휴대용 정보기기를 위한 플래시 기반 2단계 로킹 기법

변 시 우* · 노 창 배** · 정 명 희***

Flash-Based Two Phase Locking Scheme for Portable Computing Devices

Siwoo Byun* · Chang-bae Roh** · Myunghee Jung***

Abstract

Flash memories are one of best media to support portable computer's storages in mobile computing environment. The features of non-volatility, low power consumption, and fast access time for read operations are sufficient grounds to support flash memory as major database storage components of portable computers. However, we need to improve traditional transaction management scheme due to the relatively slow characteristics of flash operation as compared to RAM memory. In order to achieve this goal, we devise a new scheme called *Flash Two Phase Locking* (F2PL) scheme for efficient transaction processing. F2PL improves transaction performance by allowing multi version reads and efficiently handling slow flash write/erase operation in lock management process. We also propose a simulation model to show the performance of F2PL. Based on the results of the performance evaluation, we conclude that F2PL scheme outperforms the traditional scheme.

Keywords : Portable Information Devices, Flash Memory, Database, Locking, Simulation

논문접수일 : 2005년 8월 28일 논문게재확정일 : 2005년 11월 8일

※ 이 논문은 2003년도 안양대학교 학술연구비의 지원을 받아 연구되었음.

* 주저자, 안양대학교 디지털미디어학부 조교수, (430-714)경기도 안양시 만안구 안양5동,
Tel : 031-467-0922, Fax : 031-467-0800, e-mail : swbyun@anyang.ac.kr

** 안양대학교 디지털미디어학부 겸임교수

*** 안양대학교 디지털미디어학부 조교수

1. 서 론

무선 통신망의 급속한 보급과 더불어, PDA (Personal Digital Assistant), HPC(Hand- held PC), PPC(Pocket PC), 개인용 휴대전화(mobile phone), 스마트 폰(Smart Phone) 등의 각종 소형 정보기기들이 대중화됨에 따라, 정보 저장용 미디어로 플래시 메모리가 보편적으로 활용되게 되었다. 따라서 플래시 미디어에 저장된 데이터를 체계적으로 관리하는 임베디드 소프트웨어가 필요하게 되었다.

세계적인 시장조사 예측기관인 Gartner社의 자료에 따르면, 정보기기용 임베디드 소프트웨어는 최근 100%이상으로 급성장하고 있다고 한다[이정배, 2002]. 특히, 세계적인 분석 기관인 IDC(International Data Corporation)社는 64% 이상의 휴대용 정보기기의 어플리케이션이 임베디드 데이터베이스 시스템이 필요하다고 보고했다[유제정, 2004]. 따라서 소형 정보기기의 플래시 메모리 기반 데이터베이스 시스템은 핵심 임베디드 소프트웨어 분야로서 상당히 유망하며, 이와 관련된 데이터 관리 기술의 개발이 매우 시급하다고 할 수 있다.

본 논문은 휴대용 소형 정보기기의 데이터 저장 장치로 많이 사용되는 플래시 메모리를 기반으로 하는 플래시 메모리 데이터베이스 시스템에서의 새로운 트랜잭션 처리 기법을 제안한다.

2. 관련 연구

현재 플래시 미디어 자체에 대한 연구나 플래시 파일 시스템에 대한 연구는 성숙단계에 진입하였다. 또한, 기존의 디스크-기반의 상용DR (Disk Resident) 데이터 처리기나 메인 메모리-기반 MM(Main Memory) 데이터 처리기는 오래전부터 인기 있는 연구 분야로 활발한 연구가 지속되고 있다. OBE[Ammann, 1985], MM-DBMS [Lehman, 1986], MARS[Eich, 1987], HALO [Garcia, 1987], System M[Garcia, 1992]등의 시스템이 과거의 메모리 기반 데이터 처리기의 기본 연구사례이다.

국내는 (주)코스모에서 개발한 메모리 상주형 DBMS인 RtPlus[조주현, 2002]는 한국전자통신연구원의 DREAM-S의 후속 모델로, 범용 멀티미디어 처리를 위하여 객체관계형 데이터 모델을 기반으로 개발된 초소형 실시간 DBMS이다. 하지만 플래시 메모리를 활용하는 데이터베이스 시스템에 대한 연구는 아직 시작 단계에 불과한 실정이다.

플래시 메모리를 저장 매체로 사용한 연구에는 eNVy 시스템이 있다[민용기, 1999]. 이 시스템은 플래시 메모리를 일반 중소형 컴퓨터 및 하드 디스크의 대용으로 사용하였다. 플래시 메모리를 다량으로 연결하여 2GB의 저장 영역을 구성하였다. 플래시 메모리의 특성으로 copy-on-write 구조를 가지고 있으며, 세그먼트 소거 연산의 오버헤드를 줄이기 위하여 MMU (Memory Management Unit)을 별도로 가지고 있다. 이 구조는 휴대형 컴퓨터에서 사용하기에는 너무 크고, 소거 연산에 너무 많은 시간을 소모한다.

또한, 데이터 처리기의 성능 개선을 위해서 플래시 메모리를 사용한 연구도 있다. Phillippe이란 연구자가 PicoDMBS란 스마트 카드용 데이터베이스를 제안하면서, 램의 사용을 줄이고 연산 속도 개선을 위하여 저장 장치인 플래시 미디어의 저장구조를 포인터 방식으로 새롭게 개선하여 제안하기도 하였다.

또한, 저장된 데이터의 접근 효율의 향상을 위하여 다중 버전 기법[Bernstein, 1987 ; 황규영, 2000]을 활용해 볼 수 있다. 이 방법은 한

데이터 항목에 대하여 여러 개의 버전을 유지하고, 한 트랜잭션이 그 항목에 대하여 접근을 요구할 때 현재 수행중인 스케줄의 직렬가능성을 유지하기 위해 가장 적절한 버전을 선택되도록 한다. 다중 버전의 제어 기법은 타임스탬프 순서 기법과 2단계 잠금 기법을 기반으로 하여 구현되어 진다.

다중 버전 기법은 원래 다른 기법에서는 거부될 일부의 읽기 연산들이 이전 버전을 읽도록 허용하여 접근의 효율성을 높이는 반면에, 여러 버전을 유지하기 위한 많은 디스크 공간이 필요한 단점이 있다. 이러한 데이터 접근 기법을 플래시 메모리에 적용하면 느린 쓰기 연산으로 인한 접근 효율의 저하를 방지하고 읽기 연산의 접근 속도를 높일 수 있다.

그러나 아직도 순수하게 플래시 미디어 기반의 데이터 처리기의 연구는 초창기의 시작 분야이다. 이유는 그 동안 플래시 메모리는 주 메모리의 성능과 비교하면 느리면서도 고가이며, 일반적인 시스템에는 사용되지 않는 특수 부품이며, 디스크에 비하여 저장 비용이 고가였기 때문이다. 그러나 최근 플래시 메모리 기술의 발전으로 대부분의 단점들이 해소된 상태이며, 저렴하고 성능 좋은 미래의 저장 메모리로서 크게 각광 받고 있다. 특히, 대부분의 임베디드 시스템에서는 공간제약, 전력소모, 중량 및 내충격성 문제로 디스크는 사용할 수 없기 때문에, 비휘발성(Non-Volatile) 저장장치로서 플래시 메모리를 반드시 사용하여야만 한다. 다만, 일반 메인 메모리와는 달리, 쓰기와 지우기 연산에 10배정도의 많은 시간이 소요되며, 쓰기 회수가 100,000번 정도로 제한된다는 특성을 가만하여, 효과적인 휴대형 정보 기기의 데이터 처리기를 개발하여야 한다.

3. 플래시 메모리의 특성을 고려한 트랜잭션 처리 기법 제안

3.1 설계 목표

플래시 메모리는 비휘발성 저장장치로서 저렴하고 속도가 빠르다. 하지만, 일반 메인 메모리와는 달리, 쓰기와 지우기 연산에 10배 이상의 많은 시간이 소요되며, 쓰기 연산에 앞서 세그먼트 단위의 저속도의 지우기 연산이 먼저 수행되어야 하는 플래시 메모리의 특성상의 약점이 있다. 이러한 고유한 약점을 고려하여 기존의 트랜잭션 처리 기법을 개선하지 않으면, 플래시 메모리를 주 저장장치로 사용하는 소형 정보 기기에서 충분한 성능을 기대하기는 어렵다.

다음은 본 연구에서 성능 개선을 위하여 제안하는 플래시 기반의 트랜잭션 처리 기법의 기본적인 설계 목표이다.

① 플래시 메모리 접근 트랜잭션의 직렬가능성을 유지한다.
② 플래시 메모리 접근 트랜잭션의 처리 성능을 높인다.
③ 플래시 메모리 접근 트랜잭션의 철회율을 낮춘다.

3.2 플래시 2단계 로킹 기법의 제안

일반적으로 트랜잭션 처리 관련된 동시성 제어 기법에서 데이터 일관성 유지를 위하여 직렬가능성(Serializability)을 보장하게 된다. 보통 트랜잭션이 입출력을 위하여 기다리는 동안 중앙처리장치는 대기상태로 되므로 이용률이 저하된다. 이때 일관성을 유지하면서 다른 트랜잭션을 동시에 수행하게 되면 처리 성능은 자연히 높아진다. 즉, 트랜잭션의 직렬화가 가능하게 되면 정확성과 동

시 실행성의 장점을 모두 얻을 수 있다. 직렬 가능성을 보장하는 보편적인 방법이 2단계 로킹(two-phase locking : 2PL) [Tamer, 1991]이며, 동시에 실행되는 트랜잭션의 상호 간섭을 방지하기 위해 데이터 항목에 록(lock)을 걸게 된다.

현실적으로 데이터베이스 관리 시스템을 구현할 때, 여러 가지 2단계 로킹의 변형 기법들 중에서 많이 사용되는 것이 엄격한 2단계 로킹(strict two-phase locking : S2PL) 기법[Bernstein, 1987]이다. <표 1>에서와 같이 S2PL은 간단히 읽기 로크와 쓰기 로크를 가지며, 읽기 로크간에만 호환성이 유지된다. 또한, S2PL은 트랜잭션 T가 완료되거나 철회될 때까지 T가 보유한 로크들 중 어떠한 로크도 해제하지 않는다. 따라서 S2PL은 트랜잭션 T가 완료되지 않았으면 어떠한 다른 트랜잭션도 T가 쓴 항목을 읽거나 쓸 수 없다. 반면 S2PL과 비교되는 보수적 2PL의 경우에는 트랜잭션이 수행을 시작하기 전에 필요로 하는 모든 로크에 대하여 로크를 획득한다는 차이가 있다.

〈표 1〉 S2PL 기법의 록 모드 및 호환성 테이블

록 모드	읽기(R)	쓰기(W)
읽기(R)	Y	N
쓰기(W)	N	N

이러한 S2PL 기법은 플래시 메모리 데이터베이스 환경에 적용할 경우 설계 목표 ①의 직렬가능성 조건을 만족한다. 그러나 설계 목표 ②와 ③을 만족하기 위해서는 플래시 메모리 데이터베이스 환경을 고려하여 더 개선할 필요가 있다.

먼저, 설계 목표 ②와 설계 목표 ③과 관련된 이슈에 대하여 살펴보자. 읽기 트랜잭션의 경우에는 플래시 메모리의 연산 속도가 일반 RAM 메모리에 비하여 크게 뒤지지 않으므로 별 문제가 없다. 하지만, 플래시 메모리의 쓰기 연산의 경우에는 속도가 상대적으로 매우 느리며, 더욱이 쓰기 연산 전에 상당히 느린 소거 연산을 반드시 수행해야 하므로 연산의 부담이 상당히 큰 것이 문제이다. 이 문제를 해결하고자, 본 논문에서는 기존의 트랜잭션 처리 방식을 개선한 플래시 2단계 로킹 기법(Flash two-phase locking : F2PL)을 제안한다. 즉, 트랜잭션 관련 데이터가 플래시 메모리에 위치한 경우에 플래시 메모리의 접근 특성을 고려하여 효율적인 트랜잭션 관리를 제공하는 기법이다. 이 기법을 통하여 설계 목표 ②와 ③을 만족시킬 수 있다.

플래시 메모리에 쓰기 트랜잭션을 수행시킬 경우, 실제 특정 번지에 쓰기 연산을 수행하기 전에 2ms의 매우 느린 속도로 16KB 크기를 가진 세그먼트 단위의 소거 연산이 먼저 수행되어야한다. 또한, 소거 연산 수행후의 쓰기 연산도 한 바이트 단위가 아닌 512 바이트의 블록 단위로 수행되어야 하는 제약이 있다. 따라서 쓰기 연산의 비중이 높을수록 빈번한 트랜잭션 충돌이 심각하게 발생하여 트랜잭션 처리 성능을 크게 저하시키게 된다. 이 때, 접근하고자 하는 세그먼트 블록을 미리 점유하지 않고 바로 쓰기 록을 진행하면 높은 충돌 특성에 의하여 다수의 트랜잭션이 세그먼트 점유 과정에서 중도에 철회되게 된다. F2PL 기법은 쓰기 연산의 록을 쓰기-의도 록과 쓰기 록으로 분리하여 쓰기 연산 진행시 미리 쓰기-의도 록을 획득하여 진행하게 제어하였다. F2PL 기법은 이 두 가지의 록을 효율적으로 관리하여서 가장 심각한 쓰기 트랜잭션 관련 철회율을 개선하게 된다. 또한, 쓰기 트랜잭션은 장시간 록을 점유하기 때문에 이 점유 시간에 발생되는 다수의 읽기 트랜잭션은 모두 대기하거나 철회되어야 한다. 이로 인한 트랜잭션 처리 성능의 저하도 심각하므로 F2PL 기법은 읽기 트랜잭션에 한하여 다중 버전을 허용하였다.

<표 2>에서 보는 바와 같이 다중 버전 기법은 읽기 로크, 쓰기 로크, 보증 로크의 3가지의

로크가 있다. 그러므로 데이터 항목의 상태는 여기에 로크 해제가 포함되어 4가지의 중 하나가 된다. 다중 버전 로킹 기법에서는 여러 읽기 연산들과 하나의 쓰기 연산이 동시에 수행될 수도 있다. 이에 비하여, 표준적인 S2PL 기법에서는 원천적으로 로크 호환이 차단되어서 수행이 불가능한 단점이 있다.

〈표 2〉 다중 버전 기법의 록 모드 및 호환성 테이블

록 모드	읽기(R)	쓰기(W)	보증
읽기(R)	Y	Y	N
쓰기(W)	Y	Y	N
인증(C)	N	N	N

F2PL 기법은 기존의 다중 버전 로킹 기법[Bernstein, 1987 ; 황규영, 2000]을 기반으로 확장하였다. F2PL 기법은 <표 3>과 같이 읽기 록(Read Lock), 쓰기-의도 록(Write-Intention Lock), 인증 록(Certify Lock)의 4가지 록 모드가 있다. <표 1>은 4가지 록 모드와 상호간의 호환성을 나타내고 있다. 일반적인 록 관리 기법에서는 한 트랜잭션이 특정 오브젝트에 대하여 쓰기 록을 점유하게 되면, 추후에 발생되는 읽기 및 쓰기 트랜잭션은 이 특정 오브젝트에 대한 연산을 수행할 수 없다. F2PL의 다중버전 허용 2단계 록 관리는 어떤 트랜잭션이 특정 오브젝트에 대하여 쓰기 록을 걸고 있을 때, 이 오브젝트에 접근하고자 하는 다른 읽기 트랜잭션을 허용하는 것이다. 이 F2PL 기법은 한 오브젝트에 대하여 두 가지 버전을 관리하여 제어한다. 한 버전 오브젝트는 완료된 트랜잭션이 쓰기를 수행한 것이고, 다른 하나의 버전 오브젝트는 어떤 트랜잭션이 그 오브젝트에 대해 쓰기 록을 점유할 때 만들어 진다. 따라서 어떤 트랜잭션이 쓰기 록을 점유하고 있어도 추후 발생된 읽기 트랜잭션은 완료된 버전의 오브젝트를 계속 읽을 수 있다. 쓰기 트랜잭션은 완료된 버전의 값에 영향을 주지 않고 원하는 대로 오브젝트의 값을 갱신할 수 있다. 그러나 일단 트랜잭션이 완료할 준비가 되면 완료하기 전에 현재 쓰기 록을 보유하고 있는 모든 항목들에 대하여 인증 록을 확보해야 한다. 인증 록은 읽기 록과 호환성이 없으므로 자신이 쓰기 록을 가지고 있는 오브젝트들을 읽고 있는 다른 트랜잭션이 없을 때까지 트랜잭션 완료가 연기된다.

〈표 3〉 F2PL의 록 모드 및 호환성 테이블

록 모드	읽기(R)	쓰기-의도(WI)	쓰기(W)	인증(C)
읽기(R)	Y	Y	Y	N
쓰기-의도(WI)	Y	N	N	N
쓰기(W)	Y	N	N	N
인증(C)	N	N	N	N

F2PL 록 관리자는 트랜잭션 관리자와 플래시 데이터 관리자와 리모트 콜과 메시지 교환을 통하여 록 관리를 수행한다. 트랜잭션 관리자로부터 요청 메시지가 오면, 이를 분석한 후 록 호환성 테이블에 의거하여 적절히 록을 설정한다. 그리고 플래시 데이터 관리자를 호출하여 읽기 연산, 쓰기 연산, 완료 연산 등을 수행하는데, 읽기 연산의 경우는 최근에 완료된 데이터를 판독하게 되며, 쓰기 연산은 쓰기-의도 록과 쓰기 록을 확보한 후 실제 플래시 메모리에 쓰기가 수행되며 이때 새 버전이 생성되게 된다. 쓰기 연산이 완료되면, 인증 록으로 전환되어 읽기 록이 모두 해제될 때까지 기다린 후 종료되게 된다. 플래시 데이터 관리자로부터 완료나 철회 메시지를 받게 되면, F2PL 록 관리자는 해당 트랜잭션에 설정된 록을 모두 해제하고 대기하는 다른 트랜잭션을 꺼내어 수행에 필요한 관련 록들을 설정한 후 다시 플래시 데이터 관리자에 연산 수행을 요청하게 된다.

간략히 의사 코드로 표현된 F2PL 기법의 록 관리 알고리즘은 아래의 <알고리즘 1>과 같다.

먼저, 플래시 트랜잭션 관리자로부터 데이터베이스 연산에 대한 요청을 받게 되면, 해당 연산의 트랜잭션 아이디와 연산의 종류 등을 파악한다. 연산의 종류가 읽기 또는 쓰기 연산이면 현재의 데이터 항목에 록을 걸고 있는 모든 록 유닛을 검색한다. 만일 록이 전혀 설정되어 있지 않거나, 록이 호환가능하다면 해당 록을 설정한 후, 플래시 데이터 관리자에게 해당 연산 실행 메시지를 보낸다. 연산의 종류가 트랜잭션 완료이면 쓰기 록이 인증 록으로 전환된다.

다음으로, 플래시 데이터 관리자로부터 연산 수행의 결과 메시지를 받게 되면, 해당 연산의 수행 결과와 연산의 종류 등을 파악한다. 연산의 종류가 트랜잭션의 완료나 철회이면 트랜잭션을 종료하면서 대기하고 있는 다른 트랜잭션을 활성화 시켜주어야 한다. 즉 자신의 록을 풀어준 후, 대기 큐에서 맨 앞에 위치한 트랜잭션을 선택하여 이 트랜잭션과 호환가능한 모든 트랜잭션들을 검색한다. 검색된 트랜잭션 집합에 대하여 각각 해당 록을 설정하고, 다시 플래시 데이터 관리자에게 실행 요청을 함으로써 계속해서 다른 트랜잭션들이 수행되도록 한다.

```
Procedure F2PL-Lock-Manager
Input: Message msg; /* messages from transaction manager or flash data manager */
do {
  wait(msg);
  switch (msg.type) {
    case DBOP: /* a database operation from the transaction manager */
        Op = msg.operation; x = msg.data; T = msg.tid;
        switch (Op) {
          case T_BEGIN:    send operation to flash data manager(FDM); break;
          case T_READ or T_WRITE:
                find the lock unit lu such that x ⊆ lu;
                if ((lock mode of lu is compatible with Op) or (no lock held)) {
                  set lock on lu in appropriate mode; /* refer to lock table */
                  /* READ: set R-Lock */
                  /* WRITE: set WI-Lock and then W-Lock */
                  send operation to flash data manager;
                  /* READ: read recently committed version */
                  /* WRITE: write and create new version */
                } else
                  wait in queue;
                break;
          case T_ABORT:    send operation to flash data manager; break;
          case T_COMMIT:
                convert WI-Lock to C-Lock; /* delayed until all R-Locks are released */
                send operation to flash data manager;
                break;
        } /* end_switch */
        break;
    case FDMRM: /* a return message from the flash data manager */
```

```
        Op = msg.operation; res = msg.result;      T = msg.tid;
        if (Op==T_ABORT or Op==T_COMMIT) {
          for each lock unit lu locked by T do {
            release lock on lu held by T;
            SOP = first operation from the queue;
            SOP = SOP ∪ {O|O is an operation on queue that can lock lu in compatible mode}
            set the locks on lu;
             for each operation in SOP do
               send each operation to flash data manager;
          }
        } /* end if */
        break;
    } /* end_switch */
  } While (true);
```

〈알고리즘 1〉 의사 코드로 표현된 F2PL의 록 관리 모듈

4. 시뮬레이션 및 성능 평가

본 연구에서 제안된 F2PL의 성능을 검증하기 위하여 컴퓨터 시뮬레이션을 수행하였으며, 실험 결과를 분석해 보았다. 컴퓨터 시뮬레이션 실험에 사용된 비교 기법은 가장 보편적으로 사용하는 S2PL 기법이며, 실험 도구는 CSIM [Schwetman, 1992] 시뮬레이션 언어와 비주얼 C++를 사용하였다. 실험이 수행된 하드웨어 환경은 펜티엄4-2.8 CPU와 메인 메모리 512M, 하드디스크 80G*2이며, 운영체제는 윈도우 2000 서버를 사용하였다.

4.1 시뮬레이션 모델

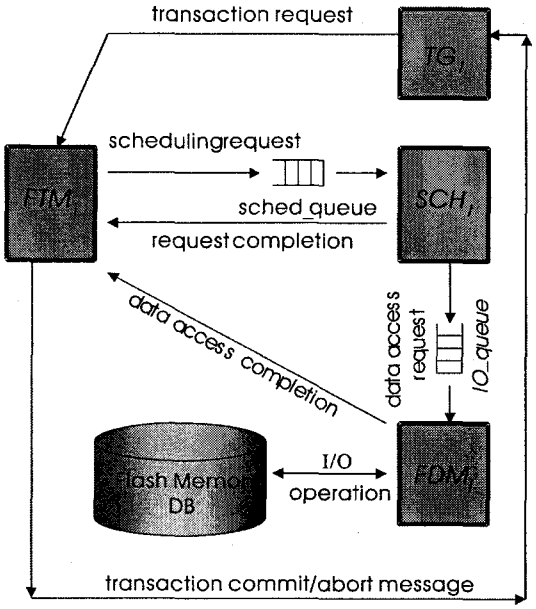
플래시 메모리 데이터베이스 운영 환경을 위한 기본적인 시뮬레이션 모델은 <그림 1>과 같으며, 큐잉 모델에 기반하고 있다. 사용된 기본 큐잉 모델은 CSIM에서 제공되는 폐쇄형 큐잉 모델(Closed Queuing Model)이며, 트랜잭션 생성 큐, 스케줄링 요청 큐, 입출력 큐, 메시지 큐 등이 사용되었다.

시뮬레이션 시스템은 트랜잭션 생성기(Transaction Generator : TG), 플래시 트랜잭션 관리자(Flash Transaction Manager : FTM), 스케줄러(Scheduler : SCH), 플래시 데이터 관리자(Flash Data Manager : FDM)로 구성하였다.

TG는 실험을 위한 트랜잭션을 가상적으로 생성하는 역할을 수행한다. FTM은 TG에서 발생한 트랜잭션을 분석하여 SCH에게 스케줄링을 요청하기 위하여 스케줄링 요청 대기열로 전달한다. SCH는 스케줄링 대기열로 부터 선입선출 방식으로 꺼내어 동시성 제어 알고리즘에 따라서 하나씩 처리한다. 들어온 읽기나 쓰기 연산이 실행가능하면 FDM의 I/O 대기열로 보내고 아니면 대기시킬 것인지 철회시킬 것인지를 결정한다. FDM은 각 연산에서 필요로 하는 객체에 대하여 플래시 메모리 미디어와 관련된 입출력 작업을 수행하고, 수행이 종료되면 완료 메시지를 리턴한다.

TG는 시뮬레이션에 필요한 작업 부하를 만들기 위하여 특정 간격*(inter_arrival_time)*으로 플래시 메모리에 접근하는 사용자 트랜잭션을 생성시킨다. FTM은 사용자 트랜잭션의 생성부

터 종료까지의 수행을 관리한다. 또한, 본 연구의 비교 실험을 위하여 S2PL과 F2PL 알고리즘이 SCH에서 수행된다.



〈그림 1〉 Simulation Model

플래시 메모리 데이터베이스 운영 환경을 위한 시뮬레이션의 주요 성능 평가 지표는 트랜잭션 처리치(throughput)와 응답시간(response time)이다. 트랜잭션 처리치는 초당 몇 개의 트랜잭션이 처리되었는지를 의미하고, 응답시간은 트랜잭션이 발생한 후 수행까지의 지연 시간을 의미한다. 또한, 실험 결과에 대한 세부 분석을 위하여 트랜잭션 철회율(Abort Ratio) 지표도 부수적으로 사용한다.

주요 시뮬레이션 파라메터는 초당 생성된 트랜잭션의 수, 데이터 크기, 읽기 연산 수행시간, 쓰기 연산 수행 시간, 소거 연산 수행시간, 갱신 비율 등이다. 초당 생성된 트랜잭션의 수는 500개에서부터 500개 단위로 3500개까지 변화시켜 보았으며, 이는 시뮬레이션 시스템에 가해지는 작업 부하를 의미한다. 읽기 연산 수행 시간은 36us로 설정하였고, 쓰기 연산 수행 시간은 266us로 설정하였으며, 소거 연산 수행 시간은 2ms로 설정하였다. 각 연산 수행 시간은 기존의 연구[임근수, 2003]에서 제시한 연산 수행에 필요한 실측치이다. 전체 읽기/쓰기 연산의 수에 대한 쓰기 연산의 비율인 갱신 비율은 시스템에 미치는 영향을 분석하기 위하여 20%에서부터 80%까지 10%단위로 변화시켜 보았다. 위의 설정된 주요 파라메터 외의 부수적인 파라메터의 수치는 고정된 값이지만 실험 진행 과정에 필요시 변화시킬 수 있게 하였다.

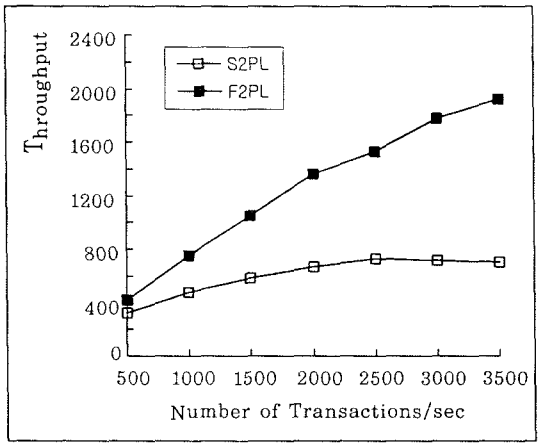
4.2 실험 결과 및 분석

시뮬레이션은 보편적인 트랜잭션 처리 방법인 S2PL기법과 본 연구에서 제안하는 F2PL 기법을 대상으로 수행하였으며, 성능에 민감한 영향을 주는 초당 트랜잭션 수와 갱신 비율 등의 주요 파라메터를 중심으로 하였다.

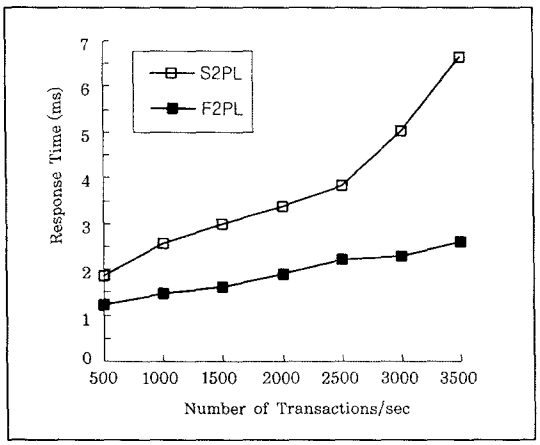
(1) 발생된 트랜잭션 수의 변화에 따른 성능 비교

이 실험은 기본적으로 플래시 메모리를 탑재한 정보기기에서 초당 발생된 트랜잭션의 수가 위의 두 기법들의 성능에 어떤 영향을 미치는지를 분석하기 위한 실험이다. <그림 2>은 초당 발생된 트랜잭션의 수의 증가에 따른 트랜잭션 처리치를 표시한 그래프이다. 발생된 초당 트랜잭션의 수가 늘어날수록 점차로 트랜잭션 처리 결과치가 증가함을 알 수 있다. 또한, 전반적인 트랜잭션 처리 성능을 측정해 보면, S2PL보다 F2PL이 높게 나타났다.

<그림 2>에서 보면, 초당 트랜잭션의 수가 대략 2500개를 넘으면서 S2PL 기법의 성능이 점점 낮아진다. 이는 초당 트랜잭션의 수의 증



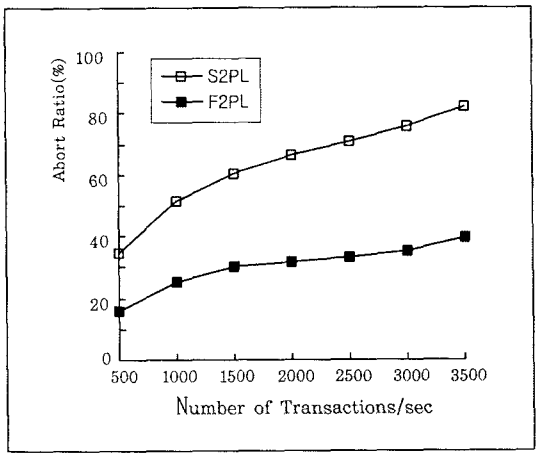
〈그림 2〉 초당 트랜잭션 처리치의 비교



〈그림 3〉 응답 시간의 비교

가에 의한 데이터 연산 집중화가 성능에 영향을 크게 미치는 주요 요소임을 의미하며, 이 이상으로 트랜잭션을 활성화시키는 것이 성능 향상에 도움이 되지 않음을 의미한다. 하지만, 동일한 조건에서도 제안한 F2PL 기법이 S2PL 기법에 비하여 성능이 상대적으로 더 높다. 또한, <그림 3>에서 알 수 있듯이 응답시간도 F2PL 기법이 S2PL 기법에 비하여 더 빠르다. 그 이유는 <그림 4>에서와 알 수 있듯이 F2PL 기법이 S2PL 기법에 비하여 트랜잭션 철회율을 대폭 낮출 수 있었기 때문이다. 즉, 철회되는 트랜잭션이 줄고 정상적으로 완료되는 트랜잭션의 수가 상대적으로 늘어남으로써 전체적인 성능 향상 효과를 볼 수 있었다.

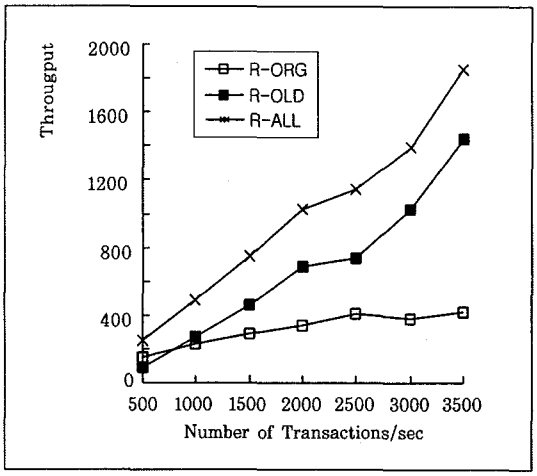
이 실험에서 초당 트랜잭션의 수의 증가하면서 데이터 연산 집중화가 가속화되는데, 특히 시간이 많이 소요되는 쓰기 연산이 크게 부하를 증가시키게 된다. 이 때, F2PL은 쓰기 연산 진행 중에 들어오는 읽기 연산에 다중 버전 읽기를 허용하여 읽기 연산의 철회율을 줄이고 처리 성능을 개선하였다. 또한, 쓰기 연산 전에 플래시 세그먼트에 미리 WI 록을 확보하도록 제어하여 쓰기 연산이 중도에 철회되지 않고 원활하



〈그림 4〉 트랜잭션 철회율의 비교

게 진행되도록 유도함으로써 결과적으로 전체적인 트랜잭션 철회율을 줄일 수 있었다. 이는 그림4의 트랜잭션 철회율 비교를 통하여 검증할 수 있는데, 시스템의 부하가 증가할수록 두 기법 모두 철회율이 증가하지만, 전반적인 구간에서 F2PL 기법이 S2PL 기법에 비하여 상대적으로 철회율이 낮음을 확인할 수 있다.

<그림 5>는 초당 트랜잭션 발생수의 증가에 대한 읽기 트랜잭션만의 성능 변화를 보여주는 그래프이다. R-ORG는 읽기 트랜잭션이 원래

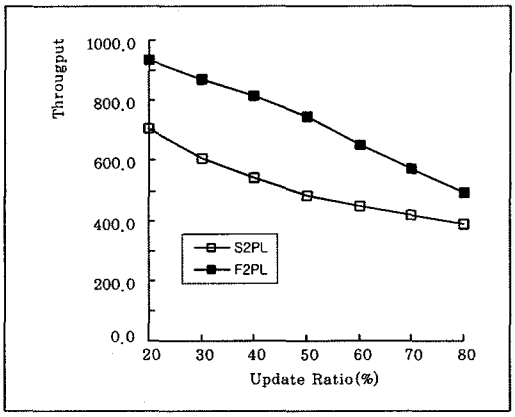


〈그림 5〉 읽기 트랜잭션 처리치의 비교

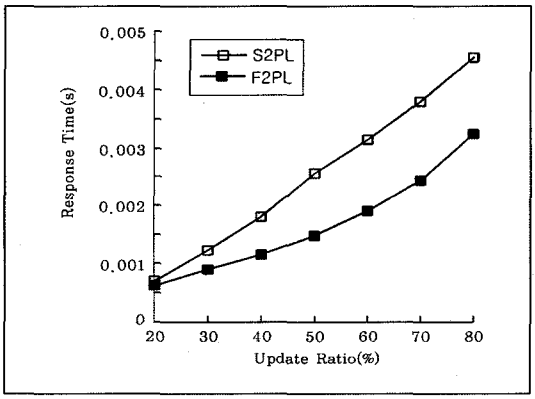
버전의 데이터를 읽은 것이고, R-OLD는 쓰기 트랜잭션이 진행 중인 동안 발생된 읽기 트랜잭션을 철회하지 않고 다중 버전 읽기를 허용하여 구 버전을 읽은 것이다. R-ALL은 R-ORG와 R-OLD을 합한 전체 읽기 트랜잭션의 처리 성능을 나타낸다. 이 <그림 5>에서 살펴보면, 시스템의 부하가 적은 초기의 1000이하 구간에서는 대부분 원래 버전으로 읽기 트랜잭션이 처리되지만, 시스템의 부하가 1000이상으로 증가할수록 다중 버전을 활용한 구 버전 읽기 트랜잭션이 크게 증가함을 알 수 있다. 즉, 시스템 부하가 증가하면서 자연적으로 쓰기 트랜잭션의 수도 증가하게 되는데, 이 경우 읽기 트랜잭션은 과도한 쓰기 트랜잭션의 부하를 피하여 F2PL의 다중 버전 읽기를 효과적으로 활용함을 알 수 있다. 이 다중 버전 활용효과로 인하여 시스템의 부하가 증가하더라도 처리 성능이 저하되지 않고 전체적인 트랜잭션 처리 성능이 실험구간 한도에서 지속적으로 높아짐을 확인할 수 있었다. 전체적인 실험 구간에서 F2PL은 S2PL에 비하여 1.97배의 빠른 응답 성능을 보였으며, 2.08배의 높은 트랜잭션 처리 성능을 보였다.

(2) 갱신 비율 변화에 따른 성능 비교

이 실험은 전체 읽기/쓰기 연산의 수에 대한 쓰기 연산의 비율인 갱신 비율이 위의 두 기법들의 성능에 어떤 영향을 미치는 지를 분석하기 위한 실험이다. 이 영향을 분석하기 위하여 갱신 비율을 최소 20%에서 최대 80%까지 10% 단위로 변화시켜 보았다. <그림 6>은 갱신 비율의 증가에 따른 트랜잭션 처리 성능을 표시한 그래프이며, <그림 7>은 응답 시간을 표시한 그래프이다.



〈그림 6〉 초당 트랜잭션 처리치의 비교

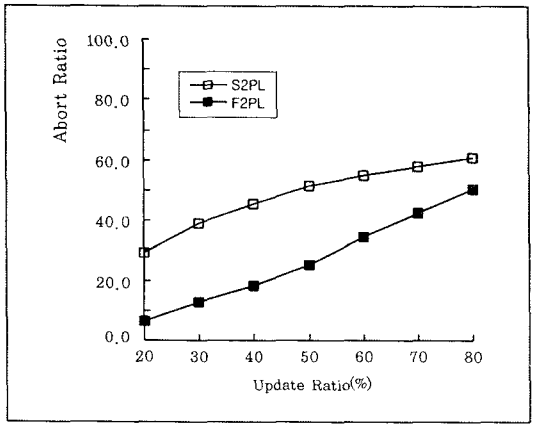


〈그림 7〉 응답 시간의 비교

<그림 6>과 <그림 7>에서 보면 두 기법 모두 갱신 비율이 증가할수록 점차로 트랜잭션 처리

성능도 저하되고, 응답 속도도 매우 느려짐을 알 수 있다. 이유는 시간이 많이 소모되는 저속의 쓰기 연산의 비율이 점차로 증가할수록 읽기-쓰기 및 쓰기-쓰기 트랜잭션간의 충돌이 빈번해면서 트랜잭션 철회가 증가하여, 시스템의 처리 성능과 응답 속도를 크게 저하시키기 때문이다.

이 그래프를 살펴보면, 전반적으로 F2PL이 S2PL에 비하여 트랜잭션 처리 성능이 우수하게 나타났다. 이는 전술한 다중 버전 허용 전략과 쓰기-의도 록 관리의 효과 때문이다. 하지만, 갱신 비율이 60% 이상으로 증가하면서, F2PL이 가지는 두 효과는 점차로 줄어들게 된다. 이유는 <그림 8>에서 보는 바와 같이, 시간을 많이 소모하는 저속의 쓰기 트랜잭션의 증가로 록을 점유하는 시간이 매우 증가하여 과도한 트랜잭션 충돌과 철회가 발생하기 때문이다. 하지만 전반적인 갱신 비율 변화 구간에서 평가해보면, F2PL은 S2PL에 비하여 1.51배의 빠른 응답 성능을 보였으며, 1.41배의 높은 트랜잭션 처리 성능을 보였다.



〈그림 8〉 트랜잭션 철회율의 비교

5. 결론 및 향후 과제

본 논문에서는 소형 정보기기의 데이터 저장 장치로 많이 사용되는 플래시 메모리와 관련된 기존의 데이터 처리 기술을 분석하고, 트랜잭션 처리 성능과 응답 성능을 개선할 수 있는 F2PL 기법을 제안하였다. F2PL 기법은 쓰기 연산과 소거 연산이 상대적으로 매우 느린 플래시 메모리 연산의 단점을 고려하여 록 관리 기법에 반영하였으며, 다중버전 읽기를 허용하여 전반적인 트랜잭션 철회율을 낮추었다. 이로부터 트랜잭션 응답 성능과 처리 성능을 높일 수 있었다.

시뮬레이션 분석 결과 작업 부하가 적은 시작 구간에서는 기존 처리 기법과 비슷하게 나타났지만, 트랜잭션이 집중되어 시스템의 부하가 증가하더라도 처리 성능이 저하되지 않고 기존 기법에 비하여 더 우수하게 나타났다. 전체적인 실험 구간에서 F2PL은 기존 기법에 비하여 1.97배의 개선된 응답 성능을 보였으며, 2.08배의 높은 트랜잭션 처리 성능을 보였다.

향후 연구 과제는 시뮬레이션을 통하여 검증된 트랜잭션 처리 모듈의 구현과 플래시 메모리에 적합한 색인 구조 및 최적화된 하부 시스템 접근 방안의 연구이다. 현재, 휴대용 정보 시스템의 데이터 저장장치로서 대부분 플래시 메모리가 사용된다는 측면에서, 플래시 미디어 기반 데이터 처리 기술은 향후 지속적인 연구가 필요하다고 사료된다.

참 고 문 헌

[1] 민용기, 박승규, "이동컴퓨터를 위한 플래시 메모리 클리닝 정책", *한국통신학회논문지*, 제24권 제5A호, 1999, pp. 657-666.

[2] 이정배, 이두원, "임베디드 시스템 동향", *정보처리* , 제9권 제1호, 2002, pp. 13-27.

[3] 임근수, 고건 "플래시 메모리 기반 저장장치의 설계 기법", *정보과학회 추계 학술대회*, 제30권 제2-1호, 2003, pp. 274-276.

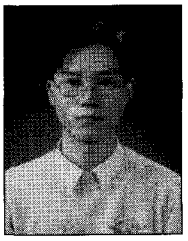
[4] 조주현, "임베디드 실시간 시스템의 개발 환경", *정보처리 논문지*, 제9권 제1호, 2002.

[5] 황규영, 홍의경, 음두현, 박영철, 김진호, *데이터베이스 시스템*, 생능출판사, 2000.
[6] Ammann A.C., Hanrahan M.B., and R. Krishnamurthy, "Design of a memory resident DBMS", *in Proc. IEEE COMPCOM Conf.* 1985.
[7] Bernstein P., Hadzilacos V., and Goodman N. *Concurrency control and recovery in database systems,* Addison-Wesley, 1987.
[8] Eich M. H., "A classification and comparision of main memory database recovery techniques", *in Proc. Int. Con. On Data Engineering*, Feb. 1987, pp. 332-339.
[9] Garcia-Molina H. and Salem K., "High performance transaction processing with memory resident data", *in Proc. Int. Workshop on High Performance Transaction Systems*, Paris, Dec. 1987.
[10] Garcia-Molina H. and Salem K., "Main Memory Database Systems : An Overview", *IEEE Trans. Knowl. Data Eng.*, Vol. 4, No. 6, Dec. 1992, pp. 509-516.
[11] Lehman T.J. and Carey M.J., "Query processing in main memory database management systems", in Proc. *ACM SIGMOD Conf.*, Washington, DC, May, 1986.
[12] Schwetman H., *CSIM User's Guide for Use with CSIM Revision 16*, Microelectronics and Computer Technology Corporation, 1992.
[13] Tamer O. and Patrick Valduriez, *Principles of Distributed Database Systems,* Prentice Hall, 1991.
[14] 유제정, "*Mobile Database*란?", http://www.mobilejava.co.kr/bbs/temp/lecture/j2me/ mdb1.html, 2004.

저자소개

변 시 우

저자는 연세대학교 전산과학과를 졸업하고, 한국과학기술원에서 전산학 석사와 박사 학위를 취득하였다. 현재 안양대학교 디지털미디어 학부의 조교수로 재직하고 있으며, 주요 관심분야는 모바일 컴퓨팅, 분산데이터베이스, 휴대용 데이터베이스, 전자상거래 분야이다.



노 창 배

저자는 대전대학교 컴퓨터공학과를 졸업하고, 한남대학교에서 교육석사와 경희대학교에서 전파공학 박사 학위 과정에 있다. 현재 안양대학교 디지털미디어 학부의 겸임교수로 재직하고 있으며, 주요 관심분야는 모바일 컴퓨팅, 휴대인터넷 Picocell, WLAN, RFID, 홈네트워크 분야이다.

정 명 희

저자는 서울대학교 계산통계학과를 졸업하고, university of Texas at Austin에서 석사와 박사 학위를 취득하였다. 현재 안양대학교 디지털미디어 학부의 조교수로 재직하고 있으며, 주요 관심분야는 원격탐사, XML, 영상처리, 의료 정보 분야이다.

◈ 이 논문은 2005년 8월 28일 접수하여 2차 수정을 거쳐 2005년 11월 8일 게재확정되었습니다.