방송 디스크 환경에서 읽기 전용 트랜잭션을 위한 타임스탬프 기반 동시성 제어

준회원 임 성 준*, 정회원 조 행 래**

Timestamp based Concurrency Control for Read-Only Transaction in Broadcast Disks Environment

Sungjun Lim*, Haengrae Cho** *Regular Member*

요 약

방송 디스크는 다수의 이동 클라이언트에게 정보를 전파하는 통신구조이다. 방송 디스크에서 서버는 데이터베이스에 저장된 모든 데이터를 연속적으로 방송하며, 클라이언트는 방송 채널을 감시하여 자신이 원하는 데이터를 수신한다. 이런 관점에서 방송 채널은 클라이언트가 데이터를 액세스할 수 있는 디스크의 역할을 담당한다. 본 논문에서는 서버에서 방송 데이터가 갱신될 경우, 클라이언트에서 실행되는 읽기 전용 트랜잭션의 정확성을 보장하기 위한 타임스탬프 기반 동시성 제어(Timestamp Based Concurrency Control: TCC) 기법을 제안한다. 기존에 제안된 동시성 제어 기법들은 트랜잭션의 철회율을 줄이기 위하여 추가적인 제어 정보들을 방송함으로써 방송 대역폭의 상당 부분을 소비한다는 단점을 갖는다. 이와는 달리, TCC는 방송 데이터의 타임스탬프 필드에 그 데이터를 갱신한 서버 트랜잭션들의 순서를 반영시키고, 이를 수신한 클라이언트에서는 타임스탬프를 이용하여 자신의 읽기 전용 트랜잭션의 정확성을 검사함으로써 보다 많은 트랜잭션 실행을 허용한다. 그 결과, TCC는 서버로부터 방송되는 제어 정보의 양을 최소화하면서 읽기 전용 트랜잭션의 철회율을 줄일 수 있다는 장점을 갖는다.

Key Words : mobile computing, broadcast disk, concurrency control, transaction processing.

ABSTRACT

Broadcast disks are suited for disseminating information to a large number of clients in mobile computing environments. In broadcast disks, the server continuously and repeatedly broadcasts all data items in the database to clients without specific requests. The clients monitor the broadcast channel and retrieve data items as they arrive on the broadcast channel. The broadcast channel then becomes a *disk* from which clients can retrieve data items. This paper proposes a *Timestamp based Concurrency Control* (TCC) scheme to preserve the consistency of read-only client transactions, when the values of broadcast data items are updated at the server. Previous schemes tried to reduce transaction aborts by consuming considerable amount of downlink communication from the server to clients for transferring control information. On the other hand, the TCC uses a timestamp field of each data item to describe execution order of server transactions. Clients can allow more transaction executions by checking consistency of their read-only transactions with timestamps of data items. As a result, the TCC can reduce the abort ratio of client transactions with minimal control information to be broadcast from the server.

* 삼성전자 정보통신총괄 무선사업부(slsjun@yumail.ac.kr), ** 영남대학교 전자정보공학부 (hrcho@yu.ac.kr)
논문번호 : KICS2004-07-094 접수일자 : 2004년 7월 9일
※본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성 · 지원사업의 연구결과로 수행되었습니다.

Ⅰ. 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 채널을 클라이언트가 데이터를 액세스할 수 있는 저장 장치(디스크)의 역할을 수행한다고 할 수 있다. 접속할 수 있는 클라이언트의 수에 제한이 없다는 장점으로 인해 방송 디스크는 이동 통신망을 이용한 경매나 전자 입찰과 같은 전자 상거래 응용분야와 주식 거래, 그리고 기상 정보나 교통 정보 방송 분야와 같은 다양한 응용분야에 활용되고 있다[2].

방송 디스크에서는 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[2,9,12]. 그러나 이러한 환경에서 2단계 로킹이나 낙관적인 동시성 제어와 같은 전통적인 동시성 제어 기법들을 적용시키기 힘들다. 이 기법들은 서버와 클라이언트 간에 빈번한 통신이 필요하기 때문이다[12].

본 논문에서는 방송 디스크 환경에서 클라이언트의 읽기 전용 트랜잭션을 위한 타임스탬프 기반 동시성 제어 (Timestamp Based Concurrency Control: TCC) 기법을 제안한다. 방송 디스크 환경을 위한 기존의 동시성 제어 기법들의 경우 클라이언트 트랜잭션의 불필요한 철회를 줄이는 것이 주요 목적이었다[7-10,12]. 이를 위해 일반적인 데이터 외에 방송 대역폭의 상당 부분을 차지하는 추가적인 제어 정보를 방송한다. 이와는 달리 TCC는 방송 데이터의 타임스탬프 필드에 그 데이터를 갱신한 서버 트랜잭션들의 순서를 반영시키고, 이를 수신한 클라이언트에서는 타임스탬프를 이용하여 읽기 전용 트랜잭션의 정확성을 검사함으로써 보다 많은 트랜잭션 실행을 허용한다. 그 결과, TCC는 서버로부터 방송되는 제어 정보의 양을 최소화하면서 읽기 전용 트랜잭션의 철회율을 줄일 수 있다는 장점을 갖는다.

본 논문의 구성은 다음과 같다. 2절에서 기존 동시성 제어 기법들을 살펴보고, 3절에서는 제안한 알고리즘의 동작 과정을 자세히 설명한다. 4절에서는 TCC의 성능을 평가하기 위하여 개발한 시뮬레이션 모형에 대해 설명하고 실험 결과를 분석한다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

클라이언트 트랜잭션은 방송 채널로부터 방송되는 방송 프로그램 상에서 원하는 데이터를 기다려야하며, 방송의 특성상 데이터 판독은 순차적으로 진행된다. 이 때 방송 프로그램의 각 주기를 사이클이라 한다. 대부분의 경우 클라이언트 트랜잭션은 판독할 데이터를 트랜잭션의 실행 전에 파악하는 것이 불가능하다[3]. 따라서 클라이언트 트랜잭션은 여러 사이클에 걸쳐 데이터를 액세스할 수 있으며, 그 결과 서버 트랜잭션들의 간섭으로 인해 클라이언트 트랜잭션의 실행 결과가 틀려질 수 있다. 그림 1에 잘못된 트랜잭션 실행의 예가 나타난다.

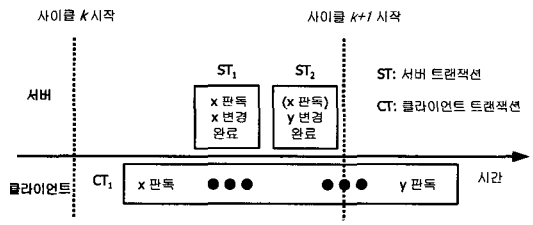

그림 1. 잘못된 트랜잭션 실행의 예.

그림 1에서 클라이언트 트랜잭션 CT_1이 데이터 x를 판독한 후 서버 트랜잭션 ST_1이 x를 변경하므로, CT_1의 실행 순서는 ST_1에 우선한다. 즉, $CT_1 \rightarrow ST_1$ 관계가 성립한다. 그리고 서버 트랜잭션 ST_2는 ST_1이 변경한 x 값을 참조하여 y를 변경하였으므로, $ST_1 \rightarrow ST_2$의 관계가 성립한다. ST_1과 ST_2는 사이클 k+1이 시작되기 전에 완료되었으므로 x, y 값이 사이클 k+1이 시작되기 전에 변경되었다. 변경된 x, y 값은 사이클 k+1부터 방송되며, CT_1이 판독하는 y 값은 ST_2에 의해 변경된 값이므로 $ST_2 \rightarrow CT_1$의 관계가 성립된다. 그 결과, 전체적으로 $CT_1 \rightarrow ST_1 \rightarrow ST_2 \rightarrow CT_1$의 실행 순서가 성립된다. 트랜잭션 실행 순서의 순환 관계가 존재하므로, CT_1의 실행은 잘못되었으며 CT_1을 철회한 후 재실행해야 한다.

방송 디스크에서 동시성 제어의 문제점은 그림 1과 같은 잘못된 트랜잭션 실행의 경우와 정확한 트

랜잭션 실행을 구분할 수 없다는 것이다. 예를 들면, 그림 1에서 ST_2가 x 값을 판독하지 않고 y 값을 변경한다면 $ST_1 \rightarrow ST_2$의 관계가 성립하지 않는다. 이 경우 순환 관계가 존재하지 않으므로 CT_1을 철회할 필요 없이 계속 실행할 수 있다. 그러나 클라이언트는 서버와의 추가적인 통신 없이는 ST_2에 대한 실행 내용을 알 수 없으며 항상 최악의 경우를 고려해야 하므로, 이 경우에도 CT_1을 철회한 후 재실행해야 한다.

방송 디스크 환경에서 이러한 문제를 해결하기 위해 동시성 제어 기법들이 제안되었다[8-10,12]. [8]의 경우, 서버 트랜잭션들의 직렬화 그래프를 데이터와 함께 방송하여 클라이언트에서 서버 트랜잭션 간의 관계(그림 1에서 $ST_1 \rightarrow ST_2$)를 파악할 수 있도록 하였다. [9], [10]에서는 데이터의 현재 값 외에 이전 값도 같이 방송함으로써 불필요한 트랜잭션의 철회율을 줄였다. [12]의 경우에는 서버가 방송 프로그램 외에 제어 행렬을 함께 방송한다. 기존에 제안된 동시성 제어 기법들의 공통점은 서버가 동시성 제어 정보를 추가로 전송하여 클라이언트 트랜잭션의 철회율을 줄인다는 것이다. 그러나 동시성 제어 정보는 데이터의 수나 서버 트랜잭션의 수에 비례하여 크기가 증가하므로, 대규모 데이터베이스 응용 분야의 경우 방송 대역폭의 상당 부분을 차지한다. 그 결과 클라이언트 트랜잭션이 방송 데이터를 판독하기 위한 대기 시간이 길어지며 전체적으로 트랜잭션의 응답 시간이 길어진다.

본 논문과 유사한 관점에서 제어 정보량을 최소화하는 기법으로 BCC-TI[6]를 들 수 있다. BCC-TI는 클라이언트에서 실행 중인 읽기 전용 트랜잭션들에 대해 타임스탬프 구간을 기록함으로써 현재의 직렬화 순서에서 읽기 전용 트랜잭션의 위치를 동적으로 조정한다. 각 사이클의 시작 시점에서 서버는 제어 정보 테이블을 방송한다. 제어 정보 테이블은 지난 사이클 동안 완료된 서버 트랜잭션의 타임스탬프와 쓰기 집합을 포함한다. 클라이언트가 제어 정보 테이블이나 데이터를 판독할 때, 서버 트랜잭션들과의 순서를 반영하기 위해 타임스탬프 구간이 조정된다. 이 때 타임스탬프의 하한계가 상한계보다 크거나 같다면 트랜잭션은 철회된다. 비록 BCC-TI가 최소의 정보량으로 트랜잭션의 철회율을 줄일 수 있으나, BCC-TI는 그림 1에서 정확한 트랜잭션의 실행을 구분하지 못한다. 그 결과 트랜잭션의 불필요한 철회를 유발한다[4].

Ⅲ. 타임스탬프 기반 동시성 제어

본 절에서는 방송 디스크 환경에서 클라이언트의 읽기 전용 트랜잭션을 효율적으로 처리할 수 있는 타임스탬프 기반 동시성 제어(Timestamp Based Concurrency Control: TCC) 기법을 제안한다. TCC는 동시성 제어를 위해 최소한의 추가 정보만을 방송함으로써 서버에서 클라이언트로의 대역폭의 대부분을 일반 데이터의 방송에 사용할 수 있다. 뿐만 아니라, 타임스탬프를 이용함으로써 클라이언트 트랜잭션의 철회율과 응답 시간을 줄일 수 있다.

A. 사이클 B_i 동안, 서버 트랜잭션 U가 완료되는 경우, 아래와 같은 과정이 수행된다.
1. TS(U)에 서버의 현재 타임스탬프를 할당한다.
2. 만약 U가 B_i동안 완료한 첫 번째 트랜잭션이라면 TS(U)를 TS(*d*), $\forall d \in$WS(U)에 할당하고 TS(U)를 변수 FIRST에 저장한다.
3. 그렇지 않다면, U가 해당 사이클에서 이전에 완료하였던 다른 서버 트랜잭션 U_c와 종속성이 존재하는지 검사한다.
   - 만약 $\{RS(U) \cup WS(U)\} \cap WS(U_c) \neq \{\}$ 이거나 $WS(U) \cap RS(U_c) \neq \{\}$이면 TS(U)를 TS(*d*), $\forall d \in$WS(U)에 할당한다.
   - 그렇지 않다면, FIRST를 TS(*d*), $\forall d \in$WS(U)에 할당한다.
4. TS(U)와 WS(U)를 CIT에 기록한다.

B. 다음 사이클 B_{i+1}의 시작 시점에서 CIT를 방송한다.

그림 2. 서버 알고리즘.

3.1 서버 알고리즘

서버 트랜잭션이 완료될 때 트랜잭션과 트랜잭션의 쓰기 집합에 존재하는 모든 데이터에 대해 타임스탬프를 할당한다. 타임스탬프 할당 방식은 그림 2와 같다. 서버 트랜잭션 U에 대해 WS(U)는 U의 쓰기 집합이고, RS(U)는 U의 읽기집합, 그리고 TS(U)는 U의 타임스탬프이고, TS(*d*)는 데이터 *d*의 타임스탬프라고 정의한다. 제어 정보 테이블(Control Information Table : CIT)은 클라이언트의 읽기 전용 트랜잭션이 직전 사이클 동안 완료한 서버 트랜잭션과 직렬화 가능한지를 검사하는데 사용된다. 이를 위해 CIT는 직전 사이클 동안 완료한 서버 트랜잭션들의 타임스탬프와 쓰기 집합을 포함한다.

서버 알고리즘의 기본 개념은 서버 트랜잭션의 완료 순서를 직렬화 순서와 구분하는 것이다. 타임스탬프는 단조증가하고 서버 트랜잭션은 완료 시점

에서 자신의 타임스탬프를 할당 받는다. 따라서 서버 트랜잭션의 타임스탬프는 그 트랜잭션의 완료 순서를 표현한다고 할 수 있다. 만약 BCC-TI 알고리즘처럼 데이터를 갱신한 트랜잭션의 타임스탬프를 데이터의 타임스탬프로 할당한다면 완료 순서와는 다른 직렬화 순서를 허용할 수 없다. 완료 순서와 직렬화 순서를 구별하기 위해서, TCC는 FIRST라는 또 하나의 타임스탬프를 사용한다. FIRST는 현재 사이클에서 첫 번째로 완료한 서버 트랜잭션의 타임스탬프 값을 나타낸다. 서버 트랜잭션 U가 완료 시, 현재 사이클에서 이전에 완료된 트랜잭션과 종속성이 존재하지 않는다면 WS(U)에 포함된 모든 데이터의 타임스탬프는 TS(U)가 아니라 FIRST를 할당한다. 그 결과 클라이언트는 다음 절에서 설명하는 알고리즘을 실행함으로써 직렬화 가능한 실행 스케줄을 보다 많이 허용할 수 있다.

3.2 클라이언트 알고리즘

TCC의 클라이언트 알고리즘이 그림 3에 나타난다. 직렬화 가능한 지를 판단하기 위하여 BCC-TI와 유사하게 TCC는 클라이언트 트랜잭션마다 타임스탬프 구간을 할당한다. 클라이언트 트랜잭션 Q에 대해서 LB(Q)는 하한계, UB(Q)는 상한계라고 정의한다. 그리고 타임스탬프 구간이 하한계가 상한계보다 커지면(LB(Q) > UB(Q)) 트랜잭션은 철회된다. 각 클라이언트 트랜잭션이 시작할 때 하한계는 0, 상한계는 ∞로 설정된다 (단계 A).

Q가 데이터 d를 읽을 때, 그림 3의 단계 B를 수행한다. 먼저 현재의 LB(Q)와 TS(d) 중에 큰 값을 LB(Q)에 할당한다. 새로운 LB(Q)가 UB(Q)보다 작은 값을 유지한다면 Q는 계속 작업을 진행할 수 있다. 만약 LB(Q)가 UB(Q)보다 크다면 Q는 직렬화 가능하지 않은 실행을 가지므로 철회된다. TCC와 BCC-TI의 차이점은 UB(Q)와 LB(Q)가 동일할 때 발생한다. BCC-TI는 UB(Q)와 LB(Q)가 동일할 때마다 트랜잭션을 철회시킨다. 그러나 TCC에서는 종속성이 존재하지 않는 서버 트랜잭션들에 의해 변경된 모든 데이터들은 동일한 타임스탬프(FIRST)를 할당한다. 이로 인해 종속성이 존재하지 않는 서버 트랜잭션들 때문에 UB(Q)와 LB(Q)의 값이 동일하다면 TCC는 클라이언트 트랜잭션 Q의 남은 작업을 진행할 수 있다. TCC는 INV라는 추가적인 정보를 유지함으로써 이러한 개념을 구현한다.

새로운 사이클의 시작 시점에서 CIT를 수신할 때, 클라이언트는 Q의 읽기 집합과 CIT에 속해 있

A. 클라이언트의 사용자가 읽기 전용 트랜잭션 Q의 실행을 요청할 경우, 클라이언트는 Q의 타임스탬프 구간을 (0,∞)으로 할당한다. 즉, LB(Q)=0, UB(Q)=∞로 설정된다.

B. Q가 데이터 d를 읽을 때 아래 과정을 수행한다.
1. RS(Q)에 d를 포함한다.
2. LB(Q) = maxium(LB(Q), TS(d)) 로 설정한다.
3. LB(Q) == UB(Q)이고 $d \in$ INV일 경우, 트랜잭션 Q는 철회되어 재실행한다.
4. LB(Q) > UB(Q)일 경우, 트랜잭션 Q는 철회되어 재실행한다.

C. 새로운 사이클이 시작되어 CIT를 수신하면 클라이언트는 INV={}로 초기화한 후 아래 과정을 수행한다.
1. CIT에 있는 모든 U_i에 대해서,
   - $WS(U_i) \cap RS(Q) \neq \{\}$인 U_i가 존재한다면, UB(Q) = minium(TS(U_i), UB(Q)) 로 설정한다.
   - UB(Q)가 변경되었다면, INV에 WS(U_i)를 포함시킨다.
2. LB(Q) > UB(Q)일 경우, 트랜잭션 Q는 철회되어 재실행한다.

그림 3. 클라이언트 알고리즘.

는 완료된 서버 트랜잭션의 쓰기 집합을 비교한다. 두 집합에서 공통된 데이터가 존재할 경우, 서버 트랜잭션이 Q를 무효화시킨 것을 의미하므로 UB(Q)와 해당 서버 트랜잭션의 타임스탬프 중에 작은 값을 UB(Q)에 할당한다. 이로 인해 UB(Q)가 변경되었다면 서버 트랜잭션의 쓰기 집합은 INV에 포함된다(단계 C.1). 단계 B.3으로 되돌아가서, d를 읽은 후 LB(Q)가 UB(Q)와 동일하다면, 클라이언트는 d가 INV에 포함되어 있는지 검사한다. 만약 INV에 포함되어 있지 않다면 d를 변경한 서버 트랜잭션과 Q를 무효화시킨 서버 트랜잭션간의 종속성이 존재하지 않는다는 것을 의미하므로 클라이언트는 남은 작업을 계속 진행시킬 수 있다.

예를 들어, 그림 1에서 ST_2가 x를 읽지 않아서 CT_1의 실행이 정확한 경우를 가정하자. 사이클 k의 시작 시점에서 TS(x)는 1이고, 사이클 k동안 ST_1이 타임스탬프 2의 시점에서 완료했다고 가정하자. 그 결과, TS(ST_1)과 TS(x)는 2로 할당되고 FIRST = TS(ST_1)으로 할당된다. ST_1과 ST_2 사이에 종속성이 존재하지 않으므로 TS(y)는 FIRST로 할당된다. 클라이언트에서는 CT_1이 x를 읽고난 후에 타임스탬프 구간이 (1, ∞)로 변경된다. 사이클 k+1의 시작 시점에서는 UB(CT_1)이 2로 변경되고 x는 INV에 포함된다. CT_1이 y를 읽을 때, 타임스탬프 구간은 (2,

2)로 변경된다. 이때 INV에 y가 포함되어 있지 않으므로 CT_1은 계속 진행할 수 있다. 만약 ST_2가 x를 읽는다면 ST_1과 ST_2 사이에 종속성이 존재함으로써 TS(y)는 TS(ST_2)가 되고 그 값은 2보다 클 것이다. 그 결과, CT_1이 y를 읽을 때 LB(Q)가 UB(Q)보다 커지므로 CT_1은 철회되고 재실행된다.

Ⅳ. 실험 결과 분석

본 절에서는 시뮬레이션을 이용하여 다양한 데이터베이스 응용 환경에서 TCC와 BCC-TI의 성능을 비교 평가한다. 먼저 실험 모형을 설명한 후, 실험 결과를 분석하도록 한다.

4.1 실험 모형

시뮬레이션 모델은 서버와 클라이언트, 그리고 방송 디스크로 구성된다[2]. 읽기 전용 트랜잭션을 위한 동시성 제어 기법은 클라이언트의 수와 관계없으므로 클라이언트는 하나라고 가정한다. 서버 트랜잭션과 클라이언트 트랜잭션 모두 데이터베이스 전체를 균일하게 판독한다. 표 1은 실험에 사용된 입력 매개 변수를 정리한 것이다. 시뮬레이션 모형은 미국의 MCC에서 개발한 CSIM 이용하여 구현하였다[11].

서버 트랜잭션은 STlength만큼 연산을 실행하고 연산이 쓰기 연산일 확률은 WriteProb이다. 클라이언트는 읽기 전용 트랜잭션을 실행한다. 클라이언트 트랜잭션이 판독하는 데이터 수는 CTlength ± CTlength × TRSizeDev 사이의 일양 분포를 따른다. 클라이언트 트랜잭션은 하나의 데이터를 판독한 후 다음 데이터를 판독하기 전에 OptDelay만큼 지연 시간을 갖는다. 그리고 클라이언트 트랜잭션이 완료되면 TranDelay만큼 지연된 후 새로운 다음 트랜잭션이 생성된다. OptDelay와 TranDelay는 지수 분포를 따른다.

표 1. 입력 매개 변수

서버 변수		
ServerDBsize	방송될 데이터의 수	300
STlength	서버 트랜잭션 길이	8
NumST	각 사이클의 서버 트랜잭션의 수	4~36
WriteProb	쓰기 연산 확률	0.0~1.0
클라이언트 변수		
CTlength	클라이언트 트랜잭션 길이	1~9
OptDelay	연산간의 지연시간	1
TranDelay	트랜잭션간의 지연 시간	2
TRSizeDev	트랜잭션 길이의 편차	0.1

각 사이클의 시작 시점에서 서버는 방송 디스크를 데이터베이스의 데이터들로 작성한다. 각 사이클은 제어 정보 테이블(CIT)과 함께 데이터베이스에 있는 모든 데이터들의 방송으로 구성된다. CIT는 이전 사이클 동안에 완료된 서버 트랜잭션의 타임스탬프와 쓰기 집합의 데이터로 구성되고, 방송 데이터는 타임스탬프를 담은 ServerDBsize 만큼의 배열을 포함하는 테이블로 구성된다. 본 실험에서는 캐쉬 효과를 고려하지 않았다. 따라서 클라이언트가 현재 사이클에서 데이터를 놓치면 다음 사이클까지 기다려야 한다.

실험에 사용된 주요 성능 지수는 철회율과 트랜잭션 응답 시간이다. 철회율은 트랜잭션이 완료하기 전까지 철회되는 횟수의 평균값이다. 트랜잭션 응답 시간은 트랜잭션이 생성된 후 완료될 때까지의 시간을 의미하며, 트랜잭션이 철회되어 재실행되는 시간도 모두 포함된다.

4.2 실험 결과

먼저 CTlength 값을 변화시키면서 성능을 비교하였다. NumST 값은 8이고 WriteProb 값은 0.5이다. 그림 4에 결과가 나타난다. 그림에서 알 수 있듯이, CTlength 값이 증가할수록 두 알고리즘 모두 성능이 나빠진다. 그 이유는 CTlength 값이 증가할수록

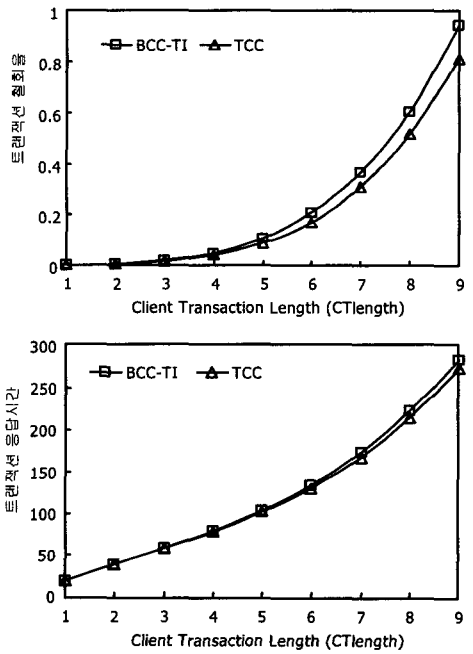

그림 4. 클라이언트 트랜잭션 길이에 따른 철회율과 응답 시간.

클라이언트가 기다려야하는 사이클의 수가 길어지고 그로인해 철회율이 증가하기 때문이다. CTlength 값이 클 때 TCC가 BCC-TI보다 성능이 좋아지는 것도 동일한 이유에 근거한다. TCC는 클라이언트가 보다 많은 직렬화 가능한 실행을 제공하기 위해서 방송될 데이터의 타임스탬프를 융통성 있게 할당함으로써 불필요한 트랜잭션 철회를 줄일 수 있다.

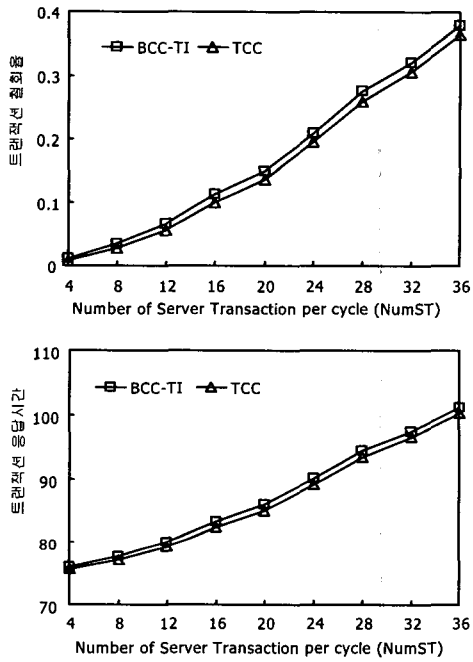

그림 5. 서버 트랜잭션 수 변화에 따른 철회율과 응답 시간.

다음은 NumST 값을 변화시키면서 성능을 비교하였다. 그림 5는 WriteProb 값이 0.5이고 CTlength 값이 4일 때 결과를 나타낸다. NumST 값이 증가함으로써 클라이언트에서 발생하는 데이터 충돌의 가능성도 증가한다. 그 결과, 철회율과 응답시간이 선형적으로 증가한다. 이전 실험과 유사하게, TCC는 BCC-TI보다 철회율과 응답 시간에 모두 향상된 성능을 보였다. 그러나 성능 차이가 큰 편은 아니었는데, 그 이유는 사이클마다 실행되는 서버 트랜잭션의 수가 많아질수록 서버 트랜잭션간의 종속성이 존재할 가능성이 커지기 때문이다.

마지막으로 WriteProb 값을 변화시키면서 성능을 비교하였다. 그림 6은 CTlength 값이 4, 5이고 NumST 값이 8일 때의 결과를 나타낸다. WriteProb 값이 증가할수록 철회율과 응답 시간이 모두 증가함을 볼 수 있다. 뿐만 아니라, WriteProb 값이 클 때 TCC가 BCC-TI보다 성능이 뛰어남을 알 수 있

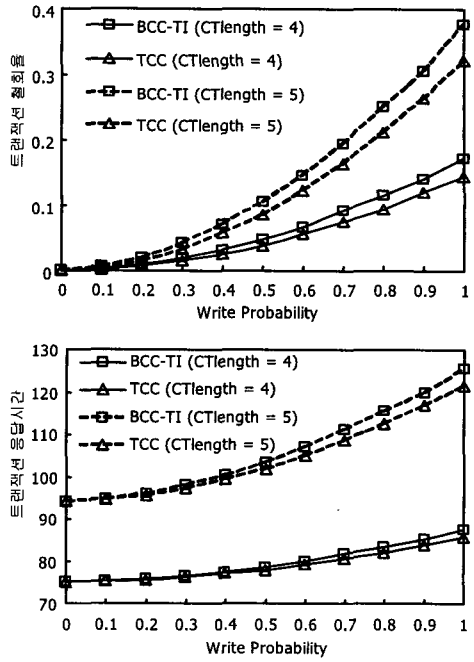

그림 6. 쓰기 확률 값의 변화에 따른 철회율과 응답 시간.

다. WriteProb 값은 클라이언트 트랜잭션과 서버 트랜잭션간의 데이터 충돌 가능성에 영향을 미친다. WriteProb 값이 클 때, 클라이언트 트랜잭션과 서버 트랜잭션 사이에 충돌이 자주 발생하고 그로 인해 트랜잭션간의 종속성 검사를 반영시켜야 할 경우가 많아진다. 이것이 CTlength 값이 5일 때, TCC의 성능 향상 정도가 더욱 커지는 이유이다.

V. 결 론

본 논문에서는 방송 디스크 환경에서 클라이언트의 읽기 전용 트랜잭션을 위한 동시성 제어 기법인 TCC를 제안하였다. 클라이언트 트랜잭션의 철회율을 줄이기 위해 제안되었던 대부분의 이전 연구들은 서버에서 제어 정보를 전송하기 위해 방송 대역폭의 상당 부분을 소비하였다. 이와는 달리, TCC는 방송되는 데이터의 타임스탬프를 융통성 있게 할당함으로써 보다 많은 직렬화 가능한 실행을 제공한다. 그 결과, TCC는 서버로부터 방송되는 제어 정보의 양을 최소화하면서 읽기 전용 트랜잭션의 철회율을 줄일 수 있다는 장점을 갖는다.

시뮬레이션을 이용하여 TCC와 기존에 제안된 BCC-TI 기법의 성능을 비교하였다. 실험 결과 TCC의 성능이 전반적으로 우수하거나 유사하게 나타났다. 특히, (1) 클라이언트 트랜잭션의 연산 수가

많고, (2) 서버에서 실행되는 트랜잭션의 수가 많거나, (3) 서버 트랜잭션의 쓰기 연산의 수가 많을 때 성능 향상의 정도가 크게 나타났다. 정보 시스템의 복잡도가 계속 증가하는 추세를 감안할 때 TCC의 이러한 성능 특성은 매우 바람직하다고 판단된다.

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.*, E86-B(10), pp.3114-3122, 2003.

[3] H. Garcia-Molina, G. Wiederhold, "Read-only transactions in a distributed database," *ACM Trans. Database Syst.*, 7(2), pp.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*, 31(2), pp. 117-157, 1999.

[6] V. Lee, K-W. Lam, and S-H. Son, "Concurrency Control Using Timestamp Ordering in Broadcast Environments," *The Computer J.*, 45(4), pp.410-422, 2002.

[7] S. Madrina, M. Mohania, S. Bhowmick, and B. Bhargava, "Mobile data and transaction management," *Inf. Sci.*, 141(3-4), pp.279-309, 2002.

[8] E. Pitoura and P. Chrysanthis, "Scalable Processing of Read-Only Transactions in Broadcast Push," *Proc. 19th Int. Conf. Distributed Comp. Syst.*, pp.432-439, 1999.

[9] E. Pitoura and P. Chrysanthis, "'Exploiting Versions for Handling Updates in Broadcast Disks," *Proc. 25th Int. Conf. VLDB*, pp.114-125, 1999.

[10] E. Pitoura and P. Chrysanthis, "'Multiversion Data Broadcast," *IEEE Trans. Computers*, 51(10), pp.1224-1230, 2002.

[11] H. Schwetmann, *User's Guide of CSIM18 Simulation Engine*, Mesquite Software, Inc. 1996.

[12] J. Shanmugasundaram et al, "Efficient Concurrency Control for Broadcast Environments," *Proc. ACM SIGMOD*, pp.85-96, 1999.

임 성 준 (Sungjun Lim) 준회원

2003년 영남대학교 전자정보공학부(공학사)
2005년 영남대학교 컴퓨터공학과(공학석사)
2005년~현재 삼성전자 정보통신총괄 무선사업부 연구원
<관심분야> Mobile Computing, 분산 데이터베이스, 성능 평가

조 행 래 (Haengrae Cho) 정회원

1988년 서울대학교 컴퓨터공학과(공학사)
1990년 한국과학기술원 전산학과(공학석사)
1995년 한국과학기술원 전산학과(공학박사)
1995년~현재 영남대학교 전자정보공학부 부교수
<관심분야> Mobile Computing, 분산 데이터베이스, 고성능 트랜잭션 처리, 성능 평가