플래시 메모리 데이터베이스를 위한 플래시 인지 트랜잭션 관리 기법☆

Flash-Aware Transaction Management Scheme for Flash Memory Database

변 시 우*
Byun Si Woo

요 약

플래시 메모리는 이동형 컴퓨팅 환경에서 휴대용 정보기기를 지원하는 우수한 미디어이다. 플래시 메모리는 비휘발성, 낮은 전력소모, 빠른 데이터 접근 속도 등의 특징으로 휴대용 정보기기용 데이터베이스의 핵심 저장 모듈이 되었다. 하지만, 일반 RAM 메모리에 비하여 상대적으로 느린 연산 특성을 고려하여 기존의 트랜잭션 처리 기법을 개선할 필요가 있다. 이를 위하여, 본 논문은 플래시 인지 트랜잭션 관리(Flash-Aware Transaction Management: FATM) 기법을 제안한다. FATM은 SRAM과 W-cache 를 활용하여 트랜잭션 처리 성능을 높인다. 또한, 성능 검증을 위하여 시뮬레이션 모델을 제안하였으며, 실험 결과 분석을 통하여 FATM이 기존의 트랜잭션 처리 기법보다 우수함을 확인하였다.

Abstract

Flash memories are one of best media to support portable computers 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-aware transaction management (FATM). FATM improves transaction performance by exploiting SRAM and W-Cache. We also propose a simulation model to show the performance of FATM. Based on the results of the performance evaluation, we conclude that FATM scheme outperforms the traditional scheme.

Keyword : Mobile Computing, Distributed Database, Embedded System

1. 서 론

최근 초고속 인터넷 및 무선 통신망의 급속한 보급과 더불어, 각종 소형 정보기기들이 대중화됨에 따라, 정보 저장용 미디어로 플래시 메모리가 보편적으로 활용되게 되었다. 따라서 플래시 미디어에 저장된 데이터를 체계적으로 관리하는 임베디드 소프트웨어가 필요하게 되었다.

Gartner社의 자료에 따르면, 정보기기용 임베디드 소프트웨어는 최근 100%이상으로 급성장하고 있다고 한다.[1] 특히, IDC(International Data Corporation)社는 64%이상의 휴대용 정보기기의 어플리케이션이 임베디드 데이터베이스가 필요하다고 보고했다.[2] 따라서 플래시 메모리 기반 데이터베이스 시스템은 핵심 임베디드 소프트웨어 분야로서 유망하며, 이와 관련된 데이터 관리 기술의 개발이 시급하다고 할 수 있다. 본 논문은 휴대용 플래시 메모리 데이터베이스 시스템에서의 새로운 트랜잭션 관리 기법을 제안한다.

* 정 회 원 : 안양대학교 디지털미디어학부 조교수
swbyun@aycc.anyang.ac.kr(제 1저자)
[2004/07/09 투고 - 2004/07/30 심사 - 2004/10/07 심사 완료]
☆ 이 논문은 2004년도 한국과학재단의 지원에 의하여 연구되었음(R05-2004-000-10961-0)

2. 관련 연구

플래시 미디어 자체에 대한 연구나 플래시 파

일 시스템에 대한 연구는 성숙단계에 진입하였다. 또한, 기존의 디스크-기반의 상용 데이터 처리기나 메인 메모리-기반 데이터 처리기는 오래 전부터 인기 있는 연구 분야로 활발한 연구가 지속되고 있다. MM-DBMS[3], MARS[4], System M[5] 등의 시스템이 과거의 메모리 기반 데이터 처리기의 연구사례이다. 하지만 플래시 메모리를 활용하는 데이터베이스 시스템에 대한 연구는 아직 시작 단계에 불과한 실정이다.

MM-DBMS는 데이터 표현과 트랜잭션 관련 접근 처리를 위하여 OBE와 마찬가지로 많은 포인터를 사용하였으며, 인덱스 구조는 인덱스된 투플을 직접 포인트 한다. 다양한 리니어 해싱이 사용되었고, T-Tree가 사용되었다. 트랜잭션 관련 커미트 처리는 로그 레코드와 복구처리기를 위해 메인 메모리가 아닌 안전 기억 장치를 활용하여 수행하였다. 트랜잭션의 동시성 제어를 위하여 two-phase 기법을 적용하였고, 전체 릴레이션을 로킹하는 기법을 사용하였다.

MARS는 메모리 상주의 휘발성 데이터에 대하여 고속의 트랜잭션 처리를 위하여 데이터베이스 처리기와 복구 처리기를 사용하였다. 두 처리기 모두 비휘발성 메모리도 접근하였는데, 복구처리기는 데이터베이스 로그와 백업 복사를 위하여 디스크에도 접근하였다. 데이터 처리기는 트랜잭션의 커미트 지점까지의 수행을 담당하였으며, 비휘발성 메모리에는 수정 부분을 기록하였다. 수정 트랜잭션이 철회된 경우에는 단순히 기록된 수정 부분을 폐기하도록 설계하였다. 복구처리기는 트랜잭션을 커미트 하기 위하여 비휘발성 메모리로부터 데이터베이스로 수정 부분을 복사하며, 수정 부분 레코드는 비휘발성 로그 버퍼에 복사하였다. 복구 처리기는 전체 로그 버퍼를 로그 디스크로 반영하게 하였다. 동시성 제어는 two-phase 로킹과 대단위 로킹 기법을 사용하였다.

플래시 메모리를 저장 매체로 사용한 연구에는 eNVy 시스템이 있다.[6] 이 시스템은 플래시 메모리를 일반 중소형 컴퓨터 및 하드 디스크의 대용으로 사용하였다. 플래시 메모리를 다량으로 연결하여 2GB의 저장 영역을 구성하였다. 세그먼트 소거 연산의 오버헤드를 줄이기 위하여 MMU (Memory Management Unit)을 별도로 가지고 있다. 이 구조는 휴대형 컴퓨터에서 사용하기에는 너무 크고, 고가의 MMU를 사용하는 것도 무리가 있으며, 소거 연산에 너무 많은 시간을 소모한다.

플래시 메모리의 데이터 접근 효율향상을 위하여 능동적 스트라이핑 기법[7] 을 사용하기도 한다. 플래시 메모리는 느린 특성상 입출력 병렬 연산을 위하여 여러 뱅크에 분산하여 스트라이핑을 사용하면 매우 효율적이다. 이 연구는 삼성에서 생산된 NAND 플래시 메모리를 대상으로 하였으며, 뱅크 할당 기법, 핫-콜드 블록 분석 기법을 활용하여 실제 실험도 하였다. 여러 뱅크에 분산하면, 블록 연산이 뱅크 셋업, 연산 준비, 데이터 접근 등의 제어 과정이 분산되어 병행 수행의 효과를 볼 수 있다. 특히, 플래시 메모리는 내장형 시스템과 같이 느린 시스템에 많이 사용되므로 상대적으로 큰 효과를 볼 수 있다. 또한, 가비지 수집 과정에서 발생하는 20% 이상의 성능 저하에 대비하여 대처 방안으로 활용가능하다.

비휘발성 저장장치로서 향후 대중화 가능성이 큰 메모리로서 FRAM(Ferroelectric RAM)도 있는데, 플래시 메모리 보다 특성 면에서 우수하며 상용화 단계에 진입하였다. 또한 MRAM(Magnetic RAM)등도 활발히 연구되고 있다. 향후 이러한 비휘발성 메모리에 적합한 저장 매체 관리 기술과 데이터 처리 기술에 대한 연구가 필요하다.

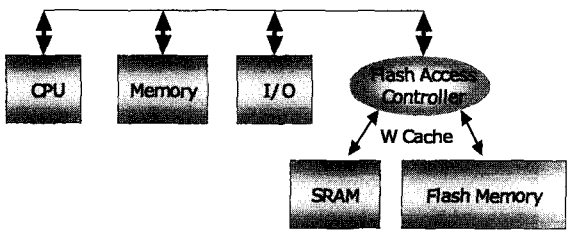
그러나 순수하게 플래시 미디어 기반의 데이터 처리기의 연구는 미개척 분야이다. 그 이유는 그동안 플래시 메모리는 주 메모리의 성능과 비교하면 느리면서도 고가이며, 일반적인 시스템에는 사용되지 않는 특수 부품이며, 디스크에 비하여 저장 비용이 고가였기 때문이다. 그러나 최근 플래시 메모리 기술의 발전으로 대부분의 단점들이 해소된 상태이다. 다만, 일반 메인 메모리와는 달리, 쓰기와 지우기 연산에 10배정도의 많은 시간

이 소요되며, 쓰기 회수가 100,000번 정도로 제한된다는 특성을 가만하여, 효과적인 휴대형 정보기기의 데이터 처리기를 개발하여야 한다.

3. 플래시 메모리 기반 트랜잭션 관리

3.1 플래시 메모리 접근 및 처리 구조의 제안

플래시 메모리는 일반 메인 메모리와 같은 직접 접근이 어려우므로 I/O 컨트롤러와 유사한 플래시 액세스 컨트롤러를 통하여 읽은 후에 메인 메모리에 매핑해서 사용하게 된다. 그런데 액세스 컨트롤러를 플래시 메모리에 바로 연결하는 기존의 연결 방식으로는 플래시 메모리의 느린 소거 연산과 쓰기 연산 속도를 개선할 수 없다. 즉, 이 두 요소 사이에 속도 개선을 위하여 추가 구조가 필요하다. 본 논문에서 제안하는 처리 구조는 기존 구조를 개선하여, 쓰기 속도가 느린 단점을 보완하기 위하여 SRAM에 쓰기를 우선 적용하고, 나중에 플래시 메모리에 반영하는 구조이다. 여기서 SRAM은 전원이 꺼져도 배터리의 전력으로 데이터가 일정기간 유지 한다.(그림 1) 플래시 액세스 컨트롤러는 플래시 메모리와 SRAM을 활용하여 플래시 접근을 제어 하게 된다. 즉, 외부에서 읽기 요청이 올 경우에는 플래시 메모리에서 직접 읽어서 데이터를 전달해 주면 되고, 쓰기 요청이 올 경우에는 W-Cache를 통하여 SRAM에 플래시 메모리의 데이터를 복사한 후에 쓰기 연산을 수행하고, 이후의 유휴시간에 플래시 메모리에 반영하게 된다.



(그림 1) 새로운 플래시 메모리 접근시스템 구조도

외부에서 읽기 요청이 올 경우에는 플래시 메모리에서 직접 읽어서 데이터를 주면 된다. 쓰기 요청이 올 경우에는 SRAM에 데이터를 복사한 후에 SRAM에 쓰기 연산을 수행 후에 플래시 메모리에 반영한다.

3.2 플래시 데이터베이스의 트랜잭션 처리

3.2.1 설계 목표

다음은 본 연구에서 제안하는 플래시 메모리 기반의 트랜잭션 처리 기법의 설계 목표이다.

① ACID 조건을 수용한다.
② 플래시 접근 트랜잭션 처리 성능을 높인다.
③ 플래시 접근 트랜잭션 응답 시간을 줄인다.
④ 플래시 접근 수명과 안정성을 높인다.

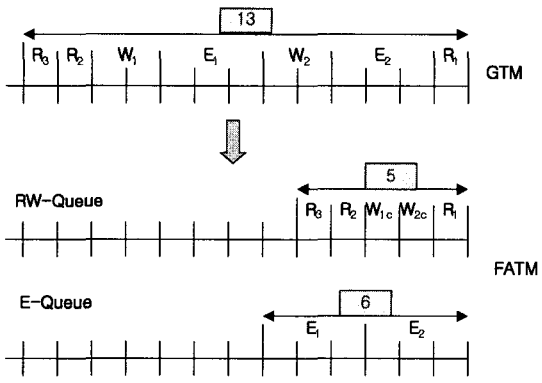
3.2.2 플래시 인지 트랜잭션 스케줄링 기법

플래시 메모리를 사용하는 데이터베이스에 기존의 일반적인 트랜잭션 관리 기법(General Transaction Management: GTM)[8]을 적용하면 기본적으로 FCFS(First Come First Served)에 준하여 순서적으로 트랜잭션을 처리하게 된다. 이는 설계 목표 ①의 ACID 조건을 만족한다. 그러나 설계 목표 ②, ③, ④를 만족하기 위해서는 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 메모리 인지 트랜잭션 관리 기법(Flash Aware Transaction Management: FATM)을 제안한다. 즉, 트

랜잭션 관련 데이터가 플래시 메모리에 위치한 경우에 플래시 메모리의 접근 특성을 인지하고, 이를 충분히 고려하여 최적화한 효율적인 트랜잭션 관리를 제공하는 기법이다. 이 기법을 통하여 설계 목표 ②와 설계 목표 ③을 만족시킬 수 있다.

먼저 그림 2를 통하여 FATM 기법의 효과를 설명하면 다음과 같다. 트랜잭션의 수행 시간을 비교하기 위하여 여러 칸으로 표시하였다. 읽기(Read: R) 연산의 수행 시간을 1단위로 정의 하고, 쓰기(Write: W) 연산은 2단위, 소거(Erase: E) 연산은 3단위로 정의하였다. 이는 편의상 시간적인 크기를 상대적으로 비교한 것이다. 실제로 쓰기 연산은 읽기 연산에 비하여 8배 느리며, 소거 연산은 50배 이상 느리다.



〈그림 2〉 GTM 기법과 FATM 기법의 처리 시간 비교

그림 2에서와 같이 트랜잭션의 연산 순서가 (R_1, E_2, W_2, E_1, W_1, R_2, R_3)순으로 들어올 경우를 가정하여 두 기법을 비교하여 보자. 이 경우 쓰기 연산에 앞서 지우기 연산이 먼저 수행되어 있음을 알 수 있으며, 총 수행 시간은, 읽기 연산이 3번, 쓰기 연산이 2번, 소거 연산이 2번으로 총 13 단위의 시간이 소요된다. 먼저, 기존의 방식인 GTM을 적용하면, 하나의 트랜잭션 큐에 순차적으로 수행하여 총 13단위의 시간이 소요된다.

다음으로 FATM 기법에서는 SRAM을 사용하여 쓰기 연산을 신속히 수행하며, 쓰기 연산에 수반되는 소거 연산을 소거 큐를 사용하여 별도로 처리할 수 있다. 즉, 읽기-쓰기 큐(RW-Queue)와 소거 큐(E-Queue)를 통하여 병렬적인 수행이 가능하다. 특히, 쓰기 연산의 경우에 플래시 메모리에 기록하지 않고, SRAM Cache에 기록하므로 연산 속도가 대폭 개선 가능하다. 그림2에서는 이 개선된 쓰기 연산의 시간을 1 단위로 정의하였다. 그리고 읽기-쓰기 큐에는 읽기 연산이 3번, SRAM 기반 쓰기 연산이 2번으로 총 5단위의 연산이 필요하며, 소거 큐에는 소거 연산이 2번으로 총 6단위의 연산 시간이 소요된다. 이 두 큐에서 수행되는 연산 시간을 모두 합하면 총 11 단위로서, FATM 방식이 GTM 방식 보다 더 연산 수행 시간이 적게 소모됨을 알 수 있다. 또한, 읽기 및 쓰기 연산이 대기하는 시간도 반 이하로 줄어 들게 되므로 응답 시간도 향상됨을 알 수 있다. 이 예를 통하여 FATM 기법은 설계 목표 ②와 설계 목표 ③을 만족시킬 수있음을 알 수 있다. 이 사실은 다음장의 컴퓨터 시뮬레이션을 통한 성능 평가 및 분석과정에서 다시 확인할 수 있다. 그리고, 읽기-쓰기 큐에서 순차적으로 연산이 수행되며, 플래시 메모리와 SRAM에 영구 기억되며, 일관성을 가진다. 즉, GTM과 기본 특성이 동일하므로 설계 목표 ①도 만족된다. 또한, SRAM을 통한 쓰기 연산은 플래시 메모리에 직접 쓰기 연산을 하는 횟수를 줄여서 플래시 메모리의 수명을 연장시키므로 설계 목표 ④를 만족시킬 수 있다. FATM의 간략한 수행과정은 다음과 같다.

1) 사용자가 플래시 메모리에 대한 읽기-쓰기 연산을 요청한다.
2) FATM은 사용자 트랜잭션의 입력을 인지하고 연산의 종류를 분류하여 해당 처리 큐의 마지막에 삽입한다. 읽기 트랜잭션과 쓰기 트랜잭션은 RW-Queue로 삽입하고, 소거 연산은 E-Queue에 삽입한다. 이후 RW-Queue와 E-Queue는 독립적으로 병렬 수행된다.
3) RW-Queue에서 큐의 맨 앞에 있는 트랜잭

션을 불러 온다. 이 트랜잭션을 매개 변수로 해서 플래시-미디어 관리자로 원격 호출을 한다. 플래시-미디어 관리자는 주소 변환 연산을 수행하여 플래시 메모리 내의 물리적인 주소를 계산한다. 또한, 해당 트랜잭션이 읽기 연산이면, 플래시 메모리에서 직접 하위 읽기 연산으로 결과 값을 리턴 한다. 쓰기 연산이면, W-cache를 통하여 SRAM에 하위 쓰기 연산을 수행 하고, 수행 결과를 리턴 한다.

4) E-Queue에서 큐의 맨 앞에 있는 트랜잭션을 불러 온다. 이 트랜잭션을 매개 변수로 해서 플래시-미디어 관리자로 원격 호출을 한다. 플래시-미디어 관리자는 주소 변환 연산을 수행하여 플래시 메모리 내의 물리적인 주소를 계산한다. 하위 소거 연산을 수행하고, 수행 결과를 리턴 한다.
5) 사용자 트랜잭션을 기다리며, 큐의 내용이 없을 때까지 3)번과 4)번을 계속 수행한다.

4. 시뮬레이션 및 성능 평가

플래시 메모리 데이터베이스 시스템을 위해 제안된 FATM의 성능을 검증하기 위하여 시뮬레이션을 수행하였으며, 결과를 분석해 보았다. 실험 도구는 윈도우 2000 환경에서 CSIM[9] 시뮬레이션 언어와 비주얼 C++를 사용하였다.

4.1 시뮬레이션 모델

기본적인 시뮬레이션 모델은 큐잉 모델에 기반하고 있다. 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.

시스템 모델은 사용자 트랜잭션 생성기(User Transaction Generator: UTG), 플래시 인지 트랜잭션 관리자(Flash Aware Transaction Manager: FATM), 플래시 미디어 관리자(Flash Media Manager: FMM), 플래시 접근 제어기(Flash Access Controller: FAC), 플래시 세그먼트 관리자(Flash Segment Manager: FSM)로 구성된다.

사용자 트랜잭션 생성기는 시뮬레이션에 필요한 작업 부하를 만들기 위하여 특정 간격(*inter_arrival_time*)으로 플래시 메모리에 접근하는 사용자 트랜잭션을 생성시킨다. 플래시 인지 트랜잭션 관리자는 사용자 트랜잭션의 생성부터 종료까지의 수행을 관리하며, 플래시 미디어 관리자는 플래시 메모리의 주소 변환과 연산 수행을 담당한다. 플래시 접근 제어기는 플래시 접근 연산을 하위 레벨에서 SRAM을 이용하여 수행하며, 플래시 세그먼트 관리자는 플래시 메모리의 할당 및 수명 관리 업무를 수행한다.

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.

· 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 400개에서부터 400개 단위로 2800개까지 변화시켜 보았으며, 이는 시뮬레이션 시스템에 가해지는 작업 부하를 의미한다. 읽기 연산 수행 시간은 36μs로 설정하였고, 쓰기 연산 수행 시간은 266μs 로 설정하였으며, 소거 연산 수행 시간은 2ms로 설정하였다. 각 연산 수행 시간은 기존의 연구[10]에서 제시한 연산 수행에 필요한 실측치이다. 전체 읽기/쓰기 연산의 수에 대한 쓰기 연산의 비율인 갱신 비율은 일반적인 기준인 25%로 설정하였다.

4.2 실험 결과 및 분석

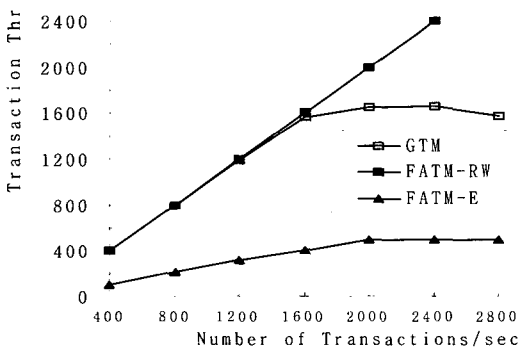
실험은 플래시 메모리를 탑재한 정보기기에서

초당 발생된 트랜잭션의 수가 위의 두 기법들의 성능에 어떤 영향을 미치는 지를 분석하기 위한 것이다. 기본 값으로 플래시 데이터 접근에 대하여 갱신 비율은 일반적인 기준인 25%로 설정하였다. 그림3은 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, GTM 보다 FATM이 높게 나타났다.

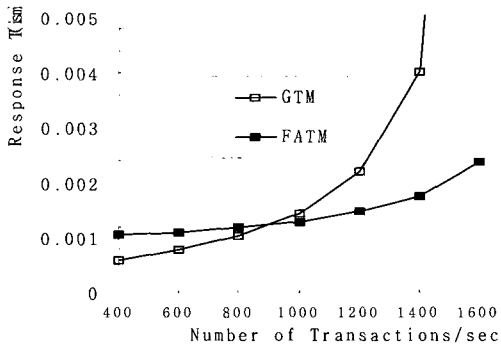
그림 3에서 보면, 초당 트랜잭션의 수가 대략 1600개를 넘으면서 GTM 기법의 성능이 점점 낮아진다. 이는 초당 트랜잭션의 수의 증가에 의한 데이터 연산 집중화가 성능에 영향을 크게 미치는 주요 요소임을 의미하며, 1600개 이상으로 트랜잭션을 활성화시키는 것이 성능 향상에 도움이 되지 않음을 의미한다. 하지만, 동일한 조건에서도 제안한 FATM 기법이 GTM 기법에 비하여 성능이 더 높은 이유는 SRAM의 W-cache를 활용하여 과도한 트랜잭션 오버헤드를 줄이고, 소거 연산을 분리하여 연산 집중을 원활하게 분산하였기 때문이라고 분석된다.

그림 3에서 FATM-RW는 FATM 기법에서 읽기와 쓰기 연산에 대한 트랜잭션 처리치를 표시한 것인데, 전 구간에서 고르게 좋은 성능이 나타난다. SRAM을 이용한 쓰기 캐싱의 효과가 상당히 좋음을 알 수 있다. 반면에 FATM-E는 FATM 기법에서 병렬적으로 수행되는 소거 연산에 대한 트랜잭션 처리치를 표시한 것인데, 초당 2000개 이상의 트랜잭션이 발생하면 성능이 더 이상 증가하지 않음을 알 수 있다. 단순비교 관점에서 볼 때, FATM-RW의 성능이 우수하더라도, FATM-E의 성능이 초당 2000개에서 한계를 보이므로, 전체적인 FATM의 성능은 GTM 기법의 1600개 보다 25% 향상된 2000개로 일단 평가할 수 있다. 그러나 데이터 접근과 무관한 소거 연산은 시스템의 유휴 시간에 지연해서 후에 처리할 수 있으므로, 실질적인 처리 성능은 몇 배 더 나을 수도 있다. 또한, 사용자가 느끼는 실질적인 연산인 읽기와 쓰기의 처리 성능과 응답이 훨씬 더 좋으므로, 체감적인 성능은 더 우수하다고 분석된다.

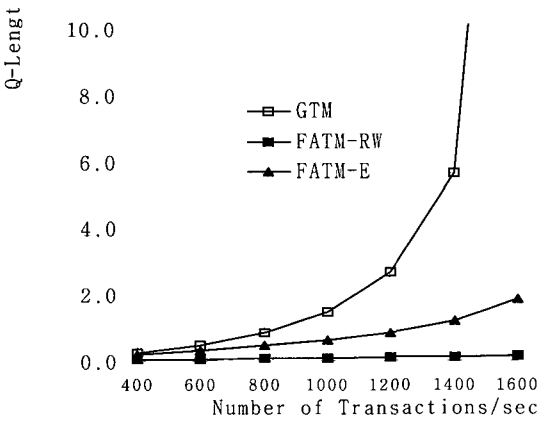
그림 4는 초당 트랜잭션 발생수의 증가에 대한 응답시간의 변화를 보여주는 그래프이다. GTM과 FATM 방식 모두가 발생된 초당 트랜잭션의 수가 늘어날수록 시스템에 부하가 증대되고 점차로 응답 시간이 증가함을 알 수 있다. 이 그래프를 살펴보면, 초당 트랜잭션 수가 900 이하일 경우는 GTM의 응답시간이 더 빠르나, 이 구간을 지날수록 FATM의 응답시간이 상대적으로 빠르게 나타났다. 이 현상은 FATM이 기존의 GTM을 개선하면서 추가한 모듈에서 발생한 부수적인 부하 때문이다. 그러나 900 구간을 지나면서 플래시 인지 트랜잭션 관리의 효과가 나타나서 결과적으로 더 우수한 응답성능을 보인다.



〈그림 3〉 초당 트랜잭션 처리치의 비교



〈그림 4〉 응답 시간의 비교



〈그림 5〉 처리 큐의 대기 길이 비교

이러한 응답 성능의 개선 효과를 뒷받침해 줄 그림5의 처리 큐의 대기 길이 비교 그래프를 살펴보자. 일단 GTM과 FATM 방식 모두가 발생된 초당 트랜잭션의 수가 늘어날수록 점차로 처리큐의 대기 길이가 증가함을 알 수 있다. 하지만, 전 구간에서 FATM의 큐의 길이가 GTM의 큐의 길이 보다 더 짧음을 알 수 있다. 또한, 큐의 길이 그래프의 기울기도 FATM이 GTM 보다 완만하게 증가하여, 전체적으로 큐의 대기 연산수가 더 적은 좋은 성능을 보인다. 즉, FATM에서 대기하는 연산의 수가 더 적으므로, 이 효과는 FATM의 응답시간 향상에 기여하고 있음을 알 수 있다. 반면에 GTM은 큐의 길이가 기하급수적으로 증가하여 결과적으로 저조한 응답성능으로 나타나고 있다. 전체적인 변화 구간에서 FATM은 GTM에 비하여 2.5배 정도의 빠른 응답 성능을 보였다.

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 시스템 구조와 플래시 인지 트랜잭션 관리 기법(FATM)을 제안하였다. 쓰기 연산과 소거 연산이 상대적으로 매우 느린 플래시 메모리 연산의 단점을 보완하기 위하여, SRAM을 활용한 W-cache 구조를 사용하였다. 또한, 이 구조와 관련된 읽기-쓰기 연산 전용 큐와 소거 연산 전용 큐를 분리하여 병렬로 수행함으로써 전체적인 성능을 높였다.

제안 기법의 효과를 검증하기 위하여, 윈도우 2000 서버에서 컴퓨터 시뮬레이션을 수행하고, 성능 평가 및 결과를 분석하였다. 작업 부하가 적은 시작 구간에서는 기존 처리 기법이 우수하였지만, 작업 부하가 증가할수록 제안 기법의 성능이 더 우수하였다. 전반적으로 평가할 때, 기존의 일반적인 트랜잭션 처리 기법에 비하여 트랜잭션 처리 성능은 25%의 개선 효과를 얻었으며, 응답 성능은 1.8배의 개선 효과를 얻을 수 있었다.

향후 연구 과제는 시뮬레이션을 통하여 검증된 데이터 관리 모듈의 구현과 제한된 휴대용 정보기기의 임베디드 시스템 자원을 최소한도로 사용하는 최적화 방안의 연구이다. 현재, 휴대용 정보 시스템의 데이터 저장장치로서 대부분 플래시 메모리가 사용된다는 측면에서, 플래시 미디어 기반 데이터 처리 기술의 연구는 향후 중요 연구 분야로 활성화될 것이다.

참 고 문 헌

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

[2] 유재정, "Mobile Database란?", http://www.mobilejava.co.kr/bbs/temp/lecture/j2me/mdb1.html, 2003.

[3] T. J. Lehman and M. J. Carey, "Query Processing in Main Memory Database Management Systems", in Proc. ACM SIGMOD Conf., Washington, DC, pp. 239-250, May 1986.

[4] M. H. Eich, "A Classification and Comparision of Main Memory Database Recovery Techniques", in Proc. Int. Con. On

Data Engineering, pp. 332-339, Feb. 1987.

[5] H. Garcia-Molina and K. Salem, "Main Memory Database Systems: An Overview" IEEE Trans. Knowl. Data Eng., Vol.4, No. 6, pp. 509-516, Dec. 1992.

[6] 민용기, 박승규, "이동컴퓨터를 위한 플래시 메모리 클리닝 정책", 한국통신학회논문지, Vol. 24 No. 5A, pp. 657-666, 1999. 5.

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

[8] P. Bernstein, V. Hadzilacos , N. Goodman, Concurrency control and recovery in database systems, Addison-Wesley, 1987.

[9] H. Schwetman, CSIM User's Guide for Use with CSIM Revision 16, Microelectronics and Computer Technology Corporation, 1992.

[10] 임근수, 고건, "플래시 메모리 기반 저장장치의 설계 기법" 정보과학회 2003년 추계 학술대회 Vol. 30, No. 2-1, pp. 274-276, 2003.10.

◐ 저 자 소 개 ◑

변 시 우(Byun Si Woo)

1989년 연세대학교 이과대학 전산과학과(공학사)
1991년 한국과학기술원 전산학과(공학석사)
1999년 한국과학기술원 전산학과(공학박사)
2000년~현재 : 안양대학교 디지털미디어학부 조교수
관심분야 : 분산 데이터베이스, 모바일 컴퓨팅, 임베디드 시스템 등
E-mail : swbyun@aycc.anyang.ac.kr