방송 디스크 환경에서 합득 전용 트랜잭션에 대한 동시성 제어

이종은*, 조형래
*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 사용자들의 폭발적인 증가와 방대한 클라이언트를 갖는 이동 통신용 운영 분야를 위해 방송 디스크 구조가 등장하였다. 방송 디스크 구조는 유선과 무선 두 환경을 모두 지원하며, 특히 데이터에 대한 경쟁이 심한 환경에 더욱 유용하다. 본 연구는 방송 디스크 환경의 클라이언트에서 실행되는 합득 전용 트랜잭션을 위한 동시성 제어 기법과 개선 기법을 제안하고, 각 클라이언트가 엑세스하는 데이터의 정확성을 보장할 수 있음을 보인다.

1. 서론

현재의 클라이언트/서버 구조는 전송 데이터의 양, 대역폭, 클라이언트와 서버 측에 걸리는 부하 등에 서 비대칭적인 통신 환경을 가진다. 예를 들어, 무선 이동 네트워크에서 서버들은 큰 방송 대역폭과 고성능의 처리 능력을 가지만, 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 응답이 많아지더라도 성능 저하는 거의 없다. 방송 디스크는 서버 측에서 클라이언트가 필요로 하는 데이터들을 선정한다. 이렇게 선정된 데이터들은 방송 주기(broadcast cycle)로 클라이언트의 주기적 단위로 방송되고, 각 클라이언트는 방송을 모니터링한다. 클라이언트는 자신이 원하는 데이터가 방송 채널에 나타나면 획득하게 된다.

그림 1 간단한 방송 디스크 구조

그림 1에서 방송 주기는 A 데이터와 E 데이터 사이로, 방송 채널에 데이터가 2번 나타난다. 이 주기 중, 방송 채널에 데이터가 2번 나타난다는 점에서 방송 주기는 전송 데이터를 순차적으로 방송하기 때문에 성능 저하 없이 모든 클라이언트들에 데이터를 엑세스할 수 있지만, 자신이 원하는 데이터를 얻기 위해서는 방송 채널에 데이터가 나타날 때까지 기다려야 한다. 따라서 클라이언트는 자신의 필요한 데이터를 놓치게 되며, 다음 주기에 그 데이터가 나타날 때까지 기다려야 하므로 옵티컬 시간이 걸린다. 그 결과 응답 시간이 줄이는 것은 방송 디스크 환경의 관련 알고리즘에서 주 고려 사항이다.
본 논문에서 서비는 주기적으로 전체 데이터베이스를 방송하며, 클라이언트는 판독 전용 트랜잭션 형태로 방송된 데이터를 액세스한다. 그리고, 데이터베이스에 대한 갱신은 서버에서만 이루어지고 가정하며, 갱신 트랜잭션에 대해서는 고려하지 않는다. 또한, 방송되는 데이터들은 서버에서 완료된 갱신 트랜잭션들이 생성한 결과들로 이루어지고 클라이언트에서는 방송 채널의 데이터를 잊기만한다고 가정한다. 클라이언트는 방송 구조에 대해 사전 지식이 없으면서 갱신된 값의 방송 시간은 방송 주기 혹은 그보다 작은 단위로 이루어진 유료 구간(interval) 단위로 변동된다. 따라서 유료 구간 중에는 갱신이 없기 때문에 판독 트랜잭션은 이 값을 서버의 연속적인 값에 따라 데이터에 액세스할 수 있다.

방송 디스크에서의 동시성 제어는 클라이언트에게 지원되는 서버의 대역폭이 매우 높다는 점에 의존하여 설계된다. 예를 들어, 방송 디스크 환경에서 각 단계 로그를 저장할 경우, 판독 연산이 여기에도 로그가 필요하다. 따라서, 수많은 클라이언트가 서버와 통신을 하여 서버를 작동 불능 상태로 만들 수 있다. 뿐만 아니라, 적절한 갱신을 유지하기 위해 트랜잭션을 불필요하게 첨부시킬 수 있고, 그 결과 성능 저하와 감소를 가져온다. 그러므로 기존의 적절한 갱신을 위한 동시성 제어 기법은 사용하기 어려우며, 방송 환경에 적응 가능한 새로운 동시성 제어 기법이 필요하다.


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

2. 판독 전용 트랜잭션을 위한 동시성 제어 기법

본 논문의 목표는 주기적이고 대규모의 동시성 제어 정보를 사용하지 않고 클라이언트 트랜잭션의 철회율을 줄이는 것이다. 단, 동시성 제어를 위하여 각 데이터마다 타임스탬프가 할당되며, 방송 주기가 시작 시점에서 무효화 되지가 방송된다고 가정한다. 타임스탬프는 d를 가장 나중에 갱신된 트랜잭션에 이용한 시점을 나타낸다. 무효화 미시지는 방송 주기 사이에 갱신된 데이터들의 리스트로 구성되며, 이동 통신 환경에서 캐시 작동성 제어를 위해서도 사용될 수 있다. 타임스탬프가 무효화 미시지를 사용한 동시성 제어 방법들은 다음과 같다.

2.1 타임스탬프를 이용한 방법

[9]에서 제안한 방법으로 클라이언트 트랜잭션 T가 처음으로 판독 연산을 실행한 유료 구간을 V0라 가정한다. 이 후 T가 액세스하는 데이터 d에 대해 d의 타임스탬프가 V0보다 작을 경우 액세스를 허용하며, d의 타임스탬프가 V0 경우에는 T를 철회한다. [8]과 같이 다중 비전이 방송되는 경우에는 T가 첫 데이터에 액세스할 가능성이 있으나, 본 논문과 같이 최신 데이터만 전송될 경우에는 T의 철회 가능성이 매우 높아진다.
2.2 무효화 메시지 이용한 방법


2.3 타임스탬프와 무효화 메시지를 모두 이용한 방법

본 논문에서 제안한 방법으로 클라이언트 트랜잭션은 액세스한 데이터가 이미 무효화된 "invalid" 상태와 아직 무효화되지 않은 "valid" 상태로 나뉘어진다. 트랜잭션의 초기 상태는 "valid"이며, "valid" 상태의 트랜잭션 T이면 T의 기존의 타임스탬프 기법과 달리 최신의 데이터를 액세스할 수 있다. 무효화 메시지 INV를 전송받은 클라이언트는 현재 실행 중인 T에 대해 "RS(Tvalid) ∩ INV"를 계산하고, 결과가 공집합이 아닐 경우 T를 "invalid" 상태의 트랜잭션 Tvalid로 변경한다. Tvalid의 실행 방식은 타임스탬프 기법과 유사하다. 즉, Tvalid의 타임스탬프 TS(Tvalid)는 현재 방송 주기의 시작 시점으로 설정되며, TS(Tvalid)보다 큰 타임스탬프를 갖는 데이터를 액세스하는 동안은 계속 실행될 수 있다. 그러나, TS(Tvalid)보다 큰 타임스탬프를 갖는 데이터를 액세스할 경우 Tvalid는 체결된다.

클라이언트 트랜잭션 T에 대해 T의 이전에 완료된 서버 트랜잭션을 Told라 하며, T가 시작된 후 완료한 서버 트랜잭션은 Tnew라 하자. 타임스탬프 기법의 경우 직류화 순서는 Told < T < Tnew로 고정함으로써 직류화 가능성을 만족한다. 무효화 기법은 Tnew < T를 허용하지만, T < Tnew는 금지함으로써 직류화 가능성을 만족한다. 이에 대해 제안한 기법은 Tnew를 T가 시작된 후 무효화될 때까지 완료한 서버 트랜잭션 Tafter와 T가 무효화될 후 완료한 서버 트랜잭션 Tfinal로 분리한 후, Told < Tafter < T < Tfinal의 순서로 직류화 가능성을 만족시킨다. 그 결과, Tafter < T를 허용함으로써 타임스탬프 기법에 비해 T의 체결 가능성을 즐일 수 있으며, T < Tfinal을 허용함으로써 무효화 기법에 비해 T의 체결 가능성을 즐일 수 있다.

3. 캐시를 고려한 동시성 제어

클라이언트 트랜잭션의 응답 시간을 줄기기 위하여, 클라이언트들은 반복적으로 액세스하는 데이터를 지역적으로 개신할 수 있다. 논문에서는 캐시의 단위를 페이지로 가정하며, 페이지는 방송 데이터의 기본 단위인 버킷과 동일한 단위로 가정한다.

2장에서 제안한 타임스탬프와 무효화를 이용한 동시성 제어 기법의 경우, 기존 기법에 비해 트랜잭션 체결율이 증가하기 위해서는 클라이언트 트랜잭션의 무효화 후 최신 데이터를 액세스할 가능성을 줄여야 한다. 클라이언트 트랜잭션의 체결율을 증가시키려는 방법은 다음과 같다. (1) 각 클라이언트의 캐시를 최신 캐시(Cnew)와 이전 캐시(Cold)의 2 가지 종류로 구분하는 것이다. Cnew는 최신의 데이터가 저장된 캐시로, Cold는 저장된 데이터가 무효화될 때 해당 데이터는 Cold로 이동하게 된다. (2) Cnew에 저장된 데이터를 액세스하도록 하고, 무효화된 트랜잭션은 Cold로 액세스하여 트랜잭션 체결율을 증가시킨다.

캐시 관리 알고리즘을 구체적으로 설명하면 다음과 같다. 단, 현재 무효화된 트랜잭션의 수는 N(Tinvalid)로 표시한다.

1. 초기에는 모든 캐시 데이터를 Cnew에 저장하며, CHashSet의 상태는 "new"로 초기화된다.
2. 무효화 메시지를 주기적으로 Cnew에 저장된 데이터가 무효화되었을 경우, 해당 데이터의 상태를 "old"로 변경한다.
3. "old" 상태의 데이터가 방송되었으며 N(Tinvalid) = 0인 경우, 기존 캐시 데이터는 삭제하며 새로운 방송 데이터를 Cnew에 저장한다.
4. "old" 상태의 데이터가 방송되었으며 N(Tinvalid) = 0인 경우, Cnew에 저장된 데이터를 Cold로 이동하며 방송 데이터를 Cnew에 저장한다.

그림 3 캐시 관리 알고리즘
위 알고리즘에 나타나듯이, C_{old}는 무효화 트랜잭션 존재할 경우, 무효화 트랜잭션이 이전 데이터를 액세스할 가능성을 증가시키기 위해 사용된다. 그리고, 무효화 트랜잭션 존재하지 않을 경우에는 캐시 효율을 증가시키기 위해 모든 캐시 공간을 C_{new}에 할당한다. C_{new}와 C_{old}를 이용한 동시성 제어 알고리즘은 다음과 같다.

1. T_{valid}는 "new" 상태의 C_{new} 데이터를 방송 데이터를 액세스한다.
2. T_{valid}가 액세스한 데이터가 무효화 메시지에 포함될 경우, T_{valid}는 T_{invalid}로 변경되며 T_{invalid}의 무효화 시점 IT(T_{invalid})는 현재 방송 주기의 시작 시점으로 설정된다.
3. T_{invalid}가 새로운 데이터 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 관독 전용 트랜잭션을 위한 동시성 제어 기법을 제안하였다. 제안한 기법은 방송 데이터에 포함된 타임스탬프와 무효화 메시지를 이용하여 트랜잭션 학회의 중지될 수 있으며, 다양한 버전이나 대규모 동시성 제어 방법을 이용하는 기존의 동시성 제어 기법에 비해 방송 주기를 줄일 수 있다는 장점을 갖는다. 뿐만 아니라, 본 논문에서는 제안한 동시성 제어 기법의 활용을 위한 트랜잭션 응답 시간을 줄일 수 있는 캐싱 알고리즘을 제안하였다. 본 논문의 향후 연구 과제는 다양한 시스템 구성과 데터바이스 스키마에서 제한된 동시성 제어 기법 및 캐싱 알고리즘의 성능을 분석하는 것이다.

참고문헌