방송 디스크 환경에서 갱신 트랜잭션을 위한 실시간 동시성 제어

임성준[0] · 조행래

Real-Time Concurrency Control for Update Transactions in Broadcast Disks Environment

Sungjun Lim[0] · Haengrae Cho

요 약

방송 디스크 환경에서는 서버와 클라이언트간의 비대칭적인 대역폭으로 인해 전통적인 동시성 제어 기법을 적용시키기 힘들다. 뿐만 아니라, 최근 방송 디스크 환경에서 대부분의 응용분야들이 실시간 트랜잭션 처리를 요구하고 있다. 실시간 트랜잭션 처리는 데이터 일관성뿐만 아니라 트랜잭션이 마감기한 내에 완료해야하는 시간적 제약을 고려해야 한다. 본 논문에서는 방송 디스크 환경에서의 많은 제약들과 트랜잭션의 실시간성을 만족시킬 수 있는 동시성 제어 기법을 제안한다. 제안한 기법은 클라이언트에서 실행되는 읽기 전용 트랜잭션을 서버와의 접촉 없이 자체적으로 처리할 수 있다. 또한 서버에서의 전역 검증 과정에서 발생하는 클라이언트 갱신 트랜잭션의 데이터 충돌을 클라이언트에서 미리 발견하여 재실행시킴으로써 마감기한을 놓치는 트랜잭션의 수를 줄일 수 있는 장점을 갖는다.

1. 서 론

방송 디스크는 이동 통신 환경에서 다수의 클라이언트에게 정보를 전파하기 위하여 제안된 구조로서, 서버는 데이터베이스에 저장된 모든 데이터를 연속적으로 반복하여 방송한다[1,5]. 클라이언트는 방송 채널을 감시하여, 원하는 데이터가 방송될 경우 방송 채널로부터 데이터를 수신한다. 이런 관점에서 방송 채널은 클라이언트가 데이터를 액세스할 수 있는 저장 장치(디스크)의 역할을 수행한다고 할 수 있다. 방송 디스크는 접속할 수 있는 클라이언트의 수에 제한이 없다는 장점으로 인해 이동 통신망을 이용한 경매나 전자 입찰과 같은 전자 상거래 응용 분야와 주식 거래, 그리고 기상 정보나 교통 정보 방송 분야와 같은 다양한 응용 분야에 활용되고 있다[2].

데이터가 방송되는 동안 서버에서 실행되는 갱신 작업에 의해 데이터의 내용이 변경될 수 있으며, 변경된 데이터는 다음 사이클에서 방송된다. 만약 클라이언트에서 실행되는 트랜잭션이 다른 사이클에서 방송된 여러 개의 데이터를 수신할 경우, 데이터들 간의 변경 시점의 차이로 인하여 트랜잭션의 정확성을 유지하기 위한 동시성 제어 기법이 필요하다[2]. 그러나 방송 환경에서는 2단계 로킹이나 낙관적인 동시성 제어와 같은 전통적인 동시성 제어 기법들을 적용시키기 힘들다. 이 기법들은 서버와 클라이언트간에 빈번한 통신이 요구되기 때문이다[4].

위에서 언급한 방송환경에서의 대부분의 응용분야들은 본질적으로 실시간 속성을 지닌다[7]. 유선 및 무선 네트워크 환경에서, 클라이언트 트랜잭션과 서버 트랜잭션 모두 다 일종의 마감기한(deadline) 형태의 시간 제약(timing constraints)을 지닌다[6,11]. 예를 들면, 주식 거래에 관련된 트랜잭션이 마감기한 내에 완료할 수 없을 때, 주식 거래자가 사무실에 있든지 이동중이든지 또는 제출한 트랜잭션의 종류에 관계없이 재정 또는 기회 손실을 발생시킬 것이다[9].

본 논문에서는 방송디스크 환경의 특성을 최대한 이용하고 트랜잭션의 정확성과 긴급성을 동시에 지원할 수 있는 실시간 동시성 제어 기법을 소개한다. 본 논문의 구성은 다음과 같다. 먼저 2절에서는 기존에 제안된 동시성 제어 기법들을 살펴보고, 3절에서는 본 논문에서 제안한 동시성 제어 기법을 설명한다. 마지막으로 4절에서 결론 및 앞으로의 연구방향에 대해 논의하기로 한다.

2. 관련 연구

방송 디스크를 위한 기존의 동시성 제어 기법들은 클라이언트에서 읽기 전용 트랜잭션만 실행된다고 가정한다. 이와는 달리 본 논문에서 제안하는 동시성 제어 기법은 클라이언트에서 읽기 전용 트랜잭션과 갱신 트랜잭션이 모두 발생한다고 가정한다. [7,8]에서는 클라이언트에서 읽기 전용 트랜잭션과 갱신 트랜잭션을 모두 지원하는 동시성 제어 기법들을 제안하였다. [8]에서는 트랜잭션의 타임스탬프 구간을 기록함으로써 동적으로 직렬화 순서를 검사하여 불필요한 트랜잭션의 철회를 줄인다. 하지만 이 기법은 클라이언트에서 실행을 마친 읽기 전용 트랜잭션도 서버로 전송함으로써, 읽기 전용 트랜잭션의 실행 시간이 길어지고 철회율이 증가하여 마감 기한을 놓치는 트랜잭션이 많아진다. [7]의 경우, 전통적인 낙관적 기법을 변형하여 서버에서는 전방향 낙관적 기법을 사용하고 클라이언트에서는 후방향 낙관적 기법을 사용한다. 이 기법에서 클라이언트는 읽기 전용 트랜잭션을 서버와의 접촉 없이 자체적으로 완료할 수 있고 서버에서는 전방향 낙관적 기법을 사용함으로써 서버로 전송된 클라이언트의 갱신 트랜잭션이 마감기한을 놓칠 가능성을 줄였다. 하지만 이들 기법은 클라이언트 갱신 트랜잭션을 트랜잭션의 실행이 끝나자마자 전역 검증을 위해 서버로 전송한다.

그림 1은 이 같은 상황을 나타낸다. 클라이언트 갱신 트랜잭션 CT_2는 데이터 y를 읽고, 자신의 작업공간에 변경된 y의 값을 저장한다. 그리고 CT_2는 실행을 마침과 동시에 전역 검증을 위해 서버로 전송된다. 그러나 서버 갱신 트랜잭션 ST_1과의 충돌로 인해 CT_2는 전역 검증 단계를 통과할 수 없다. 서버에서의 충돌로 인해 CT_2는 철회되고 다음 사이클의 시작 시점에서 서버는 클라이언트에게 CT_2가 철회됐음을 통보한다. 그 결과, 클라이언트에서 서버로의 값비싼 대역폭을 사용함으로써 비싼 통신비용이 소비되고, 서버에서 발생하는 트랜잭션보다 클라이

언트의 갱신 트랜잭션이 상대적으로 마감기한을 놓칠 가능성이 커진다.

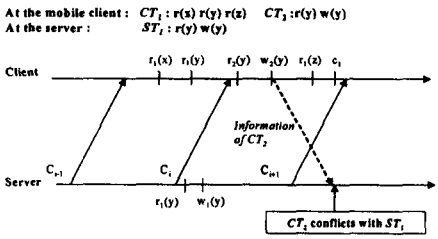

그림 1. 서버로 전송된 클라이언트의 갱신트랜잭션이 철회되는 경우

3. 동시성 제어 기법

본 절에서는 방송 디스크 환경에서 클라이언트의 갱신 트랜잭션을 효율적으로 처리할 수 있는 새로운 동시성 제어 기법을 제안한다. 제안한 기법의 기본 개념은 다음과 같다.

- 서버에서 방송 프로그램 작성 시, 제어 정보 테이블(CIT)의 방송 주기를 주 사이클보다 짧아지도록 조절한다. 따라서, 클라이언트는 자체적으로 갱신 트랜잭션을 철회할 수 있는 기회를 더 많이 갖는다.
- 클라이언트에서 트랜잭션의 실행을 마쳤을 경우, 읽기 전용 트랜잭션은 완료하고 갱신 트랜잭션은 다음 CIT의 전송을 기다린다. 클라이언트에서 마지막 검증을 마친 다음 서버로 전역 검증을 위해 갱신 트랜잭션을 전송한다.

3.1 서버 알고리즘

서버는 두 가지 기본적인 작업을 수행한다. 한 가지는 주기적으로 다수의 클라이언트에게 데이터들을 방송하는 것이다. 방송의 한 주기를 사이클이라 하며, 각 사이클에 방송되는 내용을 방송 프로그램이라 한다. 방송 프로그램의 작성 예가 그림 2에 나타난다. 서버의 데이터베이스는 1부터 11까지 11개의 데이터를 저장하고 있으며, 데이터의 예상 참조 수에 따라 3개의 디스크로 나누어진다. 디스크 D_1은 가장 빈번히 참조되며, D_3은 참조 수가 가장 낮다고 가정한다. 그 결과로 작성된 방송 프로그램의 경우, D_1과 D_2, 그리고 D_3의 방송 빈도수는 4:2:1이 됨을 알 수 있다. 이때 방송 빈도수가 가장 높은 D_1이 방송되는 주기를 부 사이클이라 하며, 방송 빈도수가 가장 낮은 D_3이 방송되는 주기를 주 사이클이라 한다[2].

데이터가 방송되는 동안에 서버에서 서버와 클라이언트의 갱신 트랜잭션으로 인해 값이 변경될 수 있고, 변경된 데이터 값은 다음 사이클의 시작 시점에 CIT를 통해서 방송된다. CIT는 직전 방송 주기에서 완료된 갱신 트랜잭션의 타임스탬프 TS(U), 쓰기 집합 WS(U)를 포함한다. CIT는 클라이언트가 현재 실행중인 트랜잭션이 지난 사이클 동안 서버에서 완료된 트랜잭션과 직렬화가능한지를 검사하는데 사용된다. 일반적으로 CIT는 방송 시작 시점에 방송된다. 그림 1에서 알 수 있듯이 CIT의 주기가 주 사이클에 맞춰짐으로써 서버에서의 갱신 트랜잭션의 충돌을 클라이언트에서 발견할 수 없다. 하지만 CIT의 주기를 짧게 함으로써 서버에서의 충돌을 클라이언트에서 미리 발견할 수 있다. 예를 들면, 그림 1에서 CIT의 주기를 부 사이클로 조정한다면 $w_1(y)$의 결과가 C_i 와 C_{i+1} 사이에 클라이언트로 방송되고 클라이언트는 트랜잭션을 서버로 전송하기 전에 CT_2와 ST_1의 충돌을 발견 할 수 있다. 따라서 좀 더 이른 시점에서 클라이언트 트랜잭션을 재실행함으로써 마감기한을 만족시킬 가능성이 높아질 수 있다.

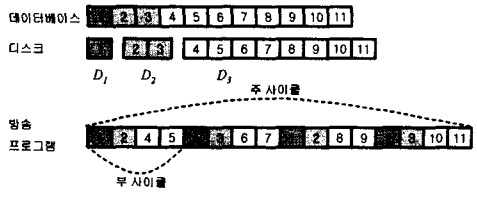

그림 2. 방송 프로그램의 작성 예[1]

또 다른 서버의 역할은 전송된 클라이언트의 갱신 트랜잭션을 직렬화 가능성이 보장되도록 전역 검증 작업을 수행하는 것이다. 전역 검증 과정은 그림 3과 같다. 갱신 트랜잭션 U에 대해 WS(U)는 U의 쓰기 집합이고, RS(U)는 읽기 집합, 그리고 TS(U)는 U의 타임스탬프이고, RTS(d)와 WTS(d)는 각각 데이터의 읽기 타임스탬프와 쓰기 타임스탬프이다. TOR(U, d)은 U가 데이터 d를 읽었을 때 WTS(d)의 값이다. LB(U)는 트랜잭션 U의 하한 계, UB(U)는 상한 계라고 정의한다. 그리고 타임스탬프 구간의 하한 계가 상한 계보다 커지면 트랜잭션은 철회된다.

· 전역 검증을 위해 전송된 클라이언트의 갱신 트랜잭션 U에 대해서 다음과 같은 과정을 수행한다.
1. WS(U)에 있는 모든 데이터 d에 대해서,
   ① WTS(d) > TOR(U, d) 일 경우, 해당 트랜잭션은 철회되고 비정상 종료를 표시한다. 그렇지 않다면, LB(U)와 RTS(d)중에 최대값을 LB(U_i)로 설정한다.
   ② LB(U_i) >= UB(U_i) 일 경우, 트랜잭션은 철회되고 비정상 종료를 표시한다.
· 전역 검증을 통과하면, 아래와 같은 과정을 수행한다.
1. 현재 타임스탬프를 TS(U)에 할당한다.
2. RS(U)에 해당하는 모든 데이터 d의 RTS와 WS(U)에 해당하는 모든 데이터 d의 WTS에 TS(U)를 할당한다.
3. TS(U)와 WS(U)를 CIT에 저장한다.
4. 해당 트랜잭션이 정상 종료되었음을 표시한다.

그림 3. 서버 알고리즘

3.2 클라이언트 알고리즘

클라이언트에서는 다음 세 가지 중요한 작업을 수행한다. 첫째, 트랜잭션의 읽기 및 쓰기 연산을 처리하고, 둘째, 사이클의 시작 시점에서 방송되는 CIT를 통해서 현재 실행중인 트랜잭션이 유효한지를 검사하며, 마지막으로 갱신 트랜잭션의 전역 검증을 위해서 갱신 트랜잭션을 서버로 전송한다. 그림 4에 위 세 가지의 동작과정이 나타난다. *Submission Set*은 전역 검증을 위해서 서버로 전송한 갱신 트랜잭션들의 집합이고, CUT는 지난 방송주기 동안 완료된 갱신 트랜잭션의 집합이다.

새로운 사이클이 시작되면 CIT를 수신하여 현재 실행중인 트랜잭션이 변경된 데이터들에 대해서 유효한지를 검사한다. 트랜잭션이 실행을 마치면 읽기 전용 트랜잭션은 완료하고, 갱신 트랜잭션은 클라이언트에서의 마지막 부분 검증 단계를 위해 새로운 CIT가 전송될 때까지 대기한다. 그리고 마지막 부분 검증 단계를 마친 후 전역 검증을 위해서 서버로 전송한다. 예를 들면, 그림 1에서 실행이 끝난 CT_2는 서버로 전송하지 않고 다음 CIT를 기다린다. 그 결과, 클라이언트는 마지막 부분 검증 단계에서 CT_2와 ST_1사이의 충돌을 탐지할 수 있다. 이것은 대기 전송 기법이 서버에서 발생할 충돌을 클라이언트에서 미리 탐지할 수 있음을 보여준다.

· 트랜잭션 T_i의 각 연산을 수행하는 경우 아래와 같은 과정을 수행한다.
1. 읽기 연산이라면,
 ① 데이터 d를 읽는다.
 ② RS(T_i)에 d를 저장하고, TOR(T_i, d)에 WTS(d)의 값을 저장한다.
 ③ LB(T_i)와 WTS(d) 중에 최대값을 LB(T_i)로 설정한다.
 ④ LB(T_i) >= UB(T_i) 일 경우, T_i은 철회되어 재실행한다.
2. 쓰기 연산이라면,
 ① LB(T_i)와 RTS(d)중에 최대값을 LB(T_i)로 설정한다.
 ② LB(T_i) >= UB(T_i) 일 경우, T_i은 철회되어 재실행한다.
 ③ WS(T_i)에 데이터 d를 저장하고 새로운 값을 임시공간에 저장한다.
3. 트랜잭션의 실행이 끝났다면,
 ① 읽기 전용 트랜잭션이라면, 완료하고 현재 타임스탬프를 트랜잭션 타임스탬프에 할당하고, RS(T_i)에 해당하는 모든 데이터 d의 RTS에 TS(T_i)를 할당한다.
 ② 갱신 트랜잭션이라면, 현재 사이클이 끝날 때까지 대기한다.
· 새로운 사이클이 시작되어 CIT를 수신하는 경우에는 아래와 같은 과정을 수행한다.
1. *Submission Set*에 있는 모든 트랜잭션 T_i에 대해서, 정상종료 됐다면 완료하고 비정상종료 됐다면 철회하고 재실행하여 *Submission Set*에서 T_i을 제외한다.
2. 현재 실행중인 모든 트랜잭션 T_a에 대해서,
 ① CUT에 있는 모든 U_i에 대해서, WS(U_i) ∩ WS(T_a) ≠ {}인 U_i가 존재한다면, 철회되어 재실행한다.
 ② CUT에 있는 모든 U_i에 대해서, WS(U_i) ∩ RS(T_a) ≠ {}인 U_i가 존재한다면, UB(T_a)를 TS(U_i)와 UB(T_a)중 최소 값으로 설정한다.
3. 실행이 끝난 갱신 트랜잭션이 존재한다면, 전역 검증을 위해 서버로 전송한다.

그림 4. 클라이언트 알고리즘

3.3 알고리즘 고찰

본 논문에서 제안한 알고리즘의 특징은 다음과 같다. 첫째, 전역 검증을 위해 클라이언트의 갱신 트랜잭션만 서버로 전송한다. 반면에 클라이언트의 읽기 전용 트랜잭션은 자체적으로 처리한다. 그 결과, 서버에서의 전역 단계에서 데이터 충돌을 줄일 수 있다. 둘째, CIT의 주기를 주 사이클보다 짧게 조절함으로써 마감기한을 놓치는 클라이언트의 갱신 트랜잭션 수를 줄일 수 있다. 이것은 클라이언트가 갱신 트랜잭션을 전역 검증을 위해 서버로 보내기 전에 자체적으로 데이터 충돌을 발견하고 재실행 시킬 수 있기 때문이다. 마지막으로 대기 전송 기법 또한 클라이언트가 서버와의 접촉 없이 데이터 충돌을 발견하는 것을 돕는다. 이것은 제안한 기법이 마감기한을 충족시킬 확률을 높일 수 있다는 것을 의미한다.

제안한 알고리즘에 대한 기회비용을 논의해 보겠다. 서버에서의 CIT의 주기를 줄이는 것에 대해서는 그 만큼 CIT를 위해서 대역폭을 많이 할당해야 하기 때문에 클라이언트가 데이터베이스에 있는 원본 데이터를 기다리는 대기 시간이 길어질 것이다. 이것은 클라이언트의 읽기 전용 트랜잭션의 실행 시간이 다소 길어질 수 있다는 것을 의미하고, 마감기한을 만족시키는 트랜잭션의 수가 줄어들 가능성이 있다. 클라이언트에서의 대기 전송 기법에서는 실행을 마친 갱신 트랜잭션을 서버로 바로 전송하지 않고 다음 CIT를 수신할 때 까지 대기한다. 이것은 CIT를 수신하기 위해 소모되는 대기 시간으로 인해 마감기한을 놓치는 트랜잭션이 있을 수 있다는 것을 의미한다.

4. 결론 및 향후과제

본 논문에서는 방송 디스크 환경에서 실시간 트랜잭션 처리를 위한 타임스탬프 기반 동시성 제어 기법을 제안하였다. 기존에 제안된 대부분의 동시성 제어 기법들이 클라이언트에서는 읽기 전용 트랜잭션만 실행되고 갱신 트랜잭션은 서버에서만 발생하는 상황에 제한되어 있었다. 또한 기존에 제안된 타임스탬프 기반 동시성 제어 기법은 클라이언트에서 실행되는 모든 트랜잭션을 전역 검증을 위해 서버로 전송하고 트랜잭션의 실행이 끝나는 즉시 서버로 전송함으로써 서버에서 트랜잭션간에 많은 충돌이 발생하고 그로 인해 마감기한을 놓칠 가능성이 높아진다. 본 논문에서 제안하는 동시성 제어 기법은 클라이언트의 읽기 전용 트랜잭션은 클라이언트에서 자체적으로 처리하고 갱신 트랜잭션만 서버로 전송하여 서버에서 발생하는 충돌을 클라이언트에서 미리 탐지하고 이른 시점에서 재실행하여 마감기한을 놓치는 트랜잭션을 줄일 수 있다. 본 논문의 향후 과제는 제안한 알고리즘을 구현하고 성능을 정량적으로 분석하여 기존의 알고리즘과 비교하는 것이다.

참 고 문 헌

[1] S. Acharya, R. Alonso, M. franklin, and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environment," *Proc. of ACM SIGMOD*, pp.199-210, 1995.

[2] H. Cho, "Concurrency Control for Read-Only Client Transactions in Broadcast Disks," *IEICE Trans. Commun.*, vol.E86-B, no.10, 2003.

[3] H. Garcia-Molina, G. Wiederhold, "Read-only transactions in a distributed dababase," *ACM Trans. Database Syst.*, 7, 209-234, 1982.

[4] Y. Huang and Y-H. Lee, "STUBcast - Efficient Support for Concurrency Control in Broadcast-based Asymmetric communication Environment," *Proc. 10th Int. Conf. on Computer Comm. and Networks*, pp.262-267, 2001.

[5] J. Jing, A. Heral, and A. Elmagarmid, "Client-Server Computing in Mobile Environments," *ACM Comp. Surveys*, vol.31, no.2, pp.117-157, 1999.

[6] K-Y. Lam, T-W. Kuo, *Real-Time DataBase Systems*, Kluwer Academic Publishers, 2001.

[7] V. Lee, K-W. Lam, T-W Kuo, "Efficient validation of mobile transactions in wireless environments," *The Journal of Systems and Software*, pp.183-193, 2004.

[8] V. Lee, K-W. Lam, S-H. Son, E. Chan, "On Transaction Processing with Partial Validation and Timestamp Ordering in Mobile Broadcast Environments," *IEEE Trans. on Computers*, vol.51, no.10, pp.1196-1211, 2002.

[9] V. Lee, K-W. Lam, and S-H. Son, "Concurrency Control Using Timestamp Ordering in Broadcast Environments," *The Computer J.*, vol.45, no.4, pp.410-422, 2002.

[10] J. Shanmugasundaram, A. Nithrakashyap, R. Sivasankaran, K. Ramamritham, "Efficient Concurrency Control for Broadcast Environments," *Proc. ACM SIGMOD*, pp.85-96, 1999.

[11] J-A. Stankovic, S-H. Son, J. Hansson, "Misconceptions about real-time databases," *Computer*, vol.32, no.6, pp.29-37, 1999.