무선 데이타 방송 환경에서 읽기-전용 트랜잭션 처리 기법

(Read-only Transaction Processing in Wireless Data Broadcast Environments)

이 상 근 † 김 성 석 †† 황 종 선 †††
(SangKeun Lee) (SungSuk Kim) (Chong-Sun Hwang)

요 약 본 논문에서는, 무선 데이타 방송 환경에서 여러 데이타 항목을 지정된 순서에 의해 접근해야 하는 읽기-전용 트랜잭션의 일관성 유지와 관련된 주제를 다룬다. 데이타 방송 환경에서 사용자는 항상 순차적으로 데이타에 접근하게 된다. 이러한 속성을 가진 환경에서, 본 논문은 기선언-기반 질의 최적화 방식을 제안하며, 이를 이용하여 지역 캐쉬와 관련된 두 가지 트랜잭션 실행 기법을 개발하였다. 또한 제안된 기법들을 수학적으로 분석함으로써 성능을 평가하였다. 평가 결과에 의해, 본 논문에서 제안한 기선언 방식이 응답 시간을 상당히 단축시켰으며, 작업량이 동적으로 변하더라도 훨씬 더 잘 적응될 수 있음을 알 수 있다.

키워드 : 이동 컴퓨팅, 무선 데이타 방송, 트랜잭션 처리

Abstract In this paper, we address the issue of ensuring consistency of multiple data items requested in a certain order by read-only transactions in a wireless data broadcast environment. To handle the inherent property in a data broadcast environment that data can only be accessed strictly sequential by users, we explore a predeclaration-based query optimization and devise two practical transaction processing methods in the context of local caching. We also evaluate the performance of the proposed methods by an analytical study. Evaluation results show that the predeclaration technique we introduce reduces response time significantly and adapts to dynamic changes in workload.

Key words : Mobile computing, Wireless data broadcast, Transaction management

1. 서 론

무선 방송 환경에서 이동 클라이언트들에게 정보를 효율적으로 전송하기 위한 메커니즘과 관련된 연구가 많이 진행되고 있다. 예를 들면, 그러한 메커니즘은 위성이나 기지국에 의해 공통된 관심을 가진 이동 사용자들에게 정보를 전송하기 위해 사용될 수 있다. 방송-기반 전송은 다수의 클라이언트들에게 정보를 전파하는 것을 포함한 여러 응용에 적용될 수 있다는 점에서 그 의미가 있다. 방송-기반 응용에는 주식, 스포츠 티켓, 전자 우편물, 메일링 리스트, 전자 은행 업무, 교통 정보 시스템 등이 있다. 그러한 응용에서 한 클라이언트가 어떤 데이타 항목을 필요로 한다면, 클라이언트는 서버가 데이타들을 전송하고 있는 동안 대기 중에서 필요한 데이타를 얻을 수 있다. 데이타를 방송하는 것은 동일한 데이타를 기다리고 있는 여러 클라이언트의 요구를 동시에 만족시켜줄 수 있으므로, 데이타를 전파하기 위한 비용은 클라이언트의 수에 영향을 받지 않으며, 결과적으로 대역폭을 효율적으로 사용할 수 있게 된다. 그러므로, 대역폭을 효율적으로 사용해야할 경우, 많은 정보 및 데이타들을 전파하는 것이 매우 적합한 방법이 될 수 있다.

무선 데이타 방송 환경에서 실행되는 트랜잭션에게 일관된 값을 제공하는 것은 주요한 이슈들 중 하나로 인식되고 있다[1,2,3]. 그러한 환경에서, 트랜잭션은 필요한 데이타를 얻기 위해, 서버에게 필요한 정보를 보내

† 비 회 원 : LG전자 단말연구소 선임연구원
yalphy@lge.com
†† 학생회원 : 고려대학교 컴퓨터학과
sskim@disys.korea.ac.kr
††† 종신회원 : 고려대학교 컴퓨터학과 교수
hwang@disys.korea.ac.kr
논문접수 : 2001년 12월 13일
심사완료 : 2002년 7월 10일

거나 혹은 록(lock)을 설정할 필요가 없다. 즉, 전파되고 있는 데이타를 공기(air) 중에서 얻을 수 있다. 만약 그 순간 서버에서 데이타가 갱신될 경우, 트랜잭션은 비일관적인 데이타에 접근하게 된다. 이전의 연구에서, 데이타 갱신으로 인해 발생하는 일관성 문제를 적절하게 해결하기 위해 일관성 기준을 약화시키기도 하였다(예를 들면, [4]에서의 갱신 일관성(update consistency)). 이와 같은 접근방식은 직렬화가능성(serializability)을 유지하는 것이 비대칭적 통신 환경에서는 상당히 많은 부하를 초래한다는 가정을 기반으로 한다. 그러나, 본 논문에서는 다음과 같은 이유로 여전히 직렬화가능성을 일관성 기준으로 채택하였다:

• 본 논문은 제안한 동시성 제어 기법에서 직렬화가능성을 채택하더라도 큰 비용이 초래하지 않음을 발견하였다. 이는 방송 환경에서 직렬화가능성을 유지하기 위한 프로토콜에서는 많은 단점이 발생함으로써 낮은 성능을 보인다는 이전의 연구결과[4]에 반하는 결과이다. 5장의 성능 평가는 이러한 우리의 주장을 뒷받침한다.

• 클라이언트의 일관성 요구조건이 엄격하지 않아서 직렬화가능성보다 약화된 일관성 조건에서 훨씬 더 효율적으로 동작할 경우, 직렬화가능성을 유지하는 것은 상당히 부하가 크다고 생각된다. 그러나, 데이타 최신성(currency)을 약화시키지 않고서는 일관성 조건을 강화시키기 쉽지 않다. 방송-기반 환경에서는 대부분의 고급 응용이 가능한 최신의 데이타를 접근하려고 한다는 점에서, 데이타 최신성을 약화시키지 않으면서 직렬화가능성을 효율적으로 유지하는 것은 매우 의미있는 점이다.

• 완화된 일관성 조건을 기반으로 한 프로토콜은 일부 응용에서는 효율적으로 동작할 수 있지만, 직렬화가능성은 여전히 여러 다른 응용에서 필수적인 조건이 된다. 예를 들면, 이동 주식 거래 시스템의 경우, 여러 주식들에 대하여 시간이 경과함에 따라 가격간의 관계를 알기 위해 매매 주문이 발생한다. 거래자 관점에서 볼 때, 직렬화가능성이 유지되지 않을 경우 금전적으로 중요한 손실이 발생할 수 있다. 즉 사용자가 여러 개의 읽기-전용 트랜잭션을 제시하여 그 결과를 비교한다면 부정확한 결과가 나올 수 있다[5].

본 논문에서의 주요한 목표는 처리 속도를 향상시키면서도 무선 트랜잭션에게 일관된 데이타 값을 제공하는 것이다. 이를 위해, 기선언-기반(predeclaration-based) 질의 최적화 기법을 지역 캐쉬 기법과 함께 개발한다. 기선언 기법은 이전의 풀(pull)-기반 (즉, 클라이언트-요구 기반) 데이타 전송 방식에서 록킹 기법의 교착상태를 방지하기 위해 개발되었었다[6]. 그러나 푸쉬(push)-기반 전송 방식에서는 트랜잭션 실행에 기선언 방식을 적용함으로써, 각 읽기-전용 트랜잭션이 한계가 있는(bounded) 최악의 실행시간 내에서 성공적으로 그 실행을 완료할 수 있다.

본 논문의 공헌도는 두 가지로 나뉘어 생각할 수 있다. 첫째, 이 연구는 무선 데이타 방송 환경에서 질의 최적화 기법을 적용하여 트랜잭션 응답시간을 상당히 줄인 첫 번째 시도이다. 이는 클라이언트가 비대칭적 통신 환경에서 트랜잭션 일관성을 유지하기 위해 보다 적극적인 역할을 수행하여야 한다는 철학에서 출발되었다. 둘째, 이 연구에서는 평균 응답시간(즉, 전체 시스템 성능)과 최악의 응답시간(즉, 개개의 성능)간의 균형을 유지시킬 수 있으며, 이는 무선 트랜잭션 실행에서 중요한 이슈이다[3]. 일반적인 상황에서, 본 논문에서 제안하는 메소드에서의 평균 응답시간과 최악의 응답시간 사이의 차이는 하나의 방송 주기 길이의 반 정도가 된다.

본 논문의 구성은 다음과 같다. 2장에서는 무선 데이타 방송 환경에서의 트랜잭션 실행을 위한 기본 설계 원칙을 설명하며, 3장에서는 지역 캐싱 기법과 연관된 두 가지의 기선언-기반 트랜잭션 실행 메소드를 제안한다. 본 논문에서 제안하는 메소드와 이전 기법들과의 비교를 위한 수학적 분석 모델 및 그 분석 결과는 4장과 5장에서 각각 보여준다. 6장에서는 접속단절을 다루는 기법 등에 대해 논의하며, 7장에서 결론을 맺는다.

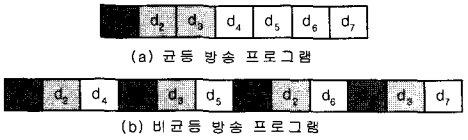

그림 1 방송 구성

2. 기본 설계 원칙

[7]과 유사한 시스템 모델에서, 그림 1에서 보여지는 두 가지 다른 방송 구성 방식을 생각해보자. 이 그림에서 서버는 하나의 방송 주기동안 데이타 항목 {d_1, d_2, d_3, d_4, d_5, d_6, d_7}들을 방송 프로그램(d_1이 가장 빈번하게 접근되고, d_2와 d_3은 다음으로 빈번하게 접근되며, d_4, d_5, d_6, d_7은 가장 덜 빈번하게 접근된다)에 따라 방송한다. 그림 1-(a)는 균등 방송이며, 그림 1-(b)는 비균등 방송 프로그램을 나타낸다(자세한 내용은 [8]을 참조).

각 방송 구성에서, 클라이언트 트랜잭션이 한 방송 주기의 중간에서 다음과 같은 연산을 실행하는 경우를

생각해보자:

$IF(d_3 \leq 3)$ $THEN$ $read(d_1)$ $ELSE$ $read(d_2)$

이 경우, 다음과 같은 두 가지 질문에 답을 하고자 한다.

- 트랜잭션을 실행하는데 걸리는 (데이타 항목의 개수 단위의) 응답시간은?
- 서버에서 데이타 항목의 값이 변경될 경우, 트랜잭션의 일관성은 어떻게 되는가?

다음의 소절에서는 위의 질문에 대한 답과 그 배경에 대해 설명한다.

2.1 기선언과 그 유용성

먼저, 트랜잭션이 데이타를 읽는 순서가 응답시간에 미치는 영향을 보이기 위해, 그림 1의 균등 방송 환경에서 클라이언트 트랜잭션 프로그램을 생각해보자. 클라이언트 입장에서 볼 때 d_1과 d_2는 d_3보다 먼저 보여지므로, 트랜잭션은 d_3을 읽은 후 d_1과 d_2 값이 전송될 때까지 기다려야 한다. 따라서 트랜잭션의 응답시간은 d_3과 d_1을 접근한 경우 11.5 (즉, $d_4 \rightarrow d_5 \rightarrow d_6 \rightarrow d_7 \rightarrow d_1 \rightarrow d_2 \rightarrow \underline{\boldsymbol{d_3}} \rightarrow d_4 \rightarrow d_5 \rightarrow d_6 \rightarrow d_7 \rightarrow \underline{\boldsymbol{d_1}}$)가 되며, d_3과 d_2를 접근한 경우에는 12.5 (즉, $d_4 \rightarrow d_5 \rightarrow d_6 \rightarrow d_7 \rightarrow d_1 \rightarrow d_2 \rightarrow \underline{\boldsymbol{d_3}} \rightarrow d_4 \rightarrow d_5 \rightarrow d_6 \rightarrow d_7 \rightarrow d_1 \rightarrow \underline{\boldsymbol{d_2}}$)가 된다. 만약 트랜잭션이 앞으로 접근할 가능성이 있는 모든 데이타 항목, $\{d_1, d_2, d_3\}$,을 미리 선언해 둔다면, 클라이언트는 필요한 모든 데이타를 미리 획득할 수 있으며, 따라서 응답시간은 6.5 (즉, $d_4 \rightarrow d_5 \rightarrow d_6 \rightarrow d_7 \rightarrow \underline{\boldsymbol{d_1}} \rightarrow \underline{\boldsymbol{d_2}} \rightarrow \underline{\boldsymbol{d_3}}$)로 줄일 수 있다. 이것은 그림 1의 비균등 방송 구성의 경우에서도 동일한 결과를 얻을 수 있다. 즉, 이전 방식의 경우, 트랜잭션의 응답시간은 d_3과 d_1을 접근한 경우 7 (즉, $d_1 \rightarrow d_2 \rightarrow d_6 \rightarrow d_1 \rightarrow \underline{\boldsymbol{d_3}} \rightarrow d_7 \rightarrow \underline{\boldsymbol{d_1}}$)이 되며, d_3과 d_2를 접근한 경우 8 (즉, $d_1 \rightarrow d_2 \rightarrow d_6 \rightarrow d_1 \rightarrow \underline{\boldsymbol{d_3}} \rightarrow d_7 \rightarrow d_1 \rightarrow \underline{\boldsymbol{d_2}}$)이 되지만, 기선언방식을 사용하면 5 (즉, $\underline{\boldsymbol{d_1}} \rightarrow \underline{\boldsymbol{d_2}} \rightarrow d_6 \rightarrow d_1 \rightarrow \underline{\boldsymbol{d_3}}$)로 줄일 수 있다. 따라서 기선언 방식을 사용한다면, 데이타에 대한 요구가 발생하는 순서대로 얻는 것이 아니라, 필요한 데이타가 방송되는 순서대로 얻을 수 있게 된다.

2.2 데이타 수정과 트랜잭션 일관성

각 주기의 방송은 데이타베이스의 일관된 값을 가지고 있다 (이러한 요구조건은 [7,9]에서도 가정하고 있다). 따라서 읽기-전용 트랜잭션이 단일 주기동안의 방송에서 필요한 모든 데이타 항목을 읽는다면, 그 트랜잭션은 어떠한 동시성 제어 실행 없이도 완료할 수 있다. 그러나, 현실적으로 대부분의 트랜잭션은 한 방송 주기 중간에 실행을 시작하며, 따라서 여러 방송 주기동안 필요한 데이타를 접근해야한다. 그러한 경우, 읽은 값이 일관성이 있다고 보장할 수 없다. 이러한 상황은 기선언-기반 데이타 접근 방식을 사용하더라도 동일한 결과가 발생한다. 이를 위하여, 필요한 데이타를 얻는 과정과 실제 트랜잭션이 데이타를 읽는 과정을 분리함으로써, 하나의 주기(동기 방식) 혹은 두 주기(비동기 방식)내의 일관된 상태의 데이타베이스의 부분집합에 접근하도록 할 수 있다.

3. 제안하는 기법

이 절에서는 지역 캐쉬 기법과 연관지어, 두 가지 기선언-기반 트랜잭션 처리 메소드를 제안한다: PA (Predeclaration with Autoprefetching)과 PA^2(PA/Asynchronous). 기본 아이디어는 트랜잭션이 데이타를 읽는 방송 주기의 개수를 줄이기 위해 읽기집합(Readset)의 기선언 기법을 채택하는 것이다. 트랜잭션에 대한 읽기집합은 클라이언트 측에서 클라이언트 시스템에 트랜잭션을 제기하기 전에 그 트랜잭션을 분석함으로써 얻을 수 있다 ([7]에서와 마찬가지로, 본 논문에서는 데이타 항목의 식별자를 통해서만 트랜잭션이 데이타 읽기 연산을 수행한다고 가정한다. 따라서, 트랜잭션이 실제로 시스템에 제기되기 이전에 전처리기(preprocessor)등을 통해 읽기집합을 얻을 수 있다. 물론 이 작업을 위해 클라이언트 측에서는 어느 정도의 전력 소모가 있을 것이다).

3.1 캐싱과 무효화 비트 패턴

클라이언트는 접근 지연을 줄이기 위해 관심있는 일부 데이타를 지역 저장공간에 캐쉬해둔다. 이러한 캐싱으로 인하여 트랜잭션은 필요한 데이타를 지역적으로 얻을 수 있기 때문에 훨씬 적은 회수만큼 방송 채널에 접근해도 되므로, 트랜잭션 실행을 위한 지연을 감소시킬 수 있다. 이 논문에서, 클라이언트는 캐쉬 공간으로써 자신의 하드디스크를 사용하며, 트랜잭션 실행을 지원할 수 있는 캐쉬 기법이 적용된다고 가정한다. 그러므로 트랜잭션 실행 중에 캐쉬 데이타가 생성되고 삭제되더라도 그 트랜잭션의 일관성이 침해되어서는 안된다.

서버에서 데이타 갱신이 발생할 경우, 캐쉬의 데이타는 비일관된 값을 가지게 된다. 무선 데이타 방송 환경에서, 클라이언트는 지역 캐쉬로부터 필요한 데이타에 접근하는 반면, 서버에서는 갱신된 데이타 항목에 대한 정보가 수집되고 있다. 이러한 상황에서 지역 캐쉬 데이타가 일관된 값을 가지도록 하기 위해, 비일관된 값을 가지는 클라이언트 캐쉬의 데이타 항목은 무효화되거나 최신 값을 갖도록 갱신되어야 한다. 클라이언트에게 데

이타 갱신 사실을 알리기 위한 다양한 기법들 중에서, 무효화보고(*invalidation report*) 기법[10,11]이 있다. 이 기법에서는 클라이언트 캐쉬 일관성은, 서버에서 최근에 갱신된 데이타 항목들의 목록을 담은 주기적인 무효화보고를 받음으로써 효율적으로 유지될 수 있다. 그러나 갱신된 항목의 식별자만을 전송하더라도 이는 방송 채널- 이 자원은 상당히 불충분하다 -의 상당 부분을 차지하게 되며, 특히 데이타 항목이 빈번하게 갱신될 경우에는 문제가 심각해진다. 게다가, 직렬화가능성 조건을 채택할 경우, 트랜잭션이 여러 데이타 항목을 읽더라도 일관성이 유지되어야 한다.

지역 캐쉬와 연관지어 실용적인 트랜잭션 처리 기법을 위해, 본 논문에서는, 서버는 방송 내용을 전송하기 전에 먼저 무효화 비트 패턴(*invalidation bit pattern*)을 전파하도록 한다. 무효화 비트 패턴의 경우, 각 비트는 데이타베이스의 단일 데이타 항목에 대응된다(본 논문에서는 방송 채널내에서 각 데이타 항목의 위치는 고정되어 있다고 가정한다). 만약 해당하는 데이타 항목이 서버에서 이전 방송 주기동안 갱신되었으며 그 주기동안 갱신된 값이 전송되지 않았다면, 그 비트는 1로 설정된다. 나머지 비트들은 0으로 설정되어 있다. 이와 같은 방식은 이전 무효화보고 기법과 비교해볼 때, 특히 상당수의 데이타 항목이 갱신되는 환경의 경우, 서버가 전송하는 무효화 정보의 크기를 상당히 줄일 수 있다. 게다가 직렬화가능한 방송 내용이 각 주기동안 전파된다면, 트랜잭션 캐쉬 일관성 유지도 용이해진다. 따라서 우리는 다음과 같은 속성이 항상 지켜진다고 가정한다.

서버 요구사항 : 각 방송 주기에서 서버는 무효화 비트 패턴을 먼저 전파하고, 그런 다음 완료된 트랜잭션이 갱신한 "직렬화가능한" 데이타 항목의 값들을 전송한다.

각 방송 주기를 시작할 때, 클라이언트는 무효화 비트 패턴을 얻게된다. 각 캐쉬의 데이타 항목 d_i에 대하여, 무효화 비트 패턴에서 해당하는 비트가 1로 설정되었다면, 그 캐쉬 데이타는 "무효화"로 설정한 후, 현재 방송 데이타로부터 해당하는 값을 다시 얻어서 캐쉬에 저장시킨다. 즉, 제안하는 기법에서의 캐쉬 관리 기법은 선반입 (*autoprefetching*) 기법[10]과 결합된 무효화 방식과 동일하다. 특히, 무효화된 데이타 항목에 대하여 다음 주기의 방송 내용에서 전송되면 새로운 값으로 교체한다.

3.2 *PA* 및 PA^2 메소드

먼저, 기선언된 트랜잭션 T의 읽기 집합, $Pre\_RS(T)$, 은 T가 잠재적으로 읽을 수 있는 데이타 항목들의 집합으로 정의한다. 모든 메소드에서, 각 클라이언트는 T를 3 단계로 실행한다: (1) 전처리 단계 : $Pre\_RS(T)$ 얻기, (2) 획득 단계 : 방송 내용 혹은 지역 캐쉬로 부터 $Pre_RS(T)$에 해당하는 데이타 항목의 값을 얻는다. 이 단계 동안, 클라이언트는 지금까지 얻어진 데이타를 $Acquire(T)$에 저장해둔다. (3) 전송 단계 : 트랜잭션이 필요로 하는 데이타를 요구하는 순서대로 전송해준다.

3.1 절에 나타난 서버에 대한 요구사항을 기반으로 하고, 각 클라이언트가 단일 주기동안 필요한 모든 데이타를 얻는다면, 각 읽기-전용 트랜잭션의 실행은 직렬화가능성을 보장받게된다. 그러나, 트랜잭션은 하나의 방송 주기가 시작한 어느 시점에서 실행을 시작할 수 있기 때문에 획득 단계는 하나 이상의 방송 주기에 걸쳐 진행될 수 있다. 이 문제를 해결하기 위해, *PA* 메소드에서는 클라이언트가 획득 단계를 동기 방식으로 시작한다. 즉 다음 방송 주기가 시작할 때 획득 단계를 시작한다. 트랜잭션이 잠재적으로 필요로 하는 모든 데이타 항목이 미리 선언되어 있으므로, 클라이언트는 하나의 방송 주기동안 획득 단계를 마칠 수 있다. 보다 명확하게 하기 위해, 다음 알고리즘 1에 의해 트랜잭션 T_i를 실행한다.

```
1.Begin(Ti)를 받았을때 {
  전처리기에 의해 Pre_RS(Ti)를 얻는다 ;
  Acquire(Ti) = Φ ;
  다음 방송 주기가 시작할 때까지 기다린다 ;
  }
2.무효화 비트 패턴을 읽는다;
  지역 캐쉬의 모든 항목 di에 대하여 {
     if(해당하는 무효화 비트가 1로 설정) {
        di를 "무효화"로 설정한다 ;  }
  }
  지역 캐쉬의 모든 "유효"한 항목 di에 대하여 {
    if(di∈Pre_RS(Ti)) { Acquire(Ti) ← di ; }
  }
  While (Pre_RS(Ti)≠Acquire(Ti)) {
     for any (지역 캐쉬내의 "무효화" 항목 dk) OR
          any (dj∈Pre_RS(Ti) - Acquire(Ti)) {
       방송 채널을 튜닝하다가 dk 혹은 dj를 읽는다 ;
       if (dk가 얻어질 경우) { 지역 캐쉬내의 값을 수정한다 ; }
       if (dj가 얻어질 경우) {
          dj를 지역 캐쉬에 삽입한다 ; Acquire(Ti)←dj ; }
     }
  }
3. Ti가 요청한 순서대로 데이타 항목을 Ti에게 넘겨준 후, Ti를 완료시킨다.
```

알고리즘 1 *PA* 메소드

*PA*의 주요한 장점은, 갱신이 빈번하게 발생하는 환경에서도 직렬화가능성을 유지하면서 트랜잭션의 실행

시간을 상당히 감소시킬 수 있다는 점이다. 특히, 이전 주기동안 데이타베이스의 모든 데이타 항목이 갱신되는 극단적인 경우에서도 최대 두 주기 내에서 각 트랜잭션은 완료할 수 있다. 그러나 PA의 단점은, 데이타베이스에서 갱신이 아주 드물게 발생하는 환경에서 짧은 트랜잭션(short transactions)을 실행할 때 불필요한 지연이 발생할 수 있다는 것이다. 예를 들면, 대부분의 필요한 데이타가 캐쉬에 유지되어 있으며 캐쉬 미스(cache miss)된 데이타가 현재의 방송 주기 내에서 얻을 수 있다면, 트랜잭션은 다음 주기까지 기다릴 필요가 없다.

PA의 이러한 단점을 해결하기 위해, 클라이언트는 비동기 방식으로 실행할 수 있다. 즉, 다음 주기가 시작할 때까지 기다리는 대신 필요한 데이타 항목을 즉시 얻도록 한다. 동기 방식과 달리, 이 경우에는 획득 단계가 두 주기에 걸쳐질 수 있음을 주의해야 한다. 본 논문에서는 이 메소드를 PA^2라고 하며, 다음 알고리즘 2처럼 동작한다.

---

1. *Begin*(T_i)를 받았을때 {
   전처리기에 의해 $Pre_RS(T_i)$를 얻는다 ;
   $Acquire(T_i) = \Phi$;
   }
2. 캐쉬의 모든 "유효한" 데이타 항목 d_i에 대하여 {
   if ($d_i \in Pre\_RS(T_i)$) {$Acquire(T_i) \leftarrow d_i$;}
   }
   While ($Pre\_RS(T_i) \neq Acquire(T_i)$) {
   for any (지역 캐쉬내의 "무효화" 항목 d_k) OR
   any ($d_j \in Pre\_RS(T_i) - Acquire(T_i)$) {
   방송 채널을 튜닝하다가 d_k 혹은 d_j를 읽는다 ;
   if (d_k가 얻어질 경우) { 지역 캐쉬의 값을 수정한다 ; }
   if (d_j가 얻어질 경우) {
   d_j를 지역 캐쉬에 삽입한다 ; $Acquire(T_i) \leftarrow d_j$; }
   }
   if (무효화 비트 패턴이 전송될 경우) {
   무효화 비트 패턴을 읽는다 ;
   for (지역 캐쉬의 모든 항목 d_i) {
   if (해당하는 무효화 비트가 1로 설정된 경우) {
   d_i를 "무효화"로 설정한다 ;
   $Acquire(T_i) = Acquire(T_i) - \{d_i\}$; }
   }
   }
   }
3. T_i가 요청한 순서대로 데이타 항목을 T_i에게 넘겨준 후, T_i를 완료시킨다.

알고리즘 2 PA^2 메소드

PA와 PA^2 메소드 모두 서버에는 큰 부하를 주지 않음을 주지하여야 한다. 서버에 대한 유일한 부하는 직렬화가능한 데이타 값을 전송하도록 하는 것과 각 주기마다 무효화 비트 패턴을 전송하도록 하는 것이다.

4. 분 석

이 절에서는 기선언-기반 트랜잭션 처리 메소드와 [7]에서 제안된 두 메소드를 비교하기 위한 분석 모델을 설명한다. 아래에 비교되는 두 가지 메소드를 간략하게 설명한다.

선반입 기반 무효화(Invalidation with Auto-prefetching; *IA*) : 이 기법에서는, 직렬화가능성을 유지하기 위하여 다른 데이타베이스 상태에 해당하는 데이타 항목을 접근한 트랜잭션을 무효화(철회)시킨다. 이를 위하여, 서버는 3.1절에서 설명한 무효화 비트 패턴을 전파하며, 각 클라이언트는 각 데이타 항목을 캐쉬 단위로 하여 관심 데이타를 지역 캐쉬에 유지한다. 캐쉬 관리 정책으로는 선반입 형태의 무효화 정책을 사용한다. 또한, 각 클라이언트는 실행중인 트랜잭션 T를 위하여 $RS(T)$를 유지하는데, 이것은 지금까지 읽은 데이타 항목을 저장해둔다. 각 클라이언트는 각 방송 주기의 시작시점에 전송되는 무효화 비트 패턴을 얻어서 캐쉬 데이타의 일관성을 유지한다. 그리고 트랜잭션 T의 하나 이상의 데이타 항목 $d_i \in RS(T)$이 무효화된다면, 즉 그 데이타 항목이 갱신되었다면, 그 트랜잭션은 철회된다.

선반입 기반 다중버전(Multiversion with Auto-prefetching; *MA*) : MA 메소드의 경우, 서버는 각 데이타 항목에 대하여 가장 최근에 완료된 값만 유지하면서 전송하는 것이 아니라, 여러 버전을 관리한다. 버전이란 각 방송 주기의 시작 시점에 다른 값에 해당하며, 버전의 개수는 방송 주기에 해당한다. 각 클라이언트에서, 트랜잭션 T는 첫 읽기 연산에 대하여 가장 최근에 완료한 버전 v_0을 읽게 한다. 이후의 연산에 대하여, T는 v_0보다 작거나 같은 버전의 값을 읽어야 한다. 만약 그러한 버전이 없다면, T를 철회시킨다. 또한, 각 클라이언트는 캐쉬를 유지하며, 캐쉬 일관성과 관련하여 선반입 기반 무효화 방식을 이용한다. 다중버전 기법을 지원하기 위해, 캐쉬의 데이타 항목 역시 버전 번호를 유지하여야 한다. 캐쉬 데이타를 읽으려고 할 경우에도, 방송에서 데이타를 읽을 경우와 동일한 검사 과정을 실행하여야 한다. 캐쉬 항목의 일관성 유지를 위해 서버는 3.1절에서 설명한 무효화 비트 패턴을 전송한다.

이제부터는 예상 평균 응답시간을 나타내는 기본 수식을 유도한다. 응답 시간의 단위로는 서버가 방송하는 데이타 항목의 개수를 사용할 것이다. 먼저, 모델에 대하여 다음과 같은 가정을 설정한다:

- 데이타베이스에서는 D개의 동일한 크기의 데이타 항목이 있다.

• 각 방송 주기는 그 주기의 시작 시점의 데이타베이스의 상태를 나타낸다.

• 각 방송 시작 시점에는 무효화 비트 패턴을 전송한다.

• 데이타 항목당 μ의 비율의 지수 분포에 따라 갱신이 발생한다.

• 각 이동 클라이언트는 높은 수준의 지역성을 가지는 D의 부분집합을 지속적으로 접근한다. 이러한 부분집합은 그 클라이언트에 대하여 "핫 스팟 (hot spot)"이 된다. 핫 스팟내의 각 데이타 항목은 γ의 비율로 접근된다.

이후의 분석을 간단히 하기 위해, 우리는 접속단절이 발생하는 가능성을 배제한다. 그리고 모든 메소드에서 무효화 비트 패턴의 크기를 무시한다. 이는 무효화 비트 패턴에서 데이타 항목 당 하나의 비트만이 할당되었으며, 따라서 그 전체 크기는 D 비트가 되며, 이는 단지 몇 개의 데이타 항목의 크기에 해당하기 때문이다.

4.1 방송 크기와 평균 적중률

4.1.1 방송 크기

비균등 방송에서, D개의 데이타 항목은 n 개의 파티션(partition)으로 나뉘어져 있으며, 각각의 파티션은 유사한 접근 빈도를 가진 데이타 항목으로 구성되어 있다. 높은 접근 확률을 가진 파티션은 낮은 접근 확률을 가진 파티션보다 더 빈번하게 방송된다. 파티션 i는 λ_i 번 전파된다고 가정한다 ($1 \le i \le n$). 또한 $0<i<j$인 경우, $\lambda_i > \lambda_j$ 이며, $\lambda_n=1$이 된다. λ은 모든 i에 대한 λ_i의 최소 공배수가 된다. [8]에서, i번째 파티션, P_i ($1 \le i \le n$), 는 c_i ($c_i = \lambda / \lambda_i$)개의 조각(chunk)으로 나뉘어 진다. 따라서 데이타 방송은 다양한 파티션 조각들을 번갈아 전송하는 방송 프로그램에 의해 구성된다. 방송 프로그램은 또한 동일한 크기의 세그먼트 열로 볼 수 있으며, 이 때 P_1은 모든 세그먼트에 나타난다. P_1 은 λ_1 번 전파되므로, λ_1 개의 세그먼트가 존재하며, 각 세그먼트는 $\sum_{i=1}^{n} \lambda_i |P_i| / \lambda_1$ 개의 항목을 가진다. 여기서 $|P_i|$ 는 파티션 i 에 있는 데이타 항목의 개수를 의미한다. $|NU|$ 를 비균등 방송에서 하나의 방송 주기내에 있는 데이타 항목의 개수라고 하면, 이는 다음과 같다.

$$|NU| = \sum_{i=1}^{n} \lambda_i |P_i| \qquad (1)$$

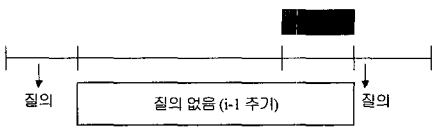

그림 2 캐쉬 적중률을 위한 시나리오

4.1.2 평균 적중률

캐쉬 적중률을 계산하기 위해 질의는 어느 특별한 순간에 발생한다고 가정하며, 그 값은 캐쉬에 있는 데이타가 유효할 조건부 확률로 계산한다. 하나의 방송 주기 내에 질의가 하나도 없을 확률 q_0 은 $e^{-\gamma |NU|}$ 이며, 한 주기동안 갱신이 하나도 없을 확률 u_0 은 $e^{-\mu |NU|}$ 가 된다. 두 개의 질의 사이에 i-1 주기가 경과되었다고 하자. 그림 2는 현재의 질의 이전의 가장 마지막 질의는 i 방송 주기 이전에 발생한 경우를 보여주고 있다. 선반입 기반 무효화 기법이 캐쉬 관리를 위해 사용되고 있으므로, 두 번째 질의가 캐쉬 적중되기 위해서는, 바로 이전 주기동안 어떤 갱신도 없으면 된다. 따라서 제안하는 기법들과 IA에서의 캐쉬 적중률에 대한 수식은 (2)와 같다.

$$h = (1-q_0) \sum_{i=1}^{\infty} q_0^{\,i-1} u_0 = u_0 \qquad (2)$$

4.2.2.절에서 설명할 MA에 대한 캐쉬 적중률은 위 수식 (2)와는 차이가 난다. 이는 이전 버전들을 포함하고 있으므로 한 주기의 방송의 크기가 증가하기 때문이다.

4.2 예상 평균 응답시간

비균등 방송 환경에서 한 트랜잭션에 대하여 a_s 와 a_t 를 단일 데이타 항목과 다중 데이타 항목을 접근하는 평균 응답시간이라고 하자. 비균등 방송 환경에서, 한 데이타 항목이 연속적으로 나타나는 시간 간격이 동일한 경우, 즉 각 데이타 항목에 대한 상호 도착시간이 고정되어 있을 경우에 a_s 는 최적 값을 나타낸다[12]. 각 데이타 항목에 대한 상호 도착시간이 고정되었을 때, 임의 시간에 요청이 도착하는데 대한 예상 평균 지연은 그 데이타 항목에 대한 방송 내에서의 간격의 반에 해당한다. 따라서 각 데이타 항목 $d_i \in D$ 에 대하여, d_i 의 평균 지연은 수식 (3)과 같다. 여기서 f_i는 d_i의 빈도수가 된다.

$$\omega(d_i) = \frac{|NU|}{2 f_i} \qquad (3)$$

어떤 데이타 요청에 대한 예상 a_s 는 각 데이타 항목에 대하여 접근 확률(본 논문에서는 $p(d_i)$ 로 표기한다)과 평균 지연 시간을 곱한 후, 이들의 합을 구하면 된다.

$$a_s = \sum_{d_i \in D} p(d_i)\, \omega(d_i) \qquad (4)$$

4.2.1 IA 메소드

IA 메소드에서, 클라이언트는 한번에-하나씩(one-at-a-time), 즉 방송으로부터 하나의 데이타 항목의 값을 얻은 후에만 다음 요청이 처리되는 방식으로 데이타 항목의 값을 얻는다. 먼저 서버에서 데이타 갱신이 없어서 항상 트랜잭션이 완료될 수 있는 경우를 생각해보자. 지

역 캐쉬 없이 m개의 데이타 항목을 접근하는 트랜잭션에 대한 평균 응답시간은 $m\sum_{d_i\in D}p(d_i)\omega(d_i)$가 된다. 따라서 IA의 평균 응답시간은 다음과 같이 계산된다.

$$E=m(1-h)\sum_{d_i\in D}p(d_i)\omega(d_i) \quad (5)$$

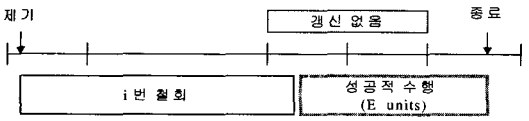

그림 3 IA 메소드를 위한 시나리오

그러나 서버에서 데이타가 갱신된다면, 트랜잭션은 성공적으로 완료될 때까지 여러 번 철회되고 다시 시작할 수도 있다. 그림 3은 성공적으로 완료하기 전에 i번 철회된 시나리오를 보여주고 있다. IA 메소드에 의한 트랜잭션 실행이 성공적으로 완료할 수 있는 확률 c 는 $e^{-\mu|NU|\lfloor\frac{E}{|NU|}\rfloor}$ 보다 작거나 같게 된다. 이는 클라이언트가 그 트랜잭션을 성공적으로 완료시키기 위해서는 m개의 데이타 항목이 적어도 $|NU|\lfloor\frac{E}{|NU|}\rfloor$ 유닛동안 갱신되지 않아야 하기 때문이다 (이때 그 트랜잭션은 적어도 $\lfloor\frac{E}{|NU|}\rfloor$ 번의 무효화 비트 패턴을 받는다). 따라서, 응답시간에 대한 수식은 (6)과 같다 (즉, 제기와 종료간의 차이가 된다).

$$a_t(IA)=\sum_{i=0}^{\infty}E(1-c)^i=\frac{E}{c} \quad (6)$$

4.2.2 MA 메소드

[7]에서, 비균등 방송 환경에서 오래된 버전을 유지할 수 있는 3가지 접근방식이 제안되었다: 클러스터링(clustering), 오버플로 버킷 풀(overflow bucket pool), 새 디스크(new disks). 어떤 방식을 방송 구성에 적용하더라도, 비균등 방송 환경에서 오래된 버전을 유지하는 것은 방송 주기의 길이를 상당히 길어지게 만들게 되며, 그 길이는 데이타 항목마다 유지되는 오래된 버전의 개수에 비례하게 된다. 결론적으로 이에 의해 응답시간도 상당히 길어지게 된다.

각 데이타 항목이 연속적으로 나타나는 상호 도착시간은 오래된 버전을 포함시키는 방송 구성방식에 따라 차이가 나겠지만, 어느 최적의 방식이 있어서 그 간격은 항상 동일하다고 가정한다. $|NU|$ 동안 갱신된 데이타 항목의 평균 개수(N_c)가 $D(1-e^{-\mu|NU|})$ 이며, 서버는 모든 읽기-전용 트랜잭션을 성공적으로 실행할 수 있도록 하기 위해 데이타 항목마다 (많은 수의) k 개의 오래된 버전을 유지한다고 할 때, 방송의 길이는 적어도 kN_c 가 된다. 따라서 각 데이타 항목 $d_i\in D$ 에 대하여, d_i 의 평균 지연은 다음 수식 (7)과 같다. 여기서 f_i 는 가장 최신 값과 이전 버전으로 구성된 방송에서 d_i 의 빈도를 의미한다.

$$\omega_c(d_i)=\frac{|NU|+kN_c}{2f_i} \quad (7)$$

임의의 데이타 요구에 대한 예상 평균 응답시간은 $\sum_{d_i\in D}p(d_i)\omega_c(d_i)$ 로 계산된다. 그러므로, 지역 캐쉬없이 m 개의 데이타 항목을 접근하는 트랜잭션에 대한 평균 응답시간은 $m\sum_{d_i\in D}p(d_i)\omega_c(d_i)$ 으로 계산된다. 따라서 MA에서의 평균 응답시간은 다음 수식 (8)이 된다 (각 트랜잭션은 성공적으로 완료된다고 가정한다).

$$a_t(MA)=m(1-h^v)\sum_{d_i\in D}p(d_i)\omega_c(d_i) \quad (8)$$

이 수식에서 방송 크기 $h^v=e^{-\mu(|NU|+kN_c)}$ 는 kN_c 만큼 증가한다 (그림 2의 캐쉬 적중률에 대한 시나리오 참조).

4.2.3 PA 메소드와 PA^2 메소드

제안하는 기법들에서, 트랜잭션 실행은 3 단계로 나뉘어 진다: 전처리, 획득, 전송 단계. 클라이언트에서 각 단계마다 걸리는 시간을 PT, AT, DT 로 각각 표현한다면, 제안하는 PA 메소드와 PA^2 메소드의 평균 응답시간은 다음과 같이 표현될 수 있다.

$$a_t(PA)=a_t(PA^2)=PT+AT+DT \quad (9)$$

동기 방식의 PA 메소드에서, PT는 평균적으로 $\frac{1}{2}\times$ 주기에 해당하며, DT는 무시할 수 있을 정도로 사소하므로, 수식 (9)는 수식 (10)으로 간략화할 수 있다.

$$a_t(PA)\approx\frac{1}{2}|NU|+AT \quad (10)$$

PA 메소드에서, 방송에서 첫 번째 항목을 획득하는데 걸리는 시간은 a_s 그 자체가 된다. 두 번 째 항목을 방송에서 얻는데 걸리는 시간은 나머지 방송 크기의 $\frac{a_s}{|NU|}(=\delta)$ 가 되며, 다음 항목에 대한 시간은 다시 그 나머지 방송 크기의 δ가 된다. 따라서 m_p 개의 기선언된 항목을 가진 트랜잭션에 대한 평균 AT는 다음 수식 (11)로 나타낼 수 있다.

$$AT(PA)=\sum_{i=1}^{m_p(1-h)}\delta(1-\delta)^{i-1}|NU| \quad (11)$$

따라서 PA에 대한 예상 평균 응답시간은 다음과 같이 계산할 수 있다.

$$a_t(PA)\approx\frac{1}{2}|NU|+\sum_{i=1}^{m_p(1-h)}\delta(1-\delta)^{i-1}|NU| \quad (12)$$

비동기 방식의 PA^2 메소드의 경우, 역시 PT와 DT는 무시하여도 좋을 정도의 사소한 시간이므로, 수식

(9)는 다음 수식 (13)으로 바꿀 수 있다.

$$a_t(PA^2) \approx AT \quad (13)$$

PA^2 메소드에서, AT에는 트랜잭션이 실행을 시작한 방송 주기에서 데이타를 얻기 시작하는 시간부터, 다음 주기에서 그 외의 나머지 데이타 항목(이전에 얻었던 데이타 항목을 다시 받는 경우도 포함할 수 있다)을 내려 받는 시간이 포함된다. PA^2에 대한 평균 응답시간은 그에 대한 상황을 다음과 같은 두 가지로 나뉜 후 PA에 대한 수식을 이용하여 생각할 수 있다: 트랜잭션이 시작한 방송 주기에서 얻었던 데이타 항목 중 어떤 데이타도 무효화되지 않은 경우와 일부 항목이 획득 단계에서 무효화된 경우. 첫번째 경우에서, 클라이언트는 트랜잭션이 시작한 주기의 방송에서 평균적으로 $\frac{|NU|}{a_s}$개의 데이타 항목을 얻게되며 그 데이타 항목들이 무효화되지 않을 확률은 $\mu_0^{\frac{|NU|}{2a_s}}$이므로, PA^2의 AT는 수식 (11)과 $\mu_0^{\frac{|NU|}{2a_s}}$의 곱과 같다. 두번째 경우에서, 평균 AT는 수식 (12)와 $(1-\mu_0^{\frac{|NU|}{2a_s}})$의 곱과 같다. 따라서 PA^2의 예상 평균 응답시간은 두 곱의 합과 같다.

$$a_t(PA^2) \approx \frac{1}{2}|NU|(1-u_0^{\frac{|NU|}{2a_s}}) + \sum_{i=1}^{m_s(1-h)} \delta(1-\delta)^{i-1}|NU| \quad (14)$$

두 메소드 모두에서, 평균 응답시간의 상한은 $\frac{3}{2}|NU|$가 된다. 게다가 하한은 데이타 항목의 개수나 캐쉬 적중률을 고려하지 않는다면 $2|NU|$가 된다.

균등 방송의 경우, 모든 메소드에 대한 예상 평균 응답시간은 이 절에서 서술하였던 동일한 방법으로부터 자연스럽게 얻을 수 있다. 그러므로 우리가 응답시간을 계산하기 위해 필요한 것은 모든 수식에서 방송 크기를 D로 대치하고, a_s를 $\frac{1}{2}D$로 대치하는 것이다.

표 1 인자 설정

인자	값
D	가변적 (1000)
μ	가변적 (단위 시간당 5×10^{-4})
n	3
λ_1, λ_2, λ_3	4, 2, 1
$\|P_1\|$, $\|P_2\|$, $\|P_3\|$	50, 150, 800
k (MA 일 경우에만)	2
m	가변적 (10)
m_b (PA와 PA^2인 경우에만)	$1.5\times m$
f_{P1}, f_{P2}, f_{P3}	가변적 (0.7, 0.2, 0.1)

5. 성능 평가

이 절에서는 분석 결과를 설명한다. 결과에서, 시간 단위는 서버에서 하나의 데이타 항목을 전송하는데 걸리는 시간과 동일하다. 표 1은 서버(윗 부분)와 클라이언트(아래 부분)에 대한 인자를 요약하고 있으며, 괄호 안의 값은 기본 값이다. 클라이언트의 접근 빈도와 관련지어, 단일 파티션 내에 데이타 항목에 접근할 빈도는 균등하다고 가정한다.

5.1 트랜잭션 크기에 대한 영향

첫번째 분석 결과로써, 단위 시간당 μ가 5×10^{-4}로 설정되었을 때[1], 여러 메소드들에서 트랜잭션의 크기가 미치는 영향을 먼저 보인다. 그림 4는 균등 및 비균등 방송 구성에서, 트랜잭션 접근하는 데이타 항목의 개수가 증가할 때의 성능 결과를 보여준다. 데이타 항목의 개수가 큰 경우(우리 분석에서 5보다 큰 경우), IA의 응답시간은 급격하게 증가한다. 이것은 큰 값 m은 트랜잭션이 완료할 확률을 감소시키기 때문이다. 그 결과, 트랜잭션이 완료할 때까지 여러 번의 재시작으로 인한 성능 저하가 발생한다. MA는 클라이언트가 오래된 버전에 접근하도록 허용하여 트랜잭션의 완료 확률을 높임으로써, 이 문제를 해결한다. MA의 성능은 IA 보다는 항목의 개수에 덜 영향을 받음을 알 수 있다. 그러나 방송 크기가 증가하는 것은 균등 및 비균등 방송 모두에 부정적인 영향을 미치게 된다. 이 사실은 적은 수의 데이타 항목을 전송하는 경우에 MA가 IA보다 더 긴 응답시간을 보이는 이유를 설명해준다. 우리가 제안한 기법의 경우, 트랜잭션이 방송되는 순서대로 데이타 항목을 접근하기 때문에 평균 응답시간은 거의 트랜잭션의 크기에 영향을 받지 않는다. 그 결과, 우리의 기법들은 접근하는 데이타 항목의 개수가 많을 경우, MA보다 나은 성능을 보여주며, 또한 당연히 IA보다도 좋은 성능을 나타낸다. 예를 들면, m이 10일 때, 두 가지 방송 모두에서 PA^2의 응답시간은 MA보다 3배만큼 단축되었다. 우리가 제안한 기법의 경우, PA^2가 PA보다 단지 약간의 향상된 결과를 보여준다. 이는 높은 갱신 비율은 비동기 방식의 장점을 감소시켰기 때문이다. 기선언된 읽기집합에 부수적인 데이타 항목이 포함되며, 따라서 적은 수의 데이타 항목에 접근할 경우에는 PA^2가 PA보다 다소 나빠진 성능을 보인다.

1) 이 값은, D가 1000으로 설정되었을 때, 한 주기 동안 대략 데이타베이스 항목의 40%와 50%가 균등 및 비균등 방송에서 수정되었음을 의미한다.

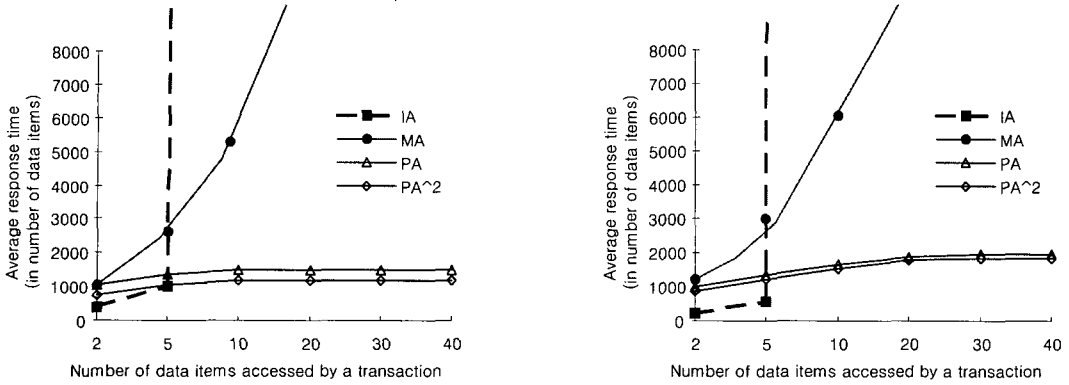

그림 4 트랜잭션 크기에 대한 영향

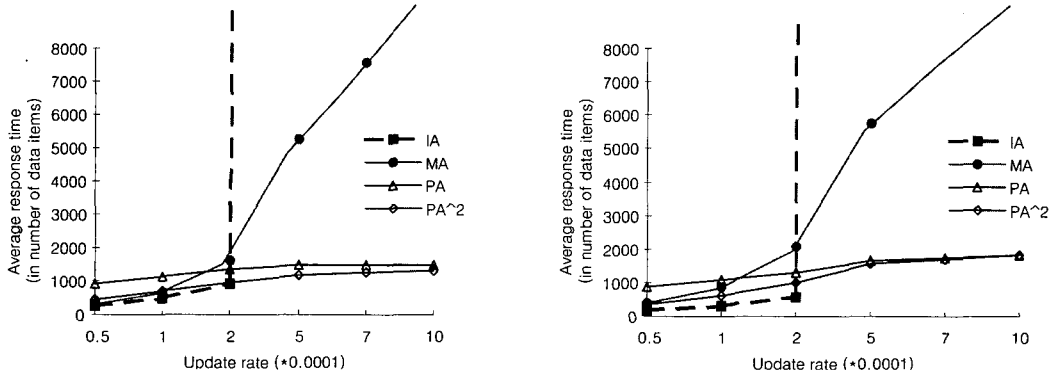

그림 5 갱신 비율에 대한 영향

5.2 갱신 비율에 대한 영향

그림 5는 m 값이 10으로 설정되었을 때, 여러 메소드들에 대하여 갱신 비율이 미치는 영향을 보여주고 있다. 높은 갱신 비율은 상대적으로 캐쉬 적중률이 감소함을 의미하며, 또한 캐쉬 데이타의 무효화 확률이 증가함을 의미한다. 이것은 *IA*의 응답시간이 그렇게 급격하게 나빠지는 이유가 된다. 특히, 단위 시간당 $\mu > 2\times10^{-4}$일 경우[2], *IA*는 아주 나쁜 응답시간을 보여주고 있다. 이것은 높은 갱신 비율은 데이타베이스에서 갱신되는 데이타 항목의 개수가 아주 많으며, 따라서 데이타베이스 크기가 더 커지기 때문이다. *IA*와는 달리 *MA*에서는, 트랜잭션은 오래된 버전 중에서 적절한 버전을 읽음으로써 완료할 수 있다. 이와 같은 완료 확률에서의 차이는 *MA*가 높은 갱신 비율 환경(우리의 분석에서 $\mu > 2\times10^{-4}$인 경우)에서 높은 성능을 보이는 이유가 된다. 상대적으로 갱신이 덜 빈번하게 발생할 경우, *IA*에서 트랜잭션이 완료할 확률이 높게 나타난다. 따라서 *IA*는 *MA*보다 더 나은 응답시간을 보여주는데, 이는 *IA*는 *MA*보다 필요한 데이타 항목을 더 빨리 얻을 수 있기 때문이다.

본 논문에서 제안한 기법의 경우, 응답시간은 갱신 비율에 크게 영향을 받지 않는다. 예상한대로, 그림 5에서, 제안한 기법들은 높은 갱신 비율에서 *MA*와 *IA*보다 더 향상된 결과를 보여준다. 단지 적은 수의 데이타 항목이 수정되는 경우에만 본 논문에서 제안하는 기법들이 *MA*와 *IA*보다 응답시간이 더 길어진다. 그러나 우리의 예상대로, PA^2는 데이타 갱신이 거의 일어나지 않아 하나의 방송 주기 내에서 트랜잭션이 실행될 수 있는 환경에서는 다른 기법들과 비슷한 평균 응답시간을 지닌다.

2) 이 값은, D가 1000으로 설정되었을 때, 한 주기 동안 대략 데이타베이스의 20%와 30%이상의 데이타 항목이 균등 및 비균등 방송에서 수정되었음을 의미한다.

5.3 접근 패턴에 대한 영향

이 분석 결과는 클라이언트의 접근 패턴과 서버의 방송 프로그램간의 불일치가 미치는 영향을 보여준다. 비균등 방송의 경우, 서버의 방송은 일부의 클라이언트들에게만 효율적으로 동작한다. 이는 일부 클라이언트의 접근빈도에 대한 부적절한 정보 때문이며, 이는 동적으로 계속 변하며, 또한 다수의 클라이언트의 요구에 맞춰 서버가 방송 프로그램을 작성해야하는 부하 때문이다.

그와 같은 한 클라이언트의 요구와 서버의 방송 프로그램간의 불일치를 모델링하기 위하여, 기본 데이타 파티션에 대하여 3가지 접근 패턴을 사용한다: AP1 = (0.1, 0.2, 0.7), AP2 = (0.5, 0.3, 0.2), AP3 = (0.7, 0.2, 0.1). 여기에서 AP1은 상대적으로 가장 덜 일치되는 접근 패턴이며, AP2는 그 다음으로 덜 일치되며, AP3이 상대적으로 가장 잘 일치되는 접근 패턴이다. 여기에서는 비균등 방송 환경에서 클라이언트가 기본 갱신 비율의 다른 접근 패턴을 보일 때, *MA*와 제안하는 기법들의 성능을 살펴본다 (*IA*는 상대적으로 매우 낮은 성능을 보이므로 이 소절부터 *IA*는 고려하지 않는다. 또한, *PA*와 PA^2의 평균 응답시간은 그 상한값과 매우 비슷한 값을 지니기 때문에 상한값으로만 표시한다).

결과에서, *MA*의 응답시간은 AP3부터 AP1까지 불일치 정도가 증가함에 따라 20%정도 악화됨을 알 수 있었다. 이는 클라이언트가 비균등 방송 프로그램의 장점을 얻을 수 없기 때문에, 성능 저하는 예상되었었다. 그러나 제안하는 기법의 경우, 응답시간의 상한은 그러한 불일치에 전혀 영향을 받지 않는다.

5.4 데이타베이스 크기에 대한 영향

이 절에서는, 이전의 실험 -데이타베이스의 크기가 1000 인 경우- 과 달리 데이타 항목의 개수가 1000부터 4000까지 증가할 때 성능에 미치는 영향을 살펴본다. 데이타베이스의 크기가 증가하면 *MA*와 제안하는 기법 모두의 성능은 나빠진다. 이는 데이타베이스의 크기가 증가하면 (1) 방송의 한 주기의 길이가 길어지고, (2) 캐쉬 적중률이 낮아지기 때문이다.

표 2는 클라이언트의 기본 접근 패턴 설정에서 응답시간 결과를 나타낸다. 이 표에서, 모든 메소드들에서의 성능은 데이타베이스의 크기가 증가함에 따라 악화됨을 알 수 있다. 그러나 제안한 기법들은 *MA*보다 데이타베이스 크기의 증가에 (특히, 균등 방송 환경에서) 더 향상된 확장성을 보인다. 이것은 다음과 같이 설명할 수 있다. 제안하는 기법들의 응답시간은 방송의 한 주기의 길이에만 영향을 받지만, *MA*의 경우 방송의 한 주기의 길이뿐만 아니라 낮은 캐쉬 적중률에도 영향을 받기 때문이다. 표 2에서 볼 때, 제안하는 기법이 모든 경우에서 *MA*보다 나은 성능을 보이며, 성능 격차는 데이타베이스의 크기가 증가함에 따라 더 커진다. 예를 들면, 균등 방송 환경에서 *MA*에 비해 2.5배에서 8.1배까지 성능이 향상됨을 볼 수 있다.

표 2 데이타베이스 크기에 대한 결과

데이터베이스 크기	균등 방송			비균등 방송		
	MA	*PA*, PA^2	성능향상 비율	*MA*	*PA*, PA^2	성능향상 비율
1000	5279	1500	2.5	6040	1950	2.1
2000	20290	3000	5.8	13938	3900	2.6
3000	30138	4500	5.7	20887	5850	2.6
4000	54355	6000	8.1	27091	7800	2.5

6. 논 의

6.1 접속단절 해결

이동 기기를 지닌 클라이언트에서는 여러 가지 이유 - 예를 들어, 배터리 전력을 절약하거나 혹은 방송 채널을 접근하는데 소요되는 금전 비용을 최소화하기 위해 - 로 방송 채널을 들을 수 없는 접속단절(disconnections)이 발생할 수 있다. 또한, 에러가 자주 발생하는 신뢰할 수 없는 무선 통신으로 인해서 클라이언트는 서버가 보내주는 방송 데이타 혹은 무효화 정보의 일부를 놓칠 수도 있다. 이 장에서는 *PA* 메소드와 PA^2 메소드의 획득 단계 수행 중 무효화 비트 패턴을 놓쳤을 경우를 해결할 수 있는 방안을 아래에 보여준다.

접속단절을 해결하기 위한 프로시져: 클라이언트가 획득 단계 수행 중 접속단절로 인해 무효화 비트 패턴을 놓치게 되면, 그 클라이언트는 접속단절로부터 깨어난 후 지역 캐쉬내의 모든 데이타를 무효화하고 *Acquire*(T_i)를 공집합(Φ)으로 설정한다. 그런 다음, 클라이언트는 뒤따르는 방송 데이타로부터 무효화된 데이타 및 *Pre_RS*(T_i)에 속하는 데이타를 읽음으로써 획득 단계와 선반입 과정을 수행한다.

위에 언급된 간단한 방법에서는 지역 캐쉬내의 몇몇 데이타가 실제로는 유효함에도 불구하고 무효화되는 이른바 "거짓 무효화(false invalidation)"가 발생할 수 있다. 즉, 접속단절로 인해 캐쉬 전체를 버리는 것은 에너지 효율 면에서 매우 비용이 높아질 수 있다. 특히, 캐쉬내의 대부분의 데이타가 여전히 유효할 경우에는 더욱 그러하다[13]. 실제로, 일대일(point-to-point) 통신 환경에서는 접속 단절되었던 클라이언트가 에너지를 많이 소모하는 요청 메시지를 보내야 하기 때문에 처음부

터 다시 무효화된 모든 데이타를 얻는 방식은 매우 부하가 크다고 볼 수 있다. 이전의 연구[9,11]에서는 무효화 정보를 주기적으로 다시 보내는 기법을 사용함으로써 클라이언트가 접속단절에 대한 적응력을 증가시킬 수 있음을 보였다. 예를 들어, 하나 이상의 최근 방송 주기 동안에 갱신된 데이타에 대한 무효화 비트 패턴을 방송함으로써 접속단절된 클라이언트가 다시 동기화를 할 수 있게끔 할 수 있을 것이다. 그러나 이러한 경우 클라이언트는 캐쉬내의 데이타를 계속 유지할 것인지 아니면 버릴 것인지에 대한 결정을 내리기 위해 다음 번 무효화 비트 패턴이 도착할 때까지 계속 기다려야 하며, 이로 인해 응답 시간이 길어지게 된다.

반면에, 푸쉬-기반 데이타 전달 환경에서는 본 논문에서 제시한 간단한 방법을 사용함으로써 접속단절에 효과적으로 대응할 수 있다. 이는 클라이언트가 명시적으로 요청 메시지를 보내지 않아도 서버가 방송한 데이타를 읽음으로써 가능하다. 이와 같이, 기선언-기반 트랜잭션 실행 기법은 접속단절로 인해 놓쳐버린 무효화 비트 패턴에 적절히 대응할 수 있게 된다. 본 논문에서 제안한 PA 메소드와 PA^2 메소드에서는 클라이언트가 트랜잭션을 수행하는 동안에 오직 하나의 무효화 비트 패턴만을 읽게됨에 유의하자. 비록 클라이언트가 무효화 비트 패턴을 놓치게 되더라도, 뒤따르는 한 주기의 방송으로부터 모든 데이타를 읽을 수 있게 된다. 이러한 이유로 인해, 본 논문에서는 무선 트랜잭션을 처리하는 동안에 발생할 수 있는 접속단절을 해결하기 위해 여기에서 제안하는 비용 면에서 매우 효과적인 방식을 채택할 수 있을 것이다.

6.2 관련 연구와의 비교평가에 대한 고찰

본 논문에서는 기선언-기반 트랜잭션 실행 기법의 우수성을 보여주기 위해 수학적 분석 및 그 분석 결과를 이용하였다. 본 논문에 나타난 수학적 분석의 타당성은 우리의 이전 논문[14,15]을 통해 보여줄 수 있다. 즉, 이전의 연구[14,15]에서 나타난 시뮬레이션을 통한 IA 및 MA 메소드의 성능 평가는 본 논문에서 보여준 수학적 분석을 통한 IA 및 MA 메소드의 성능과 일치된 형태를 보여주고 있다. 또한, 여기에서 제안하는 PA 및 PA^2 메소드에 대해서도 수학적 분석과 일관된 결과를 시뮬레이션을 통해 확인할 수 있었다. 그럼에도 불구하고, 본 논문의 저자들은 아주 작은 값의 신뢰 구간(confidence intervals)을 얻기 위해 많은 수의 시뮬레이션을 수행하는 방법 대신, 좀 더 정확한 전체 응답시간을 얻기 위해 수학적 분석을 통해 기존 기법들과 비교 평가를 수행하였다.

흥미롭게도, 이전의 연구[7]에서는 트랜잭션의 읽기집합을 미리 알 경우에 대한 질의 최적화에 대해 간략히 소개하고 있다. 즉, 서버가 방송하는 데이타 항목 순서대로 트랜잭션이 접근하는 데이타 항목의 순서를 재순서화(reordering)한다. 하지만 [7]에서 제시하는 질의 최적화 방법은 클라이언트에서 수행되는 읽기-전용 트랜잭션이 여러 데이타 항목을 순서에 상관없이 접근하는 경우에만 적용 가능하다. 즉, 트랜잭션에서 요청하는 데이타 항목 접근 순서와는 틀리게, 서버가 방송하는 데이타 항목 순서대로 데이타를 읽어 사용자에게 그대로 보여지게 된다. 이와는 달리, 본 논문에서 제안하는 기법은 여러 데이타 항목을 지정된 순서에 의해 접근하는 읽기-전용 트랜잭션을 지원할 수 있다는 점에서 [7]에서 소개한 질의 최적화와는 차이가 있다.

7. 결 론

본 논문에서는 무선 데이타 방송 환경에서, 트랜잭션의 직렬화가능성을 유지하면서도 무선 읽기-전용 트랜잭션의 처리 속도를 향상시킬 수 있는, 간단하면서도 효율적인 기선언-기반 메소드를 제안하였다. 이전의 트랜잭션 처리 기법과 비교할 때 처리 및 저장공간에서 부하가 다소 발생할 수는 있겠지만, 분석 작업에서 알 수 있듯이, 본 논문에서 제안한 기법이 보여주는 상당한 응답시간의 단축은 그러한 부하를 능가한다. 앞으로의 연구방향은 기선언-기반 트랜잭션 처리 기법을 여러 개의 무선 방송 채널 환경[16]에 적용하여 그 효과를 측정하는 것이다.

참 고 문 헌

[1] D. Barbara, "Mobile computing and databases: a survey," *IEEE Transactions on Knowledge and Data Engineering*, Vol.11, No.1, pp. 108-117, 1999.

[2] T. Imielinski and R. Badrinath, "Wireless mobile computing: challenges in data management," *Communications of the ACM*, Vol.37, No.10, pp. 18-28, 1994.

[3] K.-L. Tan and B. C. Ooi, *Data Dissemination in Wireless Computing Environments*, Kluwer Academic Publishers, 2000.

[4] J. Shanmugasundaram, A. Nithrakashyap, R. Sivasankaran, and K. Ramamritham, "Efficient concurrency control for broadcast environments," *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 85-96, 1999.

[5] H. Garcia-Molina and G. Wiederhold, "Read-only transactions in a distributed database," *ACM Transactions on Database Systems*, Vol.7, No.2,

pp. 209-234, 1982.
[6] P. A. Bernstein, V. Hadzilacos and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, Massachusetts, 1987.
[7] E. Pitoura and P. Chrysanthis, "Exploiting versions for handling updates in broadcast disks," *Proceedings of the 25th Conference on Very Large Data Bases*, pp. 114-125, 1999.
[8] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast disks: data management for asymmetric communication environments," *Proceed- ings of the ACM SIGMOD conference on Management of Data*, pp. 199-210, 1995.
[9] E. Pitoura and P. Chrysanthis, "Scalable processing of read-only transactions in broadcast push," *Proceedings of the 19th International Conference on Distributed Computing Systems*, pp. 432-439, 1999.
[10] S. Acharya, M. Franklin, and S. Zdonik, "Disseminating updates on broadcast disks," *Proceedings of the 22nd International Conference on Very Large Data Bases*, pp. 354-365, 1996.
[11] D. Barbara, and T. Imielinski, "Sleepers and workaholics: caching in mobile environments," *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 1-12, 1994.
[12] N. H. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments," *Wireless Networks*, Vol.5, No.3, pp. 171-182, 1999.
[13] K.-L. Wu, P. S. Yu, and M.-S. Chen, "Energy-efficient caching for wireless mobile computing," *Proceedings of the 12th International Conference on Data Engineering*, pp. 336-343, 1996.
[14] S. Kim, C.-S. Hwang, H. Yu, and S. Lee, "Optimistic scheduling algorithm for mobile transactions based on reordering," *Proceedings of the 2nd International Conference on Mobile Data Management*, pp. 105-117, 2001.
[15] S. Kim, S. Lee, S. Jung and C.-S. Hwang, "O-PreH: Optimistic transaction processing algorithm based on pre-reordering in hybrid broadcast environment," *Proceedings of the 10th International Conference on Information and Knowledge Management*, pp. 553-555, 2001.
[16] K. Prabhakara, K. Hua, and J. Oh, "Multi-level multi-channel air cache designs for broadcasting in a mobile environment," *Proceedings of the 16th International Conference on Data Engineering*, pp. 167-176. 2000.

이 상 근
1994년 2월 고려대학교 전산과학과 학부 졸업. 1996년 2월 고려대학교 전산과학과 석사 졸업. 1999년 8월 고려대학교 전산과학과 박사 졸업. 2000년 4월 ~ 2001년 3월 동경대학교 생산기술연구소 Visiting Postdoc 연구원 2001년 4월 ~ 현재 LG 전자 단말연구소 선임연구원. 관심분야는 이동 분산 데이타베이스 시스템, WAP프로토콜 등

김 성 석
1997년 2월 고려대학교 전산과학과 학부 졸업. 1999년 2월 고려대학교 전산과학과 석사 졸업. 2001년 2월 고려대학교 컴퓨터 학과 박사 수료. 이동 정보 시스템, 이동 객체 관리 시스템 등. 관심분야는 분산 데이타베이스 시스템 등

황 종 선
1966년 고려대학교 수학과 학부 졸업. 1970년 고려대학교 수학과 석사 졸업. 1978년 The university of Georgia, computer science and statisitcs 박사 졸업. 현재 고려대 학교 컴퓨터학과 교수. 관심분야는 이동 컴퓨팅, 분산 시스템, 지식-기반 시스템 등