

이동 세일즈 응용을 위한 이동 트랜잭션 모델

최미선⁰, 김영국
충남대학교 컴퓨터학과
(mschoi, ykim)@cs.cnu.ac.kr

Mobile Transaction Model for Mobile Sales Application

Mi-Seon Choi⁰, Young-Kuk Kim
Dept. of Computer Science, Chungnam National University

요 약

무선 통신 및 컴퓨터 기술의 발달은 사용자로 하여금 이동 중에도 휴대용 컴퓨터를 소지하고 무선 통신을 통해 다양한 비즈니스 트랜잭션을 수행할 수 있는 모바일 오피스(mobile office)환경을 제공하고 있다. 그러나 무선 통신과 휴대용 컴퓨터의 고유한 특성으로 인해 이동 비즈니스 트랜잭션 처리에 유선망에서 사용되는 트랜잭션 처리 모델을 그대로 적용할 경우 트랜잭션 응답시간이 길어지고 데이터 관리가 복잡해지며, 통신 요금이 증가하는 등의 문제점이 발생한다. 본 연구에서는 고가의 무선통신 비용 및 이동 컴퓨터의 배터리 소모를 절감하기 위해 접속단절을 효과적으로 이용하여 회사와 이동 세일즈 사원들 간의 효율적인 비즈니스 트랜잭션 처리 시스템 구축을 위한 트랜잭션 처리 모델을 제안한다.

1. 서론

이동 컴퓨팅(mobile computing)이란 사용자가 이동 중일 때도 휴대용 컴퓨터를 소지하고 무선 통신을 통해 네트워크에 접속할 수 있도록 해주는 컴퓨팅 패러다임을 말한다. 최근 몇 년 동안 활발히 이루어지고 있는 무선 네트워크와 인터넷의 통합은 무선 모뎀을 장착한 노트북과 같은 휴대용 컴퓨터 뿐만 아니라 팜(Palm), 포켓 PC(Pocket PC), 인터넷 지원 셀룰러 폰 등과 같은 상대적으로 저가의 휴대용 무선 디바이스들이 폭발적으로 증가하는 원인이 되고 있다. 사용자들은 휴대용 무선 디바이스를 이용하여 전자 메일을 송수신하거나 웹 사이트를 보고 인터넷 콘텐츠를 액세스 하게 될 것이다. 사무실 밖에서도 휴대용 무선 디바이스를 이용하여 제품을 사고 파는 일 뿐만 아니라 구매 주문 입력 및 승인, 통보 수신, 영업 정보 이용, 서비스 요청 접수 등의 다양한 비즈니스 트랜잭션을 수행할 수 있게 된다. 그러나 이와 같이 휴대용 디바이스와 무선 통신을 이용한 비즈니스 트랜잭션은 언제 어디서나 어떠한 디바이스를 통해서나 원하는 모든 정보를 이용할 수 있는 사용자의 편의성 면에서의 요구는 만족시키고 있지만, 무선 통신의 고유한 특성과 휴대용 무선 디바이스의 이동성과 휴대성으로 인해 기존의 유선망에서 사용되는 트랜잭션 처리 모델을 그대로 적용할 경우 트랜잭션 응답시간이 길어지고 데이터 관리가 복잡해지며, 통신 요금이 증가하는 등의 문제점이 발생한다.

이동 컴퓨팅 환경에서 가장 제한된 2 가지 자원으로는 무선 대역폭(bandwidth)과 무선 디바이스의 배터리(battery) 용량을 들 수 있다. 제한된 통신 대역폭은 송/수신되는 데이터의 양에 제약 가하며, 전송 매체에 대한 외부 간섭(기후, 전자 간섭, 지형 등)에 의해 발생하는 잦은 접속단절은 전송 데이터의 파손 및 분실을 증가시킬 수 있고, 무선 통신 요금이 유선 통신 요금에 비해 상대적으로 비싼 원인이 된다. 유선망에 연결되어 있는 컴퓨터에 비해 무선 디바이스는 전력 사용에 있어 매우 제한적이므로 전력 부족을 위해 의도적인 무선통신망에서의 접속단절을 유발할 수 있다[1,2].

접속단절은 전송 매체에 대한 외부 간섭에 의해 발생하는 비의도적 접속단절과 고가의 무선통신 비용 및 이동 컴퓨터의 제

한된 배터리 전력 소모를 절감하고자 사용자가 의도적으로 접속단절 상태를 유지하는 의도적 접속단절로 분류할 수 있다. 비의도적 접속단절은 비교적 자주 발생하며 접속단절 지속 기간이 짧은 반면에 의도적 접속단절은 접속단절 기간이 오랫동안 지속되는 경향을 보인다.

본 연구에서는 의도적 접속단절을 이용하여 보험회사나 이동 세일즈 전문 회사, 이동 서비스 전문 회사등과 같이 다수의 이동 세일즈 사원들을 보유하고 있는 기업에서 회사와 세일즈 사원들 간의 효율적인 비즈니스 트랜잭션 처리 시스템 구축을 위한 이동 트랜잭션 모델을 제안하고자 한다. 의도적 접속단절을 효과적으로 활용할 경우, 휴대용 무선 디바이스의 배터리 소모와 무선 통신비용 절감이 가능하다. 이동 세일즈 사원들은 이러한 트랜잭션 처리 시스템을 사용하여 언제 어디서나 회사의 엔터프라이즈(enterprise) 서버에 저장되어 있는 담당 고객 정보를 비롯한 관련 정보를 이용하여 상품을 판매한다거나 서비스를 제공하는 것이 가능할 것이다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구로서 기존에 연구되었던 접속단절을 지원하는 이동 트랜잭션 처리 모델에 대하여 기술하며, 3 장에서는 본 연구에서 제안하는 이동 세일즈 응용을 위한 트랜잭션 처리 모델에 대하여 기술하고 마지막으로 4 장에서는 결론 및 향후 계획에 대해 기술한다.

2. 관련 연구

이동 컴퓨팅 환경의 물리적인 특성들은 유선망에 연결된 데이터베이스 서버와 이동 컴퓨터 사용자 간의 트랜잭션 처리에도 많은 영향을 미친다. 이동 컴퓨팅 환경에서 수행되는 이동 트랜잭션의 특징은 다음과 같다[4]. 첫째, 잦은 접속단절, 고가의 통신 비용은 장기 실행 트랜잭션(long-lived transaction)을 유발한다. 둘째, 무선 통신 매체의 특성상 자주 발생하게 되는 접속단절 및 이동 컴퓨터의 파손, 분실 등에 의한 트랜잭션의 실패 확률이 많아지게 된다. 셋째, 이동 중인 이동 컴퓨터의 위치가 변화에 따라 트랜잭션에 연관된 재배치(relocation) 문제가 발생한다. 따라서, 이동 컴퓨팅을 위한 트랜잭션 모델은 사용자에게 네트워크 연결 상태에 관계없는 투명한(transparent) 서비스를 제공하기 위해, 이동 컴퓨터의 트랜잭션 처리에 있어서 접속단절 시에도 트랜잭션을 계속 수행 할 수 있는 자치성을 가지고 있어야 한다. 더불어, 접속단절시의 연산 수행으로 인해 문제가

* 이 연구는 BK21 충남대학교 정보통신인력양성사업단의 지원을 받았음.

되는 고정 호스트의 데이터베이스에 저장되어 있는 원본 데이터와 이동 컴퓨터의 데이터베이스에 저장되어 트랜잭션 처리에 사용되고 있는 데이터간에 일관성이 보장되어야 한다.

Coda는 서버 장애나 네트워크 분할(network partition) 하에서도 높은 수준의 가용성을 달성하기 위한 범용 분산 파일 시스템이다[3]. Coda에서는 클라이언트가 서버와의 접속단절 시에도 서버에 있는 파일에 대한 작업을 계속 수행할 수 있도록 하기 위하여 "hoarding" 기법을 사용한다. 클라이언트는 서버와 장시간 접속 단절이 예상될 때, 지역 캐쉬에 접속단절 동안 사용할 수 있는 파일을 서버에서 전송 받아 local hoard로서 유지한다. 클라이언트는 접속단절 동안에 local hoard를 사용하여 작업을 수행한 후, 서버와 재 연결 시 서버에 있는 원본 데이터와 local hoard 데이터를 일치시키는 재통합(reintegration) 과정을 수행한다. 이러한 낙관적 기법에 기반을 둔 일관성 유지 방법은 재통합시 발생할 수 있는 충돌(conflicts)을 해결해야 하는 문제를 야기하고, 그 과정은 상당한 오버헤드를 요구하게 된다. 그러나 Coda에서는 UNIX 환경에서 두명 이상의 사용자가 동시에 같은 파일을 변경하는 일은 거의 없다는 UNIX 파일 사용 패턴의 특징을 이용하여 충돌이 거의 발생하지 않는다는 전제하에 문제들을 해결한다. 각 응용 프로그램들이 자신의 응용 프로그램에 적당한 재통합 작업을 효과적으로 수행하여 발생할 수 있는 충돌을 해결하기 위한 ASR(Application Specific Resolver)로서 사용되어질 프로그램들을 정의할 수 있는 메커니즘이 제공된다.

Pitura가 제안한 클러스터 모델은 잦은 접속단절이 발생하는 이동 분산 환경에서 데이터베이스의 일관성을 유지하기 위한 트랜잭션 모델이다[5]. 클러스터는 의미적(semantically)으로 연관이 있거나 네트워크가 연결되어 있어 통신이 가능한 상태인 사이트들에 저장되어 있는 데이터의 집합을 말한다. 같은 클러스터내에 있는 데이터의 중복 사본(replicated copy)들은 엄격한 일관성을 유지해야 하는 반면, 서로 다른 클러스터에 저장되어 있는 데이터의 중복 사본 간에는 무선 연결 상태에 따라 어느 정도의 불일치를 허용하게 된다. 이를 위해서 데이터베이스 시스템에 기존의 엄격한 일관성을 요구하는 읽기(read), 쓰기(write) 연산 이외에 완화된 일관성을 보장하는 weak read, weak write 연산이 별도로 추가된다. weak read, weak write 연산은 클러스터 내에서 지역적으로 사용 가능한 중복 데이터를 접근할 수 있도록 해준다. 반면 데이터의 모든 중복 사본들 사이에 엄격한 일관성을 유지해야 되는 경우에는 strict read, strict write 연산을 사용하여 변경된 데이터 값이 중복되어 있는 모든 사이트에 반영되도록 한다는 것이다. 클러스터 구성은 네트워크 연결 상태에 따라 동적으로 변경될 수 있다. 클러스터 모델은 동적으로 변하는 네트워크 연결 상태와 응용에 따라 효과적으로 strict read, strict write 와 weak read, weak write 를 적절히 사용함으로써 무선 대역폭을 효율적으로 사용할 수 있고 전송 지연 및 비용면에서 이득을 얻을 수 있다는 특징을 가진다.

Walborn은 무선 통신 비용 및 전송 지연을 줄일 수 있을 뿐만 아니라 접속단절 시에도 이동 호스트가 자치적으로 고정 호스트에 저장되어 있는 공유 데이터에 대한 연산을 계속 수행할 수 있도록 하는 의미 기반(semantics-based) 트랜잭션 처리 모델로서 "단편화 가능 객체"의 개념을 제안하였다[6]. 이 모델은 데이터베이스에 저장되어 있는 객체의 구조를 이용하여 크고 복잡한 객체를 관련이 있는 데이터끼리 분할하여 작고 다루기 쉬운 단편(fragment)으로 나눈다. 공유 데이터의 원본(master copy)을 가지고 있는 고정 호스트의 데이터베이스 서버는 이동 컴퓨터가 "split" 연산을 사용하여 공유 데이터의 단편을 요청할 경우, 객체의 단편을 이동 호스트에 분배해준다. 일단 이동 호스트에 분배된 데이터베이스 내의 단편은 다른 호스트들에게는 보이지 않게 된다. 이동 호스트는 분배 받은 객체의 단편을 자신의 메모리에 캐쉬해 놓고, 다른 호스트들과 통신할 필요 없

이 캐쉬 내의 단편 내에서만 일관성을 유지하면서 독립적으로 필요한 연산을 수행한다. 필요한 연산이 끝나고 네트워크가 연결된 상태이면, 이동 호스트는 "merge" 연산을 통해 캐쉬되었던 단편을 데이터베이스 서버에 반환하게 된다. 재고 물품의 총 개수나 금액의 합계 등과 같은 총계 항목(aggregate item)이나, 집합(sets), 스택(stack), 큐(queue)등이 단편화 가능 객체의 예이다. 이 모델은 객체 구조의 의미를 이용하여, 객체를 단편으로 나누고 이 단위로 일관성을 유지함으로써 호스트들에서 발생할 수 있는 충돌 가능성을 사전에 제거하고, 이동 컴퓨터에서 캐쉬한 단편에 대해서는 네트워크의 상태와 관계없이 독자적으로 트랜잭션을 수행할 수 있는 장점이 있는 반면, 이 모델이 적용될 수 있는 데이터의 종류는 서로 일관성이 없는 단편으로 분할할 수 있는 특징을 가지는 경우로 제한되는 단점을 가진다.

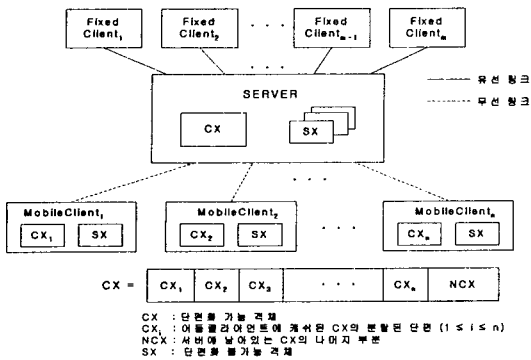
3. 이동 세일즈 응용을 위한 트랜잭션 처리 모델

본 연구에서 제안하는 이동 비즈니스 트랜잭션 처리 시스템은 이동 세일즈 사원들의 휴대용 디바이스들로 이루어지는 이동 클라이언트들과 회사의 엔터프라이즈 서버로 구성되는 클라이언트/서버 모델을 따른다. 이동 클라이언트들은 휴대용 디바이스의 배터리 절약과 무선 통신 비용 절감을 위해 대부분 시간 동안 서버와 접속단절 상태에서 트랜잭션을 수행한다. 서버 데이터베이스에 저장되어 있는 객체들은 대부분 여러 클라이언트들이 공유하는 공유 데이터로 구성된다. 접속단절 동안 이동 클라이언트의 자치적인 트랜잭션 수행을 지원하기 위해서는 접속단절 이전에 필요한 데이터를 미리 이동 클라이언트에 캐쉬해놓아야만 한다. 그러나 서버 데이터베이스 객체들 중 크기가 큰 객체는 무선 전송 비용과 무선 디바이스의 제한된 저장 용량 및 배터리 소모 등을 고려할 때 객체 전체를 이동 클라이언트에 캐쉬하는 것은 많은 오버헤드가 따른다. 만일 객체를 접하지 않는 단편으로 적절히 분할할 수 있다면, 이 단편을 캐쉬의 단위로 하여 이동 클라이언트에 캐쉬하여 사용함으로써 이동 클라이언트의 자치성을 증가시킬 수 있다.

서버에 저장되어 있는 객체는 성격에 따라 단편화가능객체(FO : Fragmentable Object)와 단편화불가능객체(NFO : Non-Fragmentable Object)로 구분할 수 있다. 단편화가능객체는 임의의 이동 클라이언트가 객체의 일부분만을 분할하여 자신의 지역 데이터베이스로 캐쉬한 후, 다른 클라이언트와 충돌없이 독자적으로 읽기/쓰기 연산이 가능한 객체로 정의한다. 단편화가능객체의 예로 보험회사의 고객 테이블을 들 수 있는데 보험 판매원마다 담당 고객이 정해져 있으므로, 각 보험 판매원은 고객 테이블로부터 자신이 담당한 고객 레코드들만을 분할하여 자신의 이동 컴퓨터의 지역 데이터베이스에 저장하여 다른 보험 판매원과 충돌없이 사용할 수 있다. 데이터베이스에 있는 모든 객체가 단편화 가능 객체로만 이루어져 있다면, 동시성 제어의 단위를 객체의 단편으로 했을 때 별도의 동시성 제어가 불필요하며, 관련 연구에서 언급하였던 Walborn의 의미기반 단편화 모델을 적용할 수 있을 것이다.

서버 데이터베이스에 저장되어 있는 또 다른 객체 유형인 단편화불가능객체는 단편화가 불가능하며 객체 전체를 여러 클라이언트들이 공유하여야만 하는 객체로, 여러 클라이언트들이 동시에 이 객체에 읽기/쓰기 연산을 제출할 수 있기 때문에 동시성 제어가 필요한 객체이다. 단편화가능객체는 객체 그 자체로는 다른 클라이언트들과 충돌이 발생하지 않지만 단편화불가능객체의 종속성(dependency)을 가지는 트랜잭션의 경우에는 종속성으로 인해 간접적인 충돌이 발생할 수 있다.

이동 클라이언트가 서버 데이터베이스에서 객체의 단편을 지역 데이터베이스에 캐쉬할 수 있도록 split/merge 의 2 가지 연산이 필요하다. split 연산은 이동 클라이언트가 서버로부터 의도



< 그림 1 > 시스템 모델

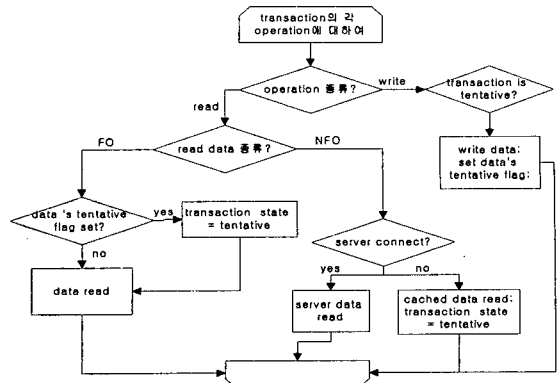
적인 접속단절 상태를 유지하고자 할 때, 트랜잭션 처리에 필요한 데이터 객체를 필요 부분만 분할하여 지역 데이터베이스로 복사하는 연산이다. merge 연산은 이동 클라이언트가 지역 데이터베이스에 복사하였던 데이터 객체의 단편을 서버에 저장되어 있는 원래의 객체에 반환하는 연산을 의미한다.

<그림 1>은 본 연구에서 제안하는 이동 비즈니스 트랜잭션 처리를 위한 시스템 모델이다. 서버 데이터베이스는 단편화가능 객체와 단편화불가능객체들로 구성된다. 클라이언트들은 유선망에 연결된 클라이언트(Fixed Client)와 무선망에 의해 연결된 이동 클라이언트(Mobile Client)의 2 종류가 있다. 이동 클라이언트는 서버에게 단편화가능객체를 단편으로 분할하여 지역 데이터베이스에 캐쉬하기 위해 객체의 분할 조건과 함께 split 연산을 제출한다. 서버는 요청된 객체 부분이 다른 클라이언트로 캐쉬되지 않았다면, 요구된 객체의 단편을 클라이언트에게 전송한다. split 연산에 의해 각 이동 클라이언트에 분할되어 캐쉬된 단편 CX_i에 대해서는 merge 연산이 제출되기 전까지는 이동 클라이언트가 단편 CX_i의 마스터(master)가 되고, 서버 데이터베이스에서 변경이 불가능한 상태가 된다. 이동 클라이언트는 트랜잭션 수행을 위해 단편화불가능객체가 필요한 경우, 객체의 크기에 따라 캐쉬 오버헤드가 크지 않은 경우, 자신의 지역 데이터베이스에 캐쉬할 수 있다.

이동 클라이언트에서 수행되는 트랜잭션을 서버와의 일관성 유지를 위한 재통합 과정의 필요 유무에 따라 마스터트랜잭션(master transaction)과 임시트랜잭션(tentative transaction)으로 분류한다. 마스터트랜잭션은 이동 클라이언트가 마스터인 데이터만을 사용하는 트랜잭션으로 이동 클라이언트에서 트랜잭션을 커밋하게 되면 서버에 재연결 되었을 때 별도의 일관성 조정 과정이 필요없는 트랜잭션이나 임시트랜잭션은 서버와의 재통합 과정의 결과에 따라 최종 결과가 변경될 수 있다.

<그림 2>는 이동 클라이언트에서의 트랜잭션 처리 방법을 보여준다. 단편화가능객체(FO)의 각 데이터 마다 tentative flag를 유지하여 merge 연산 시에 데이터 값에 대한 조정 과정이 필요하지 유무를 표시하도록 한다. 서버와의 접속단절 상태에서 단편화 불가능객체(NFO)의 값을 읽어서 단편화가능객체(FO)의 값을 변경 시켰다면 트랜잭션은 임시트랜잭션이 되고, 임시트랜잭션의 로그는 별도의 재통합 로그에 기록되어 일관성 조정 작업에 사용되도록 한다. 임시 트랜잭션에 의해 변경된 데이터는 데이터의 tentative flag=1로 지정되어, 이후, 이 데이터를 읽게 되는 트랜잭션들은 임시트랜잭션이 된다. 이동 클라이언트는 휴대용 디바이스의 배터리 잔량이나 통신 상태, 트랜잭션 종류에 따라 서버와의 접속 상태를 유지할 수도 있다.

이동 클라이언트가 서버에게 merge 연산을 제출하게 되면, 서버는 데이터 일관성을 맞추기 위한 재통합 과정을 실행한다. 이동 클라이언트는 접속단절 동안 유지하였던 재통합 로그를



< 그림 2 > 트랜잭션 처리 알고리즘

서버에게 전송하여 서버에서 그동안 단편화 불가능 객체에 대해 발생하였던 연산들과의 조정 작업을 실행한다.

4. 결론 및 향후 연구 방향

본 논문에서는 이동 컴퓨팅 환경에서의 비즈니스 트랜잭션 처리 시스템 구축에 필요한 트랜잭션 모델과 트랜잭션 처리 알고리즘을 제안하였다. 무선 통신 비용과 이동 컴퓨터의 배터리 전력 소모를 줄이기 위해 대부분 시간동안 서버와 접속단절 상태에서 비즈니스 트랜잭션을 처리할 수 있도록 하기 위해 서버 데이터베이스 객체를 특성에 따라 단편화가능객체와 단편화 불가능객체로 분류하고 각각을 위한 캐쉬 전략을 제안하였다. 서버와의 데이터 일관성 유지를 위해 이동 컴퓨터에서의 트랜잭션을 마스터트랜잭션과 임시트랜잭션으로 분류하고 트랜잭션 처리 알고리즘을 제시하였다. 제안된 모델은 접속 단절을 기존의 연구들에서와 같이 무선 통신의 특성에 의해 직면하게 되는 피할 수 없는 수동적인 접속 단절로 다루는 대신 접속 단절을 능동적으로 이용하여 이동 컴퓨터에서 수행되는 트랜잭션의 종류와 이동 컴퓨터의 작업 조건에 맞게 적절하게 스케줄함으로써 고가의 무선 통신 비용과 무선 통신 오버헤드에 따르는 이동 컴퓨터의 전력 사용량을 줄일 수 있는 특성을 가진다.

향후 계획으로는 본 연구에서 제안한 트랜잭션 처리 모델의 성능을 평가하기 위한 성능 평가 실험을 실시할 예정이다.

참고문헌

- [1] R. Alonso and H. Korth, Database issues in nomadic computing, In Proc. Of the ACM SIGMOD Conference on Management of Data, pp. 388-392, 1993.
- [2] Tomasz Imielinski and B. R. Badrinath, Mobile wireless computing: Solutions and challenges in data management, Technical Report DCS-TR-296, Dept. of Computer Science, Rutgers Univ., New Brunswick, NJ08903, 1993
- [3] James Kistler and M. Satyanarayanan, Disconnected operation in the CODA file system, ACM Trans. on Computer Systems, Vol. 10, No. 1, pp. 3-25, Feb. 1992.
- [4] Pitoura, E., and Bhargava, B. Revisiting Transaction Concepts for Mobile Computing. In first IEEE Workshop on Mobile Computing Systems and Applications, 1994
- [5] Pitoura, E., and Bhargava, B., Wolfson O., Data Consistency in Intermittently Connected Distributed Systems. IEEE Transaction on Knowledge and Data Engineering, 1998
- [6] Chrysanthis, P. K. Walborn G. Supporting Semantics-Based Transaction Processing in Mobile Database Applications, In Proceedings of the 14th IEEE Symposium on Reliable Distributed Systems, Sept. 1995