복제 일관성을 위한 혼합 그룹 갱신 프로토콜 설계

Design of Hybrid Group Update Protocol for Replica Consistency

이 병 욱*
Byung-Wook Lee

요 약

분산 데이터베이스에서 사본을 복제하는 것은 성능, 가용성, 자치성 향상을 위한 것이다. 그러나 일관성을 엄격하게 적용하는 것은 실제 시스템에서 매우 어렵다. 본 논문에서는 지연 갱신 전파로 일관성을 완화하고, 즉시 갱신 전파로 그룹 갱신을 허용하는 혼합 프로토콜을 제시한다. 이 프로토콜은 그룹 갱신의 일관성을 위하여 사본들의 버전번호를 관리하고, 지연 갱신 트랜잭션의 직렬성을 위하여 반순서화한 사본 스패닝 트리를 생성한다. 비동기 갱신과 동기 갱신 기법을 혼합하여 약한 일관성을 유지한다. 성능 분석에 관한 연구는 향후 연구로 한다.

Abstract

Replication in distributed databases is to improve efficiency, availability, and autonomy. But applying the strict consistency in real systems is very difficult. In this paper, I introduce a hybrid protocol permitting group update through eager update propagation and weakening consistency through lazy update. This protocol manages replica version number for the group update consistency and generates partially ordered replica spanning tree for the serializability of lazy update transactions. It preserves weak consistency by combining synchronous and asynchronous update mechanism. Study of performance analysis for the efficiency of the protocol be included in next paper.

1. 서 론

분산 데이터베이스 시스템은 데이터베이스 일관성뿐만 아니라 복제 일관성을 충족해야 한다. 특히 데이터 웨어하우스와 데이터 마트 분야에서 필요성이 대두되고 있다. 분산 환경에서 복제 일관성에 강한 일관성을 부여하면 정확성은 향상되지만 성능이 저하되므로 해당 응용분야에 적합한 일관성을 설계하는 것이 본 연구의 목표이다. 자료의 가용성과 성능을 향상하기 위하여 복제를 지원하고 있으나 다른 사이트의 사본들을 갱신하는 방법에 따라 일관성 유지가 문제되고 있다. 복제 일관성을 위하여 갱신을 전파하는 방법에는 즉시(eager) 갱신 전파 기법과 지연(lazy) 갱신 전파 기법이 있다[1,7].

즉시 갱신 전파 기법은 갱신 시에 모든 노드의 사본들을 하나의 원자성 트랜잭션으로 동기화시킨다. 2단계 완료 프로토콜과 원본 사이트 록킹(PSL, primary site locking) 등의 관련 알고리즘이 있으나 너무 일관성이 엄격하여 비현실적이다. 따라서 일관성을 약화시킨 기법들을 적용하여 사용하고 있다[5,8].

지연 갱신 전파 기법은 갱신 트랜잭션이 완료한 후에 비동기적으로 다른 사이트의 사본들에게 갱신을 전파한다. 지연 갱신은 직렬화 문제가 탐지되었을 때는 이미 다른 사이트의 트랜잭션들이 완료된 것을 해결하기 힘들다. 즉, 갱신 도중에 탐지되는 충돌 갱신이 문제이다. 완료된 복제 갱신 결과를 복원시키는 자동화 도구가 없기 때문이다[2].

분산 복제에서는 자료를 갱신하는 자격에 따라 원본 갱신 기법과 그룹 갱신 기법이 있다. 원본 갱신 기법은 원본 사이트에서만 갱신이 가능하고, 사본 사이트들은 읽는 것만 가능하므로 다중읽기/

* 종신회원 : 경원대학교 컴퓨터공학과 교수
leebw@mail.kyungwon.ac.kr

단일쓰기(multiple read single write) 기법이라고도 한다. 그룹 갱신 기법은 모든 사이트에서 자료를 읽고 갱신하는 것이 가능하므로 다중읽기/다중쓰기(multiple read multiple write) 기법이라고 하며, 절차가 복잡하지만 모든 사이트가 동일한 코드를 사용할 수 있는 장점이 있다[7].

원본 갱신 기법은 일관성을 엄격하게 유지할 수 있으나 사용자 편의성이 낮으며, 그룹 갱신 기법은 절차가 복잡한 대신 사용자 편의성이 높다. 본 연구는 사본 사이트에서 갱신하기 위하여 원본에 즉시 갱신을 적용하고, 나머지 사본들에게 지연 갱신 전파를 적용하는 혼합 그룹 갱신(HGU, hybrid group update) 프로토콜을 제안한다.

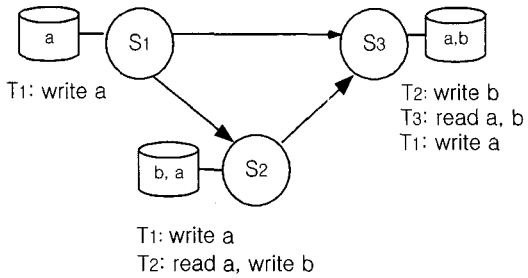
2. 직렬성 문제

지연 갱신 기법은 트랜잭션의 크기가 감소하고 성능이 우수하지만 비직렬성에 의한 충돌 연산이 문제가 된다. 사이트마다 자료 갱신 트랜잭션을 완료한 다음에 다른 사이트의 사본들을 갱신하므로 이미 완료된 트랜잭션에서 충돌이 발견되면 문제 해결이 어렵다. 분산 트랜잭션들의 직렬성을 사전에 확보하고 갱신하려면 병행성이 저하된다.

[예제 1]

그림 1과 같이 사이트 S_1에는 원본 a가 저장되어 있고, 사이트 S_2에는 사본 a와 원본 b가 저장되어 있고, 사이트 S_3에는 사본 a, b가 저장되어 있다. 사이트 S_1에서 트랜잭션 T_1이 a를 갱신하고, 사이트 S_2에서 트랜잭션 T_2가 a를 읽고 b를 갱신하고, 마지막으로 사이트 S_3에서 트랜잭션 T_3는 a와 b를 읽는다.

예제 1은 지연 갱신을 진행하고 마지막 단계에서 비직렬성을 확인하게 되는 대표적인 실례이다. 트랜잭션을 실행한 결과 순환이 발견되었을 때는 문제를 해결하기에 너무 늦다. 네트워크 사정으로 인하여 사이트 S_2에서는 T_1T_2 순으로 실행되고, 사이트 S_3에서는 $T_2T_3T_1$ 순으로 실행될 수가 있다.



(그림 1) 지연 갱신 기법에 의한 비직렬성 문제

따라서 전역 사본 그래프는 $T_1T_2T_3T_1$이 되어 순환을 야기한다. 이 문제를 해결하기 위하여 PSL 기법과 DAG(WT) 기법 등이 있다[8].

2.1 관련 프로토콜

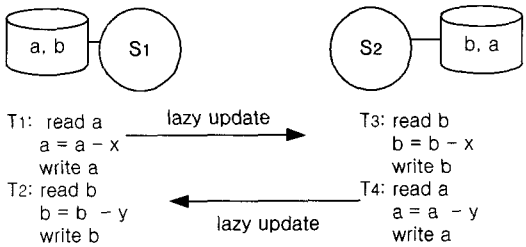
예제 1과 같은 갱신 기법으로 야기되는 문제를 해결할 수 있는 기법이 PSL이며, DAG(WT)이 가장 대표적인 프로토콜이다. 그러나 이들 기법들에도 부분적인 일관성 문제점이 있어서 DAG(WT)의 기능을 강화한 것이 Backedge 혼합 프로토콜이다[5].

2.1.1 DAG(WT) 프로토콜

PSL 프로토콜은 복제 일관성을 유지할 수 있으나 사본 사이트에서는 갱신 할 수 없고, 사본을 읽으려면 원본 사이트에서 록을 얻어야 하므로 성능 저하가 따른다. [5]의 DAG(WT) 프로토콜은 PSL 프로토콜보다 일관성을 완화한 대신 성능을 향상하였다. 즉, 갱신은 원본 사이트에서만 가능하므로 PSL과 동일하지만, 읽기는 모든 사이트에서 가능하다. 원본 갱신 시에는 트랜잭션이 완료되는 즉시 사본 사이트로 지연 갱신을 전파하여 사본들을 갱신한다. 사본 사이트에서 사본을 읽을 때 원본 사이트에 록을 요청하지 않는다. 지연 갱신에 의하여 야기되는 충돌이 발생하면 이미 완료된 트랜잭션 처리를 회복하기 곤란하므로 사본 그래프에서 트리를 생성하여 처리 순서를 반순서화(partial order)함으로써 직렬성을 유지한다.

〔예제 2〕

어떤 은행이 수표와 저축계좌를 두 사이트 S_1과 S_2의 데이타베이스에 저장한다. 사이트 S_1은 수표계좌 원본 a와 저축계좌 사본 b를 저장하고 S_2는 저축계좌 원본 b와 수표계좌 사본 a를 저장한다.



(그림 2) 지연 갱신의 문제점

그림 2와 같이 사이트 S_1과 S_2가 a와 b에서 각각 전액을 출금하면, 원본 사이트에서 출금하고 사본에서 또 출금하게 되어 이중 출금되는 문제가 있다. 모든 사이트에서 분산된 자료를 읽고 갱신하는 그룹 갱신은 예제 2와 같은 문제점이 있으므로 직렬성을 유지할 수 있는 갱신 기법이 필요하다. 사본 사이트의 트랜잭션이 사본을 갱신할 때는 원본과 함께 일관성을 유지하는 동기화 수단이 필요하다[3].

2.1.2 혼합 Backedge 프로토콜

그룹 갱신 기법은 모든 사이트에서 사본을 읽고 갱신할 수 있으므로 편리하다. 2단계 완료 프로토콜 같은 갱신 기법을 사용하면 모든 사본들이 동기적으로 갱신되기 때문에 강한 일관성이 유지된다. 그러나 성능 향상을 위하여 지연 갱신 기법을 이용하면 직렬성 유지가 어렵다. 예를 들어 예제 2와 같이 기존의 지연 갱신 방식을 적용하면 순환에 의한 오류가 발생할 수 있다.

BackEdge 프로토콜은 DAG(WT) 프로토콜을 확장하여 사본 그래프가 순환을 포함하더라도 직렬화 스케줄을 보장하는 혼합 기법이다[5]. 사본 그래프에서 특정한 간선(edge)들을 삭제할 경우에 모든 순환을 깰 수 있는 간선들의 집합을 backedge라고 한다. T는 비순환 방향성 그래프를 충족하는 Gdag에 의하여 얻어진 트리이다. 사이트 Si부터 Sj까지 backedge가 존재한다면, backedge 집합의 최소성에 의하여 Gdag에 Sj부터 Si로 경로가 존재한다. 그러므로 T의 특성에 의하여 Sj는 T에서 Si의 조상이다. 갱신하는 트랜잭션은 T에서 자신의 조상들인 사이트들에게 backedge를 따라서 즉시 갱신을 수행한다. 조상들에 대한 즉시 갱신이 완료되면, 후손들에게 DAG(WT) 프로토콜을 따라서 지연 갱신을 수행한다.

Backedge 프로토콜은 비순환 사본 그래프에서도 직렬성을 유지하는 장점이 있으나 조상 노드들에 대한 동기화 갱신으로 인하여 병행성이 저하되는 단점이 있다.

2.2 복제 일관성의 범위

분산 환경에서 각 사이트에 저장된 사본들은 복제 일관성을 유지해야 한다. 일관성에는 제약조건에 따라 원자, 순차, 프로세서 일관성 등 다양한 수준의 일관성들이 있으므로 응용분야에 가장 적합한 일관성을 설계하는 것이 중요하다.

본 연구에서 적용하는 약한 일관성은 Dubois 등이 1986년에 정의하였다[9]. 이것의 조건은 1) 동기화 변수들에 대한 접근들이 순차 일관성을 유지하며, 2) 이전의 발신된 모든 자료에 대한 접근들이 수행되기 전에는 동기화 변수에 접근을 허용하지 않으며, 3) 동기화 변수에 대한 접근이 수행되기 전에는 자료에 대한 접근을 허용하지 않는다. 즉, 동기화 변수에 대한 접근이 펜스 역할을 수행한다.

분산 데이터베이스의 복제 일관성은 약한 일관성만으로도 직렬성을 유지할 수 있다. 본 연구에서는 일관성 수준을 약한 일관성으로 완화하기 위하여 모든 사이트가 세마포를 이용하여 트랜잭션들을 선입선출로 동기화 처리함으로써 트랜잭션들을 직렬화한다.

3. 혼합 그룹 갱신 프로토콜

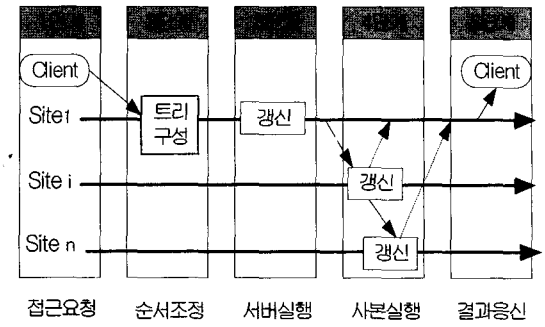
본 연구에서 제안하는 HGU 프로토콜은 단순성을 위하여 자료가 모든 사이트에 저장되어 있는 분산 데이터베이스로 가정한다.

3.1 시스템 모델

사본 그래프는 사이트들에 해당하는 정점(vertex)들의 집합에 자료들이 갱신되는 순서대로 정점들을 연결하는 간선으로 구성된 방향 그래프이다. 즉, 사이트 Si에서 Sj로 갱신을 전파하면 Si에서 Sj로 간선이 존재한다. 사본 그래프에는 순환이 있을 수 있으므로 직렬성을 보증하기 위하여 순환을 제거한 사본 트리를 만든다. 사본 트리는 같은 사본을 기억하는 모든 사이트들이 연결되며 갱신이 순차적으로 전파되는 스패닝 트리(spanning tree)이다. 스패닝 트리는 원본 갱신 시에 모든 사본들에게 순차적으로 갱신을 전파할 수 있는 비순환 그래프이다.

원본을 저장하고 관리하는 사이트를 원본 사이트라 하고, 원본이 다른 사이트로 복제된 자료를 사본이라 하며, 그 사이트를 사본 사이트라고 한다. 기동된 사이트에서 실행되는 트랜잭션을 주 트랜잭션이라고 하고, 주 트랜잭션에 의하여 다른 사이트로 보내는 메시지를 리플(ripple)이라고 하며, 리플에 의하여 기동된 트랜잭션을 부 트랜잭션이라고 한다. 각 사이트는 2PL을 적용하여 엄격한 병행제어 기능이 수행되므로 지역 트랜잭션들의 직렬성을 보증한다고 가정한다.

복제 프로토콜은 일반적으로 다섯 단계로 기술할 수 있으며 프로토콜에 따라서 다양하게 표현할 수 있다. 그림 3은 HGU의 실행 논리를 갱신 기준으로 기술한 것이다. 1단계는 클라이언트가 원본 사이트에 접근을 요청하는 것으로 분산 데이터베이스의 모든 사이트는 투명성 유지를 위하여 원본 사이트만 기억하고 있다. 2단계는 직렬성을 유지할 수 있는 실행 순서를 결정하는 단계로서



(그림 3) HGU 모델의 5단계 기능

사본 트리를 생성한다. 3단계는 원본이 갱신되는 단계이며, 4단계는 지연 갱신 전파에 의하여 사본들이 갱신되는 단계이다. 5단계는 원본 사이트가 최종 실행 결과를 클라이언트에게 응답하는 것이다. 갱신은 전역 트랜잭션에 의하여 반순서로 진행되며, 읽기-전용은 각 사이트에서 지역 트랜잭션으로 실행된다[10].

주 트랜잭션 Ti가 원본 자료 d를 읽는 것과 쓰는 것은 Ri(d), Wi(d)이다. 사이트 Sj에서 Ti의 부 트랜잭션 Tji가 자료 d를 읽는 것과 쓰는 것은 Rji(d), Wji(d)이다. sendi와 recvdi는 사이트들 간의 송신과 수신을 의미한다. 트랜잭션 Ti가 사이트 Sj에서 완료하는 액션은 Cji이며, 취소하는 액션은 Aji이다[3].

3.2 약한 일관성의 직렬성

HGU는 사본 그래프로부터 생성된 사본 스패닝 트리를 따라서 갱신을 전파한다. 이 기법에서 원본을 갱신하고 사본들에게 갱신을 전파하는 부분은 기존의 DAG(WT) 기법과 유사하다. 트랜잭션 모델은 갱신들이 저장-전진(store-and-forward) 방식으로 최종 목적지까지 전파되는 신뢰성 있는 모델이다.

3.2.1 트랜잭션들의 직렬성

원본 사이트 Si의 주 트랜잭션 Ti는 원본 di를

갱신하고 완료한 후 Si의 자식 사이트 Sj에게 갱신을 전파한다. 갱신을 전파 받은 사이트 Sj에서는 Ti의 부 트랜잭션 Tji는 사본을 갱신하고 자식 사이트에게 갱신을 전파한다. 따라서 한 사이트에서 부 트랜잭션이 실행하는 것은 그 앞의 모든 트랜잭션들이 그 사이트에서 완료되었음을 의미한다.

[정리 3.1]

원본을 갱신하는 HGU에 의하여 생성 된 스케줄은 직렬성이다.

원본 사이트 Si의 주 트랜잭션 Ti에 의하여 기동된 일련의 부 트랜잭션들은 스패닝 트리를 따라 생성되어 갱신을 전파한다. 특정 사본 사이트 Sk에서 사본을 갱신하는 트랜잭션 Tk는 조상 사이트의 트랜잭션이 기동되어 완료하였기 때문에 기동된 것이며 Tk가 완료해야 자식 사이트의 트랜잭션이 기동될 수 있다. 각 사이트는 세마포를 이용하여 트랜잭션들이 생성된 순서대로 임계 영역에 진입하게 함으로써 트랜잭션들을 반순서로 처리한다. 따라서 모든 부 트랜잭션들은 트리를 따라 순차적으로 완료되므로 직렬성이 유지된다.

3.2.2 버전 제어

사본 사이트에서 원본을 갱신하려는 트랜잭션은 사본이 원본과 동일해야 갱신할 자격이 있다. 만약 사본이 원본과 다르면 그 사본은 이미 낡은 자료이므로 오류가 발생할 수 있다. HGU는 그룹 갱신을 허용하므로 갱신 유실(lost update) 문제가 예상된다. 이를 예방하기 위해 버전번호를 관리한다. 버전 관리자가 각 투플들의 버전을 분산 방식으로 관리한다. 버전 관리자는 투플마다 긴 정수의 버전번호를 부여한다. 원본을 갱신하는 트랜잭션은 버전번호를 1씩 증가하고, 갱신 자료 값과 함께 버전번호를 자식 사이트에게 전파한다. 부 트랜잭션이 사본을 갱신할 때도 사본과 원본의 버전번호가 일치해야 갱신을 허용한다[6].

[정리 3.2]

사본을 갱신하는 HGU에 의하여 생성 된 스케줄은 직렬성이다.

사본 사이트 Sk에서 사본을 갱신하려는 주 트랜잭션 Tk는 원본 di와 사본 dk의 버전이 동일해야 갱신할 수 있다. 버전번호가 다르다면 다른 트랜잭션이 원본을 갱신한 것이므로 지연 갱신이 전파되고 있는 도중임을 의미한다. 따라서 트랜잭션 Tk를 취소하고, 현재 진행 중인 지연 갱신이 완료된 다음에 트랜잭션을 재 시작해야 한다. 사본과 원본의 버전번호가 같으면 동기적으로 즉시 갱신한다. 사본과 원본이 갱신된 다음에는 Tk를 완료하고 원본의 자식 사이트 Sj로 지연 갱신을 전파한다. 지연 갱신은 정리 3.1의 스케줄에 의하여 실행되므로 이 스케줄은 직렬성이 보증된다.

[정리 3.3]

HGU가 생성한 스케줄은 약한 일관 성을 유지한다.

자료를 갱신하는 트랜잭션은 세마포의 임계영역에 진입하여 자료와 버전번호를 갱신한다. 만약 진입하지 못하면 세마포 큐에서 대기해야 한다. 임계영역을 접근했다는 것은 그 이전에 자료를 갱신하는 모든 트랜잭션들이 갱신을 완료한 것이므로 본 문의 2.3절에서 언급한 약한 일관성의 조건 2)를 충족한다. 임계영역을 접근하는 트랜잭션들은 대기 큐에서 FIFO 순으로 대기하므로 상기 조건 1)을 충족한다. 갱신을 수행하는 트랜잭션이 세마포를 벗어나기 전에는 다른 트랜잭션이 임계영역에 진입할 수 없으므로 상기 조건 3)을 충족한다. 따라서 HGU가 생성한 스케줄은 약한 일관성을 보증한다.

예제 2의 은행 계좌 실례를 살펴보자. 사이트 S_1과 S_2에서 수표계좌 a와 저축계좌 b에서 각각

(표 1) 예제 2의 사이트별 스케줄

사이트	주 트랜잭션	부 트랜잭션
S_1	T_1 : $R_1(a),W_1(a)$ T_2 : $R_2(b),W_2(b)$	T^1_3 : $R^1_3(b),W^1_3(b)$ T^1_4 : $R^1_4(a),W^1_4(a)$
S_2	T_3 : $R_3(b),W_3(b)$ T_4 : $R_4(a),W_4(a)$	T^2_1 : $R^2_1(a),W^2_1(a)$ T^2_2 : $R^2_2(b),W^2_2(b)$

출금하려고 한다. T_1은 원본 a에서 출금하고, S_2의 사본 b에서 출금하기 위해 리플을 전송한다. 또한 T_2는 원본 b에서 출금하고, S_1의 사본 a에서 출금하기 위해 리플을 전송한다. 그러나 S_2에서는 원본 b가 이미 갱신되었으므로 버전번호가 달라서 취소되고, S_1에서도 같은 이유로 취소된다. 따라서 a, b의 원본들이 정상적으로 갱신된다. 예제 2의 실행 스케줄을 혼합 그룹 갱신 기법으로 기술하면 표 1과 같다.

각 사이트에서 실행되는 트랜잭션들의 스케줄은 다음과 같다.

S_1 : $R_1(a),W_1(a),C_1,R_2(b),R^1_4(a),A^1_4,C_2,R^1_3(b),W^1_3(b),C^1_3$
S_2 : $R_3(b),W_3(b),C_3,R_4(a),R^2_2(b),A^2_2,C_4,R^2_1(a),W^2_1(a),C^2_1$

S_1에서 T_1이 완료하고(C_1), T_2가 사본 b를 갱신하기 위하여 자료 값을 읽고($R_2(b)$) 원본 사이트 S_2에서 b를 갱신하려 한다. 그러나 원본 b를 읽은 결과($R^2_2(b)$) 이미 버전번호가 바뀌어 갱신이 불가하므로 T^2_2는 취소(A^2_2)된다. 마찬가지로 S_2에서도 T_3를 완료하고(C_3), S_1에서 원본 a를 갱신하려고 원본 a를 읽지만($R^1_4(a)$) 이미 버전번호가 바뀌어 갱신하지 못하고 취소(A^1_4)된다. 취소된 후에는 지연 갱신 기법에 의하여 S_1의 사본 b와 S_2의 사본 a가 각각 $R^1_3(b),W^1_3(b)$와 $R^2_1(a),W^2_1(a)$에 의하여 갱신된다. 이 스케줄은 원본을 갱신하는 부 트랜잭션들이 취소되기 때문에 데드록에 걸리지 않으며, 지연 갱신 전파에 의하여 사본들이 갱신되고 일관성이 보증된다.

3.2.3 자료구조와 알고리즘

분산 시스템에서는 클라이언트가 모든 사이트의 사본들을 직접 접근할 수 있으나, 분산 데이터베이스에서는 항상 하나(원본)만을 접근 허용한다. 클라이언트가 모든 사본들을 접근하게 하려면 많은 정보를 유지해야하므로 데이터베이스의 투명성이 저하되기 때문이다. 사본과 원본이 일치하는지 확인하려면 버전번호를 관리하고, 클라이언트의 자료마다 원본 사이트에 대한 간단한 정보를 기억한다. 또한 사본 스패닝 트리는 2진 트리를 전제하므로 두 사이트의 정보가 필요하다. 따라서 일관성 유지를 위한 기본적인 정보는 "자료 id, 원본 사이트, 버전번호, 자식 사이트1, 자식 사이트2, 록 상태, 록 모드" 등이다. 각 사이트는 자료를 접근하는 트랜잭션들의 처리를 순서화하기 위하여 선입선출 대기 큐를 가진 세마포를 유지한다.

HGU 알고리즘은 크게 주 트랜잭션의 원본 갱신 모듈과 즉시사본갱신 모듈 그리고 부 트랜잭션의 지연갱신 모듈과 즉시원본갱신 모듈 등으로 구성된다. 그림 4는 사본 갱신을 위하여 즉시 갱신을 실행하는 핵심 루틴이다. 즉, 그림 4(a)는 사본 사이트 Sj에서 사본을 갱신하기 위하여 주 트랜잭션 tj를 기동하고 원본 사이트의 부 트랜잭션 ti를 호출하여 동기화를 요청한다. 그림 4(b)에서 ti는 사본과 원본의 버전이 같으면 갱신을 허용하는 메시지를 사본 사이트의 tj에게 보낸다. 이 때 ti는 지연 갱신 전파를 위하여 사본 스패닝 트리를 생성하고, 자식 사이트를 tj에게 전달한다. 갱신 허용을 수신한 그림 4(a)의 tj는 사본을 갱신하고 원본 사이트의 자식 사이트 Sc에게 직접 지연 갱신을 전파한다. 이 알고리즘은 사이트들이 동일한 코드를 사용하므로 경제성이 좋고, 사본 트리에 의하여 반순서로 처리되므로 고장 투명성이 우수하다. 항상 록을 해제하고 갱신을 전파하므로 충돌 연산이 적다.

```
//실행 환경: 사이트들의 집합 S = {S1,S2,...,Sn}
  Si: original site, Sj: copy site, Sc: child site
 자료들의 집합 D= {d1,d2,...,dm }
  Gc: connected components for replica tree
//ti: 트랜잭션, dj: 자료, vj: 버전번호, r: 결과
세마포 접근
read dj
find Si  // original site
send ripple(dj,vj) to Si
  대기
receive ripple(r,Sc) from Si
  //Sc: child of Si in Gc
if r = true
then //버전 일치, 갱신 허용
  update dj,vj+1) in Sj
  세마포 해제
  commit
  send ripple(dj,vj+1) to Sc
else //버전 불일치
  세마포 해제
  abort tj
end
```

(a) 주 트랜잭션의 즉시사본갱신

```
//ti: 트랜잭션, dj: 자료, vj: 버전번호, r: 결과
if vi = vj
then //버전 일치, 갱신 허용
  construct Gc  //사본 트리 생성
  세마포 접근
   update dj,vj+1) in Sj
   세마포 해제
   commit
   send ripple(r,Sc) to Sj
 else //버전 불일치
   abort ti
   send ripple(r,Si) to Sj
 end
```

(b) 부 트랜잭션의 즉시원본갱신

(그림 4) HGU의 사본 갱신 알고리즘

3.3 성능 평가 및 분석 계획

본 연구에서 성능을 비교하려는 Backedge 프로토콜은 DAG(WT) 기법에 동기화 갱신을 추가한 혼합 지연 갱신 프로토콜이다. 따라서 즉시 갱신을 수행하거나, 지연 갱신을 전파하기 위하여 사본 트리로부터 backedge 간선 집합을 계산해야 하는 부담이 있다. 또한 사본 사이트에서 자료 갱신 시에는 모든 조상 노드들을 동기화 갱신한 후에 후손 노드들을 지연 갱신한다. 따라서 우연하게 사본 트리의 마지막 노드에서 갱신이 발생하면 거의 모든 사본들에 록을 걸고 갱신해야 하므로 갱신 시간이 길고 데드록 가능성이 높다. 또한 동기화 갱신 시 고장이 발생하면 모든 사이트의 갱신이 취소된다.

HGU 기법은 사본 사이트에서 갱신 시에 원본하고만 동기화 갱신을 수행하고, 나머지 사이트들은 지연 갱신하기 때문에 록킹 시간이 짧다. 즉 HGU는 대부분의 경우를 지연 갱신 전파하므로 각 사이트 단위로 트랜잭션이 완료하면 즉시 록을 해제하므로 데드록 가능성이 감소한다. 동기화 갱신 시에도 사본과 원본의 버전번호가 다르면 취소되기 때문에 일관성이 유지된다. HGU는 서버 고장 시에 사본 트리 순서대로 서버를 선정하며, 모든 사이트가 동일한 코드를 사용하기 때문에 고장 투명성이 좋다.

부수적으로 Backedge를 계산하는 효율적인 알고리즘은 없으나, HGU를 위한 스패닝 트리를 구하는 효율적인 알고리즘은 이미 존재한다[4]. Backedge는 선형 리스트이므로 탐색 시간이 O(n)이지만 스패닝 트리는 너비우선 탐색을 하므로 log(n)이다. BackEdge의 경우에 동기화 갱신을 위해 록을 거는 사이트의 수가 평균적으로 n/2개이므로 록을 거는 시간도 O(n)이다. 록이 걸리면 병행처리가 불가능하므로 그만큼 비효율적이다. HGU는 동기화 갱신을 원본 사이트하고만 수행하므로 록이 걸리는 시간은 O(1)이다.

이들 평가 요소들을 기반으로 모의실험을 통하여 성능을 분석한 다음에 구현을 통하여 성능을 평가할 계획이다.

4. 결 론

HGU는 복제된 사본들의 일관성을 완화하고,

그룹 갱신이 가능하며, 충돌 연산을 감소하는 분산 데이터베이스 병행제어 관리 기법이다. HGU는 갱신 트랜잭션들의 직렬성을 보증하고, 데드록을 예방하며, 사이트 고장 시에도 강건하다. HGU는 이를 위해 두 가지 핵심 개념을 제시한다. 한 가지는 지연 갱신 전파 기법을 개선하여 일관성 제약조건을 완화한다. 사본 갱신은 너비우선 탐색에 의한 스패닝 트리를 이용하여 갱신 시간을 감소하고자 한다. 또 하나는 사본 사이트에서 갱신 시에 원본과 즉시 갱신함으로써 록킹 시간을 감소하는 것이 설계의 목적이다.

지연 갱신 기법은 갱신을 완료한 후 충돌이 발생하면 그 사본을 복원시키는 것이 어려웠으나, HGU는 갱신 절차가 사본 트리와 세마포에 의하여 직렬화된다. 즉 한 사이트에서 발생하는 원본과 사본들의 갱신은 선입선출 작업 큐에 의하여 갱신과 갱신 전파가 직렬화된다. 따라서 HGU는 약한 일관성을 도입하여 병행처리 적용 분야를 확대하고, 그룹 갱신으로 사용자 편리성을 향상한다. 단 성능 개선에 관한 사항은 다음 연구에서 평가하고 분석하고자 한다. 향후 연구에서 이러한 개선 요소들을 모의실험을 통하여 확인하고, 프로토콜을 구현한다.

참 고 문 헌

[1] Todd Anderson, Yuri Breitbart, Henry F. Korth and Avishai Wool, "Replication, consistency and practicality: Are these mutually exclusive?," In Procs. of the ACM SIGMOD International Conf. on Management of Data, Seattle, WA, 1998.

[2] P. A. Bernstein, V. Hadzilacos, N. Goodman, "Concurrency Control and Recovery in Database Systems," pp. 271-276, Addison-Wesley, Reading MA, 1987.

[3] Y. Breitbart, H. Korth, "Replication andconsistency: Being lazy helps sometimes," In Procs. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Tucson, Arizona, 1997.

[4] Yuri Breitbart, Raghavan Komondoor, Rajeev Rastogi, S. Seshadri, Avi Silberschatz, "Update Propagation Algorithms for Replicated Database systems," Technical Report BL0112370-981028TM, Bell Labs,Oct. 1998.

[5] Yuri Breitbart, Raghavan Komondoor, Rajeev Rastogi, S. Seshadri, Avi Silberschatz, "Update Propagation Protocols for Replicated Databases," In Procs. of ACM SIGMOD International Conf. on Management of Data, Philadelphia, 1999.

[6] M. Colton, "Replicated Data in a Distributed Environment," In Procs. ofACM SIGMOD International Conf. on Management of Data, Washington, DC, 1993.

[7] Jim Gary, Pat Helland, Patrick O'Neil, Dennis Shasha, "The Danger of Replication and a solution," In Procs. of ACM SIGMOD International Conf. on Management of Data, Montreal, Canada, 1996.

[8] P. Chundi, D. Rosenkrantz, S. Ravi, "Deferred Updates and Data Placement in Distributed DataBases," 12th International Conference on Data Engineering, IEEE, 1996.

[9] Dabid Mosberger, "Memory Consistency models. Operating System Review," ACM. Jan. 1993.

[10] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso, "Understanding Replication in Databases and Distributed Systems," In Procs. of 20th Inter. Conference on DCS. IEEE, 2000.

◐ 저 자 소 개 ◑

이 병 욱

1973년 연세대학교 화학공학과 졸업(학사)
1984년 George Washington 대학교 대학원 전자계산학과 졸업(석사)
1994년 중앙대학교 대학원 전자계산학과 졸업(박사)
1995년~현재 : 경원대학교 컴퓨터공학과 교수
관심분야 : 데이터베이스, 분산 시스템
E-mail : leebw@mail.kyungwon.ac.kr