클라이언트 변경 트랜잭션에서 공간 데이터의 동시성 및 일관성 제어

신영상[0]*, 최진오**, 조대수*, 홍봉희*
*부산대학교 컴퓨터공학과, **경동대학교 정보통신공학부

Concurrency and Consistency Control of Spatial Data in Client-Side Update Transactions

YoungSang Shin[0]*, JinOh Choi**, DaeSoo Cho*, BongHee Hong*
*Dept. of Computer Engineering, Pusan National University,
**School of Information Technology Engineering, KyungDong University

요 약

서버의 공간 데이터가 클라이언트 캐쉬에 중복 저장되어 있는 클라이언트-서버 GIS 환경에서 동시에 수행될 수 있는 대화식 클라이언트 트랜잭션의 변경은 일관성 제어를 위해 서버와 다른 클라이언트에 전파되어야 한다. 이때 한 클라이언트의 캐쉬 변경이 다른 클라이언트의 변경과 충돌이 될 수 있는데, 클라이언트 트랜잭션은 기존의 기법으로는 동시성과 캐쉬 일관성이 제어 될 수 없다. 지도 수정 트랜잭션은 긴 트랜잭션이며 공간 관련성에 의한 종속성을 가지기 때문이다. 또한 캐쉬 변경 내용의 전파는 캐쉬 사용의 이점을 잃지 않기 위해 통신 부하 최소화 대책이 고려되어야 한다

이 논문은 클라이언트-서버 GIS 환경에서 클라이언트 수정 트랜잭션의 동시성과 캐쉬 일관성 제어를 위하여 기존의 잠금 기법을 확장하고, 통신 부하의 최소화를 고려한 새로운 변경 전파 프로토콜을 설계하고 구현한 결과를 보인다.

1. 서론

클라이언트-서버 GIS 응용에서는 서버에 있는 지도 데이터가 클라이언트로 전송되어야 하므로 사용자의 응답 시간 지연을 일으킨다. 이를 해소하기 위해서 클라이언트에 캐쉬를 사용할 수 있다. 캐쉬의 사용은 클라이언트와 서버간의 공간 데이터 전송을 줄이고 서버에서 각 클라이언트의 데이터 요구 처리에 대한 과부하를 줄일 수 있는 이점을 가지기 때문이다[2].

클라이언트에 캐쉬를 사용하는 클라이언트-서버 GIS 환경의 클라이언트에서 지도를 수정하는 트랜잭션에서는 동시성 지원과 일관성 제어를 어렵게 하는 문제점들이 있다 이는 첫째, 지도 수정 트랜잭션은 대화식 작업을 필요로 하는 긴 트랜잭션이고 둘째, 각 클라이언트의 캐쉬에는 서버에 유지되는 공간 데이터베이스가 동적으로 중복되어 있으며 셋째,, 공간 객체는 서로 공간 관련성을 가지기 때문에 서로 다른 객체일지라도 기하의 공간 관련성에 따라 변경이 종속적일 수 있다는 특징 때문이다[5]

이 논문에서는 [1]의 연구를 기반으로, 클라이언트 변경 트랜잭션에서 공간 데이터의 동시성 및 일관성 제어를 위해 전통적인 잠금 기법을 확장하고 새로운 변경 전파 프로토콜을 제시한다 이 논문에서 제안하는 변경 전파 프로토콜은 공간 관련성에 의한 변경 종속성 문제를 2단계 완료 프로토콜에 기반한 협동 작업을 통해 해결한다

이 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 기술하고 3장에서는 클라이언트 트랜잭션 들의 동시성 지원을 위하여 확장된 잠금 기법을 설명한다. 4장에서는 확장된 잠금 기법에 기반한 변경 전파 프로토콜과 예제를 설명하고, 5장에서는 구현 내용을 기술한다. 그리고 6장에서는 결론을 맺는다.

2. 관련연구

클라이언트-서버 환경에서의 협동 트랜잭션을 위해서는 전통적으로 체크아웃 모델이 사용되어 진다. 그러나 이 모델은 공간 데이터의 특성인 **분산 공간관련성**[1]에 의해 체크인 시 충돌이 발생할 수 있으므로 체크아웃 모델은 그대로 적용되어 질 수 없다.

이 논문은 [1]의 영역 잠금에 대한 연구에 기반하고 있다. [1]은 분산환경을 대상으로 연구되었기 때문에 데이터가 각 사이트의 데이타베이스에 정적 중복되어 있고 변경 시 모든 사이트로 변경 내용이 전파되어야 한다. 이 논문에서 대상으로 하는 클라이언트-서버 환경은 불필요한 메시지 과부하의 최소화와 동적 중복 데이터의 캐쉬 일관성 제어가 주된 고려 대상이다. 이 논문은 [1]의 동시성 제어 기법과 정적 중복 일관성 제어를 위한 변경 전파 프로토콜을 확장하여, 통신 과부하를 최소화한 동적 캐쉬 중복 일관성 제어 기법을 제안한다.

3. 잠금 모드

이 장에서는 클라이언트 변경 트랜잭션 들의 동시성 지원을 위해 확장된 잠금 기법[5]을 설명한다.

3.1. CS 잠금

CS (Cached Shared) 잠금은 클라이언트의 캐쉬 매니저가 서버에게 공간 객체들을 요구할 때 서버에 설정되고, 클라이언트 트랜잭션이 끝나거나 CS 잠금이 설정된 공간 객체가 클라이언트 캐쉬에서 교체될 때 해제되는 잠금이다. 이를 통하여 서버는 각 클라이언트의 동적 중복 정보 즉 어느 공간 객체들을 캐쉬에 중복 저장하고 있는지에 대한 정보를 유지할 수 있다

3.2. COD 잠금

COD (Cached Out-of-Date) 잠금 모드는 각 클라이언트 캐쉬의 유효성을 탐지하기 위한 잠금 모드이다. 특징 클라이언트가 객체를 수정하여 서버에 commit하였을 때 다른 클라이언트 캐쉬에 중복 저장된 그 객체는 무효화 되어야 한다 이 논문은 이러한 캐쉬 무효화를 COD 잠금으로 구현한다. 즉, CS 잠금 후 중복 저장되었던 객체

는 다른 클라이언트 트랜잭션의 수정에 의해 COD 잠금 모드로 변경되어 진다.

3.3. CR 잠금

CR (Cached Region) 잠금은 사용자가 수정하고자 하는 특정 영역내의 객체들에 대하여 읽기 잠금을 미리 설정하고 그 중 일부 객체들에 대한 CX 잠금 요청을 의도하는 잠금[1]이다. 이 CR 잠금은 multiple granularity 잠금의 SIX 잠금과 유사하지만, CR 잠금은 SIX 잠금과는 달리 잠금한 객체들에 대하여 다른 트랜잭션들의 CS 잠금을 허용한다. 왜냐하면 한 클라이언트의 지도 수정을 위해 SIX 잠금을 걸 경우 다른 클라이언트들이 오랫동안 공간 데이터를 참조할 수 없기 때문이다. 이때 발생할 수 있는 dirty read 문제는 협동작업에 의한 변경전파 프로토콜로 변경 내용이 통보됨으로 해결될 수 있다.

3.4. CX 잠금

CX (Cached eXclusive) 잠금은 한 공간 객체에 대하여 수정 하고자 할 때, 그 객체 또는 그 객체와 **분산 공간 관련성**[1]에 의한 종속 관계에 있는 객체들의 수정을 허용하지 않는 잠금이다. CX 잠금을 도입하는 이유는 공간 데이터는 **분산 공간 관련성에 의한 종속성**[1]으로 인하여 잠금한 객체만의 잠금 호환성을 따져 동시성을 제어할 수 없으므로 분산 공간 관련성에 기반한 쓰기 잠금이 필요하기 때문이다. 기존의 쓰기 잠금 기법은 분산 공간 관련성을 고려하지 않고 있기 때문에 공간 데이터 수정은 전체 data set에 대한 잠금을 필요로 하였다.

3.5. 잠금 모드간의 호환성

지금까지 제시된 잠금 모드간의 호환성은 표1과 같다.

표1 잠금 호환성 표

	CS	COD	WRITE	CR	CX
CS	yes	yes	no	yes	yes
COD	yes	yes	no	yes	N/A
WRITE	no	no	no	no	no
CR	yes	yes	no	yes	yes
CX	yes	yes	no	yes	no

클라이언트들의 캐쉬에 중복되어 있는 공간 데이터의 수정을 위해 표1에 적용한 호환성 규칙은 다음과 같다

한 객체에 대하여 WRITE 잠금이 설정되어 있지 않다면 CR 잠금을 허용한다. 이것은, 공간 데이터는 화면으로 출력되므로 한 클라이언트에서의 지도 검색이 다른 모든 클라이언트의 지도 수정을 금지시킬 수 있으므로 WRITE 잠금이 아닌 CR 잠금과 CX 잠금을 이용하여 지도를 수정하기 위함이다.

CX 잠금 후 수정 중인 객체를 CS 잠금 또는 CR 잠금을 걸고 읽는 것을 허용하는 것은 데이터의 부정확성 문제를 초래하지만, 이 논문은 동시성 지원을 위해 이를 허용한다. 그렇지만, 지도의 수정은 CX 잠금 권한을 얻어야 하며 CX 잠금은 서로 호환되지 않으므로 부정확한 수정은 발생하지 않는다 그리고, CS 잠금에 의한 부정확한 읽기는 COD 잠금으로 탐지되며, CR 잠금에 의한 부정확한 읽기는 협동 작업에 의한 변경 전파 프로토콜로 변경 내용이 통보된다 같은 이유로 CR 잠금과 CX 잠금이 설정되어 있더라도 다른 클라이언트에서 지도를 캐쉬할 수 있도록 허용한다

3.6. CR 잠금과 CX 잠금에 의한 동시성 제어 기법

한 개 이상의 클라이언트들이 동시 수정 시, 각 클라이언트들의 CR 잠금 영역들 중 상호 중첩 영역이 존재할 수 있는 데 이것을 NDJ(Non Disjoint area)라 정의한다

CR 잠금에 의한 동시성 제어에서 NDJ가 존재하지 않는 경우에는 두 트랜잭션이 분산 공간 관련성에 의한 종속성을 지닌 객체들을 동시 수정할 가능성이 없으므로 서로 병렬 수행될 수 있다. 그러나, NDJ가 존재하는 경우에는 분산 공간 관련성에 의한 종속성으로 인해 충돌이 발생할 수 있으므로 병렬 수행을 허용할 수 없다. 이 때에는 두 트랜잭션이 협동작업으로 수정을 진행하여야 한다.

지도 수정 트랜잭션은 사용자와의 상호작용에 의해 진행되며 수정 결과의 공간 관련성에 의한 종속성이 고려된 정확성을 사용자의 판단에 근거한다. 그러므로 CR 잠금을 통하여 공간 관련성에 의해 종속적이지 않은 객체들의 수정은 병렬 수행할 수 있도록 보장된다

CX 잠금을 이용한 동시성 제어에서는 다음과 같이 지도 수정 작업 시 병렬성을 높일 수 있다 첫째, 서로 다른 두 트랜잭션 CR 잠금이 NDJ를 가지더라도 현재 수정중인 객체들이 상호 분산 공간 관련성에 의한 종속성이 없을 경우 동시 수정을 허용하여 병렬성을 높인다. 둘째, 분산 공간 관련성에 의한 종속성이 있을 경우에는 하나의 수정단위가 변경되는 동안만 wait하게 하여 병렬성을 높일 수 있다.

4. 변경전파 프로토콜

이 장에서는 새로 제시한 잠금에 기반하여 변경 전파 및 탐지를 통해 캐쉬 일관성을 제어하는 프로토콜을 제시한다.

4.1. mid-commit

이 논문에서는 버전병합 기법에서의 변경 후 충돌로 인한 rollback 문제를 해결하기 위해 점진적 변경 전파 기법을 사용한다 [5]. 점진적 변경 전파 기법을 사용하기 위해서 mid-commit[1] 개념을 도입한다.

mid-commit은 점진적 변경 전파 기법으로 캐쉬 트랜잭션 들의 동시성을 높이고 분산 공간 관련성으로 인해 발생하는 지도 수정 결과의 오류를 없애기 위해 협동작업을 수행한다. 캐쉬 트랜잭션 들의 협동 작업은 기존의 2단계 완료 프로토콜(이하 2PC)을 확장한 공간 관련성 기반 2PC(이하 SR기반 2PC)[1]로 지원한다.

4.2. 협동작업에 참여하지 않았던 클라이언트 캐쉬의 일관성 유지

mid-commit은 수정 작업을 하는 클라이언트와 NDJ 클라이언트들만이 참여 한다. 이는 [1]에서의 분산환경과 같이 모든 클라이언트들에게 mid-commit을 전파할 경우에는 메시지 부하가 높아지기 때문이다. 그러므로 mid-commit에 참여하지 않았지만 수정된 객체를 캐쉬에 가진 클라이언트들의 일관성 유지를 위한 알고리즘이 필요하다.

이 논문에서는 이러한 클라이언트들의 일관성 유지를 위해서 탐지기반 기법(detection-based)을 사용한다 즉 클라이언트가 서버에게 자신이 가진 데이터가 유효한지를 탐지하는 기법에 기반하여 일관성을 제어 하는 것이다 이는 전체 프로토콜의 복잡도를 줄여 준다[4].

또한 탐지 시기를 늦추는 낙관적 접근법을 사용한다. 이는 메시지 부하를 최소화하여 캐쉬의 이점을 최대한 이용하기 위함이다[4]

4.3. 프로토콜 예제

이 절에서는 프로토콜 예제를 소개한다. 그림1은 도로 확장 공사에 의한 지도 수정 과정을 보인 그림이다. 클라이언트 1의 트랜잭션 T_A는 도로1 객체를 수정하고 클라이언트 2의 트랜잭션 T_B는 필지1 객체를 수정하며 T_A의 수정이 T_B의 수정보다 앞섬을 가정한다

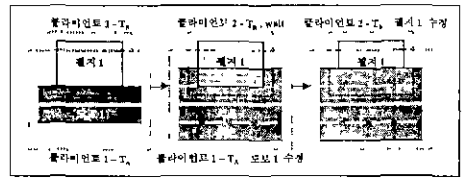

그림1 도로 확장 공사에 의한 지도 수정

트랜잭션 연산을 이용하여 그림1에 대한 변경 전파 프로토콜을 보인 것이 그림2이다. 그림2에서 T_B의 수정 객체 '필지1'은 T_A의 수정객체 '도로1'과 분산 공간 관련성에 의한 종속이므로 CX 잠금의 요청이 허용되지 않는다(①). 그러므로 T_B는 wait하였다가 T_A의 '도로1'에 대한 수정작업이 완료된 후에 변경을 시작할 수 있다. T_A의 '도로1'에 대한 수정 결과는 협동작업을 하는 클라이언트2로 mid-commit에 의해 전파되고 T_B와 SR기반 2PC를 수행한다(②). T_B가

mid-accept로 응답했을 경우 서버는 global-mid-commit을 전파하여 '도로1'에 대한 수정을 완료한다 그러고 T_B는 수정작업이 완료된 후 mid-commit을 서버에 전파하고 서버는 T_A와 SR기반 2PC를 수행하여 T_A의 확인을 받는다(③) ④에서 T_A는 수정 작업을 끝냈지만 T_B와 SR기반 2PC를 수행하여 지도 수정의 정확성을 상호 검증하여야 하므로 wait하게 된다.

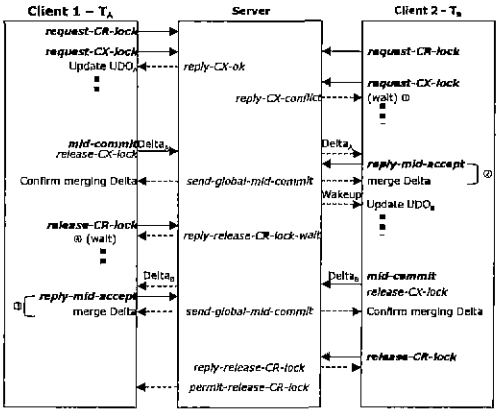

그림2 변경 전파 프로토콜의 예

5. 구현

그림3은 이 논문의 실험 구현 환경과 시스템 구조이다. 서버는 객체지향 GIS 도구인 Gothic 3.0을 대상으로 구현하였으며, 클라이언트는 Win98/NT환경에서 구현하였다. 'Data Server'는 Gothic 3.0의 객체지향 DBMS이다.

이 논문에서 Gothic 3.0의 기존 잠금 기능과 별도의 잠금 관리기(Lock Manager)를 구현하였다 즉 실제 서버의 data set에 WRITE lock을 설정하고 관리하는 것은 Gothic 3.0이며 변경 전파 프로토콜에 의한 확장된 잠금을 설정을 관리하는 것은 별도로 구현한 잠금 관리기이다

또한 Gothic 3.0의 트랜잭션 관리기를 이용하는 별도의 트랜잭션 관리기(Transaction Manager)를 구현하였다. 이는 Gothic 3.0의 트랜잭션 관리기는 서버 내의 지역 트랜잭션 관리 기능만이 있기 때문이다. 이 논문에서는 각 클라이언트들의 수정 트랜잭션 들을 전역적으로 관리하는 트랜잭션 관리기를 구현하였다

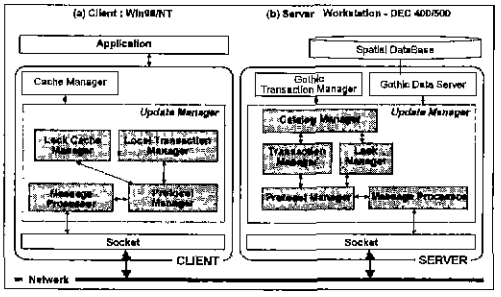

그림3 구현 환경 및 시스템 구조

그림4는 구현된 Update Manager를 이용하여 두 클라이언트에서 두개의 필지 객체를 각각 동시에 수정하는 예를 보이고 있다 현재 클라이언트 화면에 출력된 dataset은 창원 UIS에서 사용되는 실제 데이터이다. 또한 지도 화면 위의 사각형은 사용자에 의해 설정된 CR 잠금의 영역이다 클라이언트 1은 왼쪽 필지를, 그리고 클라이언트 2는 오른쪽 필지를 수정한다 그림1 (b)에서 클라이언트 1은 왼쪽 필지를 수정한 후 DELTA를 서버로 보내면 서버는 클라이언트 2에 DELTA를 전파한다. 클라이언트 2가 클라이언트 1의 수정 결과를 받아들이고, 그림4 (c)에서 자신의 수정 결과를 서버로 보내어서 클라이언트 1로 전파한다. 이러한 과정의 제어는 CR 잠금, CX 잠금, SR기반 2PC에 의해 제어된다.

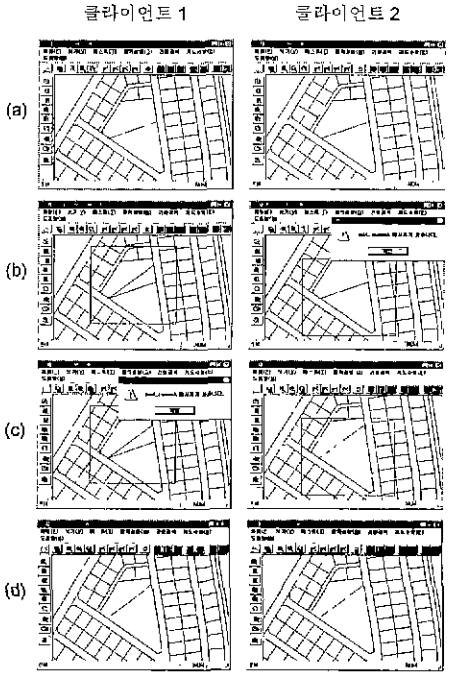

그림4 구현된 **Update Manager**를 이용한 지도 수정 예

6. 결론 및 향후 연구

이 논문에서는 서버의 공간 데이터가 클라이언트 캐쉬에 동적으로 중복 저장 되어있는 클라이언트-서버 GIS 환경에서 지도 수정 트랜잭션 들의 동시성과 캐쉬 일관성 제어를 지원하기 위한 기법들을 설계하고 구현하였다.

이 논문의 주된 내용은, 캐쉬를 사용하는 클라이언트-서버 GIS 환경에서 기존의 제어 기법으로 지원할 수 없는 클라이언트 트랜잭션 들의 동시성과 캐쉬 일관성 제어를 위해 CR 잠금 및 CX 잠금과 CS 잠금 및 COD 잠금 그리고 SR기반 2PC를 이용하는 프로토콜을 설계 및 구현한 것이다.

향후 연구로는 메시지 부하 정도의 성능 평가를 통하여 프로토콜 최적화를 하며, 프로토콜에 예외처리 기능들을 추가 확장하는 것이다.

참고문헌

[1] JinOh Choi, YoungSang Shin, BongHee Hong "Update Propagation of Replicated Data in Distributed Spatial Databases" International Conference on DEXA 99, 1999

[2] Michael J Franklin 외, "Local Disk Caching for Client-Server Database Systems" Proc. of 24th Int Conf on VLDB, pp641-654, 1993

[3] Kevin Wilkinson, Marie-Anne Neimat "Maintaining Consistency of Client-Cached Data", in VLDB 1990

[4] Michael J Franklin, 외 "Transactional Client-Server Cache Consistency : Alternative and Performance", in ACM TODS, Vol 22, No 3, 1997

[5] 신영상, 최진오, 홍봉희, "클라이언트-서버 환경에서 캐쉬 공간 데이터의 변경 전파", 한국정보과학회 '99 봄학술논문집 pp86-88