韓國 컴퓨터情報學會 論文誌
第9卷 第2號, 2004. 6.

2004-9-2-1-5

비동기적 캐쉬 일관성 유지 기법

이 찬 섭*

Asynchronous Cache Consistency Technique

Lee Chan Seob*

요 약

컴퓨터 성능과 정보통신 기술의 발달로 클라이언트/서버 환경이 보편화됨에 따라 서버는 제한된 대역폭의 절감과 빠른 응답시간, 그리고 확장성을 위해 클라이언트는 지역 캐쉬를 이용한다. 이때 서버와 클라이언트 간에는 캐쉬된 데이타의 일관성 유지가 필요하며 이에 따라 많은 기법이 제안되고 있다. 이 논문은 기존의 갱신 빈도 캐쉬 일관성 기법을 개선한 것이다. 기존의 일관성 기법은 선언을 동기적으로 하여 응답시간이 늦거나 정확성 검사를 지연시켜 철회 단계가 증가하는 단점이 있다. 본 논문에서 제안된 기법은 이러한 문제점을 해결하기 위해 페이지 요청 또는 완료시 갱신 연산이 일어난 객체에 대해 갱신 시간을 참조하여 수행하도록 하였다. 따라서, 갱신 연산이 이루어지는 경우 비동기적으로 선택 모드에 따라 갱신의도 선언 또는 갱신을 선택적으로 수행할 수 있어 응답이 빠르고 철회 단계가 감소하며 더 명확한 선택이 가능하다는 장점을 갖는다.

Abstract

According as client/server is generalized by development of computer performance and information communication technology, Servers uses local cache for extensibility and early response time, and reduction of limited bandwidth. Consistency of cached data need between server and client this time and much technique are proposed according to this. This paper improved update frequency cache consistency in old. Existent consistency techniques is disadvantage that response time is late because synchronous declaration or abort step increases because delaying write intention declaration. Techniques that is proposed in this paper did to perform referring update time about object that page request or when complete update operation happens to solve these problem. Therefore, have advantage that response is fast because could run write intention declaration or update by sel_mode electively asynchronously when update operation consists and abort step decreases and clearer selection.

▸ Keyword : Cache consistency, Database, Client/Server

* 혜천대학 전임강사

Ⅰ. 서 론

컴퓨터 성능 향상과 정보통신 기술의 발달로 클라이언트/서버 환경이 보편화 되면서 사용자는 데이타의 크기를 고려하지 않고 유무선 통신망을 통해 언제 어디서나 쉽게 정보를 주고받을 수 있게 되었다.

클라이언트/서버 환경은 하나의 서버와 다수의 클라이언트 또는 다수의 서버와 다수의 클라이언트로 구성될 수 있으며 유.무선 네트워크를 통해 통신하는 이동 컴퓨팅 환경과 웹 환경 등에 적용된다. 이러한 환경에서 데이타가 자주 이동되면 고정되고 협소한 대역폭으로 인해 통신 채널을 점유하게 되고 그로인해 네트워크 지연 및 부하를 가져오게 된다.

클라이언트/서버 환경에서 데이타를 이용하는 방법은 질의(query) 전송과 데이타(data) 전송으로 나뉜다. 전자의 방법은 서버 측에서 모든 연산이 이루어진 후에 결과만 클라이언트에 전송되므로 클라이언트 측의 연산 능력과 데이타 저장이 불필요하다는 장점이 있는 반면, 잦은 질의로 인해 통신 대역폭의 점유 경쟁이 발생하고, 서버에서 질의를 처리함으로써 서버의 부하(overhead)와 질의에 대한 응답 시간이 오래 걸리는 등의 단점을 가진다. 후자의 방법은 제한된 통신 대역폭을 절감할 수 있고, 데이타가 지역 캐쉬에 저장되어 있어 자체 연산능력으로 인한 서버의 부하 감소, 독립성과 확장성 증가 그리고 빠른 응답이 가능하다는 장점을 가지는 반면, 클라이언트 측에 데이타를 전송받아 처리해야 하므로 클라이언트와 서버간에 서로 일관성을 유지해야 하는 부담을 갖고 있다. 클라이언트의 능력이 점점 커지면서 후자의 방법을 주로 사용하고 있으며[1] 그에 따른 많은 일관성 유지 기법들이 많이 발표되었다.

클라이언트/서버 환경에서 일관성을 유지하기 위한 기법은 아래 (그림 1, 2)와 같이 크게 탐지-기반 기법과 회피-기반 기법으로 구분된다.

탐지-기반 기법들은 데이타의 정확성 여부를 확인하기 위해 많은 메시지를 전송하여 일관성 유지 하므로 많은 대역폭을 차지하게 된다.

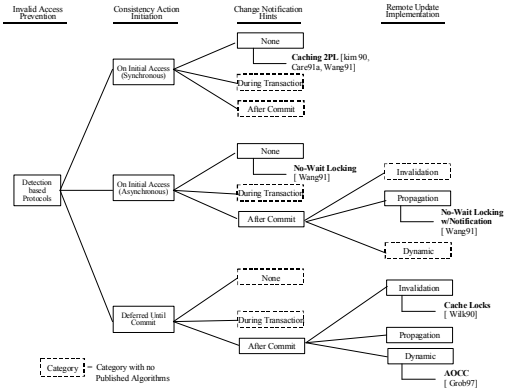

그림 1. 탐지-기반 기법
Fig 1. Detection-based method

회피-기반 기법들은 정확성 검사를 나중으로 지연시킨 후 일관성 검사를 함으로써 메시지 전송 횟수 감소와 신속한 응답은 가능하지만 충돌이 늦게 발견되어 철회단계가 증가하는 등 각각의 장단점들을 가지고 있다[2][3].

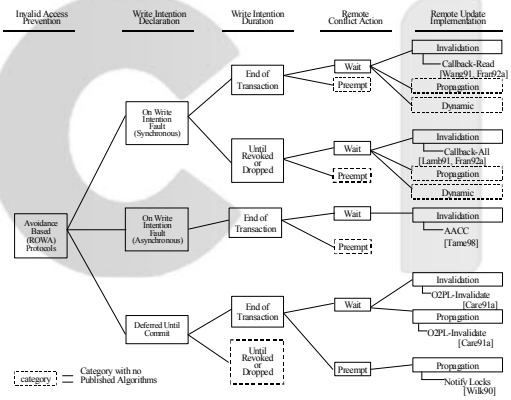

그림 2. 회피-기반 기법
Fig 2. Avoidance-based method

이 논문은 기존 갱신 빈도 캐쉬 일관성 유지 기법에 데이타가 갱신되는 시간 간격을 고려함으로써 기존의 기법보다 더 효율적으로 캐쉬 일관성을 유지할 수 있도록 한 기법이다. 갱신 빈도 캐쉬 일관성 기법은 갱신 횟수를 이용하여 비동기적으로 갱신-의도 선언을 함으로써, 자주 갱신되지 않는 데이타에 대해서는 갱신부터 수행하므로 메시지 전송 횟수를 감소시키고 자주 갱신되는 데이타는 먼저 갱신-의도부터 선언하므로 철회단계 감소와 다른 클라이언트들로부터 CB(callback) 응답 전에 갱신-의도를 선언한 클라이언트에

응답함으로써 신속한 응답이 가능하도록 한 기법이다[22][23]. 하지만 갱신이 자주 발생하더라도 시스템에서 정해진 갱신 횟수에 도달하지 못하면 자주 갱신되지 않는 데이타로 간주하며 갱신 횟수에 도달한 후에는 갱신이 자주 발생하지 않더라도 자주 갱신되는 데이타로 간주하는 문제를 가진다.

II. 관련연구

1. 트랜잭션 처리 방법

클라이언트/서버는 일반적으로 두 개의 컴퓨터(또는 프로그램) 사이에서 정보의 요청자와 제공자의 역할 관계를 나타낸다. 클라이언트는 다른 컴퓨터에게 서비스를 요청하는 측을 말하며, 서버는 요청한 측에 대해 응답을 해주는 컴퓨터이다. 이러한 클라이언트/서버에서는 클라이언트가 서버에 필요한 정보를 요청 시 서버가 질의의 결과를 제공할 것인지 아니면 데이타를 제공할 것이지에 따라 질의 전송(Query Shipping) 방법과 데이타 전송(Data Shipping)방법으로 나뉜다[3]. 초기 시스템들은 질의 전송 방법을 사용했으나 최근 클라이언트의 성능이 증가하면서 데이타 전송 방법을 주로 사용한다.

데이타 전송 방법의 장점은 트랜잭션 처리에 필요한 여러 처리 기능들을 클라이언트가 갖게 함으로써, 서버의 부하를 줄일 수 있으며 그로인해 클라이언트의 확장성이 증가되고 데이타가 응용프로그램 가까이 있기 때문에 응답시간이 빠르다. 단점으로는 서버에서 데이타를 전송 받아 트랜잭션을 처리하므로 데이타 전송 시 네트워크 대역폭을 많이 사용하며, 클라이언트와 서버에 캐쉬된 데이타의 일관성, 즉 캐쉬 일관성 유지 기법이 필요하고 구현이 복잡하다.

클라이언트에서 트랜잭션을 처리하는 방법은 클라이언트들이 트랜잭션 처리에 필요한 데이타를 서버로부터 전송 받아 클라이언트의 캐쉬에 저장한 후 트랜잭션 처리를 수행하는 방법으로 클라이언트는 질의 처리를 위한 기능을 가지고 있다. 따라서 클라이언트는 응용 프로그램, 인터페이스 계층, 질의 처리, 객체 관리기, 트랜잭션 관리기, 버퍼 관리기 등으로 구성될 수 있으며 서버는 전체적인 트랜잭션 처리를 조정하는 트랜잭션 관리기, 각 클라이언트에 캐쉬된 데이타 사본 정보를 관리하는 버퍼 관리기, 그리고 데이타의 입출력을 담당하는 입/출력 계층으로 구성될 수 있다.

2. 캐쉬 일관성 유지 기법

캐쉬 일관성 유지 기법은 데이타의 정확성 검사를 하는 방법에 따라 탐지-기반(Detection-based) 기법과 회피기반(Avoidance-based) 기법으로 나뉜다. 탐지-기반 기법은 C2PL(Cache 2 Phase Locking), NWL(No-Wait Locking), AOCC(Adaptive Optimistic Concurrency Control) 등이 있으며 회피-기반 기법은 CB(CallBack), ACBL(Adaptive CallBack Locking), O2PL(Optimistic 2PL), AACC(Asynchronous Avoidance Cache Consistency) 등이 연구되었다[3][4][5][6][7][8][9][10][11].

탐지 기반 기법은 클라이언트의 캐쉬에 부정확한 데이타(stale data)가 있는 것을 허용하므로 트랜잭션이 종료하기 전에 참조한 데이타의 정확성을 마지막에 검사하는 방법이다. 따라서 서버는 클라이언트가 데이타의 정확성을 검사하는데 필요한 정보를 유지해야 하며 일관성 유지는 오직 한 클라이언트와 서버를 포함하기 때문에 클라이언트의 캐쉬 관리 방법은 간단하지만 서버의 의존도가 높고, 서버와의 통신 증가로 오버헤드 발생 가능성이 있다[2][3].

회피 기반 기법은 ROWA(read one/write all)에 따라 사본 관리를 하며 트랜잭션들이 그들 자신의 지역에 있는 정확하지 않은 데이타를 참조할 기회를 주지 않음으로써 일관성을 유지하는 기법이다. 이 기법은 서버에 대한 의존도가 감소하고 데이타 충돌의 탐지 비용이 높은 환경에서 성능이 좋다는 장점이 있다[2][3].

캐쉬 일관성 유지 기법 중 가장 최근의 기법은 AOCC와 AACC기법이다. AOCC기법은 클라이언트에 캐쉬된 객체가 정확하고, 현재 다른 트랜잭션에 의해 이용되지 않고 있다는 가정 하에 종료 시점까지 서버와의 통신을 하지 않음으로써 메시지 전송 비용을 줄이기는 했으나 철회 시 단계가 증가한다[7,10]. AACC기법은 페이지가 한 클라이언트에 캐쉬됨을 의미하는 개인-판독 록(private-read lock)과 페이지가 여러 클라이언트에 캐쉬됨을 의미하는 공유-판독 록(shared-read lock)으로 구분하여 효율성을 높이기는 했으나 개인-판독 록에서만 큰 성능 향상을 보인다[5].

3. 갱신 빈도 캐쉬 일관성 기법

갱신 빈도 캐쉬 일관성 기법은 갱신이 자주 일어나지 않는 객체는 갱신-의도를 선언하지 않고 갱신 연산을 먼저 수행한 후 Commit 단계에서 정확성 검사를 수행하게 함으로

써 트랜잭션의 신속한 처리 및 메시지 전송 횟수를 감소시키고 갱신이 빈번히 일어나는 객체는 먼저 갱신-의도를 비동기적으로 선언한 후 갱신을 수행하게 함으로써 충돌을 조기에 발견할 수 있어 철회 단계를 줄인 기법으로 부분적으로 AOCC기법과 AACC기법에 비해 성능의 우수성을 보였다[21,22]. 하지만 이 기법은 갱신 횟수만 가지고 갱신 또는 갱신-의도 선언 여부를 결정하므로 빈번하게 갱신이 일어나더라도 갱신 횟수에 도달하기 전까지는 항상 갱신부터 수행하므로 철회단계를 증가시킴으로써 오히려 효율성이 떨어질 수 있는 소지가 있다.

III. 일관성 유지 기법의 제안

1. 갱신 시간을 고려한 일관성 유지 기법

제안한 기법은 갱신 빈도 일관성 유지 기법에 갱신 시간을 적용하여 기존 기법을 보완시킨 비동기적 일관성 유지 기법이다. 갱신되는 데이타에 대해 서버에서 갱신 시간을 유지하고 있다가 클라이언트의 데이타 요청 시 또는 완료 시 데이타와 선택 모드를 함께 보냄으로써 추후 클라이언트에서 갱신을 하고자 할 때 이 갱신 시간에 따른 선택 모드 정보를 이용하여 비동기적으로 일관성을 유지할 수 있도록 하였다.

갱신-의도를 먼저 선언하는 경우는 메시지 수가 증가하는 반면 충돌을 조기에 발견할 수 있어 철회 단계를 줄일 수 있으며, 갱신을 먼저 수행하는 경우에는 메시지 수가 감소하는 반면에 충돌을 늦게 발견하는 단점을 가지고 있다. 제안한 기법은 갱신 시간 정보에 따라 자주 갱신이 일어난 데이타는 연산 수행 도중 다른 클라이언트에서 갱신될 가능성이 크므로 갱신-의도부터 선언하도록 함으로써 메시지 수는 증가하지만 철회단계를 감소할 수 있도록 하였으며 자주 갱신이 일어나지 않는 데이타는 연산 수행 도중에 갱신될 가능성이 적으므로 갱신을 먼저 수행하도록 하여 메시지 수를 줄일 수 있다. 따라서 이 선택적 갱신-의도 선언 및 갱신의 수행은 갱신 시간에 따라 수행될 수 있으므로 서버에서 갱신 시간에 따라 클라이언트에게 정보를 제공해야만 한다.

2. 클라이언트/서버의 구조

본 논문의 연구 환경은 하나의 서버와 다수의 클라이언트들이 네트워크를 통해 연결된 클라이언트/서버 데이타베이스 시스템 구조로 록 단위는 페이지와 서버로 한다[12][13][14][15][16][17]. 클라이언트는 트랜잭션을 생성하는 응용 프로그램, 서버와의 통신을 담당하는 통신 프로세스, 그리고 클라이언트 DBMS로 구성된다. 서버는 데이타베이스와 로그를 저장하는 디스크, 클라이언트와 통신을 담당하는 통신 프로세스, 그리고 서버 DBMS로 구성된다[18][19][20].

캐쉬 일관성을 위해 클라이언트와 서버에서 유지하고 있는 정보는 (그림 4)와 같다.

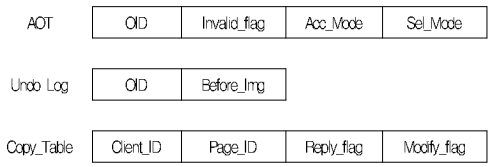

그림 3. 클라이언트와 서버의 테이블 정보
Fig 3. Table Info of Client and Server

클라이언트에서는 트랜잭션이 이용한 객체에 대한 정보를 기록하는 AOT와 Undo log를 유지하고, 서버는 클라이언트들이 복사해간 페이지 정보를 기록하는 Copy_Table을 유지한다.

AOT는 트랜잭션이 이용할 객체 식별자(OID), 객체에 적용한 연산의 종류를 나타내는 접근 모드(Acc_Mode : 판독 연산[Acc_Mode = 0], 갱신 연산[Acc_Mode = 1])와 트랜잭션 종료 시 삭제될 객체를 나타내는 Invalid_flag(초기값 "0"), 갱신 시간 값에 따라 갱신-의도 선언과 갱신을 선택할 수 있는 선택 모드(Sel_Mode : 갱신 [Sel_Mode = 0], 갱신-의도 선언[Sel_Mode = 1])로 구성된다.

Invalid_flag는 갱신통보를 받은 객체가 현재 판독 모드로 이용중이면 Invalid_flag를 "1"로 설정하고, 트랜잭션 종료 시 Invalid_flag가 "1"인 객체를 삭제(invalid) 표시를 함으로써 캐쉬 데이타의 일관성을 유지한다. Undo log는 OID와 갱신 이전의 상태를 나타내는 Before_Img로 구성되며 트랜잭션이 철회될 때 재시작을 위해 이용된다.

클라이언트에 캐쉬된 페이지 정보를 포함하는 Copy_Table은 서버에서 유지하는 자료 구조이다. Copy_Table은 페이지를 캐쉬한 클라이언트를 나타내는 클라이언트 식별자(Client_ID), 페이지 식별자(page_ID), 갱신통보를 보낸 클라이언트로부터의 응답 메시지 회신 여부를 나타내는

Reply_flag("0" : 회신을 받았거나 갱신통보 메시지를 보내지 않은 상태, "1" : 클라이언트에게 갱신통보를 보내고 아직 응답 메시지를 받지 않은 상태), 클라이언트로부터 갱신된 페이지가 서버의 디스크에 반영여부를 나타내는 Modify_flag("0" : 반영된 상태, "1" : 아직 반영되지 않은 상태)로 구성된다. 클라이언트의 데이타 요구가 있을 때 시스템의 지정 값에 따라 일정 시간 초과 이전에는 클라이언트의 Sel_Mode 값을 "0"으로 일정 횟수를 초과할 때에는 클라이언트의 Sel_Mode 값을 "1"로 설정시킨다.

3. 일관성 유지 기법의 처리 절차

클라이언트 DBMS는 자신의 버퍼에 캐쉬한 페이지와 객체의 정보를 유지하며 객체 단위의 록 관리를 수행한다. 트랜잭션은 클라이언트에서만 생성되고 어느 한순간에 하나의 트랜잭션만 활성화되며 갱신 연산 이전엔 반드시 판독 연산을 수행한다고 가정한다. 클라이언트와 서버간의 데이타 전송은 페이지 단위로 한다. 서버로부터 객체에 대한 갱신통보를 받았을 경우 객체를 포함한 페이지가 사용 중이 아니면 페이지를 삭제하고, 갱신통보를 받은 객체도 이용 중이지는 않지만 해당 페이지의 다른 객체가 이용중이거나 객체가 판독 모드로 이용 중이라면 삭제 표시를 한다. 만일 객체가 갱신 모드로 이용중이라면 트랜잭션을 철회하고 서버에 정보를 보낸다.

```
//클라이언트에서의 트랜잭션 수행
START(T1) { // 트랜잭션 시작
 while(트랜잭션 연산 끝이 아닌 동안) {
   switch( 연산 ) {
     case "read" :
       if (이용하려는 오브젝트가 클라이언트
           버퍼에 캐쉬 되지 않았다) then
          {서버에게 오브젝트 요구를 보냄;
           서버로부터 오브젝트가 포함된
           페이지를 받을 때까지 기다림;
          }
       ACT에 OID, Invalid_flag = 0,
       Acc_Mode = 0,
       Sel_Mode = 서버 지정 값 첨가;
       break;
     case "write":
       ACT에 OID, Invalid_flag  = 0,
       Acc_Mode = 1 변경;
       Undo log에 OID, Before_Img를 첨가;
       If (Sel_Mode = 0) then
         {오브젝트 갱신;}
       else
         {서버에게 갱신의도를 보냄;
              :
```

서버는 가장 최근의 원본 데이타를 갖고 클라이언트들의 요구를 처리한다. 클라이언트에게 페이지의 사본을 복사해 주며 이 사본 데이타를 캐쉬한 클라이언트의 정보를 Copy_Table에 유지한다. 전역적인 록 관리의 역할과 데이타가 갱신되었을 때 갱신된 페이지를 캐쉬한 클라이언트들에게 갱신을 통보하는 역할을 한다.

```
//서버에서의 처리
Client_MSG = 클라이언트로부터 받은 메시지;
If Sel_Mode = 0 then
   Client_MSG = "W_intention"
 switch (Client_MSG)  // 클라이언트로부터 갱신
 {                          의도 처리
 case "W_Intention" :
  if (갱신의도를 받은 OID에 갱신록) then
     클라이언트에게 트랜잭션 철회 응답을 함
  else {
     if (갱신의도를 받은 OID에 판독 록) then
        기록 록을 건 트랜잭션이 종료할 때까지
        기다림 ;
        OID에 갱신 록을 설정 ;
     while (Copy_Table의 끝이 아닌 동안) {
     if (OID가 포함된 페이지가 Copy_Table에
        있다) then {
        Copy_Table의 Reply_flag = 1로 설정,
        If (요청시간>서버 지정 시간) then
          클라이언트의 Sel_Mode = 1로 설정 ;
        else
          클라이언트의 Sel_Mode = 0로 설정 ;
          클라이언트에게 OID, Sel_Mode와
           함께 갱신통보를 보냄 ;
        }}}
              :
```

4. 시나리오에 따른 일관성 유지 기법

일관성 유지 기법의 수행 과정을 보여주기 위해 각 기법마다 동일하게 클라이언트1(C1)에서는 트랜잭션 T1이 수행되고 클라이언트(C2)에서는 트랜잭션 T2가 수행되며, T1과 T2의 연산은 다음과 같이 구성된다고 가정한다.

T1 : Read(Y) - Read(X) - Write(X) - commit
T2 : Read(X) - Read(Z) - commit

본 논문에서 제안한 캐쉬 일관성 기법에서 발생할 수 있는 트랜잭션 처리는 갱신 빈도 일관성 유지 기법과 동일하게 서버에서 주어진 갱신 시간 정보에 따라 갱신모드가 0인 경우와 1인 경우로 구분한다. 갱신모드 0과 1은 이전의 경우와 달리 수시로 바뀔 수 있다. (그림 5)의 시나리오는 갱신모드가 0인 경우에 나타날 수 있는 서버의 갱신통보에 대해 페이지가 삭제되는 경우, 미 사용 중인 객체가 삭제되는

경우/사용 중인 객체가 삭제되는 경우, 그리고 트랜잭션이 철회되는 경우이다.

시스템에 설정된 일정 갱신 빈도에 도달할 때까지 항상 갱신을 수행하는 것과 달리 갱신 시간 값을 이용하는 경우에는 초기에 빈번히 갱신될 때 갱신모드가 1로 변경되며 (그림 6)의 시나리오가 수행되므로 일정 시간대에 빈번히 갱신되는 경우에 효과적일 수 있다. 빈번하게 갱신되지 않는 경우에는 갱신모드가 0으로 변경되어 (그림 5)의 시나리오가 수행된다. 따라서 클라이언트는 갱신 연산이 필요할 때 갱신 연산부터 수행할지 또는 갱신-의도부터 선언할지를 판별하여 선택적으로 수행할 수 있어 기존 기법들보다 구분이 명확하다.

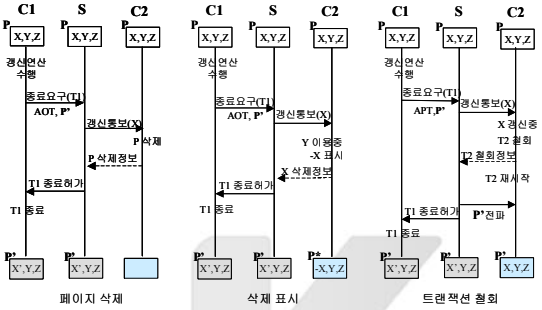

그림 3. 선택 모드 = 0
Fig 4. Sel_mode = 0

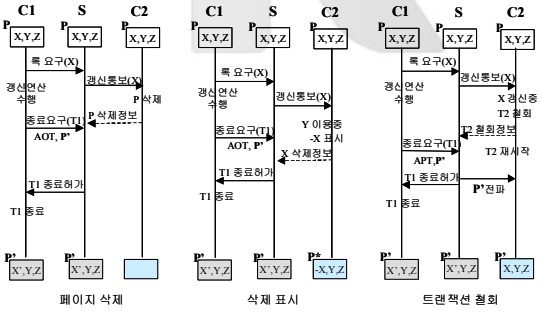

그림 4. 선택 모드 = 1
Fig 5. Sel_mode = 1

위 (그림 5, 6)은 선택 모드에 따라 일관성을 유지하는 방법을 나타내고 있다. 갱신 빈도 캐쉬 일관성 기법은 이전에 반복적으로 나타난 갱신연산이 있었더라도 갱신 빈도의 값에 도달하기까지 항상 동일하게 갱신 연산부터 수행하는 단점이 있으며 설정된 갱신 빈도에 도달한 경우에는 빈번하게 갱신이 일어나지 않더라도 갱신-의도부터 선언하는 단점 또한 발생한다. 따라서 갱신에 따른 시간 간격으로 갱신의 정도를 파악하고 모드를 변경함으로써 갱신 또는 갱신-의도를 수행하도록 문제점을 해결하였다.

Ⅳ. 비교 분석

실험은 Pentium IV 2.4 GHz, Windows2000 Advanced Server환경에서 AweSim 시뮬레이션 패키지 v3.0을 사용하였으며 데이터 판독, 기록, 갱신, 요청, 응답 등에 대해 각 기법에서 동일한 시간이 소요된다고 가정하고 응답 시간, 메시지 전송 횟수, 충돌 횟수를 평가하였다. 트랜잭션은 10000개가 발생하며 객체는 30개 대해 판독과 갱신이 일어나도록 하였다.

표 1. 기존 일관성 기법과 비교
Table 1. Comparison with Existing Consistency

구분		ACCC 기법	갱신빈도 기법	제안한 기법
평가항목	Client			
응답 시간	2	19.114	14.794	13.523
	4	21.070	16.061	15.206
	6	24.039	17.717	16.821
메시지 수	2	3.203	3.872	3.872
	4	3.558	4.149	4.149
	6	4.098	4.513	4.513
충돌 수	2	0.101	0.059	0.032
	4	0.278	0.197	0.145
	6	0.549	0.379	0.318

제안한 기법은 갱신 빈도 일관성 유지 기법에 발생된 시간을 고려한 기법으로써 <표 1>과 같이 기존 기법과 동일하게 메시지 수 같으나 시간을 고려함으로써 빠른 응답 시간과 적은 충돌이 일어나도록 한 기법이다. 기존의 기법은 갱신이 많은 객체에 대해 시간의 기준이 없으므로 일정 시간대에 갱신이 많은 경우에는 효율적일 수 없다. 즉, 일정 시간을 기준으로 초기에는 갱신이 자주 일어났으나 이후에는 갱신이 자주 일어나지 않는 경우와 일정한 간격으로 갱신이 일어난 경우에 모두 동일한 결과만을 얻도록 되어 있다. 초기에 자주 갱신이 발생한다 하더라도 시스템에서 주어진 빈

도 값 이전일 경우에는 갱신부터 선언하므로 철회율이 높으며 주어진 빈도 값을 초과한 이후부터는 갱신이 드물게 발생한 경우라도 항상 갱신-의도부터 선언하므로 응답이 느릴 수 있다. 따라서 제안한 갱신 시간을 고려한 비동기적 일관성 유지 기법은 초기에 자주 갱신이 발생하면 시스템에서 주어진 갱신 시간 값에 따라 갱신-의도부터 선언하여 철회율이 감소되며 갱신이 드물게 발생한 경우에는 갱신 시간 값에 따라 갱신부터 선언하여 빠른 응답이 가능하였기 때문에 기존의 기법보다 성능 향상을 보인다.

V. 결론

클라이언트/서버 환경에서 클라이언트는 성능향상을 위하여 자신의 버퍼에 서버 데이타의 사본을 캐쉬한다. 클라이언트/서버 환경에서 캐쉬 일관성 유지에 관한 기존의 연구들은 탐지기반의 기법과 회피기반의 기법으로 나뉘며 캐쉬 데이타의 일관성 유지에 드는 오버헤드가 존재함에도 불구하고 시스템 성능 향상에 우선을 둔다.

본 논문에서 제안한 갱신 시간을 이용한 비동기적 캐쉬 일관성 유지 기법은 클라이언트와 서버간의 통신비용을 줄이기 위하여 갱신 연산에 대해서 서버에 갱신-의도를 선언하는 방법을 선택했으며, 트랜잭션이 서버의 응답을 기다리는 시간을 줄이기 위하여 비동기적인 기법과 서버의 즉시 완료 허용을 하도록 하는 방법을 사용하였다. 또한 갱신 시간에 따라 비동기적으로 갱신 또는 갱신-의도를 수행할 수 있게 함으로써 갱신 빈도 일관성 유지 기법보다 선택이 명확하고 철회되는 트랜잭션을 줄이며 빠른 응답이 가능하도록 하였다.

추후 연구 과제로는 본 논문에서 제안한 비동기적 캐쉬 일관성 기법의 매개 변수를 더욱 구체화하여 더 세밀한 성능평가가 이루어져야 하며 제안한 기법을 이동 컴퓨팅 환경으로 확장하여 적용하는 등 다양한 확장이 필요하다.

참고문헌

[1] A. Delis and N. Roussopouls, "Modern Client-Server DBMS Architectures," Proc., ACM SIGMOD RECORD, pp.52-61, 1991.

[2] M. Franklin, M. Carey, M. Livny, "Transactional Client-Server Cache Consistency: Alternatives and Performance," Proc., ACM TODS, pp.315-363, 1997.

[3] M. Franklin, M. Carey, "Client Data Caching : A Foundation for High Performance Object Database System," Kluwer Academic Publishers, 1996

[4] M. Carey, M. Franklin, M. Livny, E. Shekita, "Data Caching Tradeoffs in Client-Server DBMS Architectures," Proc., ACM SIGMOD, pp.357-366, 1991.

[5] M. Tamer., and Kaladhar.,"An Asynchronous-Based Cache Consistency Algorithm for Client Caching DBMSs," Proc., VLDB, pp.440-451, 1998.

[6] M. Zahariousdakis, M. Carey, M. Franklin, "Adaptive, Fine-Grained Sharing in a Client-Server OODBMS: A Callback-Based Approach," to appeared ACM TODS.

[7] R. Gruber, "Optimism VS. Locking: A Study of Concurrency Control for Client-Server Object-Oriented Databases," PhD thesis, MIT, 1997.

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

[9] KIM, W., GARZA, J. F., BALLOU, N., AND WOELK, D, "Architecture of the ORION next-generation database system," IEEE Trans. pp. 109-124. 1990.

[10] ADYA, A., GRUBER, R., LISKOV, B., AND MAHESHWARI, U, "Efficient optimistic concurrency control using loosely synchronized clocks," Proc., ACM SIGMOD, 23-34. 1995.
[11] Michael J. Franklin , Michael J. Carey, "Client-Server Caching Revisited," IWDOM 57-78, 1992.
[12] M. Franklin, M. Carey, "Crash Recovery in Client-Server EXODUS," Proc., ACM SIGMOD, pp. 165-174, 1992.
[13] M. Franklin, M. Carey, M. Livny, "Local Disk Caching in Client-Server Database Systems," Proc., VLDB, pp.543-554, 1993.
[14] M. Franklin, M. Carey, M. Livny, "Global Memory Management in Client-Server DBMS Architectures," Proc., VLDB, pp.596-609, 1992.
[15] L. Amsaleg, M. Franklin, O. Gruber, "Efficient Incremental Garbage Collection for Client-Server Object Database Systems," Proc. of the 21th VLDB, pp. 42--53 1995.
[16] J. C.Lamb, G.Landis and D. Weinreb, "The objectstore database system," ACM, 34(10), 1991.
[17] Alex Delis, Nick Roussopoulos, "Management of Updates in the Enhanced Client-Server DBMS," International Conference on Distributed Computing Systems, 1994
[18] D. DeWitt et al., "A Study of three Alternative Workstation-Server Architectures for Object Oriented Database Systems," Proc., VLDB, pp.107-121, 1990.
[19] C. Mohan, Don Haderle, et al, "ARIES : A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," ACM TODS, pp.94-162, 1992.
[20] K. Wilkson and Marie-Anne Neimat, "Maintaining Consistency of Client Cached Data," Proc., VLDB, pp.122-133, 1990.
[21] 이찬섭, 김동혁, 백주현, 최의인, "클라이언트/서버 환경에서 접근 빈도를 고려한 캐쉬 일관성 기법", 한국컴퓨터교육학회 논문지, 2003.
[22] 이찬섭, 김동혁, 백주현, 최의인, "이동 컴퓨팅 환경에서 대기시간을 감소시키는 갱신 빈도 캐쉬 일관성 기법", 한국정보처리학회 논문지, 2003.

저 자 소 개

이 찬 섭

1990년 한남대학교 컴퓨터공학과(학사)
2000년 한남대학교 컴퓨터공학과(석사)
2003년 한남대학교 컴퓨터공학과(박사)
2003년 혜천대학 전자상거래과 전임강사

<관심분야> Caching, Web DB, Mobile Computing,