클라이언트-서버 환경에서 동적 갱신을 이용한 탐지기반의 캐쉬 일관성 알고리즘

김치연* · 정종면*

A Detection-based Cache Consistency Algorithm using Dynamic Update in Client-Server Environments

Chi-yeon Kim* · Jong-myeon Jeong*

요 약

클라이언트-서버 환경에서 클라이언트는 캐쉬에 데이터를 저장함으로써 서버와 접촉하지 않고도 응용을 수행할 수 있다. 클라이언트에서 수행되는 응용의 정확성을 보장하기 위해서는 캐쉬의 일관성 유지가 필요하고, 이를 위해 탐지 기반의 방법과 회피 기반의 방법이 사용되어져 왔다. 이 논문에서는 서버에 비하여 자원이 빈약한 클라이언트의 부담을 덜어줄 수 있는 탐지 기반의 새로운 캐쉬 일관성 유지 방법을 제안한다. 제안하는 방법에서는 타임스탬프를 이용하여 트랜잭션의 동시성을 제어하며, 캐쉬를 갱신할 때 갱신된 데이터의 삭제와 덮어쓰기를 동적으로 수행함으로써 캐쉬를 효율적으로 관리한다. 또한 1-사본 직렬성을 이용하여 제안하는 방법의 정확성을 증명하였다.

ABSTRACT

In client-server environments, clients can execute applications without contact a server by storing data in its cache. To guarantee correction of applications, we need a cache consistency algorithm. Many cache consistency algorithms have been proposed, these were categorized by detection-based and avoidance-based algorithms. In this paper, we propose a new detection-based cache consistency algorithm that can reduce a burden of a client that has poor resource compare with a server. Our method controls concurrency of transactions using timestamp ordering and updates a cache dynamically. In addition to we show that our method maintains a one-copy serializability.

키워드

캐쉬, 회피 기반, 탐지 기반, 타임스탬프, 타당성 검사

Ⅰ. 서 론

분산 클라이언트-서버 시스템에서 서버는 안정된 저장장치에 데이터를 유지하며 트랜잭션을 관리한다. 클라이언트에서는 각종 응용 프로그램을 수행하며, 각 응용에 필요한 데이터를 서버로부터 공급받는다. 클라이언트-서버 시스템에서는 서버와 클라이언트의 역할 분담을 통하여 응용을 수행함으로써 서버의 부하를 덜 수 있다. 또한, 데이터를 클라이언트에 위치시킴으로써 매번 서버에 접근하는 방법에 비하여 접근 비용과

* 목포해양대학교 해양전자 · 통신공학부

접수일자 : 2005. 5. 10

시간을 절감할 수 있다는 장점이 있다.

이러한 장점을 최대화하기 위해 클라이언트는 캐쉬를 사용한다. 클라이언트에서 수행되는 응용에서 데이터에 대한 접근이 발생할 때마다 서버가 아닌 캐쉬를 접근함으로써 클라이언트와 서버 사이의 메시지 교환을 줄일 수 있어 성능을 향상시킬 수 있고 전체 데이터베이스 시스템의 확장을 용이하게 한다.

클라이언트 캐싱은 두 가지 유형으로 분류할 수 있다[1]. 첫 번째 유형은 intra-transaction 캐싱인데, 이는 한 트랜잭션의 수행 동안에만 캐쉬에 데이터를 유지하는 방법이다. 즉, 하나의 트랜잭션이 수행되는 동안에만 캐쉬된 데이터가 유지되며 새로운 트랜잭션을 시작할 때 캐쉬는 비워진다. 두 번째 유형은 inter-transaction 캐싱인데, 트랜잭션의 종료 여부에 상관없이 데이터가 캐쉬에 유지된다. intra-transaction 캐싱은 inter-transaction 캐싱보다 간단하지만 새로운 트랜잭션이 수행될 때마다 데이터를 새로 얻어야 하는 단점이 있다. 따라서 inter-transaction 캐싱이 더 일반적이라고 할 수 있다. 하지만 inter-transaction 캐싱에서는 서버에 유지되는 데이터와 클라이언트에 캐싱된 데이터 사이의 연속적인 일관성이 유지되어야 하며, 이를 위하여 캐쉬 일관성 프로토콜을 필요로 한다.

지금까지 제안된 캐쉬 일관성 프로토콜은 탐지 기반(Detection-based)과 회피 기반(Avoidance-based)으로 분류될 수 있다. 탐지 기반은 트랜잭션 수행동안 캐쉬 내에 얼마간의 비일관된 데이터를 허용하되, 트랜잭션 완료 전 접근한 데이터의 타당성을 검증하여 완료 여부를 결정하는 방법이다. 회피 기반은 트랜잭션이 비일관된 데이터를 접근할 기회를 전혀 갖지 않는다. 이 방법에서 서버와 클라이언트의 캐쉬에 유지되는 데이터는 항상 일치해야 한다. 이를 위하여 ROWA(Read-One Write-All) 프로토콜을 사용한다. 회피 기반 프로토콜은 탐지 기반 프로토콜에 비하여 클라이언트 소프트웨어가 복잡하다.

이 논문에서는 갱신이 드문 환경에서 탐지 기반의 새로운 캐쉬 일관성 알고리즘을 제안한다. 제안하는 알고리즘은 탐지 기반이므로 회피 기반에 비하여 클라이언트가 갖는 부담이 적다. 데이터는 트랜잭션이 시작하기 전 타당성을 검증받기 위해 서버에 보내지며 트랜잭션은 검사 결과를 기다리지 않고 바로 수행을 시작한다. 이런 특징 때문에 제안하는 방법은 갱신이 드문 환경에 적합하다. 트랜잭션 수행 도중 다른 클라이언트에서 완료된 갱신 트랜잭션의 결과가 도착하면 타임스탬프 순서대로 처리하여 수행중인 트랜잭션의 완료 여부가 결정된다. 또한 전달된 갱신에 대하여 캐쉬된 데이터의 삭제나 덮어쓰기가 동적으로 결정되는데, 이는 데이터의 성질에 따라 결정된다. 새롭게 제안하는 방법은 탐지 기반이므로 회피 기반에 비하여 서버와 교환하는 메시지 수가 적고, 동적 갱신을 사용함으로써 삭제나 덮어쓰기만을 사용하는 방법에 비하여 캐싱의 장점을 최대한 살릴 수 있다.

논문의 구성은 다음과 같다. 2장에서는 캐쉬 일관성 유지 알고리즘의 연구 동향과 문제점에 대하여 기술하고, 3장에서는 새롭게 제안하는 캐쉬 일관성 알고리즘을 기술한다. 4장에서는 제안하는 알고리즘의 분석과 결론을 기술한다.

Ⅱ. 관련 연구

이 장에서는 캐쉬 일관성을 다룬 연구들에서 제안한 방법들에 대하여 기술한다.

캐쉬 일관성 알고리즘은 클라이언트가 타당하지 않은 데이터의 접근을 제어하는 방법에 따라 회피 기반과 탐지 기반으로 분류할 수 있다[1].

회피 기반의 캐쉬 일관성 방법은 일관되지 않은(stale) 데이터의 접근을 불가능하게 함으로써 일관성을 유지하는 방법이다. 캐싱은 서버와 클라이언트 사이의 중복이라고 볼 수 있다. 따라서 이들 사이의 완벽한 일치를 위해서는 ROWA 프로토콜을 사용해야 한다. 서버와 클라이언트의 데이터 일치를 위하여 서버는 모든 사본의 위치를 유지해야 한다. 이 방법은 클라이언트에서 데이터를 접근하는 모든 시점에서 일관성이 유지되어야 하므로 비관적 알고리즘으로 분류하기도 한다. 이 방법은 서버보다 클라이언트 측의 복잡도를 증가시킨다.

ACBL(Adaptive CallBack Locking) 알고리즘은 동기적 회피 기반의 알고리즘이다[2]. 이 방법은 inter-transaction 캐싱을 사용하며 잠금(lock)을 사용하여 트랜잭션의 동시성을 제어한다. 이 방법은 회피 기반 알고리즘이므로 일관되지 않은 데이터를 접근하지 않으나, 철회와 관련된 교착 상태가 발생한다는 문제점이

있다.

위의 ACBL 방법을 변형시킨 알고리즘에서는 서버와 클라이언트에서 각기 다른 레벨의 잠금을 관리하고 잠금의 충돌이 발생할 때 교착상태를 탐지한다[3]. 클라이언트의 잠금 모드는 전용(private)과 공유(shared) 모드로 분류하고, 기록을 위한 배타적 잠금 없이도 데이터를 수정할 수 있다.

CBL(CallBack Locking) 방법의 한 변형으로 두 개의 버전을 사용한 회피 기반의 2V-CBL 알고리즘[4]에서는 트랜잭션의 잠금을 세분화하여 동시성을 높임으로써 AACC(Asynchronous Avoidance-based Cache Consistency) 방법보다 성능이 나음을 보였다.

탐지 기반의 캐쉬 일관성 유지 방법의 특징은 회피 기반의 알고리즘에 비하여 단순하다는 점이다. 정확하지 않은 데이터가 얼마간 클라이언트에 존재하는 것이 허락되며, 트랜잭션의 종료 전에 접근한 데이터의 타당성을 검사하여 완료 여부를 결정한다. 따라서 갱신이 잦은 환경이라면 탐지 기반의 성능은 떨어지게 된다. 이런 관점에서 낙관적 방법으로 분류되기도 한다.

탐지 기반의 일관성 알고리즘은 데이터 타당성 검사 시점, 갱신의 통지 시점, 갱신 전달 방법 등에 따라 세분하게 되는데, AOCC(Adaptive Optimistic Concurrency Control) 방법[5][6]의 경우, 데이터의 타당성 검사는 트랜잭션의 완료 직전에 수행하고, 갱신의 통지는 갱신 트랜잭션의 완료 후에, 그리고 갱신 전달 방법으로 전파와 무효화를 동적으로 사용한 알고리즘을 제안하였다. 데이터의 타당성 검사를 트랜잭션의 완료 직전까지 지연할 수 있는 것은 낙관적 가정에 근거한 것이다. 동시성 제어의 정확성을 위해 직렬성과 외부 일관성을 사용하였고, 각 객체에 대하여 하나의 버전만을 유지하였다. 이 방법의 가장 큰 문제점은 데이터의 타당성 검사가 트랜잭션의 수행이 끝나고 완료 직전에 이루어짐에 따라, 접근한 데이터의 타당성 검사에 실패할 경우, 하나 이상의 트랜잭션의 늦은 철회를 야기시키고 작업의 낭비를 가져온다는 점이다.

CBL(CallBack Locking) 방법은 [7]에서 처음 제안되었고, THOR 객체 지향 데이터베이스에 알고리즘을 적용한 결과를 보였다.

탐지 기반의 캐쉬 일관성 유지 알고리즘은 회피 기반의 알고리즘에 비하여 단순하여 구현이 쉽다는 장점을 갖는다. 또한 서버에 비하여 자원이 빈약한 클라이언트의 부하를 덜어줄 수 있는 방법이기도 하다. 따라서 이 논문에서 탐지 기반의 새로운 캐쉬 일관성 알고리즘을 제안하고자 한다.

Ⅲ. 제안하는 방법

이 장에서는 새롭게 제안하는 캐쉬 일관성 유지 방법에 대하여 기술한다. 제안하는 방법은 탐지 기반의 캐쉬 일관성 유지 방법으로, 트랜잭션의 동시성을 제어하기 위해 타임스탬프 순서화(Timestamp Ordering) 기법을 사용한다. 제안하는 방법의 가장 큰 특징은 데이터 갱신에 대하여 전파와 무효화를 동적으로 사용함으로써 캐쉬 실패와 잦은 메시지 요구를 줄일 수 있다는 점이다. 제안하는 방법의 정확성을 증명하기 위한 기준으로는 중복 환경에서 주로 사용되는 1-사본 직렬성(1-copy serializability)을 사용한다. 1-사본 직렬성은 캐쉬를 사용하는 환경과 같이 데이터가 중복된 환경에서 가장 일반적으로 사용하는 정확성 기준이다[8]. 편의를 위해 제안하는 방법을 EDDU (an Efficient Detection-based Dynamic Update) 알고리즘이라 명명한다.

1. 시스템 모델 및 가정

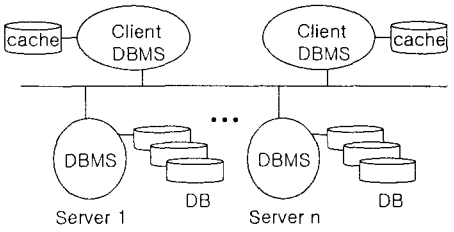

그림 1. 시스템 모델
Fig. 1 System Model

제안하는 방법이 적용되는 시스템 모델은 그림 1과 같고, 제안하는 알고리즘에서 사용된 가정은 다음과 같다.

- 서버들 사이에서 데이터는 정적으로 분포된다. 즉, 서버들 사이에 데이터 중복은 없다.

- 데이터의 가장 작은 단위는 페이지이다.
- 트랜잭션 사이에 충돌이 많지 않다.
- 클라이언트에서는 한 번에 하나씩의 트랜잭션만 수행된다.

2. EDDU 알고리즘

▶ 타임스탬프 할당

클라이언트-서버 환경에서, 서버들은 유선으로 안정적인 연결을 유지하고 있다고 가정할 수 있다. 타임스탬프 순서화 기법을 적용하기 위해서는 클라이언트에 제출되는 트랜잭션에 유일한 타임스탬프를 할당할 수 있어야 하고, 이를 위해서는 서버와 클라이언트들의 클럭 동기화가 필요하다. 하지만 서버와 클라이언트 사이의 클럭 동기화는 통신 지연 등으로 인하여 달성하기에 쉽지 않은 제약조건이다. 따라서 제안하는 방법에서는 서버와 클라이언트 사이의 클럭을 동기화하지 않고, 클라이언트 트랜잭션이 데이터 타당성 검사를 위해 서버에 전달되었을 때 서버의 클럭을 이용하여 타임스탬프를 부여하는 방법을 사용한다. 타임스탬프를 부여받기 위해 트랜잭션이 제출된 서버를 홈 서버라고 정의한다. 이렇게 함으로써 얻을 수 있는 이점은, 서버와 클라이언트의 클럭 동기화를 위해 필요한 부가적인 메시지 교환을 줄일 수 있고, 타당성 검사와 타임스탬프 할당 프로세스를 하나로 합쳐 알고리즘이 더욱 간단해질 수 있다는 점이다. 각 클라이언트에서 제출된 트랜잭션에 유일한 타임스탬프를 할당하기 위해 각 클라이언트들은 클라이언트 식별자를 함께 보낸다. 서버에서는 현재 클럭과 클라이언트 식별자를 조합하여 타임스탬프를 부여한다. 가정에 의해 한 클라이언트에서 동시에 두 개의 트랜잭션이 제출되지 않기 때문에 모든 트랜잭션에 유일한 타임스탬프를 부여할 수 있다. 이 방법의 정확성은 [9]에서 증명되었다.

▶ 타당성 검사

클라이언트에 트랜잭션이 제출되고 트랜잭션이 시작되기 전, 트랜잭션이 접근하는 모든 데이터에 대한 타당성 검사가 시작된다. 타당성 검사는 캐쉬에 저장되어 있는 데이터 중 트랜잭션이 접근하는 데이터가 유효한(가장 최신의) 데이터인지를 검사 받는 절차이다. 현재 캐쉬에 유지되고 있는 데이터가 최신의 데이터가 아니면 타당성 검사에 실패하게 되고 수행중인 트랜잭션은 철회된다. 접근할 데이터에 대하여 타당성 검사를 의뢰한 후, 클라이언트는 결과를 기다리지 않고 연산의 수행을 바로 시작한다. 또한, 트랜잭션이 접근하는 데이터가 캐쉬에 존재하지 않았을 경우에는 타당성 검사와 데이터 요구를 함께 보낸다. 클라이언트는 현재 소속되어 있는 서버에 존재하지 않는 데이터도 홈 서버를 경유하여 캐쉬할 수 있다.

트랜잭션의 수행 도중 언제라도 클라이언트에 타당성 검사 결과가 도착할 수 있다. 타당성 검사에 성공하면 연산의 수행을 계속하지만, 실패하면 수행중인 트랜잭션은 즉시 철회된다.

▶ 갱신 통지서

클라이언트에서 트랜잭션을 수행하는 도중, 언제라도 다른 클라이언트에서 완료된 갱신 트랜잭션의 결과가 전달될 수 있다. 이것을 갱신 통지서(notification)라고 한다[1]. 이 기술은 ROWA에서 빌려온 기술인데, 캐쉬의 최신성을 가능한 한 유지하기 위해 사용한다. 클라이언트에서 갱신 트랜잭션이 완료되면 홈 서버에 먼저 알린다. 갱신 통지서가 홈 서버에 전달되면 이웃한 서버들로 먼저 전달되고, 각 서버에 홈을 두고 있는 클라이언트들에게도 네트워크 지연만큼 후에 전달된다.

▶ 갱신의 전파

데이터의 갱신이 각 클라이언트에 전달되면, 클라이언트에서는 갱신된 데이터를 캐쉬에 반영하기 위한 액션을 취한다. 일반적으로 갱신을 처리하기 위한 방법은 무효화(invalidation)와 전파(propagation)가 있다. 무효화는 갱신된 데이터를 단순히 캐쉬로부터 삭제하는 방법이고, 전파는 새로운 값으로 덮어쓰는 방법이다. 지금까지 제안된 방법들은 모두 한 가지 방법만을 사용하고 있으나 데이터의 성질에 따라 무효화와 전파를 동적으로 사용하면 캐쉬의 효율을 높일 수 있다. 전파는 캐쉬내의 값을 갱신하고 데이터를 여전히 유지하게 되므로 자주 접근되는 데이터에 적용하고, 무효화는 데이터를 삭제하게 되므로 상대적인 접근 빈도가 낮은 데이터에 대하여 적용한다. 하지만 접근 빈도가 아닌 다른 기준을 사용하여 전파와 무효화를 결정할 수도 있다. 이것은 클라이언트의 정책에 따라 결정할 수 있

다. 각 클라이언트에서 사용하는 hot-spot 결정 알고리즘에 따라 hot-spot이라면 전파를, 아니라면 무효화를 적용한다. 이렇게 함으로써 중요한 데이터는 캐쉬에 유지하면서 가능한 한 최신의 값으로 유지하고, 반면에 중요한 데이터가 아니라면 삭제되므로 다른 데이터가 캐쉬될 수 있고 결과적으로 서버와의 메시지 교환을 줄일 수 있다.

갱신이 전파되었을 때 갱신된 데이터와 충돌 관계의 연산을 수행중인 트랜잭션이 존재할 수 있다. 즉, 데이터 x에 대한 갱신이 전달되었을 때 x를 읽거나 쓰는 트랜잭션이 수행중인 경우, 수행 중인 트랜잭션의 타임스탬프가 더 작다면 철회하지 않고 수행을 계속한다. 이렇게 함으로써 잠금을 사용한 방법에 비하여 트랜잭션의 철회율을 줄일 수 있다.

▶ 완료 프로토콜

트랜잭션의 마지막 연산이 성공적으로 끝나면 트랜잭션은 완료 여부를 결정하기 위해 예비완료(precommit) 상태로 들어간다. 이 상태에서 클라이언트는 자신보다 작은 타임스탬프를 가진 갱신 트랜잭션이 먼저 완료되었는지를 서버에 문의한다. 만약 그렇다면 트랜잭션은 완료할 수 없다. 그렇지 않다면 트랜잭션은 완료할 수 있고, 데이터의 타임스탬프를 갱신한다. 갱신 트랜잭션의 경우는 완료 후 홈 서버에 결과를 알려 다른 서버와 클라이언트에 갱신이 전달되도록 한다.

▶ 시나리오

다음은 EDDU 알고리즘을 적용한 예이다.

서버 S1에는 데이터 항목 x, y, z가, 서버 S2에서는 a, b, c가, 서버 S3에는 d, e, f가 각각 저장되어 있다고 가정한다. S1의 클라이언트 C1의 캐쉬에는 데이터 항목 x, y가 캐쉬되어 있으며, S2의 클라이언트 C2의 캐쉬에는 a, b가 캐쉬되어 있다. 또한 S3의 클라이언트의 C3에는 데이터 항목 d, x가 캐쉬되어 있다고 가정한다. 이 때, T1, T2, T3이 C1, C2, C3에서 각각 제출된다고 가정한다.

T1 : r1(x) r1(y) r1(z)
T2 : r2(a) r2(b) r2(y)
T3 : w3(x)

T1의 타임스탬프 ts(T1) = 18.1
T2의 타임스탬프 ts(T2) = 20.2
T3의 타임스탬프 ts(T3) = 10.3

Client 3에서 T3은 가장 먼저 제출되어 수행된다. 완료 후 홈 서버를 경유하여 다른 서버로 갱신 통지서가 보내진다. Client 1에서 T1은 캐쉬에 없는 데이터 z에 대한 요구와 함께 서버에 타당성 검사를 의뢰하고 수행을 계속한다. 수행 도중 타당성 검사 결과가 도착하고 수행을 계속한다. Client 2에서 T2가 제출되고, 타당성 검사와 함께 데이터 y를 홈 서버를 경유하여 요청한다. T1의 마지막 연산이 끝난 후, w3(x)에 대한 갱신 통지서가 Client 1에 도착하고 r1(x)와 w3(x) 연산이 충돌관계이나 ts(T1) > ts(T3)이므로 T1은 아무 문제없이 precommit 상태로 들어가게 된다. T2 연산 수행 도중 타당성 검사 결과와 함께 데이터 y가 전달되면 T2는 precommit 상태로 들어가고 문제없이 완료하게 된다. 이 때, T1이나 T2의 타당성 검사 결과가 부정적이거나, 갱신 통지된 트랜잭션의 타임스탬프가 수행중인 트랜잭션의 타임스탬프보다 크면 트랜잭션은 철회된다. □

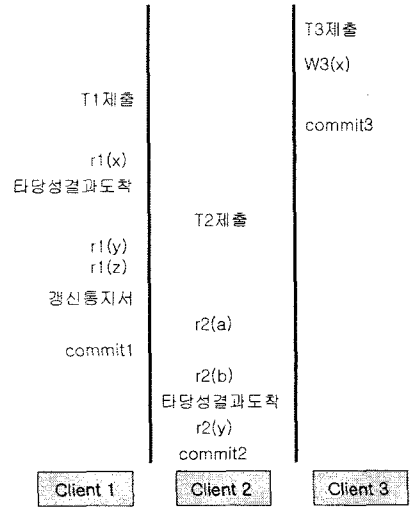

그림 2. EDDU 시나리오
Fig. 2 A EDDU Scenario

위의 내용을 알고리즘으로 정리하면 다음과 같다.

▶ 클라이언트 알고리즘

```
1. A transaction T is submitted;
2. sends access_set of T to home server;
3. executes each operation;
4. if (!validation_check)  // 검사 실패
   aborts the active transaction;
5. if (a notification) // 갱신통지서
   5.1 if (ts(notification) < ts(active_tr))
          aborts the active transaction;
       else {
          if (data(noti) is hot_spot)
             installs new value;
          else invalidates data;
       }
6. when last operation of T was finished
   T enters precommit state;
   6.1 waits decision of server;
   6.2 if (decision == commit)
          commits T;
       else aborts T;
```

▶ 서버 알고리즘

```
1. if (a client requests a validation check)
      check requested data;
2. assigns a timestamp to T;
3. if (a T enters precommit) {
   3.1 if (T can commit) {
       sends a commit_decision;
       3.1.1 if (Committed T includes a write op)
                sends updates other servers;
             else sends a abort_decision;
       3.1.2 updates data timestamp with ts(T);
```

[정리 1] EDDU 알고리즘은 1-사본 직렬가능(1CSR)하다.

1-사본 직렬성은 모든 i, j에 대하여 Ti가 Tj로부터 read-from관계에 있을 때, i=j이거나, Tj는 x의 임의 버전에 기록한 Ti에 선행하는 마지막 트랜잭션일 때 만족된다.

제안하는 알고리즘에서는 클라이언트에서 한 번에 하나의 트랜잭션만 수행되고, 뒤에 제출되는 트랜잭션의 타임스탬프는 이전 트랜잭션보다 크므로 한 클라이언트 내의 트랜잭션은 직렬가능한 순서로 수행됨을 보장할 수 있다.

판독 연산들은 어떤 순서로 수행되어도 상관없다. 갱신 연산을 수행하기 위해서는 서버에서 타당성 검사와 더불어 접근하는 데이터의 마지막 판독 타임스탬프와 기록 타임스탬프보다 자신의 타임스탬프가 커야 한다. 문제는 갱신 트랜잭션의 타임스탬프보다 작은 판독 트랜잭션이나 갱신 트랜잭션이 수행중이었을 때 발생하게 된다.

i) 갱신 트랜잭션과 판독 트랜잭션의 관계를 살펴본다. 갱신 트랜잭션보다 작은 타임스탬프의 판독 트랜잭션이 수행중이었고 갱신 트랜잭션이 먼저 완료하게 되는 경우, 판독 트랜잭션이 예비완료 상태에 들어가거나 갱신 통지서를 받으면 알고리즘에 의해 철회되기 때문에 직렬화 순서에 위배되는 스케줄이 만들어지지 않는다. 갱신 트랜잭션이 판독 트랜잭션보다 나중에 완료되는 경우는 타임스탬프 순서에 위배되지 않으므로 직렬가능한 스케줄을 만들 수 있다.

ii) 갱신 트랜잭션과 갱신 트랜잭션의 관계에서도 위의 경우와 동일하다.

i)과 ii)의 두 경우 모두 타임스탬프 순서에 위배되는 경우가 없으므로 EDDU 알고리즘에 의해 만들어지는 히스토리 H에는 사이클이 없고, 직렬가능한 히스토리만을 생산함을 증명할 수 있다. □

Ⅳ. 결 론

클라이언트-서버 환경은 클라이언트와 서버 사이의 상호 작용을 통하여 응용을 수행한다. 클라이언트-서버 환경의 구조적 장점을 최대한 이용하려면 서버에 집중되어 있는 데이터를 클라이언트에 분산시키는 것이 한 가지 방법이 된다. 이를 위해 클라이언트는 캐쉬를 사용하고, 자주 접근하는 데이터를 캐쉬에 저장함으로써 트랜잭션의 응답 시간과 서버와의 메시지 전송을 줄일 수 있다.

캐쉬에는 서버 데이터의 사본이 유지되므로 캐쉬와 서버 사이의 일관성 유지가 중요하다. 이를 위해 탐지 기반의 알고리즘과 회피 기반의 알고리즘이 제안되어져 왔다. 탐지 기반은 낙관적 가정하에 연산을 먼저 수

행하고 데이터의 타당성 결과에 따라 완료 여부를 결정하고, 회피 기반은 항상 서버와 캐쉬의 데이터가 동일하게 유지되는 방법이다. 하지만 회피 기반은 탐지 기반보다 알고리즘이 복잡하고 서버의 부담이 많다.

이 논문에서는 탐지 기반을 사용한 새로운 캐쉬 일관성 알고리즘을 제안하였다. 제안하는 방법은 잠금이 갖는 잠금 관리 부담을 덜고 트랜잭션의 철회율을 줄이기 위해 타임스탬프 순서화 기법을 사용하여 트랜잭션을 제어하였다. 갱신이 드문 환경의 장점을 최대한 살리기 위해 타당성 검사의 결과를 기다리지 않고 연산을 바로 시작하여 응답 시간을 줄일 수 있도록 하였다. 갱신이 드물긴 하지만 전혀 일어나지 않는 건 아니다. 다른 클라이언트에서 데이터가 갱신되면 홈 서버는 다른 서버로 전달하고, 각 서버들은 클라이언트들에게 갱신을 전달한다. 클라이언트들은 전달된 갱신 통지서에 대하여 캐쉬의 데이터를 점검하고 전파와 무효화를 동적으로 결정하여 캐쉬 실패를 최소화할 수 있도록 하였다. 마지막으로 제안하는 알고리즘이 직렬성을 유지함을 증명하였다.

여기에 추가하여 앞으로 제안한 알고리즘의 성능을 증명할 수 있는 모의실험을 통하여 알고리즘의 효율성을 증명하고자 한다. 주요한 매개변수는 트랜잭션의 응답 시간과 철회율이 될 것이다.

참고문헌

[1] M. J. Franklin, M. J. Carey, and M. Livny, "Transactional Client-Server Cache Consistency : Alternatives and Performance," ACM Transactions on Database Systems, Vol. 22, Num. 3, pp 315-363, 1997.

[2] A. Adya, R. Gruber, B. Liskov, and U. Maheshwari, "Efficient Optimistic Concurrency Control Using Loosely Synchronized Clocks," ACM SIGMOD Conference, pp. 23-34, June, 1995.

[3] M. T. Ozsu, K. Voruganti, and R. C. Unrau, "An Asynchronous Avoidance-based Cache Consistency Algorithm for Client Caching DBMSs," Proceedings of the 24th VLDB Conference, pp. 440-451, 1998.

[4] 강흠근, 민준기, 전석주, 정진완, "클라이언트-서버 DBMS 환경에서 콜백 잠금 기반 다중 버전의 활용," 정보과학회 논문지, Vol. 31, Num. 5, pp. 457-467, Oct. 2004.

[5] M. J. Franklin, M. Carey, "Client-server Caching Revisited," Proceedings International Workshop in Distributed Object Management, pp. 57-78, MAY 1992.

[6] B. Liskov, M. Castro, L.Shrira, and A. Adya, "Providing Persistent Objects in Distributed Systems," Proceedings of the 13th European Conference on Object-Oriented Programming, pp. 230-257, 1999.

[7] Y. Wang and L. Rowe, "Cache Consistency and Concurrency Control in a Client/Server DBMS Architecture," ACM SIGMOD Conference, pp. 367-376, June, 1991.

[8] P. A. Bernstein, V. Hadzilacos, and N. Goodman, "Concurrency Control and Recovery in Database Systems," Addison-wesley, 1987.

[9] 김치연, 황부현, "이동 트랜잭션의 완료율 향상을 위한 다중버전 타임스탬프 순서화 스케줄링 기법", 정보처리논문지 D, Vol. 6, Num. 5, pp. 1143-1152, 1999.

저자소개

김치연(Chi-yeon Kim)

1992년 2월 전남대학교 전산통계학과 (이학사)

1994년 2월 전남대학교 대학원 전산통계학과 (이학석사)

1999년 8월 전남대학교 대학원 전산통계학과 (이학박사)

2002년 3월 ~ 현재 목포해양대학교 해양전자통신공학부 교수

※ 관심분야 : 이동 컴퓨팅, 트랜잭션 관리, 전자상거래

정종면(Jong-Myeon Jeong)

1992년 2월 한양대학교 전자계산학과(공학사)
1994년 8월 한양대학교 전자계산학과(공학석사)
2001년 2월 한양대학교 전자계산학과(공학박사)
2001년 3월 ~ 2004년 2월 한국전자통신연구원 디지털방송연구단 선임연구원
2004년 3월 ~ 현 재 목포해양대학교 해양전자통신공학부 전임강사
※관심분야: 전자상거래, 데이터 방송, 컴퓨터 비전 및 영상처리, 멀티미디어 응용