 경우, T_i의 k번째 단계 S_k부터 완료까지의 모든 단계(step)를 순서대로 포함하는 단계들의 시퀀스(sequence)를 서브트랜잭션 ST_{ik}라고 정의한다.

트랜잭션 관리자에서 두 트랜잭션 T_i와 T_j가 수행 중이라면, 먼저 수행되고 있던 T_i에 대해서 T_j가 동시에 수행되기 위해서는 현재 시각에 수행될 서브트랜잭션인 ST_{ik}와 ST_{jl}이 동시에 수행되어도 데이터 일관성을 손상시키지 않아야 할 것이다. 이러한 조건들은 크게 4가지로 나뉘는데, 두 트랜잭션이 접근하는 데이터 항목 중 임의의 데이터 항목을 x와 y라고 하면, 첫째, 두 데이터 항목 x와 y가 서로 다른 데이터베이스에 속해 있거나, 둘째, 두 데이터 항목 x와 y가 서로 다른 테이블에 속해 있거나, 셋째, 두 트랜잭션 T_i와 T_j가 수행하는 질의의 술어가 서로 소이거나, 마지막으로, 두 서브트랜잭션 ST_{ik}와 ST_{jl}의 시맨틱 타입이 인터리빙을 허용하는 경우에 속하는 경우가 이에 해당한다.

ST_{ik}의 인터리빙 허용 여부를 판단하는 알고리즘을 정리해 보면 다음의 <그림 1>과 같다.

```
function isAllowableInterleaving(step S_ik, step S_jl)
  for each (x ∈ access_item_set(S_ik), y ∈ access_item_set(S_jl))
    if (x ∈ db_n and y ∈ db_m and n ≠ m)
      return true
    else if (x ∈ table_r and table_s and r ≠ s)
      return true
    else if (predicate(S_ik) and predicate(S_jl) are disjoint)
      return true
    else if (type(ST_ik) is compatible with type(ST_jl))
      return true
    else
      return false
```

<그림 1> 인터리빙 허용 여부를 판단하는 알고리즘

트랜잭션의 시맨틱스를 이용한 단계분할 기법의 장점으로 크게 두 가지를 들 수 있다. 첫째로 트랜잭션 전체에 걸쳐 유효한 록 때문에 다른 트랜잭션의 진행이 방해되던 것을 피하고, 단계들끼리 인터리빙되어 동시에 수행될 기회를 더 많이 가질 수 있다는 점과 둘째로 시맨틱 타입을 세분화하여 REPORTING 타입의 트랜잭션 단계가 UPDATING 타입의 단계와 동시에 수행될 수 있도록 허용하여 정확한 값에 의존하지 않는 읽기 작업이 쓰기 작업의 완료까지 지연됨 없이 수행될 수 있도록 허용하는 것이다.

특히 이동 컴퓨팅 환경에서의 대부분의 트랜잭션이 읽기 연산을 위주로 한다는 점을 감안해 보면, REPORTING 타입의 사용의 기회가 많아지고, 결국 READ-ONLY 타입만을 사용하는 경우보다 나은 성능의 트랜잭션 처리를 지원할 수 있다.

4. 결론 및 향후 과제

본 연구에서는 트랜잭션의 시맨틱스를 구성하는 요소인 트랜잭션의 단계와 서브트랜잭션의 시맨틱 타입 등을 지정하고, 이러한 시맨틱스에 기반하여 트랜잭션을 단계분할하여 동시성을 높이는 기법을 제시하였다. 이것은 이동 컴퓨팅 환경에서 빈번하게 발생하는 작업 지연에 적절하게 대처할 수 있는 기법이다.

이후로 본 연구는 인터리빙 여부를 판단하는 알고리즘의 효율적인 개선과 실제적인 성능 평가, 알고리즘을 실제로 구현한 트랜잭션 매니저의 구조 및 단계분할의 명세화 작업에 대해 초점을 맞출 계획이다.

5. 참고 문헌

[1] A. Adya, B. Liskov, P. O'Neil, "Generalized Isolation Level Definitions," ICDE, 2000

[2] A.J. Bernstein, P.M. Lewis, "Transaction Decomposition Using Transaction Semantics," Distributed and Parallel Databases, Vol. 4, No. 1, pp. 25-47, Jan 1996

[3] P.K. Chrysanthis, "Transaction Processing in Mobile Computing Environment," IEEE Workshop on Advances in Parallel and Distributed Systems, pp. 77-83, Oct 1993

[4] A.A. Farrag, M. Tamer Özsu, "Using Semantic Knowledge of Transactions to Increase Concurrency," ACM Transactions on Database Systems, Vol. 14, No. 4, pp. 503-525, Dec 1989

[5] H. Garcia-Molina, "Using Semantic Knowledge for Transaction Processing in a Distributed Database," ACM Transactions on Database System, Vol. 8, No. 2, pp. 186-213, Jun 1983

[6] J.N. Gray, R.A. Lorie, G.R. Putzolu, I.L. Traiger, "Granularity of Locks and Degrees of Consistency in a Shared Data Base," Research Report RJ1654, IBM, Sep 1975

[7] T. Imielinski, B.R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management," CACM, 1994