2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

방송 디스크 환경에서 판독 전용 트랜잭션에 대한 동시성 제어

이종은*, 조행래
*영남대학교 컴퓨터공학과
e-mail:{prodigal, hrcho}@cse.yeungnam.ac.kr

Concurrency Control for Read-only Transactions in Broadcast Disks Environments

Jong-Eun Lee*, Haengrae Cho
*Department of Computer Engineering, Yeungnam University

요약

최근 인터넷 사용자의 폭발적인 증가와 방대한 클라이언트를 갖는 이동 통신 응용 분야를 위해 방송 디스크 구조가 등장하였다. 방송 디스크 구조는 유선과 무선 두 환경을 모두 지원하며, 특히 데이타에 대한 경쟁이 심한 환경에 더욱 유용하다. 본 연구는 방송 디스크 환경의 클라이언트에서 실행되는 판독 전용 트랜잭션들을 위한 동시성 제어 기법과 캐싱 기법을 제안하고, 각 클라이언트가 액세스하는 데이타의 정확성을 보장할 수 있음을 보인다.

1. 서론

현대의 클라이언트/서버 구조는 전송 데이타의 양, 대역폭, 클라이언트와 서버 측에 걸리는 부하 등에서 비대칭적인 통신 환경을 가진다. 예를 들면, 무선 이동 네트워크에서 서버들은 큰 방송 대역폭과 고성능의 처리 능력을 가지며, 클라이언트는 작은 대역폭과 부족한 에너지원, 그리고 최소한의 처리 능력을 가진다. 이러한 비대칭적 통신 환경을 위해 제안된 구조로 방송 디스크(broadcast disks)[2]를 들 수 있다. 방송 디스크 구조는 소수의 서버가 다수의 클라이언트에게 정보를 방송(dissemination)하는 구조이므로, 클라이언트의 숫자가 많아지더라도 성능 저하는 거의 없다. 방송 디스크는 서버 측에서 클라이언트가 필요로 하는 데이타들을 선정한다. 이렇게 선정된 데이타들은 방송 주기(broadcast cycle)로 불리우는 주기적 단위로 방송되고, 각 클라이언트는 방송을 모니터링한다. 클라이언트는 자신이 원하는 데이타가 방송 채널에 나타나면 획득하게 된다.

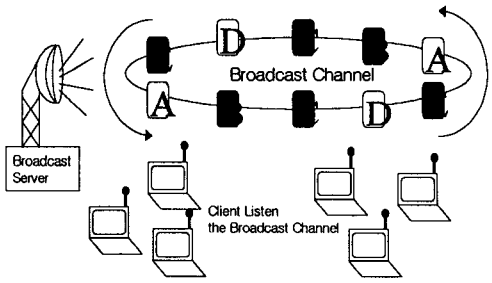

그림 1 간단한 방송 디스크 구조

그림 1에서 방송 주기는 A 데이타와 E 데이타 사이로, 방송 채널에 데이타가 2번 나타난다. 이 주기성은 방송 디스크의 장점이자 단점으로 데이타를 순차적으로 방송하기 때문에 성능 저하 없이 모든 클라이언트들이 동시에 데이타를 액세스할 수 있지만, 자신이 원하는 데이타를 얻기 위해서는 방송 채널에 데이타가 나타날 때까지 기다려야 한다. 따라서 클라이언트는 자신이 필요한 데이타를 놓치게 되면, 다음 주기에 그 데이타가 나타날 때까지 기다려야 하므로 응답 시간이 길어진다. 그 결과 응답 시간을 줄이는 것은 방송 디스크 환경의 관련 알고리즘에서 주 고려 사항이다.

본 논문에서 서버는 주기적으로 전체 데이타베이스를 방송하며, 클라이언트는 판독 전용 트랜잭션 형태로 방송된 데이타를 액세스한다고 가정한다. 그리고, 데이타베이스에 대한 갱신은 서버에서만 이루어진다고 가정하며, 갱신 트랜잭션에 대해서는 고려하지 않는다. 또한, 방송되는 데이타들은 서버에서 완료된 갱신 트랜잭션들이 생성한 값들로 이루어지고 클라이언트에서는 방송 채널의 데이타를 읽기만 한다고 가정한다. 클라이언트는 방송 구조에 대해 사전 지식이 없으며, 갱신된 값의 방송 시기는 방송 주기 혹은 그보다 작은 단위로 이루어진 유효 구간(currency interval) 단위로 반영되어 방송된다. 따라서 유효 구간 중에는 값의 변경이 없기 때문에 판독 트랜잭션은 이 값으로 서버와의 접촉 없이 데이타를 액세스할 수 있다.

방송 디스크에서의 동시성 제어는 클라이언트에게 지원되는 서버로의 대역폭이 매우 낮다는 점에 유의하여 설계하여야 한다. 예를 들면, 방송 디스크 환경에 이 단계 로킹을 적용할 경우, 판독 연산에 대해서도 로크가 필요하다. 따라서, 수많은 클라이언트가 서버와 통신을 하여 서버를 작동 불능 상태로 만들 수 있다. 뿐만 아니라, 직렬화 가능성을 유지하기 위해 트랜잭션을 불필요하게 철회시킬 수 있고, 그 결과 성능의 지대한 감소를 가져온다. 그러므로 기존의 직렬화 가능성을 위한 동시성 제어 기법은 사용하기 어려우며, 방송 환경에 적용 가능한 새로운 동시성 제어 기법이 필요하다.

Datacycle[6,7]은 방송 환경에서 제안된 최초의 동시성 제어 기법으로 기본 개념은 판독 전용 트랜잭션이 실행되는 동안 액세스한 데이타가 갱신 트랜잭션에 의해 갱신되었을 경우 판독 트랜잭션을 철회한다. Datacycle은 직렬화 가능성을 보장하지만, 판독 트랜잭션의 빈번한 철회로 인해 성능이 저하된다. F-Matrix[9]는 Datacyle의 단점인 빈번한 트랜잭션 철회를 줄이기 위하여 제안된 방법으로, 서버는 방송 주기마다 동시성 제어 정보를 추가로 전송한다. 동시성 제어 정보는 모든 데이타 쌍들에 대해 각 데이타를 최근에 갱신한 트랜잭션들간의 관계를 표현한다. 동시성 제어 정보를 이용하여 클라이언트는 불필요한 트랜잭션 철회를 줄일 수 있지만, 제어 정보를 계산하고 방송하기 위한 서버의 오버헤드가 커진다는 단점을 갖는다. F-Matrix와 동일하게 동시성 제어 정보를 전송하는 방법으로 직렬화 그래프 테스트 기법[8]을 들 수 있는데, 갱신 트랜잭션들간의 직렬화 그래프를 클라이언트에게 전송하여 판독 전용 트랜잭션과의 직렬화 가능성이 위배될 때 판독 전용 트랜잭션을 철회하는 것이 주요 개념이다. 그러나, 이 기법은 직렬화 그래프를 유지하고 전송하기 위한 오버헤드가 존재한다. 클라이언트 트랜잭션의 빈번한 철회를 막기 위해 서버에서 다중 버전을 전송하는 방법도 제안되었으나[9], 다중 버전의 전송으로 방송 주기가 길어지는 단점을 가진다.

본 논문의 구성은 다음과 같다. 2절에서는 본 연구에서 제안한 판독 전용 트랜잭션을 위한 동시성 제어 기법에 대해 설명한다. 3절에서는 캐싱을 고려한 동시성 제어 기법의 알고리즘을 설명하고. 4절에서 결론을 맺는다.

2. 판독 전용 트랜잭션을 위한 동시성 제어 기법

본 논문의 목적은 다중 버전이나 대규모의 동시성 제어 정보를 사용하지 않고 클라이언트 트랜잭션의 철회율을 줄이는 것이다. 단, 동시성 제어를 위하여 각 데이타마다 타임스탬프가 할당되며, 방송 주기의 시작 시점에서 무효화 메시지가 방송된다고 가정한다. 데이타 d에 할당된 타임스탬프는 d를 가장 나중에 갱신한 트랜잭션이 완료한 시점을 나타낸다. 무효화 메시지는 방송 주기사이에 갱신된 데이타들의 리스트로 구성되며, 이동 통신 환경에서 캐쉬 일관성 제어를 위해서도 사용될 수 있다. 타임스탬프와 무효화 메시지를 사용한 동시성 제어 방법들은 다음과 같다.

2.1 타임스탬프를 이용한 방법

[9]에서 제안한 방법으로 클라이언트 트랜잭션 T_c가 처음으로 판독 연산을 실행한 유효 구간을 v_0라 가정한다. 이 후 T_c가 액세스하는 데이타 d에 대해 d의 타임스탬프가 v_0보다 작을 경우 액세스를 허용하며, d의 타임스탬프가 클 경우에는 T_c를 철회한다. [9]와 같이 다중 버전이 방송되는 경우에는 T_c가 이전 데이타를 액세스할 가능성이 있으나, 본 논문과 같이 최신 데이타만 전송될 경우에는 T_c의 철회 가능성이 매우 높아진다.

2.2 무효화 메시지를 이용한 방법

[6]과 [9]에서 제안한 방법으로 클라이언트는 실행 중인 판독 트랜잭션 T에 대해 T가 지금까지 액세스한 데이타의 집합 RS(T)를 각각 유지한다. 그리고 서버는 직전의 방송 주기동안 서버에서 갱신된 데이타들의 식별자를 포함하는 무효화 메시지 INV를 새로운 방송 주기의 시작 시점에서 방송한다. INV를 전송받은 클라이언트는 현재 실행 중인 트랜잭션 T_c에 대해 "$RS(T_c) \cap INV$"를 계산하고, 결과가 공집합이 아닐 경우 T_c를 철회한다.

2.3 타임스탬프와 무효화 메시지를 모두 이용한 방법

본 논문에서 제안한 방법으로 클라이언트 트랜잭션은 액세스한 데이타가 이미 무효화된 "invalid" 상태와 아직 무효화되지 않은 "valid" 상태로 나누어진다. 트랜잭션의 초기 상태는 "valid"이며, "valid" 상태의 트랜잭션 T_{valid}는 기존의 타임스탬프 기법과 달리 최신의 데이타들을 액세스할 수 있다. 무효화 메시지 INV를 전송받은 클라이언트는 현재 실행 중인 T_{valid}에 대해 "$RS(T_{valid}) \cap INV$"를 계산하고, 결과가 공집합이 아닐 경우 T_{valid}를 "invalid" 상태의 트랜잭션 $T_{invalid}$로 변경한다. $T_{invalid}$의 실행 방식은 타임스탬프 기법과 유사하다. 즉, $T_{invalid}$의 타임스탬프 $TS(T_{invalid})$는 현재 방송 주기의 시작 시점으로 설정되며, $T_{invalid}$는 $TS(T_{invalid})$보다 작은 타임스탬프를 갖는 데이타를 액세스하는 동안은 계속 실행될 수 있다. 그러나, $TS(T_{invalid})$보다 큰 타임스탬프를 갖는 데이타를 액세스할 경우 $T_{invalid}$는 철회된다.

클라이언트 트랜잭션 T_c에 대해 T_c 이전에 완료된 서버 트랜잭션을 T_{old}라 하며, T_c가 시작된 후 완료한 서버 트랜잭션을 T_{new}라 하자. 타임 스탬프 기법의 경우 직렬화 순서를 $T_{old} < T_c < T_{new}$로 고정함으로써 직렬화 가능성을 만족한다. 무효화 기법은 $T_{new} < T_c$를 허용하지만, $T_c < T_{new}$는 금지함으로써 직렬화 가능성을 만족한다. 이에 대해 제안한 기법은 T_{new}를 T_c가 시작된 후 무효화될 때까지 완료한 서버 트랜잭션 T_{after}와 T_c가 무효화된 후 완료한 서버 트랜잭션 T_{final}로 분리한 후, $T_{old} < T_{after} < T_c < T_{final}$의 순서로 직렬화 가능성을 만족시킨다. 그 결과, $T_{after} < T_c$를 허용함으로써 타임스탬프 기법에 비해 T_c의 철회 가능성을 줄일 수 있으며, $T_c < T_{final}$을 허용함으로써 무효화 기법에 비해 T_c의 철회 가능성을 줄일 수 있다.

3. 캐싱을 고려한 동시성 제어

클라이언트 트랜잭션의 응답 시간을 줄이기 위하여, 클라이언트들은 빈번히 액세스하는 데이타를 지역적으로 캐싱할 수 있다. 본 논문에서는 캐싱의 단위를 페이지로 가정하며, 페이지는 방송 데이타의 기본 단위인 버킷과 동일한 단위로 가정한다.

2절에서 제안한 타임스탬프와 무효화를 이용한 동시성 제어 기법의 경우, 기존 기법에 비해 트랜잭션 철회율이 줄어들기 위해서는 클라이언트 트랜잭션이 무효화된 후 최신 데이타를 액세스할 가능성을 줄여야 한다. 클라이언트 트랜잭션의 철회율을 줄일 수 있는 한 가지 방법은 각 클라이언트의 캐쉬를 최신 캐쉬(C_{new})와 이전 캐쉬(C_{old})의 2가지 종류로 구성하는 것이다. C_{new}에는 최신의 데이타가 캐싱되고, 최신 캐쉬에 저장된 데이타가 무효화될 때 해당 데이타는 C_{old}로 이동하게 된다. 캐싱을 이용한 동시성 제어의 기본 개념은 무효화되지 않은 트랜잭션들은 C_{new}나 방송 데이타를 이용하여 최신의 데이타를 액세스하도록 하고, 무효화된 트랜잭션들은 C_{old}를 액세스하여 트랜잭션 철회율을 줄인다는 것이다.

캐쉬 관리 알고리즘을 구체적으로 설명하면 다음과 같다. 단, 현재 무효화된 트랜잭션의 수는 $N(T_{invalid})$로 표시한다.

1. 초기에는 모든 캐쉬 데이타를 C_{new}에 저장하며, 캐쉬 데이타의 상태는 "new"로 초기화된다.
2. 무효화 메시지를 참조하여 C_{new}에 저장된 데이타가 무효화되었을 경우, 해당 데이타의 상태를 "old"로 변경한다.
3. "old" 상태인 데이타가 방송되었으며 $N(T_{invalid}) = 0$인 경우, 기존 캐쉬 데이타는 삭제하며 새로운 방송 데이타를 C_{new}에 캐싱한다.
3. "old" 상태인 데이타가 방송되었으며 $N(T_{invalid}) \neq 0$인 경우, C_{new}에 저장된 데이타를 C_{old}로 이동하며 방송 데이타는 C_{new}에 캐싱한다.
4. $N(T_{invalid}) = 0$인 경우, C_{old}에 저장된 모든 데이타를 삭제한다.

그림 3 캐쉬 관리 알고리즘

위 알고리즘에 나타나듯이 C_{old}는 무효화 트랜잭션이 존재할 경우, 무효화 트랜잭션이 이전 데이타를 액세스할 가능성을 증가시키기 위해 사용된다. 그리고, 무효화 트랜잭션이 존재하지 않을 경우에는 캐쉬 효율을 증가시키기 위해 모든 캐쉬 공간을 C_{new}에 할당한다. C_{new}와 C_{old}를 이용한 동시성 제어 알고리즘은 다음과 같다. 단, 무효화되지 않은 트랜잭션은 T_{valid}로 표현하며, 무효화된 트랜잭션은 $T_{invalid}$로 표현한다.

1. T_{valid}는 "new" 상태의 C_{new} 데이타나 방송 데이타를 액세스한다.
2. T_{valid}가 액세스한 데이타가 무효화 메시지에 포함될 경우, T_{valid}는 $T_{invalid}$로 변경되며 $T_{invalid}$의 무효화 시점 $IT(T_{invalid})$은 현재 방송 주기의 시작 시점으로 설정된다.
3. $T_{invalid}$가 새로운 데이타 d를 액세스하려고 할 경우:
   3.1 d가 C_{new}나 C_{old}에 캐싱되지 않을 경우, 방송 디스크에서 d를 액세스한 후 d의 타임스탬프 < $IT(T_{invalid})$인 경우 방송 디스크의 d를 액세스하며; 클 경우에는 $T_{invalid}$를 철회.
   3.2 C_{new}에 저장된 d의 타임스탬프 < $IT(T_{invalid})$인 경우 C_{new}의 d를 액세스한다. 클 경우, C_{old}를 검사하며 C_{old}에 d가 없으면 $T_{invalid}$를 철회.
   3.3 C_{old}에 저장된 d의 타임스탬프 < $IT(T_{invalid})$인 경우 C_{old}의 d를 액세스한다. 클 경우, $T_{invalid}$를 철회.

그림 4 동시성 제어 알고리즘

4. 결론

본 논문에서는 방송 디스크 환경의 클라이언트에서 실행되는 판독 전용 트랜잭션들을 위한 동시성 제어 기법을 제안하였다. 제안한 기법은 방송 데이타에 포함된 타임스탬프와 무효화 메시지를 이용하여 트랜잭션 철회율을 줄일 수 있으며, 다중 버전이나 대규모 동시성 제어 정보를 이용하는 기존의 동시성 제어 기법에 비해 방송 주기를 줄일 수 있다는 장점을 갖는다. 뿐만 아니라, 본 논문에서는 제안한 동시성 제어 기법의 철회율 및 트랜잭션 응답 시간을 줄일 수 있는 캐싱 알고리즘을 제안하였다. 본 논문의 향후 연구 과제는 다양한 시스템 구성과 데이타베이스 부하에서 제안한 동시성 제어 기법 및 캐싱 알고리즘의 성능을 분석하는 것이다.

참고문헌

[1] S. Acharya, R. Alonso, M. Franklin, S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communications Environment," in: Proc. ACM SIGMOD Conf. (1995) 199-210.

[2] S. Acharya, M. Franklin, S. Zdonik, "Dissemination-based Data Delivery Using Broadcast Disks," IEEE Personal Communications, 2(6) (1995).

[3] S. Acharya, M. Franklin, S. Zdonik, "Prefetching from a Broadcast Disk," in: Proc. 12th ICDE (1996) 276-287.

[4] D. Aksoy, and et al, "Research in Data Broadcast and Dissemination," in: Proc. 1st Int. Conf. on Advanced Multimedia Content Processing (1998).

[5] P. Bober and M. Carey, "Multiversion Query Locking," in: Proc. VLDB Conf. (1992) 497-510.

[6] T. Bowen, and et al., "The Datacycle Architecture," CACM 35(12) (1992) 71-79.

[7] G. Herman, G. Gopal K. Lee, A. Weinrib, "The Datacycle Architecture for Very High Throughput Database System," in: Proc. ACM SIGMOD Conf. (1987) 97-103.

[8] E. Pitoura and P.K. Chrysanthis, "Scalable Processing of Read-Only Transactions in Broadcast Push," in: Proc. 19th ICDCS (1999) 432-439.

[9] E. Pitoura and P.K. Chrysanthis, "Exploiting Versions for Handling Updates in Broadcast Disks," in: Proc. VLDB Conf. (1999) 114-125.

[9] J. Shanmugasundaram, and et al., "Efficient Concurrency Control for Broadcast Environments," in: Proc. ACM SIGMOD Conf. (1999) 85-96.