한국항행학회 논문지 제10권 제2호 2006년 6월

불필요한 콜백을 제거한 회피 기반의 캐쉬 일관성 알고리즘

An Avoidance-Based Cache Consistency Algorithm without Unnecessary Callback

김치연*

Chi-Yeon Kim*

요 약

클라이언트의 데이터 캐슁은 클라이언트에서 데이터가 캐쉬되고 운영되는 환경에서 서버와의 상호작용을 하기 위한 중요한 기술이다. 캐슁은 네트워크 지연을 감소시키고 클라이언트의 자원 활용을 증가시키는 방법이다. 이와 같은 클라이언트-서버 환경에서 클라이언트에서 수행되는 응용 프로그램의 정확성을 보장하기 위해서는 캐쉬 일관성 알고리즘이 필요하다. 이 논문에서는 새로운 비동기적 회피 기반의 캐쉬 일관성 알고리즘을 제안한다. 제안하는 방법에서는 잠금 모드를 단순하게 유지하고, AACC에서 잠금 상승으로 인하여 발생하는 불필요한 callback 메시지를 제거함으로써 AACC보다 적은 메시지 교환을 가짐을 보였다. 제안하는 알고리즘의 정확성을 증명하기 위해서는 1-사본 직렬성을 사용하였다.

Abstract

Data-caching of a client is an important technology for interaction with a server where data are cached and operated at the client node. Caching reduces network latency and increases resource utilization at the client. In a Client-server environment, the absence of cache consistency algorithm, we cannot guarantee the correctness of client applications. In this paper, we propose a new asynchronous avoidance-based cache consistency algorithm. We remove the additional callback due to the lock escalation in AACC. With a comprehensive performance analysis, we show that the proposed algorithm has less message exchange than the AACC. One-copy serializability is used for proving correctness of the proposed algorithm.

Key words : 캐쉬, 회피 기반의 캐쉬 일관성, callback, locking, 직렬성

I. 서 론

클라이언트-서버 데이터베이스 관리 시스템(DBMS : DataBase Management System) 구조에서 서버와 클라이언트의 상호작용은 query-shipping과 data-shipping 두 가지 방법으로 분류할 수 있다. query-shipping은 클라이언트가 서버로 질의를 보내면 서버가 질의를 실행하여 결과를 클라이언트에게 보내주는 방식이고, data-shipping은 질의를 처리하기 위한 데이터를 클라이언트에 유지하여 클라이언트가 질의를 처리하는 방식이다. query-shipping의 방식은 관계형 DBMS에서 행해지는 일반적인 방법인데 비해, data-shipping은 객체지향 DBMS의 전형적인 방법이라 할 수 있다[1].

* 목포해양대학교 해양전자통신공학부 (Division of Marine Electronic & Comm. Engineering, Mokpo National Maritime University)
· 제1저자 (First Author) : 김치연
· 접수일자 : 2006년 5월 2일

data-shipping의 장점은 서버의 작업부하를 많이 줄일 수 있다는 점과 데이터를 응용 프로그램에 최대한 가깝게 위치시켜 효율적인 데이터 조작을 할 수 있다는 점이다.

data-shipping을 사용할 때 클라이언트의 데이터 캐슁은 서버로부터 얻어야 하는 데이터의 양을 감소시켜 성능을 향상시키고 데이터베이스 시스템의 확장성을 향상시키는 중요한 기술이며, 성능의 최적화를 위해 일반적으로 받아들여지는 기술이다[2]. 클라이언트 캐슁은 두 가지 유형으로 분류할 수 있다. 첫 번째 유형은 intra-transaction 캐슁인데, 이는 한 트랜잭션의 수행 동안에만 캐쉬에 데이터를 유지하는 방법이다. 즉, 하나의 트랜잭션이 수행되는 동안에만 캐쉬된 데이터가 유지되며 새로운 트랜잭션이 종료되면 캐쉬는 비워진다. 두 번째 유형은 inter-transaction 캐슁인데, 트랜잭션의 종료 여부에 상관없이 클라이언트가 데이터를 삭제할 때까지 캐쉬에 유지된다. intra-transaction 캐슁은 inter-transaction 캐슁보다 간단하지만 새로운 트랜잭션이 수행될 때마다 데이터를 새로 얻어야 하는 단점이 있다. 따라서 inter-transaction 캐슁이 더 일반적이라고 할 수 있다. 하지만 inter-transaction 캐슁에서는 서버에 유지되는 데이터와 클라이언트에 캐슁된 데이터 사이의 연속적인 일관성이 유지되어야 하며, 이를 위하여 캐쉬 일관성 프로토콜을 필요로 한다. 그림 1은 캐쉬를 사용하는 클라이언트-서버 환경의 시스템 모델이다.

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.

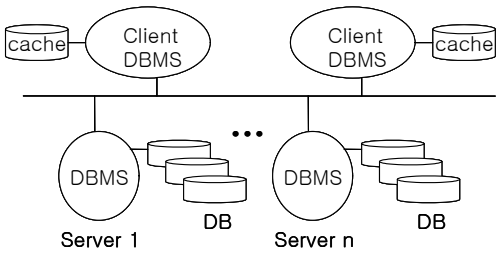


그림 1. 클라이언트-서버 환경의 시스템 모델
Fig. 1. System model of client-server environments.

이 논문에서는 판독 연산과 갱신 연산으로 구성된 일반적인 트랜잭션 수행되는 클라이언트-서버 환경에서 회피 기반의 새로운 알고리즘을 제안한다. 제안하는 방법은 비동기적 방법으로 서버의 응답을 기다리지 않고 트랜잭션의 수행을 바로 시작하므로 동기적 방법에 비하여 응답 시간이 짧다. 트랜잭션의 동시성 제어를 위해 잠금(locking)을 사용하는데, 캐쉬에 존재하지 않은 데이터에 대한 잠금을 얻는 과정과 트랜잭션 완료 후 보유한 잠금을 처리하는 방법에서 불필요한 callback을 제거함으로써 동일 환경에서 제안된 AACC(Asynchronous Avoidance-based Cache Consistency) 방법에 비해 메시지 교환을 감소시킬 수 있다.

이 논문의 구성은 다음과 같다. Ⅱ장에서는 캐쉬 일관성 알고리즘의 분류와 기 제안된 알고리즘에 대하여 분석하고, Ⅲ장에서는 새롭게 제안하는 회피 기반의 캐쉬 일관성 알고리즘을 기술한다. Ⅳ장에서는 제안하는 알고리즘과 AACC 방법과의 성능 비교를 다루며, Ⅴ장에서는 결론을 기술한다.

Ⅱ. 관련 연구

클라이언트-서버 환경에서 캐쉬 일관성을 다룬 알고리즘들을 몇 가지 기준으로 분류해보면 표 1과 같다.

표의 세로 축 기준은 타당하지 않은(invalid) 데이터의 접근을 방지하는 방법에 따른 분류이다. 회피 기반은 stale 데이터의 접근 자체를 처음부터 막는 방법이고, 탐지 기반은 stale 데이터의 접근을 허락하되, 나중에 비일관적인 접근은 걸러내는 방법이다. 회피 기반은 서버보다는 클라이언트의 부하가 크며,

탐지 기반은 알고리즘이 간단하다는 장점이 있다.

표 1. 캐쉬 일관성 알고리즘의 분류
Table 1. A classification of cache consistency algorithms.

구분	Synchronous	Asynchronous	Deferred
Avoidance-based	**ACBL[3]**	**AACC[4]**	O2PL[5]
Detection-based	C2PL[5]	NWL[6]	**AOCC[7]**

가로 축의 분류 기준은 클라이언트가 갱신 의도를 서버에 처음 알렸을 때 서버와의 상호작용 방식에 따른 분류이다. 동기적 방법은 서버로부터 명확한 응답이 주어진 후에 트랜잭션의 연산을 시작하는 방법이며, 비동기적 방법은 서버의 응답을 기다리지 않고 연산의 수행을 바로 시작하는 방법이다. 지연된 방법은 트랜잭션의 수행을 모두 끝내고 완료 직전에 갱신 요구를 하는 방법이다. 이 절에서는 이들 중 대표적인 알고리즘인 ACBL(Adaptive CallBack Locking)], AACC, AOCC(Adaptive Optimistic Concurrency Control) 및 2V-CBL(2-Version CallBack Locking)[8]방법에 대하여 살펴본다.

• ACBL : 이 방법은 inter-transaction 캐싱을 사용하며 잠금을 사용하여 트랜잭션의 동시성을 제어한다. 이 방법은 회피 기반 알고리즘이므로 일관되지 않은 데이터를 접근하지 않으나, 교착상태의 발생 가능성이 있다.

• AACC : 비동기적 방법으로 ACBL 방법을 변형시킨 알고리즘이다. 서버와 클라이언트에서 각기 다른 레벨의 잠금을 관리하고 잠금의 충돌이 발생할 때 교착상태를 탐지하는 알고리즘을 수행한다. 클라이언트의 잠금 모드를 전용 판독(private-read)과 공유 판독(shared-read) 모드, 기록(write)모드로 분류하였고, 충돌이 발생할 때는 callback을 사용하여 잠금의 허가 여부를 결정하였다. 캐쉬에 없는 데이터를 처음으로 캐쉬할 때 일단 전용 판독 모드로 잠금을 얻고 기록을 위한 기록 잠금을 나중에 callback을 이용하여 잠금 상승을 함으로써 불필요한 callback 메시지가 발생한다.

• AOCC : 데이터의 타당성 검사는 트랜잭션의 완료 직전에 수행하고, 갱신의 통지는 갱신 트랜잭션의 완료 후에, 그리고 갱신 전달 방법으로 전파(propagation : 새로운 값의 인스톨)와 무효화(invalidation : 캐쉬된 데이터 삭제)를 동적으로 사용한 알고리즘이다. 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 야기시켜 작업의 낭비를 가져오고, 교착상태로 인한 기아상태(Starvation) 발생의 문제점이 있다.

• 2V-CBL : CBL방법의 한 변형으로 두 개의 버전을 사용한 회피 기반의 2V-CBL 알고리즘이다. 이 방법에서는 트랜잭션의 잠금을 세분화하여 동시성을 높임으로써 읽기 전용 트랜잭션의 경우, AACC 방법보다 월등한 성능을 보임을 증명하였다.

Ⅲ. 제안하는 알고리즘 : ACC-UC 알고리즘

3-1 개요

제안하는 알고리즘(ACC-UC : Avoidance-based Cache Consistency algorithm without Unnecessary Callback)은 AACC처럼 클라이언트가 서버에 데이터 기록 의도를 알리고 서버의 응답을 기다리지 않고 바로 수행을 시작하는 비동기적인 방법을 사용한다. 비동기적인 방법은 동기적 방법에 비하여 서버의 응답을 기다리지 않아도 되므로 그만큼 응답 시간이 짧아진다. 데이터에 대한 갱신 잠금은 트랜잭션이 종료되면 일단 판독 잠금으로 하향 조정되어 불필요한 충돌을 발생시키지 않도록 유지된다. 서버는 페이지 단위의 잠금을 관리하며, 클라이언트에서는 페이지 단위와 객체 단위의 잠금을 관리한다. 설정된 잠금과 요청된 잠금 사이에서 충돌이 발생하는 경우, 요청된 잠금은 대기 상태로 변경되며 서버는 교착상태를 탐지하기 위한 처리를 수행

한다. 이 논문에서는 별도의 교착 상태 해결 알고리즘을 제안하지는 않는다. 페이지에 대한 잠금은 판독 잠금과 기록 잠금 두 가지만 유지한다. 잠금의 유형을 다양하게 유지하면 트랜잭션의 접근 의도를 정확히 파악할 수 있으나, 결국엔 잠금 상승(lock escalation)을 위한 부가적인 메시지가 발생하게 된다. 서버는 여러 클라이언트에서 캐쉬하고 있는 데이터의 분포에 관련된 정보를 유지하고 있다가 클라이언트에게 제공한다. 클라이언트에서 수행되는 트랜잭션은 특별한 가정없이 판독 연산과 기록 연산으로 구성되는 일반 트랜잭션으로 간주한다.

3-2 캐쉬 일관성 유지 방법

클라이언트가 캐쉬에 없는 데이터를 얻기 위해 서버에 요구를 보내는 경우는 다음과 같다.

• **다른 클라이언트에 의해 사용되지 않는 데이터를 요구하는 경우** : 요구한 데이터에 대한 잠금이 판독이든 기록이든 성공적으로 허가될 수 있다. 클라이언트는 서버에 해당 잠금의 유형과 함께 데이터를 요구하고, 서버는 요구된 잠금을 바로 수여하고, 잠금 테이블에 데이터를 요구한 트랜잭션 식별자 엔트리를 삽입한다.

• **판독 잠금이 설정되어 있는 데이터를 요구하는 경우 :** 요구하는 잠금이 판독 잠금인 경우, 서버는 클라이언트에게 해당 데이터에 대한 판독 잠금을 설정하여 반환한다. 요구하는 잠금이 기록 잠금인 경우 서버는 동일 데이터를 캐쉬하고 있는 원격 클라이언트들에게 callback 메시지를 보내고, 응답이 오는 동안 기록 연산을 수행한다. 부정적인 응답의 경우에는 원격 클라이언트의 트랜잭션이 종료될 때까지 대기 큐에서 대기하며, 긍정적인 응답의 경우에는 잠금 상승을 수행한다. 원격 클라이언트에서 판독 연산이 실행중이어서 충돌이 발생하면 서버는 교착상태 탐지 알고리즘을 수행한다.

• **기록 잠금이 설정되어 있는 데이터를 요구하는 경우** : 제안하는 알고리즘에서 클라이언트가 요구한 데이터에 기록 잠금이 이미 설정되어 있는 경우는 원격 클라이언트에 의해 갱신이 진행되고 있음을 의미하므로, 데이터를 요구한 클라이언트는 설정된 기록 잠금을 가진 트랜잭션이 완료할 때까지 대기 큐에서 대기한다.

3-3 트랜잭션 종료

트랜잭션의 수행이 성공적으로 끝나 완료하는 경우, 클라이언트는 로그를 서버에 보낸다. 그리고 보유한 기록 잠금은 판독 잠금으로 하향 조정한다. 서버는 로그 레코드를 참조하여 완료된 트랜잭션이 갱신한 데이터에 대한 잠금을 기록 잠금으로 변경한다. 완료된 트랜잭션에 의해 갱신된 값은 서버를 경유하여 다른 서버에 전달되고, 각 서버는 클라이언트들에게 무효화 메시지를 보내 캐쉬 내에서 유효하지 않은 데이터가 삭제되도록 한다. 클라이언트 트랜잭션이 완료되면 서버는 대기 큐를 검색하여 완료된 트랜잭션과의 충돌로 대기 상태에 있는 트랜잭션을 찾아 활성화시킨다.

교착 상태를 해결하기 위한 victim으로 선정되는 등 여러 원인에 의해 트랜잭션이 철회되는 경우, 잠금 테이블과 대기 큐, WFG에 있는 트랜잭션 엔트리는 모두 삭제된다.

3-4 서버 자료 구조

제안하는 알고리즘을 위해 서버에 다음과 같은 자료 구조를 유지하며, 그림 2와 같다.

• 대기 큐 : 각 데이터에 대하여, 원격 클라이언트와의 충돌로 허가되지 못한 트랜잭션 식별자를 링크드 리스트로 유지한다.

• 잠금 테이블 : 각 데이터에 대하여, 판독이나 기록 잠금을 소유한 트랜잭션 식별자를 유지한다.

• WFG(Wait-For Graph) : 충돌이 발생했을 때 선후 관계를 표현한 그래프로 노드 Tj에서 Ti로의 화살표는 Tj가 Ti의 완료를 기다림을 의미한다. WFG는 각 데이터에 대하여 유지되며, 교착상태가 발생되면 전체적인 WFG에 순환이 발생한다.

• Cached_List : 데이터가 캐쉬된 클라이언트를 추적하기 위한 리스트. 각 데이터에 대하여, 데이터를 캐쉬하고 있는 클라이언트의 식별자를 유지한다.

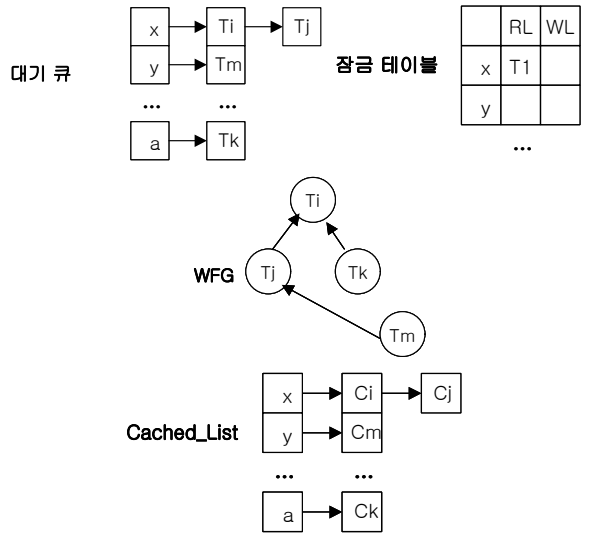


그림 2. 대기 큐, 잠금 테이블, WFG, Cached_List의 예
Fig. 2. Example of Wait Queue, Locking Table, WFG, and Cached_List.

3-5 알고리즘

이 절에서는 클라이언트와 서버에서 수행되는 알고리즘을 의사 코드 형태로 기술한다.

■ 클라이언트 알고리즘

1. 수행하려는 연산인 "판독"이면,
   1.1 페이지가 캐쉬에 존재하면, 연산 수행
   1.2 페이지가 캐쉬에 존재하지 않으면, 서버에 요구, 응답 대기
2. 수행하려는 연산이 "기록"이면,
   2.1 캐쉬 존재 여부에 상관없이 서버에 기록 잠금 허가를 위한 메시지 전송
   2.2 응답 대기
3. 완료 절차
   3.1 로그를 서버에 전송
   3.2 소유한 잠금을 모두 판독 잠금으로 하향조정
   3.3 갱신 데이터를 서버에 전송
4. 철회 절차
   4.1 접근한 모든 데이터에 대하여, Cached_List로부터 트랜잭션 엔트리 삭제
   4.2 WFG 그래프에서 트랜잭션 엔트리 삭제
   4.3 대기 큐에서 트랜잭션 엔트리 삭제
5. 서버로부터 페이지에 대한 callback 메시지를 받으면,
   5.1 수행중인 트랜잭션에 의해 접근되지 않으면 페이지를 삭제하고 서버에 'y' 응답
   5.2 수행중인 트랜잭션에 의해 접근중이면 서버에 'n' 응답
6. 갱신된 페이지에 대한 무효화 메시지를 받으면,
   6.1 캐쉬에 보유한 데이터가 무효화 메시지에 포함되어 있으면 캐쉬에서 삭제

■ 서버 알고리즘

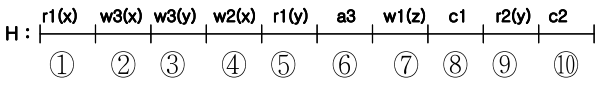
1. 클라이언트로부터의 페이지 요구에 대하여,
   1.1 아무 잠금이 설정되어 있지 않으면, 즉시 수여하고, cached_list 변경, 잠금 테이블 변경
   1.2 요구한 잠금과 설정된 잠금이 충돌 관계이면,
      1.2.1 캐싱중인 클라이언트에게 callback 메시지 전송
      1.2.2 교착상태 탐지 알고리즘 수행
   1.3 callback 메시지의 응답이 모두 'y'이면 요구한 잠금 허가
   1.4 callback 메시지의 응답에 'n'이 포함되어 있으면 대기 큐에 삽입, 클라이언트에게 응답
2. 클라이언트 트랜잭션의 완료 통보에 대하여,
   2.1 잠금 테이블 갱신
   2.2 서버들에 갱신 전달
   2.3 각 서버들은 클라이언트들에게 무효화 메시지 전달
   2.4 대기 큐를 검색하여 활성 가능한 트랜잭션 선택 및 활성화

■ 예제 시나리오

시나리오 기술을 간단하게 하기 위해, 하나의 서버와 그에 연결된 세 개의 클라이언트 C1, C2, C3을 가정한다. 서버에는 세 개의 페이지 x, y, z가 유지되고 있으며, 각 클라이언트에서는 각 하나씩의 트랜잭션 T1, T2, T3이 수행되는데, 다음과 같이 구성된다.

T1 : r1(x) r1(y) w1(z)
T2 : w2(x) r2(y)
T3 : w3(x) w3(y) w3(z)

트랜잭션 수행을 시작할 때 각 클라이언트의 캐쉬에는 어떤 페이지도 캐쉬되어 있지 않다고 가정하고, 연산이 수행되는 히스토리 H는 다음과 같다.



① : cached_list에 추가, 잠금 테이블 갱신, 데이터와 잠

금 허용, 연산 수행
② : 서버에 잠금 요청, C1에 callback 전송, n 응답, 대기 큐에 삽입
③ : 서버에 잠금 요청, cached_list에 추가, 잠금 테이블 갱신, 데이터와 잠금 허용, 연산 수행
④ : 서버에 잠금 요청, C1에 callback 전송, n 응답, 대기 큐에 삽입
⑤ : 교착 상태 발생, 희생자 선정(T3), r1(y) 수행
⑥ : 교착상태 해결을 위해 T3 철회, cached_list에서 삭제, WFG 삭제, 대기 큐 삭제
⑦ : cached_list에 추가, 잠금 테이블 갱신, 데이터와 잠금 허용, 연산 수행
⑧ : T1의 완료, 잠금 테이블 갱신, WFG 갱신, 대기 큐에서 T2 활성화, w2(x) 수행
⑨ : cached_list에 추가, 잠금 테이블 갱신, 데이터와 잠금 허용, 연산 수행
⑩ : T2의 완료, 잠금 테이블 갱신, WFG 갱신, 대기 큐에서 T2 활성화, w2(x) 수행
⑪ : T3의 재 제출

시나리오의 세 개의 트랜잭션에 대하여, 하나는 교착상태 발생으로 인한 희생자로 선정되어 철회되고, 나머지 두 개의 트랜잭션은 성공적으로 완료된다.

Ⅳ. 정확성 증명 및 성능 분석

4-1 정확성 증명

이 절에서는 제안하는 알고리즘이 중복 환경에서 만족해야 할 정확성 기준인 1-사본 직렬성(1-copy Serializability)을 만족함을 보인다.

(정리 1) 제안하는 ACC-UC 알고리즘은 1-사본 직렬성을 만족한다.

(증명) 1-사본 직렬성은 중복 환경에서 제안된 동시성 제어 방법의 정확성을 증명하기 위해 가장 일반적으로 사용되는 기준이며, 다중 버전 직렬화 그래프(MVSG : Multi-Version Serialization Graph)에 순환이 발생하지 않으면 보장된다[9]. 즉, 트랜잭션이 접근한 페이지의 충돌 순서에 순환이 발생하지 않으면 유지되므로, 다음의 충돌 연산을 중심으로 기술한다.

ⅰ) r-w 충돌 : 제안하는 방법에서 캐쉬된 저장된 데이터는 항상 판독이 가능하다. 그래서 판독 연산을 수행할 때는 서버와 접촉하지 않는다. 판독은 복수개의 클라이언트에서 동시에 수행되어도 문제를 발생시키지 않는다. 하지만 기록 연산은 캐쉬에 데이터를 보유하고 있더라도 서버에 잠금을 요청해야 한다. 이 때 서버에는 Cached_List가 유지되어 동일 페이지를 캐쉬하여 판독의 가능성이 있는 클라이언트들을 알 수 있다. 제안하는 알고리즘에서는 기록 연산 전에 callback 메시지를 보내어 무효화 여부를 판단한 후 기록 잠금을 수여하므로 r-w 충돌은 항상 감지되며 충돌 관계는 서버에 유지되고, 충돌이 발생되면 선행하는 트랜잭션이 완료하기 전까지는 충돌 관계의 트랜잭션은 대기 큐에서 기다리게 된다.

ⅱ) w-r 충돌 : 클라이언트에서 요청한 기록 잠금이 수여된 후, 클라이언트의 캐쉬에 보유한 데이터에 대한 판독이 자유롭게 이루어진다면 충돌이 감지되지 않아 직렬화 그래프에 순환이 발생할 수 있다. 하지만, 클라이언트가 기록 잠금을 요청하면 이미 동일한 페이지를 보유하고 있는 클라이언트에게 callback 메시지가 전달되는데, 원격 클라이언트에서 페이지를 사용하고 있지 않다면 무효화(삭제)시키므로, 기록 잠금이 수여되어 수행중이라면 임의의 원격 클라이언트에서는 동일 페이지를 캐쉬하고 있지 않다는 것을 보장할 수 있다. 만약 하나의 원격 클라이언트에서 callback 메시지에 긍정적인 응답을 보내지 않았다면 기록 잠금은 수여되지 못한다.

ⅲ) w-w 충돌 : 위의 두 경우와 마찬가지로, 기록 잠금은 반드시 서버에 의해 검증된 후 수여되므로 하나의 트랜잭션에 의해 이미 기록 잠금이 설정되어 수행중이라면 다른 트랜잭션은 충돌 잠금이 해제될 때까지는 대기 상태에 들어가므로 이러한 유형의 충돌은 해결된다.

위의 세 가지 경우에서 보는 바와 같이 어떤 경우에도 직렬화 그래프에는 순환이 발생하지 않도록 스케줄이 가능하므로, 제안하는 방법은 1-사본 직렬성을 만족한다고 증명할 수 있다.

4-2 성능 분석

이 절에서는 현재까지 제안된 비동기적 알고리즘 중 가장 최신의 방법이며 진보된 방법으로 알려진 AACC 방법과 ACC-UC 알고리즘을 메시지 교환 측면에서 정성적 방법으로 비교하고자 한다.

1) 잠금 상승으로 인한 메시지 교환

AACC에서는 캐쉬에 없는 데이터를 요구할 때, 서버는 일단 전용 판독 모드나 공유 판독 모드를 수여하고 갱신 연산인 경우에는 다시 잠금 상승을 수행한다. 따라서 잠금 상승을 위해서는 서버와 클라이언트 사이에 추가적인 메시지 교환이 필요하다. 제안하는 ACC-UC 알고리즘에서는 필요한 잠금 요청을 데이터 요구와 함께 하나의 절차로 해결하므로 잠금 상승으로 인한 추가적인 메시지 교환이 없다. 이러한 상황은 갱신 연산이 많아질수록 비례하여 증가한다.

2) 판독 모드 변경으로 인한 메시지 교환

AACC 방법에서는 캐쉬에 없는 데이터를 캐쉬할 때, 전용 판독 모드나 공유 판독 모드로 캐쉬한다. 캐쉬하려는 데이터가 임의의 원격 클라이언트에서 전혀 캐쉬되지 않았다면 전용 판독 모드로, 다른 클라이언트에서 이미 전용 판독 모드로 캐쉬되었다면 공유 판독 모드로 캐쉬된다. 어떤 클라이언트가 공유 판독 모드로 데이터를 캐쉬하게 되면 전용 판독 모드로 캐쉬하고 있던 클라이언트에서도 잠금 모드의 변경이 일어나야 한다. 제안하는 ACC-UC 알고리즘에서는 판독 잠금 모드가 하나로 유지되므로 이런 판독 모드 변경으로 인한 메시지 교환은 전혀 발생하지 않는다.

클라이언트의 캐쉬는 데이터의 접근성과 데이터베이스 시스템의 확장성을 향상시키는데 중요한 기술이다. 하지만 서버와 클라이언트에 유지되는 데이터의 일관성이 보장되지 않는다면 캐쉬를 접근하는 클라이언트 어플리케이션은 정확성을 보장받을 수 없다. 이를 위해 캐쉬 일관성 알고리즘이 필요하다.

V. 결 론

지금까지 제안된 캐쉬 일관성 알고리즘은 크게 탐지 기반과 회피 기반으로 분류된다. 탐지 기반 방법은 일단 트랜잭션을 수행한 후 비일관된 데이터의 접근을 찾아내어 해결하는 방법으로 낙관적 방법으로 간주된다. 회피 기반 방법은 트랜잭션이 비일관된 데이터를 접근할 가능성을 미리 통제하는 것으로 비관적 방법으로 간주된다. 탐지 기반의 방법은 알고리즘이 간단한데 비하여 트랜잭션의 철회가 수행 후에 이루어지므로 자원 및 시간의 낭비가 문제가 된다. 회피 기반은 탐지 기반에 비하여 알고리즘의 복잡도가 크고, 서버보다는 클라이언트 측의 부담이 크지만 일반적으로 많은 알고리즘이 제안되고 연구가 활발히 이루어지고 있다.

탐지 기반의 알고리즘은 기록 의도를 선언하는 시기, 기록 잠금의 보유 시간, 원격 충돌 해결 방법, 원격 갱신의 전달 방법 등에 따라 다양한 알고리즘을 구성할 수 있다. 이 논문에서는 회피 기반의 비동기적 캐쉬 일관성 유지 방법으로, 기존의 AACC 방법에서 발생하는 불필요한 callback을 제거하는 알고리즘을 제안하였다. 클라이언트-서버 환경에서 불필요한 메시지 교환은 네트워크에 부담을 주어 성능을 저하시키는 원인이 된다.

제안하는 ACC-UC 알고리즘에서는 캐쉬에 없는 데이터를 서버에 요구할 때 잠금 모드를 함께 전달하여 추후에 잠금 상승으로 인한 메시지를 제거하고, 잠금의 형태를 단순화하여 잠금 관리를 위한 오버헤드를 감소시켰다. 또한, 트랜잭션 종료 후 기록 잠금을 판독 잠금으로 하향조정하여, 원격 클라이언트에서 거짓 충돌로 간주되어 callback 메시지를 보내는 과정이 발생되지 않도록 하였다. 제안하는 알고리즘에서는 트랜잭션의 유형을 특별히 제한하지 않고 기록 연산과 판독 연산을 갖는 일반 트랜잭션으로 가정하였는데, 블필요한 callback은 갱신이 많아질수록 비례하여 발생하므로 갱신이 활발한 응용에서 효율적으로 적용가능할 것으로 예상된다.

이 연구에 추가하여, 시뮬레이션을 통한 성능 평가와 회피 기반의 캐쉬 일관성을 사용하는 환경에서 발생가능한 여러 고장의 유형으로부터 시스템을 견고하게 유지하기 위한 회복 기법에 대하여 연구할 예정이다.

참 고 문 헌

[1] M. J. Franklin, M. Carey, and M. Livny, "Transactional client-server cache consistency : alternatives and performance," *ACM Transactions on Database Systems,* vol. 22, no. 3, pp 315-363, 1997.

[2] M. Franklin, and Carey, M. "Client-server caching revisited," *In Proc. International Workshop on Distributed Object Management,* Morgan Kautmann, pp. 57-78, 1994.

[3] Carey, M., A. Adya, B. Liskiv, and M. Zaharioudakis. "Fine grained sharing in a page server OODBMS," *In Prod. ACM SIGMOD International Conference on Management of Data,* pp. 359-370, 1994.

[4] 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.

[5] Carey, M., M. Franklin, and M. Zaharioudakis, and E. Shekita, "Data caching tradeoff in client-server DBMS architectures," *In Proc. ACM SIGMOD International Conference on Management of Data,* pp. 357-366, 1991.

[6] Wang, Y. and L. Rowe, "Cache consistency and concurrency control in a client/server DBMS architecture," *In Proc. ACM SIGMOD International Conference on Management of Data,* pp. 367-376, 1991.

[7] 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.

[8] 강흠근, 민준기, 전석주, 정진완, "클라이언트-서버 DBMS 환경에서 콜백 잠금 기반 다중 버전의 활용," *정보과학회 논문지,* vol. 31, no. 5, pp. 457-467, Oct. 2004.

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

김 치 연 (金致連)

1992년 2월 : 전남대학교 전산통계학과(이학사)
1994년 2월 : 전남대학교 전산통계학과(이학석사)
1999년 8월 : 전남대학교 전산통계학과(이학박사)
2002년 3월~현재 : 목포해양대학교 해양전자통신공학부 교수
관심분야 : 이동 컴퓨팅, 전자상거래, 트랜잭션 관리, 유비쿼터스 컴퓨팅