이동 컴퓨팅 환경에서 대기 시간을 감소시키는 갱신 빈도 캐쉬 일관성 기법

이 찬 섭[†] · 김 동 혁[†] · 백 주 현[†] · 최 의 인[††]

요 약

무선 네트워킹 기술과 통신기기의 발달로 이동 컴퓨팅 환경이 보편화됨에 따라 제한된 대역폭의 절감과 빠른 응답시간, 그리고 확장성을 위해 이동 호스트는 지역 캐쉬를 이용한다. 이때 이동 호스트와 지구국간에 캐쉬된 데이터의 일관성 유지가 필요하며 이에 따라 많은 기법이 제안되고 있다. 기존의 일관성 기법은 탐지기반의 기법들이 주로 사용되며 잦은 접속단절을 고려해 주기적인 무효화 메시지를 브로드캐스트 하여 캐쉬 일관성을 유지한다. 하지만 이러한 기법들은 데이터의 정확성 검사에 따른 전송 메시지 수의 증가나 지연을 통한 철회 단계를 증가시키며 이동 호스트에 캐쉬된 데이터를 삭제함에 따라 자치성과 확장성이 떨어진다. 본 논문에서 제안된 기법은 이러한 문제점을 해결하기 위해 페이지 요청 또는 완료시 갱신 연산이 일어난 객체에 대해 갱신 빈도를 참조하여 수행하도록 하였다. 따라서, 갱신 연산이 이루어지는 경우 비동기적으로 갱신 빈도에 따른 갱신의도 선언 또는 갱신을 선택적으로 수행할 수 있어 응답이 빠르고 철회 단계가 감소하는 장점을 갖는다. 또한 접속단절 이후 일괄적으로 진행되는 주기적인 무효화 메시지에 대해서도 갱신 빈도에 따라 선택적으로 삭제 또는 전파를 수행함으로써 자치성과 확장성을 높였다.

Update Frequency Cache Consistency for Reducing Wait Time in Mobile Computing

Chan-Seob Lee[†] · Dong-Hyuk Kim[†] · Joo-Hyun Baek[†] · Eui-In Choi[††]

ABSTRACT

According as mobile computing environment is generalized by development of wireless networking technology and communication device, mobile host uses local cache for extensibility and early response time, and reduction of limited bandwidth. This time, between mobile host and mobile support station cache need consistency interested person of done data accordingly much techniqueses propose. Existent consistency techniques because detection based techniqueses are used mainly and broadcast periodic invalid message considering frequent disconnection. However, these techniqueses increase abort step through increase or delay of transmission message number by accuracy examination of data. Therefore, because mobile host deletes cached data, and extensity are decreased. Techniques that is proposed in this paper did to perform refering update frequency 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 update frequency electively asynchronously when update operation consists and abort step decreases. Also, improved extensity running delete or propagation electively according to update frequency about periodic invalid message gone since disconnection.

키워드 : 이동 컴퓨팅(mobile computing), 캐쉬(cache), 일관성(consistency), 갱신 빈도(Updata)

1. 서 론

대량의 데이터를 무선으로 빠르게 전송할 수 있는 무선 네트워킹 기술의 발전과 통신기기의 성능향상으로 사용자는 이동 중에도 쉽게 데이터베이스에 접근할 수 있게 되었다. 이동 컴퓨팅 환경에서 이동 호스트들은 빠른 응답과 확장성을 위해 이동 호스트내의 지역 캐쉬에 데이터의 사본을 유지하며 이때, 지구국(Mobile Support Station)과의 캐쉬 일관성 유지하기 위해 많은 기법이 연구되고 있다[1]. 이동 컴퓨팅 환경은 고정 컴퓨팅 환경에 비해 좁은 대역폭과 잦은 단절이 발생된다. 즉, 빈번한 데이터 요청으로 인해 통신 채널에 대한 점유 경쟁이 발생하고, 셀을 벗어나거나 제한된 배터리의 수명으로 의도적 혹은 비의도적인 통신 단절이 일어나므로 상태 유지가 어렵다. 따라서, 기존의 일관성 유지 기법들에 대한 적절한 재구성이 필요하다[2].

† 준 회 원 : 한남대학교 대학원 컴퓨터공학과
†† 종신회원 : 한남대학교 컴퓨터공학과 교수
논문접수 : 2002년 9월 30일, 심사완료 : 2002년 10월 19일

이 논문에서는 좁은 대역폭 내에서 지구국에 빈번히 일어나는 데이터 요청으로 발생되는 대역폭의 점유 경쟁을 줄일 수 있도록 메시지 전송 횟수를 줄이면서 빠른 응답이 가능하도록 하는 기법과 통신 단절에 대해 일괄적으로 진행되었던 무효화 메시지로 인해 자치성이 떨어지는 것을 줄일 수 있도록 선택적인 삭제 또는 전파를 하는 캐쉬 일관성 유지 기법을 제안하였다. 즉, 갱신이 자주 일어나는 데이터와 자주 일어나지 않는 데이터를 구분하여 비동기적 갱신-의도를 선언하고 Commit 단계에서 정확성 검사를 수행하게 함으로써 트랜잭션의 신속한 처리 및 메시지 전송 횟수를 감소시켰고, 다른 이동 호스트들로부터 CB(callback) 응답 전에 갱신-의도를 선언한 이동 호스트에 응답함으로써 신속한 응답이 가능하도록 하였다. 또한 갱신 빈도에 따라 삭제나 전파가 선택적으로 이루어지도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 이야기하고, 3장에서는 본 논문에서 제안한 기법을 위한 시스템 모델과 갱신 빈도를 이용한 효율적인 일관성 기법 그리고 트랜잭션 처리를 위한 시나리오를 기술하며, 4장에서는 시나리오를 바탕으로 전송 메시지 수와 대기 시간 등을 통하여 제안한 기법과 기존의 다른 일관성 기법을 비교 분석한 후, 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

이동 컴퓨팅 환경은 고정 컴퓨팅 환경과 달리 이동 호스트와 지구국간의 무선 구간과 지구국과 고정 호스트간의 유선 구간으로 구분되며 이동 호스트와 지구국간의 관계는 고정 컴퓨팅 환경에서의 클라이언트와 서버간의 관계에 해당된다. 고정 컴퓨팅 환경에서 캐쉬 일관성 유지 기법은 트랜잭션이 클라이언트에 캐쉬된 데이터에 접근할 때 그 데이터의 정확성 검사 확인 여부에 따라 탐지-기반(Detection-based) 기법과 회피-기반(Avoidance-based) 기법으로 구분한다. 이동 컴퓨팅 환경에서 이동 호스트와 지구국 사이는 NWL-MH와 같은 탐지기반의 기법을 사용하고, 고정 호스트들 사이는 O2PL 같은 회피-기반기법이 사용되며[1], 단절 상태에서 일관성 유지를 위해 데이터베이스의 갱신을 정기적으로 방송(broadcast)하는 TS(broadcasting TimeStamp), AT(Amnesic Terminals), SIG(SIGnatures)와 같은 방법이 사용된다[3].

2.1 탐지기반의 일관성 유지 기법

탐지기반의 일관성 유지 기법들은 지역 캐쉬에 부정확한 데이타(stale data)가 있는 것을 허용하므로 판독 또는 기록 연산 수행시 서버에 정확성 검사를 하는 기법이다.

C2PL은 일관성 유지를 위하여 클라이언트/서버간의 동기화가 필요하며, 다른 클라이언트에게 갱신통보를 보내지 않고 모든 록킹과 교착상태 탐지에 대한 책임을 서버가 갖는다. 중앙 집중적인 데이타베이스 시스템 환경에서 이용되는 2PL(two phase locking)기법을 클라이언트/서버 환경에서 클라이언트가 캐쉬를 이용하는 환경으로 확장시킨 기법으로 구현이 간단한 반면 판독과 기록 연산 수행 시 매번 서버에 정확성 검사 확인 절차를 거치므로 클라이언트와 서버간 메시지 전송이 너무 자주 일어나고 대역폭을 많이 차지하는 단점을 가진다[8, 16, 19].

NWL은 C2PL과달리 기록 연산 수행시에 서버에게 정확성 검사 절차를 수행하게 함으로써 메시지 전송 횟수를 줄인 비동기적 기법이며[16], NWL-MH는 이동환경에 적용시킨 기법으로 서버는 갱신된 데이터들을 정기적으로 브로드캐스트 하여 캐쉬에 있는 오래된 값들을 무효화한다[1].

AOCC 기법은 트랜잭션이 클라이언트에 캐쉬된 객체가 정확하고, 현재 다른 트랜잭션에 의해 이용되지 않고 있다는 가정 하에 객체를 이용하는 낙관적 성격을 갖는 탐지 기반 기법이다. fetch, commit, abort 요구 이외의 메시지는 피기-백(piggy-back) 기법을 이용하여 전송하고, 트랜잭션은 클라이언트의 종료 시점까지 서버와의 통신을 하지 않음으로써 메시지 전송비용을 줄인 기법이다[15, 20].

2.2 회피기반의 일관성 유지 기법

회피기반의 일관성 유지 기법들은 자신의 지역에 있는 정확하지 않은 데이타 참조 기회를 주지 않음으로써 일관성을 유지한다.

CB 기법은 동기적인 일관성 행동에 기반을 두는 회피 기반 기법으로 클라이언트는 트랜잭션에게 지역 록을 허용하기 전에 즉시 서버에게 갱신하려는 페이지에 대해 자신의 갱신의도를 선언한다. 클라이언트가 다른 클라이언트에 캐쉬된 페이지에 갱신의도를 선언할 때, 서버는 그 페이지를 가지고 있는 사이트에 메시지를 보냄으로써 충돌이 발생한 사본을 "callback"하여 일관성을 유지하는 기법이다[16, 17, 21].

ACBL 기법은 CB 기법을 확장한 기법으로 동시성 제어에 제한적인 페이지/서버 구조를 동시성 제어와 사본 관리에 대해 객체와 페이지 단위로 선택적으로 수행하도록 하는 기법이다[14, 15].

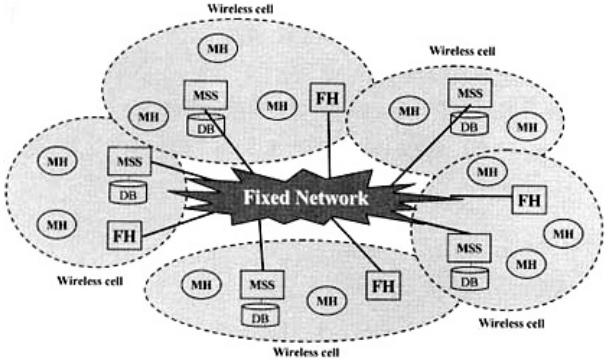
AACC 기법은 클라이언트/서버 모두 록 관리를 수행하며 페이지와 객체 단위의 록 관리를 하며, 판독 록을 한 클라이언트가 페이지를 캐쉬함을 의미하는 개인-판독 록(private-read lock)과 페이지가 여러 클라이언트에 캐쉬됨을 의미하는 공유-판독 록(shared-read lock)으로 구분하여 효율성을 높인 기법으로 ACBL 기법보다 성능이 좋고 AOCC 기법보다 철회율이 낮다는 특징을 갖는다[13].

O2PL 기법은 판독 및 기록 록을 지역적으로 획득하고 트랜잭션 완료 때까지 유지하며 완료 때 서버에 정확성 검사를

하므로 C2PL에 비해 메시지 교환 횟수를 현저히 줄였으며 다른 클라이언트에 삭제 또는 전파에 따라 O2PL-I와 O2PL-P로 구분된다[8, 17].

3. 제안한 캐쉬 일관성 유지 기법

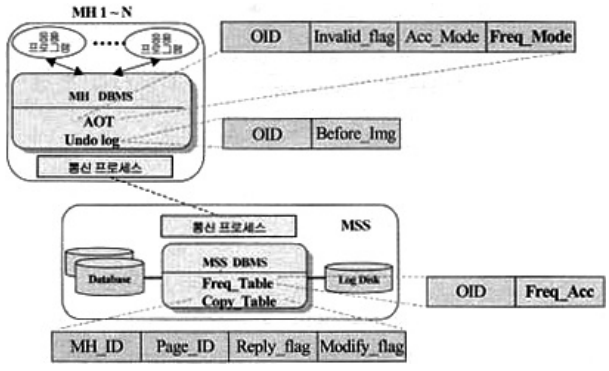
본 논문의 연구 환경은 (그림 1)과 같이 하나의 지구국에 다수의 이동 호스트들이 고속 무선 네트워크로 연결된 시스템으로 데이터는 데이타 전송(data-shipping) 형태를 가지며 록 단위를 페이지와 객체로 한다[9, 10, 14, 18, 22, 23].



(그림 1) 이동 컴퓨팅 환경

3.1 이동 호스트/지구국 구조

아래 모델 (그림 2)에서 지구국은 가장 최근의 원본 데이터를 유지하고 있으며 이동 호스트의 데이터 요구시 데이터의 사본을 복사해 준다. 지구국은 이동 호스트에 캐쉬된 이 데이터의 정보를 유지하고 있으며, 이동 호스트의 갱신 의도에 따라 해당 데이터의 전역적인 록 관리를 하고 페이지가 갱신되면 갱신되기 전의 사본을 캐쉬하고 있는 이동 호스트들에게 갱신을 통보한다. 지구국은 데이터베이스를 저장하는 안전한 저장장치인 디스크, 로그 디스크, 이동 호스트와 통신을 담당하는 통신 프로세스, 그리고 지구국 DBMS로 구성되며 지구국 DBMS는 동시성 제어기, 자원 관리기, 회복 관리기로 구성된다.

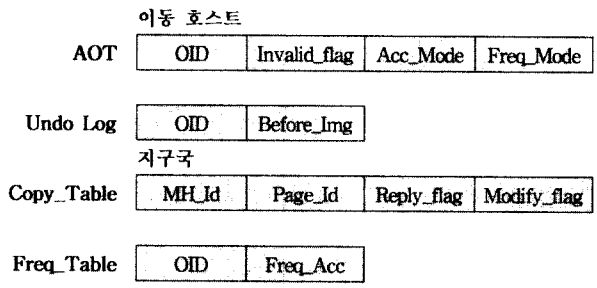


(그림 2) 이동 호스트/지구국 모델

이동 호스트는 자신의 버퍼에 캐쉬한 페이지의 정보를 유지하며 지역적인 록 관리를 수행한다. 데이터 갱신시 이동 호스트는 지구국에 갱신 의도를 선언한 후 곧바로 갱신 연산을 수행한다. 이동 호스트는 트랜잭션을 생성하는 응용 프로그램, 지구국과의 통신을 담당하는 통신프로세스, 그리고 이동 호스트 DBMS로 구성되며 이동 호스트 DBMS는 지역 동시성 제어기, 이동 호스트 버퍼 관리기, 이동 호스트 관리기, 자원관리기로 구성된다. 이동 호스트 버퍼 관리기는 현재 이동 호스트 버퍼에 캐쉬한 페이지들의 정보를 유지한다[5-7].

3.2 데이터 구조

제안한 기법은 지구국과 이동 호스트간의 데이터 전송을 페이지 단위로 하고 록 관리는 객체 단위로 한다. 이동 호스트의 사본 데이타 관리는 페이지 또는 객체 단위를 선택적으로 한다. 데이터 구조는 아래 (그림 3)와 같다. 이동 호스트는 자신의 메모리에 캐쉬된 페이지에 속해있는 객체에 대한 정보(AOT : Access Object Table)와 철회시 이전 데이터로 되돌리기 위한 정보(Undo log)를 유지하고, 지구국은 각 이동 호스트들에게 복사해준 페이지에 어떤 객체들이 속해있는지에 대한 정보(Copy_Table) 테이블과 갱신 빈도(Freq_Table) 테이블을 유지한다.



(그림 3) 데이터 구조

- AOT(Access Object Table)는 OID, Invalid_flag, Acc_Mode, Freq_Mode로 구성된다.
  - OID : 트랜잭션이 이용하는 객체 식별자를 나타낸다.
  - Invalid_flag : 트랜잭션이 종료 시 삭제될 객체를 나타내는 플래그이다. 갱신 통보를 받은 객체가 현재 판독 모드로 이용중이면, Invalid_flag = 1로 이용중이 아니면 Invalid_flag = 0으로 설정한 후 트랜잭션 종료와 함께 Invalid_flag = 0인 객체를 삭제함으로써 캐쉬된 데이터의 일관성을 유지한다.
  - Acc_Mode : 객체에 적용한 연산의 종류를 나타내는 접근 모드이다. 판독 연산일 경우 Acc_Mode = 0, 갱신 연산일 경우 Acc_Mode = 1을 유지한다.
  - Freq_Mode : 갱신 의도 선언 또는 갱신을 선택할 수 있도록 하는 모드이다. Freq_Mode = 0이면 갱신 연산

을 먼저 수행한 후 Commit 단계에서 갱신 의도를 선언하고, Freq_Mode = 1이면 갱신 의도부터 선언한 후 갱신을 수행한다.

- Undo log는 OID, Before_Img로 구성된다.
  - Before_Img : 갱신되기 전의 데이터를 저장하며 트랜잭션 철회시 이전 데이터로 복구하기 위해 사용된다.
- Copy_Table은 Client_ID, Page_ID, Reply_flag, Modify_flag로 구성된다.
  - MH_ID : 페이지를 캐쉬한 이동 호스트 식별자를 나타낸다.
  - Page_ID : 사본의 페이지 식별자를 나타낸다.
  - Reply_flag : 갱신통보를 보낸 이동 호스트로부터의 응답 메시지 회신 여부를 나타낸다. Reply_flag = 0이면 회신을 받았거나 갱신통보 메시지를 보내지 않은 상태이고, Reply_flag = 1이면 이동 호스트에게 갱신통보를 보내고 아직 응답 메시지를 받지 않은 상태를 나타낸다.
  - Modify_flag : 이동 호스트로부터 갱신된 페이지가 지구국의 디스크에 반영여부를 나타낸다. Modify_flag = 0이면 반영된 상태이고, Modify_flag = 1이면 아직 반영되지 않은 상태를 나타낸다.
- Freq_Table은 OID, Freq_Acc로 구성된다.
  - Freq_Acc : 객체의 갱신 정보를 나타낸다. 초기값은 0이며 해당 객체가 갱신될 때마다 1씩 증가한다.

3.3 이동 호스트/지구국 처리 절차

3.3.1 이동 호스트 처리 절차

이동 호스트에서는 일관성 유지를 위해 자신의 버퍼에 캐쉬한 페이지에 대해 트랜잭션이 수행하는 판독, 갱신, 트랜잭션 종료와 지구국으로부터 받는 메시지 갱신 통보 및 종료 허가가 발생할 때마다 AOT에 객체의 정보를 유지하며, 철회를 대비해 Undo log에 갱신 연산 수행전의 객체를 저장한다.

- 페이지 요청
  - 이동 호스트의 캐쉬에 데이터가 부재중일 경우 객체 식별자(OID)와 접근모드(Acc_Mode)로 지구국에 객체가 포함된 페이지를 요청한다.
  - 판독 연산
- AOT에 OID, Invalid_flag = 0, Acc_Mode = 0, Freq_Mode를 갱신한다. Freq_Mode는 페이지 요청시 지구국에서 받은 빈도에 따른 0 또는 1값을 유지하고 있다.
- 갱신 연산
  - AOT의 Acc_Mode = 1로 갱신하고 철회를 대비해 Undo log에 이전 객체를 저장한다. Freq_Mode에 따라 Freq_Mode = 0인 경우 갱신을 먼저 수행한 후 종료 요구 메시지와 AOT, 갱신된 객체를 지구국으로 전송한다. Freq_Mode = 1인 경우 갱신전에 지구국에 갱신 의도를 선언한다.
- 갱신 통보
  - AOT에 삭제할 객체가 포함되어 있고 객체가 사용중이 아닌 경우 페이지를 삭제한다. AOT의 Acc_Mode = 0인 경우 객체가 판독 모드로 사용중이므로 Invalid_flag = 1로 갱신하여 트랜잭션 종료시 삭제한다. Acc_Mode = 1인 경우 객체가 갱신 모드로 사용중이므로 트랜잭션을 철회한다.
- 종료
  - 갱신이 완료된 객체는 종료 요구 메시지와 AOT, 갱신된 객체를 지구국에 전송한 후 지구국의 응답을 기다린다. 지구국의 응답이 종료 허가일 경우 Invalid_flag = 1인 객체와 Undo log의 데이터를 삭제하고 AOT를 갱신한다. 철회 메시지를 받은 경우 Undo log를 이용해 갱신되기 이전의 상태로 되돌린다.

3.3.2 지구국 처리 절차

지구국에서는 일관성 유지를 위해 이동 호스트의 페이지 요청과 갱신의도 선언에 대한 록관리와 록에 따른 철회 메시지, invalid 메시지, 종료허가 연산을 위해 Copy_Table을 유지하며, 이동 호스트가 갱신빈도에 따라 갱신 또는 갱신 의도 선언을 선택적으로 할 수 있도록 Freq_Table을 유지한다.

- 데이터 요청
  - 요청 받은 OID가 갱신 록이 걸려있으면 종료할 때까지 기다리며 일부 객체가 갱신중인 경우 사용 불가능 표시를 한다. Copy_Table에 정보를 갱신한 후 Freq_Mode를 포함한 페이지를 이동 호스트에게 전송한다.
- 이동 호스트의 갱신 의도 선언 요구
  - 이동 호스트의 Freq_Mode = 0인 경우 갱신 시 갱신 의도 선언하는 절차와 동일하게 수행된다.
    갱신 의도를 받은 OID에 갱신 록이 걸려있는 경우 철회하도록 응답하며 그렇지 않은 경우 갱신 록을 설정한다. 해당 데이터를 가지고 있는 이동 호스트에 Callback 메시지를 전송한다. Freq_Acc의 값을 1증가시키며 이동 호스트에서 정해놓은 제한 값에 따라 이동 호스트의 Freq_Mode을 변경시킨다.
- 갱신 통보에 대한 응답
  - Callback 메시지를 전송했던 이동 호스트의 응답 없이 즉시 갱신 의도 선언을 요구했던 이동 호스트에 Commit 응답을 한다. Callback 메시지를 전송했던 이동 호스트로부터 Invalid을 받은 경우 Copy_Table의 Reply_flag를 0으로 갱신한다.
- 주기적인 삭제 및 전파

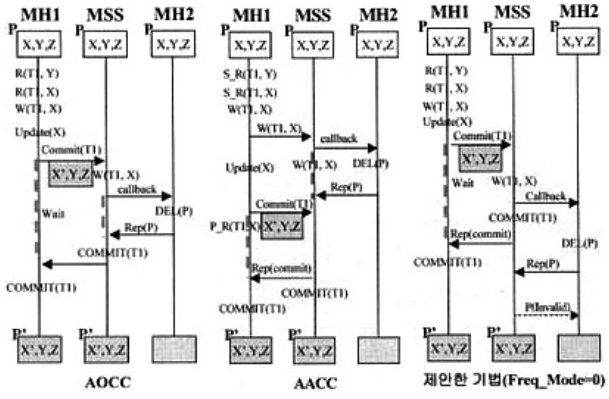
• 단절 상태를 고려하기 위해 지구국은 Freq_Acc의 값에 따라 갱신 빈도가 적은 데이터에 대해 주기적인 무효화 메시지를 이동 호스트들에게 보내며 갱신 빈도가 많은 데이터에 대해 이동 호스트들에게 주기적으로 갱신데 데이터를 전파한다.

3.4 일관성 유지를 위한 시나리오

이동 호스트 1과 이동 호스트 2에서 다음과 같은 연산 순서로 트랜잭션(T1, T2)이 각각 T1 : read(Y) -> read(X) -> write(X), T2 : read(Z) -> read(X)와 같이 수행된다고 가정하며 제안한 기법은 갱신이 드물게 일어나는 경우(Freq_Mode = 0)만을 비교하였다. 갱신이 자주 일어나는 경우는 갱신 의도를 먼저 선언하는 차이가 있다. 아래 (그림 4)~(그림 6)은 트랜잭션 수행 과정에서 발생할 수 있는 3가지 캐쉬 일관성 수행 시나리오이다. 록의 단위가 페이지와 객체이고 빠른 응답 시간과 철회율 감소의 특성을 갖고 있으며 다른 기법에 비해 가장 최근 연구되었던 AOCC, AACC 기법과 비교하였다.

3.4.1 시나리오 1

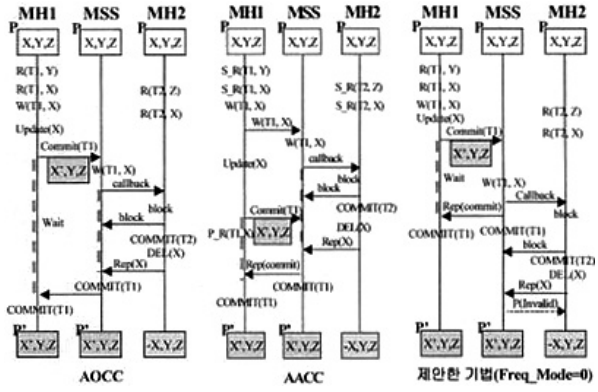
(그림 4)은 MH1에서만 갱신이 일어나고 MH2에서는 아무 연산도 이루어지고 있지 않은 경우이다. T1은 갱신 연산을 먼저 수행한 후 지구국에 갱신의도를 선언한다. 지구국은 현재 X에 록이 걸려 있지 않기 때문에 갱신 록을 설정하고 X의 사본을 갖고 있는 MH2에게 갱신 통보 메시지를 보낸다. MH2는 X를 사용하고 있지 않으므로 X를 삭제하고 지구국에 삭제 메시지에 대한 응답을 한다. 지구국은 MH2의 응답을 받은 후 MH1에 종료 허가 메시지를 전송한다.



(그림 4) 페이지 삭제

3.4.2 시나리오 2

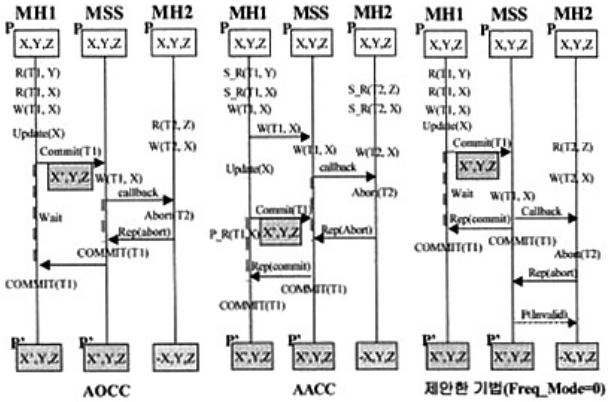
(그림 5)는 MH1에서 갱신하고자 하는 객체를 MH2에서 판독 연산중인 경우이다. 시나리오 1과 달리 객체가 사용중이므로 페이지를 삭제하지 않고 해당 객체만 삭제한다.



(그림 5) 객체 삭제

3.4.3 시나리오 3

(그림 6)는 MH1에서 갱신하고자 하는 객체를 MH2에서 갱신의도를 선언하거나 갱신 연산을 수행하고 있는 경우로 MH1이 MH2보다 Commit 선언을 먼저 했으므로 MH2의 트랜잭션 T2의 갱신은 철회하게 된다.



(그림 6) 트랜잭션 철회

AACC 기법은 AOCC 기법 보다 낮은 철회율과 높은 성능을 가지며 충돌이 적은 상황에서 좋은 성능을 보이고 있다. 갱신 연산을 수행하기 전에 먼저 갱신의도를 선언하기 때문에 충돌을 일찍 발견하여 철회 동작을 줄인다. 판독 록에 대해 Private-Read와 Shared-Read로 페이지를 구분함으로써 P-Read시에는 갱신 연산을 그냥 수행하고, S-Read시에는 갱신의도 선언 후 다른 이동 호스트의 CallBack을 기다리지 않고 갱신 연산을 수행하는 방법으로 성능을 높였다. AACC 기법이 갱신의도부터 선언하면서도 높은 성능을 가지는 이유는 한 이동 호스트에만 캐쉬된 페이지에 대한 갱신이 일어나는 경우 갱신의도 선언 없이 갱신 연산을 먼저 수행하기 때문이며 CallBack을 기다리지 않고 이동 호스트에서 갱신 연산을 먼저 수행하기 때문이다. 여러 이동 호스트에 캐쉬된 페이지에 대한 갱신은 한 이동 호스트에만 캐쉬된 페이지에 대한 갱신과 다르다. AOCC와 제안한 기법(Freq_Mode = 0)은 갱신 연산을 먼저 수행한 후 Commit 단계에서 정확성 검사를 수행하므로 높은 성능을 가진다. 대신 Commit 단계

에서 정확성 검사를 수행하므로 충돌을 늦게 발견할 수 있어 철회 단계가 길어지는 단점이 있을 수 있다.

시나리오 1, 2, 3과 같이 AOCC 기법은 메시지 전송 횟수가 적은 장점이 있으나 갱신의도를 선언하지 않고 갱신을 수행함으로써 철회시에 갱신했던 데이터를 갱신전 데이터로 되돌리는데 소요되는 단계가 증가될 수 있다. AACC는 여러 이동 호스트에 사본이 복사된 경우 Shared-read를 사용하고 갱신의도부터 선언하므로 철회 단계를 줄일 수 있으나 전송 횟수가 AOCC 또는 제안한 기법보다 많으므로 일반적으로 한 이동 호스트에 사본이 복사된 경우와 같은 특정 환경에 유리하다. 제안한 기법은 AOCC와 유사하나 갱신빈도(Freq_Mode = 0, 1)에 따라 갱신연산을 수행할 수 있어 특정 환경에 구애받지 않으며 AOCC와 메시지 전송 횟수는 같으나 이동 호스트의 갱신의도 선언에 대해 Callback 요청에 대한 응답 없이 즉시 Commit 선언을 함으로써 이동 호스트의 대기기간과 지구국의 대기시간의 총 합이 가장 짧아 빠른 응답이 가능하다. 또한 단절 상태를 고려하여 Freq_Mode = 0인 경우에는 갱신이 자주 일어나지 않는 데이터이므로 주기적인 무효화 메시지를 이동 호스트에 전송하며 Freq_Mode = 1인 경우에는 자구 갱신이 일어나는 데이터이므로 주기적인 갱신 데이터를 전파한다.

4. 비교 분석

이 장에서는 이전 장의 시나리오를 기반으로 AOCC, AACC 기법과 제안된 기법에 대해 비교 분석한다.

〈표 1〉 트랜잭션 처리 시간 계산 요소

$E = T_{CPU} + T_{NET} + T_{I/O} + T_{Valid}$
T_{CPU} = (트랜잭션이 이용하는 데이타 수 + 갱신 연산 수) × 읽기 연산 처리비용
T_{NET} = 전송에 필요한 비용($T_{NET/CPU}$) + 데이타/메시지를 실제 전송하는데 걸리는 시간($T_{NET/Transmit}$)
$T_{NET/CPU}$ = 트랜잭션 크기×(이동 호스트에서 메시지 처리비용 + 지구국에서 메시지 처리비용 + 지구국에서 전송비용) + 갱신 데이타들×(이동 호스트에서 전송비용)
$T_{NET/Transmit}$ = 네트워크를 통해 전송하는데 걸리는 시간
Tvalid = 지구국에서의 메시지 처리비용×(이동 호스트 수 − 1) + 이동 호스트에서의 메시지 처리비용

<표 1>과 같이 트랜잭션 수행 시간(E)은 트랜잭션이 연산을 적용하기 위하여 페이지를 참조하는데 사용되는 CPU 사용 시간(TCPU)이며, 이동 호스트와 지구국 사이에 페이지와 메시지를 전송하는데 걸리는 시간은 TNET이고, 트랜잭션이 참조하는 페이지를 디스크로부터 읽어오는데 필요한 디스크 I/O(TI/O) 시간 그리고 트랜잭션의 갱신 연산으로 인해 이동 호스트와 지구국 사이에 발생하는 네트워크 통신비용(TValid)으로 정의할 수 있다.

각 기법들에 대한 비교를 위해서 트랜잭션이 연산 처리를 위해 페이지를 참조하는 비용과 이동 호스트와 지구국 사이의 메시지와 페이지 전송 비용은 같다고 가정한다. 따라서 각 기법들에서 트랜잭션 처리비용은 이동 호스트와 지구국 사이에서 전송하는 메시지 수와 이동 호스트와 지구국에서 요구를 보내고 응답을 대기하는 시간으로 비교할 수 있다.

시나리오 1의 경우, 제안한 기법은 갱신 후 Commit 단계에서 갱신 의도를 선언함으로써 메시지 수는 AOCC와 같으나 응답 대기 시간이 짧아졌으며 AACC에 비해서는 메시지 수와 응답 대기 시간이 짧아졌음을 알 수 있다. 시나리오 2와 3의 경우에서도, 한 이동 호스트의 갱신 연산으로 인해 다른 이동 호스트의 갱신 연산 중에 블록이 발생하지만 시나리오 1의 경우에서처럼 메시지수와 응답 대기 시간에서 효율적임을 알 수 있다.

〈표 2〉 기존 캐쉬 일관성 기법과의 비교

		AOCC	AACC	제안한 기법
기 반		탐지기반	회피기반	탐지/회피 기반
록 획득방법		지 연	비동기	비동기/동기적
사본관리		삭 제	삭 제	삭제/전파
메시지 수		적 음	보 통	적 음
대기시간/응답속도		짧음/보통	짧음/보통	가장 짧음/빠름
록 단위		객 체	페이지, 객체	페이지, 객체
데이타 구조	이동 호스트 (클라이언트)	ROS, MOS, Undo log		AOT, Undo log
	지구국(서버)	MOS, Dir(P) Invalid(C),		Copy_Table Freq_Table
기 타		충돌이 적은 환경	p-read, s-read로 lock 구분	일반적인 환경

<표 2>는 기존 캐쉬 일관성 기법중 록단위를 페이지와 객체로 하고 CB 기반의 기법인 AACC, AOCC와 본 논문에서 제안한 캐쉬 일관성 유지 기법을 비교한 결과이다. AOCC 기법은 충돌이 적은 환경에서는 유리하나 갱신이 많은 환경에서는 충돌에 따른 철회 절차가 길어지는 단점을 갖는다. AACC 기법은 판독 록을 개인 판독 록(Private-read)과 공유 판독 록(Shared-read)으로 구분함에 따라 데이터가 한 이동 호스트에 캐쉬되어 있는 경우 갱신을 먼저 수행함으로 성능이 좋아지며 여러 이동 호스트에 캐쉬되어 있는 경우는 갱신 의도부터 선언함으로 철회율을 낮출 수 있는 장점이 있지만 두 개의 이동 호스트에 캐쉬되어 있고 갱신이 자주 일어나지 않는다 하더라도 항상 공유 판독 록(Shared-read)를 획득하여 갱신 의도부터 선언함으로 충돌이 적은지 많은지를 판별할 수 없는 단점이 있다. 본 논문에서 제안한 기법은 이동 호스트들이 갱신 시에 Freq_Mode에 따라 Freq_Mode = 0인 경우에 자주 갱신이 일어나지 않는 데이터로 간주하고 갱신 연산을 수행한 후 완료 시점에서 지구국에 갱신 의도를 선언하며 트랜잭션 처리율을 높이도록 하였으며, Freq_Mode = 1인 경우에는 빈번히 갱신이 일어나는 데이터로 간주하여 갱신 의도를 선언한 후 갱신을 수행하므로 트랜잭션을 철회율을 낮춤으로써 충돌 가능성을 판단하여 효율

적인 처리를 할 수 있다는 장점이 있다. 또한 Freq_Mode에 따라 선택적으로 삭제와 전파를 함으로써 단절 상태에서 발생하는 일관성 문제를 해결하고 있다.

5. 결 론

본 논문에서는 이동 컴퓨팅 환경에서 갱신 빈도에 따른 캐쉬 일관성 유지 기법을 제안했다. 본 논문의 특징은 데이터의 갱신에 대해 선택적으로 갱신 의도 선언 시나 갱신 후에 지구국에게 갱신 의도를 선언할 수 있도록 하여 충돌이 자주 일어나거나 자주 일어나지 않는 경우에 모두 효과적으로 적용될 수 있으며, 단절 상태를 고려하여 선택적으로 삭제 및 전파가 이루어지도록 하여 이동 호스트의 자치성을 높였다. Freq_Mode = 0인 경우 갱신을 먼저 한 후 완료 시점에서 갱신 의도를 선언하므로 지구국의 응답을 기다리는 대기시간을 줄였고 삭제하는 방법으로 단절 상태를 고려하였으며, Freq_Mode = 1인 경우 갱신 의도를 먼저 하도록 하여 철회율을 줄였고 전파하는 방법으로 자치성을 높였다. 따라서, 동기적인 기법들보다 이동 호스트가 지구국의 응답을 기다리는 대기시간을 줄일 수 있고, 트랜잭션의 갱신정보를 종료(commit) 시점까지 지연시키는 기법보다 트랜잭션 철회를 줄였으며 선택적인 삭제와 전파로 자치성을 높였다.

향후 연구 과제로는 기존 기법과 제안한 기법에 대해 트랜잭션을 처리하는데 따른 CPU 사용시간, 이동 호스트와 지구국간 메시지 전송 시간, 디스크 I/O시간, 네트워크 통신 비용 등을 매개 변수로 하여 성능평가가 이루어져야 하며, 제안한 캐쉬 일관성 알고리즘을 기반으로 동시성 제어에 대한 연구가 필요하다.

참 고 문 헌

[1] Jin Jing et al., "Distributed Lock Management for Mobile Transaction," Proc. IEEE Distributed Computing System, 1995.

[2] Michael J. Carey et al., "Conflict Detection Tradeoffs for Replicated Data," ACM Transactions on Database Systems, Vol.16, No.4, Dec., 1991.

[3] Daniel Barbara, Tomasz Imielinski, "Sleepers and Workaholics : Caching Strategies in Mobile Environments," Proc. ACM SIGMOD, pp.1-12, 1994.

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

[5] D. DeWitt et al., "A Study of three Alternative Workstation-Server Architectures for Object Oriented Database Systems," Proc., VLDB, pp.107-121, 1990.

[6] 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.

[7] K. Wilkson and Marie-Anne Neimat, "Maintaining Consistency of Client Cached Data," Proc., VLDB, pp.122-133, 1990.

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

[9] M. Franklin, M. Carey, "Crash Recovery in Client-Server EXODUS," Proc., ACM SIGMOD, pp.165-174, 1992.

[10] M. Franklin, M. Carey, M. Livny, "Local Disk Caching in Client-Server Database Systems," Proc., VLDB, pp. 543-554, 1993.

[11] M. Franklin, M. Carey, M. Livny, "Global Memory Management in Client-Server DBMS Architectures," Proc., VLDB, pp.596-609, 1992.

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

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

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

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

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

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

[18] 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.

[19] Kim, W., Garza, J. F., Ballou, N., And Woelk, D, "Architecture of the ORION next-generation database system," IEEE Trans. pp.109-124, 1990.

[20] Adya, A., Gruber, R., Liskov, B., And Maheshwari, U, "Efficient optimistic concurrency control using loosely synchronized clocks," Proc., ACM SIGMOD, pp.23-34. 1995.

[21] Michael J. Franklin, Michael J. Carey, "Client-Server Caching Revisited," IWDOM, pp.57-78, 1992.

[22] J. C. Lamb, G. Landis and D. Weinreb, "The objectstore database system," ACM, 34(10), 1991.

[23] Alex Delis, Nick Roussopoulos, "Management of Updates in the Enhanced Client-Server DBMS," International Conference on Distributed Computing Systems, 1994.

이 찬 섭

e-mail : cslee@dblab.hannam.ac.kr
1990년 한남대학교 컴퓨터공학과 졸업 (학사)
2000년 한남대학교 대학원 컴퓨터공학과 (공학석사)
현재 한남대학교 대학원 컴퓨터공학과 (박사과정)
관심분야 : Mobile Computing/Middleware, 웹 DB

김 동 혁

e-mail : dhkim@dblab.hannam.ac.kr
1987년 한밭대학교 산업공학과 졸업(학사)
2001년 한남대학교 대학원 컴퓨터공학과 (공학석사)
현재 한남대학교 대학원 컴퓨터공학과 (박사과정)
관심분야 : Data mining, 웹 DB, XML

백 주 현

e-mail : bbaek@nca.or.kr
1999년 한남대학교 컴퓨터공학과 졸업(학사)
2002년 연세대학교 대학원 컴퓨터공학과 (공학석사)
현재 한남대학교 대학원 컴퓨터공학과 (박사과정)
관심분야 : 전자상거래, 실시간 데이타베이스

최 의 인

e-mail : eichoi@hannam.ac.kr
1982년 숭전대학교 계산통계학과 졸업 (학사)
1984년 홍익대학교 전자계산학과 졸업 (이학석사)
1995년 홍익대학교 전자계산학과 졸업 (이학박사)
1985년~1988년 공군 교육사 전산실장
1992년~1996년 명지전문대학 전자계산과 조교수
1996년~현재 한남대학교 컴퓨터공학과 부교수
관심분야 : 실시간데이타베이스, 주기억데이타베이스, 클라이언트/서버 데이터베이스