모바일 환경에서 동시 양방향 동기화 프로토콜의 설계

김홍기, 김동현

동서대학교

A Design of Concurrent Two-Way Synchronizations Protocol on a Mobile Environments

Hong-Ki Kim, Dong-Hyun Kim

Dongseo University

E-mail : inthestream@nate.com, pusrover@dongseo.ac.kr

요 약

모바일 기기와 무선 통신 기술이 발달함에 따라 모바일 기기에서 수집 또는 변경되는 대용량 시공간 데이터를 서버와 현장에서 동기화하는 서비스의 제공이 가능해졌다. 다수의 모바일 기기에서 변경된 대용량 시공간 데이터를 서버와 동기화하는 효율적인 양방향 동기화 프로토콜이 필요하다. 그러나 다수의 모바일 기기에 대하여 동기화 작업을 수행할 때 처리 시간이 길어지는 문제가 있다. 이 논문에서는 다수의 양방향 동기화 작업에 대하여 다중 큐를 이용하여 서버에서 동시 수행하는 처리기법에 대하여 제안한다.

ABSTRACT

As the mobile devices and the wireless networks have high-performance capabilities, it is possible to synchronize the spatio-temporal data of a server with the spatio-temporal data of a mobile device which are collected at a field. However, since the server process the synchronization which the model device requests, the whole synchronizations of mass mobile devices take long time. In this paper, we propose the scheme to process concurrently the synchronizations of mobile devices which does not conflict with others using the scheme of a multi-queue.

키워드

동기화 프로토콜, Mobile, Multi-queue

Ⅰ. 서 론

모바일 기기들이 고성능화되고 무선 네트워크도 기존에 비하여 대용량의 디지털 데이터를 고속으로 전송할 수 있게 되었다. 따라서 고성능화된 모바일 기기와 무선 네트워크 기술을 기반으로 다양한 형태의 지도 또는 위치 데이터를 수집할 수 있게 되었으며, 특히 기존에 제공하기 어려웠던 대용량의 시공간 데이터를 현장에서 바로 수집하여 동기화 서비스하는 것이 가능하게 되었다. 이러한 동기화 서비스는 서비스의 분야에 따라 수많은 모바일 기기로부터 동기화가 요청될 수 있다. 예를 들어 우체국에서 우체통 및 우편물의 위치정보를 관리하는 서비스에 이용할 경우 우편배달을 하는 모든 배달원들이 동기화를 요청할 수 있다. 그러나 기존의 서버와 모바일 기기간의 동기화 프로토콜은 다수의 동기화 작업에 대한 처리는 고려되지 않았다[1].

기존의 양방향 동기화 프로토콜은 동기화를 요청한 순서에 따라 순차적으로 동기화하는 프로토콜이다. 순차적 동기화 프로토콜은 다수의 모바일 기기로부터 동기화가 요청되었을 때 처리시간의 길이에 상관없이 동기화 요청 순서대로 동기화한다. 이 프로토콜은 처리시간이 긴 동기화작업 중에 처리시간이 짧은 동기화 요청이 들어와도 처리중인 동기화가 끝날 때까지 오랜 시간 동안 대기 하는 문제가 있다.

이 논문에서는 이러한 대기발생을 해결할 수 있는 동시 양방향 동기화 프로토콜을 제안한다. 동시 양방향 동기화 프로토콜은 다중 큐를 사용하여 변경충돌이 발생하지 않는 동기화 작업들 간에 동시 처리를 지원한다. 제안한 프로토콜을 사용하면, 처리시간이 긴 동기화를 처리 중에도 변경충돌이 발생하지 않는 다른 동기화 작업을

대기 없이 동기화 할 수 있다.

II. 관련연구

2.1 Exchange ActiveSync

Exchange ActiveSync[2]는 대기 시간이 길고 대역폭이 낮은 네트워크에서 작동하도록 최적화된 Microsoft Exchange 동기화 프로토콜이다.

HTTP와 XML을 기반으로 한 이 프로토콜은 브라우저 기능이 있는 휴대폰이나 Microsoft Windows Mobile® 탑재 장치 같은 모바일 장치에서 Microsoft Exchange를 실행하는 서버에 있는 조직의 정보에 액세스하는 데 사용된다. 모바일 장치 사용자는 Exchange ActiveSync를 통해 전자 메일, 일정, 연락처, 작업 등에 액세스할 수 있으며 오프라인 작업 시에도 이 정보를 사용할 수 있다.

2.2 ActMAP

ActMAP[3] 시스템은 유럽 자동차 회사들과 디지털 지도 제작 업체와의 프로젝트 팀에 의해 2007년도에 제안된 네비게이션 지원 시스템으로 자동차에 탑재된 네비게이션 기기에 무선 네트워크를 이용하여 지도의 최신 변경사항을 반영할 수 있도록 지원하는 시스템이다.

ActMAP은 오프라인 방식과 온라인 방식의 변경을 모두 지원한다. 오프라인 방식의 변경은 기존의 방식을 따르지만 무선 네트워크를 이용한 온라인 방식은 여러 제약조건을 고려한다.

첫 번째는 무선 네트워크에서 야기되는 제한된 대역폭이다. ActMAP은 제한된 대역폭 환경에서 전달되는 변경 지도 데이터의 양을 줄이기 위하여 지도 데이터를 계층(Layer)과 분할(Partition)로 구분하여 관리한다. 그리고 변경사항이 발생한 계층의 분할에 포함된 지도 데이터만을 전송하여 변경하는 점진적 부분 변경 방식을 사용한다.

두 번째는 다수의 변경 데이터 제공자와 수요자가 사용하는 고유 데이터 형식이다. 각 제공자와 수요자가 고유 데이터 형식을 사용하기 때문에 호환성이 떨어지고 모든 형식에 대하여 다수의 변환기가 필요한 문제가 있다. ActMAP은 이를 해결하기 위하여 GDF 데이터 모델에 기반하여 XML을 이용한 표준 데이터 형식을 제공한다.

세 번째는 변경이 반영되기까지 소요되는 지연으로 인하여 변경이 발생한 현장과 변경 내용들이 반영되는 지도 제공자들, 그리고 최종 네비게이션 시스템의 데이터 시간이 각각 다른 문제가 있다. 이를 위하여 ActMAP은 변경 전략을 제시하고 각각의 경우에 대한 시간 모델과 이를 기반으로 한 변경 방법을 제공한다.

마지막으로 지도 데이터가 변경됨에 따라 발생하는 변경 데이터의 품질 유지 문제이다. 이를 위하여 ActMAP은 변경을 위한 트랜잭션 개념을 제공하고 변경 데이터간의 계층·분할 의존성을 고려한다.

III. 기존 동기화 프로토콜의 문제점

기존의 동기화 프로토콜은 동기화를 요청한 순서대로 동기화를 처리한다. 이러한 방식으로 동기화를 하게 되면 다음과 같은 문제가 발생한다.

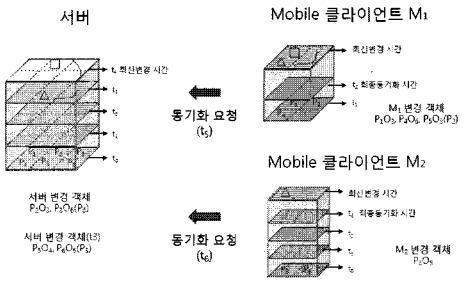

그림 1) M1과 M2가 동기화요청

예를 들어 그림 1)을 보면 M1은 t2시간이후 3개의 변경데이터에 대해서 동기화를 요청하였고, M2는 t3시간이후 1개의 변경데이터에 대하여 동기화를 요청하였다. M1과 M2를 비교하여 보면 M1이 M2보다 동기화 하는데 더 많은 시간이 걸릴 것이다.

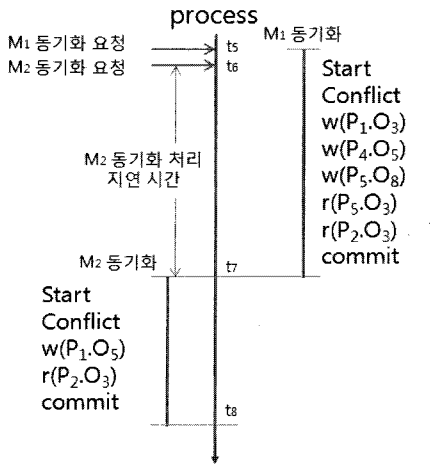

그림 2) M1과 M2의 동기화 처리 시간모델

그림 2)는 그림 1)의 동기화 요청을 기존의 동기화 프로토콜을 이용하여 순차적으로 동기화 하였을 때의 프로세스 처리의 시간 모델이다.

M2는 t6시간에 동기화를 요청하였지만 M1이 동기화가 끝나는 t7 시간까지 대기하였다가 동기화를 시작하였다. M2는 t6에서 t7사이의 시간을 낭비한 결과가 된다. 만약 M1의 처리시간이 사용자가 불편을 느끼지 못할 만큼 짧다면 문제가 없다. 그러나 고용량의 시공간데이터의 특성상 처리시간이 긴 동기화 작업이 될 수도 있다. 이러한 긴 처리시간으로 인해 동기화를 요청한 다른 클라이언트들이 오랜 시간 동기화를 못하고 대기하는 문제가 발생한다[4].

Ⅳ. 동시 양방향 동기화 프로토콜

이 절에서 사용되는 표기는 다음과 같다.

- 클라이언트 변경데이터(Temporary Delta recOrd Set, TDOS)
  - 클라이언트에서 변경한 시공간데이터들의 집합
- 서버 변경데이터(Delta recOrd Set, TDOS)
  - 서버에서 클라이언트로 동기화할 시공간데이터들의 집합
- 최종갱신시간(Last Update Time, LUT)
  - 서버 파티션의 최종갱신 시간
- 최종동기화시간(Last Sync Time, LST)
  - 클라이언트의 파티션이 서버와 최종으로 동기화한 시간
- PDOS(P_i)
  - 서버에서 클라이언트 파티션 P_i로 갱신할 객체들의 집합
- PTDOS(P_i)
  - 클라이언트에서 서버로 갱신할 파티션 P_i의 객체들의 집합
- TDOS(M_i)
  - 클라이언트 M_i에서 서버로 갱신할 객체들의 집합
- QS(Queue State)
  - 큐의 상태 값을 가진다. 큐가 비어있으면 0, 큐에 데이터가 있을 경우는 데이터가 있는 큐의 개수에 따라 1, 2, 3으로 표현
- CR(Copy Region)
  - 클라이언트가 서버로부터 전송 받은 시공간DB의 영역정보

4.1 동시 동기화 개요

그림 4)는 M1과 M2가 비슷한 시간에 동기화를 요청하였을 때 동시 동기화의 개요이다.

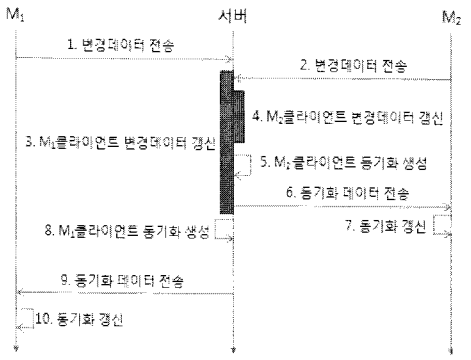

그림 4) 동시 동기화 개요

1. M1에서 자신의 TDOS를 서버로 동기화 하기 위해 전송한다.
2. M2에서 자신의 TDOS를 서버로 동기화 하기 위해 전송한다.
3. M1부터 전송받은 TDOS를 서버의 시공간데이터베이스에 갱신하는 과정으로 TDOS의 변경충돌 검사를 하고 충돌이 없는 경우 다중 큐에 TDOS를 삽입하여 시공간데이터베이스에 갱신한다.
4. 3번과 같은 작업과정으로 M2의 TDOS를 서버의 시공간데이터베이스에 갱신하는 과정이다. 이때는 다중 큐에 삽입하기 전에 M1의 TDOS와 변경충돌이 발생하는지도 검사한다. 변경충돌이 없다면 다중 큐에 삽입하여 동기화를 진행하면 M1 동기화 작업과 라운드 로빈 방식으로 병렬 처리한다.
5. 먼저 TDOS의 갱신이 완료된 M2로 동기화할 서버의 DOS를 생성한다.
6. 서버에서 M2로 DOS를 전송한다.
7. M2는 전송받은 DOS를 모바일 시공간데이터베이스에 갱신하여 동기화를 완료한다.
8. TDOS의 갱신이 완료된 M1로 동기화할 DOS를 생성한다.
9. 서버에서 M1로 DOS를 전송한다.
10. M1은 전송받은 DOS를 모바일 시공간데이터베이스에 갱신하여 동기화를 완료한다.

4.2 변경충돌검사

동시 양방향 동기화 프로토콜의 변경충돌검사는 클라이언트 서버 구조의 시스템에서 동기화 작업을 위한 변경충돌검사 방식으로 그림 5)에서 보여주는 형태로 변경충돌검사를 한다.

대상	파티션 단위	객체단위	변경충돌
서버 vs 클라이언트	$LUT(P_i) \le LST(P_i)$		X
	$LUT(P_i) > LST(P_i)$	$PDOS(P_i) \cap PTDOS(P_i) = 0$	X
		$PDOS(P_i) \cap PTDOS(P_i) \neq 0$	O
클라이언트 vs 클라이언트		$TDOS(M_1) \cap TDOS(M_2) = 0$	X
		$TDOS(M_1) \cap TDOS(M_2) \neq 0$	O

그림 5) 변경충돌검사

서버와 클라이언트의 변경충돌검사는 기존의 데이터베이스시스템에서 데이터를 갱신할 때의 구조와 거의 동일한 구조이다. 여기에 분할 단위의 LUT와 LST를 이용한 검사가 추가되어 분할 단위에서 변경충돌 발생 여부를 먼저 판단하고 변경충돌이 발생 가능한 분할에 대해서 객체단위의 변경충돌검사를 하여 변경충돌검사의 작업량을 줄일 수 있는 구조이다. 클라이언트와 클라이언트 사이의 변경충돌검사는 동시 동기화 처리를 하는 경우에 동기화하는 클라이언트들 간의 변경충돌여부를 검사하는 것이다.

4.3 다중 큐

다중 큐는 동시 양방향 동기화를 위한 핵심기술로서 변경충돌이 없는 동기화 작업을 다수의 큐에 각각 삽입하여 라운드 로빈 형태로 동기화를 진행하기위한 일종의 버퍼이다. 큐의 각 노드의 단위는 객체단위의 연산 작업이다.

큐의 상태	중복영역	중복 큐의 수	삽입방법
QS = 0		0	비어있는 큐에 삽입
QS = 1	CR(Q) ∩ CR (M) = 0	1	중복이 발생한 큐에 삽입
	CR(Q) ∩ CR (M) ≠ 0	0	비어있는 큐에 삽입
QS = 2	CR(Q) ∩ CR (M) = 0	1	중복이 발생한 큐에 삽입
		2	데이터가 많은 큐에 삽입
	CR(Q) ∩ CR (M) ≠ 0	0	비어있는 큐에 삽입
QS = 3	CR(Q) ∩ CR (M) = 0	1	중복이 발생한 큐에 삽입
		2	데이터가 많은 큐에 삽입
		3	데이터가 가장 많은 큐에 삽입
	CR(Q) ∩ CR (M) ≠ 0	0	데이터가 가장 적은 큐에 삽입

그림 6) 다중 큐 삽입 프로토콜

그림 6)는 다중 큐의 큐 개수가 3개인 경우의 삽입프로토콜이다. QS = 0 은 모든 큐가 비어있을 말하고 1, 2, 3 은 각각 연산 작업이 있는 큐의 개수이다. 3은 모든 큐에 연산 작업이 있음을 의미한다. 중복영역검사는 서로 다른 클라이언트가 동시에 동기화 할 때 직렬성을 보장하기위한 검사이다. 직렬성 보장은 중복영역이 있는 M_1과 M_2 클라이언트가 동기화를 요청하였을 때 M_1이 먼저 동기화를 요청하였다면, 중복된 영역의 갱신작업에 대해서 M_1이 먼저 갱신되도록 보장하는 것이다.

규 삽입 알고리즘

enqueue
input : TDOS, CR, MQHead
output : BOOL
MQHead (다중 큐의 헤더)
TQHead (TDOS를 삽입할 큐의 헤더)
insert() : 다중 큐에 TDOS를 삽입한다.
compare(): 다중 큐에서 동기화중인 작업과 다중 큐에 삽입할 작업사이의 중복영역을 검사 후 입력할 큐의 헤더를 리턴

```
1. TQHead : QNODE;
2. if QS := 0 then
3. insert(TDOS, MQHead)
4. else
5. TQHead := compare(CR)
6. insert(tdos, TQHead)
```

큐 삭제 알고리즘

dequeue
input : MQHead
output : QNODE
QCount (다중 큐의 큐 개수)

```
1. QCount : int;
2. for i := 0 to Qcount do
3. if MQHead[i] != NULL then
4. return MQHead[i].node;
```

V. 결 론

이 논문에서는 처리시간이 긴 동기화 작업으로 인한 대기발생 문제점을 해결하기위해 다중 큐를 이용한 동시 양방향 동기화 프로토콜을 제안 하였다. 향후 실험을 통하여 실제 처리시간의 효율성에 대한 검토가 필요하면 서비스 환경에 따른 효율적인 다중 큐의 개수에 관한 연구가 필요할 것이다.

참고문헌

[1] 류석상, "디지털 컨버전스로 나타나는 유비쿼터스 사회,"한국전산원 u-전략팀, 2005.9
[2] Exchange ActiveSync , "http://technet.microsoft.com/ko-kr/library/aa998357(EXCHG.80).aspx"
[3] "ActMAP White Paper and Interfaces to the FeedMAP framework," white paper, 2007
[4] 최우영, 이경아, 염태진, 진성일, "내장형 DBMS를 위한 동기화 서버 시스템" 한국정보과학회, 한국정보과학회 학술발표논문집 한국정보과학회 2004년도 봄 학술발표논문집 제31권 제1호(B), 2004. 4, pp. 127 ~ 129