모바일 트랜잭션을 위한 회복 기법의 설계 및 구현

강주호[0] 김동현 홍봉희
부산대학교 컴퓨터공학과 데이터베이스연구실
(jooho78[0], pusrover, bhhong)@ pusan.ac.kr

Design and Implementation of a Recovery Scheme for Mobile Transactions

Joo-Ho Kang[0] Dong-Hyun Kim Bong-Hee Hong
Dept. of Computer Engineering, Pusan National University

요 약

기존 검사점 기반의 회복 기법은 단절된 모바일 트랜잭션의 회복 정보를 서버의 안정 저장소에 저장할 수 없다. 따라서 단절 상태에서 공간 객체를 수정하고 있는 모바일 클라이언트에 장애가 발생하면 단절 시의 검사점 수행 상태로 회복하지 못하는 문제가 있다. 이 논문은 모바일 트랜잭션의 회복 정보를 서버에 저장하기 위하여 강제 로깅 기법을 사용한다. 그리고 모바일 트랜잭션의 장애를 회복하기 위해 강제 로깅된 로그를 이용하는 회복 기법을 제안한다. 이 회복 기법은 서버에서 공간 데이터에 로그를 순차적으로 반영하여 쓰기 집합을 생성한 후 클라이언트로 반환하는 기법이다. 또한 이러한 회복 기법을 지원하는 시스템을 공간 데이터 서버상에서 설계하고, 프로토타입을 구현하였다.

1. 서론

무선망과 모바일 기기의 발달로 현장에서 공간 데이터의 변경 작업이 가능해졌다. 사용자는 모바일 클라이언트에 공간 데이터를 다운로드 받은 후 현장에서 직접 변경한다. 모든 변경 작업이 완료되면 무선망을 통해 변경된 공간 데이터를 서버에 병합한다. 무선망은 유선망에 비해 자주 단절되는 특징을 가지기 때문에 일반적으로 모바일 클라이언트는 단절된 상태에서 변경 작업을 수행한다[2]. 그리고 모바일 클라이언트에는 안정 저장소가 없기 때문에 작업 수행 중에 장애가 발생하면 모든 변경 내용을 손실하게 된다[1]. 따라서 변경 작업의 손실 비용을 줄이기 위한 모바일 트랜잭션의 회복 기법에 관한 연구가 필요하다.

기존의 회복 기법[3,4]은 서버에서 각 클라이언트로 검사점을 수행할 때 동기적으로 로그와 변경된 객체들을 서버의 안정 저장소에 저장한다. 그러나 이 기법을 무선망을 이용하는 모바일 트랜잭션의 회복에 적용하면 단절된 클라이언트로부터 서버는 회복에 필요한 정보를 획득할 수 없다. 따라서 단절 상태의 모바일 트랜잭션은 검사점 수행 시 자신의 회복 정보를 서버에 저장할 수 없기 때문에 만약 해당 트랜잭션에 장애가 발생하면 최근의 검사점 상태로 회복할 수 없는 문제가 발생한다. 이러한 모바일 트랜잭션의 회복 비용은 단절 기간에 비례하여 증가한다.

이 논문은 단절 상태에서 회복에 필요한 정보를 저장하기 위하여 강제 로깅 기법을 이용한다. 강제 로깅 기법은 모바일 클라이언트가 주기적으로 서버에 연결하여 서버의 안정 저장소에 자신의 로그를 강제로 저장하는 기법이다. 그리고 대용량의 로그 데이터를 무선망을 통하여 효과적으로 전송하기 위하여 공간 객체에 대한 수정 연산을 세분화하여 로그 레코드의 크기를 줄인 새로운 로그 구조를 사용한다.

이 논문에서는 모바일 트랜잭션에 장애가 발생하였을 때 최근 상태로 회복하기 위하여 강제 로깅된 로그 데이터를 이용한 회복 기법을 제안한다. 제안한 회복 기법은 서버의 공간 객체에 대하여 기록된 로그의 연산을 적용하여 쓰기 집합을 생성하고, 이 쓰기 집합을 모바일 클라이언트로 전송한다. 그리고 이러한 회복 기법을 지원하는 회복 시스템을 공간 데이터 서버상에서 설계하고, 프로토타입을 구현하였다.

이 논문의 구성은 다음과 같다. 2장은 이 논문과 유사한 주제를 다루는 관련 연구를 소개한다. 3장에서는 강제 로깅 기법과 로그 구조를 제시한다. 4장에서는 새로운 로그 구조를 이용한 회복 기법을 설명한다. 5장에서는 회복 시스템의 구조 및 프로토타입을 제시한다. 마지막으로 6장에서 이 논문의 결론을 맺는다.

2. 관련 연구

클라이언트-서버 환경의 대표적인 회복 기법에는 ESM-CS와 ARIES/CSA가 있다. ESM-CS는 WAL 기법과 퍼지 검사점을 사용하여 클라이언트의 크래쉬 장애를 회복한다[3]. ARIES/CSA는 기존의 ARIES를 클라이언트-서버 환경으로 확장한 기법으로서 ESM-CS 기법과 유사하다[4]. 이 두 기법은 네트워크 장애를 고려한 처리가 없으므로 유선망에 비해 불안정한 모바일 환경에서는 문제점을 드러낸다. 또한 클라이언트는 회복을 위해 로그와 변경된 객체 모두를 서버에 전송하므로 좁은 대역폭을 가지는 모바일 환경에서는 시간 및 통신 비용이 크다.

모바일 분산 환경의 회복 기법인 [5]는 단절 상태에서도 회복 정보를 클라이언트의 지역 저장소에 저장하여 메시지 교환 없이 검사점을 수행한다. 그리고 장애가 발생하면 여러 클라이언트에 분산된 회복 정보를 수집하는 과정을 거친다. 그러나 단절 상태에 있는 클라이언트의 회복 정보에는 접근할 수 없으므로 완전한 회복이 불가능한 경우가 발생한다. [6]은 트랜잭션의 타이머를 동기화하여 검사점 수행 시점을 동기화하며, 클라이언트 간에 메시지를 교환할 때 각 트랜잭션의 타이머를 동기화할 수 있는 정보를 포함하여 전송한다. 그러나 본 논문의 대상 환경에서는 타이머 동기화를 위해 불필요한 메시지 교환을 필요로 하고 단절 상태의 클라이언트와는 타이머 동기화가 불가능하므로 적용하기 어렵다.

3. 로그 저장 기법

1. 강제-로깅 기법

모바일 클라이언트는 트랜잭션 수행 중 그림 1과 같이로그를 서버의 안정 저장소에 주기적으로 강제 로깅한다. 이 강제 로깅 기법은 로그 레코드 카운트(LRC: Log Record Count)와 최대 로그 레코드 카운트(MAX_LRC)에 의해 로깅 주기를 결정한다. 공간 객체를 변경하여 로그 레코드가 생성될 때마다 로그 레코드 카운트는 '1'씩 증가되고, 미리 설정된 최대 로그 레코드 카운트와 같아지면 강제 로깅을 수행한 후 로그 레코드 카운트를 '0'으로 재설정한다.

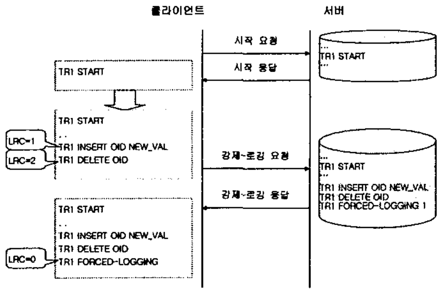

그림 1 LRC와 MAX_LRC를 이용한 강제 로깅

2. 로그 구조

서버에 저장되는 로그는 공간 데이터의 변경 내용을 저장하는 로그로서 표 1과 같이 삽입, 삭제, 수정 연산에 따라 레코드 유형이 분류된다[7].

TR_ID	INSERT	GEOM_TYPE	{(X, Y)}	
TR_ID	DELETE	OID		
TR_ID	MODIFY	OID	{(X, Y)}	
TR_ID	PARTIAL_MODIFY	OID	{(SEQ_NUM)}	{(X, Y)}
TR_ID	MOVE	OID	(ΔX, ΔY)	
TR_ID	INSERT_POINTS	OID	{(SEQ_NUM)}	{(X, Y)}
TR_ID	DELETE_POINTS	OID	{(SEQ_NUM)}	{(X, Y)}

표 1 로그구조

4. 회복 기법

1. 트랜잭션 정보 테이블

모바일 클라이언트는 안정 저장소가 없으므로 장애가 발생하면 모든 회복 정보를 손실하게 된다. 따라서 서버에서 클라이언트 식별자를 포함한 트랜잭션의 메타정보를 유지한다. 표 2는 서버에서 유지하는 트랜잭션 정보 테이블의 구조이다[7].

TR_ID	MC_ID	MBR

표 2 트랜잭션 정보 테이블

2. 회복 절차

공간 객체의 변경 작업을 수행하는 중 모바일 클라이언트에 장애가 발생하면 모든 공간 데이터와 트랜잭션 정보를 손실하게 된다. 따라서 사용자는 모바일 클라이언트의 모바일 아이피(Mobile IP)와 포트 넘버로 구성된 클라이언트 식별자를 메시지에 포함하여 서버에 회복 요청을 한다. 서버는 클라이언트 식별자를 이용하여 트랜잭션 정보 테이블을 검색하여 장애가 발생한 클라이언트의 트랜잭션 식별자와 최소경계영역 정보를 획득한다. 반면에 만약 트랜잭션 정보 테이블에 요청한 클라이언트의 정보가 없으면 해당 클라이언트에는 수행 중이던 트랜잭션이 없었음을 응답한다. 회복 연산은 최소경계영역에 해당하는 공간 데이터를 획득한 후, 트랜잭션 식별자를 이용하여 검색한 로그의 변경 연산을 순차적으로 반영한다. 생성된 쓰기 집합과 장애가 발생한 트랜잭션 식별자를 회복 요청한 클라이언트에 반환함으로써 서버에서의 회복 연산은 끝이 난다. 회복 응답을 받은 클라이언트는 쓰기 집합과 트랜잭션 식별자를 이용하여 이전 강제 로깅 시점에서 트랜잭션을 재시작한다. 그림 2는 회복 절차이다. TR_ID, MBR은 표 2의 트랜잭션 정보 테이블에 저장된 트랜잭션의 정보이다.

```
TR_ID          //MC_ID에 실행 중이던 트랜잭션의 식별자
MBR            //수정하던 최소경계영역
OBJs           //공간 데이터
status         //장애 이전의 트랜잭션 상태
MC_ID          //모바일 클라이언트의 식별자

Procedure SERVER_recovery( MC_ID )
Begin
    OBJs ← Ø;
    status ← START;

    //MC_ID를 이용하여 해당 모바일 트랜잭션 검색
    WHILE scan the TR information table
        IF MC_ID of TR information record = MC_ID THEN
            (TR_ID, MBR) ← retrieve recovery information
        END IF
    END LOOP

    //장애 발생 이전의 트랜잭션 상태를 조사한다
    WHILE scan the LOG backward UNTIL log record = <TR_ID
      START>
        IF log record = <TR_ID FORCED-LOGGING> THEN
            status ← ACTIVE;
        END IF
    END LOOP

    IF status = START THEN
    //강제-로깅된 로그가 없으면 MBR 영역의 공간 데이터를
    획득하여 반환
        OBJs ← retrieve spatial objects within MBR;
        send a message "reply_recovery( TR_ID, OBJs )";
    ELSE IF status = ACTIVE THEN
        //공간데이터와 로그를 이용하여 회복 연산 실행
        OBJs ← retrieve the spatial objects within MBR;
        generate write-set with TR_ID, MBR;
        send a message "reply_recovery( TR_ID, OBJs )";
    END IF
End
```

그림 2 회복 절차

회복 요청한 클라이언트로 반환되는 쓰기 집합은 최소경계영역 정보를 이용하여 획득한 공간 데이터에 트랜잭션 식별자로 검색한 로그의 변경 연산을 해당 공간 객체에 순차적으로 반영함으로써 생성된다. 그림 3은 쓰기 집합을 생성하는 절차이다.

```
OBJs           //반환할 쓰기 집합
Obj            //한 공간 객체
log_record     //공간 데이터에 반영할 로그 레코드
TR_ID          //트랜잭션 식별자
MBR            //최소경계영역

Procedure Recovery_Operation( TR_ID, MBR )
Begin
    OBJs ← spatial objects within MBR;
    WHILE scan LOG backward UNTIL log record = <TR_ID START>
    WHILE scan LOG forward
        log_record ← read a log record from LOG;
        IF TR_ID of log_record = TR_ID THEN
            WHILE scan OBJs
                Obj ← read a spatial object from OBJs;
                IF OID of Obj = OID of log_record THEN
                    update Obj with log_record;
                END IF
            END LOOP
        END IF
    END LOOP

    return OBJs;
End
```

그림 3 쓰기 집합 생성 절차

5. 시스템 구조 및 구현

1. 시스템 구조

강제 로깅 기법과 로그를 이용한 회복 기법을 적용하여 구현한 시스템의 구조는 그림 4와 같다. 서버는 리눅스 환경의 공간 데이터베이스를 대상으로 구현하였으며, 모바일 클라이언트는 Windows CE

3.0 환경에서 구현하였다.

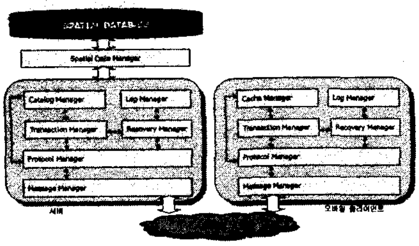

그림 4 시스템 구조

회복 시스템의 구현에서는 강제 로깅 기법과 로그를 이용한 회복 기법이 어떻게 정확히 동작하도록 설계하느냐가 중요하다. 서버 시스템은 메시지 관리 모듈, 프로토콜 관리 모듈, 트랜잭션 관리 모듈, 회복 관리 모듈, 로그 관리 모듈, 카탈로그 관리 모듈로 구성된다.

• **강제 로깅 기법** : 모바일 클라이언트의 회복 관리자는 한 공간 객체가 변경될 때마다 로그 레코드 카운트를 '1'씩 증가시킨다. 로그 레코드 카운트가 최대 로그 레코드 카운트와 같아지면 회복 관리자는 로그 관리자로부터 서버의 안정 저장소에 기록할 로그 데이터를 획득하고 프로토콜 관리자에게 강제 로깅을 요청한다. 프로토콜 관리자는 메시지 관리자가 서버에 연결을 설정한 후에 획득한 로그와 트랜잭션 식별자로 강제 로깅 요청 메시지를 구성하여 서버에 전송한다. 메시지를 받은 서버의 프로토콜 관리자는 회복 관리자에게 트랜잭션 식별자와 로그를 보내어 강제 로깅하도록 한다. 회복 관리자는 로그를 로그 관리자를 통해 안정 저장소에 저장한 후 프로토콜 관리자에게 강제 로깅이 성공적으로 수행되었음을 클라이언트에게 응답하도록 요청한다.

응답 메시지를 받은 클라이언트의 프로토콜 관리자는 회복 관리자에게 강제 로깅이 성공적으로 수행되었음을 알린다. 회복 관리자는 로그 관리자에게 트랜잭션 식별자와 함께 강제 로깅 로그 레코드를 로그 파일에 삽입하도록 요청한다.

• **로그를 이용한 회복 기법** : 모바일 클라이언트에 장애가 발생하면 클라이언트의 프로토콜 관리자는 클라이언트 식별자와 함께 회복 요청 메시지를 서버로 전송한다. 서버의 프로토콜 관리자는 회복 요청 메시지를 받은 후 모바일 클라이언트 식별자를 이용하여 카탈로그 관리자로부터 트랜잭션 식별자와 최소경계영역 정보를 획득한다. 그리고 획득한 트랜잭션 식별자와, 최소경계영역을 이용하여 회복 관리자에게 회복 요청을 한다. 회복 관리자는 최소경계영역 정보를 이용하여 공간 데이터 관리자로부터 공간 데이터를 획득하고, 트랜잭션 식별자를 이용하여 로그 관리자에게 해당 트랜잭션의 로그를 요청한다. 그리고 획득한 공간 데이터에 순차적으로 로그의 변경 연산을 반영하여 쓰기 집합을 생성한 후 프로토콜 관리자에게 보낸다. 프로토콜 관리자는 쓰기 집합과 트랜잭션 식별자로 회복 응답 메시지를 구성하여 메시지 관리자를 통해 장애가 발생한 클라이언트에게 전송한다.

서버로부터 회복 응답 받은 모바일 클라이언트의 프로토콜 관리자는 쓰기 집합을 캐쉬 관리자에게 보내어 관리하도록 하고, 트랜잭션 식별자를 이용하여 회복 관리자에게 회복 요청을 한다. 회복 관리자는 트랜잭션 식별자를 이용하여 트랜잭션 관리자에게 트랜잭션을 재시작하도록 요청한다.

2. 구현 예

회복 시스템의 구현 예는 그림 5와 같다. 이 예에서 최대 로그 레코드 카운트는 '2'라고 가정하였다. 즉 두 객체가 수정된 후에 강제 로깅이 일어난다. 그림 5 (a)는 트랜잭션을 시작하기에 앞서 서버의 공간 데이터의 일부를 클라이언트에 다운로드하여 디스플레이한 것이다. 트랜잭션을 시작한 후 그림 5 (b)와 같이 임의의 두 객체를 수정하면 그림 5 (c)와 같이 로그가 서버의 안정 저장소에 강제 로깅된다. 그림 5 (d)와 같이 임의의 한 객체를 수정한 다음 클라이언트에 장애가 발생하였다. 클라이언트가 서버에 회복을 요청하면 그림 5 (e)와 같이 이전 강제 로깅 시점, 즉 오류! 참조 원본을 찾을 수 없습니다.(b)시점의 공간 데이터를 받아서 트랜잭션을 재시작한다. 강제 로깅되지 않은 그림 5(d)의 변경 내용은 그림 5(e)에서 보듯이 회복에 반영되지 않았다

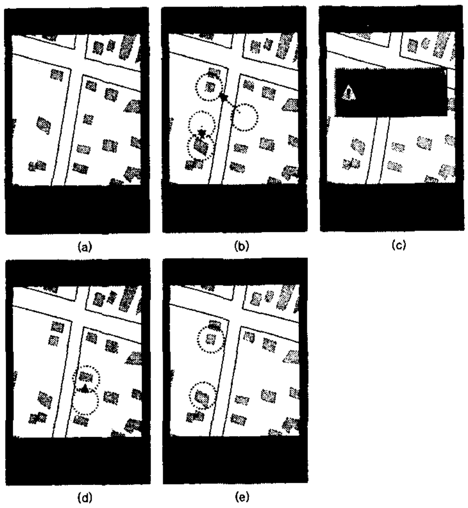

그림 5 구현 예

6. 결론

서버에서 동기적으로 수행하는 검사점은 단절 상태의 모바일 트랜잭션의 회복 정보를 획득하지 못한다. 이러한 모바일 클라이언트에 장애가 발생하면 단절 시의 검사점 상태로 회복하지 못하는 문제가 발생한다. 이 논문은 클라이언트가 회복 정보를 주기적으로 서버에 강제 로깅하는 기법을 사용하여 이 문제를 해결하였다. 그리고 모바일 트랜잭션의 장애를 회복하기 위해 강제 로깅된 로그를 이용한 회복 기법을 사용한다. 이 회복 기법은 서버에서 공간 데이터에 로그를 순차적으로 병합하여 쓰기 집합을 생성한 후 장애가 발생한 클라이언트로 반환하는 기법이다. 그리고 강제 로깅 기법과 로그를 이용한 회복 기법을 지원하는 시스템을 공간 데이터베이스 상에서 설계하고, 그 프로토타입을 구현하였다.

7. 참고 문헌

[1]. Gail E. Kaiser, "Cooperative Transactions for Multiuser Environments", Modern Database Systems, pp.409-433, 1995

[2] James J Kistler, M Satyanarayanan, "Disconnected Operation in the CODA file system", ACM Transactions on Computer Systems, Vol 10, No1, pp. 3-25, 1992

[3] Franklin, M., Zwilling, M., Tan, C.K., Carey, M., DeWitt, D., "Crash Recovery in Client-Server EXDOUS", Proc. ACM SIGMOD International Conf. on Management of Data, San Diego, June 1992

[4] C. Mohan, Inderpal Narang, "ARIES/CSA: A Method for Database Recovery in Client-Server Architectures", DataBase Tehnology Institute, IBM Almaden Research Center, San Jose, ACM SIGMOD, pp55-66, May. 1994

[5] R. E. Strom and S. Yemini, "Optimistic recovery in distributed systems", ACM Transactions on Computer Systems, vol. 3, no. 3, pp. 204-226, Aug. 1985

[6] N. Neves and W. K. Fuchs "Adaptive Recovery for Mobile Environments", Communications of the ACM, 40(1), Jan. 1997

[7] 강주호, 김동현, 홍봉희, "모바일 환경에서 공간 데이터 변경 트랜잭션을 위한 회복 기법, 한국정보과학회 데이터베이스 연구회, 세션B-3, pp. 177-184 2002