이동 방송 환경에서 최신 현행성을 보장하는 병행성 제어 기법

(A Concurrency Control Mechanism Ensuring Latest-Bound Currency in Mobile Broadcasting Environments)

한 부 형 † 정 성 권 †† 조 유 근 †††
(Boohyung Han) (Sung-Kwon Chung) (Yookun Cho)

요 약 이동 방송 환경(mobile broadcasting environment)은 정보를 가지고 있는 서버가 다수의 이동 클라이언트에게 모든 데이타를 주기적으로 방송하면 클라이언트가 자신이 원하는 데이타가 방송 채널에 나타날 때 이를 검색하는 환경을 말한다. 이 때 클라이언트가 수행하는 읽기 전용 트랜잭션은 일관적이면서도 가장 최근에 갱신된 데이타를 필요로 한다. 이동 방송 환경에 적합하도록 연구된 기존의 병행성 제어 기법들은 클라이언트가 일관적인 데이타를 읽는 것은 보장하지만, 가장 최근에 갱신된 데이타를 읽을 수 있게 하는 최신 현행성(latest-bound currency)은 보장하지 못한다. 본 논문에서는 기존 기법들과 동일한 수준의 일관성뿐만 아니라 최신 현행성을 보장하는 효율적인 병행성 제어 기법을 제안하고 이를 검증한다. 아울러 제안된 기법의 성능을 평가하기 위하여 최신이 아닌 데이타를 읽은 경우(stale read)의 수와 트랜잭션 중단(abort)의 수를 각각 측정하였다. 시뮬레이션을 통한 실험 결과, 본 논문에서 제안하는 기법은 기존의 방법과 비교할 때 항상 최신의 데이타를 읽으면서 트랜잭션 중단의 수를 감소시킨다는 것을 알 수 있었다.

Abstract In mobile broadcasting environments, an information server periodically broadcasts all data items to a large mobile client population and mobile clients retrieve data items they need when the data items arrive on the broadcast channel. The read-only transactions executed at a client require the data to be consistent and current. Previous concurrency control mechanisms designed for the mobile broadcasting environments ensure that the clients read consistent data, but cannot ensure latest-bound currency that they read the most up-to-date data. In this paper, we propose an efficient concurrency control scheme that ensures the latest-bound currency as well as the consistency comparable to that of the previous mechanisms. To evaluate the performance of the proposed mechanism, we have done simulation experiments and measured the number of stale reads and transaction aborts. The result shows that the proposed mechanism produces no stale reads at all and reduces the number of transaction aborts compared with the previous mechanism.

1. 서 론

근래에 들어오면서 이동 통신 기술이 빠르게 발전함에 따라 이동 컴퓨터를 사용하여 사용자가 원하는 정보를 언제 어느 때라도 접근하는 것이 가능하게 되었다. 이러한 이동 컴퓨팅 환경에서 사용자에게 정보를 제공하는 데는 두 가지 방법이 존재한다[1][2][3]. 첫 번째 방법은 요청 모드(on-demand mode)로서, 이동 클라이언트가 원하는 데이타를 서버에게 요청하면 서버가 그 데이타를 전송하여 응답하는 방식이다. 이 방법은 간단하다는 장점을 가지지만, 이동 클라이언트의 수가 많아지게 되면 확장성(scalability)이 떨어진다는 단점을 가진다. 게다가 이동 클라이언트는 여러 가지 자원이 제한되어 있기 때

† 학생회원 : 서울대학교 컴퓨터공학과
bhhan@ssrnet.snu.ac.kr
†† 비 회 원 : 유비쿼스(주) 대표이사
sung@ubiquix.com
††† 종신회원 : 서울대학교 컴퓨터공학과 교수
cho@ssrnet.snu.ac.kr
논문접수 : 2000년 1월 4일
심사완료 : 2001년 4월 17일

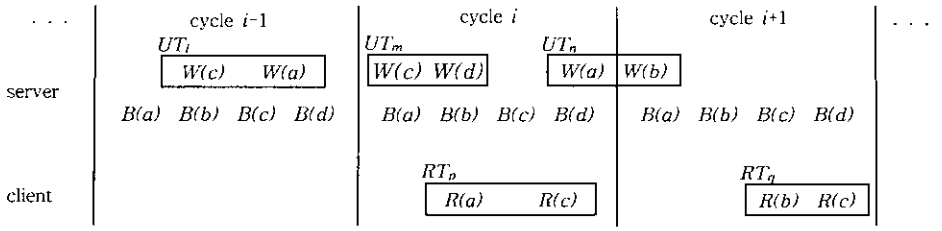


그림 1 이동 방송 환경

문에 자신의 요청을 서버로 보내는 데 비용이 많이 든다. 두 번째 방법은 방송 모드(broad- casting mode)로서, 클라이언트의 특별한 요청 없이 서버가 다수의 클라이언트에게 모든 데이타를 매 방송 주기(broadcast cycle)마다 주기적으로 방송하는 방식이다. 클라이언트는 방송을 계속 관찰하다가 자신이 원하는 데이타가 방송 채널에 나타나면 이를 검색한다. 이렇게 하면 클라이언트의 수에 관계없이 데이타를 전달하는 비용이 일정하게 든다는 장점을 가진다. 방송에 기반한 응용은 여러 가지가 있는데, 주식 거래, 스포츠 관람 티켓, 전자 신문, 메일 검색, 교통 관리 시스템 등이 그 예이다[3].

이러한 이동 방송 환경에서 클라이언트는 필요한 정보를 검색하기 위하여 읽기 전용 트랜잭션(read-only transaction)을 수행하고, 서버는 갱신 트랜잭션(update transaction)을 통해 자신의 데이타베이스의 내용을 계속적으로 갱신한다. 읽기 전용 트랜잭션은 자신이 읽는 데이타와 서버의 갱신 트랜잭션이 갱신하는 데이타가 일관적이어야 한다는 요구조건과 자신이 읽은 데이타가 갱신된 시점이 클라이언트의 제약 조건을 만족해야 한다는 요구조건을 가진다. 전자의 조건을 일관성(consistency)이라 하고 후자의 조건을 현행성(currency)이라 한다[4]. 이 두 가지 요구조건을 보장하기 위해서는 2단계 잠금 기법(two phase locking), 낙관적 방법(optimistic method), 타임스탬프(timestamp) 등과 같은 병행성 제어(concurrency control) 기법이 사용되어 왔다. 그러나 이러한 전통적인 기법들은 이동 방송 환경에서는 적합하지 않다. 왜냐하면 이동 방송 환경에서는 통신이 불안정하고, 이동 클라이언트의 자원이 제한적이고, 서버와 클라이언트간의 통신 대역폭이 비대칭적이기 때문이다. 이러한 이유 때문에 Pitoura와 Lam 등은 각각 이동 방송 환경에 적합한 병행성 제어 기법을 제안하고 실험하였다[3][5]. 그러나, 이들은 엄격한 일관성 요구조건을 채택하였기 때문에 클라이언트가 일관적인 데이타를 읽었음에도 불구하고 트랜잭션이 중단(abort)되는 경우가 발생할 수도 있다는 단점을 가진다. 이에 반해 Shanmugasundaram 등은 이보다 완화된 일관성 요구조건인 갱신 일관성(update consistency)을 채택하여 클라이언트가 수행하는 읽기 전용 트랜잭션이 불필요하게 중단되는 경우를 제거하였다[6]. 그러나 이 기법에서 서버는 매 방송 주기마다 클라이언트가 일관성 검사를 하기 위한 정보를 방송해야 하는데, 이를 위해서는 현재의 방송 주기가 시작하기 전에 이미 갱신된 데이타 값만을 방송해야 하는 제약을 가진다. 따라서 클라이언트는 가장 최근에 갱신된 값을 읽어야 하는 최신 현행성(latest-bound) 요구조건을 만족하지 못하는 경우가 발생할 수도 있다. 특히 경매나 주식 거래와 같은 응용에서는 정확한 결과를 위해서 최신의 데이타 값이 필요하기 때문에 현행성 요구 조건은 매우 중요하다고 할 수 있다. 본 논문에서는 Shanmugasundaram의 기법을 개선하여 갱신 일관성뿐만 아니라 최신 현행성 요구조건을 모두 만족하는 병행성 제어 기법을 제안하고자 한다.

본 논문은 다음과 같이 구성된다. 2절에서는 본 논문의 배경으로서 이동 방송 환경과 이동 방송 환경에서 필요로 하는 일관성 및 현행성 요구조건을 설명한다. 3절에서는 이동 방송 환경에 적합하도록 제안된 이전의 병행성 제어 기법들의 문제점을 살펴본다. 4절에서는 본 논문에서 새로이 제안하는 병행성 제어 기법을 자세히 서술하고, 5절에서는 4절에서 제안한 알고리즘을 검증한다. 6절에서는 시뮬레이션에 기반한 실험 결과를 보이고, 마지막으로 7절에서 결론을 맺는다.

2. 배경

2.1 이동 방송 환경

그림 1은 네 개의 데이타 a, b, c, d를 가지고 있는 간단한 이동 방송 환경의 예를 보인다. UT_i, UT_m, UT_n은 서버에서 수행되는 갱신 트랜잭션을 나타내고, RT_p와 RT_q는 클라이언트에서 수행되는 읽기 전용 트랜잭션을 나타낸다. 또한 $R(a)$는 데이타 a에 대한 읽기를, $W(a)$는 데이타 a에 대한 쓰기를, $B(a)$는 데이타 a에 대한 방송을 각각 나타낸다. 이 예에서 서버는 세 개의

갱신 트랜잭션을 수행하며 주기적으로 네 개의 데이타를 클라이언트로 방송하며, 클라이언트는 두 개의 읽기 전용 트랜잭션을 수행하면서 자신이 원하는 데이타가 방송되면 이를 읽는다.

2.2 일관성 요구조건

[7]에서는 읽기 전용 트랜잭션에서 고려되는 네 개의 일관성을 정의하였다. 이들은 완전 일관성(strict consistency), 강 일관성(strong consistency), 약 일관성(weak consistency), 갱신 일관성(update consis tency)으로서, 완전 일관성이 가장 엄격한 일관성이고 갱신 일관성이 가장 완화된 일관성이다. 완전 일관성에서는 모든 읽기 전용 트랜잭션이 보는 갱신 트랜잭션의 직렬 순서(serial order)가 이들이 실제 종료(commit)된 순서와 일치해야 하며, 강 일관성에서는 갱신 트랜잭션의 직렬 순서가 실제 종료된 순서와 일치하지 않더라도 모든 읽기 전용 트랜잭션이 동일한 직렬 순서를 볼 수만 있으면 된다. 약 일관성에서는 각 읽기 전용 트랜잭션이 보는 갱신 트랜잭션의 직렬 순서가 서로 다르더라도 각자가 갱신 트랜잭션의 일관된 직렬 순서만 볼 수 있으면 되며, 갱신 일관성에서는 이를 더 완화시켜 각 읽기 전용 트랜잭션이 자신이 직접적으로나 간접적으로 읽은 데이타 값을 갱신한 트랜잭션에 대해서만 일관된 직렬 순서를 볼 수 있으면 된다. 이 중 갱신 일관성을 제외한 나머지 일관성들은 이동 방송 환경에 적용하기에는 여러 가지 단점을 가지고 있다고 알려져 있다[6]. 첫째, 2 단계 잠금 기법과 같은 예에서는 일관성을 유지하기 위해서 서버와 클라이언트간에 과다한 통신을 필요로 한다. 그러나, 이동 방송 환경에서는 클라이언트로부터 서버로의 통신 채널이 없거나 대역폭이 제한되어 있기 때문에 일관성 유지에 필요한 많은 양방향 통신을 하기가 불가능하거나 비용이 매우 크다. 둘째, 이 일관성들을 그대로 적용하면 병행성 제어 프로토콜이 너무 엄격해져서 불필요한 트랜잭션의 중단(abort)이 발생하게 된다. 갱신 일관성은 이러한 일관성들을 완화시켜서 위에서 설명한 두 가지 단점을 피할 수 있게 한다.

그림 2는 일관성을 설명하기 위한 스케줄과 직렬화 그래프(serialization graph)의 예를 보여준다. 그림 2에서 서버는 세 개의 갱신 트랜잭션 UT_1, UT_2, UT_3을 수행하고, 임의의 클라이언트가 두 개의 읽기 전용 트랜잭션 RT_1과 RT_2을, 다른 클라이언트가 하나의 읽기 전용 트랜잭션 RT_2를 수행한다. 총 6개의 트랜잭션이 그림 2의 스케줄처럼 세 개의 데이타 a, b, c에 대하여 읽기와 쓰기 연산을 수행한다. 이 스케줄에서 세 개의 갱신 트랜잭션이 실제 종료된 순서는 UT_1, UT_2, UT_3이

트랜잭션	시간 →
UT_1	$R(a)W(a)$
UT_2	$R(c)R(b)W(b)$
UT_3	$R(c)W(c)$
RT_1	$R(a)$ $R(b)$
RT_2	$R(b)$ $R(a)$
RT_3	$R(b)$ $R(c)$

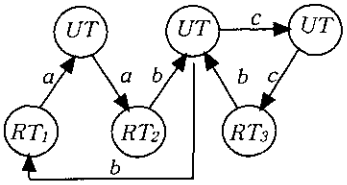


그림 2 스케줄 및 직렬화 그래프의 예

다. 그러나, RT_1의 입장에서 보면 자신이 데이타 a를 읽은 후 UT_1이 이를 갱신하였고 UT_2가 갱신한 데이타 b를 자신이 읽었기 때문에 UT_2, RT_1, UT_1이라는 순서로 수행되었다고 인식하게 된다. 그러므로, RT_1이 보는 UT_1과 UT_2의 직렬 순서가 실제 종료된 순서와 다르기 때문에 이 스케줄은 완전 일관성을 만족하지 못한다. 한편 RT_2의 입장에서 보면, 자신이 데이타 b를 읽은 후 UT_2가 이를 갱신하였고 UT_1이 갱신한 데이타 a를 자신이 읽었기 때문에 UT_1, RT_2, UT_2라는 순서로 수행되었다고 인식하게 된다. 따라서 이 스케줄에서 RT_1과 RT_2가 보는 갱신 트랜잭션의 직렬 순서가 서로 다르기 때문에 강 일관성도 만족하지 못한다. 그러나, 비록 RT_1과 RT_2가 UT_1과 UT_2의 서로 다른 직렬 순서를 보더라도 UT_1과 UT_2가 서로 전혀 영향을 주지 않기 때문에 이 읽기 전용 트랜잭션들은 여전히 트랜잭션에 일관적인(하나의 갱신 트랜잭션을 수행한 결과가 전혀 반영되지 않거나 전부 반영된) 데이타 값을 읽을 수 있다. 그러므로 RT_1, RT_2, UT_1, UT_2의 스케줄만 고려하면, RT_1과 RT_2가 각각 UT_1과 UT_2의 일관된 직렬 순서를 볼 수 있기 때문에 약 일관성을 만족한다. 반면 RT_3의 입장에서 보면, 자신이 데이타 b를 읽은 다음 UT_2가 이를 갱신하였고 UT_2가 읽은 데이타 c를 UT_3이 갱신하였고 다시 RT_3이 데이타 c를 읽었기 때문에 RT_3은 UT_2, UT_3이라는 직렬 순서와 UT_3, UT_2라는 직렬 순서를 모두 인식하게 된다. 따라서, RT_3이 일관된 직렬 순서를 보지 못하기 때문에 전체 스케줄은 약 일관성을 만족하지 못한다. 이를 직렬화 그래프로 보면 더욱 명백해지는데, 직렬화 그래프는 각 트랜잭션을 의미하는 노드들과 트랜잭션간의 의존성(dependency)을 간선으로

나타내는 그래프이다. 임의의 노드 T_1로부터 노드 T_2로의 간선이 존재하면 이는 T_1이 읽거나 갱신한 데이타를 T_2가 읽거나 갱신하였다는 것을 의미하는데, 두 트랜잭션의 연산 중 적어도 하나는 갱신이어야 한다. 간선에 표시되는 레이블은 두 트랜잭션이 의존하는 데이타 항목을 나타낸다. 완전 일관성과 강 일관성을 만족하려면 직렬화 그래프가 어떤 형태의 순환(cycle)도 포함하지 않아야 하고, 약 일관성에서는 여러 개의 읽기 전용 트랜잭션과 한 개 이상의 갱신 트랜잭션을 포함하는 순환(당연히 갱신 트랜잭션간에는 간선이 전혀 없어야 한다)은 허용되지만 한 개의 읽기 전용 트랜잭션과 한 개 이상의 갱신 트랜잭션을 포함하는 순환은 허용되지 않는다. 그림 2의 직렬화 그래프는 UT_1, RT_2, UT_2, RT_1의 순환과 UT_2, UT_3, RT_3의 순환을 가지고 있기 때문에 완전 일관성과 강 일관성을 모두 만족하지 못한다. 또한 UT_1, RT_2, UT_2, RT_1의 순환은 약 일관성에서 허용할 수 있지만 UT_2, UT_3, RT_3의 순환은 한 개의 읽기 전용 트랜잭션과 두 개의 갱신 트랜잭션을 포함하기 때문에 약 일관성을 만족하지 못한다.

그러나, RT_3이 UT_3의 수행 결과를 모두 볼 수 있지만 UT_2의 수행 결과는 전혀 보지 못하기 때문에 여전히 트랜잭션에 일관적인 데이타 값을 읽을 수 있다(이 때 UT_2는 UT_3의 수행에 전혀 영향을 주지 않는다). 그러므로, RT_3은 자신이 읽은 데이타를 갱신한 유일한 트랜잭션인 UT_3의 일관된 직렬 순서를 인식할 수 있기 때문에 갱신 일관성을 만족한다. 갱신 일관성은 직렬화 그래프에서 한 개의 읽기 전용 트랜잭션과 한 개 이상의 갱신 트랜잭션을 포함하는 순환 중에서 임의의 두 갱신 트랜잭션이 읽기-쓰기 간선(read-write edge)으로 연결되어 있으면 이를 허용한다. 읽기-쓰기 간선은 간선의 시작 노드 T_1이 가리키는 트랜잭션이 읽은 데이타를 도착 노드가 가리키는 트랜잭션 T_2가 갱신하였을 때 표현되는 간선이다. 이는 T_1이 T_2의 수행에 전혀 영향을 주지 않기 때문이다. 그림 2의 직렬화 그래프에서는 UT_2와 UT_3이 읽기-쓰기 간선으로 연결되어 있기 때문에 갱신 일관성을 만족한다. 갱신 일관성을 사용하면 클라이언트는 다른 클라이언트가 수행한 트랜잭션 스케줄을 전혀 알 필요가 없고 단지 서버와 자신이 수행한 트랜잭션 스케줄만 고려하여 일관성 검사를 할 수 있다. 그러므로, 갱신 일관성은 클라이언트가 다른 클라이언트와 통신하지 않아도 되기 때문에 이동 방송 환경에 적합하다[6].

2.3 현행성 요구조건

[4]에서는 읽기 전용 트랜잭션의 현행성 요구조건으로서 t-이전(t-vintage)과 t-이후(t-bound) 요구조건을 정의하였다. t-이전 요구조건에서는 읽기 전용 트랜잭션이 t 시간 전에 갱신된 값을 읽을 수는 있지만 t 이후에 갱신된 값은 읽을 수 없다. 반면 t-이후 요구조건에서는 t 시간 이전에 갱신된 값을 읽는 것을 최소로 보장하면서 t 시간 이후에 갱신된 값을 읽을 수도 있게 한다. 특히, t가 현재의 시간이면 읽기 전용 트랜잭션은 최신 또는 가장 최근에 갱신된 값을 요구한다. 이러한 요구조건을 최신(latest-bound) 현행성이라 한다.

예를 들어, 2.1절에서 설명하였던 그림 1에서 서버가 매 방송 주기마다 그 주기가 시작하기 전에 갱신된 데이타 값을 방송한다고 가정하면, RT_p와 RT_q는 모두 t-이전 현행성을 만족하는 읽기 전용 트랜잭션(t는 각 방송 주기의 시작 시간)이 된다. 이 경우 데이타 c는 RT_p가 읽기 전에 UT_m에 의해 갱신되었지만 방송 주기 i에서 방송되지는 않기 때문에, RT_p가 읽은 데이타 c의 값은 UT_l에 의해 갱신된 값이다. 이러한 읽기를 실효된 읽기(stale read)라 한다. 마찬가지로 RT_q의 읽기 연산 $R(b)$도 실효된 읽기이다. 반면에 서버가 항상 가장 최근에 갱신된 값만을 방송할 수 있다면, RT_p와 RT_q는 최신 현행성을 만족하는 트랜잭션으로서 실효된 읽기를 전혀 수행하지 않는다. 일반적으로 과거 어느 시점의 데이타를 읽어도 무방한 경우에는 t-이전 현행성을 요구하지만, 경매나 주식 거래와 같은 다수의 이동 방송 응용에서는 실효된 읽기가 클라이언트에게 정확한 결과를 줄 수 없기 때문에 최신 현행성을 요구한다. 본 논문에서 제안하는 병행성 제어 기법은 클라이언트가 항상 최신의 데이타를 읽도록 하는 것이다.

3. 이전 연구들

Pitoura는 읽기 전용 트랜잭션의 일관성 및 현행성을 보장하는 여러 가지 기법을 제안하였다[3]. 하나는 각 데이타 항목마다 여러 가지 버전을 모두 방송하는 것이고, 다른 하나는 트랜잭션의 직렬 수행을 보장하기 위하여 직렬화 그래프와 무효화 보고(invalidation report)를 함께 방송하는 것이다. 그러나 이 기법들은 클라이언트가 계속 방송을 감시하고 있어야 한다는 단점을 가진다. 만약 클라이언트가 어떤 이유로 몇 개의 방송 주기를 놓친다면, 더 이상 읽기 전용 트랜잭션의 일관성을 보장할 수 없다. 또한 이 기법들은 강 일관성을 채택하였기 때문에 갱신 일관성과 비교하였을 때 불필요한 트랜잭션의 중단이 더 발생할 수도 있다. 아울러 이 기법들에서 방송되는 모든 데이타 값은 그 방송 주기가 시작하기 전에 갱신된 값이기 때문에, t-이전 현행성밖에 보장하지 못한다.

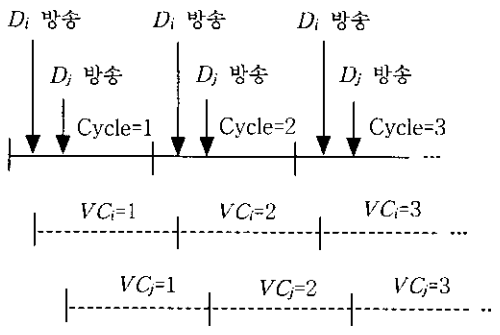


그림 3 가상 방송 주기의 예

Lam 등은 이동 방송 환경에서의 병행성 제어 기법인 UFO(Update-First with Order) 알고리즘을 제안하였다[5]. 이 알고리즘에서 데이타의 방송은 방송 트랜잭션이라고 하는 트랜잭션으로서 모델된다. 따라서 서버는 갱신 트랜잭션과 읽기 전용 트랜잭션간의 충돌을 탐지하는 것이 아니라 갱신 트랜잭션과 방송 트랜잭션과의 충돌을 탐지한다. 만약 두 트랜잭션간에 충돌이 발생하였고 그 데이타가 이미 방송되었다면, 서버는 충돌을 해결하기 위해 그 데이타를 다시 전송해야 한다. 이 알고리즘에서는 갱신 트랜잭션과 방송 트랜잭션간의 충돌을 탐지하기 때문에 클라이언트가 충돌 탐지를 위해 서버와 통신할 필요가 전혀 없다는 장점을 가진다. 또한 방송 트랜잭션과 갱신 트랜잭션이 함께 수행될 수 있기 때문에 최신 현행성 요구조건을 만족한다. 그러나, 이 방법도 Pitoura의 방법과 마찬가지로 강 일관성을 사용하였고, 클라이언트는 자신이 읽은 데이타가 충돌이 발생하였는지를 전혀 알 수 없기 때문에 항상 재전송 될지도 모르는 데이타를 계속 기다려야 한다는 단점이 있다.

Shanmugasundaram 등은 갱신 일관성을 보장할 수 있는 기법을 제안하고 평가하였다[6]. 이 기법에서 서버는 각 방송 주기마다 그 주기가 시작하기 전에 마지막으로 갱신된 값뿐만 아니라 갱신 일관성을 검사하기 위한 제어 행렬을 함께 방송한다. 제어 행렬을 방송하는 방법은 매 방송 주기마다 j번째 데이타가 방송 될 때 바로 이어서 제어 행렬의 j번째 열을 함께 방송하는 것이다. 이 기법은 갱신 일관성을 보장하기 때문에 이동 방송 환경에 적합하지만, 각 방송 주기의 시작 시간인 t 이후에 갱신된 값을 현재의 방송 주기에서 방송할 수 없기 때문에 읽기 전용 트랜잭션의 현행성은 t-이전이 된다는 단점을 가진다. 이는 일관성을 유지하는 것이 방송 주기 단위로 이루어져서 한 방송 주기가 시작하고 난 후에는 새로 갱신된 값을 방송할 수 없기 때문이다. 따라서 이 기법을 사용하면 클라이언트는 갱신 일관적인 데이타를 읽을 수는 있지만 항상 최신의 데이타를 읽을 수 있는 것은 아니다.

4. 최신(latest-bound) 현행성을 보장하는 기법

이 절에서는 갱신 일관성뿐만 아니라 최신 현행성도 보장해주는 병행성 제어 기법을 제안한다. 본 논문에서 제시하는 기법은 Shanmugasundaram의 병행성 제어 기법[6]을 개선한 것으로, 현재의 방송 주기가 시작한 이후에 갱신된 값도 아직 방송되지 않았으면 방송할 수 있다는 특징을 가진다. 이를 위하여 "가상 방송 주기"를 새로이 제안하는데, 우선 사용되는 용어를 정의하면 다음과 같다.

- D_i 서버가 관리하고 있는 데이타 중 i번째 데이타 항목을 의미한다.
- T_j 데이타 D_j를 마지막으로 갱신한 갱신 트랜잭션을 의미한다. 최신 현행성을 보장하기 위해서 T_j는 현재의 방송 주기가 시작하고 난 후에 종료되어도 무방하지만, D_j를 방송하기 전에 종료되어야 한다.
- $Live(T)$ 트랜잭션 T가 직접적으로나 간접적으로 읽은 데이타를 갱신한 트랜잭션인 유효 트랜잭션(live transaction)의 집합을 의미한다. $Live(T)$는 다음 두 개의 규칙을 만족하는 최소 집합이다. (1) T는 $Live(T)$에 포함된다. (2) 만약 T'가 $Live(T)$에 포함되면, T'가 읽은 데이타를 갱신한 모든 트랜잭션 T''도 $Live(T)$에 포함된다.
- $Last_i(T_j)$ $Live(T_j)$에 포함되면서 데이타 D_i를 갱신한 트랜잭션 중 가장 최근에 종료된 트랜잭션을 의미한다. 즉, $Last_i(T_j)$는 D_j의 최신 값에 영향을 주면서 동시에 D_i를 마지막으로 갱신한 트랜잭션을 가리킨다.
- VC_i 데이타 D_i의 가상 방송 주기 번호를 의미한다. 가상 방송 주기란 그림 3과 같이 각 데이타가 방송되는 주기를 말한다. 그림 3에서 Cycle은 원래의 물리적인 방송 주기를 나타내며, D_i의 가상 방송 주기는 D_i가 방송될 때 시작하고 D_i가 다음에 방송될 때 종료한다. 본 논문에서는 모든 데이타의 방송 순서가 고정되어 있다고 가정한다.
- R_T 트랜잭션 T가 가상 방송 주기 번호가 VC_i일 때 D_i를 읽었다는 것을 나타내는 (D_i, VC_i)의 집합을 의미한다.
- RS_T 트랜잭션 T가 읽은 데이타의 집합을 의미한다.
- WS_T 트랜잭션 T가 갱신한 데이타의 집합을 의미한다.

이제 위에서 설명한 용어를 사용하여 갱신 일관성과 최신 현행성을 보장하기 위해 필요한 제어 행렬을 정의한다.

```
알고리즘 1.
1 Compute_Control_Matrix (Update_Transaction T)
2 begin
3   for each data item Dj in the server
4   begin
5     for each data item Di in the server
6     begin
7       if Di∈WST ∧ Dj∈WST then
8         Cnew(i,j)← current VCi
9       else if Di∉WST ∧ Dj∈WST then
10        if RST={} then Cnew(i,j) ← 0
11        else Cnew(i,j) ← max(Cold(i,k))
                where Dk∈RST
12      else Cnew(i,j) ← Cold(i,j)
13    end
14  end
15 end
```

그림 4 제어 행렬을 계산하기 위한 알고리즘

정의 1. 제어 행렬 C는 $n \times n$ (n은 전체 데이타의 수) 크기의 행렬로서, 제어 행렬의 임의의 항 $C(i,j)$는 트랜잭션 $Last_i(T_j)$가 종료된 가상 방송 주기 번호 VC_i로 설정된다. 만약 $Last_i(T_j)$가 존재하지 않는다면, 0으로 설정된다. ■

다음에 오는 절에서는 서버가 제어 행렬을 계산하기 위한 알고리즘과 클라이언트가 제어 행렬을 사용하여 갱신 일관성을 검사하는 방법을 차례로 설명한다.

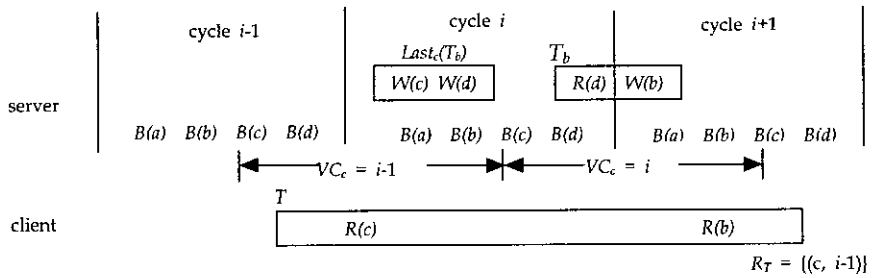
4.1 서버에서의 제어 행렬 계산

임의의 갱신 트랜잭션 T는 종료될 때마다 새로운 제어 행렬 C_{new}를 점진적으로 계산하는데, 이 때 수행되는 알고리즘 Compute_Control_Matrix가 그림 4에 나타나 있다. 제어 행렬의 모든 항은 처음에 0으로 초기화된다. C_{old}는 T가 종료하기 이전에 종료된 모든 갱신 트랜잭션을 고려한 제어 행렬을 의미한다. 그러면, 알고리즘은 다음과 같은 세 가지 조건을 검사한다. 첫째, 갱신 트랜잭션 T가 D_i와 D_j 둘 다 갱신하였다면, T가 바로 $Last_i(T_j)$가 되기 때문에 $C_{new}(i,j)$는 T가 종료된 가상 방송 주기 번호인 현재의 VC_i로 설정된다. 둘째, 갱신 트랜잭션 T가 D_j는 갱신하였지만 D_i는 갱신하지 않은 경우이다. 이 경우 RS_T에 포함되는 임의의 데이타 D_k를 마지막으로 갱신한 트랜잭션을 T_k라 할 때 T는 T_k에 의존하게 된다. 따라서 $C_{new}(i,j)$는 $Live(T_k)$에 포함되는 트랜잭션 중 가장 최근에 종료된 트랜잭션의 가상 방송 주기 번호인 $C_{old}(i,k)$로 설정된다. 만약 T가 읽은 데이타가 하나도 없다면 T가 의존하는 갱신 트랜잭션도 없기 때문에 $C_{new}(i,j)$는 0으로 설정된다. 셋째, 위의 두 가지 경우를 제외한 나머지 경우에는 T가 D_j를 갱신하지 않았기 때문에 $C_{new}(i,j)$는 $C_{old}(i,j)$의 값이 그대로 유지된다. 이렇게 계산된 제어 행렬은 열 단위로 분리되어서 데이타 D_j가 방송될 때 바로 이어서 j번째 열이 함께 방송된다.

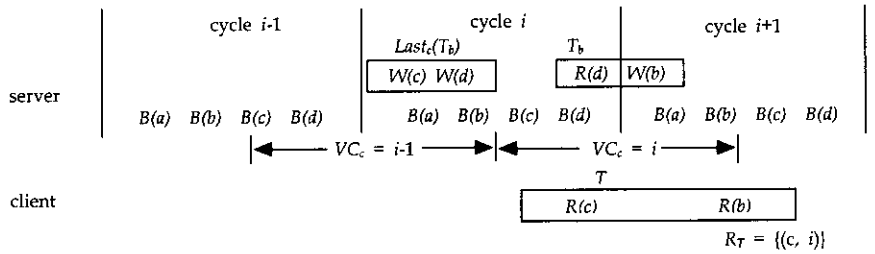
4.2 클라이언트에서의 갱신 일관성 검사

읽기 전용 트랜잭션 T에 대하여 T가 D_j를 읽을 때 클라이언트는 다음과 같은 조건을 검사해야 한다.

수식 1. $\forall (D_i, VC_i) \in R_T,\ C(i,j) < VC_i$



(a) T가 b를 읽는 것이 실패

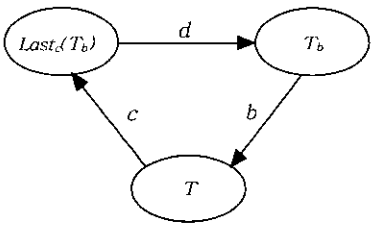


(b) T가 b를 읽는 것이 성공

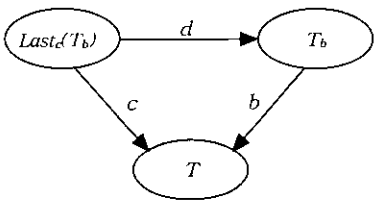
그림 5 제어 행렬을 사용하는 예

이 때 VC_i은 T가 D_i를 읽었을 때의 가상 방송 주기 번호이다. 만약 이 조건이 실패하면, T는 D_j를 읽지 못하고 중단되어야 한다. 직관적으로 보았을 때, T가 D_i를 읽은 다음 $Live(T_j)$에 포함되는 어떤 트랜잭션도 D_i를 갱신하지 않은 경우에만 T의 읽기 연산이 허용된다.

그림 5는 읽기 전용 트랜잭션 T가 데이타 b를 읽으려고 할 때 수식 1을 검사하는 두 가지 예를 보여준다. 서버는 a, b, c, d 네 개의 데이타를 주기적으로 방송하며, 그림에서는 데이타 c의 가상 방송 주기만을 보여준다. 두 예에서 갱신 트랜잭션 T_b는 갱신 트랜잭션 $Last_c(T_b)$가 갱신한 데이타 d를 읽기 때문에 $Last_c(T_b)$에 의존하며, 제어 행렬 $C(c,b)$의 값은 $Last_c(T_b)$가 종료된 가상 방송 주기 번호인 i-1로 설정된다. 첫 번째 예는 갱신 일관성 때문에 T가 b를 읽는 것이 실패한 경우이고, 두 번째 예는 T가 b를 읽는 것이 성공한 경우를 각각 보여준다. 그림 5 (a)에서 $C(c,b)$의 값이 T가 c를 읽은 가상 방송 주기 번호와 동일하기 때문에 T는 자신이 읽은 c의 값이 $Last_c(T_b)$가 갱신하기 전의 값이라는 사실을 알 수 있다. 따라서 그림 6 (a)와 같이 세 개의 트랜잭션이 순환을 형성하고 두 갱신 트랜잭션이 쓰기-읽기 간선으로 연결되어 있기 때문에 T가 b를 읽는 것은 실패한다. 이와는 반대로 그림 5 (b)에서는 $C(c,b)$의 값이 T가 c를 읽은 가상 방송 주기 번호인 i보다 작기 때문에 T는 자신이 읽은 c의 값이 $Last_c(T_b)$가 갱신한 값이라는 사실을 알 수 있다. 따라서 그림 6 (b)와 같이 세 개의 트랜잭션이 순환을 형성하지 않기 때문에 T가 b를 읽는 것은 성공한다.



(a) T가 b를 읽는 것이 실패



(b) T가 b를 읽는 것이 성공

그림 6 그림 5에 대한 직렬화 그래프

5. 알고리즘 검증

이 절에서는 4절에서 제안한 병행성 제어 기법을 사용하면 갱신 일관성과 최신 현행성이 보장된다는 것을 검증한다. 이를 위해 먼저 그림 4의 알고리즘 1이 올바른 제어 행렬을 생성한다는 것을 보인 다음에, 제안된 기법이 갱신 일관성과 최신 현행성을 보장한다는 것을 증명하겠다.

정리 1. 알고리즘 1을 사용하여 생성된 제어 행렬 C는 정의 1을 만족한다.

증명. 전체 트랜잭션의 스케줄 상에서 갱신 트랜잭션이 종료된 직렬 순서를 사용하여 수학적 귀납법으로 증명한다. 처음에 어떤 트랜잭션도 아직 종료되지 않았을 때 제어 행렬 C의 모든 항은 0이어야 한다. 이는 알고리즘 1에서 보장된다. 이제 스케줄 상에서 h번째로 종료된 트랜잭션 T에 대하여 알고리즘 1이 정의 1을 만족하는 제어 행렬을 생성한다고 가정하면, h+1번째로 종료된 트랜잭션 T'에 대하여도 정의 1을 만족하는 제어 행렬을 생성한다는 사실을 보이면 된다. 귀납적 가설에 따라 알고리즘 1에서 C_{old}는 T까지의 모든 트랜잭션을 고려한 제어 행렬이 된다. 그러면 알고리즘 1의 세 가지 경우를 고려하여 생성된 제어 행렬이 모두 정의 1을 만족하는지 살펴보자. 첫째, 임의의 데이타 D_i와 D_j가 모두 $WS_{T'}$에 포함되는 경우, T'는 D_j의 가장 최근 값에 영향을 주면서 동시에 D_i를 마지막으로 갱신한 트랜잭션이다. 따라서 알고리즘 1의 8번째 줄에서 $C_{old}(i,j)$의 값에 관계없이 $C_{new}(i,j)$의 값은 T'가 종료한 시점인 현재의 VC_i로 설정되는데, 이는 정의 1을 만족한다. 둘째, 데이타 D_j가 $WS_{T'}$에 포함되지만 D_i는 포함되지 않은 경우, $RS_{T'}$에 포함된 임의의 데이타 D_k에 대하여 D_k의 가장 최근 값에 영향을 주면서 동시에 D_i를 마지막으로 갱신한 트랜잭션인 $Last_i(T_k)$이 존재하며 귀납적 가설에 따라 $C_{old}(i,k)$는 이 트랜잭션이 종료한 가상 방송 주기 번호를 가지고 있다. 한편 T'는 이 트랜잭션이 갱신한 데이타 D_k를 읽었기 때문에 $Last_i(T_k)$에 의존하게 되고, $Last_i(T_k)$ 중 가장 마지막으로 종료된 트랜잭션이 바로 $Last_i(T_j)$가 된다. 알고리즘 1의 11번째 줄에서 $C_{new}(i,j)$의 값은 $C_{old}(i,k)$ 중에 가장 큰 값으로 설정되어 $Last_i(T_j)$가 종료한 가상 주기 번호가 되는데, 이는 정의 1을 만족한다. 만약 이러한 조건을 만족하는 트랜잭션이 하나도 없다면 T'는 어느 트랜잭션에도 의존하지 않기 때문에 알고리즘 1의 11번째 줄에서 $C_{new}(i,j)$의 값은 0이 되고, 이는 정의 1을 만족한다. 셋째, $WS_{T'}$에

포함되지 않는 모든 데이타 D_j에 대하여, T'는 D_j를 갱신하지 않았기 때문에 T_j가 될 수 없다. 따라서 알고리즘 1의 12번째 줄에서 $C_{new}(i,j)$의 값은 변경되지 않고, 이는 정의 1을 만족한다. ■

정리 2. 수식 1을 검사하면 갱신 일관성을 보장할 수 있다.

증명. 갱신 일관성을 보장하기 위해서는 현재까지 수행된 트랜잭션으로 구성된 직렬화 그래프에서 (1) 한 개의 읽기 전용 트랜잭션과 한 개의 갱신 트랜잭션으로 구성된 순환 및 (2) 한 개의 읽기 전용 트랜잭션과 두 개 이상의 갱신 트랜잭션을 포함하는 순환 중 두 개의 갱신 트랜잭션이 쓰기-읽기 간선으로 연결된 순환이 존재하지 않는다는 것을 검사하면 된다. 먼저 (2)의 경우를 증명하면 다음과 같다. 읽기 전용 트랜잭션 T가 임의의 데이타 D_j를 읽을 때, D_j를 마지막으로 갱신한 트랜잭션 T_j와 다른 임의의 데이타 D_i에 대하여 $Last_i(T_j)$가 존재한다. $Last_i(T_j)$는 k개의 갱신 트랜잭션들을 중간에 두고 쓰기-읽기 간선으로 T_j까지 연결되어 있고 T_j와 T도 쓰기-읽기 간선으로 연결된다. 이 때 T가 D_i를 읽은 시점이 $Last_i(T_j)$가 D_i를 갱신한 시점보다 앞서면(수식 1이 거짓인 경우), T와 $Last_i(T_j)$가 읽기-쓰기 간선으로 연결되기 때문에 순환이 발생한다는 사실을 클라이언트가 알 수 있다. (1)의 경우는 k가 0인 경우이기 때문에 (2)와 마찬가지로 이유로 순환이 발생한다는 사실을 알 수 있다. 그러므로 수식 1을 검사하면, 클라이언트가 (1)과 (2)와 같은 순환을 탐지할 수 있으므로 갱신 일관성을 보장할 수 있다. ■

정리 3. 알고리즘 1과 수식 1을 사용한 병행성 제어 기법은 최신 현행성을 보장한다.

증명 알고리즘 1과 수식 1을 사용한 병행성 제어 기법에서는 현재의 방송 주기 c가 시작되었더라도 임의의 데이타 D_i가 아직 방송되지 않았으면 D_i에 대한 가상 방송 주기 번호 VC_i는 $c-1$이다. 그러므로 서버는 현재의 방송 주기가 시작된 이후에 갱신된 최신의 값을 방송하더라도 제어 행렬에 포함된 정보를 사용하면 갱신 일관성을 보장할 수 있다. 그러므로, 이 병행성 제어 기법은 최신 현행성을 보장한다. ■

6. 성능 평가

본 논문에서는 4절에서 제안한 병행성 제어 기법의 성능을 시뮬레이션을 통해 측정한다. 시뮬레이션한 알고리즘은 Shanmugasundaram의 기법과 본 논문의 기법이다. 이 절에서는 전자를 t-이전 현행성, 후자를 최신 현행성 기법으로 각각 부른다. 본 논문에서 구현한 시뮬레이터는 하나의 서버와 클라이언트를 가진다. 서버는 계속적으로 갱신 트랜잭션을 수행하고 주기적으로 모든 데이타들을 방송한다. 클라이언트는 읽기 전용 트랜잭션을 수행하다가 읽기 연산이 실패하면 트랜잭션을 중단한다. 이동 방송 환경에서는 클라이언트의 수가 전체 성능에 전혀 영향을 주지 않기 때문에 본 논문에서는 하나의 클라이언트만 실험한다. 또한 클라이언트가 캐쉬를 가지고 있는 경우는 고려하지 않는다. t-이전 현행성 기법과 최신 현행성 기법은 모두 동일한 방법으로 제어 행렬을 방송하기 때문에 제어 행렬을 방송하기 위한 통신량이나 오버헤드가 항상 동일하다. 그러므로 본 실험에서는 이들을 별도로 측정하지 않고 다음과 같은 두 개의 척도를 사용하여 성능을 측정한다.

실효된 읽기의 수 읽기 전용 트랜잭션이 가장 최근에 갱신된 데이타를 읽지 못한 횟수를 의미한다.

트랜잭션 중단의 수 읽기 전용 트랜잭션이 읽은 데이타가 갱신 일관성을 만족하지 못하기 때문에 중단된 횟수를 의미한다.

6.1 시뮬레이션 인자

표 1 시뮬레이션 인자

인자	내정 값
총 데이타의 수	1000
총 갱신 트랜잭션의 수	10000
갱신 트랜잭션의 평균 도착간 시간	100
갱신 트랜잭션 당 데이타 접근 수	10
갱신 트랜잭션 당 데이타 갱신 수	2
읽기 전용 트랜잭션의 평균 도착간 시간	200
읽기 전용 트랜잭션 당 데이타 읽기 수	4
서버 읽기의 평균 도착간 시간	10
서버 읽기의 범위	1000
서버 갱신의 평균 도착간 시간	10
서버 갱신의 범위	500
클라이언트 읽기의 평균 도착간 시간	100
클라이언트 읽기의 범위	250
Zipf 분포에서의 θ	0.95
서버와 클라이언트 접근간의 오프셋	0

표 1은 본 실험에 사용된 인자와 내정(default) 값이다. 본 논문에서는 종료된 갱신 트랜잭션의 수가 10000개일 때까지 실험하였다. 모든 트랜잭션과 데이타 접근은 표 1에 나온 평균 시간을 가지는 지수 분포(exponential distribution)를 따라 도착한다고 가정한다. 서버

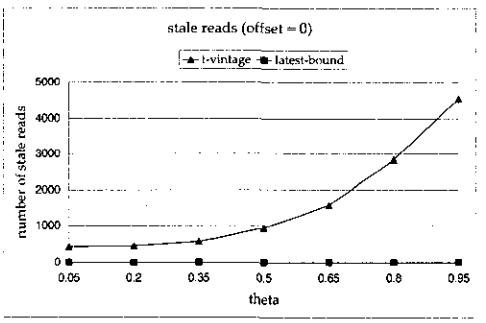
는 총 1000개의 데이타를 가지고 있으며, 이들 모두를 항상 방송한다. 서버의 읽기 범위는 1000개이고, 갱신 범위는 500개이고, 클라이언트의 읽기 범위는 250개이다. 이 범위 내에서 서버와 클라이언트의 접근 확률은 Zipf 분포[8][9]를 따른다. Zipf 분포는 보통 비 규칙적인 접근 패턴을 모델할 때 사용된다. 데이타 i를 접근할 확률은 $(1/i)^{\theta}$에 비례하며, θ가 증가함에 따라 접근 패턴도 비대칭적이 된다. 본 실험에서의 θ의 내정 값은 0.95로서, 접근 패턴이 매우 비대칭적인 경우이다. 마지막으로 오프셋 인자는 서버와 클라이언트의 접근 패턴 간의 차이를 모델한다. 오프셋이 0이면, 서버와 클라이언트의 접근 분포가 가장 많이 겹치는 것을 의미하며 클라이언트가 집중적으로 접근하는 데이타를 서버도 집중적으로 갱신하는 경우이다. 오프셋이 k이면, 클라이언트가 집중적으로 접근하는 데이타가 k만큼 이동하므로 서버가 그 데이타를 덜 접근한다.

6.2 실험 결과

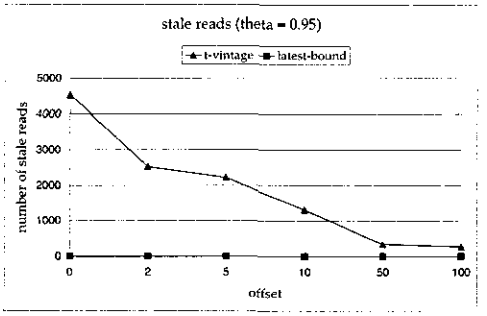
본 논문에서는 t-이전과 최신 현행성 기법에서 발생하는 실효된 읽기의 수를 먼저 측정하였다. 그림 7은 두 기법의 실효된 읽기의 수를 보여주는데, (a)는 오프셋을 0으로 고정시키고 θ를 0.05부터 0.95까지 변화시키면서 측정한 값이고 (b)는 이와 반대로 θ를 0.95로 고정시키고 오프셋을 0부터 100까지 변화시키면서 측정한 값이다. 그래프에서 볼 수 있듯이 최신 현행성 기법에서는 항상 서버가 가장 최근에 갱신된 값을 방송하기 때문에 클라이언트가 실효된 읽기를 전혀 하지 않는다. 반면에, t-이전 현행성 기법에서는 θ가 증가함에 따라 428개부터 최대 4550개의 실효된 읽기가 발생하고, 오프셋이 감소함에 따라 266개부터 4550개의 실효된 읽기가 발생한다. 이는 θ가 크고 오프셋이 작을수록 서버와 클라이언트가 동시에 집중적으로 동일한 데이타를 접근하기 때문이다.

다음으로 갱신 트랜잭션과의 비 일관성 때문에 읽기 전용 트랜잭션이 중단되는 횟수를 측정하였다.

그림 8은 그림 7에서와 동일하게 인자 값을 변화시킬 때 발생하는 트랜잭션 중단의 수를 보여준다. 두 기법 모두에서 θ가 증가하거나 오프셋이 작아질수록 더 많은 트랜잭션이 중단되는 것을 볼 수 있다. 그러나, 최신 현행성 기법에서의 트랜잭션 중단 수가 t-이전 현행성 기법에서의 수보다 항상 작게 나타나는 것을 볼 수 있다. 이는 항상 클라이언트가 가장 최근에 갱신된 데이타

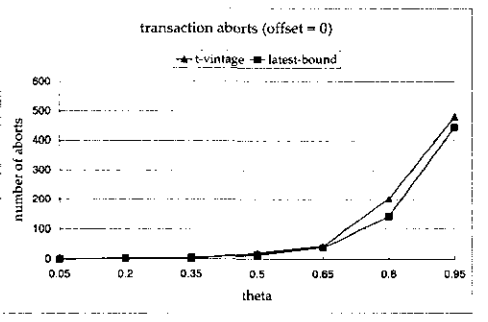


(a) θ 를 변화시킬 때

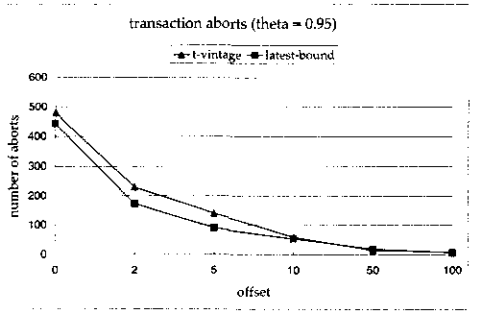


(b) 오프셋을 변화시킬 때

그림 7 실효된 읽기의 수



(a) θ 를 변화시킬 때



(b) 오프셋을 변화시킬 때

그림 8 트랜잭션 중단의 수

를 읽음으로써 직렬화 그래프에서 순환이 발생하는 경우를 제거할 수 있기 때문이다. 이를 정리하면, t-이전 현행성 기법에서는 클라이언트의 전체 읽기 중에서 최대 23.53%(θ가 0.95일 때)부터 최소 2.13%(θ가 0.05일 때)까지 실효된 읽기가 발생하지만 최신 현행성 기법에서는 이를 모두 0%로 감소시킨다는 사실을 알 수 있다. 또한, 최신 현행성 기법은 트랜잭션이 중단되는 횟수도 t-이전 현행성 기법과 비교할 때 최대 29.35%(θ가 0.8일 때)부터 최소 7.48%(θ가 0.95일 때)까지 감소시킨다.

7. 결 론

본 논문에서는 이동 방송 환경에서의 효율적인 병행성 제어 기법을 제안하였다. 제안된 기법에서는 서버에서 가상 방송 주기를 이용하여 제어 행렬을 계산하고 클라이언트가 이를 검사함으로써, 데이타의 최신 현행성을 보장해준다. 제안된 기법을 검증한 결과, 클라이언트가 수행하는 읽기 전용 트랜잭션이 갱신 일관성뿐만 아니라 최신 현행성도 만족함을 알 수 있었다. 제안된 기법의 성능을 측정하기 위해서 서버와 클라이언트의 접근 패턴을 다양하게 변화시키면서 두 개의 비용 척도인 실효된 읽기의 수와 트랜잭션 중단의 수를 측정한 결과, 기존의 병행성 제어 기법과 달리 클라이언트가 전혀 실효된 읽기를 하지 않음을 알 수 있었다. 또한, 읽기 전용 트랜잭션이 비 일관성 때문에 중단되는 경우도 줄일 수 있음을 확인하였다. 제안된 기법은 주식 거래나 경매와 같이 클라이언트가 정확한 결과를 얻기 위해 최신 현행성을 요구하는 환경에 유용하게 사용될 수 있다.

참 고 문 헌

[1] T. Imielinski and B. R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management," *Communications of the ACM*, Vol. 37, No. 10, pp. 18-28, October 1994.

[2] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Energy Efficient Indexing on Air," In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 25-36, May 1994.

[3] E. Pitoura, "Supporting Read-Only Transac tions in Wireless Broadcasting," In *Proceedings of the International Workshop on Mobility in Databases and Distributed Systems*, pp. 428-433, August 1998.

[4] H. Garcia-Molina and G. Wiederhold, "Read- Only Transactions in a Distributed Database," *ACM Transactions on Database Systems*, Vol. 7, No. 2, pp.209-234, June 1982.

[5] K.-Y. Lam, M.-W. Au, and E. Chan, "Broadcast of Consistent Data to Read-Only Transactions from Mobile Clients," In *Proceedings of the International Workshop on Mobile Computer Systems and Applications*, pp. 80-88, February 1999.

[6] J. Shanmugasundaram, A. Nithrakashyap, R. Sivasankaran, and K. Ramamritham, "Efficient Concurrency Control for Broadcast Environments," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 85-96, June 1999.

[7] P. M. Bober and M. J. Carey, "Multiversion Query Locking," In *Proceedings of the 18th International Conference on Very Large Data Bases*, pp. 497-510, August 1992.

[8] J. Gray, P. Sundaresan, S. Englert, K. Baclawski, and P. J. Weinberger, "Quickly Generating Billion-Record Synthetic Databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 243-252, May 1994.

[9] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 199-210, May 1995.

한 부 형

1991년 서울대학교 컴퓨터공학과 학사. 1991년 서울대학교 컴퓨터공학과 석사. 1993년 ~ 현재 서울대학교 컴퓨터공학과 박사과정. 관심분야는 운영체제, 이동 방송 시스템, 분산 시스템 등임.

정 성 권

1982년 서울대학교 컴퓨터공학과 학사. 1984년 서울대학교 컴퓨터공학과 석사. 1990년 University of Washington, Department of Computer Science & Engineering, 박사. 1998년 ~ 2000년 서울대학교 컴퓨터공학과 초빙교수. 1996년 ~ 현재 유비쿼스 주식회사 대표이사. 관심분야는 이동 컴퓨팅, 분산 시스템 등임.

조 유 근

정보과학회논문지 : 정보통신

제 28 권 제 1 호 참조