Journal of Digital Contents Society Vol. 8 No. 4 Dec. 2007(pp. 593-600)

플래시 메모리용 DBMS를 위한 스토리지 시스템의 계층 통합에 대한 연구

심효기*, 윤경훈**, 박성민***, 정호영****, 강수용*****, 차재혁******

요 약

휴대용 기기나 디지털 미디어 기기와 같은 소형 컴퓨터는 저장 매체로 NAND 타입의 플래시 메모리를 사용한다. 하지만 이러한 기기에 사용되는 DBMS의 경우 대부분 하드디스크를 저장매체로 사용되도록 최적화되어 있다. 플래시 메모리를 사용하는 소형 컴퓨터 시스템에서는 DBMS를 사용할 때 플래시 메모리를 기존 하드디스크와 같은 인터페이스로 제어하기 위해 플래시전용의 파일시스템이나 FTL 등의 계층을 추가적으로 사용하게 되며, 이 때 DBMS는 플래시 메모리를 직접 제어할 수 없게 된다. 본 논문에서는 DBMS가 파일 시스템이나 FTL과 같은 부가적인 계층 구조를 이용하지 않고 플래시 메모리를 직접 제어할 수 있는 통합된 저장 시스템을 제안한다. 또한 제안한 시스템을 실제 시스템에 직접 구현해 DBMS의 성능이 기존 시스템에 비해 크게 향상됨을 보였다.

A Study of the Merging Layers of the Storage System for Flash-Based DBMS

Hyo-Gi Sim*, Kyounghon Yoon**, Sungmin Park***, Ho-young Jung****, Sooyong Kang*****, Jaehyuk Cha******

Abstract

Small computer systems such as mobile devices adopt NAND flash memories as their storage media. However, DBMS running on such systems are optimized to hard disks. When small computer systems use DBMS they usually use additional system layer, like FTL, that emulates flash memories as normal hard disks and DBMS cannot control flash memories directly. In this paper, we propose unified storage system that DBMS controls flash memories directly. We implemented the system in a real environment and proved the proposed system outperforms legacy systems.

Keywords : Flash Memory, DBMS

1. 서론

휴대용 기기나 디지털 미디어 기기의 성능이 향상되면서 소형 컴퓨터 시스템에서도 주소록 데이타, 미디어 파일과 같은 많은 양의 데이타를 처리하게 되었다[1][2]. 큰 규모의 데이타의 효과적인 관리를 위해 소형 컴퓨터 시스템에서도 DBMS가 이용되고 있다. 하지만 기존의 DBMS는 저장장치로 하드디스크가 이용되는 컴퓨터 시스템에서 효율적으로 동작하도록 작성되어 있는데 반해 소형 컴퓨터 시스템에서는 저장장치로 하드디스크 대신 플래시 메모리를 이용하는 경우가 많다.

플래시 메모리는 비휘발성 기억장치로서 기존의 하드디스크에 비해 I/O 연산의 속도가 빠르고 외부 충격에 강하며 전력소모가 작고 무게가 가볍다는 장점이 있다. 하지만 플래시 메모리는

※ 제일저자(First Author) : 심효기
접수일자:2007년08월10일, 심사완료:2007년08월17일
* 한양대학교 석사과정
dahila@hanyang.ac.kr
** 한양대학교 석사과정
*** 한양대학교 박사과정
**** 한양대학교 박사과정
***** 한양대학교 컴퓨터교육과 교수(교신저자)
****** 한양대학교 전자컴퓨터통신공학과 교수
▣ 본 연구는 한국과학재단 특정기초연구(R01-2006-000-10630-0)지원으로 수행되었음

하드디스크에 비해 다루기 어려운 단점도 가지고 있다. 플래시 메모리는 하드디스크와는 다르게 읽기와 쓰기 연산의 기본 단위인 페이지에 대한 재쓰기가 불가능하다. 페이지에 재쓰기를 하기 위해서는 페이지가 포함된 블록에 대한 지우기 연산이 필요하다. 하지만 플래시 메모리의 지우기 연산은 읽기나 쓰기 연산에 비해서 보통 수십 배 정도 느리며, 한 블록에지우기 연산을 할 수 있는 횟수에 10,000번에서 100,000번 정도로 제한이 있다[3][4]. 이러한 플래시 메모리의 단점을 보완하기 위해서는 플래시 메모리 특성을 고려해 지우기 연산을 줄일 수 있는 저장 시스템이 필요하다.

플래시 메모리를 저장 장치로 사용하기 위해 플래시 메모리를 하드디스크와 같은 블록 디바이스로 에뮬레이션 해주는 추가적인 FTL (Flash Translation Layer)[5] 계층을 시스템에 추가하거나 JFFS2[6]나 YaFFS[7] 플래시 메모리 전용 파일 시스템을 이용한다. 하지만 이렇게 구성된 플래시 메모리 기반의 시스템에서에서는 DBMS의 물리적 I/O 관리 계층 아래에 OS의 페이지 캐시, 블록 디바이스 관리 계층, 파일시스템 그리고 플래시 디스크의 FTL 계층이 존재하게 되며, DBMS는 물리적인 저장 매체인 플래시 메모리를 직접 제어할 수 없다.

본 논문에서는 DBMS의 물리적 I/O 관리 계층에서 플래시 메모리를 직접 제어할 수 있는 시스템을 제안했으며 실제 시스템에서 이를 구현해 기존의 시스템에 비해 성능이 70%이상 향상되었음을 보였다.

논문의 구성은 다음과 같다. 2장은 관련 연구로 플래시 메모리의 특성과 플래시 메모리를 이용하는 시스템 위에 구성된 DBMS의 계층 구조를 알아본다. 3장에서는 본 논문에서 제안한 시스템의 구조를 설명하며 4장은 실제로 구현한 시스템과 기존 시스템과의 성능을 비교한다. 마지막으로 5장에서는 본 논문의 결론과 추후 연구 방향을 제시한다.

2. 관련 연구

2.1 플래시 메모리

플래시 메모리는 EEPROM(Electrically Erasable Read Only Memory)의 한 종류이며 NAND, NOR, AG-AND, ECC-NOR 등 여러 타입이 존재한다. 이들은 모두 서로 다른 특성을 가지고 있지만 쓰기 연산과 지우기 연산이 구분되어 있다는 공통점을 가지고 있다. 플래시 메모리의 모든 비트 1로 초기화 되어 있으며 쓰기 연산을 수행하면 1로 초기화된 비트 중 일부를 0으로 바꾸게 된다. 이렇게 0으로 바뀐 비트를 다시 1로 되돌리기 위해서는 지우기 연산이 필요하게 되며 이 때 지우기 연산은 블록(Erase Block 또는 Erase Unit) 단위로 실행되어야 한다.

위에 언급한 여러 종류의 플래시 메모리 중 상용화되어 많이 쓰이는 플래시 메모리는 NOR 타입과 NAND 타입이다. <표 1>은 NAND 타입과 NOR 타입 플래시 메모리의 특징을 요약한 것이다.

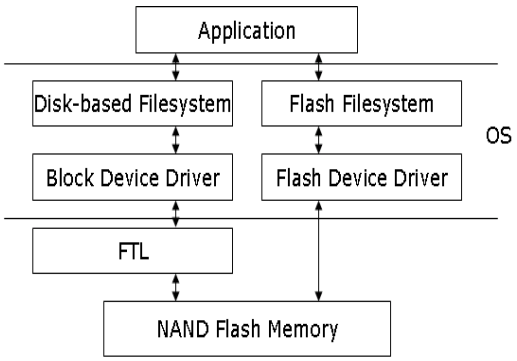
<표 1> NOR 플래시 메모리와 NAND 플래시 메모리[8]

Device	Current (mA)		Access time (4 KB)		
	Idle	Active	Read	Write	Erase
NOR	0.03	32	23 us	28 ms	1.2 sec
NAND	0.01	10	25 us	250 us	2 ms

NAND 타입의 NOR 타입의 플래시 메모리를 비교해보면, 읽기 연산의 속도는 NOR 플래시 메모리가 상대적으로 빠르며 반면 쓰기와 지우기 연산의 속도는 NAND 플래시 메모리가 더 빠르다. 그리고 NAND 플래시 메모리는 하드디스크의 섹터와 유사한 페이지 단위로 읽기와 쓰기 연산을 수행하며 반면 NOR 플래시 메모리는 메인메모리와 유사하게 바이트 단위로 읽기와 쓰기 연산을 수행한다. 이러한 특징들 때문에 NOR 플래시 메모리는 보통 실행 코드 등을 저장하는 XIP(Execute In Place) 용으로 사용되며 NAND 플래시 메모리는 하드디스크를 대체한 2차 저장장치 용도로 사용된다[6].

NAND 플래시 메모리는 512-2048 바이트 크기의 페이지 단위로 읽기와 쓰기 연산이 실행된다. 그리고 32개 또는 64개의 페이지가 하나의 블록(Erase Unit)을 구성한다. 쓰기 연산의 속도

는 읽기 연산에 비해 느리며 지우기 연산의 속도는 쓰기에 비해 더 느리다. 그리고 한 블록에 지우기 연산을 수행할 수 있는 횟수에 제한이 있는데 보통 10,000에서 100,000번의 지우기 연산이 실행된 블록은 닳아서(Wear Out) 더 이상 사용할 수 없게 된다[3][4][9].

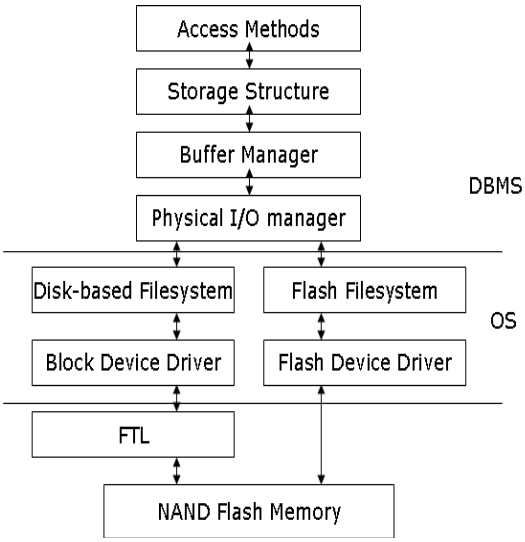


(그림 1) FTL과 플래시 파일시스템의 계층구조

NAND 플래시 메모리를 2차 저장장치로 사용하기 위해 FTL이나 플래시메모리 전용 파일 시스템과 같은 여러 저장 기법들이 제시되어왔다. (그림 1)은 FTL과 플래시 파일 시스템의 계층구조를 나타낸 것이다. FTL은 블록 디바이스와 NAND 플래시 메모리 사이에 위치하여 블록디바이스에서 요청하는 논리적인 블록 주소를 NAND 플래시 메모리 내의 물리적인 주소로 변환시켜주는 역할을 한다. FTL을 사용하면 NAND 플래시 메모리를 하드디스크와 같은 블록 디바이스로 사용할 수 있으며 Ext2나 FAT와 같은 기존 파일시스템을 변경 없이 사용할 수 있게 된다. 반면 플래시 파일 시스템은 파일 시스템이 NAND 플래시 메모리의 특성을 고려하여 FTL 없이 직접 NAND 플래시 메모리를 제어한다. 보통 USB 메모리 스틱이나 SD 카드와 같이 이동성과 호환성이 중요한 제품에는 FTL을 이용한 저장기법이 사용되며 반면 이동성 없이 내장된 플래시 메모리에는 플래시 전용 파일 시스템을 사용한다. FTL과 플래시 파일시스템 모두 논리적인 블록의 주소를 물리적인 블록으로 사상시키는 기능이 필요한데, NAND 플래시 메모리의 페이지 단위로 사상시키는 방법과 블록(Erase Unit) 단위로 사상시키는 방법이 있으며 그 외에 로그 블록을 이용하는 방법[10]등이 제시되었다.

2.2 플래시 메모리 기반의 DBMS

플래시 메모리를 저장장치로 사용하는 컴퓨터 시스템에서 DBMS의 저장 시스템은 (그림 2)와 같은 계층적인 구조를 가지게 된다.



(그림 2) 플래시 메모리 기반 DBMS 저장 시스템의 계층 구조

DBMS는 물리적인 저장장치에 저장된 데이타를 페이지 단위로 읽고 쓰게 된다. 물리적 I/O 관리 계층(Physical I/O Manager)은 저장장치와의 페이지 입출력을 관리한다. 그리고 상대적으로 느린 I/O 연산을 최소화하기 위해서 물리적 저장장치에서 읽어 들인 페이지를 선택적으로 메모리의 버퍼에 남겨두게 된다. 버퍼 관리 계층(Buffer Manager)은 버퍼에 남길 페이지를 결정하고 물리적 I/O 관리 계층에 페이지의 입출력을 요청한다[11].

플래시 메모리를 저장장치로 쓰는 시스템에서는 DBMS의 물리적 I/O 관리 계층이 관리하게 되는 저장장치는 실제 물리적인 저장장치가 아닌 플래시 파일 시스템내의 파일이나 FTL이 에뮬레이션 하는 블록디바이스에 포맷된 디스크 기반 파일 시스템의 파일이 된다. 그리고 물리적 I/O 관리 계층과 실제 저장장치인 플래시 메모리 사이에 OS의 페이지캐시, 파일시스템, 블록 I/O 관리 계층 그리고 FTL 까지 존재하게 된다.

DBMS의 버퍼 관리 계층과 물리적 I/O 관리 계층의 기능이 OS와 FTL의 기능과 중복되게 되며 이러한 상황에서는 버퍼 관리 계층과 같은 DBMS의 구성 요소를 재작성 한다고 해도 이로 인한 성능의 향상은 제한적일 수밖에 없다. 그리고 DBMS에서 데이타를 조작할 때 여러 접근 방법에 대한 각각의 비용을 계산해 비용이 가장 적게 드는 접근 방법을 선택해 실행하게 된다. 이러한 비용을 계산할 때 가장 중요한 요인은 I/O 장치의 특성이 된다. 기존 DBMS는 하드디스크에 알맞게 비용을 계산하도록 작성되어 있으며 장치의 특성 상 읽기/쓰기 연산의 구분 없이 I/O의 횟수만을 변수로 사용한다. 하지만 플래시 메모리를 저장장치로 사용한다면 비용 계산의 방법도 플래시 메모리의 특성에 맞게 고쳐져야 할 것이다. 플래시 메모리는 하드디스크와는 다르게 읽기/쓰기의 속도가 서로 다르며 하드에서는 필요하지 않았던 지우기 연산에 대한 고려가 필요하다. 하지만 DBMS가 플래시 메모리를 직접 제어할 수 없다면 어떤 경로로 데이타에 접근했을 때 플래시 메모리에서 어떤 연산이 얼마나 발생하게 될 지 예측할 수 없게 된다.

따라서 플래시 메모리를 저장 장치로 사용하는 DBMS에 대한 연구를 위해서는 DBMS의 각 구성요소에 대한 변화에 대한 연구와 더불어 플래시 메모리를 직접 제어할 수 있는 DBMS가 필요하게 된다.

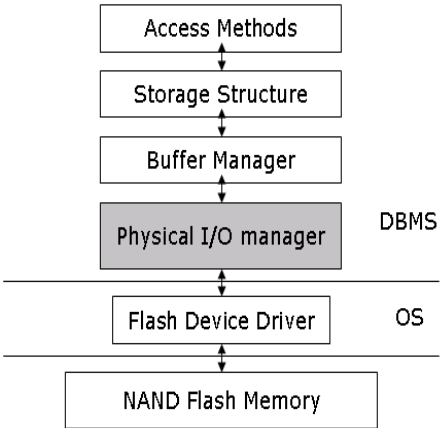
3. 통합 스토리지 시스템

본 논문에서는 플래시 메모리 기반의 DBMS에서 물리적 I/O 관리 계층과 플래시 메모리사이의 계층 구조를 단순화하고 통합하였다. 이를 위해 DBMS의 물리적 I/O 관리 계층이 플래시 메모리의 특성을 고려해 플래시 메모리를 직접 제어할 수 있도록 재작성 되었다.

제안된 시스템은 리눅스 기반에서 WiSS (Wisconsin Storage System)[11]의 스토리지 시스템에 MiniSQL[12] 쿼리 프로세서를 통합한 DBMS에 직접 구현되었다.

3.1 통합된 DBMS 시스템의 계층 구조

(그림 3)은 제안한 시스템의 계층 구조를 나타낸다. 제안한 시스템에서는 DBMS의 물리적 I/O 관리 계층은 리눅스의 MTD[13] 캐릭터 디바이스 인터페이스를 이용해 플래시 메모리를 직접 제어하게 된다.



(그림 3) 통합된 DBMS 저장 시스템의 계층 구조

3.2 물리적 I/O 관리 계층의 재작성

기존 DBMS의 물리적 I/O 관리 계층은 요청된 하드디스크의 블록 주소에 페이지를 읽고 쓰는 일만을 담당하였다. 하지만 제안한 DBMS의 물리적 I/O 관리 계층에서는 요청된 블록 주소를 플래시 메모리의 물리적 페이지 주소로 사상시켜주는 기능이 필요하다. 목적 시스템은 64MB의 내장형 낸드 플래시 메모리를 사용하는 임베디드 보드로, 저장장치 크기에 대한 확장성이 필요하지 않은 시스템이었으므로 페이지 단위로 사상시키는 알고리즘을 사용했다.

DBMS 한 블록의 크기를 플래시 메모리의 페이지 크기인 512 Byte로 설정해 플래시 메모리 한 페이지에 데이타 블록 한 개가 저장된다. 그리고 저장된 블록의 논리적 블록 번호를 플래시 메모리의 스페어 영역에 저장했다. 따라서 DBMS에서 원하는 블록 데이타를 읽기 위해서는 스페어 영역의 정보를 참고해 해당 블록 데이타가 저장된 플래시 메모리 페이지를 검색해야 한다. DBMS가 동작하는 도중에 이러한 검색 부하를 줄이기 위해서 DBMS는 볼륨을 마운트 할 때 스페어 영역의 정보를 참고해 플래시 메모리의 페이지에 대한 사상 테이블을 만들게 된다. 또한 시스템 초기화 때 마운트 시간을 단축

시키기 위해 사상 정보를 램디스크 영역이나 특정 플래시 블록에 저장할 수 있도록 했다.

구현한 DBMS에서는, 어떤 플래시 메모리 페이지 내에 저장된 데이타 블록에 대한 수정이 발생하게 될 때 플래시 메모리의 새로운 페이지를 할당받아 수정된 블록 데이타를 쓰게 된다. 이 때 똑같은 논리 블록 번호가 적힌 플래시 메모리 페이지가 두 개 생기게 된다. 이 때 새로 적힌 플래시 메모리 페이지와 더 이상 유효하지 않은 블록 데이타를 가진 이전의 플래시 메모리 페이지를 구분하기 위해, 새로운 데이타가 적힐 플래시 메모리 블록(지우기 단위)을 할당할 때 순차적으로 번호를 붙였으며 플래시 메모리 블록의 첫 페이지의 스페어 영역에 그 번호를 기록했다. 따라서 같은 논리 블록 번호를 가진 여러 개의 플래시 메모리 페이지가 있을 경우 해당 플래시 메모리 블록의 블록할당번호를 비교해서 가장 최근에 적힌 플래시 메모리 페이지를 알아 낼 수 있었다.

구현한 DBMS에 지속적으로 레코드를 입력했을 때의 물리적인 I/O 요청 트레이스를 분석한 결과 전체 페이지 쓰기 요청의 90%가 시스템 카탈로그 테이블의 업데이트를 위한 것이었다. 레코드가 추가될 때마다 시스템 카탈로그 테이블에 해당 테이블 정보를 업데이트를 위한 페이지 쓰기 요청이 발생하였다. 또한 시스템 카탈로그 테이블의 경우 DBMS의 메모리 버퍼 공간에 저장되지 않는 구조였기 때문에 모든 페이지 요청이 실제 플래시 페이지 쓰기를 발생시켰다. 플래시 메모리에서 같은 페이지에 대한 연속적인 쓰기 요청은 많은 플래시 페이지를 소비해 지우기 연산을 발생시키며, 지우기 연산의 경우 읽기나 쓰기 연산에 비해 느린 연산이므로 결국 전체 시스템의 성능이 감소되었다. 제안된 시스템에서는 이러한 문제를 극복하기 위해서 물리적인 관리 계층에 플래시 메모리 페이지에 대한 캐시를 추가해서 시스템 카탈로그 테이블로 인한 쓰기 연산을 감소시킬 수 있도록 했다.

4. 성능평가

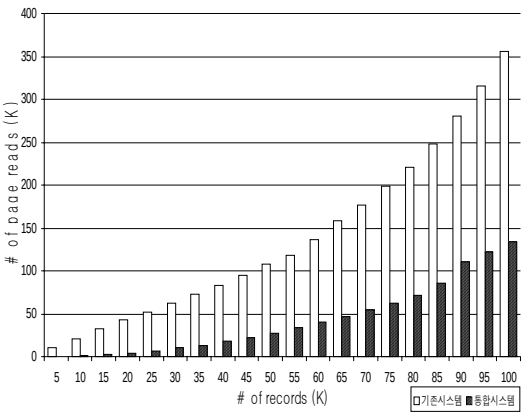
논문에서는 제안된 DBMS 시스템을 Linux-2.6.8을 이용하는 S3C2410 임베디드 보드[14]에 구현하고 테스트하였다. 실험용 임베디드 보드에는 삼성 K9F1208R0B NAND 플래시 메모리[15]가 내장형으로 포함되어 있다.

실험을 위해 WiSS 스토리지 시스템과 MiniSQL 쿼리 프로세서를 통합한 DBMS를 구현했으며 이 DBMS가 플래시 파일 시스템인 YaFFS에서 동작할 때와 DBMS의 물리적 I/O 관리 계층이 직접 MTD를 이용해 플래시를 제어할 때와의 성능을 비교했다.

플래시 메모리는 2장에서 살펴본 바와 같아 읽기 연산의 경우는 빠르지만 쓰기 연산과 지우기 연산이 상대적으로 느리다. 따라서 성능 평가를 위해 쓰기 연산과 지우기 연산을 발생시킬 수 있는 쿼리를 실행시켰다. 각각의 INSERT 쿼리 요청은 새로운 데이타 페이지에 대한 쓰기 요청 뿐 아니라 시스템 카탈로그 페이지에 대한 업데이트 요청을 발생시키게 된다. 각각의 시스템에 1,000,000개의 INSERT 쿼리를 발생시켰을 때 플래시 메모리에서 발생되는 읽기/쓰기/지우기 연산의 횟수와 전체 실행 시간을 측정했다.

4.1 페이지 읽기 횟수

(그림 4)는 1,000,000개의 INSERT 쿼리를 실행시켰을 때 페이지 읽기에 대한 실험 결과이다.

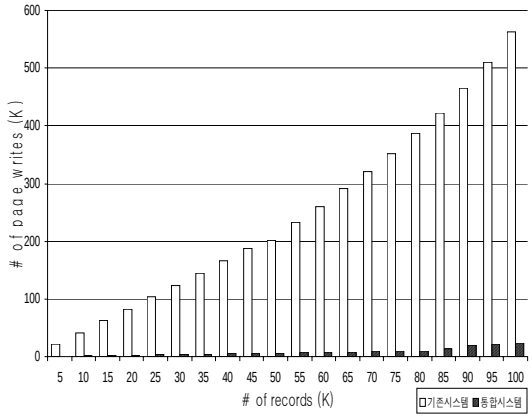


(그림 4) 페이지 읽기 요청 횟수

DBMS에서 레코드를 저장할 때 시스템 카탈로그 테이블에 대한 검색이 이루어지므로 읽기 연산이 지속적으로 발생하게 된다. YaFFS 파일 시스템을 사용한 시스템의 경우는 데이타 페이지에 대한 읽기 외에도 파일시스템 메타데이타

에 대한 읽기가 추가적으로 발생하지만 통합된 시스템의 경우에는 실제로 읽어야 하는 데이타 페이지 외에 추가적으로 읽어야 할 페이지가 없기 때문에 읽기 횟수가 크게 감소한 것을 확인할 수 있다.

4.2 페이지 쓰기 횟수

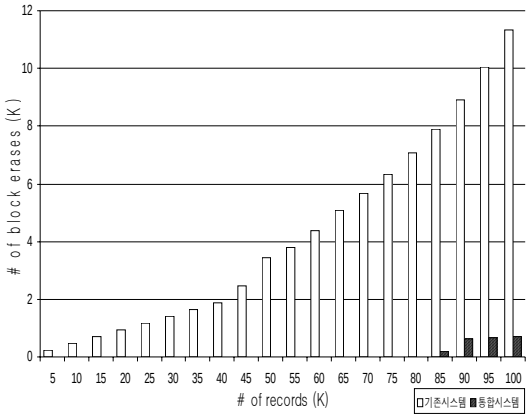


(그림 5) 페이지 쓰기 요청 횟수

(그림 5)는 1,000,000개의 INSERT 쿼리를 실행시켰을 때 페이지 쓰기에 대한 실험 결과이다. YaFFS를 이용한 시스템에서는 시스템 카탈로그 페이지와 같이 버퍼 관리 계층을 거치지 않는 모든 페이지 쓰기 요청에 대해 물리적인 페이지 쓰기가 발생할 뿐 아니라 파일 시스템의 메타데이타 업데이트가 지속적으로 발생한다. 하지만 통합된 시스템에서는 물리적 I/O 관리 계층에서 INSERT 쿼리 하나당 한 번씩 발생하는 시스템 카탈로그 페이지에 대한 쓰기 요청을 효과적으로 캐시하며, 파일시스템의 유지를 위한 추가적인 페이지 쓰기 요청도 없기 때문에 플래시 메모리의 페이지 쓰기가 크게 감소된 것을 확인할 수 있다.

4.3 블록 지우기 횟수

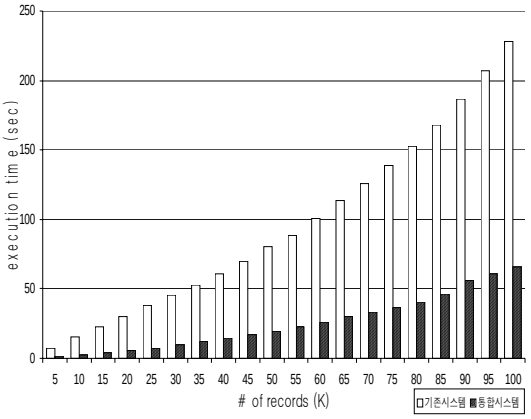
(그림 6)은 블록 지우기 연산의 횟수를 비교한 것이다. YaFFS를 사용하는 기존 시스템에 비해 적은 페이지를 소비하기 때문에 블록 지우기 요청도 크게 감소된 것을 알 수 있다.



(그림 6) 블록 지우기 요청 횟수

4.4 수행시간

(그림 7)은 전체 시스템의 수행 시간을 측정한 결과이다. 플래시 메모리의 물리적인 읽기, 쓰기, 지우기 요청이 크게 감소되어 70% 이상 성능이 향상된 것을 알 수 있다. 제안된 시스템에서는 중복된 기능을 가진 계층의 통합으로 플래시 메모리에 파일시스템 메타데이타와 같은 추가적인 쓰기 연산의 감소시키고 시스템 카탈로그 페이지를 효과적으로 캐싱했다. 따라서 제안된 시스템은 기존 시스템보다 더 적은 양의 페이지를 소비해 플래시 메모리에서 상대적으로 느린 지우기 연산을 크게 감소시켜 수행 시간이 단축된 것을 확인할 수 있다.



(그림 7) 수행시간

5. 추후연구 및 결론

본 논문에서는 기존의 플래시 메모리 기반의 DBMS와 OS의 저장 시스템의 계층을 통합해 플래시 메모리를 DBMS에서 직접 제어할 수 있는 시스템을 제안했으며 이를 실제 시스템에서 구현했다. 실제 구현된 시스템과 기존 시스템과의 성능 비교를 통해 통합된 시스템이 효과적으로 플래시 메모리를 제어함으로써 시스템 성능이 크게 향상되었음을 보였다.

향후 연구로 기존의 하드디스크 기반의 DBMS의 버퍼 관리 알고리즘, 인덱스 구조, 비용 계산을 통한 access path 선택, external sort 알고리즘 등을 플래시 메모리에서 효과적으로 동작하도록 개선시키는 연구를 수행할 계획이다.

참 고 문 헌

[1] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, J.A. Tauber, Storage alternatives for mobile computers, in: Proceedings of the 1st Symposium on Operating Systems Design and Implementation, November 1994, pp. 25 - 37.

[2] K. Saisho, A. Fukuda, Effect of memory allocation on flash memory file system, IPSJ J. 42 (6) 027, June 2001.

[3] H. Kim and S. Lee., "A New Flash Memory Management for Flash Storage System," In 32nd Annual Intl. Computer Science and Applications Conference, October 1999.

[4] Chanik Park, Jeong-Uk Kang, Seon-Yeong Park, Jin-Soo Kim, "Energy-aware demand paging on NAND flash-based embedded storages," Proceedings of the 2004 international symposium on Low power electronics and design table of contents, pp338 - 343, 2004

[5] Intel. Understanding the Flash Translation Layer (FTL) Specification. Application Note AP-684, Intel Corporation, December 1998.

[6] D. Woodhouse, "JFFS: The journaling flash file system," July 2001, presented in the Ottawa Linux Symposium, July 2001 (no proceedings); a 12-page article available from: http://sources.redhat.com/jffs2/jffs2.pdf

[7] YAFFS, A Flash Filesystem for Embedded Use. Available from: http://www.yaffs.net/

[8] Samsung Electronics: NAND flash memory & Smart Media data book, 2004

[9] ZEINALIPOUR-YAZTI, D., LIN, S., KALOGERAKI, V., GUNOPULOS, D., AND NAJJAR, V. MicroHash: An efficient index structure for flash-based sensor devices. In USENIX FAST (2005).

[10] Sang-Won Lee, Dong-Joo Park, Tae-Sun Chung., Dong-Ho Lee, Sangwon Park, Ha-Joo Song, "A Log Buffer based Flash Translation Layer using Fully Associative Sector Translation", ACM Transactions on Embedded Computing Systems

[11] Hong-Tai Chou, David J. DeWitt, Randy H. Katz, Anthony C. Klug: Design and Implementation of the Wisconsin Storage System. Softw., Pract. Exper. 15 (10): 943-962(1985)

[12] Hughes Technologies, MiniSQL available from: http://www.hughes.com.au/products/msql/

[13] Memory Technology Device(MTD) Subsystem for Linux, available from: http://www.linux-mtd.infradead.
org

[14] FALINUX, EZ-S2410 available from:http://falinux.com
/zproducts/ez-s2410.php

[15] Samsung Elec., "64Mx8 Bit NAND Flash Memory", available from: http://www.datasheetarchive.com/pdf/
1880138.pdf

심 효 기

2005년 : 한양대학교 도시공학과 (학사)

현 재 : 한양대학교 전자컴퓨터통신공학과(석사)

관심분야 : 데이터베이스, 플래시 메모리 기반 저장 시스템

윤 경 훈

2006년 : 한양대학교 컴퓨터교육과 (학사)

현 재 : 한양대학교 전자컴퓨터통신공학과(석사)

관심분야 : 플래시메모리 기반 저장장치, 파일 시스템, 임베디드 시스템

박 성 민

2005년 : 한양대학교 컴퓨터교육과 (학사)

2007년 : 한양대학교 전자컴퓨터통신공학과 (석사)

현 재 : 한양대학교 전기컴퓨터통신공학과(박사)

관심분야 : 파일시스템, 플래시메모리 기반 저장장치

정 호 영

2004년 : 한양대학교 재료공학부 (학사)

2006년 : 한양대학교 정보통신대학원 (석사)

현 재 : 한양대학교 전자컴퓨터통신공학과(박사)

관심분야 : 데이터베이스, 플래시 메모리 기반 저장장치, 임베디드 시스템

강 수 용

1996년 : 서울대학교 수학과(학사)

1998년 : 서울대학교 전산과학과 (석사)

2002년 : 서울대학교 전기컴퓨터공학부(박사)

2003년～현 재 : 한양대학교 컴퓨터교육과 교수

관심분야 : 멀티미디어 시스템, 분산시스템, 플래시메모리 기반 저장시스템, e-Learning

차 재 혁

1987년 : 서울대학교 계산통계학과 (학사)

1991년 : 서울대학교 컴퓨터공학과 (석사)

1997년 : 서울대학교 컴퓨터공학과 (박사)

1997년～1998년 : 한국학술진흥재단 부설 첨단학술정보센터 선임연구원

1998년～2001년 : 한양대학교 컴퓨터교육과 조교수

2001년～현 재 : 한양대학교 정보통신대학 정보통신학부 교수

관심분야 : XML, 데이터베이스, 플래시 메모리 기반 저장 시스템, 멀티미디어 콘텐츠 적응화, e-Learning