모바일 클라이언트에 캐쉬된 공간 데이타의 일관성 제어 기법

(A Consistency Control Method for Spatial Data Cached in Mobile Clients)

안 경 환 † 차 지 태 †† 홍 봉 희 †††
(Kyounghwan An) (Jietae Cha) (Bonghee Hong)

요 약 모바일 클라이언트-서버 환경에서 모바일 클라이언트는 무선 통신의 높은 비용으로 인하여 서버와 단절된 상태를 가지며, 캐쉬된 지도의 효율적인 갱신을 위해 지역 복사본을 클라이언트가 갖는다. 이러한 환경에서 서버 데이타베이스의 일부가 변경된 경우 일관성 제어시에 캐쉬가 무효화될 수 있고, 이로 인해 캐쉬된 전체 데이타가 재전송되는 것은 비효율적이다. 그러므로 서버에서 변경된 객체 정보를 로그로 기록하여, 캐쉬된 영역에 속하는 최근 변경 정보만을 전파하는 방법이 필요하다. 그러나 순차적으로 저장되는 로그의 경우 다음과 같은 문제점을 가진다. 첫째, 로그가 계속해서 누적된다. 이는 서버의 저장 공간 문제와 로그의 검색 성능 저하를 가져온다. 둘째, 순차적인 검색을 해야 한다. 로그의 양이 많을 경우 순차 검색은 비효율적이다. 셋째, 해당 클라이언트에게 불필요한 로그도 함께 전파된다. 이는 변경 정보 전파시 높은 무선 통신 비용을 발생시킨다. 이 논문에서는 이러한 문제점들을 해결하기 위해서 불필요한 로그들을 정의하고, 이들을 생성 및 검색 단계에서 제거, 여과하기 위한 기법들을 제시한다. 또한 순차 저장된 로그에 대한 검색 시간을 단축시키기 위한 구조와 기법을 제시한다.

키워드 : 모바일 데이타베이스, 공간 데이타베이스, 일관성 제어, 로그 데이타

Abstract In mobile client-server environments, mobile clients usually are disconnected with their server because of high cost of wireless communication and keep their own local copies to provide efficient updating the cached map. The update of the server database leads to invalidation of the cached map in the client side. To solve the issues of invalidation of the cached map, it is not efficient to resend part of the updated server database to clients whenever the updating of the server database occurs. This paper proposes a log-based update propagation method to propagate the server's update into its relevant clients by using only update logs. Too many logs increasingly accumulate as the server database is updated several times. The sequential search of the relevant log data for a specific client is time-consuming. Sending of unnecessary logs should be avoided for reducing the overhead of communication. To solve these problems, we first define unnecessary logs and then suggest log reduction methods to avoid or cancel creating unnecessary logs. The update log index is used for quickly retrieving relevant logs.

Key words : mobile database, spatial database, GIS, consistency control, log data

1. 서 론

최근 무선 통신, 무선 단말기(이동 전화, PDA 등), GPS의 발달로 이동 중에 데이타를 검색하거나 변경할 수 있는 모바일 필드 시스템에 대한 요구가 증가하고 있다[1]. 이러한 시스템의 예로는 물류 택배 서비스, 견인 및 차량등을 추적할 수 있는 모바일 행정 업무, 원격지 시설물 관리 서비스, 원격 검침 시스템 등이 있으며, 이들 응용의 주된 처리 요건은 사용자가 항상 최신의 데이타를 조회할 수 있도록 최소한의 무선 통신 비용으로 서버 데이타와의 일관성을 유지하는 것이다. 사용자들은 이러한 시스템을 이용하여 현장에서 바로 데이타

· 이 연구는 부산대학교 컴퓨터 및 정보통신 연구소에서 지원받았음
† 학생회원 : 부산대학교 컴퓨터공학과
khan@pusan.ac.kr
†† 비 회 원 : 부산대학교 컴퓨터공학과
jtcha@pusan.ac.kr
††† 종신회원 : 부산대학교 컴퓨터공학과 교수
bhhong@pusan.ac.kr
논문접수 : 2003년 2월 25일
심사완료 : 2004년 3월 4일

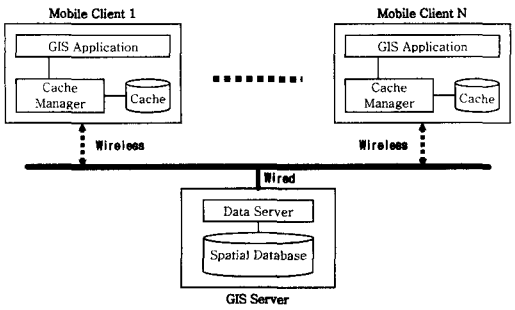


그림 1 캐쉬를 사용하는 모바일 클라이언트-서버 환경

를 확인하거나 필요에 따라서 입력을 할 수 있다.

모바일 필드 시스템은 일반적으로 그림 1과 같은 시스템 구성을 갖는다. 그림 1에서 모바일 클라이언트(Mobile Client)는 낮은 대역폭과 저속의 통신 속도를 가지게 되므로 중앙의 GIS 서버로부터 데이타의 일부분을 다운로드 받은 뒤 서버와 단절된 상태에서 단말기에 저장된 공간 데이타(캐쉬)에 대해 검색 및 변경 작업을 수행하게 된다. 그런 뒤 무선 통신 또는 유선 통신을 이용하여 다시 중앙 GIS 서버에 재접속하여 데이타에 대한 일관성 제어를 수행하게 된다.

이와 같은 모바일 클라이언트/서버 환경은 유선 환경에서의 클라이언트/서버 환경과 다른 다음과 같은 특징을 갖는다[2-5]. 첫째, 무선망을 통해 서버와 통신할 수 있다. 현장에서 작업할 때 필요에 따라 무선으로 서버와 통신을 할 수 있는데 이 때 유선망에 비해 대역폭이 작으므로 같은 양의 데이타에 대한 전송 비용이 증가하게 된다. 둘째, 서버와 단절 상태를 가지게 되므로, 서버의 적절한 제어를 받지 못한다. 유선망과는 달리 무선망에서는 서버와 통신을 할 때 높은 비용이 발생하므로 자발적으로 서버와 통신을 단절한다. 셋째, 무선 단말기는 제한적인 자원을 가진다. CPU와 메모리 용량, 전원의 경우 PC에 비해 상대적으로 제약을 가지므로 이를 고려한 시스템 설계가 필요하다. 특히 전원의 경우, 서버와 통신을 자발적 혹은 비자발적으로 단절시키는 원인을 제공한다.

위와 같은 환경에서 일관성 제어시 문제점은 모바일 클라이언트가 서버와 긴 단절 상태에서 서버 데이타베이스의 일부가 변경되었을 경우 적절한 일관성 제어를 받지 못하므로 캐쉬가 무효화될 가능성이 커지게 된다는 것이다[5]. 만약 캐쉬가 무효화되어 캐쉬된 전체 내용을 재 전송해야 한다면 공간 데이타와 같이 캐쉬의 크기가 큰 경우 무선을 통한 일관성 제어시 통신 비용이 크게 증가하는 문제가 발생한다. 이러한 문제를 해결하기 위해 서버는 변경 정보를 로그로 기록하여 일관성 제어시에 해당 모바일 클라이언트에게 필요한 최신 변경 정보만을 전송하는 것이 필요하다.

그런데 순차적으로 저장되는 로그를 이용할 경우 다음과 같은 문제점이 발생한다. 첫째, 시간이 지남에 따라 로그가 계속 누적된다. 만약 로그가 삭제되지 않고 계속 누적된다면 서버의 저장 공간이 부족하게 될 것이고, 검색 대상 로그의 양이 늘어남에 따라 검색 성능이 저하될 수 있다. 둘째, 로그를 순차적으로 검색해야 한다. 만약 누적된 로그의 양이 많을 경우 효율적인 검색을 위해 순차적인 검색이 아닌 다른 접근 방법이 필요하다. 셋째, 해당 모바일 클라이언트에게 불필요한 로그가 전달될 수 있다. 모바일 클라이언트의 경우 서버의 전체 데이타셋에 대한 중복이 아니라 일부분만을 중복하고 있으므로, 불필요한 변경 정보는 전달하지 않는 것이 바람직하다.

이 논문에서는 이러한 문제점들을 해결하기 위해서 데이타베이스의 모든 연산에 대해 로그를 기록하는 대신 commit된 논리적 연산에 대한 로그들을 변경 로그라는 개념으로 정의하고 이를 생성, 관리하는 기법을 제시한다. 변경 로그 관리 기법에서는 저장, 검색되는 양을 줄이고, 최소한의 정보만을 모바일 클라이언트로 전달하기 위해서 불필요한 로그를 4가지로 분류하고, 이를 제거하기 위한 방법을 제시하고 있다. 불필요한 로그의 첫번째 예로는 오래된 로그로 현재 서버의 데이타를 중복하여 캐쉬하고 있는 모바일 클라이언트들이 이미 일관성 유지에 사용한 로그들을 의미하고, 두번째는 무관심 로그로 모바일 클라이언트들의 캐쉬 영역을 벗어난 영역에서 일어나는 연산에 대한 로그들을 의미한다. 세번째는 반복된 로그로 동일 객체에 대해서 여러 번 연산이 일어날 때 모든 로그들을 남길 필요가 없는 경우를 나타내고 마지막으로 중첩된 로그는 해당 모바일 클라이언트에게 불필요한 변경 로그가 검색되어진 경우를 의미한다. 이러한 4가지 변경 로그들은 생성 및 검색 단계에서 제거할 수 있는데 이 논문에서는 로그 회피, 로그 취소, 로그 여과, 로그 정화라는 기법을 제시하고 있다. 또한 빠른 검색을 위하여 변경 로그에 대한 인덱싱 기법에 관해서도 논의하고 있다.

이 논문에서 제시하는 기법을 사용할 경우 다음과 같은 장점을 얻을 수 있다. 첫째, 저장되는 공간을 줄일 수 있다. 변경 정보의 경우 시간이 지남에 따라 계속해서 서버에 누적되게 되는데, 이 논문에서 제시하는 관리 기법을 이용하면 불필요한 로그들은 생성되지 않으므로 저장공간을 절약할 수 있다. 두번째, 무선 통신 비용을 줄일 수 있다. 최소한의 변경 정보만을 모바일 클라이언트로 전달함으로써 무선 통신 비용을 줄일 수 있다. 셋째, 일관성 유지 시간이 짧다. 서버에서 변경 로그를 검색하는 시간과 무선을 통한 데이타 전송 시간, 캐쉬에

변경 정보를 REDO하는 시간을 최소화할 수 있다.

논문의 구성은 2장에서는 모바일 환경에서 일관성 제어 기법에 대한 관련 연구에 대해서 알아보고, 3장에서는 이 논문에서 다루고자 하는 모바일 일관성 제어 개요에 대해서 기술한다. 4장에서는 일관성 제어를 위해 변경 로그를 생성하고, 관리하는 기법에 대해서 설명하고 5장에서는 이 논문에서 제시한 기법에 대해서 실제로 구현, 성능 평가를 해보고, 6장에서 결론을 맺는다.

2. 관련 연구

모바일 환경에서 클라이언트 캐쉬의 일관성 유지를 위한 방법은 서버에서 유지하는 클라이언트 정보의 유무에 따라 비상태 유지 서버와 상태 유지 서버로 나눌 수 있다. 기존의 연구에서는 주로 불특정 다수의 모바일 클라이언트를 대상으로 하기 때문에 비상태 유지 서버 기반으로 활발한 연구가 이루어져 왔다[6-9]. 비상태 유지 서버(Stateless-Based Server)란, 서버에 모바일 클라이언트 캐쉬에 대한 정보가 없음을 의미하며, 모바일 클라이언트의 일관성 제어를 위해 서버는 주로 변경 정보를 주기적 혹은 비주기적으로 브로드캐스팅(Broadcasting)한다. 주기적 브로드캐스팅 방식은 서버에서 일정한 주기로 변경 리포트(Update Report, UR)를 보내고, 비주기적인 방법은 서버에서 변경 정보가 발생할 때마다 보내는 방식이다. 주기적 방법의 경우 변경 정보가 발생하여도 곧바로 브로드캐스팅되지 않기 때문에 변경 정보 전달까지의 지연이 발생할 수 있다.

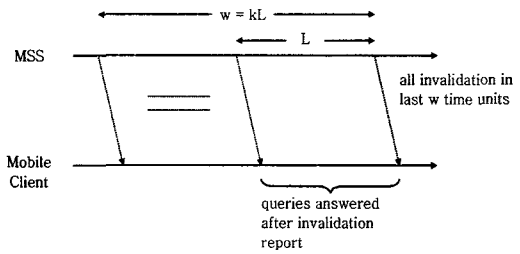


그림 2 주기적 브로드캐스팅 방식[7]

브로드캐스팅은 모바일 클라이언트가 일정 시간 동안 서버의 변경 리포트를 수신하지 못할 경우, 캐쉬 내용 전체를 무효화시켜야 하는 문제가 있다[5]. 그림 2에서 서버는 변경 리포트를 L시간마다 브로드캐스팅하며, 변경 리포트는 w = kL 시간 동안의 변경 정보를 담고 있다. 이 때, 모바일 클라이언트가 단절 상태에서 k개의 연속적인 리포트를 수신하지 못할 경우, 모바일 클라이언트의 캐쉬는 모두 버려지게 된다. 모바일 GIS 응용의 경우 자발 또는 비자발적인 단절로 인하여 서버의 변경 리포트를 수신할 수 없는 경우가 빈번하므로 앞의 방법은 부적절하다고 할 수 있다. 이에 반해 상태 유지 기반에서는 서버가 모바일 클라이언트의 캐쉬에 대한 정보를 가지고 있으므로 필요한 모바일 클라이언트에게만 변경 정보를 전파할 수 있다.

상태 유지 서버를 구성하는 방법에 관한 관련 연구로는 [5]가 있으며, 그림 3은 상태 유지 서버 구성의 예를 보여 주고 있다. Data Server는 유선망을 통해 변경 정보를 Home MSS에 전달한다. Home MSS는 HLC에 각 Mobile Host(MH)별로 캐쉬의 변경 정보 목록을 담고 있다가 MH가 일관성 제어를 요청할 때 해당 HLC를 이용하여 변경 정보를 신속하게 반영할 수 있다. 그러나 MH별로 변경 정보 리스트를 따로 유지하게 되므로, 변경된 정보가 중복되어 저장되는 문제가 발생한다. 또한, 공간 데이타와 같이 모바일 클라이언트에 저장된 캐쉬의 내용이 크고 중복된 정보가 많은 경우, HLC를 유지하는 서버의 비용이 커지는 문제점이 발생하게 된다.

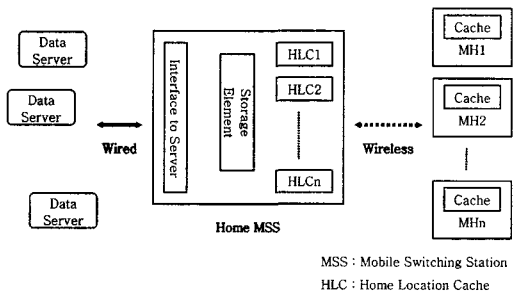


그림 3 상태 유지 서버 시스템의 구성[5]

이 논문에서는 위에서 설명한 비상태 유지 서버와 상태 유지 서버의 방법을 혼합한 새로운 방식을 제시한다. 제시할 기법에서는 비상태 유지 서버와 같이 변경 정보를 각 클라이언트별로 중복하여 저장하지 않고, 통합하여 관리함으로써 저장되는 변경 정보의 양을 줄이고, 상태 유지 서버와 같이 각 모바일 클라이언트가 필요로 하는 정보만을 전달하기 위해서 모바일 클라이언트에 관한 메타 정보만을 저장하고, 일관성 유지 요청시 검색을 통하여 변경정보를 전달한다.

3. 모바일 일관성 제어 개요

이 논문에서 대상으로 하는 모바일 GIS 환경은 그림 4와 같다. 그림에서 GIS 서버는 전체 공간 데이타를 가지고 있으며, 무선 단말기내의 모바일 클라이언트(MC_i)는 자신이 필요로 하는 데이타만을 중복하여 로컬 캐쉬에 저장하고 있다. 사용자는 서버와 단절된 상태에서 캐쉬를 검색하여 결과를 얻고, 일관성 유지가 필요할 경우

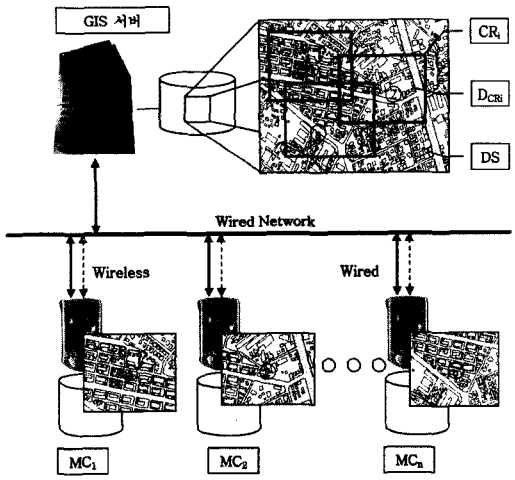


그림 4 모바일 GIS 환경

에만 유/무선을 통해 GIS 서버에 연결한다. 이 때 전체적인 일관성 제어 흐름은 다음과 같다([10]의 알고리즘 일부 수정).

알고리즘 1. *Mobile Consistency Control*

Step1 (Register) : 모바일 클라이언트(MC_i)가 관심영역(CR_i)을 서버에 등록

Step2 (Replication) : 서버로부터 전체데이타셋(DS)의 subset(D_{CRi})을 중복하여 캐쉬

Step3 (Disconnect and Consistency Control) : 단절 후 사용 및 일관성 제어 요청

Step3-1 (Use) : 공간 데이타 검색 및 변경 작업

Step3-2 (Consistency Control - Upload Phase)

Step3-2-1 : 모바일 클라이언트의 변경 사항 검색

Step3-2-2 : 변경 사항 서버 전송

Step3-2-3 : 서버에 변경 충돌 검사

Step3-2-4 : 서버에 결과 반영

Step3-3 (Consistency Control - Download Phase)

Step3-3-1 : 서버의 변경 사항 검색

Step3-3-2 : 변경 사항을 모바일 클라이언트로 전송

Step3-3-3 : 모바일 클라이언트의 캐쉬에 변경 사항 반영

Step3-4 (Repeat) : 사용자가 원할 때까지 Step 3 반복

Step4 (Unregister) : 모바일 클라이언트가 서버에 등록 해제

이 논문에서는 상태 유지 서버처럼 모바일 클라이언트에 대한 정보를 유지하기 때문에 Step1 및 Step4처럼 모바일 클라이언트(Mobile Client - 이하 MC_i라 지칭)의 캐쉬 영역에 대한 정보를 서버에 등록/해제하는 단계가 필요하게 된다. 만약 MC_i에 중복, 캐쉬하려는 영역(Cached Region - 이하 CR_i라 지칭)이 바뀔 경우 등록을 해제한 다음 바뀐 영역을 재등록하면 된다. Step2의 경우 CR_i에 속한 데이타(전체 데이타셋 중 CR_i와 Overlap되는 객체들의 집합 - 이하 DCR_i라 지칭)를 로컬 캐쉬로 내려 받는 과정인데 공간 데이타의 양이 클 수 있으므로 유선 네트워크를 이용한다고 가정한다(무선으로도 가능). Step2 이 후에 사용자는 서버와 연결을 단절한 뒤 이동하면서 지도를 검색할 수 있다. 또한 이동 중에 데이타에 대한 변경 작업을 수행할 수 있으며, 일정 시간 뒤에 일관성 유지를 서버에 요청할 수 있다. Step3의 경우 일관성 요청의 단계로 MC_i의 변경사항을 서버로 전송하여 반영하는 단계와 서버의 변경 사항을 MC_i로 전송하여 반영하는 단계로 나뉘어진다. 이 때 변경 사항의 전송은 유/무선 네트워크 모두 이용할 수 있으나, 모바일 GIS의 특징인 이동성을 고려하여 주로 무선으로 일어난다고 가정한다. 그리고 이 논문에서는 논의 범위를 한정하기 위해 다음 사항을 가정한다.

- 공간 데이타에 대한 변경 작업만 다룬다.
- 데이타의 일관성 및 정확성은 기존의 낙관적 일관성 제어 기법을 이용하여 보장한다.
- 모바일 클라이언트 및 GIS 서버의 고장 및 네트워크 장애는 없다.
- 하나의 모바일 클라이언트(MC_i)는 하나의 캐쉬 영역(CR_i)을 가진다.

즉 이 논문에서는 일관성 유지 기법 그 자체보다는 서버에서 변경된 내용을 어떻게 관리, 검색하여 MC_i로 전파할 것인가(Step3-3)에 관한 변경 전파 방법 문제를 다루고 있다.

4. 변경 로그 관리 기법

이 논문에서는 변경 내용을 로그로 기록하고 이를 이용하여 GIS 서버와 MC_i간에 일관성을 유지하는 로그-전달(log-transfer)기법을 사용하고 있다. 이 장에서는 모바일 환경의 통신 비용을 줄이기 위해 서버에서 생성 및 전송되는 로그의 양을 줄이고 로그 인덱스를 만들어 효율적으로 일관성을 제어할 수 있는 방법에 대해서 설명한다.

4.1 변경 로그

이 논문에서 변경 전파에 사용되는 로그는 기존의 회복 기법(Recovery)에 사용되는 로그와는 차이가 있다. 회복 기법에 사용되는 로그의 경우 트랜잭션 수행 중에 발생하는 모든 연산에 대해서 안정 저장소(stable storage)에 기록되고, 서버 시스템의 장애 발생 후 회복 시에 UNDO, REDO 연산에 사용된다. 이에 반해 변경 로그의 경우 성공적으로 COMMIT된 트랜잭션에 대한 연산들만 기록되고, 모바일 클라이언트에서 REDO 연산을 하기 위한 목적으로만 사용된다. 아래 정의 1은 변경 로그에 대한 정의를 보여주고 있다.

정의 1. 변경 로그(Update Log)

트랜잭션 T에 의해서 수행되는 연산들 SO = {O_1,

O_2, …, O_n}이 있고, 각각의 연산에 대한 로그들을 SL = {L_{O1}, L_{O2}, …, L_{On}}라 하자. 이 때 T가 성공적으로 DB에 COMMIT되었을 때, 로그 집합 SL의 원소 L_{Oi}를 T에 대한 *변경 로그*라 한다.

이 논문에서는 트랜잭션에 의해 수행되는 연산 및 그에 대한 변경 로그를 아래 표와 같이 가정하고 있다. 연산의 경우 삽입/삭제/수정의 3가지 종류가 있는데, 수정의 경우 공간 객체의 기하 정보(Geometry)에 대한 수정만을 의미하고 있다. 각 연산에 해당되는 변경 로그는 <Log ID, 연산의 종류, 객체 ID, 로그가 기록되는 시간(Timestamp), 객체값>으로 이루어진다. 수정 연산의 경우 객체 ID를 유지한 상태에서 Delete 로그 뒤에 다시 Insert 로그를 기록하는 방식을 따른다.

표 1 변경 연산의 종류와 변경 로그

연산의 종류	변경 로그
공간 객체의 삽입	<LID, Insert, OID, TS, Val>
공간 객체의 삭제	<LID, Delete, OID, TS>
공간 객체의 수정 (기하 정보)	<LID, Delete, OID, TS> <LID, Insert, OID, TS, Val>

앞에서 언급한 변경 로그는 서버 데이타에 대한 트랜잭션이 수행될 때마다 계속 쌓이게 되는데, MC_i로 변경 내용을 효율적으로 전송하기 위해서는 그 크기를 줄이고, 빠른 속도로 검색하는 것이 필요하다. 변경 로그의 경우 변경 내용만 유지하는 것이 주목적이므로 REDO 연산을 하기 위한 최소한의 로그를 제외한 불필요한 로그들은 삭제되어야 한다. 이 논문에서는 이러한 불필요한 로그가 발생하는 경우를 다음 4가지로 분류하고 있다.

참조시간 원인의 경우 모든 MC_i들의 최근 일관성 제어 시간 보다 더 오래된 변경 로그들의 경우 서버에서 더 이상 유지할 필요가 없다는 것을 의미하며, 영역외생성 원인의 경우 MC_i들의 CR_i(Cached Region)를 벗어나는 지역에서 발생하는 변경 연산의 경우 로그를 유지할 필요가 없다는 것을 나타낸다. 연산중복 원인은 하나의 객체에 대해서 여러 번의 변경 연산이 일어났을 경우 최근의 것만 유지해주면 되는 경우를 의미하고, 통합관리 원인의 경우 모든 클라이언트들을 위한 변경 로그들을 통합해서 관리함으로 인해서 해당 MC_i에게 필요 없는 로그들도 검색될 수 있음을 말해주고 있다. 이러한 불필요한 로그들은 생성 단계에서 로그 회피(Log Avoiding) 기법 및 로그 취소(Log Canceling) 기법을 이용하여 제거할 수 있으며, 검색 단계에서 로그 여과(Log Filtering) 기법과 로그 정화(Log Purging) 기법을 이용하여 제거할 수 있다. 이 논문에서 제시하는 불필요한 로그에 대한 정의 및 제거 기법에 관해서는 다음 절부터 구체적으로 설명한다.

4.2 변경 로그의 생성

공간 객체에 대한 변경 작업(삽입/삭제/수정)이 일어나면 변경 로그가 생성되는 데, 이 때 불필요한 로그도 같이 생성되게 된다. 변경 로그의 생성을 캐쉬된 영역 CR_i를 기준으로 분류해 보면 다음과 같다. 먼저 삽입/삭제를 살펴보면, CR_i들과 중첩되는 영역에 변경 작업이 일어나는 경우(그림 5의 ①)와 중첩되지 않은 영역에 변경 작업이 일어나는 경우(그림 5의 ②)로 나눌 수 있다. 그런데 전자의 경우 변경 정보를 필요로 하는 MC_1이 존재하므로 Insert/Delete 로그를 각각 생성하면 되지만 후자의 경우는 CR_i들의 영역 바깥이므로 일관성 유지를 하는데 불필요한 로그로 생성할 필요가 없다(영역외 생성).

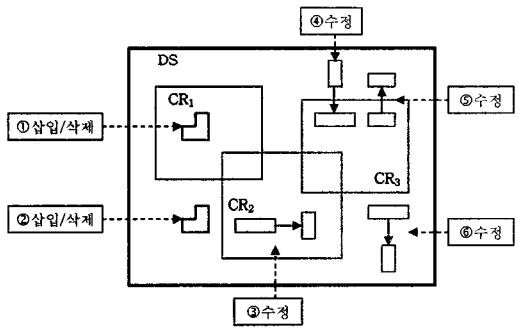


그림 5 영역외생성(Spatial Outbound)

이 논문에서는 이와 같은 변경 로그를 무관심 로그(Indifferent Log)라 하고 아래와 같이 정의한다.

정의 2. 일관성 유지 영역(CCR : Consistency Control Region)

서버에서 유지되고 있는 모든 CR_i들의 Union

정의 3. 변경 로그 영역(MBR(Log_i) : Update Log

표 2 발생 원인의 종류와 불필요한 로그

발생 원인	불필요한 로그	제거 방법
Reference Time(참조시간)	Obsolete Log(오래된 로그)	로그 정화(Log Purging)
Spatial Outbound(영역외생성)	Indifferent Log(무관심 로그)	로그 회피(Log Avoiding)
Repeated Operation(연산중복)	Repeated Log(반복된 로그)	로그 취소(Log Canceling)
Integrated Management(통합관리)	Overlapped Log(중첩된 로그)	로그 여과(Log Filtering)

Region)

Log_i가 공간 객체 O_i에 대한 변경 로그라고 할 때, O_i의 기하 정보(Geometry)에 대한 MBR(Minimum Bounding Rectangle)을 Log_i의 *변경 로그 영역*이라 한다.

정의 4. 무관심 로그(Indifferent Log)

Indifferent Log = {Log_i | CCR *Overlap* MBR(Log_i)= ∅,i∈all Update Log Index}

수정 작업시 발생하는 변경 로그를 수정전/후의 CR_i와 MBR(Log_i)사이의 관계에 따라 분류해 보면 1) Overlap ⇨ Overlap(그림 5의 ③), 2) Disjoint ⇨ Overlap(그림 5의 ④), 3) Overlap ⇨ Disjoint(그림 5의 ⑤), 4) Disjoint ⇨ Disjoint(그림 5의 ⑥)인 경우로 나눌 수 있다. 1)의 경우 수정전/후가 모두 무관심 로그에 해당되지 않으므로 Delete/Insert 로그가 생성되지만 2)의 경우에는 Delete 로그가 무관심 로그에 해당되므로 생성하지 않고, Insert 로그만을 생성하게 된다. 3)의 경우는 2)와 반대의 경우로 Delete 로그만을 생성하게 되고, 4)의 경우 모두가 무관심 로그가 되어 변경 로그를 생성하지 않게 된다. 이 논문에서는 무관심로그에 대해 로그를 생성하지 않는 기법을 로그 회피(Log Avoiding)라 한다.

앞에서 말한 영역외생성 원인 외에 같은 공간 객체에 대해서 중복해서 연산이 일어났을 때도 불필요한 로그가 발생하게 된다(연산중복 원인). 예를 들어 아래 그림 6과 같은 경우를 살펴보자. 먼저 MC_1은 t0에서 일관성 제어(그림에서 CC_1)를 서버에 요청하고, 서버는 t0 이전에 발생한 변경 로그들을 MC_1에게 전송한다. 그 뒤 t1에서 ID가 O1인 공간 객체가 CR_1내에 삽입되고, 이에 해당하는 L1 로그가 생성된다. 그 뒤 t2, t3에서 O1을 수정(그림에서 Move)하게 되고, 그에 대한 로그 L2, L3, L4, L5가 생성되게 된다. 만약 MC_1이 t4에서 일관성 제어를 서버에 요청하게 되면 L1-L5에 해당하는 변경 로그가 전파되게 되고, MC_1은 이 로그를 차례대로 캐쉬에 적용하여 일관성을 유지하게 된다. 그런데 자세히 살펴보면 변경 로그 집합(ordered를 가정) {L1, L2, L3, L4, L5} 모두를 MC_1에서 REDO를 하는 것과 {L5}만을 전달하여 REDO를 한 결과는 같다는 것을 알 수 있다. 즉 L1-L4 변경 로그는 일관성 유지에 불필요한 반복된 로그로 제거되어야 한다. 반복된 로그에 대한 정의는 아래 정의 5와 같다.

정의 5. 반복된 로그(Repeated Log)

Repeated Log = {Log_i | (TS(Log_i)≠TS(Log_j))∧ (OID(Log_i)=OID(Log_j)), i,j∈all Update Log Index}

(TS(Log_i)는 Log_i의 timestamp, OID(Log_i)는 Log_i가 가지고 있는 OID값)

정의에서 의미하는 바는 만약 timestamp가 다르고 OID가 같은 변경 로그가 존재 한다면 그 변경 로그들은 반복된 로그가 됨을 의미한다. 예를 들어 그림 6에서 L2와 L3의 경우 timestamp가 t2로 같기 때문에 반복된 로그가 되지 않으며 L1과 L2의 경우에는 같은 객체 ID에 다른 timestamp를 가지므로 반복된 로그임을 알 수 있다. 이러한 반복된 로그들은 아래 규칙 1에 의해서 제거될 수 있다(로그 취소(Log Canceling)).

규칙 1. 변경 로그 취소(Update Log Canceling)

하나의 공간 객체에 대한 반복된 로그들의 집합(RL)에서 다음 조건을 만족하는 로그쌍(Log_i, Log_j)은 취소(cancel)될 수 있다.

조건)

1. (연속조건) TS(Log_i)<TS(Log_k)<TS(Log_j)인 Log_k가 존재하지 않아야 한다.
2. (연산종류조건) Log_i는 Insert에 대한 로그, Log_j는 Delete에 대한 로그이어야 한다.(또는 그 반대의 경우)
3. (순서조건) 1,2 조건을 만족하는 로그쌍 (Log_i, Log_j)에 대해서 Log_k<Log_i이고 조건 1,2를 만족하는 Log_k가 존재해서는 안된다.

위 규칙에서 연속 조건과 연산 종류 조건의 경우 Insert 로그와 Delete 로그가 연속적으로 존재해야함을

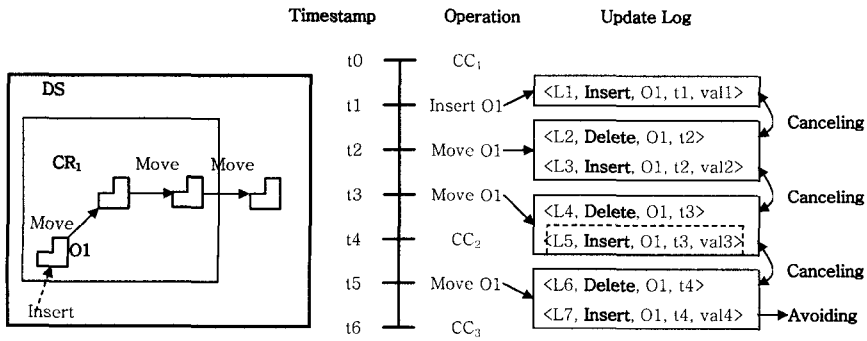


그림 6 연산중복(Repeated Operation)

의미한다. 순서조건의 경우에는 반복된 로그들에 이 규칙을 적용할 때 timestamp가 작은 로그에서 큰 로그로 점차적으로 적용시켜 나가야 함을 의미한다. 그림 6에서 이 규칙을 적용시켜 보면, t2에서 (L1, L2)가 규칙을 만족하는 로그쌍이 되어 삭제가 되고, t3에서는 (L3, L4)가 삭제되게 된다. 결국 t4에서는 L5만 MC_1으로 전달되어 효율적으로 일관성을 유지시킬 수 있게 된다. 만약 t5에서 수정연산이 일어나게 된다면 (L5, L6)는 변경 로그 취소 규칙에 의해 삭제되고, L7의 경우 무관심 로그에 해당되어 생성되지 않게 된다. 이 경우 t6에서 일관성 제어 요청이 있으면 어떤 변경 로그도 MC_1으로 전파되지 않는다. 즉 로그 회피(Log Avoiding)와 로그 취소(Log Canceling) 기법을 적용하지 않았을 경우 전달되는 L1-L7의 변경 로그가 전파되지 않고도 똑같은 결과를 얻을 수 있다.

앞의 두가지 기법을 적용하여 각 연산에 대한 변경 로그 생성 기법 알고리즘은 다음과 같다. Insert 연산일 경우 알고리즘 1에 의해 생성 가능하며, Delete 연산일 경우 알고리즘 2에 의해서 생성 가능하다. Update 연산의 경우 수정전의 객체를 인자로 하여 알고리즘 2를 호출한 뒤 수정 후의 객체를 인자로 하여 알고리즘 1을 호출하면 된다.

알고리즘 1의 Step1에서는 먼저 삽입하려는 객체가 일관성 유지 영역에 속하는지 테스트하기 위해 공간 객체의 MBR과 Overlap 관계에 있는 CR_i의 개수를 구한다. Step2에서 만약 개수가 0일 경우 일관성 유지 영역 밖에 공간 객체의 MBR이 존재한다는 의미이므로 변경 로그를 생성하지 않고 return 하게 된다. Step3에서는 일관성 유지영역에 속하는 경우이므로 변경 로그를 생성하고, 변경 로그 저장소에 저장하게 된다.

알고리즘 2. ***Create_Insert_Log(Obj:Object) : Bool***

```
begin
   // Step1 : 삽입하려는 공간 객체의 MBR과 모든
     CRi간의 overlap 개수 구하기
   RefCount = FindOverlaps(Obj);
   // Step2 : 무관심 로그일 경우 로그를 생성하지
     않음 - 로그 회피(Log Avoiding)
   if(RefCount == 0) return FALSE;
   // Step3 : 변경 로그 생성 및 저장
   Log = CreateLog(RefCount, INSERT, Obj);
   StoreLog(Log);
   return TRUE;
end
```

알고리즘 2에서는 Step1과 Step2, Step4의 경우 위의 알고리즘과 동일하며, Step3에서 로그 취소 과정을 거친다는 점에서 차이가 난다. Step3에서 먼저 같은 객체 ID를 가진 제일 마지막 로그를 찾은 뒤, 그 로그가 Insert 로그인지 검사하여 만약 Insert 로그일 경우 그 로그를 삭제하고, Delete 로그는 생성하지 않게 된다.

알고리즘 3. ***Create_Delete_Log(Obj:Object):Bool***

```
begin
   // Step1 : 삽입하려는 공간 객체의 MBR과 모든
     CRi간의 overlap 개수 구하기
   RefCount = FindOverlaps(Obj);
   // Step2 : 무관심 로그일 경우 로그를 생성하지
     않음 - 로그 회피(Log Avoiding)
   if(RefCount == 0) return FALSE;
   // Step3 : 중복된 로그일 경우 로그를 생성하지
     않음 - 로그 취소(Log Canceling)
   // 같은 객체에 대한 이전 로그가 있는지 검색
   PrvLog = FindLog(Obj);
   if(PrvLog != NULL){
      // 조건을 만족하는 Insert 로그가 있는가?
      if(IsInsertLog(PrvLog)){
         // 이전 로그를 지우고 delete로그는
           생성하지 않음
         CancelLog(PrvLog);
         return FALSE;
      }
   }
   // Step4: 변경 로그 생성 및 저장
   Log=CreateLog(RefCount,DELETE,Obj);
   StoreLog(Log);
   return TRUE;
end
```

4.3 변경 로그의 검색

MC_i가 일관성 제어 요청을 서버로 하게 되면 서버는 MC_i에 해당하는 CR_i를 이용하여 변경 로그들을 검색한 다음 MC_i로 전달하게 된다. 이 때 검색되어진 변경 로그는 MC_i의 마지막 일관성 제어 timestamp보다 이전 값을 가지게 되어 더 이상 서버에 유지할 필요가 없는데 이러한 변경 로그들을 오래된 로그(Obsolete Log)라 부르고 이를 제거하게 된다(로그 정화(Log Purging)). 오래된 로그에 대한 정의는 아래와 같다.

정의 6. 오래된 로그(Obsolete Log)

Obsolete Log={Log_i | CR_i *Overlap* MBR(Log_i)≠∅)∧ (TS(Log_i)<TS(Current CC)), i∈all Update Log Index}

(TS(Current CC) = MC_i가 현재 일관성 제어를 요청한 시간)

그런데 오래된 로그의 정의에 의해서 MC_i가 일관성 유지를 요청했을 때 해당 CR_i에 포함되는 변경 로그들을 무조건 삭제할 수는 없다. 그림 7의 경우를 살펴보면 MC_1과 MC_2가 각각 일관성 유지 요청을 한 후에 O1이 Insert, O2가 Insert, O3가 Update 되었다. 그리고 t5에서 MC_1이 일관성 요청을 하게 되었을 때 {L1, L2} 변

경 로그가 검색 결과로 선택되어 MC_1으로 전송되고, 더 이상 일관성 유지에 사용되지 않는 오래된 로그이므로 삭제하게 된다. 그러나 L2가 삭제될 경우 t6에서 MC_2가 일관성 요청을 할 때 필요한 로그임에도 불구하고 {L3, L4}만 전송되는 문제점이 발생한다. 이러한 문제점은 CR_i들간의 중첩된 영역안에서 연산이 이루어질 경우 발생하는데, 이 때 변경 로그에 참조 확률 카운트(Reference Probability Count)를 둠으로써 해결할 수 있다. 즉 CR_i의 중첩된 회수에 따라서 참조 카운트를 로그에 기록하고, 로그가 제거될 때에는 참조 확률 카운트를 1씩 감소시켜서 0이 될 때 완전히 삭제를 시키면 된다. 그림에서 L2의 경우 변경 로그의 제일 뒷부분에 2가 있는데 이것은 2개의 CR_i가 중첩된 영역에 로그가 생겼기 때문이다.

참조 확률 카운트에 의해서 변경 로그를 바로 삭제하지 않는 방법을 사용하면 나중에 이미 일관성 유지에 사용되었던 변경 로그가 해당 MC_i에게 다시 검색되어질 가능성이 있는데 이 논문에서는 이를 통합관리 원인이라 정의한다. 이 원인이 발생하는 이유는 변경 로그를 통합하여 한번만 저장하는 방식을 따르기 때문인데, 만약 상태 유지 서버와 같이 각 MC_i별로 변경 로그를 저장하게 되면 CR_i들이 중첩되어진 영역에서 발생하는 변경 로그의 경우 같은 로그가 여러 번 저장되어 서버의 저장공간을 낭비하게 되므로 이 논문의 방식을 따르게 된다. 이 논문에서는 통합 관리에 의해서 삭제되지 않고 남아 있는 로그를 중첩된 로그라 정의하고 검색시에 해당 MC_i에게 필요한지를 판단한 뒤 여과 또는 선택하는 방식을 취한다. 아래는 중첩된 로그에 대한 정의를 보여주고 있다.

정의 7. 중첩된 로그(Overlapped Log)

Overlapped Log={Log_i | Count(CR_i *Overlap* MBR(Log_i)) ≥2,i∈all Update Log Index}

그림 7의 t6에서 MC_2가 아닌 MC_1이 다시 일관성 제어 요청을 했다고 가정해보자. 이 경우 마지막으로 일관성 제어를 요청한 시점 이 후로 변경된 사항이 없으므로 검색 결과가 없어야 되는데, 중첩된 로그인 L2가 검색되게 된다. 이 경우 다음 조건에 해당되는 변경 로그를 검색 결과에서 여과(로그 여과(Log Filtering))함으로써 MC_i로 전송되는 것을 막을 수 있다. 규칙 2에서는 마지막으로 일관성 유지를 요청했던 시간 이전의 변경 로그들은 검색 결과에서 제거해야 함을 말해주고 있다.

규칙 2. 변경 로그의 여과(Update Log Filtering)

CR_i와 Overlap 되어지는 Log_i에 대해서 다음 조건을 만족할 경우 검색결과에서 제외시킨다.

조건) TS(Log_i) < TS(Last CC) (TS(Last CC)는 MC_i가 마지막으로 일관성 제어를 요청한 timestamp)

알고리즘 4의 경우 일관성 유지 요청시에 호출되는 함수이다. Step1에서는 모바일 클라이언트의 캐쉬 영역 정보를 이용하여 영역내에 포함되는 변경 로그들을 검색하는 과정이다. Step2에서는 규칙 2를 적용하여 마지막 일관성을 유지했던 시간 이후의 로그들만을 결과에 포함시키는 과정이며, Step3에서는 현재 시간을 일관성 유지 시간으로 갱신해주는 과정이다. Step4에서는 참조 확률 카운트를 1 감소시켜 0이 되는 변경 로그의 경우 삭제를 시켜주게 되고, 이 과정이 끝난 뒤 결과 변경 로그들을 MC_i로 전달하게 된다.

알고리즘 4.

Search_Update_Log(MC:ClientInfo):set(Update_Log)

```
begin
  // Step1 : CR과 Overlap되는 모든 변경 로그를
     검색
  UpdateLogs = FindUpdateLogs(MC.CR);
  // Step2 : 해당 MCi에 필요없는 로그들은 여과
     - 로그 여과(Log Filtering)
  for each log in UpdateLogs do
     // 마지막 일관성 제어 시간 이후 로그만
        결과에 추가
```

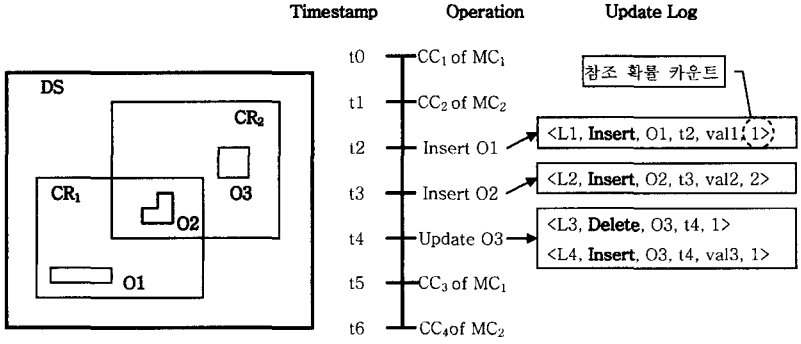


그림 7 참조시간(Reference Time), 통합관리(Integrated Management)

```
        if(log.ts > MC.ts) ResultLogs += log;
    end for
    // Step3 : MC의 timestamp 갱신
    MC.ts = CurrentTime();
    // Step4 : 더 이상 참조될 가능성이 없는
      로그들을 삭제 - 로그 정화(Log Purging)
    for each log in ResultLogs do
        // 참조 확률 카운트가 0이 되면 삭제
        log.ref--;
        if(log.ref == 0) RemoveLog(log);
    end for
    return ResultLogs;
end
```

4.4 변경 로그 인덱싱

일반적으로 회복 기법에서 사용하는 로그의 경우 순차적으로 저장하고, 순차적으로 탐색하는 기법을 기본으로 하고 있다. 그러나 이 논문에서는 MC_i에 해당되는 변경 로그만을 빠르게 검색하여 전달하여야 하므로, 모든 변경 로그를 순차적으로 검색하는 것은 문제가 있다. 그러므로 이 논문에서는 변경 로그를 공간 인덱스 또는 시공간 인덱스를 이용하여 변경 로그를 인덱싱하는 방법에 관하여 설명한다.

인덱싱을 하기 위해서는 객체와 키(Key)에 해당하는 객체의 필드(Field)가 있어야 한다. 여기에서는 변경 로그를 객체라 생각할 수 있고, 변경 로그 영역(정의 3 참고)을 키로 생각할 수 있다. Insert 로그와 Delete 로그의 경우 연산이 일어나는 객체의 기하 정보에 대한 MBR을 이용하면 되나 수정의 경우에는 두가지로 나눠볼 수 있다. Delete 로그에 대한 키는 객체가 수정되기 전의 기하 정보에 대한 MBR(before value)을 키로 사용하고, Insert 로그의 경우에는 수정 후의 기하 정보에 대한 MBR(after value)을 키로 사용하면 된다. 만약 시공간 인덱스로 구축을 하고자 한다면 변경 로그내의 timestamp값을 추가적으로 이용하면 된다.

일관성 제어 요청이 들어 올 때는 CR_i의 값을 이용하여 인덱스에 질의를 하여 변경 로그를 검색하면 되고, 로그내에 timestamp에 대해서 정렬을 한 뒤에 MC_i로 전달해주면 된다. 검색 단계에서 이루어지는 로그 여과(Log Filtering)를 수행하기 위해 공간 인덱스의 경우 정렬된 결과에 대해서 마지막 일관성 제어 시간(timestamp) 이전 변경 로그들을 여과하면 되고, 시공간 인덱스의 경우 질의 단계에서 시간값을 주어 질의를 하면 된다.

그림 8과 그림 9는 3개의 MC_i가 중복하여 캐싱하고 있을 때 변경 로그의 생성 및 인덱스 생성의 예를 보여주고 있다. t0에서 객체 ID가 O1인 객체를 삽입하게 되면 로그의 ID가 L1인 변경 로그가 생성되고, 객체 O1의 기하에 대한 MBR을 이용하여 R*-tree에 삽입하게 된다. t1에서 MC_1이 일관성 유지를 요청하게 되면 서버는 CR_1에 해당하는 영역 질의를 R*-tree로 수행하여 L1을 결과로 전달하게 된다. 그 이후 t2-t7까지 일관성 유지 영역내에서 Insert연산이 발생하여 변경 로그 L2-L7이 생성되게 된다. 이 때 L3의 경우 CR_1과 CR_2의 중첩 영역내에서 발생한 연산이므로 참조 확률 카운트가 2로 설정된다. t8에서 객체 O1을 이동하게 되면 L8과 L9이 생성되게 되는데, 그림9에서 나타나는 바와 같이 L8(Delete 로그)의 경우 객체가 수정되기 전의 기하에 대한 MBR을 이용하여 R*-tree에 삽입하고 L9(Insert 로그)의 경우 수정 후의 기하에 대한 MBR을 이용하여 R*-tree에 삽입하게 된다. 또한 L9의 경우 수정후 CR_1과 CR_2의 중첩 영역내에 포함되므로 참조 확률 카운트가 2로 설정된다.

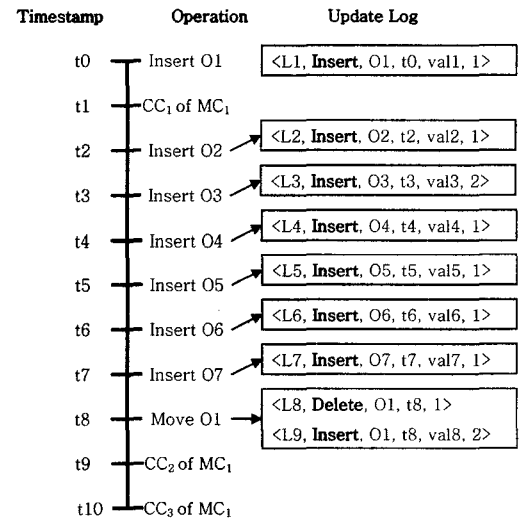
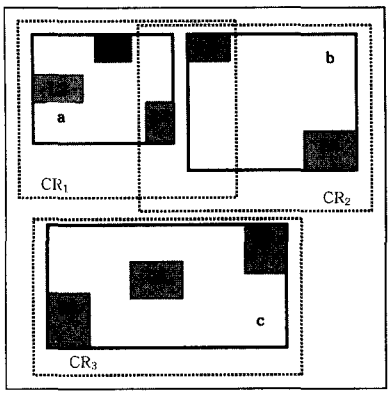


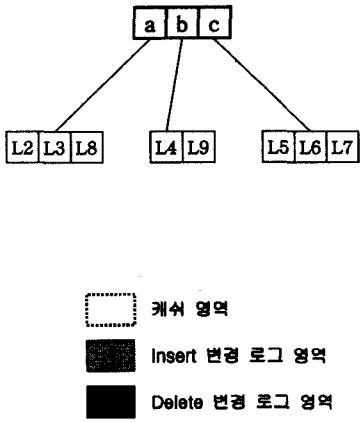
그림 8 변경 로그 생성 시나리오 예

t9에서 MC_1이 다시 일관성 유지 요청을 서버로 하였을 때 서버에서는 CR_1을 이용하여 인덱스에 질의하여 {L2, L3, L8, L9}를 결과로 얻게 된다. 이 때 변경 로그들의 확률 참조 카운트를 1씩 감소시키게 되는데 L2와 L8의 경우 0이 되어 변경 로그 및 인덱스내 엔트리들이 삭제되게 되고 L3와 L9의 경우 참조 확률 카운트가 1이 되어 그대로 남아 있게 된다. MC_1은 전달받은 결과에 대해서 timestamp 순으로 로컬 캐쉬에 적용하여 일관성을 유지하게 된다.

마지막으로 한 가지 더 고려해야 할 사항으로, 만약 MC_i들이 일관성 유지를 오랫동안 하지 않을 경우 서버에 저장되는 변경 로그 인덱스 및 변경 로그의 양이 방대해질 수 있는데 이 때는 두가지 방법이 있을 수 있다.



(a) 로그의 공간적 분포



(b) 공간 로그 인덱스 구조(R-Tree)

그림 9 변경 로그 인덱스 생성 예

첫번째는 일관성 유지 간격 한계치를 두는 방법이다. 만약 MC_i중에 일관성 유지 한계치를 넘는 시간 동안 일관성 유지를 하지 않을 경우 해당 CR_i에 속하는 변경 로그들을 삭제하고, 해당 MC_i가 다음에 일관성 유지 요청을 했을 때 로컬 캐쉬를 무효화시키고 다시 데이타를 내려받게 하는 방법이다. 두번째 방법은 CR_i에 속하는 변경 로그의 개수가 데이타의 개수보다 많지 않도록 유지하는 것이다. 만약 CR_i내에 변경 작업이 많이 일어나서 변경 로그의 개수가 실제 포함된 데이타의 수보다 많아질 경우 변경 로그들을 삭제하고, 첫번째 방법과 마찬가지로 해당 MC_i가 다음에 일관성 유지 요청을 했을 때 로컬 캐쉬를 무효화하고 다시 데이타를 내려받게 하는 방법이다.

5. 구현 및 성능 평가

5.1 시스템 구조

이 논문에서는 구현을 위하여 리눅스용 웹 GIS 솔루션[11]과 PDA용 지도 브라우저[12]를 사용하여 구현하였다. 다음은 서버의 모듈에 관한 설명이다.

- 질의 처리 모듈(Query Processor) : MC_i의 질의를 해석하고 처리한다.
- 로그 매니저(Log Manager) : 변경 로그 관련 질의를 해석하여 적절한 모듈에 전달하고, 처리된 결과를 질의 처리 모듈에 전달한다.
- 변경 로그 인덱스 매니저 (ULI Manager) : 변경 로그 인덱스에 대한 질의를 처리한다. 즉, 변경 로그 인덱스에 대한 삽입, 삭제, 검색을 수행한다.
- 변경 로그 최적화 매니저 (ULO Manager) : 변경 로그 DB에 대한 질의를 처리한다. 즉, 공간 로그의 삽입, 삭제 및 검색을 수행하며 로그 회피(Log Avoiding), 로그 취소(Log Canceling), 로그 여과(Log Filtering), 로그 정화(Log Purging)를 수행하는 모듈이다.
- MC_i 정보 매니저 (MCI Manager) : MC_i들이 서버에 등록/해제할 때 정보들(CR_i, TS 등)을 관리한다.

서버 모듈 구현시 고려해야 할 사항으로는 공간 데이타의 업데이트 툴에서 수정 작업이 일어날 때 반드시 로그 매니저를 호출하여 변경 로그를 생성하게 해주어야 한다는 점이다. 또한 4.4절의 마지막에 언급한 변경 로그 및 변경 로그 인덱스가 방대해 지는 것을 막기 위한 방법도 고려해 주어야 한다.

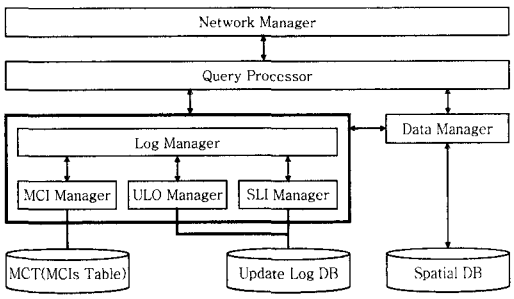


그림 10 변경 로그를 지원하는 GIS 서버

다음은 모바일 클라이언트(MC_i) 시스템 구조이다. 캐쉬 매니저(Cache Manager)는 MC_i의 요청을 받아서 서버에 연결하여 일관성 유지를 담당하는 모듈이다. 또한 서버에서 받은 일관성 유지 정보(변경 로그)를 로컬 캐쉬에 반영하는 기능을 수행한다. 캐쉬 매니저는 로컬 캐쉬에 대한 정보(MC Information)를 유지하고 있으며 서버에 요청시 참조하게 된다.

모바일 클라이언트를 구현할 때 고려해야할 사항으로는 무선 환경 및 단말기의 불안정한 특성으로 인하여 변경 정보 전송 실패가 일어났을 때의 상황을 고려해주

어야 한다는 것이다. 즉 서버에서는 전송 뒤에 변경 로그를 삭제했는데 모바일 클라이언트에서 발생한 문제로 인하여 일관성 유지를 하지 못했을 경우 문제가 발생할 수 있다. 이 때는 서버의 변경 로그를 전송 뒤 바로 삭제하지 않고 모바일 클라이언트의 확인 메시지를 수신한 뒤에 삭제하는 방법을 취할 수 있을 것이다.

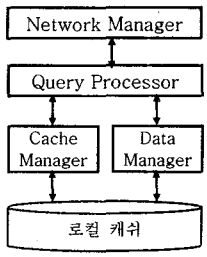


그림 11 모바일 클라이언트 시스템 구조

5.2 구현 결과

구현 및 성능 평가를 위해서 서버로는 Pentium III 1GHz 512MB의 리눅스가 설치된 컴퓨터에 리눅스용 웹 GIS 솔루션[11]의 서버 코드를 일부 수정하여 사용하였다. 또한 모바일 클라이언트를 위해서는 iPaq3630 32MB RAM에 PDA용 지도 브라우저[12]의 코드를 수정하여 사용하였다. 서버의 데이타셋으로는 부산광역시 금정구와 동래구를 합친 지역(가로 약 8Km, 세로 약 14.5Km)을 가정하였다. 그리고 5개의 MC_i 정보가 서버에 존재한다고 가정하였다.

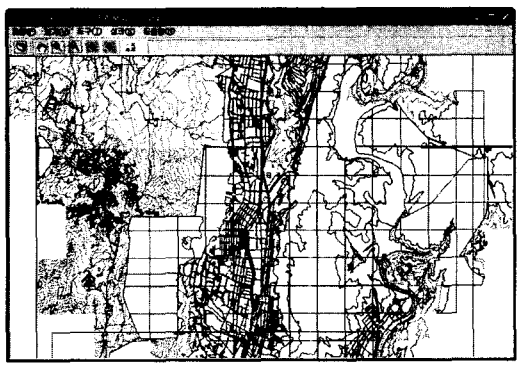


그림 12 데이타셋 화면

아래는 그림 6의 t4까지 진행되었을 경우의 화면 덤프를 보여주고 있다. 그림과 같이 사용자는 변경 로그만(아래의 경우 변경 로그 1개)을 이용하여 일관성 제어를 하게 되므로 무선 환경에서 통신 비용을 상당히 줄일 수 있다.

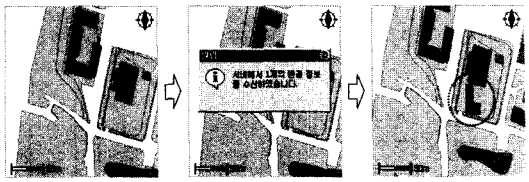


그림 13 변경 로그를 이용한 일관성 유지

5.3 성능 평가

성능 평가를 위해 이 논문에서는 세가지를 비교하고 있다. SL(Simple Logging)의 경우 단순히 순차적으로 변경된 정보를 로그로 기록하는 방식을 말하며, ULI (Update Log Indexing)기법은 변경 로그에 인덱싱 (R*-tree) 기법을 적용한 방식을 말하고, ULO(Update Log Optimization)는 이 논문에서 제시하는 기법들을 적용한 방식을 의미한다. 다음은 이에 대한 실험 결과를 보여주고 있다.

• 실험 1 : 변경 정보의 증가에 따른 검색 시간

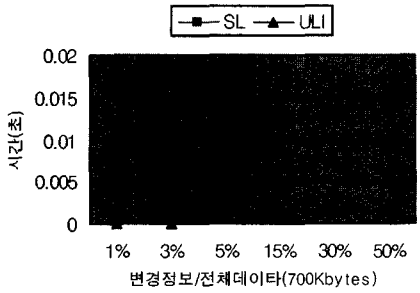


그림 14 변경 정보 증가에 따른 검색 시간

변경 정보가 전체 데이타 집합에서 차지하는 비율이 점점 증가함에 따라서 일관성 유지를 위해 필요한 정보를 검색하는 시간을 측정하였다. 이 실험을 통해 순차적으로 검색하는 SL방식과 변경 로그 인덱스를 이용하는 ULI기법의 성능을 비교하고자 하였다. 이 실험의 결과, 첫째, SL 기법은 변경 정보가 증가함에 따라서 검색 시간이 로그를 처음부터 끝까지 순차 검색하기 때문에 일정한 비율로 증가함을 알 수 있다. 둘째, ULI 기법에서는 변경 정보가 증가하더라도 R*-Tree에 의한 검색을 수행하므로 검색 시간이 거의 변함이 없음을 알 수 있다. 그러나, 변경 정보의 수가 들어나면 공간 로그 인덱스의 크기도 커지므로, 검색 시간이 조금 증가한 것을 알 수 있다.

• 실험 2 : 변경 정보의 증가에 따른 전송량

변경 정보가 전체 데이타 집합에서 차지하는 비율이 점점 증가함에 따라서 일관성 유지를 위해 필요한 정보의 양을 측정하였다. 이 실험을 통해서 이 논문에서 제

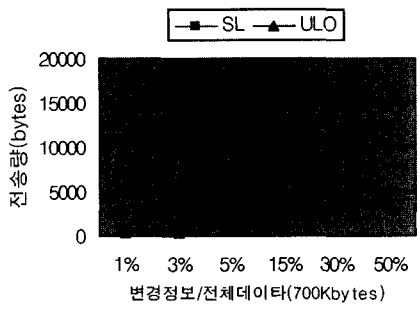


그림 15 변경 정보 증가에 따른 전송량

시하는 불필요한 로그 제거 기법의 효과를 검증하고자 하였다. 이 실험의 결과에서는, 첫째, SL 기법에서는 변경 정보가 누적되어 불필요한 로그가 검색, 전송되기 때문에 수정(Update)되는 객체의 수에 영향을 받지만, 대체로 증가하는 것을 알 수 있다. 둘째, ULO 기법은 변경 정보가 누적되지 않고 불필요한 정보가 전송되지 않으므로 전송량이 SL기법에서 보다 항상 같거나 적게 된다. 전송량이 같거나 비슷한 경우는 모든 변경 정보가 새로운 객체의 삽입인 경우이다. 그러나, 수정되는 객체가 많을수록 ULO 기법의 전송량은 SL 기법보다 효과적임을 알 수 있다. 시간이 지남에 따라서 SL 기법에서는 로그가 계속해서 누적되는 반면 ULO기법에서는 불필요한 로그들이 생성되지 않거나 제거되기 때문에 거의 일정한 수준을 유지하고 있다.

- 실험 3 : 변경(Insert, Delete, Update)에서 수정(Update)의 비중에 따른 전송량 변화

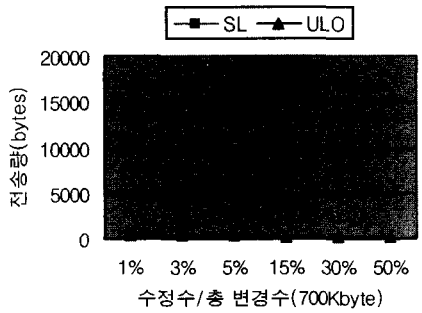


그림 16 변경에서 수정의 비중에 따른 전송량

변경(Insert, Delete, Update)중에서 수정(Update)이 차지하는 비율이 점점 증가함에 따라서 일관성 유지를 위해 필요한 정보의 양을 측정하였다. 이 실험에서는 중복 로그 취소 기법의 효과를 측정하고자 하였다. 이 실험의 결과에서는 첫째, SL 기법에서 로그는 변경의 종류에 상관없이 누적되고, 해당 영역에서 최근 발생한 모든 변경 정보를 찾으므로 변경에 횟수에 비례하여 전송량이 증가함을 알 수 있다. 둘째, ULO 기법에서는 중복 로그를 취소하기 때문에 전송량이 줄어들 가능성이 높으며 특히 동일 객체의 수정이 많을수록 전송량은 줄어들게 된다.

6. 결 론

이 논문에서는 모바일 환경에서 클라이언트에 캐쉬된 공간 데이타에 대한 효율적인 일관성 제어를 위해 변경 로그 전파 기법을 제시하였다. 로그의 경우 변경 작업이 계속 됨에 따라서 점차 누적되어 서버 공간을 많이 차지하게 되고, 검색 시간이 늘어나는 문제점을 가지고 있다. 그러므로 이 논문에서는 불필요한 로그 4가지(오래된 로그, 무관심 로그, 반복된 로그, 중첩된 로그)를 정의하고 이를 제거하기 위한 기법 4가지(로그 회피(Log Avoiding), 로그 취소(Log Canceling), 로그 여과(Log Filtering), 로그 정화(Log Purging))를 제시하였다. 또한 순차적인 로그 검색 문제를 해결하기 위해서 변경 로그를 인덱싱하는 기법에 관해서도 논의하였다. 이 논문에서 제시하는 기법을 이용하면 최소한의 변경 정보만을 빠르게 검색하여 모바일 클라이언트로 전송할 수 있기 때문에, 무선 통신 비용을 최소화 할 수 있다. 향후 연구로는 모바일 클라이언트에서 서버로 변경내용을 효율적으로 전파할 수 있는 기법에 관한 연구가 필요하다.

참 고 문 헌

[1] Field Solutions Technical White Paper, Tadpole Technology, Inc.

[2] G. Forman and J. Zahorjan, "The challenges of mobile computing," IEEE Computer, Vol.27, No.4, pp.38-47, Apr. 1994.

[3] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," Proc of the fifteenth annual ACM symposium on Principles of distributed computing, ACM Press, New York, 1996, pp. 1-7.

[4] A.Zaslavsky and Z.Tari., "Mobile Computing: Overview and Current Status," Australian Computer Journal, 30, 1998.

[5] A. Kahol, S. Khurana, Sandeep S.S. Gupta, Pradip K. Srimani, "A Strategy to Manage Cache Consistency in a Disconnected Distributed Environments," IEEE Transaction On Parallel and Distributed Systems, vol. 12, no. 7, JULY 2001.

[6] Kian-Lee, Jun Cai, Beng Chin Ooi, "An Evaluation of Cache Invalidation Strategies in Wireless Environments," IEEE Transaction On Parallel and Distributed Systems, vol. 12, no. 8, AUGUEST 2001.

[7] D. Barbara and T. Imielinski, "Sleepers and Workaholics : Caching Strategies in Mobile Environments(Extended Version)," MOBI-DATA : An

Interactive J. Mobile Computing, Vol 1, Nov. 1994.
[8] H. Chung, H. Cho, "Data caching with Incremental Update Propagation in Mobile Computing Environments," Proc. Australian Workshop on Mobile Computing and Databases and Applications, Feb. 1996.
[9] J. Cai, K.L. Tan, and B.C Ooi, "On Incremental Cache Coherency Schemes in Mobile Computing Environment," Proc. 13th Int'l Conf. Data Eng. Apr. 1997.
[10] http://www.syncml.org, SyncML Sync Protocol version 1.1.
[11] http://www.cybermap.co.kr, ㈜사이버맵월드 홈페이지
[12] 모바일 사이버맵 2002, PDA용 지도 브라우저, 2002년 ㈜ 사이버맵월드 출시.
[13] M. Ruffin, "A Survey of Logging Uses," Proceedings of the 11th SRDS, Oct, 1992.
[14] B. Weiler, "SpeedLog : A Generic Log Service Supporting Efficient Node-Crash Recovery," IEEE, 1994.
[15] J.Jing, A.Elmagarmid, A.Helal, and R.Alonso, "Bit-Sequences : An Adaptive Cache Invalidation Method in Mobile Client/Server Environments," Mobile Networks and Applications, vol.2, no.2, 1997.
[16] A.P. Sistla, O. Wolfson, and Y. Huang, "Minimization of Communication Cost Trough Caching in Mobile Environments," IEEE Trans. Parallel and Distributed System, Vol 9, No. 4, Apr. 1998.
[17] V. Gaede, O. Gunther, "Multidimensional Access Methods," ACM Computing Surveys, Vol 30, no. 2, June 1998.

안 경 환
1997년 부산대학교 컴퓨터공학과 졸업(공학사). 1999년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 1999년~현재 부산대학교 대학원 컴퓨터공학과 박사과정

차 지 태
2000년 부산대학교 컴퓨터공학과 졸업(공학사). 2002년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사)

홍 봉 희
정보과학회논문지 : 데이타베이스
제 31 권 제 2 호 참조