휴대용 데이터베이스를 위한 지연된 소거 리스트를 이용하는 플래시 메모리 쉐도우 페이징 기법

변 시 우*

Flash Memory Shadow Paging Scheme Using Deferred Cleaning List for Portable Databases

Siwoo Byun*

Abstract

Recently, flash memories are one of best media to support portable computer's storages in mobile computing environment. We propose a new transaction recovery scheme for a flash memory database environment which is based on a flash media file system. We improved traditional shadow paging schemes by reusing old data pages which are supposed to be invalidated in the course of writing a new data page in the flash file system environment. In order to reuse these data pages, we exploit deferred cleaning list structure in our flash memory shadow paging (FMSP) scheme. FMSP scheme removes the additional storage overhead for keeping shadow pages and minimizes the I/O performance degradation caused by data page distribution phenomena of traditional shadow paging schemes. We also propose a simulation model to show the performance of FMSP. Based on the results of the performance evaluation, we conclude that FMSP outperforms the traditional scheme.

Keywords : Portable Information Devices, Page Management, Shadow Paging, Flash Memory, Database Recovery, Simulation

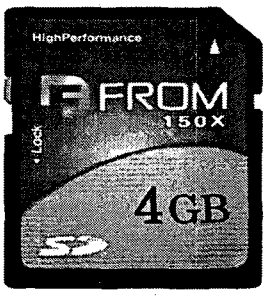
논문접수일 : 2005년 6월 29일 논문게재확정일 : 2006년 6월 14일

※ 본 연구는 한국학술진흥재단의 지역대학연구(D00557(R05-2004-000-10961-0))지원으로 수행되었음.

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

1. 서 론

고속의 인터넷 컴퓨팅의 급속한 보급과 더불어, 최근에는 PDA, 포켓 PC, 자동차 네비게이션, 휴대폰, 스마트폰 등의 플래시 메모리를 사용하는 각종 소형 정보 단말기들이 대중화되게 되었다. 초기 플래시 메모리는 1985년 IEEE의 ISSCC(International Solid State Circuit Conference)에서 256k bit의 플래시가 발표되었다. 추후 발전을 거듭하고 있으며(<그림 1>), 향후에는 하드 디스크와 메인 메모리를 대체하여 거대한 저장 장치 시장을 형성하리라 예측된다. 이와 관련한 기술 분야 중에서 플래시 미디어 기반의 데이터 관리 연구는 최근에 시작된 유망한 연구 분야이다[변시우, 2004].



<그림 1> 플래시 메모리 제품

본 논문은 최근 휴대용 소형 정보기기의 데이터 저장 장치로 많이 사용되는 플래시 메모리 파일 운영 시스템의 특성을 효과적으로 활용한 새로운 데이터베이스 회복 기법을 제안한다.

2. 관련 연구

일반적으로 데이터베이스 시스템에서 회복이란 데이터베이스가 물리적 파손과 같은 1) 재해적 고장 또는 잘못된 운영으로 인하여 일관성이 파괴되는 2) 비재해적 고장이 발생했을 때, 데이터베이스를 고장 발생 이전의 상태로 복원시키는 기능을 의미한다. 첫째, 재해적 고장인 경우는 주로 테이프와 같은 저장장치에 복사된 과거의 사본들을 사용하여 복구하게 된다. 둘째, 비재해적 고장인 경우는 일부 연산들을 redo/undo 함으로써, 운영 도중에 트랜잭션의 수행이 실패한 시점으로부터 가장 가까운 과거의 안정된 데이터베이스 상태로 데이터를 복구하는 것이다. 데이터베이스 시스템에서는 트랜잭션 관리자의 하부에 데이터 관리자가 연동되어 있으며, 데이터 관리자는 내부에 회복 관리자를 거쳐서 실제 스토리지에 접근하므로, 평상시에도 회복 기능이 원활히 작동되어야 한다. 이를 위하여 데이터베이스 관리 시스템은 트랜잭션이 수행되는 동안 데이터 변경에 대한 정보를 지속적으로 유지하여야 한다[황규영 외 3인, 2000].

트랜잭션의 수행 실패와 같은 비재해적 고장에서 데이터베이스를 회복하는 기법은 크게 즉시 갱신(update-in-place) 기법[Vijay and Albert, 1992]과 쉐도우 페이징(shadow paging) 기법[Matthew and John 1983; Jack and Hector, 1988]으로 나누어진다. 즉시 갱신 기법은 수정된 데이터 항목을 디스크와 같은 저장 장치로 반영(flush)할 때, 같은 위치에 직접 기록하여 이전의 데이터를 덮어쓰므로, 각 데이터 항목에 대하여 하나의 사본이 존재하며, 회복을 위한 로그들을 반드시 유지해야 한다. 반면에, 쉐도우 페이징 기법에서는 새로 저장되는 데이터 항목은 디스크 저장 장치의 다른 위치에 저장하므로 같은 데이터 항목에 여러 개의 사본이 존재 가능하다. 여기서, 데이터 항목이 수정되기 이전의 값을 '이전 값(before image)'이라고 지칭하고, 수정 후의 새 값을 '이후 값(after image)'이라고 지칭한다. 쉐도우 페이징 기법은 이전 값과 이후 값을 모두 디스크에 보관하므로 즉시 갱신 기법과는 달리 회복을 위하여 로그를 유지

할 필요가 없는 장점이 있다.

일반적으로 즉시 갱신 기법은 redo/undo 로그 저장에 따른 오버헤드로 시스템의 성능 저하가 공통적으로 발생하는 반면에, 쉐도우 페이징 기법에 비하여 저장 공간이 적게 소모되는 장점이 있다. 쉐도우 페이징 기법은 트랜잭션이 성공적으로 수행 완료될 때까지 다른 영역에 수정분을 보관하고 있다가 완료 시점에 수정분이 데이터베이스에 반영된다. 따라서 오손 페이지(dirty page)가 발생하지 않고, 트랜잭션의 redo/undo 연산이 불필요하므로 관련된 로그의 부담이 없으며, 트랜잭션 실패 후 회복 작업이 매우 간편하다. 반면에 쉐도우 페이지를 보관하기 위한 많은 저장 공간이 필요하며, 이를 효율적으로 관리하기 위한 시스템 오버헤드가 수반된다. 또한, 갱신된 데이터베이스의 페이지가 디스크 공간상에서 위치가 자주 바뀌므로, 연관된 페이지가 집중되지 않고 분산되므로 디스크 입출력 성능이 저하되고, 결과적으로 데이터베이스 시스템의 처리 성능이 저하된다[Song et. al., 1999; Choi et. al., 2000].

기존의 디스크-기반의 상용 데이터베이스 시스템이나 주 메모리-기반 데이터베이스 시스템은 오래 전부터 인기 있는 연구 분야로서 최근에도 활발한 연구가 지속되고 있다. MARS, HALO, OBE, MM-DBMS, System M 등의 시스템이 기본적인 과거의 연구 사례이다[Garcia and Salem, 1992]. 그러나 순수하게 플래시 메모리 기반의 데이터베이스 시스템의 연구는 시도된 적이 거의 없는 분야이다. 그 이유는 과거 플래시 메모리는 주 메모리의 성능과 비교하면 느리면서도 고가이며, 일반적인 시스템에는 사용되지 않는 특수 부품이었으며, 디스크에 비하여 저장 비용이 고가였었기 때문이다. 그러나 최근 플래시 메모리 기술의 발전으로 대부분의 단점들이 해소된 상태이며, 이제는 저렴하고 성능 좋은 미래의 저장 메모리로서 크게 각광 받고 있다. 특히, 대부분의 소형 기기에서는 공간 제약, 전력소모, 중량 및 내충격성 문제로 디스크는 사용할 수 없기 때문에, 비휘발성 저장장치로서 플래시 메모리를 반드시 사용하여야만 한다[임근수, 고건, 2003]. 다만, 일반 메인 메모리와는 달리, 쓰기와 지우기 연산에 10배 정도의 많은 시간이 소요되며, 쓰기 횟수가 100,000번 정도로 제한된다는 특성을 가만하여, 효과적인 휴대형 정보 기기의 데이터 처리 모듈들을 개발하여야 한다.

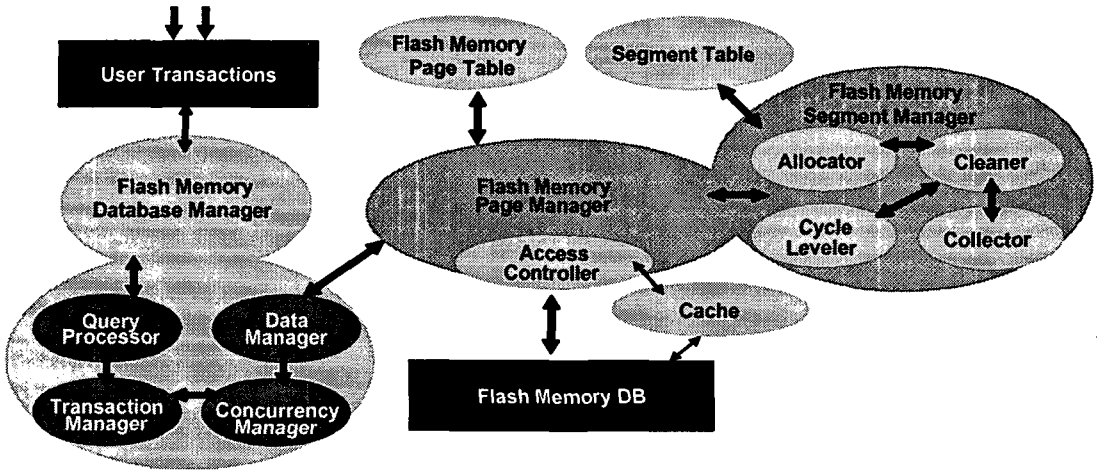
본 논문에서는 이러한 플래시 메모리의 특성을 고려하여, 플래시 메모리 데이터베이스의 효과적인 트랜잭션 회복을 위한 새로운 쉐도우 페이징 기법을 제안한다.

3. 플래시 메모리 기반 쉐도우 페이징 기법의 제안

본 연구에서는 NAND형 플래시 메모리를 데이터 처리기의 기본 저장장치로 설정하였다. 왜냐하면, NAND형 플래시는 NOR형 플래시 메모리에 비하여 절반 정도로 가격이 저렴하며, 소거 연산 속도가 빠르며, 쓰기 연산 속도가 더 우수하기 때문이다[최리군, 2001].

3.1 플래시 메모리 접근 및 처리 구조

플래시 메모리 데이터의 시스템 처리 구조를 더 구체적으로 살펴보면, <그림 2>와 같이 플래시 메모리 데이터베이스 관리자, 플래시 메모리 페이지 관리자, 플래시 메모리 세그먼트 관리자를 근간으로 구성된다. 플래시 메모리 데이터베이스 관리자는 사용자로부터 트랜잭션 처리 요청을 받아서, 트랜잭션 관리자, 쿼리 처리자, 동시성 관리자, 데이터 관리자의 도움으로



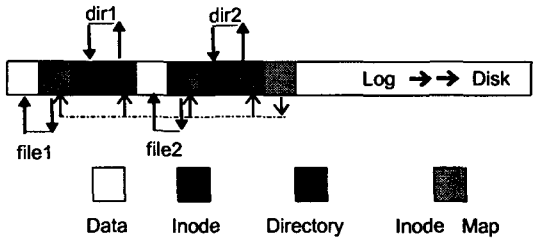
〈그림 2〉 플래시 메모리 데이터베이스 시스템구조도

트랜잭션의 수행 완료까지의 전체적인 과정을 담당한다. 플래시 메모리 페이지 관리자는 플래시 메모리 매핑 테이블과 세그먼트 테이블을 관리하며, 플래시 접근 제어기를 통하여 요청된 데이터가 플래시 메모리 데이터베이스 저장소에 안전하게 저장되도록 관리한다.

플래시 메모리 세그먼트 관리자는 부수적으로 allocator, cleaner, cycle leveler, collector의 4개의 모듈을 포함한다. Allocator는 프리 세그먼트들의 풀(Free Segment Pool)을 유지하며, 어떤 세그먼트가 다음에 할당되어야 하는지를 결정한다. Cleaner는 프리 세그먼트(페이지)를 생성하기 위하여 무효화된 페이지를 제거하는 작업을 수행한다. Cycle Leveler는 전체적인 플래시 메모리의 세그먼트들이 균등하게 소거되도록 조절하는 작업을 수행한다. Collector는 콜드 데이터와 핫 데이터를 분리하는 작업을 수행하여 Cleaner의 작업 부하를 경감시켜준다[Chang and Kuo, 2002].

3.2 플래시 메모리 파일 시스템의 구조

전술한 것과 같이 플래시 메모리는 지우고 쓰는 횟수가 상온에서 10만 번으로 제한되어 있으므로, 어느 특정 블록에 집중하여 사용하면, 수명이 다하여 나머지 다른 부분도 사용할 수 없게 된다. 따라서 전체 영역에 걸쳐서 비슷한 비율로 데이터 기록이 분산되어야 한다. 이러한 특성 때문에 플래시 메모리를 사용하는 파일 시스템에서는 로그-구조 파일 시스템(Log-Structured File System : LFS)[Mendel and John, 1992; Jeanna et al., 1987]을 기본으로 응용하여 사용한다. LFS에서는 데이터를 로그 형태로 순차적으로 기록하고, 읽기의 경우에는 로그를 역순으로 검색하여 가장 최신의 데이터를 읽는다. 즉, 수정된 데이터를 현재 저장된 블록에 기록하지 않고, 다른 빈 블록에 기록하고 기존의 데이터가 저장된 블록은 무효 플래그를 설정하여 사용하지 않는 방식을 쓰고 있다.



〈그림 3〉 Log-Structured File System

플래시 메모리를 사용하는 임베디드 시스템 분야에서 가장 많이 사용하는 파일 시스템은 JFFS(Journaling Flash File System)[JFFS, 2004]와 최근의 JFFS2[JFFS2, 2004]이다. JFFS는 스웨덴의 Axis Communications에서 개발된 LFS 기반 파일 시스템이며, 플래시 메모리의 제약점을 극복하여 복구능력을 향상시켰다[권우일 외 2인, 2004; 박성호, 정기동, 2002]. JFFS는 일반적인 파일 시스템을 개선하여 플래시 메모리의 특성을 고려한 파일 시스템이다. 기존의 파일 시스템이 메타 정보를 특정 블록에 고정하고 파일 시스템이 수정될 때마다 해당 블록을 갱신하던 것을 JFFS는 플래시 메모리의 특성에 맞게 특정 블록을 고정해 사용하지 않고 플래시 메모리 전체 영역을 고르게 사용할 수 있게 한다. 또한 비동기적 전원 실패에 대비하여 데이터 손실이 없도록 일정 시간마다 전체 파일 시스템의 메타 정보를 저장한다.

본 논문에서는 새로운 블록(페이지)에 데이터를 저장할 경우 무효화되어 폐기되는 이전 데이터 블록을 활용하여 데이터베이스 시스템의 성능을 향상시키고 안정성을 높이는 기법을 제안한다.

3.3 지연된 소거 리스트를 활용한 플래시 메모리 쉐도우 페이징(FMSP) 기법

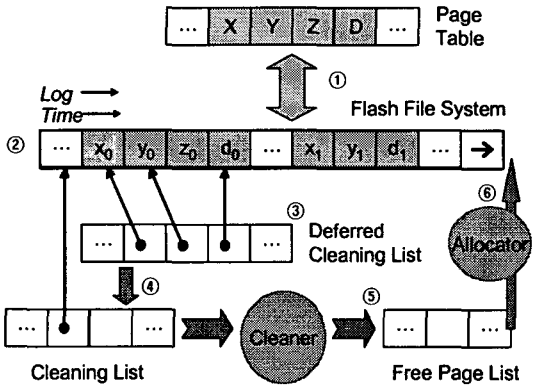
쉐도우 페이징 기법은 트랜잭션이 실행을 시작할 때, 현재 페이지 테이블을 쉐도우 페이지 테이블로 복사한다. 현재 페이지 테이블의 목록은 가장 최근의 데이터베이스 페이지들을 가리키며 트랜잭션이 수행되면서 수정되는 반면, 쉐도우 페이지 테이블은 트랜잭션이 실행되는 동안에 절대 수정되지 않고 유지된다. 즉, 쓰기 연산 실행시 새로운 페이지 사본이 생성되지만 그 페이지의 이전 사본을 덮어쓰지 않는다. 따라서 트랜잭션이 갱신한 페이지들에 대하여 두 개의 버전이 유지되는데, 이전 버전은 쉐도우 페이지 테이블이 참조하고 새 버전은 현재 페이지 테이블이 참조한다.

만일 데이터베이스 운영중에 트랜잭션 실패가 발생하여 복구해야 한다면 현재 페이지 테이블을 버리고 이전의 쉐도우 페이지 테이블을 현재 페이지 테이블로 교체하면 된다. 반대로 트랜잭션이 성공적으로 수행되어 정상적으로 완료되면 쉐도우 페이지 테이블을 더 이상 필요가 없으므로 폐기하면 된다. 이 과정이 쉐도우 페이징 기법의 기본적인 운영 방식이며 redo/undo 연산이 없으므로 no-redo/no-undo 기법으로 분류된다[황규영 외 4인, 2000].

본 논문에서 제안하는 플래시 메모리 쉐도우 페이징(Flash Memory Shadow Paging : FMSP) 기법은 플래시 메모리 데이터베이스의 특성을 고려하여 최적화한 트랜잭션 회복 기법이다. 즉, 기존의 쉐도우 페이징 기법의 큰 장점인 간편한 처리 구조는 유지하되, 큰 단점인 쉐도우 페이지를 저장하기 위한 별도의 저장 공간 오버헤드를 제거하여 데이터베이스 시스템의 성능과 안정성을 향상시키는 것을 목표로 한다.

이를 위하여 앞 절에서 설명한 플래시 파일 시스템의 특성을 효율적으로 활용해야 한다. 즉, 플래시 메모리의 물성적인 특성을 보완하기 위하여 거의 대부분의 플래시 파일 시스템이 사용하는 LFS 기반 방식에서 한 번 저장 후 폐기되는 데이터 블록(페이지)을 재활용해야 한다. 전술한 바와 같이 가장 보편적인 JFFS와 같은 LFS 기반 플래시 파일시스템에서는 새로운 블록에 데이터를 저장할 경우 이전 데이터 페이지는 무효화된 후 폐기된다. 이러한 무효화된 페이지의 재활용을 위해서는 기존 파일 시스템과의 연동, 지연된 소거 리스트(Deferred Cleaning List) 구조, 트랜잭션 처리 과정의 개선이 필요

하다. <그림 4>는 제안된 플래시 메모리 데이터베이스를 위한 플래시 메모리 쉐도우 페이징 기법의 구조를 나타내며, 처리 프로시저에 대한 설명은 다음과 같다.



<그림 4> 플래시 메모리 데이터베이스를 위한 쉐도우 페이징 구조

① 플래시 메모리 데이터베이스 시스템에서 트랜잭션이 수행되면, 페이지 테이블을 통하여 플래시 파일 시스템에 저장된 데이터 항목에 접근한다.

② 플래시 파일 시스템은 데이터 항목에 대하여 과거의 쉐도우 페이지를 덮어 쓰지 않고, 새로운 현재 페이지에 수정된 값을 저장한다. 데이터 항목 저장시 페이지 할당자(Allocator)로부터 순차적 방향으로 페이지를 할당받아서 사용하게 된다.

③ 데이터 항목에 대하여 가장 최근에 완료된 페이지는 지연된 소거 리스트에서 유지되어 쉐도우 페이지로 활용되며, 트랜잭션이 수행 완료시까지 소거 연산이 보류된다.

④ 트랜잭션이 수행 완료되어 쉐도우 페이지가 불필요하게 되면 소거 리스트로 등록된다.

⑤ 주기적으로 페이지 소거자(Cleaner)에 의하여 소거 리스트에 등록된 페이지는 세그먼트 소거 과정을 거친 후 프리 페이지 리스트에 삽입된다.

⑥ 프리 페이지 리스트에 삽입된 페이지는 페이지 할당자를 통하여 플래시 파일 시스템에 새로운 페이지를 공급하게 된다.

여기서 폐기되는 페이지의 활용효과는 플래시 메모리 파일 시스템이 데이터들로 완전히 차지 않고, 어느 정도 여유 공간이 있다는 가정을 전제하고 있다. 예를 들어 파일 시스템 풀이 발생한다면, 쉐도우 페이지로 갈 여유 블록이 없으며, 바로 클리닝 과정을 거쳐서 새로운 블록으로 할당시켜야 하기 때문에 페이지 활용효과는 없어지고, 오히려 오버헤드만 증가한다. 다만, 일반적으로 파일 시스템은 최소 10% 정도는 여유 공간을 유지하는 것이 보통이므로 본 논문에서는 평상시의 경우를 가정하였다.

제안된 FMSP 기법을 적용하면 갱신 연산을 로깅 오버헤드 없이 처리 가능하다. 또한, 로깅에 대한 지연과 체크포인팅 부담이 제거되므로 결과적으로 트랜잭션의 응답성능과 처리 성능을 높일 수 있다. 앞 절에서 설명한 쉐도우 페이징 기법의 두 단점을 재검토해 보자.

제안된 FMSP 기법을 적용하면 갱신 연산을 로깅 오버헤드 없이 처리 가능하다. 또한, 로깅에 대한 지연과 체크포인팅 부담이 제거되므로 결과적으로 트랜잭션의 응답성능과 처리 성능을 높일 수 있다. 앞 절에서 설명한 쉐도우 페이징 기법의 두 단점을 재검토해 보자.

첫째, 기존 기법에서는 쉐도우 페이지를 보관하기 위한 많은 저장 공간이 소모된다는 단점이 있다. 하지만, FMSP 기법에서 이 단점은 플래시 파일 시스템에서 버려지는 과거 데이터 페이지를 쉐도우 페이지로 재활용하기 때문에 추가적인 공간이 낭비되지 않는다. 따라서 쉐도우 페이지를 위한 공간 부담이 제거되어 이 단점은 극복된다.

둘째, 디스크 기반 파일 시스템 환경에서는 쉐

도우 페이징이 진행될수록 갱신된 데이터 페이지가 디스크 저장 공간상에서 페이지 위치가 자주 바뀌는 단점이 있다. 이는 연관된 페이지가 집중되지 않고 넓게 분산되므로 입출력 성능이 저하되고, 결과적으로 데이터베이스 시스템의 처리 성능이 저하된다. 그러나 FMSP 환경에서는 플래시 메모리를 저장 장치로 사용하므로 메인 메모리와 같은 순간적인 임의 접근이 가능하다. 따라서 데이터 페이지 분산에 의한 입출력 성능 저하가 발생하지 않으므로 이 단점도 극복된다.

결과적으로 제안된 FMSP 기법은 플래시 파일 시스템에서 버려지는 데이터 페이지를 재활용함으로써 기존 쉐도우 페이징의 단점을 제거하고 플래시 메모리 데이터베이스 시스템의 효율과 성능을 높일 수 있다. 이를 위하여 기존 플래시 파일 시스템과 쉐도우 페이징 기법이 서로 연동하는 추가적인 인터페이스 부담과 지연된 소거 리스트 구조를 위한 추가적인 저장 공간의 부담이 발생한다. 하지만, 플래시 메모리 쉐도우 페이징 기법의 적용으로 수반되는 트랜잭션 처리 성능과 의 저장 효율 향상을 고려하면 그 추가 부담의 정도는 작다고 판단된다.

3.4 제안 기법의 트랜잭션 처리 연산

지연된 소거 리스트(DCL)를 활용한 플래시 메모리 쉐도우 페이징 기법을 적용하여 트랜잭션을 수행하기 위하여, *OP_WRITE, OP_READ, OP_COMMIT, OP_ABORT, OP_RESTART* 처리 연산이 필요하며, 그 내용은 다음과 같다.

- *OP_WRITE(Tr, x, val)* : 데이터 항목 *x*에 *val*값을 쓰는 쓰기 트랜잭션 *Tr* 실행.
  ① 플래시 메모리의 새로운 데이터 페이지에 데이터 항목 *x*에 대한 *val*값을 저장한다. 즉, 데이터 항목 *x*에 대한 최신 버전을 생성하고, 지연된 소거 리스트에 보관하고 있는 이전 버전인 쉐도우 페이지는 그대로 유지한다.
  ② 저장한 새 버전의 데이터 페이지의 위치 주소를 *x*에 대한 참조 주소로 기록한다.
  ③ 스케줄러에게 쓰기 연산 처리에 대한 ack (승인) 메시지를 보낸다.
- *OP_READ(Tr, x)* : 데이터 항목 *x*의 값을 판독하는 읽기 트랜잭션 *Tr* 실행.
  ① 트랜잭션 *Tr*이 이미 데이터 항목 *x*에 쓰기 연산을 수행하였다면, 현재 *x*에 대한 쓰기 참조 위치에 저장된 새 버전의 값을 스케줄러에게 보낸다.
  ② 위의 경우가 아니면, 현재 데이터베이스에 저장된 데이터 값을 스케줄러에게 보낸다.
- *OP_COMMIT(Tr)* : 트랜잭션 *Tr*에 대한 커미트 실행.
  ① 트랜잭션 *Tr*이 수정한 모든 데이터 항목에 대하여 데이터 참조 위치가 쉐도우 버전 페이지에서 현재 새로운 버전 페이지로 변경한다.
  ② 트랜잭션 *Tr*과 관련되어 있는 쉐도우 버전 페이지를 지연된 소거 리스트에서 제거한 후 프리 페이지 리스트로 만들기 위해 소거 리스트로 삽입한다.
  ③ 스케줄러에게 트랜잭션 *Tr*의 커미트 처리에 대한 ack 메시지를 보낸다.
- *OP_ABORT(Tr)* : 트랜잭션 *Tr*에 대한 철회 실행.
  ① 트랜잭션 *Tr*과 관련되어 있는 현재 새 버전 페이지를 소거 리스트로 보내어 추후 폐기한다.
  ② 트랜잭션 *Tr*과 관련되어 있는 쉐도우 버전 페이지를 지연된 소거 리스트에서 제거한 후 현재 데이터 페이지로 변경한다.
  ③ 트랜잭션 *Tr*과 관련하여 할당된 리소스를 반납하고, 스케줄러에게 트랜잭션 *Tr*에 대

한 철회 처리에 대한 ack 메시지를 보낸다.

- *OP_RESTART(Tr)* : 트랜잭션 *Tr*에 대한 재시작 실행.
  ① 시스템을 재시작 한다.
  ② 트랜잭션 *Tr*과 관련되어 있는 데이터 페이지를 메모리로 로드 한다.
  ③ 스케줄러에게 트랜잭션 *Tr*의 재시작 처리에 대한 ack 메시지를 보낸다.

4. 시뮬레이션 및 성능 평가

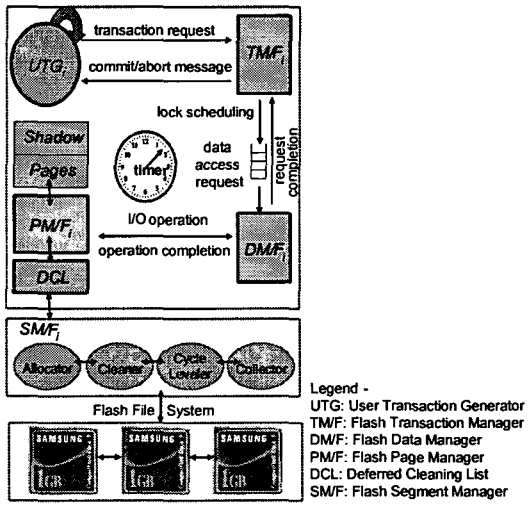
본 컴퓨터 시뮬레이션 실험에서 사용된 비교 기법은 전술한 GSP 기법이며, 실험 도구는 CSIM[Schwetman, 1992]시뮬레이션 언어와 비주얼 C++를 사용하였다. 실험이 수행된 하드웨어 환경은 펜티엄4-2.8 CPU와 메인 메모리 512M, 하드디스크 80G*2이며, 운영체제는 윈도우 2000 서버를 사용하였다.

4.1 실험 모델

본 실험의 플래시 메모리 데이터베이스 운영 환경을 위한 기본적인 시뮬레이션 모델은 그림 5와 같으며, 큐잉 모델에 기반하고 있다. 사용된 기본 큐잉 모델은 CSIM에서 제공되는 폐쇄형 큐잉 모델(Closed Queuing Model)이며, 트랜잭션 생성 큐, 트랜잭션 처리 큐, 읽기-쓰기 큐, 소거 큐, 입출력 큐, 메시지 큐 등이 사용되었다.

본 시뮬레이션 시스템은 사용자 트랜잭션 생성기(User Transaction Generator : UTG), 플래시 트랜잭션 관리자(TM/F), 플래시 데이터 관리자(DM/F), 플래시 페이지 관리자(PM/F), 플래시 세그먼트 관리자(SM/F)로 구성된다.

사용자 트랜잭션 생성기는 시뮬레이션에 필요한 작업 부하를 만들기 위하여 특정 간격으로 플래시 메모리에 접근하는 사용자 트랜잭션을 생성시킨다. 플래시 트랜잭션 관리자는 사용자 트랜잭션의 생성부터 종료까지의 수행을 관리하며, 트랜잭션을 분석하여 플래시 데이터 관리자의 데이터 접근 요청큐에 보낸다. 플래시 데이터 관리자는 논리적인 데이터 접근을 수행한 후, 실제 물리적인 접근 요청은 플래시 페이지 관리장에게 보낸다. 플래시 페이지 관리자는 쉐도우 페이지를 포함한 플래시 메모리 매핑과 지연된 소거 프로시저를 통하여 입출력 요청을 수행한다. 플래시 세그먼트 관리자는 플래시 메모리의 할당 및 수명 관리 업무를 수행한다.



〈그림 5〉 컴퓨터 시뮬레이션 모델

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. 보조 지표로서 쉐도우 대역폭(shadow bandwidth)를 사용하는데, 트랜잭션 실행에 수반되는 쉐도우 페이지와 관련 정보를 저장하고 관리하는 과정에서 발생되는 데이터의 초당 전

송률이다. 즉, 쉐도우 대역폭이 더 크다는 것은, 해당 쉐도우 페이징 기법이 쉐도우 페이지 저장 및 관리상의 작업부하가 더 크다는 것을 의미한다. <표 1>은 주요 시뮬레이션 파라메터를 보여준다. 각 연산의 수행 시간은 기존의 연구[임근수, 고건, 2003]에서 제시한 연산 수행에 필요한 실측치이다.

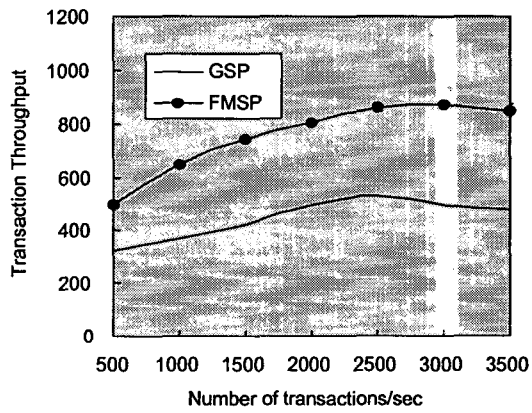
〈표 1〉 주요 시뮬레이션 변수

System Parameter	Description	Value
num_TRs	Number of transactions per second	500~3500 in steps of 500
cpu_delay	CPU time for accessing an data object	5μsecs
Flash_read_delay	I/O time for reading an object in flash memory	36μsecs
Flash_write_delay	I/O time for writing an object in flash memory	266μsecs
Flash_erase_delay	I/O time for erasing an object in flash memory	2 msecs
Flash_page_size	Page size in flash memory	512 Bytes
DC_object_size	Object size in a deferred cleaning list	20 Bytes

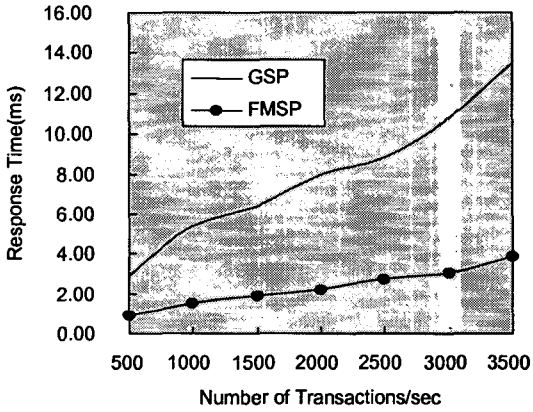
4.2 결과 및 분석

실험을 통하여 두 쉐도우 페이징 기법, FMSP와 GSP의 성능을 분석하였다. 초당 발생되는 트랜잭션의 수(*num_TRs*)를 변화시킴으로써 두 기법의 작업부하의 변화를 살펴보았다. 실험 결과로 나타난 트랜잭션 처리치의 변화 그래프는 <그림 6>에, 평균 응답시간의 변화 그래프는 <그림 7>에 나타나 있다.

<그림 6>에서, 트랜잭션의 발생량이 증가할수록 점차로 큐잉 시스템에 들어 오는 트랜잭션의 수가 많아지므로, 점차로 작업부하가 증가하며, 트랜잭션 처리치도 증가함을 알 수 있다. 실험 결과는 전체 구간에서 FMSP 기법의 그래프가 GSP 그래프 보다 상단에 위치하므로, 트랜잭션 처리 결과가 더 우수함을 알 수 있다. <그림 7>에서도 트랜잭션의 초당 발생량이 증가할수록, 작업부하가 늘어나므로 응답시간도 서서히 증가함을 알 수있다. 두 기법의 그래프를 비교해 보면 전체 구간에서 평균 응답시간이 FMSP 기법이 GSP 기법보다 더 빠름을 알 수 있다.



〈그림 6〉 트랜잭션 처리치의 변화

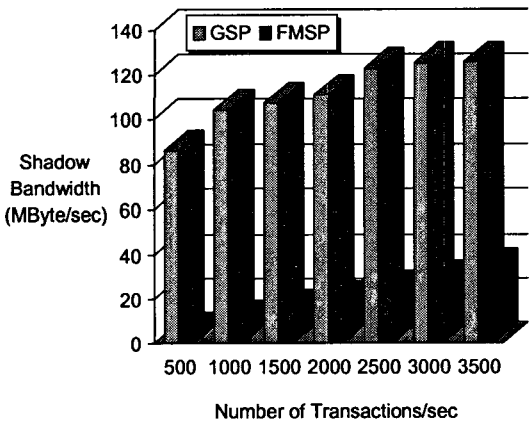


〈그림 7〉 평균 응답 시간의 변화

<그림 6>에서 살펴보면, 두 기법 모두 초당 트랜잭션 발생량이 2500개를 넘으면서 트랜잭션 처리치가 더 이상 증가하지 않고 서서히 처리 성능이 감퇴됨을 알 수 있다. 즉, 2500개 이

상으로 시스템에 트랜잭션 처리를 요청하여도, 이 이상으로는 시스템이 처리할 수 없고, 오히려 데이터 처리 경합이 발생하여 무의미하게 작업부하만 증가시킴을 의미한다.

<그림 6>과 <그림 7>의 그래프의 분석결과로 FMSP가 GSP가 더 높은 트랜잭션 처리치와 빠른 응답 성능을 보였다. 이는 FMSP가 플래시 파일 시스템 내부에서 무효화되어서 폐기되는 데이터 페이지들을 지연된 소거 프로시저를 통하여 쉐도우 페이지로 재활용함으로써 저장 공간 부담과 관련 처리 시간을 감소시켰기 때문으로 분석된다. 이 분석은 <그림 8>의 쉐도우 대역폭의 변화 그래프를 살펴봄으로써 재확인된다. 그림 8의 전체 구간에서 FMSP가 GSP가 훨씬 더 적은 양의 쉐도우 대역폭을 사용하고 있다. 평균값으로 환산하면, FMSP가 GSP에 비하여 약 82%정도 낮은 쉐도우 대역폭을 사용하고 있다. 이는 쉐도우 페이지 저장 및 관리에 부하가 더 적음을 의미하며, 결과적으로 플래시 메모리 데이터베이스의 데이터 처리 효율성을 증가시키게 된다.



〈그림 8〉 쉐도우 대역폭의 비교

5. 결론 및 향후 과제

본 논문에서는 플래시 메모리 데이터베이스 환경에서 새로운 트랜잭션 회복 기법을 제안하였다. 먼저, 플래시 메모리의 특성을 고려한 데이터 처리를 위한 기반 구조를 제시하였고, 플래시 파일 시스템의 일반적인 구조를 기술하였다. 제안된 플래시 메모리 쉐도우 페이징 기법은 플래시 파일 시스템에서 버려지는 데이터 페이지를 재활용함으로써 기존 쉐도우 페이징의 추가 저장 공간 부담과 입출력 성능 저하의 단점을 제거하고 플래시 메모리 데이터베이스 시스템의 저장 효율과 성능을 높일 수 있다. 이를 위하여 지연된 소거 리스트 구조를 제안하였으며, 플래시 메모리 쉐도우 페이징 기법을 적용하여 트랜잭션을 수행하기 위하여 필요한 데이터 관리자의 하부 처리 연산 과정을 보였다. 또한, 제안 기법의 효과를 검증하기 위하여, 윈도우 2000 서버에서 컴퓨터 시뮬레이션을 수행하고, 성능 평가 및 결과를 분석하였다.

향후 연구는 제안된 기법의 실제 실험을 통한 성능 검증과 휴대용 소형 정보기기에 적용하여 유효성을 확인하는 것이다. 또한, 플래시 메모리의 용량도 기가 바이트 단위로 늘어나는 최근의 현실을 반영하여, 효율적인 색인구조 및 저장 기법의 연구 개발이 필요하다. 현재, 휴대용 정보 시스템의 데이터 저장장치로서 대부분 플래시 메모리가 사용된다는 측면에서, 플래시 미디어 기반 데이터 처리 기술은 향후 지속적인 연구가 필요하다.

참 고 문 헌

[1] 권우일, 박현희, 양승민, "리눅스 기반의 사용자 수준 플래시 파일 시스템의 구현", *한국정보처리학회논문지(KIPS Transactions : PartA)*, 제11A권 3호, 2004년 6월, pp. 139-148.

[2] 박성호, 정기동, "내장형 운영 체제의 파일 시스템-Flash File System", *한국정보처리*

학회지(Korea Information Processing Society Review), 제9권 3호, 2002년 5월, pp. 103-109.

[3] 변시우, "휴대용 컴퓨터를 위한 플래시 메모리 기반 데이터 관리 기술 동향", *대한전자공학회 하계종합학술대회 논문집*, 제27권 1호, 2004년, pp. 823-826.

[4] 임근수, 고건, "플래시 메모리 기반 저장장치의 설계 기법", *정보과학회 추계학술대회*, 제30권 2-1호, 2003년 10월, pp. 274-276.

[5] 최리군, "플래시 메모리 카드 동향", *주간 전자정보*, 제4권 3호, KETI, 2001.

[6] 황규영, 홍의경, 음두현, 박영철, 김진호, *데이터베이스 시스템*, 생능출판사, 2000.

[7] Chang L. and Kuo T., "An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded Systems", *in : Proc. 8th IEEE Real-Time and Embedded Technology Symposium*, California, San Jose, 2002, pp. 187-196.

[8] Choi Mi-Seon, Yoon Hye-Sook, Song Eun-Mi, Kim Young-Keol, Kim Young-Kuk, Jin Seong-il, Han Mi-Kyong, and Choi Wan, "Two-Step Backup Mechanism for Real-Time Main Memory Database Recovery", *in Proc. Of the Seventh Intl Conf on Real-Time Systems and Applications(RTCSA'00)*, Cheju Island, South Korea, Dec. 2000, pp. 453-457.

[9] Garcia-Molina H. and Salem K., "Main Memory Database Systems : An Overview", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, No. 6, 1992, pp. 509-516.

[10] Jack Kent, Hector Garcia-Molina, "Optimizing Shadow Recovery Algorithms", *IEEE Transactions on Software Engineering*, Vol. 14, No. 2, Feb. 1988, pp. 155-168.

[11] Jeanna Neefe Matthews, Drew Roselli, Adam M. Costello, Randolph Y. Wang, and Thomas E. Anderson, "Improving the performance of log-structured file systems with adaptive methods", *in Proc. of the sixteenth ACM symposium on Operating systems principles*, Saint Malo, France, 1997, pp. 238-251.

[12] Matthew S. Hecht, and John D. Gabbe, "Shadowed Management of Free Disk Pages with a Linked List", *ACM Transactions on Database Systems*, Vol. 8, No. 4, 1983, pp. 503-514.

[13] Mendel Rosenblum John K. Ousterhout, "The design and implementation of a log-structured file system", *ACM Transactions on Computer Systems,* Vol. 10, No. 1, February 1992, pp. 26-52.

[14] Song, E.M., Kim Y.K., and Ryu C.H., "No-Log Recovery Mechanism Using Stable Memory for Real-Time Main Memory Database Systems", *RTCSA'99, IEEE CS*, Dec 1999, pp. 428-431.

[15] Vijay Kummar, Albert Burger, "Performance Measurement of Main Memory Database Recovery Algorithms Based on Update-in-Place and Shadow Approaches", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, No. 6, 1992, pp. 567-571.

[16] JFFS, http://developer.axis.com/software/jffs/, 2004.

[17] JFFS2, http://source,redhat.com/jffs2/, 2004.

□ 저자소개

변 시 우

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

◈ 이 논문은 2005년 6월 29일 접수하여 1차 수정을 거쳐 2006년 6월 14일 게재확정되었습니다.