방송환경에서 타임스탬프 구간에 기반을 둔 낙관적 동시성 제어 기법

(Optimistic Concurrency Control based on TimeStamp Intervals for Broadcast Environment: OCC/TI)

이 욱 현 † 황 부 현 ††
(Uk-hyun Lee) (Bu-hyun Hwang)

요 약 방송환경은 서버와 클라이언트간 대역폭이 서버에서 클라이언트쪽으로는 크고 클라이언트에서 서버쪽으로의 대역폭은 상대적으로 많이 작은 비대칭적(asymmetric) 특수한 환경이다. 또한 대부분의 방송환경 응용 시스템들은 클라이언트측에서 발생한 주식 데이타, 교통 정보와 새로운 뉴스와 같은 여러 가지 다양한 정보를 검색하는 주로 읽기전용 즉 질의 거래들을 허락한다. 그러나, 기존의 여러 가지 동시성 제어 기법들은 이러한 특수성을 고려하지 않음으로써 방송 환경에 적용될 때 거래들의 불필요한 철회를 일으킨다. 이 논문에서는 방송환경에서 타임스탬프 구간에 기반을 둔 낙관적 동시성 제어 기법을 제안한다.

이 기법은 서버에 의해 관리 유지되고 클라이언트에 의해 읽혀지는 데이타의 상호 일관성과 데이타의 현재성을 만족시키기에 적절한 정확성 검증 기준인 약한 일관성(weak consistency)을 채택하였다. 또한, 그것을 효율적으로 실행할 수 있는 타임스탬프 구간 기법을 적용하였다. 그 결과, 전역적 직렬화를 적용할 때 발생되는 질의 거래의 불필요한 철회 및 재시작의 횟수를 줄임으로써 성능향상을 도모하였다.

키워드 : 방송환경, 이동 클라이언트/서버, 비대칭적 대역폭, 질의 거래, 낙관적 동시성 제어, 타임스탬프 구간 기법, 약한 일관성

Abstract The broadcast environment has asymmetric communication aspect that is typically much greater communication bandwidth available from server to clients than in the opposite direction. In addition, mobile computing systems generate mostly read-only transactions from mobile clients for retrieving different types of information such as stock data, traffic information and news updates. Since previous concurrency control protocols, however, do not consider such a particular characteristics, the performance degradation occurs when previous schemes are applied to the broadcast environment. In this paper, we propose optimistic concurrency control based on timestamp interval for broadcast environment. The following requirements are satisfied by adapting weak consistency that is the appropriate correctness criterion of read-only transactions: (1) the mutual consistency of data maintained by the server and read by clients (2) the currency of data read by clients. We also adopt the timestamp Interval protocol to check the weak consistency efficiently. As a result, we improved a performance by reducing unnecessary aborts and restarts of read-only transactions caused when global serializability was adopted.

Key words : broadcast environment, mobile client/server, asymmetric bandwidth, read-only transactions, optimistic concurrency control, timestamp interval method, weak consistency

· 본 논문은 한국과학재단 2000년도 목적기초연구(R02-2000-00401)비에 의하여 연구되었음.
† 정 회 원 : 전남대학교 전산학과
uhlee@inha.ac.kr
†† 종신회원 : 전남대학교 전자계산학과 교수, 전남대학교 정보통신연구소
bhhwang@chonnam.chonnam.ac.kr
논문접수 : 2001년 7월 9일
심사완료 : 2002년 9월 6일

1. 서 론

많은 데이타베이스 응용분야에서 특히, 동시에 수행하는 엄청나게 많은 수의 클라이언트들을 가진 응용에서 데이타 전달을 위해 방송 기술이 요구되어진다. 예를 들어, 경매와 같은 전자상거래는 단지 소수에게 낙찰될지

라도 수 만 명의 사용자들이 참여한다. 낙찰이 이루어질 때의 신청 가격 갱신 데이타는 신속하고 일관성 있게 전파되어져야 한다. 다행히 데이타베이스의 상대적으로 작은 부분 즉, 경매의 현재 상황인 최고 가격 및 신청인 수와 같은 데이타만이 방송을 요구한다. 그러나, 클라이언트가 서버와 통신하기 위해 필요한 통신 대역폭이 상당히 제한되어 있다. 그러므로, 갱신에 대해 통신하도록 허락되어지는 시간에 클라이언트가 작은 대역폭의 무선망을 사용하여 경매 신청 가격을 보내기 위해 방송 매체를 사용하는 것이 일반적이다. 즉, 방송환경은 서버와 클라이언트간 대역폭이 서버에서 클라이언트쪽으로는 크고 클라이언트에서 서버쪽으로의 대역폭은 상대적으로 많이 작은 비대칭적(asymmetric)인 특수한 환경이다.[1,2,3,4,5,6,7]

무선환경 및 이동 컴퓨팅 환경을 위한 여러 가지 동시성 제어 기법들이 국내외에서 제안되었다. 이전 연구 노력의 대부분이 주로 이동 컴퓨팅 환경은 통신 비용이 비싸고 통신자체가 안전하지 못하다는 특성을 갖는다는 데 기반을 두어 방송환경의 특수성을 제대로 고려하지 않았다. 클라이언트가 자체에서 발생한 거래를 제어 완료하지 못하고 서버에게 상당히 많이 의존하는 형태라면 클라이언트측에서 서버측으로 정보 요구가 자주 생기게 된다. 이것은 각 클라이언트에게 주어진 짧은 시간 안에 상대적으로 작은 대역폭을 가지고 윗방향(uplink) 통신을 해야하는 방송환경에서는 거래 실행이 지연되어 전체 성능면에서 단위 시간당 처리율이 떨어진다. 그러므로 비대칭적 방송환경 특수성을 감안하여 클라이언트가 서버로의 윗방향 정보 요구는 가능하면 줄이는 거래 동시성 제어 기법이 필요하다.

또한 기존의 제안된 기법들은 대부분 질의 거래에 대해서는 특별히 고려하지 않았다. 대부분의 이동 컴퓨팅 시스템에서의 거래들은 클라이언트측에서 발생되는 주로 읽기전용(read_only) 거래들이다. 이러한 거래들은 주식 정보, 교통 정보와 새로운 뉴스 등 여러 가지 다양한 종류의 정보를 요구 검색하는 것들이다. 읽기전용 거래는 읽기 연산으로만 이루어진 것으로 이후 질의(query) 거래라 칭한다. 실제적으로 날씨 예보와 교통 정보 시스템과 같은 대부분의 많은 방송 시스템에서 갱신 거래가 질의 거래보다 발생하는 확률이 훨씬 낮다. 이러한 실제 상황을 고려하더라도 질의 거래를 갱신 거래 방해 없이 즉, 질의 거래와 갱신 거래의 충돌연산으로 인한 철회 없이 질의 거래를 최대한 많이 실행 완료시키는 동시성 제어 기법이 연구되어야 한다.

마지막으로 기존의 제안된 기법들은 정확성 검증 기준으로써 직렬화(serializability)를 사용하는데 그것은 질의 거래가 많은 방송환경에서 매우 비용이 많이 들고 제한적이며 불필요하다[7]. 직렬화[8]는 데이타베이스시스템에서 거래를 위한 가장 공통적인 정확성 검증 기준이다. 그러나, 이것은 본질적으로 전역적(global)인 성질을 가지고 있다. 즉, 여러 클라이언트에서 동시 수행중인 모든 거래들로 인한 영향이 그 거래들이 직렬 순서로 수행된 결과와 같아야 한다. 그 결과 다음과 같은 상황이 발생한다. 첫째 로크와 같은 것을 얻기 위해 분산된 개체들 사이에 과도한 통신이 요구된다. 둘째 그것을 기반으로 한 기법은 극도로 보수적인 성격을 요구하므로 그것과는 다른 관점에서 볼 때 올바르다고 판단되는 실행이 허락되지 않을 수 있다. 위의 첫째 요인으로 인하여 클라이언트가 서버와 통신할 수 있는 대역폭이 제한적인 방송환경에서는 통신비용이 매우 비싸진다. 또한, 둘째 요인으로 인해 불필요한 질의 거래 철회를 양산하여 바람직하지 않은 상태가 초래된다. 그러므로 이러한 문제점을 보완할 수 있는 정확성 검증 기준 채택 및 그것을 기준으로 하는 효율적인 동시성 제어 기법이 연구되어야 한다.

본 논문의 목적은 다음과 같다. 이 논문에서 우리는 방송환경의 특수성을 극복하고 성능 향상에 기여할 수 있는 효율적인 동시성 제어 기법을 제안하고 평가한다. 우리의 기법은 (1) 상호 일관성과 (2) 현재성이라는 두 가지 특성을 만족한다. 상호 일관성은 (a) 서버가 상호적으로 일치하는 데이타를 유지하고 (b) 클라이언트들은 상호적으로 일치하는 데이타를 읽을 수 있다. 현재성은 클라이언트들이 항상 방송 주기 시작 시점의 최근 데이타를 보는 것을 보장한다.

우리는 실제 방송 시스템의 거래 형태를 고려하여 방송환경에서의 질의 거래 관리를 위해 약한 일관성(weak consistency)[9]이라 불리는 정확성 검증 기준을 적용한다. 그러므로써, 상호 일관성을 보장하고 질의 거래의 완료율을 최대한 높여 거래 처리율을 향상시키는 방법을 고안하고자 한다. 그에 대한 타당성은 3장에서 언급한다. 제안한 기법은 이동 클라이언트에서 수행중인 질의 거래가 항상 서버와 접촉 없이 가장 최근의 그리고 일관성을 만족하는 데이타를 읽도록 허락한다.

또한, 본 논문에서는 기존의 이동 컴퓨팅 환경에서 제안된 동시성 제어 기법과 다르게 클라이언트/서버 간 분산환경의 본연의 취지를 살리고자 주요 역할을 클라이언트에게 최대한 전담시키는 클라이언트 역할 최대화에 기본 철학을 둔다. 클라이언트측 활용은 방송환경과 기존의 분산환경을 연관시키는데 필요한 개념 설정을

쉽게 해 주며, 방송환경의 특수성을 감안할 때 비대칭적 대역폭을 최대한 활용할 수 있는 방법을 제공해 준다.

그러므로, 본 논문에서는 비대칭적 방송환경을 고려하여 데이타 일관성을 효율적으로 성취할 수 있으며 거래 동시성과 처리율을 최대한 향상시키는 동시성 제어 기법을 연구 제안하고자 한다.

본 논문의 구성은 다음과 같다. 제2장에서는 방송환경에서 제안된 기존의 논문들을 살펴본다. 또한, 성능 향상을 위해 제안된 기존의 타임스탬프 구간 기법인 BCC-TI를 자세히 소개하고 방송환경 본연의 특성을 살려 적용할 때 나타나는 문제점에 대한 사례를 제시한다. 제3장에서는 방송환경의 특수성을 최대한 극복하고 동시성을 향상시킨 타임스탬프 구간에 기반을 둔 새로운 낙관적 동시성 제어 기법을 제안한다. 제4장에서는 본 논문에서 제안한 동시성 제어 기법의 정확성 검증을 하고 제5장 성능평가에서는 제안한 새로운 기법과 기존의 기법을 모의 실험을 통해 성능을 비교 분석한다. 결론과 향후연구는 제6장에 나타나 있다.

2. 관련연구

방송환경에서 데이타 전달 방법에 관한 많은 연구가 국내외에서 있었다. 푸쉬(push) 기법과 풀(pull) 기법 그리고 이 두 가지를 혼합한 기법에 대한 성능 비교 분석이 이루어졌다.[1] 그리고 방송 디스크(broadcast disk) 즉, 무선 채널을 통해 방송되는 데이타 구조에 관한 많은 연구가 있었다. 데이타의 특성에 따라 다른 속도를 가지고 주기적으로 회전하는 방송 디스크를 제안하고 그의 성능 분석을 하였다.[11] 쓰기 연산이 이루어진 데이타를 방송 디스크를 통해 클라이언트들에게 갱신 결과를 알리는 여러 가지 방법, 즉 유효화(validation), 자동선취(auto-prefetch), 전파(propagation) 기법을 비교하는 연구가 이루어졌다.[12]

또한, 방송환경에서 동시성 제어에 관련된 연구 논문들이 국내외에서 발표되었으나 많은 논문들이 거래 개념을 도입하지 않았다. [13]의 논문에서는 이동 클라이언트/서버 컴퓨팅 환경에서 데이타 캐슁과 방송을 기반으로 한 방법을 처음으로 제안했다. 이 방법에서는 서버가 변화된 데이타 항목을 알리는 무효화(invalidation) 메세지를 주기적으로 방송하면 이동 클라이언트는 무선 채널을 통해 이 무효화 내용을 듣는다. 이 논문에 따르면, 이전 연구 노력의 대부분이 방송 보고서(broadcast report) 구조의 최적화[14], 그룹기반 캐쉬 무효화 전략[15], 그리고 분산된 캐슁 사용[16]과 같은 성능 분야에 초점을 두었다. 그것들은 주로 이동 컴퓨팅 환경이 통신 비용은 비싸고 통신자체가 안전하지 못한 특성을 갖는다는데 기반을 둔다. 그러나, 거래 개념과 거래 실행을 위한 동기화 문제가 이들 논문에서는 다루어지지 않았다.

그 외 거래 개념을 도입한 연구 논문들[17,18]도 발표되었으나 이러한 논문들은 실시간(real-time)의 쟁점이 없어 방송 기법의 효율성을 살리지 못했다는 문제점이 있다. [17]에서는 거래 관리에 있어서 방송 기술이 푸쉬 기반 데이타 전달을 위해 채택되었다. 예를 들어, 서버는 클라이언트의 구체적인 요구 없이 각 데이타 항목의 여러 버전을 버전넘버와 함께 반복적으로 방송한다. 그리고 이 논문에서는 읽기 전용 거래들의 일관성과 현재성을 보장하는 문제가 언급되어진다. 그러나, 이 방법은 상당한 방송 주기 횟수 증가와 그럼으로써 상당한 응답 시간(response time)의 증가를 가져온다. 이것은 매우 제한적이며 융통성이 부족하다. [18]에서는 직렬화 순서 그래프 검사에 기반을 둔 접근법을 사용하였으나 클라이언트들이 계속해서 방송에 귀 기울이고 있어야 함을 요구한다. 이러한 방법은 통신 단절과 같은 무선 환경의 특성으로 인해 많은 문제점이 노출된다.

우리와 비슷한 관심사로 동기가 되어 발표된 데이타주기(data cycle) 접근법[19]은 방송환경에 초점을 맞춘 최초의 동시성 제어 기법이나 무선 환경에 적용하기는 성능상 문제점이 있다. 이 데이타주기 구조는 유선망(network) 분산 데이타베이스시스템의 처리율을 높혔다. 데이타베이스의 전체 내용이 상당히 대역폭이 큰 채널을 가진 유선망에서 클라이언트로 반복적으로 방송된다. 이것은 클라이언트와 서버에서 수행하는 모든 거래들의 전역적 직렬화를 보장한다. 그러나, 직렬화는 그것을 보장하기 위해 서버와 무수한 통신이 필요하므로 불안정한 무선 채널을 가진 방송환경에 적용하기는 매우 비용이 많이 들기 때문에 상당한 성능 감소를 유발한다. 우리가 직렬화를 보장하기 위해 필요한 서버와의 상호작용으로 야기된 통신 비용을 피하기 위해서는 클라이언트들은 보수적이어야 하며 이것은 불필요한 철회를 발생시킨다.

갱신 일관성(update consistency)[9]이라 불리는 새로운 검증 기준을 적용한 논문[7]의 동시성 제어 기법에서는 방송환경에서 질의 거래들이 서버와 접촉 없이 현재성과 일관성을 만족하는 데이타를 읽도록 하였으나 계산 비용이 많이 드는 등 여러 문제점을 가진다. 이 논문에서는 직렬화를 정확성 검증 기준으로 채택하지 않아 위에서 언급한 통신비용과 성능적 측면에서 상당히 설득력이 있다. 또한 이 논문에서는 이러한 개념을 바탕으로 새로운 기법인 F-Matrix와 R-Matrix를 선보였다. 그러나, 이 기법들은 앞서 언급한 논문[19]에서의 데이타주기 접

근법보다 더 좋은 성능을 보였으나 제어 정보를 유지하기 위해 계산비용이 상당히 비싸고 높은 대역폭을 요구하여 시스템이 알고리즘을 처리하는데 상당한 부담이 있다.

가장 최근 발표된 방송환경에서 타임스탬프에 기반을 둔 동시성 제어 기법인 BCC-TI(Broadcast Concurrency Control using Timestamp Interval)[20]에서도 또한 질의 거래의 정확성 검증 기준으로 직렬화를 채택 즉, 질의 거래에 대한 비직렬화의 가능성을 미리 차단하기 때문에 이에 따르는 성능 감소의 문제점을 내포하고 있다. 이 논문에서는 방송응용시스템에서 대부분을 차지하는 질의 거래에 중점을 두어 타임스탬프 순서를 융통적으로 적용 가능한 장점을 활용하는 타임스탬프 기법을 사용하였다. 그리고, 이동 클라이언트들은 서버와 접촉없이 방송중인 일관성을 만족하는 데이타를 읽을 수 있도록 하여 클라이언트 서버간 자치성을 보장한다. 그러나 이 논문은 기존에 발표된 이동클라이언트/서버 환경을 위한 동시성 제어 기법에서 완료된 갱신 거래들이 항상 직렬화 순서에서 현재 수행중인 모든 질의 거래들보다 앞선다는 암시적인 가정 때문에 불필요한 철회가 발생한다는 문제점을 너무 단순하게 보완하였다. 그러므로, 상당한 거래 철회 및 재시작의 비효율성을 가진다. 이러한 문제점이 2.1 절에 자세히 나타나 있다.

2.1 BCC-TI를 질의 거래가 많은 환경에 적용했을 때의 문제점

갱신 거래에 대한 완료 단계에서 서버는 다음과 같이 수행한다. $WS(U)$를 갱신 거래 U의 쓰기 연산 대상 집합이라 하고 $TS(U)$를 U의 마지막 타임스탬프 값이라 한다. 그리고, $WTS(d)$를 데이타 항목 d를 쓰기 연산한 완료된 갱신 거래들의 타임스탬프 중 가장 큰 값이라 한다. U가 완료될 때 다음의 과정이 수행된다.

1단계 : 현재 타임스탬프를 $TS(U)$에 할당한다.

2단계 : $TS(U)$를 $WS(U)$에 속한 모든 데이타 항목 d에 대한 $WTS(d)$로 복사한다.

3단계 : $TS(U)$와 $WS(U)$를 제어정보테이블에 기록한다.

질의 거래의 읽기 연산 단계에서 각 이동 클라이언트는 다음과 같이 수행한다. 읽기 연산 단계에 있는 모든 질의 거래는 타임스탬프 구간이 할당되는데 그것은 거래 수행동안 야기된 임시 직렬 순서를 기록하는데 사용된다. 실행 시작 시점에서 질의 거래 타임스탬프 구간은 $[0, \infty]$으로 초기화된다. 그 후, 질의 거래 Q의 타임스탬프 구간의 최저점 $LB(Q)$와 최고점 $UB(Q)$는 각 읽기 연산 단계와 데이타 방송 주기 때마다 재갱신된다. 먼저, 질의 거래 Q가 데이타 항목 d를 읽을 때 다음과 같은 과정이 수행된다. 이 때에 $UB(Q)$는 변화되지 않는다.

1단계 : d를 읽는다.

2단계 : 현재의 $LB(Q)$와 $WTS(d)$ 중 큰 값을 $LB(Q)$로 새로이 정한다.

3단계 : $LB(Q)$가 $UB(Q)$보다 크거나 같다면 Q를 재시작한다.

데이타 방송 주기마다 각 이동 클라이언트는 다음과 같이 수행한다. CUT(Committed Update Transactions)는 마지막 방송 주기에서 완료된 갱신 거래들의 집합이라 하고 $CRS(Q)$(Current Read Set of Q)는 전 주기에서 질의 거래 Q에 의해 읽혀진 데이타 항목들의 집합이라 한다. 각 방송 주기에서 CUT에 속하는 거래 U에 대한 $WS(U)$와 $CRS(Q)$의 교집합이 공집합이 아니면 다음과 같은 과정이 그러한 모든 U에 대해 반복 수행된다.

1단계 : $UB(Q)$를 현재의 $UB(Q)$와 $TS(U)$ 중 작은 값을 $UB(Q)$로 새로이 정한다.

그러나, 이 기법은 질의 거래가 완료된 갱신 거래들을 직렬화 순서에서 앞설 수 있는 점은 보완하였으나 질의 거래가 여러 방송주기에 거쳐 일어날 경우 그 외 다른 갱신 거래들이 완료된 후 직렬화 순서에서 해당 질의 거래보다 또한 동시에 앞설 수 있다는 점을 함께 고려하지 않아 불필요한 철회가 발생한다. 예2.1에서 이러한 문제가 자세히 설명되고 있다.

[예 2.1] BCC-TI 기법을 질의 거래가 많은 시스템에 적용했을 경우의 철회:

방송 환경은 질의 거래가 대부분이지만 질의 거래간 데이타 충돌은 발생하지 않으므로 이 예에서는 임의의 질의 거래와 데이타 충돌을 일으킬 수 있는 갱신 거래들만을 한정하여 살펴본다. 질의 거래 T_4 그리고 갱신 거래 T_1, T_2, T_3가 이동 클라이언트측에서 각각 실행하고 있다고 가정한다. 또한, 매3초마다 제어정보테이블을 방송한다고 가정한다(그림 1).

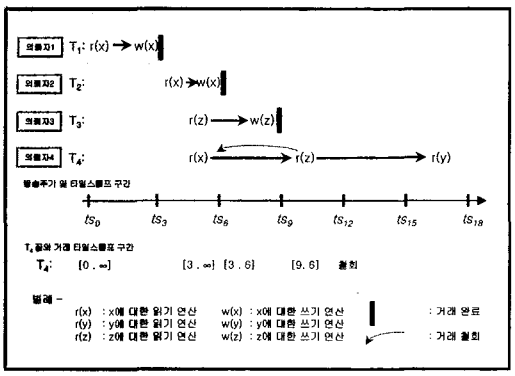


그림 1 BCC-TI를 적용했을 경우 거래들의 철회

위의 그림에서 질의 거래 T_4는 갱신 거래 T_2, T_3와 질의 거래 T_4간에 직렬 순서가 다음과 같이 어긋나지 않음에도 불구하고 철회된다(그림 2).

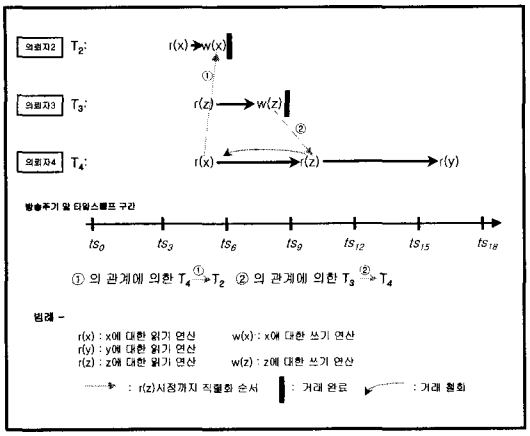


그림 2 직렬화 가능성의 T_2, T_3와 T_4의 거래 스케줄

즉, 다음과 같은 과정에 의해 T_4는 철회된다. 질의 거래 시작시점에서 타임스탬프 구간은 [0, ∞]이다. T_1이 x를 갱신하고 ts_3시점에서 완료한 가장 최근 거래이기 때문에 질의 거래 T_4의 r(x) 연산시점에서 WTS(x)는 3이다. 그리므로 $LB(T_4)$가 3으로 조정된다. 그 후 갱신 거래 T_2가 x를 갱신하고 완료되어 $UB(T_4)$가 6으로 조정된다. ts_9시점에서 T_3가 완료되고 WTS(z)는 9가 된다. 연산 r(z)시점에서 WTS(z)의 값이 9이므로 $LB(T_4)$가 다시 9로 조정된다. 그 결과 $LB(T_4)$가 $UB(T_4)$값 6보다 크므로 $T_3 \rightarrow T_4 \rightarrow T_2$의 직렬 순서가 어긋나지 않음에도 불구하고 그 시점에서 수행중인 T_4는 철회된다. □

BCC-TI 후에 발표된 Quasi-TI[21]는 유사 일관성(quasi consistency)이라 불리는 BCC-TI보다 더 느슨한 검증 기준을 채택하였다. 유사 일관성이란 질의 거래에 의해 읽혀지는 데이타의 부정확성을 어느 정도 수준에서 허용하는 것을 말한다. 그러므로, 이 기법은 클라이언트측에 캐싱(caching)기법을 도입하여 캐쉬데이타와 서버 데이타와의 값에 있어서의 차이, 캐쉬 데이타 버전이 존재하여 발생하는 오차와 캐쉬 데이타와 서버 데이타가 기록되는 시간 상의 차이의 세 가지 제약 사항을 어느 한계 범위 내에서 허용할 때 BCC-TI보다 더 성능이 향상됨을 보여주었다. 그러나, Quasi-TI에서의 이러한 세 가지 제약 조건 즉, 부정확한 값의 한계, 버전 수의 한계, 시간의 한계 측정을 위해 유지 비용이 많이 든다는 단점이 있다.

3. 방송환경에서 타임스탬프 간격에 기반을 둔 낙관적 동시성 제어 기법

새로이 제안하고자 하는 기법은 방송환경에서 타임스탬프 간격에 기반을 둔 낙관적 동시성 제어 기법(Optimistic Concurrency Control based on Timestamp Interval: OCC/TI)이라 칭한다.

이동 거래가 완료되면 거래에 의해 접근된 데이타 항목에 대한 거래 일관성이 보장되어야 한다. 이 논문에서 이동 갱신 거래들을 검증하기 위해 거래를 우선 수행하고 나중에 유효화 단계를 거치는 낙관적 동시성 제어 기법을 사용한다. 낙관적 기법이 이동 클라이언트들과 서버 사이에 주고받는 메시지 수를 최소화하기 때문이다. 낙관적 동시성 제어 기법을 사용할 때 서버는 매 거래 완료 시점에서 그 거래를 포함하는 실행이 직렬화를 만족하는지 아닌지를 검사 할 필요가 있다. 직렬화를 만족하지 않으면 완료를 하려는 거래는 철회되고 그렇지 않으면 완료를 무사히 마친다. 낙관적 동시성 제어에서 거래를 유효화하는데 있어서 후방향(backward)과 선방향(forward)의 두 가지 유효화 과정 형태가 있다[22]. 후방향 유효화는 완료하려는 거래가 최근에 완료된 다른 거래에 의해 무효화되어지지 않는다는 것을 보장하는 것을 검사한다. 선방향 유효화는 완료하려는 거래가 다른 어떤 활동중인 거래와 충돌하지 않는다는 것을 보장하는 것을 검사한다.

이동 클라이언트/서버 컴퓨팅 환경에서 이동 클라이언트측 거래는 서버가 실행중인 이동 거래들에 대해 잘 모르기 때문에 후방향 유효화 방법을 선택한다. 클라이언트측은 유효화 될 거래의 읽기연산 항목 집합이 이미 완료된 다른 거래의 쓰기연산 항목 집합과 비교되는 순수한 후방향 유효화 방법을 사용한다. 클라이언트측 거래의 유효화 과정 대부분을 그 클라이언트내에서 자체적으로 해결하도록 함으로써 유효화 과정이 진정으로 분산되어 행해질 수 있다.

또한, 완료된 갱신 거래들은 직렬화 순서에서 항상 실행중인 모든 질의 거래들을 앞선다는 가정 때문에 불필요한 질의 거래 철회가 발생되는 기존의 낙관적 동시성 제어 기법과는 달리 질의 거래에 대해서는 타임스탬프 순서에 기반을 둔 타임스탬프 구간 기법을 사용하여 이러한 문제점을 보완한다. 질의 거래를 위한 본 논문의 타임스탬프 구간 기법에서는 정확성 검증기준으로 직렬화를 바탕으로 하는 엄격한 일관성(strict consistency)[9]을 채택하지 않고 더 느슨한 약한 일관성[9]을 적용시켜 질의 거래의 철회를 줄이고 데이타의 현재성을 만

족시킨다. 약한 일관성이란 직렬화 그래프상 갱신 거래들만으로 이루어진 순환(cycle)과 하나의 질의 거래와 갱신 거래들로 이루어진 순환은 허용이 안되고 그 외 사이클은 허용하는 일관성 형태를 말한다. 즉, 질의 거래는 갱신 거래들과의 직렬화 순서를 유지하되 동시에 수행 중인 다른 질의 거래들은 그것과 같은 순서를 반드시 유지하지는 않는다.

직렬화를 바탕으로 하는 엄격한 일관성을 정확성 검증 기준으로 채택할 때의 문제점은 관련연구에서 살펴본 논문[7]에 잘 나타나 있다. 첫째, 각각 다른 클라이언트에서 수행 중인 모든 질의 거래들이 동시에 수행중인 갱신 거래들과의 직렬화 순서를 똑같이 유지해야함으로써 불필요한 질의 거래 철회가 많이 일어난다는 것이다. 둘째, 각 질의 거래는 그 질의 거래가 읽은 데이타에 어떤 영향도 주지 않는 갱신 거래를 포함하여 동시에 수행 중인 모든 갱신 거래들과의 사이에 직렬 순서를 유지해야함으로써 또한 불필요한 질의 거래 철회가 발생한다는 것이다.

질의 거래를 위한 정확성 검증기준으로 약한 일관성을 채택할 때의 장점은 이동 교통 정보 또는 기상 정보 시스템과 같은 응용에서 잘 나타난다. 거리 X와 Y의 교통량을 읽는 각 클라이언트의 질의 거래를 Q_1과 Q_3라 하고 U_2와 U_4를 각각 X와 Y의 교통량을 갱신하는 갱신 거래들이라 하자. 또한, 거래 X와 Y의 처음 교통량을 둘 다 100이라 가정한다. 다음의 거래 실행 스케줄을 고려해 보자.

$r_1(X)$ $w_2(X)$ c_2 $r_3(X)$ $r_3(Y)$ $w_4(Y)$ c_4 $r_1(Y)$....

이 실행 스케줄은 $Q_1 \rightarrow U_2 \rightarrow Q_3 \rightarrow U_4 \rightarrow Q_1$이다. 이 거래 스케줄은 전역적(global) 직렬성(serialization)을 만족하지 않지만 질의 거래 Q_1과 Q_3는 위에서 언급한 약한 일관성을 만족한다. 그러므로, 질의 거래 Q_1과 Q_3는 철회 없이 무사히 진행하여 완료가능하다. Q_1과 Q_3가 완료할 때 각 클라이언트들은 직렬 가능한 실행을 본다. 단, Q_1과 Q_3는 갱신 거래들과의 사이에 발생하는 직렬화 순서가 같지는 않다. 즉, 갱신 거래들간 직렬 순서는 $U_2 \rightarrow U_4$이고 Q_1과 Q_3의 다른 갱신 거래들과의 직렬 순서는 각각 $U_4 \rightarrow Q_1 \rightarrow U_2$과 $U_2 \rightarrow Q_3 \rightarrow U_4$이다. X의 교통량이 110으로 오르는 것을 갱신 거래 U_2가 행하고 Y의 교통량이 90으로 내려가는 것을 갱신 거래 U_4가 행한다고 가정하자. 여기서 질의 거래 Q_1과 Q_3는 다른 직렬화 순서를 보므로 Q_1은 거리 Y의 교통량은 감소하고 X의 교통량은 그대로인 것으로 Q_3는 거리 X의 교통량은 증가하고 Y의 교통량은 그대로인 것으로 볼 것이다. 그러나, 이동 교통 정보 시스템에서 각 클라이언트 단말기를 통해 보는 교통량은 교통량의 증가 또는 감소 데이타가 서버에 주기적으로 보내어지기 때문에 클라이언트 단말기에 보여지는 교통량 정보는 시간 간격으로 인해 또는 단말기의 처리 상태 등으로 인해 실제 데이타와는 오차가 존재한다. 그러므로, 약한 일관성을 적용함으로 인해 생긴 이동 교통 정보 시스템에서의 교통량 정보의 실시간 오차는 사용자들이 지름길을 판단하는데 큰 문제를 일으키지 않는다.

OCC/TI의 시스템 모델은 다음과 같다. 이동 클라이언트의 사용자들은 읽기와 쓰기 연산으로 이루어진 이동 거래에 의해 데이타베이스에 자주 접근할 것이다. 한 이동 클라이언트에 의해 어느 시각에 단지 한 거래만 실행할 수 있다고 가정한다. 즉, 클라이언트는 한 거래가 완료된 후에야 다른 거래를 발생시킬 수 있다. 또한, 각 이동 거래는 먼저 읽기 연산에 의해 읽혀진 데이타 항목들의 부분집합만을 갱신할 수 있고 하나의 데이타 항목은 질의 거래에서 단지 한 번 읽힌다고 가정한다. 데이타베이스는 서버에 의해 저장되고 유지되는 데이타 항목들의 집합이다. 갱신은 이동 클라이언트에 의해 발생되지만 궁극적으로 데이타베이스에 반영되어야 한다. 클라이언트는 서버에게 데이타를 요구하고 서버는 주로 클라이언트들에게 해당 데이타와 제어 정보를 방송하기 위해 방송매체를 사용한다. 또한, 거래의 실행은 원자적(atomic)이다. 즉, 거래는 완료되거나 철회된다.

클라이언트측 거래 일관성을 보장하기 위해 서버는 주기적으로 매 L초마다 $C_i = iL$인 시간에 제어정보를 방송하고 모든 실행중인 이동 클라이언트들은 이 정보를 기다려 그 내용에 따라 그들 거래의 유효화 과정을 진행한다. 서버측의 제어정보테이블(control information table)에는 가장 최근 마지막 방송주기에서 완료된 각 갱신 거래 T에 대한 T의 타임스탬프 $TS(T)$와 그것에 의해 갱신된 데이타 항목 집합 $WS(T)$ 및 그 거래에 의해 읽혀진 데이타 항목 집합 $RS(T)$를 유지한다. 또한, 마지막 방송주기를 C_i라 할 때 C_i에서 갱신된 모든 데이타 항목들의 집합을 최근 갱신 데이타 집합(current data set)인 $CD(C_i)$에 유지한다.

앞 장에서 설명한 BCC-TI의 문제점을 보완하고자 본 논문의 알고리즘은 다음과 같은 과정을 밟는다. 질의 거래의 어느 읽기 연산시점에서 타임스탬프 구간의 최저임계치 LB가 최고임계치 UB보다 크거나 같게 되더라도 앞장의 예2.2와 같이 동시 수행 중인 갱신 거래들과의 직렬화가 가능하다면 UB값을 거래 시작 시점의 UB값으로 복원한다. 따라서, 타임스탬프 구간이 넓혀짐으로써 해당 질의 거래의 실행이 철회 없이 계속되어지도록 한다.

질의 거래와 다른 갱신 거래간의 직렬화가 가능한 경우는 앞장의 예2.1에서 처럼 $T_4 \rightarrow T_2$와 $T_3 \rightarrow T_4$의 직렬 순서가 존재할 경우에 $T_2 \rightarrow .. \rightarrow T_3$의 직렬 순서의 가능성이 없으면 약한 일관성을 보장할 수 있다. T_3의 쓰기 연산 집합과 T_2와 그 밖의 모든 완료된 거래들의 읽기 연산 집합과의 사이에 공통된 데이타 항목이 없고 동시에 T_3의 읽기 연산 집합과 T_2와 그 밖의 모든 완료된 거래들의 쓰기 연산 집합과의 사이에 공통된 데이타 항목이 없는 경우에는 $T_2 \rightarrow .. \rightarrow T_3$의 직렬 순서가 존재할 수 없다. 그러므로 본 논문이 제시하는 알고리즘은 어느 읽기 연산 시점에서 LB가 커져서 UB보다 크게 되더라도 위의 요구조건을 만족하는 경우에는 해당 질의 거래와 갱신 거래들간의 비직렬화 가능성이 없으므로 UB를 최초의 값인 ∞로 복원하여 철회없이 계속 거래 수행을 하도록 한다.

이동 거래들의 약한 일관성을 성취하기 위한 이동 클라이언트측 알고리즘은 다음과 같다. 각 이동 클라이언트들은 각 거래가 갱신 거래인지 질의 거래인지를 거래 수행 시작시점에서 안다고 가정한다. 각 이동 클라이언트들은 하나의 질의 거래가 시작되면 거래 완료시점까지 완료된 갱신 거래들에 대한 정보를 저장할 질의 거래 당 하나의 *TotalCommitList*를 유지한다. 이 리스트는 *CommitList*를 참조하여 이루어지는데 *CommitList*는 서버에서 무사히 완료된 거래들을 기록한 리스트로 제어정보테이블과 함께 방송된다. 즉, *TotalCommitList*에는 완료된 거래 T_{id}와 그에 대한 제어정보테이블의 내용인 T_{id}에 의해 갱신된 데이타 집합 $WS(T_{id})$와 T_{id}에 의해 읽혀진 데이타 집합 $RS(T_{id})$를 보관한다. 이동 클라이언트는 또한 제어정보를 받았던 마지막 시간을 가리키는 C_{lb}를 유지한다. C_{lb}는 이동 클라이언트가 통신 단절로부터 복구된 후 마지막 받은 제어정보의 시간을 알 수 있도록 신뢰할만하게 유지된다.

각 제어정보테이블이 도착할 때 각 이동 클라이언트는 접근된 데이타에 대한 거래 일관성을 체크하고 <알고리즘 3.1>과 같은 알고리즘을 수행한다.

읽기 연산을 수행하는 과정 중 특히 질의 거래의 읽기 연산은 다음과 같이 처리된다. 모든 질의 거래들은 타임스탬프 구간이 할당되는데 그것은 그 거래의 실행 동안 다른 갱신 거래들과의 관계에서 직렬화를 만족하는지를 체크하기 위해 사용된다. 실행 시작시점에서 질의 거래의 타임스탬프 구간은 $[0, \infty]$으로 시작된다. 즉, 타임스탬프 전체 영역 공간이다. 질의 거래의 직렬화 순서가 그것의 읽기 연산 또는 갱신 거래들의 쓰기 연산에 대한 데이타 충돌 검사에 의해 정해질 때마다 그 질의 거래의 타임스탬프 구간이 다른 갱신 거래들과의 의존성을 반영하기 위해 조정된다.

알고리즘 3.1 이동 거래를 처리하기 위한 클라이언트측 알고리즘

```
1. On receiving r_T(d) {
      if T is query(read_only) transaction {
            LB(T) = max(LB(T), WTS(d));
            if LB(T) ≥ UB(T) {
                  valid = true;
                  while ( exists U_i ∈ all the transactions in TotalCommitList ) {
                   ( i ≠ TID(d) and U_i ∉ TidList )
                        if (RS(U_i) ∩ WS(TID(d)) ≠ {} or WS(U_i) ∩ RS(TID(d)) ≠ {}) {
                              valid = false; exit;
                        }
                  }
                  if ( valid == true) {
                        UB(T) = ∞;
                        copy TID(d) into TidList.
                  }
                  else {abort T; }
            }
      }
      }
2. On receiving commit_T request {
      if T is update transaction {
          send a RequestCommit message, along with T_id, RS(T_id), WS(T_id) and C_T
      }
      else {commit T; }
      }
3. On receiving the control information table CIT(C_i) {
      if T_id appears in CommitList { commit T; }
      if T_id identifier appears in AbortList { abort T; }
      if (C_i - C_lb > wt) { abort T; }
      else {
            if CO(C_i) ∩ RS(T) ≠ {} {
                        if T is update transaction { abort T; }
                        else {
                              UB(T) = min(TS(U_i), UB(T))
                  ( ∀ U_i ∈ all the transactions in CommitList and WS(U_i) ∩ RS(T) ≠ {} )
                        }
            }
            else {
                        if T is update transaction {
                              record C_i ;
                              T is allowed to continue;
                        }
            }
            copy U_i , RS(U_i), WS(U_i) to TotalCommitList
            ( ∀ U_i ∈ all the transactions in CommitList )
      }
      }
```

하나의 질의 거래와 갱신 거래들과의 관계에서 질의 거래에 의한 비직렬화 가능성을 가진 실행의 발견은 단지 타임스탬프 구간을 사용함으로써 이루어진다. 질의 거래의 직렬 순서는 완료된 갱신 거래들에 대해 검사되어지기 때문에 우리는 갱신된 데이타 항목들에 대한 타임스탬프가 필요하다. 즉, 질의 거래가 읽은 데이타 항목 d를 갱신한 완료된 갱신 거래들의 타임스탬프 중 가장 큰 값인 $WTS(d)$가 필요하다.

질의 거래가 한 데이타 항목을 읽을 때마다 그 자신과 완료된 갱신 거래들 사이에 형성된 직렬화를 반영하기 위해 타임스탬프 구간이 조정된다. $LB(T)$와 $UB(T)$를 각각 질의 거래 T의 타임스탬프 구간 최저 임계치와 최고 임계치라 하자. $LB(T)$는 데이타 항목 d를 읽는 시점에서 현재의 $LB(T)$와 $WTS(d)$ 중 큰 값으로 조정한다. 이때, 타임스탬프 구간의 $LB(T)$가 $UB(T)$보다 크거나 같게 되면 BCC-TI는 무조건 그 질의 거래를 철회시켰다.

그러나, 우리의 알고리즘은 데이타 항목 d를 읽는 시

점에서 $LB(T)$가 $UB(T)$보다 크거나 같게 되면 다음과 같은 과정으로 직렬화 가능성이 있는지를 점검한다. 먼저, 데이타 항목 d를 마지막 갱신하고 완료한 거래를 $TID(d)$라 하자. *TotalCommitList*에 유지된 모든 완료된 갱신 거래 U에 대해 $TID(d)$의 쓰기 연산 집합과 U의 읽기 연산 집합 또는 $TID(d)$의 읽기 연산 집합과 U의 쓰기 연산 집합의 교집합이 공집합이 아닌 U가 존재하면 비직렬화 가능성이 있으므로 그 거래를 철회한다. 그리고, 그러한 U가 존재하지 않는다면 직렬화를 만족하므로 좁혀진 타임스탬프 구간을 만든 최고 임계치 $UB(T)$를 무시하고 거래 시작 시점의 $UB(T)$값 ∞으로 복원하여 타임스탬프 구간을 넓힘으로써 해당 질의 거래가 철회 없이 계속 진행되도록 한다. 이러한 경우에 d를 마지막 갱신하고 완료한 거래 $TID(d)$는 해당 질의 거래와 갱신 거래들로 이루어진 직렬화 그래프상 순환을 만들 수 있는 직렬 순서에 영향을 주지 않으므로 *TidList*에 따로 구분 보관한다. 그럼으로써, *TidList*에 속한 거래는 다음 직렬화 검사 때부터는 *TotalCommitList*에 유지하여 위의 과정처럼 비교되는 거래 대상에서 제외시킨다.

마지막 읽기 연산까지 철회 즉, 재시작이 발생되지 않으면 그 질의 거래는 무사히 완료된다. 이와 같이 서버에게 어떤 완료요구 없이 각 클라이언트가 갱신 거래들과의 직렬성을 위배하지 않으며 질의 거래를 독립적으로 처리할 수 있도록 한다. 또한, 질의 거래의 각 읽기 연산시마다 그 동안 완료된 갱신 거래들과의 사이에 직렬성을 만족하지 않을 가능성이 있다면 발견 즉시 그 거래를 철회, 재시작하기 때문에 이른 데이타 충돌 탐지가 가능하다.

갱신 거래 T에 대한 완료를 해야할 때 거래의 구분자 T_{id}, $RS(T_{id})$, $WS(T_{id})$ 그리고 해당 방송 주기 C_i와 함께 *RequestToCommit*메시지를 서버에게 보낸다.

그 후에 각 제어정보테이블이 방송될 때마다 거래 T에 대한 완료를 요구했던 이동 클라이언트는 접근된 데이타에 대한 거래 일관성을 체크하기 위해 $CIT(C_i)$ 즉, 제어정보테이블과 함께 실려온 *CommitList*와 *AbortList*를 주시한다. 그리고, 해당 T_{id}가 이 두 개의 리스트 중 어느 곳에 있는지를 검사한다. T_{id}가 *CommitList*에 있다면 T는 무사히 완료된다. 그러나, T_{id}가 *AbortList*에 나타나면 T는 철회된다.

질의 거래와 갱신 거래들을 포함한 모든 이동 거래들은 요구하여 방송되어진 데이타 항목을 접근하기 앞서 방송주기 시작시점에서 자체내 부분 유효화 과정이 행해진다. 그것은 마지막 방송 주기에서 완료된 거래들에 대해서 이루어진다. 완료된 거래들은 직렬화 순서에서 유효화 단계의 거래를 앞서기 때문에 유효화 단계의 갱신 거래의 읽기 연산 집합과 완료된 거래들의 갱신된 항목 집합을 비교함으로써 데이타 충돌이 감지된다. 그러한 데이타 충돌이 일어나면 유효화 과정의 갱신 거래를 철회하고 재시작시킴으로써 갱신 거래들의 직렬화 순서가 보장된다.

즉, 각 방송주기 시작시점에서 마지막 방송 주기 C_i에서 갱신 완료된 데이타 항목들의 집합인 $CD(C_i)$와 유효화 과정에 있는 거래 T의 $RS(T)$ 사이에 공통 데이타 항목이 존재한다면 T가 갱신거래일 경우는 철회되고 T가 질의 거래일 경우는 다음 단계와 같이 질의 거래 T의 타임스탬프 구간의 최고 임계치를 갱신한다. *CommitList*에 속하는 거래 U 중 U의 쓰기 연산 집합 $WS(U)$와 $RS(T)$의 교집합이 공집합이 아닌 모든 U에 대해 $UB(T)$를 현재의 $UB(T)$와 $TS(U)$ 중 작은 값을 $UB(T)$로 조정하는 과정을 반복 수행한다.

그러나, $CD(C_i)$와 $RS(T)$ 사이에 공통 데이타 항목이 존재하지 않는다면 T가 갱신거래일 경우에 방송 주기 C_i를 저장한 후 수행을 계속한다. 마지막으로, *CommitList*에 있는 모든 갱신 거래들과 그 거래들의 $WS(T_{id})$와 $RS(T_{id})$를 위해서 언급한 직렬성 조사 때에 사용되도록 *TotalCommitList*에 저장한다.

방송 환경에서 통신 단절이 발생할 경우, 각 이동 클라이언트는 $C_i - C_{lb} > \omega L$(ω는 무효화 방송 윈도우)과 같이 ωL보다 더 오래 단절된다면 그것의 수행중인 거래를 철회한다. 즉, 이동 거래는 무효화 방송 윈도우 ω값까지는 단절로 인한 문제를 극복할 수 있다.

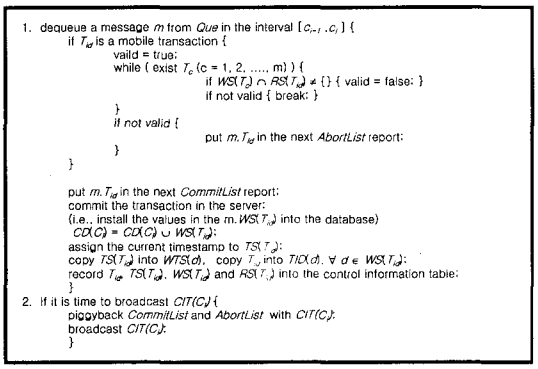
이동 거래들의 일관성을 유지하기 위해 서버는 이동 클라이언트들로부터 *RequestToCommit*메시지들을 받아 큐에 유지한다. 각 *RequestToCommit*메시지 m은 $m.T_{id}$, $m.RS(T_{id})$, $m.WS(T_{id})$, $m.C_i$와 같은 항목필드들을 갖는다.

서버측 유효화 과정은 이동 클라이언트에서 행해진 부분 유효화 과정 이후에 완료된 거래들이 있을 수 있기 때문에 필요하다. 그러므로, 이동 클라이언트에서 행해진 마지막 유효화 과정의 방송 주기가 최종 유효화 과정을 위해 서버로 보내진다. T_{id}를 유효화 단계 거래라 하고 C_k는 T_{id}와 함께 딸려온 방송주기 값이라 하자. 즉, T_{id}는 방송주기 C_k때 이동 클라이언트에서 부분 유효화 처리되었다. 이동 거래는 자체 부분 유효화 과정 후 최종 유효화 과정을 위해 서버로 보내어지기 전에 그 거래가 갱신한 데이타 항목을 다른 거래들이 읽거나 쓸 수 있다. $T_c(c = 1, 2,..., m)$를 C_k 이후 완료된 거래들이라 하자. 클라이언트측 알고리즘에서 언급한 것처럼 거래 T_{id}의 읽기 연산 집합과 쓰기 연산 집합을 각각

$RS(T_{id})$와 $WS(T_{id})$로 표기한다. $CD(C_i)$는 현재 방송주기에서 갱신된 데이타 항목들의 집합이고 그것은 모든 방송주기 시작시점에서 초기화된다.

서버가 처리해야 할 자세한 알고리즘은 <알고리즘 3.2>에 나타나 있다.

알고리즘 3.2 이동 갱신 거래를 처리하기 위한 서버측 알고리즘



최종 후방향 유효화 과정은 다음과 같다. T_{id}의 $RS(T_{id})$와 임의의 T_c의 $WS(T_c)$의 교집합이 공집합이 아니면 즉, T_{id}보다 T_c가 먼저 같은 데이타 항목을 갱신하고 완료하였다면 T_{id}는 철회된다. 무사히 완료된 T_{id}의 $WS(T_{id})$는 서버측 데이타베이스에 반영되고 현재의 방송주기 C_i의 $CD(C_i)$에 $WS(T_{id})$가 포함된다. 이러한 내용의 제어정보테이블이 다음 방송주기에 각 이동 클라이언트에게 방송되어진다. 클라이언트측 알고리즘에서 언급했듯이 이러한 제어정보테이블은 이동 클라이언트가 자체내 부분 유효화 과정을 행하는데 사용된다. 그리고 이러한 이동 거래의 완료 또는 철회 결과가 각각 *CommitList*와 *AbortList*에 기록되어 각 방송 주기마다 제어정보테이블을 방송할 때 함께 실어 보내진다.

모든 갱신 거래는 완료될 때 최종 타임스탬프 값이 할당된다. $TS(T_{id})$를 거래 T_{id}의 최종 타임스탬프라 하자. T_{id}가 완료할 때 다음과 같은 단계를 실행한다. 먼저, 현재 타임스탬프를 $TS(T_{id})$에 할당하고 $TS(T_{id})$와 T_{id}를 각각 $WS(T_{id})$에 속한 모든 데이타 항목 d에 대한 $WTS(d)$와 $TID(d)$에 복사한다. 그 다음 T_{id}, $TS(T_{id})$, $WS(T_{id})$와 $RS(T_{id})$를 제어정보테이블에 기록한다.

OCC-UI에서는 질의 거래가 마지막 읽기 연산 시점까지 다른 갱신 거래들과의 비직렬화 가능성이 없는 즉, 거래 일관성이 만족되는 경우에는 다른 갱신 거래들의 완료와 상관없이 무사히 완료된다. 이러한 사항이 예 3.1에서 자세히 설명되어 있다.

[예 3.1] OCC/TI를 질의 거래가 많은 환경에 적용했을 경우 완료:

갱신 거래 T_1, T_2, T_3, T_5및 T_6과 질의 거래 T_4가 이동 클라이언트측에서 각각 실행하고 있다고 가정한다. 또한, 매3초마다 제어정보테이블을 방송한다고 가정한다. T_6은 T_5보다 거래 시작이 먼저 된 후 무선환경의 단절 특성에 따른 실행 지연으로 완료가 늦어진 경우라 가정한다(그림 3).

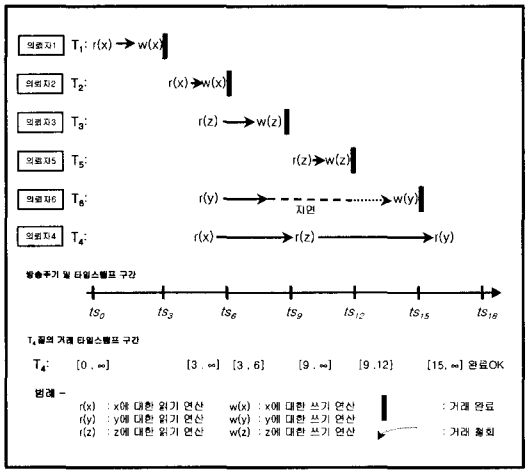


그림 3 OCC/TI를 적용했을 경우 질의 거래의 완료

질의 거래 시작시점에서 타임스탬프 구간은 [0, ∞]이다. 질의 거래 T_4의 r(x)시점에서 T_1이 x를 갱신하고 타임스탬프 3에서 완료한 가장 최근 거래이기 때문에 $wds(x)$는 3이다. 그러므로, $LB(T_4)$가 3으로 조정된다. 그 후 갱신 거래 T_2가 x를 갱신하고 완료되어 $UB(T_4)$가 6으로 조정된다. 갱신 거래 T_3가 z를 갱신하고 완료 방송되어 $wds(z)$는 9가 된다. 그러므로, 연산 r(z)시점에서 $LB(T_4)$가 9가 되어 $UB(T_4)$인 6보다 커서 이 시점에서 다른 갱신 거래들과의 직렬화 가능성이 있는지 검사한다. *TotalCommitList*에 있는 거래 T_2의 $RS(T_2)$와 $TID(z)$인 T_3의 $WS(T_3)$ 사이에 공통의 데이타 집합이 없고 T_2의 $WS(T_2)$와 $TID(z)$인 T_3의 $RS(T_3)$ 사이에 공통의 데이타 집합이 없기 때문에 비직렬화 가능성이 없으므로 $UB(T_4)$을 ∞으로 복원한다. 따라서, 질의 거래 T_4의 진행은 철회 없이 계속된다. 그 후 갱신 거래 T_5가 z를 갱신하고 완료되어 $UB(T_4)$가 12로 다시 조정된다. 갱신 거래 T_6가 y를 갱신하고 완료 방송되어 $wds(y)$는 15가 된다. 마지막 읽기 연산 r(y)시점에서

$LB(T_4)$가 15가 되어 $UB(T_4)$인 12보다 크게 된다. 이 시점에서 바로 전단계 연산시처럼 직렬화 가능성이 있는지 검사한다. *TotalCommitList*에 있는 거래들 중 T_3를 제외한 T_2, T_5의 $RS(T_2)$ 또는 $RS(T_5)$와 $TID(y)$인 T_6의 $WS(T_6)$ 사이에 공통의 데이타 집합이 없고 $WS(T_2)$ 또는 $WS(T_5)$와 $TID(y)$인 T_6의 $RS(T_6)$ 사이에 공통의 데이타 집합이 없기 때문에 동시 수행 중인 갱신 거래들과의 직렬화가 가능하다. 그 결과, $UB(T_4)$가 다시 ∞으로 복원되어 마지막 읽기 연산시점까지 $LB(T_4)$가 $UB(T_4)$보다 크거나 같지 않으므로 질의 거래 T_4는 동시에 수행 중인 거래들 T_2, T_3, T_5및 T_6와의 사이에 직렬성이 보장되어 무사히 완료된다. □

4. 정확성 검증

이동 클라이언트측 질의 거래는 어느 읽기 연산 시점에서든지 거래 일치 상태에 있다. 이것은 서버가 단지 갱신 거래들의 직렬가능한 수행만 허락하고 제어정보테이블을 이용하여 거래적으로 일치하도록 함으로써 항상 일관성 있는 데이타베이스 상태를 유지한다. 이동 클라이언트에 의해 접근된 데이타에 대한 일관성 검사가 다음의 정리 4.1과 정리 4.2에 기반을 둔다.

정리 4.1 이동 클라이언트에서 수행 중인 질의 거래 T_1에 의한 읽기 연산 시점 ts_i에서 타임스탬프 구간 값 중 최저 한계치가 최고 한계치보다 크거나 같은 상황이 발생할 때 그 시점까지 다른 갱신 거래들과의 비직렬화 가능성이 없으면 T_1의 실행이 계속될 수 있도록 최고 한계치를 ∞으로 복원하여 최저 한계치보다 최고 한계치를 크게 하여도 T_1과 갱신거래들은 직렬가능하다.

증명: ts_i시점까지 최저 한계치가 최고 한계치보다 크거나 같은 상황이 발생할 때 다른 갱신 거래들과의 비직렬화 가능성이 없다고 가정했다. T_1전에 완료된 갱신 거래들을 순서대로 T_2, T_3 등이라 하면 직렬화 그래프상 직렬 순서 $T_1 \rightarrow T_2$와 $T_3 \rightarrow T_1$이 존재한다. 즉, $T_1 \rightarrow T_2$가 존재한다는 것은 T_1에 의해 읽혀지고 T_2에 의해 갱신된 공통의 데이타 항목이 존재한다는 것이다. T_1이 시작할 때 [0, ∞]의 타임스탬프 구간이 T_2가 완료되는 시점에서 $[0, TS(T_2)]$가 된다. 그 후 T_3가 완료되고 T_1의 읽기 연산 ts_i시점에서 $T_3 \rightarrow T_1$이 존재한다는 것은 T_3가 갱신한 데이타 항목이 T_1의 읽기 연산 대상이 된다는 것이므로 $LB(T_1)$의 값은 $TS(T_3)$가 되어 타임스탬프 구간이 $[TS(T_3), TS(T_2)]$가 된다. 가정에서 비직렬화 가능성이 없다고 했기 때문에 $T_1 \rightarrow T_2$, $T_3 \rightarrow T_1$과 함께 직렬 순서 $T_2 \rightarrow .. \rightarrow T_3$가 존재함으로 인한 직렬화 그래프상 순환이 존재하지 않는다. 그러므로, $T_3 \rightarrow T_1 \rightarrow T_2$ 형태만 존재하고 가정에서처럼 T_1의 타임스탬프 구간이 $[TS(T_3), ∞]$가 되어 실행이 계속 되어진다. 이제는 ts_i 시점 이후 완료된 갱신 거래로 인해 비직렬화가 가능한지 살펴보자. $[TS(T_3), ∞]$에서 $UB(T_1)$이 변화되는 경우와 $LB(T_1)$이 $UB(T_1)$보다 크게 되는 경우에 직렬가능한지 증명하면 된다. 첫째, $UB(T_1)$의 값이 변화되는 경우를 살펴본다. ts_i시점에서 T_1의 읽기 연산 대상과 공통의 데이타 항목을 쓰기 연산 후 ts_i시점 이후 완료한 갱신 거래를 T_4 등이라 하면 T_4가 완료되는 시점에서 타임스탬프 구간이 $[TS(T_3), TS(T_4)]$가 되고 직렬 순서 $T_3 \rightarrow T_1 \rightarrow T_2 \rightarrow T_4$ 또는 $T_3 \rightarrow T_1 \rightarrow T_2$와 함께 $T_3 \rightarrow T_1 \rightarrow T_4$가 존재할 수 있다. 둘 중 어떤 형태라도 T_4는 T_3가 완료된 후에 시작하므로 $T_4 \rightarrow .. \rightarrow T_3$의 직렬 순서는 동시에 존재할 수 없다. 왜냐하면, T_4는 T_3가 완료되기 전에 시작하여 $T_4 \rightarrow .. \rightarrow T_3$가 존재하려면 완료순서가 T_3, T_4이므로 T_3의 쓰기 연산 집합과 수행 중인 T_4의 읽기 연산 집합과의 교집합이 공집합이 아닌 경우인데 이런 경우는 이미 ts_i시점에서 비직렬화 가능성으로 인해 $UB(T_1)$의 값을 ∞으로 복원하지 않고 철회되었을 것이기 때문이다. 둘째, $LB(T_1)$이 $UB(T_1)$보다 크게 되는 경우를 살펴본다. ts_i시점 이후 읽기 연산 시점에서 T_1의 읽기 연산 대상과 공통의 데이타 항목을 쓰기 연산한 후 가장 최근 완료된 그리고 T_4 뒤에 완료된 갱신 거래를 T_5 등이라 하면 해당 읽기 연산 시점에서 타임스탬프 구간이 $[TS(T_5), TS(T_4)]$가 되고 직렬 순서 $T_3 \rightarrow T_5 \rightarrow T_1 \rightarrow T_2$ 또는 $T_3 \rightarrow T_1 \rightarrow T_2$와 함께 $T_5 \rightarrow T_1 \rightarrow T_2$가 존재할 수 있다. 어떤 경우에든지, 위에서 언급한 T_4를 포함한 직렬 순서와 함께 고려될 때 $T_4 \rightarrow .. \rightarrow T_5$와 $T_4 \rightarrow .. \rightarrow T_3$의 직렬 순서는 존재하지 않는다. $T_4 \rightarrow .. \rightarrow T_5$의 직렬 순서는 $T_2 \rightarrow .. \rightarrow T_3$가 존재하지 않는 것과 같은 이유로 존재하지 않으며 $T_4 \rightarrow .. \rightarrow T_3$의 직렬 순서는 존재하지 않음이 위에서 증명되었다. 그러므로, 타임스탬프 구간이 가정에서처럼 $[TS(T_5), ∞]$가 되고 실행이 계속되어질 수 있다. 이와 같이, ts_i시점 이후 완료된 T_1과 관련 있는 어떤 갱신 거래일지라도 직렬화 그래프상 ts_i 시점까지 형성된 직렬 순서와 함께 순환을 만들지 않으므로 현재 수행중인 T_1을 포함한 갱신 거래들의 실행은 직렬가능하다. □

정리 4.2 이동 클라이언트에서 수행 중인 질의 거래 T_1에 의한 읽기 연산 시점 ts_i에서 타임스탬프 구간 값 중 최저 한계치가 최고 한계치보다 크거나 같지 않다면 질의 거래 T_1을 포함한 갱신 거래들의 실행은 ts_i시점까지 직렬가능하다.

증명: T_1을 포함한 실행은 ts_i까지 직렬가능하지 않다

고 가정하자. 이것은 T_1이 거래 시작과 ts_i사이 어느 시점에서 직렬화 그래프에 약한 일관성을 위배하는 순환을 포함하고 있다는 것을 의미한다. 즉, T_1전에 ts_i시점까지 완료된 갱신 거래를 순서대로 T_2, T_3라고 하면 $T_1 \rightarrow T_2 \rightarrow .. \rightarrow T_3 \rightarrow T_1$이 존재하여야 한다. 우선 $T_1 \rightarrow T_2$가 존재한다는 것은 T_1에 의해 읽혀지고 T_2에 의해 갱신된 공통의 데이타 항목이 존재한다는 것이다. T_1이 시작할 때 [0, ∞]의 타임스탬프 구간이 T_2가 완료되는 시점에서 [0, $TS(T_2)$]가 된다. 그 후 T_3가 완료되고 T_1의 읽기 연산 ts_i시점에서 $T_3 \rightarrow T_1$이 존재한다는 것은 T_3가 갱신한 데이타 항목이 T_1의 읽기 연산 대상이 된다는 것이므로 $LB(T_1)$의 값은 $TS(T_3)$가 된다. 또한, $T_2 \rightarrow .. \rightarrow T_3$가 존재한다는 것은 완료된 거래들을 모아놓은 *Total CommitList*에 있는 어느 거래의 읽기 연산 집합과 T_3의 쓰기 연산 집합 또는 *TotalCommitList*에 있는 어느 거래의 쓰기 연산 집합과 T_3의 읽기 연산 집합과의 사이에 공통된 데이타 항목이 존재하여 갱신 거래들과의 비직렬화 가능성 즉, 약한 일관성이 위배될 수 있으므로 정리 4.1에서처럼 $UB(T_1)$값이 ∞으로 바뀌지 않고 그대로 $TS(T_2)$가 되어 타임스탬프 구간은 [$TS(T_3)$, $TS(T_2)$]가 된다. $LB(T_1)$의 값인 $TS(T_3)$은 $UB(T_1)$의 값인 $TS(T_2)$보다 분명히 커서 철회될 것이므로 이것은 가정과 모순이다. 그러므로 현재 수행중인 질의 거래 T_1을 포함한 갱신 거래들의 실행은 ts_i시점까지 직렬가능하다. □

정리 4.3 (OCC/TI의 정확성): OCC/TI는 질의 거래들의 약한 일관성을 만족하는 수행을 보장한다.

증명: 완료하려는 이동 질의 거래를 T_n, 이미 완료된 갱신 거래들을 T_1, T_{n-1} 등이라 하고 우리는 n>1일 때 $T_n \rightarrow T_1 \rightarrow .. \rightarrow T_{n-1} \rightarrow T_n$이 OCC/TI에 의한 직렬화 그래프에 존재하지 않음을 보일 것이다. 먼저 직렬가능하지 않아 그러한 순환이 있다고 가정하자. 즉, T_n이 아직 완료된 상태는 아니고 완료하려는 시점 즉, 마지막 읽기 연산 시점에서 직렬화 그래프상 약한 일관성을 위배하는 순환이 존재한다는 것이다. 이것은 우리가 증명한 정리 4.2에 대하여 모순이다. 즉, 질의 거래는 읽기 연산으로만 이루어진 거래이므로 어느 읽기 연산 시점에서라도 정리 4.2에 의해 다른 갱신 거래들과의 직렬가능성 즉, 약한 일관성을 만족하므로 마지막 읽기 연산 시점 즉, 완료 시점까지 전역적 직렬화 그래프는 약한 일관성을 위배하는 순환을 포함하지 않음을 의미한다. 그러므로, OCC/TI는 약한 일관성을 만족하는 수행을 보장한다. □

5. 성능 평가

이 장에서는 모의 실험을 통해 본 논문에서 제안한 OCC/TI를 기존 기법인 BCC-TI의 성능과 비교하여 우리의 기법이 더 우수함을 보이고자 한다. OCC/TI가 방송 환경을 배경으로 하는 응용 시스템에서 질의 거래를 위한 적합한 동시성 제어 기법인지를 판단하기 위해서는 갱신 거래 도착 비율, 데이타 객체 크기와 다양한 부하 등의 변수를 이용하여 모의 실험을 통한 검증 작업이 필요하다.

5.1 모의 실험 환경

5.1.1 입력 매개 변수

모의 실험에 사용된 입력 매개변수는 시스템 자원을 사용하기 위한 매개변수, 시스템 부하 매개변수로 이루어진 시스템 매개변수와 질의 유형을 표현하기 위한 응용 매개변수들로 나누었으며 그 내용은 표 1과 표 2에서 보여준다. 표 2의 시스템 변수값들은 BCC-TI[20]과 유사하게 설정하였다.

표 1 시스템 매개변수의 값

매개변수	값	매개변수	값
서버데이타수	600	서버측갱신빈발데이타	200
클라이언트데이타수	300	클라이언트측접근빈발데이타	100
데이타크기	8000	갱신거래완료시간간격	100
질의거래발생간격	132	읽기연산간격	64
최소갱신거래크기	3	최대갱신거래크기	8
최소질의거래크기	3	최대질의거래크기	8
실행시간관련요소최소값	2.0	실행시간관련요소최대값	8.0
제어정보테이블방송주기	100	빈발데이타갱신가능성	70
데이타가 갱신되는 최대 횟수	5.0		

표 2 응용 매개변수 값

구분	입력 매개 변수	단위	변화하는 값
응용 매개변수	갱신 거래 도착 비율	trans/Mbittime	0 - 60 (10씩 증가)
	데이타 객체 크기	bits	8000 - 16000 (2000씩 증가)

클라이언트는 단지 서버가 방송한 데이타들의 부분집합만을 접근한다. 즉, 클라이언트가 접근하는 데이타집합은 서버데이타집합의 부분집합이다. 이것은 서버가 다른 클라이언트들에 대한 서비스도 제공한다는 것을 모델링한다. 전체 서버데이타집합 중 일정한 접근빈발가능성 값을 가진 접근빈발데이타들이 존재한다. 실험은 비트 시간 단위로 수행한다(1 비트 시간 = 한 비트를 방송하는 데 걸리는 시간). 우리는 서버가 선입선출 순서로 데이타들을 방송한다고 가정한다. 각 데이타는 일정한 비트의 크기를 갖는다. 일정한 킬로비트(Kbit) 시간마다 서

버에서 갱신 거래가 완료된다. 갱신 거래 크기는 그것의 최소값과 최대값 사이에 균등하게 분산되어 있다. 갱신은 데이타 값을 일정한 횟수 내에서 변화시킬 수 있다. 서버는 방송주기마다 데이타들을 먼저 방송한 후에 제어 정보테이블을 방송한다. 클라이언트에서 자주 접근되는 데이타들이 정해진다. 그러한 데이타들은 일정한 값의 읽기 연산의 가능성을 갖는다. 서버와 클라이언트에서 자주 접근되는 데이타들을 맞추려는 특별한 시도는 하지 않는다. 질의 거래는 일정한 킬로비트 시간 단위당 발생하며 일정한 킬로비트 시간 단위당 읽기 연산을 한다. 질의 거래 크기는 최소질의크기에서 최대질의크기 사이에 균등하게 분포한다. 최소크기를 가진 질의거래일지라도 하나 이상의 방송구간동안 수행하도록 설정한다. 질의 거래는 완료하기 전에 최신의 데이타를 읽지 않음이 발견되면 철회된다. 철회된 질의 거래는 그것이 수행마감시간을 놓쳤을지라도 재시작하여 완료될 때까지 계속한다. 우리는 재시작을 새로운 질의 거래를 시작시킴으로써 실험한다. 수행마감시간은 현재 시간 + 실행시간을 늦추는 요소 * 예상된 실행 시간이다. 실행시간을 늦추는 요소값은 그것의 최소값과 최대값 사이에서 균등하게 분포한다. 성능 측정치는 평균 질의 응답 시간, 재시작된 질의 비율, 수행마감시간을 초과한 질의 비율이다.

성능에 가장 큰 영향을 주는 것은 서버로부터 완료된 갱신 거래 도착 비율과 데이타 객체 크기 증가이다. 서버로부터 완료 후 방송되는 갱신 거래가 많을수록 BCC-TI에서는 질의 거래 철회율이 높아지고 그 거래를 재시작시킴으로써 평균 응답 시간이 느려진다. 그에 반해 OCC/TI는 질의 거래에 대한 검증 기준으로 약한 일관성을 채택함으로써 완료된 갱신 거래에 덜 민감하다. 따라서, 갱신 거래 도착 비율이 성능 평가의 주요 요인이 된다. 데이타 객체 크기의 경우 크기가 증가함에 따라 방송 주기 길이는 증가한다. 그것은 질의 수행 시간을 길게 하고 무효화 방송에 의한 거래 철회 비율이 달라질 수 있다. 따라서, 질의 거래를 위한 동시성 제어 기법의 평균 응답 시간 등의 전체 성능에 막대한 영향을 줄 수 있는 성능 지표가 된다.

5.1.2 성능 지수

우리의 모의 실험에서 사용하는 주요 성능 지수는 질의 거래당 완료하는 데 걸리는 시간인 평균 응답 시간이다. 실험 결과 분석에 도움을 주는 다른 성능 지수는 철회 비율, 수행마감시간 미엄수 비율 등이다. 응답시간은 한 클라이언트가 거래를 발생하여 그 거래가 성공적으로 마칠 때까지의 시간으로 측정되어지고 철회 후 재실행에 드는 시간까지를 포함한다. 또한 거래의 철회 비율이 조사되어지는데 그것은 어느 특정 기간에 클라이언트에서 처리되는 모든 질의 거래들에 대한 철회된 질의 거래의 백분율로 측정된다. 그것을 위해 우리는 주기적으로 거래 상태 변화를 조사한다. 철회된 거래는 정해진 수행마감시간을 초과할 확률이 높기 때문에 수행마감시간 미엄수 비율은 새로운 기법과 기존의 기법 중 어느 것이 거래 재시작 횟수를 줄이는 효과가 있는지를 증명하는 성능 지수로 사용된다.

5.2 실험 결과

5.2.1 실험 1: BCC-TI와 비교된 OCC/TI

OCC/TI는 직렬화를 정확성 검증 기준으로 하는 BCC-TI와 다르게 약한 일관성을 질의 거래의 정확성 검증 기준으로 채택했다. 약한 일관성은 3장에서 언급한 것처럼 해당 질의 거래가 갱신 거래들과의 직렬화 순서를 유지하되 동시에 수행 중인 다른 질의 거래들은 그것과 같은 순서를 반드시 유지할 필요가 없으므로 직렬화 즉, 엄격한 일관성을 기준으로 사용하는 것보다 질의 거래의 철회율이 더 낮다. 철회가 준다는 것은 질의 거래의 재시작 횟수가 줄기 때문에 평균 응답 시간이 줄어든다.

그림 4.a는 서버로부터 완료된 갱신 거래 도착 비율에 따른 철회된 질의 거래의 비율을 보여준다. 최소 질의 거래일지라도 하나 이상의 방송 주기를 거치도록 설정했기 때문에 그 해당 방송 주기 동안 갱신 거래가 더 자주 나타남에 따라 BCC-TI의 철회율은 가파르게 상승한다. 그에 비해 OCC/TI는 천천히 증가한다. 증가된 철회의 결과로 BCC-TI의 평균 응답 시간이 OCC/TI보다 더 빠르게 증가한다(그림 4.b). 이 실험에서 사용된 갱신 거래 도착시간은 25 킬로비트 시간 단위이다. OCC/TI는 BCC-TI보다 질의 거래에 대해 더 느슨한 일관성 검사를 적용하므로 즉, 직렬화 그래프상 갱신 거래들만으로 이루어진 순환(cycle)과 하나의 질의 거래와 갱신 거래들로 이루어진 순환은 허용이 안되고 그 외 사이클은 허용함으로써 그것의 응답 시간이 더 느리게 증가한다. OCC/TI에 의한 철회율은 BCC-TI에 비해 약 44% 감소하고 그 결과 OCC/TI의 응답 시간은 BCC-TI에 비해 약 21.6%까지 향상한다. 또한, 그림 4.c는 질의 거래 응답 시간의 감소로 인해 수행마감시간 미엄수(miss) 비율이 낮아짐을 나타낸다.

5.2.2 실험 2: 데이타 객체 크기의 영향

데이타 객체 크기가 증가함에 따라 방송 주기 길이는 증가한다. 그러므로 한번의 읽기 연산을 위한 평균 대기 시간은 더 길어지고 그것은 질의 수행 시간을 길게 한다. 질의 거래 수행 시간이 길어질수록 데이타 충돌의 가능성은 높아짐을 암시하고 그럼으로써 철회율이 더

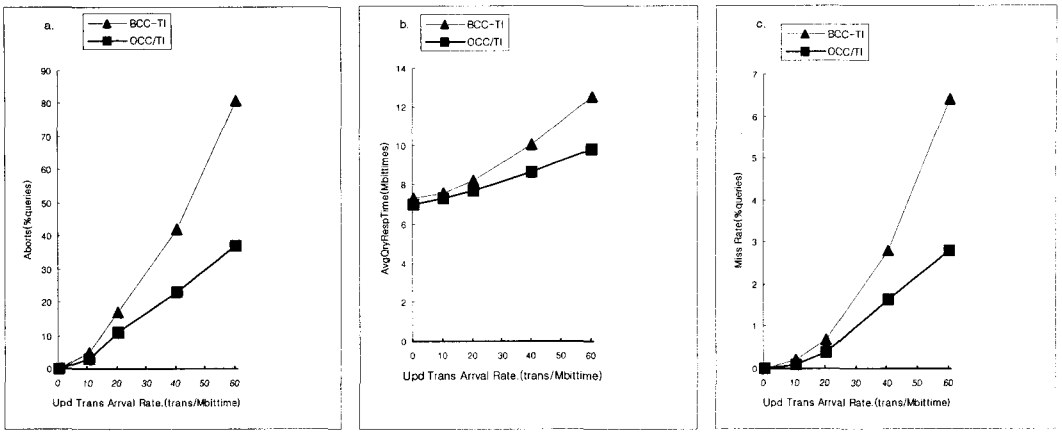


그림 4 BCC-TI와 OCC/TI에 대한 갱신 거래 도착 비율의 영향

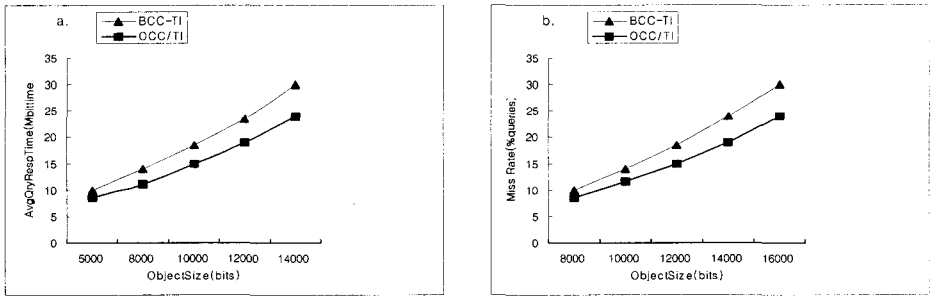


그림 5 데이타 객체 크기의 영향

높아진다. 그러므로, OCC/TI와 BCC-TI에 있어서 질의 거래 평균 응답 시간은 데이타 객체 크기와 함께 증가한다. 질의 거래가 한번 이상 철회되면 정해진 시간 안에 완료되기 어렵기 때문에 두 기법의 수행마감시간 미엄수 비율도 또한 증가한다. OCC/TI가 BCC-TI보다 데이타 객체 크기에 덜 민감하게 반응한다. 그것은 OCC/TI가 BCC-TI보다 철회율과 재시작율이 더 낮기 때문이다. 그림 5.a에서는 평균 응답 시간의 관점에서 그리고 그림 5.b에서는 수행마감시간 미엄수 비율의 관점에서 여러 데이타 객체 크기에 대한 두 기법의 성능을 보여준다. OCC/TI와 BCC-TI의 평균 응답 시간의 차이는 객체 크기가 작을 때는 약 10.7%이며 더 큰 객체 크기에서는 약 17%까지 차이가 난다.

6. 결론 및 향후 연구

본 논문에서는 방송환경에서 이동 클라이언트/서버 시스템을 위한 효율적인 동시성 제어 기법인 OCC/TI를 제안했다. 일관성과 현재성을 동시에 요구하는[23] 주식 거래 또는 교통 정보 시스템과 같은 방송 기반 데이타베이스시스템의 여러 응용들이 있다. 본 논문은 이러한 요구사항에 부응하기 위한 효율적인 동시성 제어 기법을 제시하였다. 방송환경의 특수성을 고려한 OCC/TI는 기존의 이동 컴퓨팅 환경을 위한 동시성 제어 기법과 비교될 때 다음과 같은 장점을 가지고 있다.

첫째, OCC/TI는 서버에 의해 관리 유지되고 클라이언트에 의해 읽혀지는 데이타의 상호 일관성과 클라이언트에 의해 읽혀지는 데이타의 현재성을 만족시키기에 가장 적절한 정확성 검증 기준인 약한 일관성을 채택하였다. 서론에서 언급했듯이 방송 환경에서 대부분을 차지하는 질의 거래의 정확성 검증 기준으로 직렬화를 채택하는 것은 매우 비용이 비싸고 불필요한 철회를 야기시킨다. 이 기법은 그러한 단점을 극복할 수 있도록 질의 거래의 일관성을 검증하는 기준으로 약한 일관성을 채택했으며 그것을 효율적으로 실행할 수 있는 타임스

탬프 구간 기법을 적용하였다. 그럼으로써 동시에 수행중인 갱신 거래들에 대한 질의 거래의 직렬 순서를 유동적으로 조정할 수 있다는 장점을 살려 완료된 갱신 거래들이 항상 직렬화 순서에서 현재 수행중인 모든 질의 거래들보다 앞선다는 암시적인 가정 때문에 불필요한 철회가 발생하는 기존의 낙관적 동시성 제어 기법을 보완하였다. 즉, 질의 거래가 완료된 갱신 거래들을 직렬화 순서에서 앞설 수 있는 점을 수용하였고 그와 동시에 OCC/TI는 기존의 타임스탬프 구간 기법과는 달리 질의 거래가 여러 방송주기에 거쳐 일어날 경우 수행중인 또다른 갱신 거래들이 완료된 후 직렬화 순서에서 해당 질의 거래보다 또한 동시에 앞설 수 있다는 점을 해결하였다. 다시 말해, 기존 기법에서 질의 거래가 완료한 갱신 거래들을 앞설 때 그 질의 거래가 다음 읽기 연산시에 다른 완료한 갱신 거래들에 의해 갱신된 데이타 항목과 동일한 항목을 읽는다면 직렬화를 만족하지 않을 확률이 많다라는 가정하에 일어난 불필요한 철회 및 재시작의 횟수를 줄임으로써 성능향상을 도모하였다.

둘째, OCC/TI는 클라이언트가 서버쪽으로 정보를 요구하는 메시지 수를 최대한 줄임으로써 비대칭적 대역폭을 가진 방송환경의 특수성을 효율적으로 잘 이용하였다. 질의 거래들이 윗방향 통신없이 클라이언트 자체 내에서 지역적으로 완료 처리됨으로써 서버에게 완료 요구를 위해 정보를 보내는 기회를 줄여 상대적으로 작은 대역폭을 가진 또한 한 클라이언트에게 허락되어지는 대역폭 사용 시간이 짧은 방송환경을 잘 극복할 수 있도록 하였다.

앞으로의 연구에서는 방송 환경에서 이동 클라이언트 내의 질의 거래들뿐만 아니라 갱신 거래들을 좀 더 효율적으로 처리할 수 있는 최적화 알고리즘을 캐슁 기법과 함께 연구할 것이다.

참 고 문 헌

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

[2] M.Franklin and S.Zdonik, "Data In Your Face": Push Technology in Prospective", in *Proceedings of 1998 ACM SIGMOD Conference*, Seattle, 1998.

[3] Pitoura, E. and Bhargava, B. "Dealing with Mobility: Issues and Research Challenges," *Technical Report,* Purdue Univ., Nov. 1993.

[4] T. Imielinski and B.R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management," *Communications of the ACM*, vol. 37, no. 10, Oct. 1994.

[5] Pitoura, E. and Bhargava, B. "Maintaing Consistency of Data in Mobile Distributed Environment," in *Proceeding of the 15th International Conference on Distributed Computing Systems,* pp. 404-413, 1995

[6] Xuan, P., O. Gonzalez, J. Fernandez & Ramamritham, K., "Broadcast on Demand: Efficient and Timely Dissemination of Data in Mobile Environments", in *Proceedings of 3rd IEEE Real-Time Technology Application Symposium,* 1997.

[7] J.Shanmugasundaram, A.Nithrakashyap, R.Sivasankaran, and K. Ramamritham, "Efficient Concurrency Control for Broadcast Environments," *ACM SIGMOD,* 1999

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

[9] P.M. Bober and M.J. Carey. "Multiversion Query Locking," *Proceedings of the VLDB Conference,* Vancouver, Canada, August 1992

[10] W. Weihl, "Distributed Version Management for Read-Only Actions," *IEEE Transactions on Software Engineering*, 13(1), January 1987

[11] S.Acharya, R.Alonso. M.Franlikn and S.Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments," *Proceedings of ACM SIGMOD Conference on Management of Data*, May 1997.

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

[13] D.Barbara and T.Imielinsky, "Sleepers and Workholics: Caching in Mobile Environment," *Proceedings of ACM SIGMOD Conference on Management of Data*, pp. 1-12, June 1994.

[14] J.Jing, A.Elmagarmid, A.Helal 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.

[15] K.L.Wu, P.S.Yu and M.S.Chen, "Energy-efficient Caching for Wireless Mobile Computing," *Proceedings of the 12th International Conference on Data Engineering*, pp. 336-343, Feb. 1996.

[16] C.F.Fong, C.S.Lui and M.H.Wong, "Quantifying Complexity and Performance Gains of Distributed Caching in a Wireless Network Environment," *Proceedings of the 13th International Conference on Data Engineering*, pp. 104-113, April 1997

[17] E.Pitoura, "Supporting Read-Only Transactions in

Wireless Broadcasting," *Proceedings of the 9th International Workshop on Database and Expert Systems Applications*, pp. 428-433, 1998

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

[19] G.Herman, et. al, "The Datacycle Architecture for Very High Throughput Database Systems", *Proceedings of the ACM SIGMOD Conference,* 1987

[20] 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

[21] R. Srinivasa, Sang H. Son, "Quasi-consistency and Caching with Broadcast Disks," *Proc. Int'l Conf. on Mobile Data Management,* Hong Kong, Jan. 2001, pp. 133-144.

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

[23] P. Xuan, et. al, "Broadcast on Demand-Efficient and Timely Dissemination of Data in Mobile Environments", *IEEE Real-Time Technology and Applications Symposium,* June 1997, pp.38-48.

이 욱 현

1992년 이화여자대학교 전자계산학과(이학사). 1997년 한국과학기술원 정보및통신공학과(공학석사). 2000년 전남대학교 전산학과 박사 수료. 2002년 ~ 인하대학교 강의전임강사. 관심분야는 분산데이타베이스, 이동컴퓨팅, 객체지향시스템, 데이타 마이닝 등

황 부 현

1978년 숭실대학교 전산학과(학사). 1980년 한국과학기술원 전산학과(석사). 1994년 한국과학기술원 전산학과(박사). 1980년 ~ 현재 전남대학교 컴퓨터정보학부 교수. 2001년 ~ 현재 전남대학교 정보통신연구소장. 관심분야는 분산시스템, 분산데이타베이스, 객체지향시스템, 전자상거래