이동 컴퓨팅 환경에서 적응적 요청 메시지를 이용한 갱신 트랜잭션 스케줄링 기법

박 준[†] · 채 덕 진[†] · 황 부 현[††] · 김 중 배[†††] · 정 승 욱[†††]

요 약

많은 수의 클라이언트를 갖는 이동 컴퓨팅 환경에서 방송 기술은 낮은 대역폭의 문제를 해결할 수 있다. 그러나 기존에 제안한 대부분의 동시성 제어 기법들은 이동 클라이어언트에서 발생하는 거래를 질의 거래로 제한하였다. 본 논문에서는 캐쉉과 방송기술을 이용하고 이동 클라이언트에서 갱신 거래를 허락하는 UTSM-ARM(An Update Transaction Scheduling Method Using an Adaptive Request Message)을 제안한다. UTSM-ARM은 이동 클라이언트의 캐쉬 데이터에 대한 유효성을 데이터의 동적인 갱신 패턴 비율을 기준으로 판단할 수 있다. 또한 이동 클라이언트에서 접근하는 캐쉬 데이터와 이동 트랜잭션에 대한 타임스탬프는 이동 트랜잭션의 직렬가능한 수행을 보장한다. 그 결과, UTSM-ARM은 방송환경의 비대칭적인 대역폭을 효율적으로 사용할 수 있으며 이동 트랜잭션의 수행 성능을 향상시킬 수 있다.

An Update Transaction Scheduling Method Using an Adaptive Request Message in Mobile Computing Environments

Joon Park[†] · DuckJin Chai[†] · Buhyun Hwang[††]
JoongBae Kim[†††] · SeungWoog Jung[†††]

ABSTRACT

A broadcast method in mobile computing environments which have very large client populations solves the problem of low bandwidth. But most of previous proposed concurrency control protocols are restricted to read-only transactions from mobile client. In this paper, we propose the UTSM-ARM method which uses caching and broadcast method, and allows update transactions in mobile client. The UTSM-ARM decides the validation of cache data consistency with the dynamic update pattern ratio of accessed cached data. Also, the timestamps of accessed cached data and transaction in mobile client guarantee the serializable execution of mobile transactions. As a result, UTSM-ARM makes efficiently use of the asymmetric bandwidth of broadcast environment and can increase the transaction throughput.

키워드 : 이동 컴퓨팅(Mobile Computing), 방송(Broadcasting), 캐쉬 일관성(Cache Consistency), 낙관적 동시성 제어(Optimistic Concurrency Control), 타임 스탬프(Timestamp)

1. 서 론

최근 무선 통신 기술의 급속한 발전과 이동 가능한 통신 기기의 대중화로 인하여 시간과 장소에 제약을 받지 않고 사용자들은 언제 어디에서나 이동 중에도 다양한 정보 서비스를 제공받을 수 있다. 또한 무선 통신 사용자들의 실시간 교통 정보, 항해 정보 그리고 실시간 주식 정보와 같은 다양하고 혁신적인 실시간 정보 서비스들도 계속해서 출현시키고 있다. 그러나 이러한 이동 가능한 컴퓨터들은 처리 능력이나 저장 용량, 사용자 인터페이스 및 신뢰성이 유선 환경의 컴퓨터 장비들에 비하여 낮은 성능을 보인다. 이동 컴퓨팅 환경이 갖는 이동성, 통신의 비신뢰성, 협소한 대역폭 그리고 제한된 전력공급 등의 제약에 대하여 서버는 방송을 통한 통신과 캐쉬를 사용한다[6-8, 10]. 캐쉬의 사용은 이동 클라이언트에서 서버로의 잦은 연결을 줄일 수 있고, 이동 클라이언트와 서버와의 통신 단절 문제의 해결 방법이며 이동 트랜잭션의 수행의 속도를 향상시킨다[12].

이동 컴퓨팅 환경에서 이동 클라이언트는 트랜잭션을 사용하여 상호 작용한다[5, 13, 14]. 이동 컴퓨팅 환경에서 수행되는 트랜잭션은 데이터의 읽기 연산만으로 구성된 읽기 전용 트랜잭션과 갱신 연산을 포함하고 있는 갱신 트랜잭션으로 분류할 수 있다[11]. 이동 트랜잭션의 정확한 수행을 보장하기 위해서 가장 중요한 과제는 캐쉬 일관성이다.

※ 본 논문은 2002년 한국전자통신연구원 모바일 비즈니스 응용서버 기술개발('3010-2002-0019')비에 의하여 연구되었음.
† 준 회 원 : 전남대학교 대학원 전산학과
†† 종신회원 : 전남대학교 컴퓨터정보학부 교수
††† 정 회 원 : 한국전자통신연구원 컴퓨터소프트웨어기술연구소 모바일응용서버연구팀
논문접수 : 2003년 8월 19일, 심사완료 : 2003년 11월 12일

캐쉬 일관성을 유지하기 위하여 서버는 갱신된 데이터에 대한 정보를 주기적인 방송을 통하여 이동 클라이언트에 전달한다. 그러나 이동 컴퓨팅 환경은 이동성이라는 편리함 때문에 데이터베이스 영역에서 빠른 속도로 확산 발전되고 있지만 사용자의 욕구를 충족시키기 위한 양질의 서비스를 제공하는데 있어 해결해야 할 많은 문제점들이 남아 있다.

다수의 이동 클라이언트에 방송을 통한 데이터 전달방법은 캐쉬 일관성을 유지하기 위한 유용한 기술이다. 그러나 데이터베이스의 데이터 변화가 빈번할 경우, 이동 클라이언트의 캐쉬 데이터의 유효화 시간이 짧아진다. 결국, 이동 클라이언트의 무효화된 캐쉬 데이터를 접근한 이동 트랜잭션의 철회율도 증가하게 된다. 따라서 이동 클라이언트의 캐쉬 데이터의 최신성과 상호 일관성을 위한 방송 기술에 관한 연구도 필요하다. 지금까지 제안된 몇몇 연구에서는 이동 클라이언트의 빠른 트랜잭션을 수행하기 위하여 읽기전용 트랜잭션에서 접근하는 데이터에 대한 참조 비율을 고려한 캐쉬 일관성 유지 방법이었다[18, 20]. 예를 들어, 날씨 예보와 교통 정보 시스템과 같은 대부분의 실시간 시스템은 갱신 트랜잭션이 읽기전용 트랜잭션보다 발생하는 확률이 낮기 때문이었다[15, 16]. 그러나 경매 시스템, 주식 거래 시스템과 같은 실시간 응용 시스템은 갱신 트랜잭션의 발생 비율이 높다. 즉, 이동 컴퓨팅 기술의 발달과 사용자의 다양한 서비스 요구에 따라 이동 클라이언트에서 발생 가능한 갱신 트랜잭션은 서버의 데이터베이스에 직접적인 변화를 줄 수 있다.

따라서, 갱신 트랜잭션이 발생할 수 있는 이동 클라이언트의 캐쉬 데이터에 대한 특성을 고려하고 이동 컴퓨팅 환경의 제한된 대역폭의 사용을 줄이며 이동 트랜잭션의 응답시간을 최소화 할 수 있는 이동 트랜잭션의 스케줄링 방법이 필요하다.

본 논문의 구성은 다음과 같다. 제2장에서는 이동 컴퓨팅 환경과 이동 컴퓨팅 환경에서 캐쉬 일관성 유지기법과 이동 트랜잭션의 동시성 제어 기법들을 소개하고, 제3장에서는 제안된 동시성 제어 기법의 직렬성이 위배되는 문제를 확인한다. 그리고 제4장에서는 본 논문에서 제안하는 적응적 요청 메시지를 이용한 이동 갱신 트랜잭션 스케줄링 방법인 UTSM-ARM을 제시하고, 제5장에서는 모의실험을 통하여 기존의 제안된 기법과의 성능을 비교 분석하였다. 마지막으로 제6장에서는 결론과 향후 연구방향을 제시한다.

2. 관련 연구

이동 컴퓨팅 환경에서 많은 이동 클라이언트의 캐쉬 일관성을 유지하기 위하여 주기적으로 무효화 보고서(invalidation report)를 방송하는 방법이 비용 효과적이다[2]. 무효화 보고서는 이전 방송 주기 동안에 완료 요청된 이동 트랜잭션들 중에서 트랜잭션 완료에 의하여 갱신된 데이터 정보를 데이터 스트림(Data Stream)으로 구성하여 이동 클라이언트에 전달된다. 그리고 다음 방송 때에 이동 클라이언트는 방송을 듣고서 갱신된 데이터의 대한 정보를 캐쉬 데이터에 반영한다. 만약, 이동 클라이언트의 캐쉬 데이터에 갱신된 데이터가 존재한다면 이동 클라이언트는 캐쉬된 데이터를 버리고 갱신된 데이터를 얻음으로써 캐쉬 데이터의 일관성을 유지할 수 있다[1]. TS(The broadcasting Time Stamp scheme)기법은 지난 방송주기 동안에 갱신된 데이터와 현재 방송되는 무효화 보고서에 대한 각각의 타임스탬프를 추가하여 전달함으로써 캐쉬 데이터의 일관성을 유지할 수 있는 기법이다[3]. 그러나 이동 클라이언트의 단절이 발생하면 이동 클라이언트의 캐쉬 데이터에 대한 일관성을 보장할 수 없으므로 다음 방송주기 동안 기다리거나, 캐쉬 데이터를 접근하는 모든 데이터에 대한 정보를 서버에 요구함으로써 제한된 대역폭의 사용을 증가시킬 수 있는 단점이 있다. 이러한 문제를 해결하기 위한 캐쉬 데이터의 유효성을 서버에 요청하는 기법[17]들이 제안되었다. 그리고 주기적으로 전달되는 무효화 메시지를 각 데이터에 대한 정보를 비트열로 구성하여 이동 클라이언트에 전달하는 BS(The Bit Sequence scheme)기법[12]이 제안되었다. 이 기법은 이동 클라이언트의 단절에 매우 유통성이 있으며 효율적이다. 그러나 전체 데이터베이스에 대한 정보를 비트열로 구성하기 때문에 서버에서 이동 클라이언트에 최근에 갱신된 정보를 전달하기 위하여 사용하는 대역폭(downlink)의 사용량이 많다는 단점을 가지고 있다. 또한 위의 두 가지 장점을 이용하여 이동 클라이언트의 단절에 적절히 대응할 수 있는 적응적 방송 기법에 대하여 제안한 연구도 있다[1]. 그리고 이동 트랜잭션의 철회율을 줄이기 위하여 무효화 보고서의 방송주기 동안에 갱신된 데이터에 대한 정보를 추가 방송함으로써 캐쉬 데이터의 유효성을 유지하고 트랜잭션의 완료율을 증가시킬 수 있는 기법[4]이 제안되었다. 그리고 데이터 방송환경에서 이동 클라이언트는 순차적으로 데이터를 접근하게 되는데, 질의 최적화(트랜잭션이 데이터를 읽는 방송 주기의 개수를 줄임)를 위하여 읽기 집합(ReadSet)의 기선언 기법을 채택[18, 20]한 방법이 제안되었다.

이동 컴퓨팅 환경에서의 효율적인 이동 트랜잭션의 동시성 제어 방법은 제한된 대역폭 사용을 줄이면서 이동 트랜잭션의 응답시간과 완료율을 높일 수 있는 방법이 요구된다. 이동 클라이언트/서버 컴퓨팅 환경에서의 캐쉬 및 동시성 제어 방법으로 방송기반의 캐쉬무효화 기법을 사용하면서 캐쉬 데이터의 일관성을 지원하는 OCC-UTS(Optimistic Concurrency Control with Update TimeStamp)를 제안하였다[19]. 이 방법의 초점은 방송기술의 사용과 접근한 데이터에 대한 유효성 검사 및 완료 프로토콜은 캐쉬 무효화 과정의 내부과정인 분산 형태로 구현하였고, 일관성 확인과

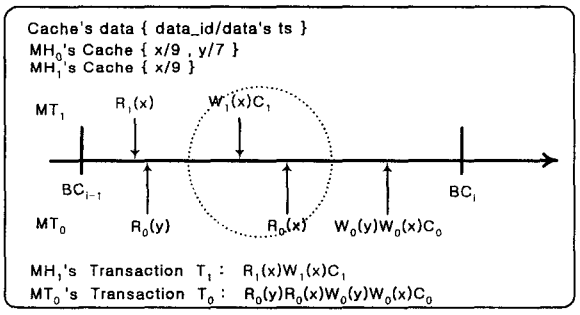
정의 대부분을 이동 클라이언트에서 수행하여 철회될 수 있는 이동 트랜잭션을 알아낼 수 있고 재 수행이 가능하다는 것이다. 그러나 제안한 방법의 문제점은 이동 트랜잭션이 많은 방송 환경에서 캐쉬 데이터의 일관성을 유지하기 위하여 타임스탬프를 부여한 데이터를 이동 클라이언트에 전달 할 경우, 갱신 트랜잭션과 읽기전용 트랜잭션의 충돌 발생으로 읽기전용 트랜잭션의 철회율이 증가하여, 시스템의 수행성능이 감소시킨다. 그리고 이 문제를 해결하기 위한 방법으로 이동 컴퓨팅 방송 환경에서 캐쉬 내에 두 버전을 사용함으로써 타임스탬프 기법으로 클라이언트가 읽기전용 트랜잭션을 자체 해결할 수 있도록 하는 OCC/2VTS (Optimistic Concurrency Control based on 2-version and TimeStamp for Broadcast Environment)를 제안하였다[21]. 이 기법은 읽기전용 트랜잭션 후 두 번의 무효화 방송을 통하여 이동 클라이언트에서 수행하는 읽기전용 트랜잭션이 갱신 트랜잭션의 완료와 상관없이 무사히 완료될 수 있도록 하였다. OCC/2VTS는 서버에게 완료 요구를 위한 이동 호스트와 서버의 통신 횟수를 줄이고, 무효화 보고서 내에 갱신된 최신의 값을 포함하여 클라이언트들에게 방송함으로써 비대칭적인 대역폭을 효율적으로 활용할 수 있다. 또한 읽기전용 트랜잭션의 완료률을 최대한 높여 시스템 성능을 향상시켰다. 그러나 방송기법을 사용하는 낙관적 동시성 제어기법에서는 이동 클라이언트에서 수행하는 대부분의 이동 트랜잭션을 갱신 트랜잭션보다는 읽기전용 트랜잭션의 비율이 높다고 기능을 축소화하였다. 그리고 OCC/UTF (Optimistic Concurrency Control with Update Transaction First)[22]는 갱신거래에 의해 무효화된 데이터를 먼저 읽기 연산한 질의 거래가 비 직렬성 가능성으로 인한 철회율을 줄이는 방법이다. 그 결과 방송기반의 비대칭적인 대역폭을 효율적으로 사용할 수 있는 결과를 얻을 수 있었다. 그러나 OCC/UTF 방법은 갱신 트랜잭션에 의해 최종 완료된 데이터를 참조하지 않고 갱신될 것으로 예상되는 캐쉬 데이터를 참조함으로써 데이터의 최신성 및 일관성이 떨어진다는 문제점이 있다. 또한 갱신 트랜잭션이 가능한 OCC-UTS에서는 이동 트랜잭션에서 접근하는 캐쉬 데이터에 대한 특성을 전혀 고려하지 않았다. 그리고 이동 갱신 트랜잭션에 의해서 접근되는 데이터에 대한 특성을 배제함으로써 갱신 비율이 높은 데이터를 참조한 이동 트랜잭션과 읽기전용 트랜잭션 사이의 직렬성이 위배되는 경우가 발생한다. 결국, 이동 트랜잭션의 철회율이 증가하여 수행 성능을 떨어뜨리는 문제점이 남아있다.

3. 문제 제시

이동 컴퓨팅 환경에서 캐쉬 데이터의 일관성을 유지하기 위하여 서버는 주기적인 무효화 보고서를 방송한다. 주기적으로 방송되는 무효화 보고서는 갱신된 데이터에 대한 가장 최근의 정보를 이동 클라이언트에 전달하고, 이동 클라이언트는 방송되는 무효화 보고서를 듣고 캐쉬 일관성을 보장한다. 그리고 캐쉬 데이터를 접근하여 수행하는 이동 트랜잭션은 가장 최근의 데이터를 접근함으로써 이동 트랜잭션에서 참조하는 데이터에 대한 최신성을 만족한다. 그러나 이동 컴퓨팅 기술의 발전과 함께 이동 클라이언트에서 갱신 트랜잭션이 가능한 경우에 캐쉬 일관성 유지와 이동 트랜잭션의 직렬성 그리고 제한된 대역폭의 효율적인 사용에 대한 연구는 아직 미흡하다. 이동 클라이언트/서버 환경에서 서버의 방송을 통하여 캐쉬 일관성을 유지하고 각 데이터에 대하여 타임스탬프를 부여한 이동 갱신 트랜잭션 규약인 OCC-UTS(Opimistic Concurrency Contro with Update TimeStamp)가 제안되었다[19]. 이 방법은 주기적으로 무효화 보고서를 방송하는 환경에서 하나의 공유 데이터를 다수의 이동 클라이언트에서 읽기 연산이 아닌 갱신 연산이 가능한 경우, 이동 트랜잭션의 직렬성이 위배되는 경우가 발생한다.

3.1 하나의 방송주기에 발생 가능한 이동 트랜잭션의 직렬성 위배

(그림 3-1)은 주기적인 무효화 보고서의 방송 시점인 BC_{i-1}에서 모든 이동 클라이언트(MH_0, MH_1)의 캐쉬 데이터에 대한 일관성이 유지되었다고 가정한다.



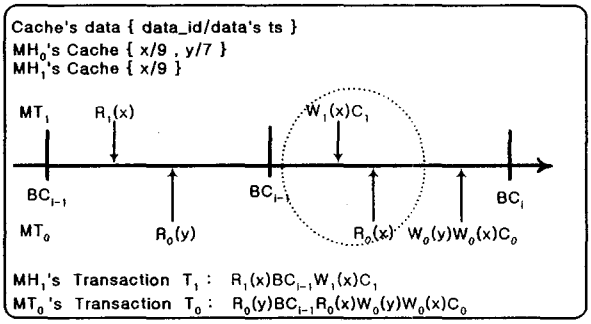
(그림 3-1) 하나의 방송주기에 이동 트랜잭션의 직렬성 위배

방송 시점인 BC_{i-1}과 BC_i사이에 이동 클라이언트(MH_1)는 이동 트랜잭션 T_1인 $R_1(x)W_1(x)C_1$은 완료 요청된다. 이동 클라이언트(MH_0)의 이동 트랜잭션 T_0은 $R_0(y)R_0(x)W_0(y)W_0(y)C_0$을 완료 요청한다. 그러나 이동 클라이언트($MH_0$)은 이동 트랜잭션 T_0의 연산 $R_0(x)$에서 접근하는 캐쉬 데이터(x)를 수행하는 도중에 임의의 이동 트랜잭션 T_1의 연산 $R_1(x)W_1(x)C_1$은 완료된다. 즉, 데이터 x는 데이터베이스에 영구 반영된다. 결국, 이동 트랜잭션 T_0의 $R_0(x)$는 무효화된 데이터 x를 참조하게 된다.

따라서 이동 트랜잭션의 의미상의 스케줄은 $R_1(x)R_0(y)R_0(x)W_1(x)W_0(y)W_0(x)$가 되며 이동 트랜잭션 T_0과 T_1이 $T_0 \rightarrow T_1 \rightarrow T_0$과 같은 사이클이 형성되어 직렬성에 위배된다.

3.2 하나 이상의 방송주기에 발생 가능한 이동 트랜잭션의 직렬성 위배

(그림 3-2)는 방송 시점인 BC_{i-1}과 BC_i사이에 이동 클라이언트 MH_0과 MH_1의 이동 트랜잭션 T_0과 T_1에서 참조하는 모든 데이터는 캐쉬에 존재한다. 그리고 이동 클라이언트 MH_0과 MH_1은 방송 시점인 BC_{i-1}에서 모든 이동 클라이언트(MH_0, MH_1)의 캐쉬 일관성이 유지되었다고 가정한다. 이동 트랜잭션 T_1의 연산 $R_1(x)BC_{i-1}W_1(x)$를 수행하는 도중에 임의의 이동 트랜잭션 T_0의 연산 $R_0(y)BC_{i-1}R_0(x)W_0(y)W_0(x)$도 완료 요청된다. 그러나, 데이터 x에 대한 두 이동 트랜잭션에서 방송 주기 기간동안 이동 트랜잭션 T_1의 $W_1(x)C_1$에서 데이터 x는 완료되어 데이터베이스에 영구 반영된다.



(그림 3-2) 하나 이상의 방송주기에 이동 트랜잭션의 직렬성 위배

결국 방송 주기 BC_{i-1}과 BC_i사이의 모든 이동 클라이언트에서 참조하는 캐쉬 데이터 x는 무효화된다. 그러나 이동 클라이언트 MH_0의 연산 $R_0(y)BC_{i-1}R_0(x)$에서 무효화된 데이터 x를 접근함으로써 이동 트랜잭션 직렬성에 위배된다. 즉, 이동 트랜잭션의 의미상의 스케줄은 $R_1(x)R_0(y)R_0(x)W_1(x)W_0(y)W_0(x)$가 되며 데이터 x에 대한 이동 트랜잭션 T_0과 T_1이 $T_0 \rightarrow T_1 \rightarrow T_0$과 같은 사이클이 형성되어 직렬성에 위배된다.

4. 적응적 요청 메시지를 이용한 동시성 제어

본 논문에서 새롭게 제안하고자 하는 기법은 이동 컴퓨팅 환경에서 주기적인 무효화 보고서에 각 데이터에 대한 동적인 갱신 패턴 비율을 추가하여 이동 클라이언트에 전달하고 이동 트랜잭션에서 접근하는 데이터에 대한 유효성을 이동 클라이언트에서 판단할 수 있는 적응적 요청 메시지를 이용한 트랜잭션 스케줄링 기법인 UTSM-ARM(An Update Transaction Scheduling Method using An Adaptive Request Message)이다. UTSM-ARM은 이동 트랜잭션의 유효성을 검증하기 위하여 트랜잭션을 우선 수행하고 나중에 유효화 단계를 거치는 낙관적 동시성 제어기법을 기반으로 한다. 낙관적 동시성 제어에서 트랜잭션의 직렬성을 검증하는데 있어서 선방향과 후방향의 두 가지 유효화 과정이 있다[9]. 선방향 유효화는 완료하려는 이동 트랜잭션이 다른 활동 중인 트랜잭션과의 충돌 유무를 확임함으로써 트랜잭션의 직렬성을 보장하는 것이며, 후방향 유효화는 완료하려는 이동 트랜잭션이 최근에 완료된 다른 트랜잭션과의 충돌 유무를 확인함으로써 트랜잭션의 직렬성을 보장하는 것이다. 이동 컴퓨팅 환경에서 각 이동 클라이언트가 현재 실행중인 트랜잭션에 대한 정보를 확인하기 어렵기 때문에 후방향 유효화 기법을 선택하는 것이 더 바람직하다[19]. 제안하는 UTSM-ARM는 후방향 유효화 기법을 갖는 낙관적인 동시성 제어 방법에 기초한다. 그리고 이 기법에서는 완료를 시도하는 이동 트랜잭션을 포함한 모든 트랜잭션의 직렬성을 만족 유무를 이동 트랜잭션에서 접근한 캐쉬 데이터의 타임스탬프와 서버에 유지된 마지막 갱신 트랜잭션에 의한 데이터의 타임스탬프를 비교함으로써 확인 할 수 있다.

그러나 낙관적인 동시성 제어 기법을 기반으로 하는 갱신 트랜잭션 스케줄링 기법[19]은 이동 트랜잭션에서 참조되는 데이터의 무효화를 인식하지 못함으로서 이동 트랜잭션이 직렬성에 위배되고 서버에 불필요한 완료 요청 메시지를 전달하여 제한된 대역폭의 낭비를 초래하게 된다. 제안하고자 하는 USTM-ARM은 각 데이터에 대한 동적인 갱신 패턴 비율을 이용함으로써, OCC-UTS[19]에서 발생 가능한 이동 트랜잭션의 직렬성이 위배되는 문제점을 해결할 수 있다.

4.1 USTM-ARM의 동적인 갱신 패턴 비율 알고리즘

이동 컴퓨팅 환경에서 서버는 캐쉬 일관성을 유지하기 위하여 데이터의 접근횟수나 접근 패턴등의 특징을 반영한 무효화 보고서를 이동 클라이언트에 전달한다. 그러나 다수의 이동 클라이언트를 포함하는 이동 컴퓨팅 환경에서 데이터의 접근횟수나 접근패턴을 포함하는 무효화 보고서는 갱신 트랜잭션을 수행함에 있어서 트랜잭션의 직렬성에 위배되는 경우가 발생하게 된다.

본 논문에서는 캐쉬 일관성 유지를 위하여 무효화 보고서에 각 데이터에 대한 동적인 갱신 패턴을 추가하여 이동 클라이언트에 전달함으로써 OCC-UTS에서 나타나는 문제점을 해결할 수 있다. (그림 4-1)은 USTM-ARM의 무효화 보고서에 갱신 패턴 비율을 적용한 알고리즘이다. 각 데이터에 대한 갱신 패턴은 최근에 방송된 방송횟수(상수 : β)를 기준으로 완료된 갱신 트랜잭션에 의한 동적인 갱신 비율을 나타낸다. 그리고 갱신된 데이터의 갱신 패턴 비율은 다음 방송주기의 무효화 보고서에 추가하여 이동 클라이언트에 전달한다. 이동 응용 서비스에서 갱신 패턴 비율의 적용 기준 상수값(α)은 서비스되는 초기시간(t_0)에서 현재 방

송되고 있는 시점(t_1)까지의 전체 데이터의 갱신 변화율의 평균값을 적용한다.

```
/* 데이터의 갱신 패턴 비율 */
Compute_Update_Pattern_rate(){
  if( CommitRequest Upadte Transaction is Commit ){
     Update Transaction J's U_rate
       = (J's Update_Cnt/β) ;
     Add to J's U_rate In Next IR ;
  }
}

// J's U_rate : 하나의 방송주기 동안 데이터의 갱신 비율.
//β(상수) : 최근의 무효화 보고서 방송 주기 수.
// J's Update_Cnt : 최근 방송주기(상수 : β)
// 동안에 데이터의 갱신 카운트
```

(그림 4-1) USTM-ARM의 동적인 갱신 패턴 비율 알고리즘

이동 응용 서비스에서 하나의 방송 주기 동안에 데이터의 갱신 가능 확률은 0.5이다(갱신될 수도 있고 그렇지 않을 수도 있다.). 그리고 각 데이터에 대한 갱신 트랜잭션이 자주 발생할 수 있는 경우의 갱신가능 확률값(α) > 0.5이다. 예를 들어, 실시간 경매 시스템과 같은 경우 경매 마감 시간에 근접할수록 데이터의 갱신 비율은 증가하게 된다. 즉, 하나의 방송 주기 동안에 데이터의 갱신 가능 비율이 증가하여 트랜잭션의 철회율이 증가하게 되고, 불필요한 트랜잭션의 완료 요청으로 인한 대역폭(uplink)의 사용율이 증가하게 되는 단점이 있다.

4.2 USTM-ARM의 이동 클라이언트 알고리즘

갱신 연산이 가능한 이동 컴퓨팅 환경에서 이동 트랜잭션의 직렬성의 위배되지 않도록 하기 위한 UTSM-ARM의 이동 클라이언트의 알고리즘은 (그림 4-2)와 같다.

```
/* 이동 클라이언트가 무효화 보고서를  수신 받은 경우 */
On receiving the invalidation report IR(ts_j){
   /* 제출된 트랜잭션(Tid)의 완료와 철회 여부 확인 */
   if T_id appears in CommitList { commit T ; }
   if T_id appears in AbortList { abort T ; }
   /* 단절 여부 확인
   // wL : 방송 주기(L : 윈도우방송간격, w : 윈도우횟수 )
   // ts_j : 현재 무효화레포트's TimeStamp
   // ts_lb : 캐쉬에 저장된 그 전에 수신한 방송 TimesTamp */
   if (ts_j - ts_lb > wL){
        drop the entire cache ;
   } else {
      for every item j in the cache {
      /* 캐쉬 데이터중에서 변화된 데이터 점검 */
      if there is a pair (j, t_j) in IR(ts_j){
          if (t^c_j < t_j) {  /* t^c_j : 캐쉬에 저장된 j의 TimeStamp */
             throw j out of the cache ;
             /* 제거된 캐쉬 데이터를 참조한 트랜잭션의 존재
                여부확인 */
             if there is an active Transaction  used cache data
                item j { Abort T ; }
             }
             } else {t^c_j = t_j ; }
      }
      }
   }
   ts_lb = ts_j
}
```

(그림 4-2) USTM-ARM의 이동 클라이언트 알고리즘 1

서버에서 방송된 무효화 보고서를 듣는 이동 클라이언트는 캐쉬 내에 데이터 j의 유효성을 판단하기 위하여 캐쉬 내에 있는 (j, $j^c ts$, *j's value*, *j's U*) (여기서 $j^c ts$는 캐쉬 내의 데이터 j의 타임스탬프, *j's value*는 캐쉬 내의 데이터 j의 값, *j's U*는 캐쉬 내의 데이터 j의 갱신비율) 리스트와 비교하여 캐쉬 데이터의 타임스탬프 보다 작다면 즉, $j^c ts$가 무효화 보고서의 *d's ts*보다 작다면 캐쉬 데이터를 제거하고 방송되는 무효화 보고서의 *d's ts*와 *d's value*로 갱신되어야 한다. 또한 이 무효화된 데이터(d)를 접근한 이동 트랜잭션은 철회된다. 그러나 그렇지 않다면 캐쉬에 있는 데이터 값은 그대로 유지되고 단지 타임스탬프 값만 현재 무효화 보고서의 데이터의 *d's ts*와 *d's value* 값으로 갱신한다. 그리고 이동 클라이언트는 무효화 보고서를 받았던 마지막 시간을 가리키는 ts_{lb}를 현재 방송되는 무효화 보고서의 타임스탬프인 IR_{ts}로 유지한다. ts_{lb}는 이동 클라이언트가 통신 단절로부터 복구된 후 마지막 받은 무효화 보고서의 타임스탬프를 가리킬 수 있도록 유지한다.

if ($j_u \geq \alpha$) : 갱신가능성이 있다고 판단하여 최신의 데이터를 요구한다.
if ($j_u < \alpha$) : 갱신가능성이 없다고 판단하여 캐쉬 데이터의 값을 사용한다.
(j_u : 캐쉬에 존재하는 데이터(j)의 갱신 패턴 비율)

따라서, 각 이동 클라이언트는 통신 단절이 되지 않고 주기적으로 방송되는 무효화 보고서를 듣고서 캐쉬 데이터나 접근된 이동 트랜잭션의 유효성을 확인할 수 있다. 이동 클라이언트에서 읽기 연산을 수행하는 과정은 다음과 같다. 이동 클라이언트는 데이터(d)에 대한 동적인 갱신 패턴 비율(*d's U*)을 추가하여 캐쉬 데이터의 일관성을 유지한다. 다음은 각 데이터에 대한 동적인 갱신 패턴 비율을 적용한 데이터의 읽기 연산을 수행하는 과정을 나타낸 것이다. 이동 클라이언트의 읽기 연산에서 캐쉬에 존재하지 않는 데이터를 접근하는 경우, 이동 클라이언트는 접근하는 데이터에 대한 최신의 타임스탬프와 데이터 값을 서버에 요청하고 요청 받은 데이터를 캐쉬에 반영하여 읽기 연산을 수행한다. 그리고 이동 트랜잭션에서 접근하는 각 데이터 항목들의 타임스탬프 값은 이동 트랜잭션의 시작 시점에 확인하여 완료 요청 때까지 기억한다. 또한, 각 이동 클라이언트의 갱신 트랜잭션을 위하여 다음과 같은 두 가지 집합을 유지한다.

ReadSet(T_{id}) = { ***[j, j^c_{ts}]*** | ***j***는 갱신 트랜잭션 ***T_{id}***에 의해 읽혀진 데이터항목, ***j^c_{ts}***는 캐쉬 데이터 ***j***의 타임스탬프 }
NewValue(T_{id}) = { ***[j, j_v]*** | ***j***는 갱신 트랜잭션 ***T_{id}***에 의해 갱신된 항목, ***j_v***는 데이터 ***j***의 새로운 갱신된 값 }

이동 클라이언트에서 이동 트랜잭션을 완료 요청 메시지와 함께 서버에 전달할 때, 이동 트랜잭션의 구분자 ***T_{id}*** 및 ***ReadSet(T_{id})***, ***NewValue(T_{id})***, 그리고 ***RequestToCommit***를 메시지를 함께 구성하여 서버에 완료 요청한다. 만약, 갱신 트랜잭션이 아닌 읽기 연산으로만 구성된 읽기전용 트랜잭션의 경우 ***NewValue(T_{id})*** 값은 ***Null***값으로 전달된다. 그 후 이동 클라이언트는 다음 방송되는 무효화 보고서를 듣는다. 각 이동 클라이언트에서 이동 트랜잭션에 대한 완료 요청, 읽기 연산, 그리고 갱신 연산에 대한 처리 과정의 알고리즘은 (그림 4-3)과 같다.

그리고 다음 방송주기의 무효화 보고서를 듣고서 완료 요구한 갱신 트랜잭션의 완료 유무를 확인한다. 즉, 이동 클라이언트는 무효화 보고서와 함께 ***CommitList***와 ***Abort List***를 주시한다. 그리고 완료 요청한 ***T_{id}***가 ***CommitList***에 있다면 완료 요청했던 이동 트랜잭션은 정상적으로 완료하여 새로운 데이터(j)의 타임스탬프 값(***$j^c ts$***)과 ***j's value***로 캐쉬 데이터의 일관성을 유지한다. 그러나 완료 요청한 이동 트랜잭션(***T_{id}***)가 ***AbortList***에 존재하면 완료 요청했던 이동 트랜잭션은 철회되고 이동 트랜잭션에서 갱신했던 캐쉬 내 데이터의 신 버전의 값과 타임스템프는 의미가 없으므로 제거된다. 그리고 갱신된 새로운 데이터(***j***)의 타임스탬프 값(***$j^c ts$***)과 ***j's value***로 캐쉬에 반영한다.

```
/* 트랜잭션 완료 요청 메시지를 서버에 전달 */
On receiving the Commit_To_Request{
  send a RequestToCommit message composed of Tid,
      Tranasction Set (Read set, Write set) and NewValue ;
}
/* 읽기 연산 트랜잭션 수행 */
On receiving the Read_Transaction T(j){
  if j is in the cache {
      /* U_rate 비교 기준 값 : α(상수값)
      if( data item j's Urate > α ) {
            go uplink to the server, along with j's id and
            load  new value of j, j's timestamp;
            /* 새로 요구된 데이터에 대한 타임스템프가 기존의
            // 데이터의 타임스템프보다 큰경우 : 이전의 모든
            // 트랜잭션 철회 */
            if (new value j's timestamp > old value j's
               timestamp){
            {
                  abort T ;
                  Update old data item j's timestamp ;
                  Update old data item j's value ;
            }
      } else {
            answer by using the calue in it's cache ;
      }
  } else {
      go uplink to the server, along with j's timestamp and
      load  new item of j, j's timestamp ;
  }
}
/* 갱신 연산 트랜잭션 수행 */
On receiving the Update_Transaction T(j){
      update the value in its cache ;
}
```

(그림 4-3) USTM-ARM의 클라이언트 알고리즘 2

4.3 USTM-ARM의 서버 알고리즘

이동 클라이언트로부터 완료 요청 받은 이동 트랜잭션의 유효성과 직렬성을 검증하기 위하여 서버는 ***RequestTo Commit***메시지들을 받아 큐에 저장한다. 각 ***RequestTo Commit***메시지 m은 ***m.T_{id}, m.ReadSet***과 ***m.NewValues***와 같은 항목 필드로 구성되어 있다. 그리고 서버에서는 ***m.ReadSet***내 존재하는 하나의 데이터 항목이라도 서버의 데이터베이스에 존재하는 동일한 데이터 항목의 타임스탬프 보다 작다면 큐에 존재하는 모든 이동 트랜잭션은 ***AbortList***에 등록되고, 그렇지 않은 경우는 ***CommitList***에 등록된다. 그리고 다음 방송주기에 방송되는 무효화 보고서의 ***AbortList***와 ***CommitList***에 완료 요청된 이동 트랜잭션의 ***m.T_{id}***를 추가한다. 그리고 완료 요청된 이동 트랜잭션이 직렬성과 데이터의 유효성에 위배되지 않는 경우, ***m.ReadSet***과 함께 완료 요청된 ***m.NewValues***가 존재한다면 ***m.NewValues***내의 값들을 데이터베이스에 영구 반영하며, ***m.NewValues*** 내의 모든 데이터 항목들이 ***$ts_{i-1} < j_{ts} < ts_i$*** (***j***는 이동 트랜잭션에 의해 갱신된 데이터)과 같은 타임스탬프 ***j_{ts}***를 갖도록 ***$U(ts_i)$***를 갱신한다. 그리고 갱신 완료된 데이터에 대한 갱신 비율을 동적으로 유지하기 위하여 각 데이터에 갱신 비율을 다시 계산한다. (그림 4-4)는 이동 클라이언트에서 완료 요청된 이동 트랜잭션의 직렬성과 데이터의 유효성을 검증하기 위하여 서버에서 수행되는 알고리즘이다.

```
/*  이동 클라이언트에서 제출된 트랜잭션 처리 */
Dequeue a message m from Que in the interval [ts i-1, ts i]{
    if there exists at least one j in m.ReadSet such that t^c_j < t_j{
      put m.T id in  Abortlist the next IR ;
    } else {
      put m.T id in the next CommitList report ;
      /* commit the transaction in the server */
      install the value in the m.newValue into the database ;
      recompute U(ts i) such that all data items in
      m.Newvalues has the same timestamp t_j and
        ts i-1 < t_j < ts i
      /* 갱신이 발생한 데이터에 대한 갱신 비율을 추가 */
      Recompute_update_rate(j) ;
    }
}
```

(그림 4-4) USTM-ARM의 서버 알고리즘 1

그리고, (그림 4-5)는 서버에서 이동 클라이언트에 방송되는 무효화 보고서와 이동 클라이언트의 즉시 캐싱에 의

하여 데이터를 요구할 경우에 대한 서버 알고리즘의 한 부분이다. 각 방송 주기마다 이동 클라이언트에 전달되는 무효화 보고서는 ***CommitList***와 ***AbortList***를 그리고 갱신된 데이터에 대한 타임스탬프, 데이터 값으로 구성된다. 또한 이동 클라이언트의 읽기 연산에 의한 즉시 캐싱으로 데이터(***data***)에 대한 요구가 있을 경우, 서버는 데이터의 값(***data's value***)과 타임스탬프(***data's TimeStamp***)를 포함하는 메시지를 이동 클라이언트에 즉시 전달된다.

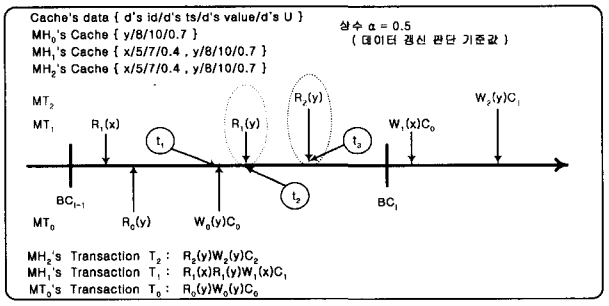
```
/* 무효화 메시지 방송 */
if  it is time to broadcast IR(tsi){
        // IR은 data_id, data's timestamp and data's
           U_rate로 구성
        // data's U_rate : 갱신비율
        piggypacked CommitList and AbortList with IR(ts_i) ;
        broadcast IR(ts_i) ;
}

/* 이동 클라이언트에서 즉시 캐싱, 갱신비율에 따른
   데이터 요청 */
if it is asked for data's Value and Timestamp by a
        Mobile Client{
        send the message containing (data's value, data's
                                TimeStamp) ;
}
```

(그림 4-5) USTM-ARM의 서버 알고리즘 2

4.4 USTM-ARM에 의한 이동 트랜잭션 스케줄링

(그림 4-6)은 이동 트랜잭션 스케줄링 기법인 USTM-ARM의 한 예이다. USTM-ARM 알고리즘을 사용하기 위하여 먼저 제한되어야 할 상수값이 두 가지 존재한다. 하나는 동적인 갱신 비율을 사용하기 위한 상수 β(동적 갱신 패턴 비율을 적용하기 위한 최근의 방송 주기 수)값과 이동 클라이언트에서 각 캐쉬 데이터에 대한 유효성 유무를 판단하기 위한 상수 α(데이터의 유효성 판단 기준)값을 설정된다. 여기에서는 상수 β = 10이라 하고 상수 α = 0.5(이전의 방송 주기 동안에 전체 데이터에 대한 평균 갱신 가능 비율값)라 가정한다. 또한 이동 클라이언트의 캐쉬 크기를 2개의 데이터가 저장될 수 있다고 가정하고 다음 예를 수행하도록 한다. 그리고 이동 클라이언트는 더 이상 데이터를 캐싱 할 수 없을 때, 이동 클라이언트의 캐쉬 데이터는 LRU(Least Recently Used) 기법을 이용한 캐쉬 대체 정책에 의하여 관리된다.

방송 시각 BC_{i-1}에서 모든 이동 클라이언트는 캐쉬 데이터의 일관성을 유지하고 있다. 그리고 각 이동 클라이언트의 캐쉬 데이터는 x(타임스탬프 : 5, 값 : 7, 동적 갱신비율 : 0.4), y(타임스탬프 : 8, 값 : 10, 동적 갱신비율 : 0.7)이며, 각 이동 클라이언트는 MH_0, MH_1, 그리고 MH_2의 3개가 존재한다고 가정하자. 이때, 데이터 y에 대한 동적 갱신 패턴 비율(y_U : 0.7) > α (0.5)이므로, 이동 클라이언트는 데이터 y를 무효화되었다고 판단하고 서버에 데이터 y의 최근 값(***y's ts, y's value***)을 요청하여 전달받는다. 그리고 전달받은 데이터 y를 접근하여 이동 트랜잭션의 연산을 수행한다. 또한 데이터 x에 대한 동적 갱신 패턴 비율(x_U : 0.4) < α (0.5)이므로, 이동 클라이언트는 캐싱된 데이터 y를 유효하다고 판단하여 서버에 새로운 데이터를 요청하지 않는다. 이동 클라이언트(MH_0)의 이동 트랜잭션 이력은 $R_0(y)W_0(y)C_0$이고 시간 t_1에서 이동 트랜잭션에 대한 완료 요청 메시지를 서버에 전달한다. 그리고 완료 요청된 이동 트랜잭션(T_0)은 ***commit*** 되어 데이터 y는 데이터베이스에 영구 반영되고 다음 방송주기(BC_i)의 무효화 보고서에 데이터(y)의 갱신된 타임스탬프(y_{ts}), 데이터값, 데이터(y)의 재 계산된 갱신 비율값, 그리고 ***CommitList***에 이동 트랜잭션 식별자(T_0)을 추가한다. 그리고 이동 클라이언트 MH_1의 이동 트랜잭션(T_1)의 이력 $R_1(x)R_1(y)BC_iW_1(y)C_1$을 서버에 완료 요청한다. 이동 트랜잭션($T_1$)의 연산 $R_1(x)$에서 캐쉬 데이터 x는 유효하다고 판단하고 연산을 수행한다. 시간 t_2에서 이동 트랜잭션(T_1)의 연산 $R_1(y)$를 수행할 경우, 캐쉬 데이터 y는 시간 t_1에서의 이동 트랜잭션(T_0)에 의해 갱신되어 다음 방송주기(BC_i)때까지 갱신되었음을 확인하지 못한다. 그러나 데이터(y)의 갱신 패턴 비율을 적용하여 캐싱된 데이터(y)를 무효화 되었다고 판단하고 서버에 데이터 y에 대한 적응적 요청 메시지를 전달하고 데이터 y의 최신 타임스탬프, 데이터 값을 받고 연산을 수행한다. 그리고 다음 방송주기(BC_i)의 무효화 보고서를 듣고 현재 수행중인 이동 트랜잭션(T_1)에서 접근한 데이터 x, y는 무효화 보고서의 데이터와 같은 타임스탬프를 갖고 있다. 따라서 이동 트랜잭션(T_1)은 철회되지 않고 정상적인 연산 $W_1(x)$을 수행하고 서버에 완료 요청 메시지를 전달한다. 그리고 완료 요청 메시지를 받은 서버는 이동 트랜잭션(T_1)에서 접근한 데이터 x, y의 타임스탬프를 비교하여 정상적으로 완료된다.



(그림 4-6) USTM-ARM에 의한 이동 트랜잭션 스케줄링

또한 이동 클라이언트(MH_2)의 이동 트랜잭션(T_2)의 이력은 $R_2(y)W_2(y)BC_iC_2$이다. 그러나, 이동 트랜잭션(T_1)의 연산과 같이 시간 t_3에서의 연산 $R_2(y)$에서 참조하는 데이터 y는 무효화되었다고 판단하고 적응적 요청 메시지를 이용하여 서버에 데이터 y에 대한 타임스탬프, 데이터값을 요구받아 연산을 수행한다. 그리고 다음 방송주기(BC_i)에 데

이터 y에 대한 타임스템프와 무효화보고서의 타임스템프와 같기 때문에 이동 트랜잭션(T_2)는 연산 $W_2(y)$를 정상적으로 수행하고 서버에 완료 요청 메시지를 전달하여 이동 트랜잭션(T_2)을 완료할 수 있다. 다음 <표 1>은 이동 컴퓨팅 환경에서 제안된 기법들에 대한 트랜잭션 스케줄링 기법의 각 특징들을 도표화 한 것이다.

〈표 1〉 낙관적인 트랜잭션 스케줄링 기법의 비교

	캐 쉬 사 용	캐 쉬 일관성 기술	Client 연산	방 송 기 술	스케줄링	기 술
OCC-UTS[19]	사 용	방 송	Update, Read	무효화 보고서	Optimistic Timestamp	Distributed Fashion in client
OCC/2VTS[21]	사 용	방 송	Read Only	무효화 보고서	Optimistic Timestamp	the 2 version timestamp of data
OCC/UTF[22]	사 용	방 송	Update, Read	무효화 보고서	Optimistic Timestamp	efficiently use Updated cache data
UTSM-ARM	사 용	방 송	Update, Read	무효화 보고서	Optimistic Timestamp	The Adaptive Request Messages

5. 모의 실험 및 성능 평가

5.1 모의 실험 매개 변수

제안하는 UTSM-ARM의 모의 실험을 통한 성능을 분석하기 위하여 전체 데이터를 포함하는 데이터베이스가 존재하고, 각 이동 클라이언트에 존재하는 캐쉬의 크기, 서버에 완료 요청되는 이동 트랜잭션에 대한 최대 수용 트랜잭션 수(큐의 크기), 이동 클라이언트에 방송되는 무효화 보고서의 방송 주기 그리고 한 방송 주기 동안에 무효화 보고서를 전달하는 윈도우 크기를 <표 2>와 같이 결정하였다. 그리고 이동 클라이언트에서 즉시 캐쉬 발생 비율을 20%로 하였다. 또한 제안하는 UTSM-ARM의 적응적 요청 메시지를 이용하기 위하여 방송되는 최근의 방송 횟수(β : 상수)와 각 캐쉬 데이터의 동적인 갱신 패턴 비율 기준값 (α : 상수)등을 설정하였다.

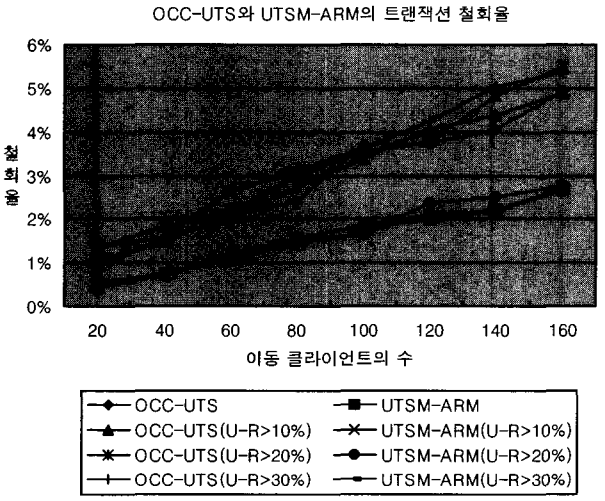
〈표 2〉 모의실험을 위한 매개변수

Parameter	Expressions /Value	Description
DataBase_Set	1000	데이터베이스 크기
Cache_Size	10	이동 클라이언트의 캐쉬 크기
Max_Transaction_Cnt	500	최대 수행 가능한 트랜잭션 수
window_size	2	한 방송주기 동안 무효화 보고서 방송 횟수
IR_interval	10	방송 주기
Imm_Caching_Rate	0.2f	즉시 캐쉬 발생 비율
BroadCasting_Cnt_Level	β/10	데이터의 갱신 패턴 비율 대상이 되는 방송 주기 수
Update_Rate_Level	α/0.5f	데이터의 갱신 패턴 비율과 비교되는 기준 값

5.2 UTSM-ARM 성능 분석

5.2.1 UTSM-ARM의 이동 트랜잭션 철회율

이동 컴퓨팅 환경에서 데이터의 상호 일관성을 유지하기 위하여 서버는 주기적인 방송으로 이동 클라이언트에 무효화 보고서를 전달한다. 이동 응용시스템에서 데이터의 갱신 가능성 비율이 증가할 경우, 캐쉬 데이터가 무효화될 가능성은 증가하게 된다. 따라서 무효화된 캐쉬 데이터를 참조한 이동 트랜잭션의 철회될 수 있는 가능성 또한 증가할 것이다. 기존에 제안된 OCC-UTS은 방송기술을 사용하고 이동 클라이언트에서 갱신 연산이 가능한 경우, 우수한 수행성능을 보였다[19].

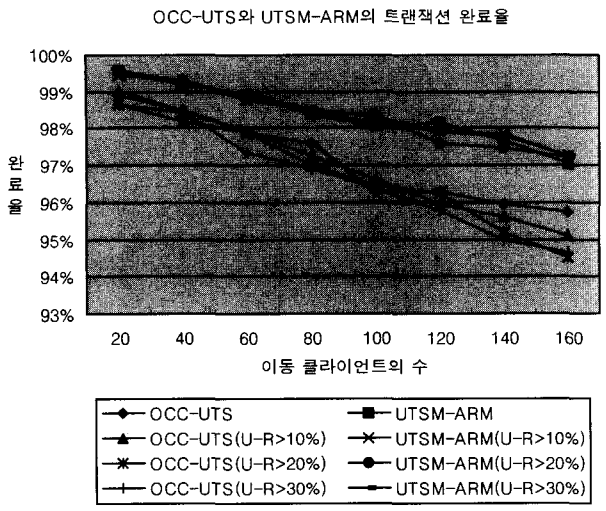


(그림 5-1) 이동 클라이언트 수에 따른 이동 트랜잭션의 철회율

(그림 5-1) 이동 클라이언트 수에 따른 이동 트랜잭션의 이동 클라이언트에서 갱신 연산의 수행이 가능하며 방송기법을 이용한 캐쉬 일관성을 유지하고, 낙관적인 타임스템프 스케줄링 기법을 이용한 동시성 제어 방법이다. (그림 5-1)은 이동 클라이언트 수에 따른 이동 트랜잭션 철회율의 성능 비교 결과이다. 이동 클라이언트의 수에 따른 이동 트랜잭션의 성능 결과는 UTSM-ARM이 OCC-UTS보다 철회율이 더 낮음을 보여준다. 그리고 데이터의 갱신연산(U : Update Transaction)과 읽기연산(R : Read Only Transaction)의 비율의 차이가 커질수록 이동 클라이언트의 수에 따른 이동 트랜잭션의 철회율이 증가함을 알수 있다.

5.2.2 UTSM-ARM의 이동 트랜잭션 완료율

(그림 5-2)는 이동 클라이언트 수에 따른 OCC-UTS과 UTSM-ARM의 이동 트랜잭션 완료율을 표시한 것으로 두 알고리즘에 대한 트랜잭션의 완료율에 대한 차이를 알아볼 수 있다.

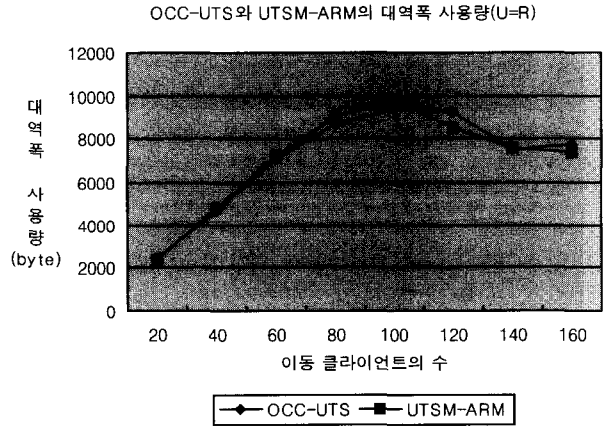


(그림 5-2) 이동 클라이언트 수에 따른 이동 트랜잭션의 완료율

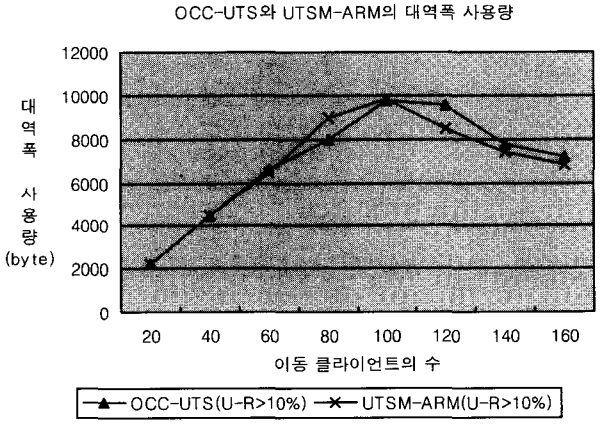
제안 방법인 UTSM-ARM이 OCC-UTS보다 이동 클라이언트의 수에 따른 이동 트랜잭션 완료율이 더 높은 것을 알 수 있다. 그리고 이동 클라이언트에서 수행가능한 갱신 연산(U : Update Transaction)과 읽기연산(R : Read Only Transaction)의 비율의 차이가 커질수록, 그리고 이동 클라이언트의 수가 증가 할수록 이동 트랜잭션 완료율에서 더 좋은 성능을 보임을 확인할 수 있다.

5.2.3 UTSM-ARM의 통신 메시지량

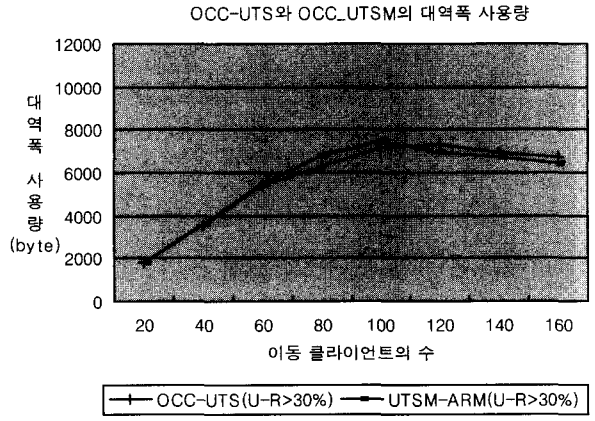
방송을 통한 캐쉬 일관성 유지 방법은 서버에서 이동 클라이언트에 전달하는 대역폭(downlink)과 이동 클라이언트에서 서버에 요청하는 대역폭(uplink)은 비대칭적이다. 즉 이동 클라이언트에서 서버에 요청하는 대역폭은 상대적으로 제한되어 있다. 새롭게 제안하는 UTSM-ARM은 서버에 완료 요청과 함께 적응적 요청 메시지를 사용하여 서버에 캐쉬 데이터의 유효성을 확인하는 과정이 추가되었다. 각 캐쉬 데이터의 갱신 비율에 따라서 서버에 전달되기 때문에 이동 클라이언트에서 수행하는 모든 트랜잭션에 대한 적응적 요청 메시지를 서버에 전달하는 것이 아니라 각 데이터에 대한 갱신비율이 갱신비율 기준값(이전의 방송 주기 동안 전체 데이터의 평균 갱신 가능 비율값)보다 클 경우만 서버에 데이터의 유효성 확인 과정이 이루어지기 때



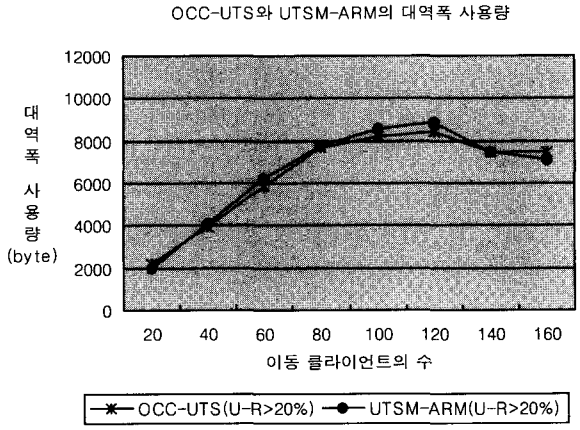
(그림 5-3) 이동 클라이언트수에 따른 메시지(Uplink)량 (U=R)



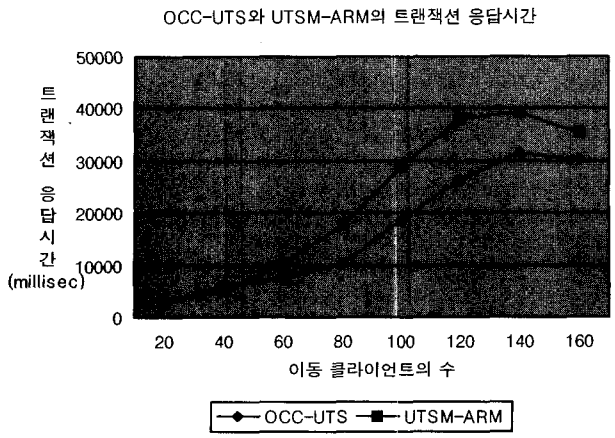
(그림 5-4) 이동 클라이언트수에 따른 메시지(Uplink)량 (U-R=10%)



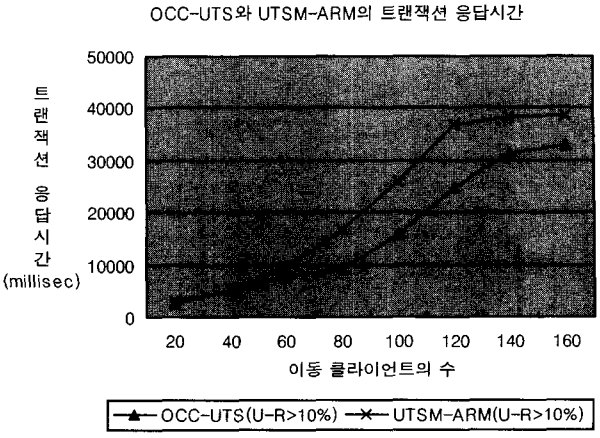
(그림 5-5) 이동 클라이언트수에 따른 메시지(Uplink)량 (U-R=20%)



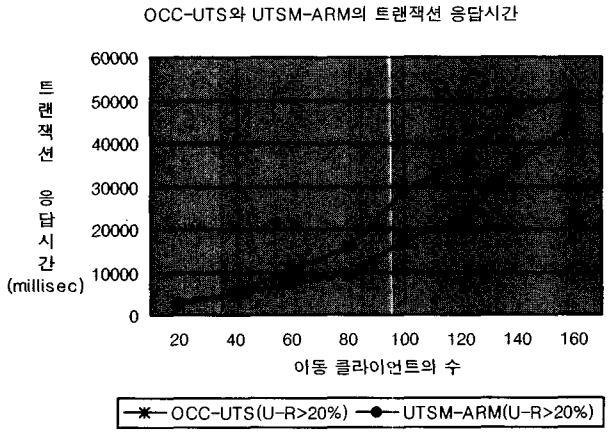
(그림 5-6) 이동 클라이언트수에 따른 메시지(Uplink)량 (U-R=30%)



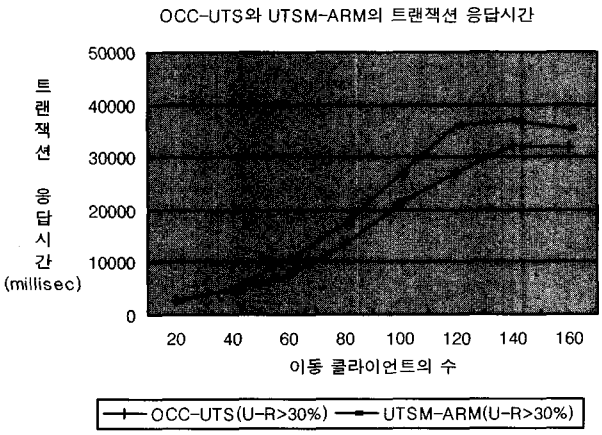
(그림 5-7) 이동 클라이언트수에 따른 트랜잭션 응답시간 (U=R)



(그림 5-8) 이동 클라이언트수에 따른 트랜잭션 응답시간 (U-R=10%)



(그림 5-9) 이동 클라이언트수에 따른 트랜잭션 응답시간 (U-R = 20%)



(그림 5-10) 이동 클라이언트수에 따른 트랜잭션 응답시간 (U-R = 30%)

문에 UTSM-ARM의 적응적 요청 메시지에서 사용되는 메시지의 양은 아주 적다. 또한 데이터의 유효성 확인과정이 이루어 질 때, 이동 클라이언트에서 서버에 요청되는 메시지는 데이터의 식별자만을 전달함으로써 전송되는 데이터 양을 최소화 시켰다. (그림 5-3), (그림 5-4), (그림 5-5), (그림 5-6)은 이동 클라이언트에서 수행 가능한 갱신연산(U : Update Transaction)과 읽기연산(R : Read Only Transaction)의 비율의 차이에 따라 제한된 대역폭(uplink)의 사용량을 비교한 결과이다.

위의 결과에서 볼수 있듯이 본 논문에서 제안하는 UTSM-ARM과 OCC-UTS에서의 이동 클라이언트에서 서버에 전달되는 대역폭(uplink)의 사용량은 큰 차이를 보이지 않는다.

5.2.4 UTSM-ARM의 트랜잭션 응답시간

본 논문에서 제시하는 UTSM-ARM과 OCC-UTS의 가장 크게 구별되는 특징은 적응적 요청 메시지를 이용하는 것이다. 적응적 요청 메시지는 최근의 방송 횟수(β : 상수)와 각 캐쉬 데이터의 동적인 갱신 패턴 비율 기준값(α : 상수)에 의해서 캐쉬 데이터의 최신성 및 일관성을 유지하는 기술이다. 갱신 패턴 비율의 기준값은 방송 횟수라는 과거의 데이터 정보값을 이용한다. 따라서, 서버에서는 하나의 데이터에 대한 갱신이 발생하게 되면 갱신된 데이터에 대한 갱신 비율값을 계산해야 한다.

(그림 5-7), (그림 5-8), (그림 5-9), (그림 5-10)은 이동 클라이언트에서 수행 가능한 갱신연산(U : Update Transaction)과 읽기연산(R : Read Only Transaction)의 비율의 차이에 따라 전체 이동 트랜잭션의 평균 응답 시간을 계산한 결과이다. 전체적으로 UTSM-ARM이 OCC-UTS보다 평균 응답시간이 길다. 이것은 제안하는 UTSM-ARM의 각 데이터에 대한 갱신 비율을 계산하는 과정이 포함되어 있기 때문으로 분석된다. 그러나 방송기술을 사용하는 이동 컴퓨팅 환경에서 각 데이터에 대한 갱신 비율을 고려한 UTSM-ARM이 OCC-UTS보다 우수한 수행 성능을 보임을 알 수 있다.

6. 결론 및 향후 연구 과제

본 논문에서는 비대칭적인 대역폭의 특징을 갖는 이동

컴퓨팅 환경에서 갱신 트랜잭션이 가능한 경우 이동 트랜잭션에서 접근하는 데이터에 대한 유효성을 데이터의 동적인 갱신 패턴을 적용하여 데이터의 유효성을 서버에 요청함으로써 캐쉬 데이터의 최신성 및 일관성을 유지하고 이동 트랜잭션에서 접근하는 데이터에 대한 유효성을 이동 클라이언트에 분산 처리함으로써 서버의 부하를 줄이고 트랜잭션 수행성능을 증가시켰다. 수행 성능 분석의 기준은 이동 클라이언트의 수에 따른 이동 트랜잭션의 완료율, 철회율, 이동 클라이언트에서 서버와의 통신 메지시량, 그리고 이동 트랜잭션의 응답시간을 비교 분석하였다. 제안하는 UTSM-ARM은 OCC-UTS보다 이동 클라이언트의 수에 따른 이동 트랜잭션의 완료율과 철회율은 월등히 우수한 수행성능을 보였다. 또한 본 논문에서 제안하는 UTSM-ARM은 OCC-UTS보다 제한된 대역폭 사용에 있어서 차이를 나타나지 않았다.

향후 본 이동 트랜잭션의 응답시간을 단축시킬 수 있는 방법, 이동 클라이언트에서 이동 트랜잭션에 대한 사이클을 탐지할 수 있는 방법과 읽기전용 트랜잭션의 비율이 높은 경우, 트랜잭션 완료율을 증가시킬 수 있는 이동 트랜잭션 스케줄링 방법 그리고 이동 클라이언트와 서버와의 메시지량을 최적화할 수 있는 방법에 대한 연구가 더욱 필요할 것이다.

참 고 문 헌

[1] S. Acharya, M. Franklin and S. Zdonik, "Disseminating Updates on Broadcasting Disks," Proceedings of the 22nd VLDB Conference, Mumbai, India, 1996.

[2] S. Acharya, M. Franklin and S. Zdonik, "Balancing Push and Pull for Data Broadcasting," Proceedings of ACM SIGMOD Conference on Management of Data, May, 1997.

[3] D. Barbara and T. Imielinski, "Sleepers and Workaholics : Caching Strategies in Mobile Environments," Proc. ACM SIGMOD, June, 1994.

[4] D. Barbara and T. Iminelinsky, "Sleepers and Workholics : Caching in Mobile Environment," Proceedings of ACM SIGMOD Conference on Management of Data ENgineering, pp.114-123, April, 1997.

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

[6] G. H. Forman and J. Zahorjan, "The Challenges of Mobile Computing," IEEE Computers, 27(6), pp.38-47, April, 1994.

[7] M. Franklin and S. Zdonik, "Dada In your Face : Push Technology in Prospective," in Proceedings of 1998 ACM SIGMOD Conference, Seattle, 1998.

[8] S. Hameed and N. H. Vaidya, "Efficient Algorithms for Scheduling Single and Multiple Channel Data Broadcast," Technical Report 97-002, Department of Computer Science, Texas, A&M University, Feb., 1997.

[9] T. Harder, "Observations on Optimistic Concurrency Control Schemes," Information Systems, Vol.9, No.2, pp.111-120, 1984.

[10] T. Imieklin and B. R. Badrinath, "Mobile Wireless Computing : Challenges in Data Management," Communications of the ACM, Vol.37, No.10, Oct., 1994.

[11] J. jing, O. Bukhres, A. Elmagarmid, "Distributed Lock Management for Mobile Transactions," Technical Report CSD-TR-94-073, Department of Computer Science, Puredue University, 1994.

[12] J. Jing, O. Bukhres, A Elmagarmid and R. Alonso, "Bit-Sequences : An Adaptive Cache Invalidation Method in Mobile Client/Server Environment," ACM/Baltzer Mobile Networks and Applications, Vol.2, No.2, 1997.

[13] E. Pitoura et al., "Revising Transaction Concepts for Mobile Computing," Proc. IEEE Workshop on Mobile Systems and Applications, Dec., 1994.

[14] E. Pitoura and B. Bhargava, "Maintaining Consistency of Data in Mobile Distributed Environments," Proc. of 15th International Conference on Distributed Computing System (ICDC'95), May, 1995.

[15] E. Pitoura, "Supporting Read-Only Transaction in Wireless Broadcasting," Proceedings of the 9th International Workshop on Database and Export Systems Applications, pp. 428-433, 1998.

[16] E. Pitoura, and P. Chrysanthis, "Scalable Processing of Read-Only Transactions in Broadcasting Push," International Conference on Distributed Computing Systems, Austin, 1999.

[17] K. Wu et al., "Energy-Efficient Caching for Wireless Mobile Computing," Proc. IEEE International Conference. on Data Eng. pp.336-343. 1996.

[18] S. Lee, M. Kitsuregawa, and C.-S. Hwang, "Efficient processing of wireless read-only transactions in data broadcast," In Proceedings of the 12th International Workshop on Research Issues on Data Engineering, pp.101-111, 2002.

[19] 이상근, 황종선, 이원규, 유헌창, "이동 클라이언트/서버 컴퓨팅 환경에서의 캐싱 및 동시성 제어", 한국정보과학회 논문지, 제26권 제8호, pp.974-987, 1999.

[20] 이상근, 김성석, 황종선, "무선 데이터 방송 환경에서 읽기-전용 트랜잭션 처리 기법", 한국정보과학회논문지, 제29권 제5호, pp.404-415, 2002.

[21] 이욱현, 황부현, "방송환경에서 이중버전과 타임스탬프에 기반을 둔 낙관적 동시성 제어 기법", 정보처리학회논문지D, 제8-D권 제2호, pp.132-144, 2001.

[22] 이욱현, 황부현,"방송환경에서 갱신 거래 우선 낙관적 동시성 제어 기법", 정보처리학회논문지D, Vol.9-D, No.2, pp. 185-194, 2002.

박 준

e-mail : kingrion@empal.com

2001년 동신대학교 컴퓨터학과(이학사)

2003년 전남대학교 대학원 전산학과 (이학석사)

2004년~현재 전남대학교 대학원 전산학과 박사과정

관심분야 : 이동 컴퓨팅, 객체지향시스템, 데이터베이스 등

채 덕 진

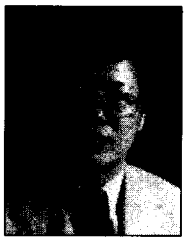
e-mail : djchai@sunny.chonnam.ac.kr

1999년 동신대학교 컴퓨터학과(이학사)

2001년 전남대학교 대학원 전산통계학과 (이학석사)

2001년~현재 전남대학교 대학원 전산학과 박사과정

관심분야 : 데이터마이닝, Bioinformatics 등



황 부 현

e-mail : bhhwang@chonnam.chonnam.ac.kr

1978년 숭실대학교 전산학과(이학사)

1980년 한국과학기술원 전산학과(이학석사)

1994년 한국과학기술원 전산학과(공학박사)

1980년~현재 전남대학교 컴퓨터정보학부 교수

관심분야 : 분산시스템, 분산데이타베이스, 객체지향시스템, 전자상거래

김 중 배

e-mail : jjkim@etri.re.kr

1986년 고려대학교 공과대학 산업공학과 (공학사)

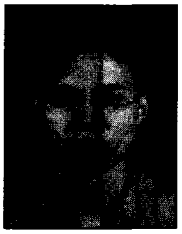
1988년 한국과학기술원 산업공학과 (공학석사)

1998년~현재 한국과학기술원 산업공학과 박사과정

1988년~1991년 대한항공(주) 시스템부

1991년~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 모바일응용서버연구팀장

관심분야 : 미들웨어, 시스템 소프트웨어 등



정 승 욱

e-mail : swjung@etri.re.kr

1996년 전남대학교 전산학과(학사)

1998년 광주과학기술원 정보통신공학과 (석사)

1998년~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 모바일응용서버연구팀 연구원

관심분야 : 웹 응용 서버, 미들웨어, 시스템 소프트웨어 등