NAND형 플래시메모리를 위한 플래시 압축 계층의 설계 및 성능평가

(Design and Performance Evaluation of a Flash Compression Layer for NAND-type Flash Memory Systems)

임 근 수 † 반 효 경 †† 고 건 †††
(Keun Soo Yim) (Hyokyung Bahn) (Kern Koh)

요 약 최근 휴대용 정보기기의 사용이 급증함에 따라 NAND형 플래시메모리를 시스템의 보조기억장치로 사용하는 사례가 급증하고 있다. 하지만, 전통적인 보조기억장치인 하드디스크에 비해 NAND형 플래시메모리는 단위 공간당 비용이 수십배 가량 높아 저장 공간의 효율적인 관리가 필요하다. 저장 공간을 효율적으로 사용하게 하는 대표적인 방법으로 데이타 압축 기법이 있다. 하지만, NAND형 플래시메모리에서는 압축 기법의 적용이 쉽지 않다. 이는 NAND형 플래시메모리가 페이지 단위 입출력만을 지원하여 압축 데이타가 플래시 페이지보다 작은 경우 내부 단편화 현상을 발생시켜 압축의 이득을 심각하게 감쇄시키기 때문이다. 이러한 문제를 해결하기 위해 본 논문에서는 작은 크기의 압축 데이타를 쓰기 버퍼를 통해 그룹화한 후 하나의 플래시 페이지에 저장하는 플래시 압축 계층을 설계하고 성능을 평가한다. 성능 평가 결과 제안하는 플래시 압축 계층은 플래시메모리의 저장 공간을 40% 이상 확장하며 쓰기 대역폭을 크게 개선함을 확인할 수 있었다.

키워드 : 플래시메모리, 플래시 변환계층, 플래시 압축계층, 데이타 압축

Abstract NAND-type flash memory is becoming increasingly popular as a large data storage for mobile computing devices. Since flash memory is an order of magnitude more expensive than magnetic disks, data compression can be effectively used in managing flash memory based storage systems. However, compressed data management in NAND-type flash memory is challenging because it supports only page-based I/Os. For example, when the size of compressed data is smaller than the page size, internal fragmentation occurs and this degrades the effectiveness of compression seriously. In this paper, we present an efficient flash compression layer (FCL) for NAND-type flash memory which stores several small compressed pages into one physical page by using a write buffer. Based on prototype implementation and simulation studies, we show that the proposed scheme offers the storage of flash memory more than 140% of its original size and expands the write bandwidth significantly.

Key words : Flash memory, flash translation layer, flash compression layer, data compression

1. 서 론

최근 컴퓨팅 패러다임이 데스크 탑 PC 중심에서 언제 어디서나 원하는 정보와 컴퓨팅 파워를 사용할 수 있게 하는 유비쿼터스 컴퓨팅 환경으로 전환되고 있다. 이에 따라 휴대폰, PDA, 디지털카메라 등 휴대용 정보기기의 사용이 급증하고 있다. 이와 같은 휴대 기기에서는 일반적으로 하드디스크 대신 플래시메모리를 보조기억장치로 사용한다. 이는 플래시메모리가 가볍고 소음과 진동이 없으며 물리적인 충격에 강해 휴대가 용이하고, 또한 전력을 적게 소모해 동일 배터리 파워로 장시간 사용할 수 있는 특성을 지녔기 때문이다[1].

플래시메모리는 일종의 EEPROM[1]으로 바이트 단위 I/O를 지원하는 NOR형과 페이지 단위의 I/O만을 지원

† 정 회 원 : 삼성 종합기술원 연구원
keunsoo.yim@samsung.com
†† 비 회 원 : 이화여자대학교 컴퓨터학과 교수
bahn@ewha.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학부 교수
kernkoh@june.snu.ac.kr
논문접수 : 2004년 8월 11일
심사완료 : 2004년 12월 16일

1) Electrically Erasable and Programmable Read Only Memory의 약어로 전기적으로 지우고 쓸 수 있는 비휘발성 메모리를 말한다.

하는 NAND형의 두 가지 대표적인 유형이 있다. NOR형 플래시메모리는 읽기 속도는 빠른 반면 쓰기 속도가 느려 주로 프로그램 코드용 메모리로 사용한다[2]. 반면 NAND형 플래시메모리는 쓰기 속도가 빠르고 단위 공간당 단가가 낮아 주로 대용량 데이타 저장장치로 사용한다[3]. 표 1은 플래시메모리와 DRAM, 하드디스크를 성능 및 비용의 관점에서 비교한 것이다. 표에서 보는 바와 같이 플래시메모리는 상대적으로 낮은 단가로 빠른 읽기속도를 제공하는 비휘발성 저장장치임을 알 수 있다.

한편, 플래시메모리를 다양한 휴대 기기에서 널리 사용하기 위해서는 크게 두 가지 성능상의 문제를 개선해야 한다. 첫째는 쓰기 대역폭 문제이다. 플래시메모리에 데이타를 쓰기 위해서는 쓰기 단위보다 큰 단위에 대한 삭제 연산을 선행해야 하며, 쓰기 연산 자체의 수행 속도도 다른 메모리 소자에 비해 느리기 때문에 실시간으로 멀티미디어 데이타를 저장하는 경우 QoS 보장이 어렵다. 둘째는 저장 공간 문제로 플래시메모리는 하드 디스크에 비해 단위 공간당 비용이 수십배 가량 높다. 따라서, 저장 공간을 효율적으로 관리하기 위한 방법이 필요하다.

이 두 가지 문제점을 완화할 수 있는 효과적으로 방법으로 데이타 압축 기법이 있다. 데이타를 압축하여 전송하면 쓰기 대역폭이 확장된 것과 동일한 효과를 얻을 수 있으며 저장 시에도 공간 확장 효과를 얻을 수 있다. 그러나, NOR형 플래시메모리를 대상으로 고안된 기존의 압축기법[4,5]과 달리 NAND형 플래시메모리에 압축기법을 적용하는 것은 기술적인 어려움이 따른다. 그 이유는 NAND형 플래시메모리가 페이지 단위의 입출력만을 지원하기 때문이다. 예를 들어 압축된 데이타의 크기가 플래시 페이지의 크기보다 작은 경우 그림 1과 같이 내부 단편화 현상이 발생하며 이는 압축에 의한 효과를 크게 떨어뜨린다. 극단적으로 입출력 단위와 플래시 페이지의 크기가 같은 경우에는 NAND형 플래시메모리에 압축기법을 적용하더라도 공간상의 이득을 전혀 얻을 수 없다. 내부 단편화 영역에 데이타를 저장하기 위해서는 이보다 큰 단위에 걸쳐서 삭제 연산을 선행해야하고 페이지 전체에 대한 쓰기 연산을 다시 수행해야 하는 등 현실적인 활용 가능성이 거의 없기 때문이다. 이러한 내부 단편화 문제는 기존의 RAM이나 NOR형 플래시메모리의 압축 기법에서는 발생하지 않았는데 그 이유는 이들 메모리 소자의 경우 바이트 단위의 입출력을 지원하기 때문이다.

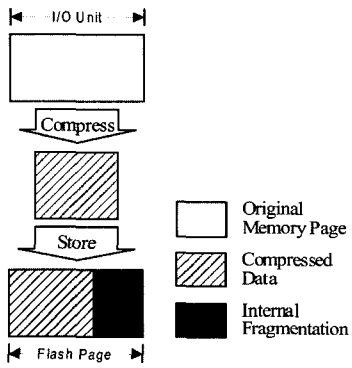

그림 1 NAND형 플래시메모리의 내부 단편화 문제

본 논문에서는 내부 단편화 문제를 효율적으로 개선하기 위하여 NAND형 플래시메모리를 위한 플래시 압축 계층(Flash Compression Layer, FCL)을 제안한다. 제안하는 플래시 압축 계층은 작은 크기의 압축 데이타를 쓰기 버퍼를 통하여 그룹화하여 하나의 플래시 페이지에 저장한다. 성능평가 결과 널리 사용되는 벤치마크용 워크로드에 대해 제안하는 플래시 압축계층이 플래시메모리의 저장 공간을 40% 이상 확장하며 쓰기 대역폭을 크게 개선함을 보인다. 또한 본 논문에서는 비록 압축 데이타를 읽을 때 복원하는 연산이 필요하지만 읽기 연산의 속도는 압축 기법을 사용하지 않은 경우보다 느려지지 않음을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 살펴본다. 3장에서는 제안하는 플래시메모리 압축 기법에 대해 설명하며, 4장에서는 이의 성능을 실측과 시뮬레이션을 통하여 평가한다. 끝으로 5장에서는 논문의 결론을 맺는다.

표 1 다양한 메모리 소자의 특성비교

	Read	Write	Erase	Cost/MB
DRAM	60ns (2B) 2.6μs (512B)	60ns (2B) 2.6μs (512B)	-	30~40
NOR-type Flash	150ns (1B) 15μs (512B)	211μs (1B) 3.5ms (512B)	1.2s (128KB)	20~30
NAND-type Flash	10μs (1B) 36μs (512B)	201μs (1B) 226μs (512B)	2ms (16KB)	10~20
Disk	12.4ms (512B)	12.4ms (512B)	-	1

2. 관련 연구

이 장에서는 플래시메모리의 종류와 특성에 살펴본 후, 이를 사용하여 저장장치를 구성하는 두 가지 방법인 플래시 변환계층(Flash Translation Layer, FTL)과 플래시메모리 전용 파일시스템에 대해 알아본다.

플래시메모리의 셀은 셀의 소스(Source)와 드레인(Drain)의 연결 상태는 중앙 상단에 위치한 제어 게이트(Control Gate)와 하단에 위치한 부동 게이트를 사용해 제어된다[12]. 추가적으로 제어 게이트가 있는 것을 제외하고는 부동 게이트에 일정 값 이상의 전압을 인가함으로써 소스와 드레인 사이에 전기적 채널을 형성해 소스의 값을 드레인으로 전달하는 일반적인 MOS-FET 트렌지스터와 유사한 구조이다[13]. 종류에 따라 NOR형은 nMOS-FET와 NAND형은 pMOS-FET와 유사한 형태로 도핑된다.

NOR형 플래시메모리는 초기에 부동 게이트에 양이온을 충전하고 있어 논리 '1'값을 나타낸다. 셀 값은 소스에 전압을 인가한 후 드레인의 값을 확인함으로써 읽을 수 있다. 만약 부동 게이트의 값이 논리 '1'이면 소스와 드레인 사이의 베이스(Base) 영역에 음이온 채널이 형성되어 소스의 값이 드레인에 전달되고, 그렇지 않으면 소스와 드레인은 전기적으로 연결되지 않아 개방(open) 상태가 된다. 쓰기 연산 시에는 제어 게이트에 높은 전압을 인가함으로써 베이스의 음전하를 부동 게이트에 축적해 부동 게이트의 값을 논리 '0'으로 설정한다. 삭제 연산은 이와는 반대로 게이트를 접지시키고 베이스에 고압을 인가해 부동 게이트에 축적된 음전하를 방출한다. 이때 삭제 단위(블록 또는 세그먼트)는 인접한 페이지(읽기 및 쓰기 단위)의 집합으로 구성되며 플래시메모리의 종류와 내부 구조에 따라 삭제 연산의 수행 시간이 길게는 1~2초에 달하며, 각 블록별로 삭제 연산을 수행할 수 있는 총 횟수에 제한이 있다. 그리고 NAND형 플래시메모리는 이와는 정반대로 동작한다.

NAND형과 NOR형 플래시메모리는 위의 셀을 사용해 전체 메모리 회로를 구성하는 단계에서 그 차이가 확연하게 드러난다. NOR형은 그림 2의 왼쪽과 같은 구조를 가지며 진하게 표시된 셀의 값은 B/L과 W/L 3에 전압을 인가하고 이외의 W/L과 C 노드를 접지시켜 B/L의 값을 확인함으로써 읽을 수 있다. 만약 선택된 W/L 3의 부동 게이트에 축적된 값이 '0'이라면 해당 셀의 소스와 드레인이 연결되어 B/L은 접지 노드 C와 연결돼 전압이 떨어지며, 축적된 값이 '1'이라면 B/L은 인가된 전압을 유지한다. NAND형의 경우 그림 2의 오른쪽과 같이 셀을 배치하고, 진하게 표시된 셀들의 값은 W/L(word line) i를 접지시키고 이를 제외한 B/L(bit line), SSL, GSL, W/L 노드들에 특정전압을 인가한 후 B/L에 인가되는 전압을 확인함으로써 읽을 수 있다. 만약 W/L i에 해당하는 셀의 값이 '1'이라면 셀 내부의 소스와 드레인이 연결되어 B/L은 접지된 것이 돼 전압이 떨어진다. 이와 같은 구조적 차이로 인해 NOR형은 바이트 단위 입출력을 지원하며 NAND형은 페이지(512 또는 2048바이트) 단위의 입출력만을 지원한다.

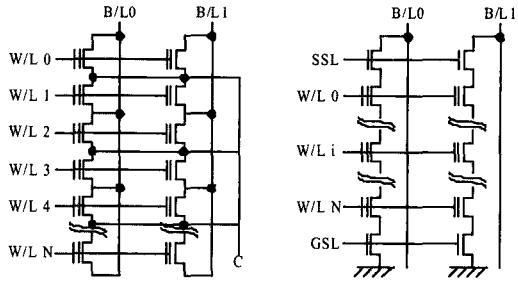

그림 2 NOR형(좌)과 NAND형(우) 플래시메모리의 회로도

이와 같은 플래시메모리를 바탕으로 저장장치를 설계하는 방법은 크게 플래시 변환계층을 이용하는 방법과 전용 파일시스템을 이용하는 방법이 있다. 플래시 변환계층은 플래시메모리의 삭제 연산을 감추기 위한 미들웨어로 파일시스템과 플래시메모리 사이에 위치한다[6-10]. 이때 삭제 연산은 쓰기 연산 시에 파일시스템이 생성한 논리 주소를 플래시메모리상의 기 삭제된 영역에 대한 물리주소로 변환함으로써 감춰진다. 또한 주소 변환 시에 플래시 블록들을 균등하게 사용함으로써 제한된 삭제 한계에 모든 물리 블록들이 비슷한 시점에 도달하도록 하여(wear-leveling) 플래시메모리의 평균 수명을 연장한다. 이를 통하여 하드 디스크와 같은 저장장치 인터페이스를 제공함으로써 상위 층에서는 디스크용 파일시스템을 사용해 플래시메모리를 제어할 수 있게 된다. 플래시 변환계층은 구현 방식에 따라 호스트에 독립된 하드웨어[6-8]와 호스트 내부의 소프트웨어[9-11] 형태로 구현된다.

플래시 변환계층은 주소를 변환하는 단위에 따라 크게 페이지 단위 변환 기법과 블록 단위 변환 기법으로 나뉜다. 그림 3(a)에 제시된 것과 같이 페이지 변환 기법은 정교하게 주소를 변환함으로써 높은 성능을 발휘하는데 반하여 주소 변환 테이블의 크기가 커져 제작 비용을 증가시키는 단점이 있다. 반대로 블록 변환 기법은 그림 3(b)와 같이 상대적으로 비정교하게 주소를 변환해 주소 변환 테이블의 크기는 작지만, 내부의 단 하나의 페이지에 대한 수정 연산이 발생하여도 전체 블록

을 삭제하고 갱신해야 하는 추가 비용을 갖는다. 더우기 갱신 과정에서 전원이 나가는 등의 예외 상황이 발생하면 갱신하는 데이타의 일관성을 깨뜨릴 수 있다는 문제가 있다.

이러한 블록 변환 기법의 단점을 보완한 기법으로 교체 블록 기법이 있는데, 이 기법은 그림 3(c)와 같이 블록 내부의 페이지에 대한 갱신 요청이 발생하면 교체 블록을 할당하여 쓰기를 수행하고 이를 연결리스트를 사용해 관리하여 향후 읽기 연산이 발생하면 이 리스트를 역순으로 검색하여 가장 최근에 수정된 데이타에 접근하는 방식이다. 더욱 최근에 제안된 로그 블록 기법은 그림 3(d)와 같이 페이지 변환 기법과 블록 변환 기법을 병합하여, 비교적 큰 단위로 발생하는 순차 입출력은 블록 변환기법으로 처리하고 작은 단위로 요청되는 임의 입출력은 페이지 변환 기법으로 처리하는 방식이다[7]. 이를 통하여 주소 변환 테이블의 크기를 줄이고도 높은 성능을 얻을 수 있다. 또한 로그 블록 기법과 비슷한 설계 철학을 가지고 운영체제의 버디 시스템의 기법을 응용하여 플래시 변환 계층을 효율적으로 설계한 연구도 있다[11].

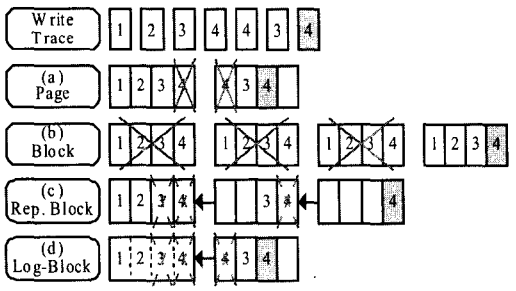

그림 3 플래시 변환계층의 설계기법에 따른 동작예제

한편, 플래시 변환계층을 사용하여도 호스트에서 디스크용 파일시스템을 사용해야 하는데, 이는 하나의 작업을 두 개의 층을 두어 처리하는 것이 되어 쓰기 비율이 높은 시스템의 경우 플래시메모리 기억 공간상의 비효율성을 야기할 수 있다. 뿐만 아니라 플래시 변환계층은 이미 국제특허로 등록이 되어 있어서 NOR형 플래시메모리의 경우 PCMCIA 카드, NAND형 플래시메모리의 경우 DiskOnChip 장치에 기반하지 않는 경우 법률상의 제약이 따른다.

이러한 문제점을 개선하기 위하여 최근에 JFFS (Journaling Flash File System)[14]와 YAFFS(Yet Another Flash Filing System)[15] 등과 같은 플래시메모리 전용 파일시스템이 개발되었다. 이들 파일시스템은 로그 구조 파일시스템[16]과 유사하게 쓰기 연산 시에 데이타를 로그 형태로 기록하고 읽기 연산 시에 이를 역순으로 검색하여 가장 최신의 데이타에 접근하는 방식을 사용한다. 또한, 플래시메모리의 저장 공간을 임계 비율 이상 사용하게 되면 가장 유효한 페이지의 수가 적은 블록을 선택해 유효한 페이지를 복사한 후 해당 블록을 삭제하는 작업(garbage collection)을 수행한다. JFFS와 YAFFS는 각각 NOR형와 NAND형 플래시메모리를 주 대상으로 하여 개발되었다.

관련 연구를 통하여 알 수 있듯이 그동안 플래시메모리 관련 저장장치에 대한 연구는 삭제 연산을 효율적으로 감추는데 초점이 맞추어져 있었다. 하지만 삭제연산을 감추어도 쓰기 연산이 다른 메모리 장치와 비교해 상대적으로 느리며 여전히 저장 공간 문제를 그대로 가지고 있기 때문에 본 논문에서 제안하는 플래시 압축 계층은 플래시메모리 분야에서 중요한 연구 과제 중의 하나이다. 실제로 NOR형을 대상으로 한 JFFS의 경우 설정에 따라 데이타를 압축해 저장할 수 있으며, NOR형과 같은 바이트 단위의 접근이 가능한 비휘발성 메모리를 위한 파일시스템 상의 메타데이타를 효율적으로 압축하는 연구가 있었다[17]. 하지만 그동안 본 논문에서 제안하는 것과 같이 페이지 단위의 입출력만을 지원하는 NAND형 플래시메모리를 위한 연구는 없었다.

3. 제안하는 플래시 압축 계층

이 장에서는 본 논문이 제안하는 플래시 압축 계층에 대해 기술한다. 플래시 압축 계층에서 사용하는 압축 데이타의 관리 기법으로 본 논문에서는 3.1절의 큰 단위 압축기법(large-unit compression scheme, LCS)과 3.2절의 내부 그룹화 기법(internal packing scheme, IPS)의 두 가지 방법을 제안하고 이를 설계하는 과정에 대해 기술한다.

3.1 큰 단위 압축 기법

NAND형 플래시메모리에서 발생하는 내부 단편화 문제는 큰 입출력 단위를 사용하거나 작은 크기의 플래시 페이지를 사용함으로써 완화시킬 수 있다. 그림 4는 플래시 페이지 크기보다 두 배 큰 입출력 단위를 사용한 경우 내부 단편화 현상을 상대적으로 줄일 수 있음을 보이는 예제이다. 이와 유사하게 플래시 페이지의 크기를 줄이는 방법을 사용하더라도 내부 단편화 현상을 완화시킬 수 있다. 하지만, 입출력 단위나 플래시 페이지의 크기를 변경하는 것은 심각한 문제를 야기할 수 있다. 큰 입출력 단위는 입출력 연산의 시간을 지연시킬 뿐만 아니라 인접한 페이지에 대한 중복 연산을 초래한다. 또한, 작은 플래시 페이지 크기는 플래시메모리의 회로 설계를 복잡하게 하며 플래시 변환 계층에 사용되는 주소 변환 테이블의 크기를 증가시키게 된다.

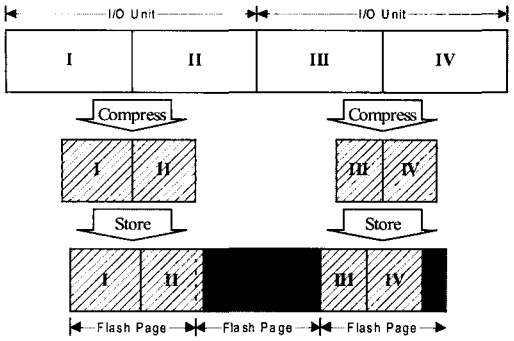

그림 4 큰 단위 압축 기법의 압축 데이타 관리 예제 (I-IV: 메모리 페이지)

이러한 문제점을 해결하기 위해 다음 절에서는 입출력 단위나 플래시 페이지 크기를 변경하지 않고 작은 크기의 압축 데이타를 쓰기 버퍼를 통하여 그룹화하고 이를 하나의 플래시 페이지에 기록하는 내부 그룹화 기법에 대해 소개한다.

3.2 내부 그룹화 기법

내부 그룹화 기법은 그림 5에 제시된 것과 같이 각각의 메모리 페이지를 개별적으로 압축한다. 멀티미디어 데이타의 경우 압축 효율이 높지 않으며 경우에 따라서는 압축 데이타의 크기가 원 데이타의 크기보다 커지는 문제(data explosion)가 발생한다. 따라서 제안하는 기법에서는 이와 같이 압축 효율이 임계 값 이상인 데이타들은 압축하지 않고 플래시메모리에 직접 저장하는 선택적 압축기법을 사용한다. 반면에 압축된 페이지들은 쓰기 버퍼를 사용하여 하나의 플래시 페이지에 저장할 수 있는 크기로 그룹화한다. 이때 쓰기 버퍼는 바이트 단위의 입출력을 지원해야 하며 예외 상황에 대응해 데이타의 일관성을 유지하기 위해서는 비휘발성이어야 한다. 이러한 조건을 만족하는 메모리 소자로는 배터리 부착형 RAM, NOR형 플래시메모리, 그리고 차세대 메모리 소자인 PRAM, MRAM, FRAM 등이 있다. 이 중에서 NOR형 플래시메모리는 NAND형 플래시메모리보다 쓰기 속도가 느려 쓰기 버퍼로 사용할 경우 플래시 압축 계층의 쓰기 성능이 크게 저하될 수 있다.

그룹화 단계의 목적은 내부 단편화를 최소화하는 것이다. 본 논문에서는 그림 6에 예시된 것과 같은 세 가지 그룹화 정책을 사용한다[18]. 그림에서는 쓰기 버퍼가 3개 있는 경우의 예를 보여주고 있다. 첫 번째 그룹화 정책인 'Best-Fit' 정책에서는 가장 내부 단편화를 적게 발생시키는 쓰기 버퍼에 새롭게 압축된 데이타를 담아가는 형태인 욕심쟁이 알고리즘(greedy algorithm)을 사용해 그룹화하는 것이다. 두 번째 정책인 'Worst-Fit' 정책은 가장 큰 내부 단편화를 발생시키는 쓰기 버퍼에 새롭게 압축된 데이타를 담아가는 형태로 동작한다. 끝으로 'First-Fit' 정책은 주어진 압축 데이타를 저장할 수 있는 가장 먼저 발견되는 쓰기 버퍼에 데이타를 담는 그룹화 기법이다. 그림 6은 이 세 가지 기법을 사용해 주어진 일련의 압축 페이지들을 그룹화한 결과를 나타낸다.

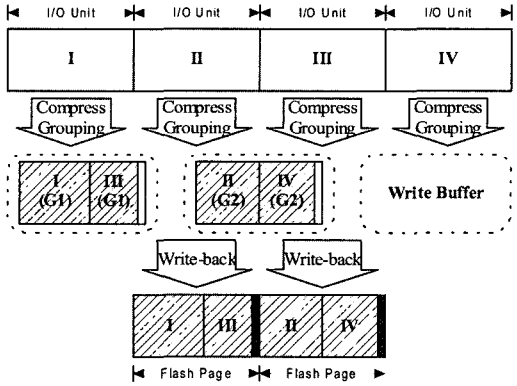

그림 5 내부 그룹화 기법의 압축 데이타 관리 예제 (I-IV: 메모리 페이지)

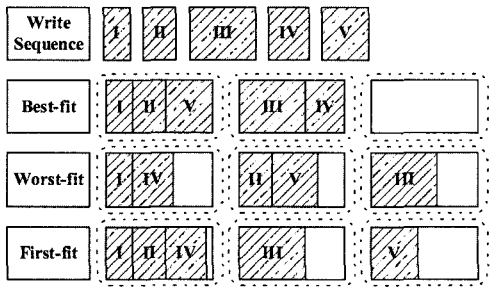

그림 6 세 가지 압축 페이지 그룹화 정책의 적용 예제

이후 쓰기 버퍼 상의 여유 공간이 임계치 이하가 되면 버퍼 내부의 그룹 중 가장 내부 단편화가 적은 그룹을 선택하여 실제로 NAND형 플래시메모리에 저장한다. 이때 압축 페이지의 메타데이타도 함께 저장되는데 메타데이타는 플래시 페이지 자체가 아니라 각 플래시 페이지마다 부착된 여분 공간(spare or out-of-band area)이라 불리우는 16바이트 또는 64바이트 크기의 영역에 저장된다. 메타데이타는 크게 플래시 변환 계층에서 사용되는 해당 플래시 페이지의 논리 주소와 압축 페이지의 크기 그리고 에러 검사 코드이다. 여분 공간의 크기가 제한되어 있기 때문에 하나의 플래시 페이지에 동시에 저장하고 메타데이타를 여분 공간에 기록할 수 있는 최대 압축 페이지의 개수(N)는 수식 (1)과 같은 제한을 갖는다.

$$N=\left|\frac{S_{spare}-S_{ECC}}{\lg\frac{S_{storage}}{S_{page}}+\lg S_{page}}\right|=\left|\frac{S_{spare}-S_{ECC}}{\lg S_{storage}}\right| \quad (1)$$

수식에서 S_{spare}는 여분 공간의 크기를, S_{ECC}는 에러 검사코드 길이를, S_{page}는 플래시 페이지 크기를, 그리고 $S_{storage}$는 플래시 메모리의 총 용량을 의미한다. 일반적으로 N 값은 여분 공간의 크기가 16 바이트와 64 바이트인 경우에는 각각 4와 19로 충분히 많은 수의 압축 페이지를 한 플래시 페이지에 저장할 수 있음을 알 수 있다. 그리고 특허와 같은 다른 기술상의 문제로 이 여분의 공간을 활용하는 것이 어려우면, 쓰기 버퍼의 일정 부분을 대신 활용할 수 있다.

그림 7은 내부 그룹화 기법을 바탕으로 설계한 플래시 압축 계층의 블록 다이어그램으로 크게 하드웨어 압축기와 복원기 그리고 쓰기 버퍼로 구성되어 있다. 이 구조는 데이타 캐싱을 통한 입출력 연산 시간을 단축시키는 장점을 갖는다. 쓰기 연산의 경우 압축을 통해 데이타의 크기가 줄어들었고 쓰기 버퍼의 속도가 저장 장치인 NAND형 플래시메모리보다 빠르기 때문에 연산 시간이 단축된다. 또한, 읽기 연산의 경우에도 요청된 페이지가 쓰기 버퍼에 저장되어 있으면 선인출(pre-fetching) 효과를 얻게 되어 연산 시간이 단축된다. 읽기 연산에서 비록 요청 페이지가 쓰기 버퍼에 존재하지 않는다 해도 여러 연산들이 쓰기 버퍼와 플래시메모리 자체에 서 병렬적으로 처리되므로 입출력 연산 시간의 단축을 기대할 수 있다.

또한 제안하는 플래시 압축 계층은 저장장치 수준의 시스템으로 블록 단위의 일관성 있는 입출력을 보증해야 하는데, 이는 파일시스템 수준의 그것과 비교해 비교적 간단하게 지원할 수 있다. 세부적으로 플래시 압축 계층은 쓰기 연산 시에 데이타를 쓰기 버퍼에 먼저 기록하고 이후에 이와 관련된 물리 주소 등의 메타데이타를 갱신하는 방식을 사용하는데, 이를 통해 안전하게 저장된 데이타에 대한 메타데이타 정보를 항상 유지할 수 있다.

본 논문에서는 다수의 하드웨어적 압축기 중에서 X-RL 압축기를 사용한다. 이는 X-RL 압축기가 플래시 페이지와 같은 작은 크기의 데이타에 대해서도 좋은 압축 효율을 보이며, 압축 및 복원 속도가 입출력 버스의 속도보다 빨라 추가적인 시간 오버헤드가 없으며, 하드웨어 구조도 비교적 간단하기 때문이다. 세부적으로 X-RL 압축기는 X-Match 압축기에 RL(Run-Length) 인코딩을 추가한 것이다[19]. X-Match 압축기는 4-Way 파이프라인 구조를 가지며 데이타 대역폭이 4 바이트여서 초기에 3 사이클 후에 매 1 사이클마다 4 바이트의 데이타를 압축 또는 복원한다. X-RL 압축기는

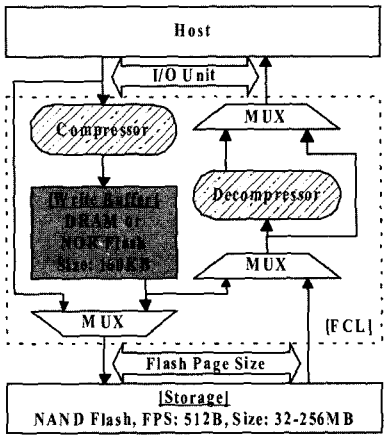

그림 7 내부 그룹화 기법을 사용한 경우의 플래시 압축 계층의 블록 다이어그램

높은 빈도로 나타나는 연속된 값이 0인 데이타를 RL 인코딩을 통해 압축하여 압축 효율을 높인 것으로 압축 속도는 X-Match와 같고 복원 속도는 이보다 빠른데 그 이유는 연속된 값이 0인 데이타를 1 사이클에 복원하기 때문이다. 현재까지 구현된 가장 빠른 X-RL 압축기의 동작 클럭은 800MHz로 I/O 버스의 클럭보다 4배 이상 빠르기 때문에 초기에 소모되는 3 사이클이 감춰져 압축 및 복원에 따른 시간상의 추가비용이 존재하지 않는다.

4. 성능 평가

제안하는 플래시 압축 계층의 성능을 평가하기 위하여 그림 8에 제시된 NAND형 플래시메모리를 내장한 개발 보드를 사용하였다. 부가적으로 다양한 설계 환경하에서 성능을 평가하기 위하여 트레이스 기반 시뮬레이션을 병행해 사용하였다. 벤치마크로는 무손실 압축 알고리즘의 평가 시에 널리 사용하는 Canterbury Corpus 벤치마크를 사용하였다[20].

그림 8 실험 환경

플래시 압축 계층의 설계 목표는 플래시메모리의 기억 공간과 쓰기 대역폭을 확장하는 것이다. 이와 같은 성능을 효율적으로 측정하기 위하여 본 논문에서는 실질 압축률을 성능평가 지표로 정의해 사용한다. 실질 압축률(effective compression rate, ECR)이란 사용된 플래시 페이지의 수를 여기에 저장된 압축되지 않은 메모리 페이지의 수로 나눈 값을 뜻한다. 실질 압축률 외에 본 논문에서는 부가적인 성능 지표로 압축률과 내부 단편화 비율을 함께 정의하였다. 압축률(compression rate, CR)이란 압축 페이지의 크기를 소스 페이지의 크기로 나눈 값이다. 따라서, 압축률이 낮을수록 압축이 더 효율적으로 이루어짐을 나타낸다. 내부 단편화 비율(internal fragmentation rate, IFR)이란 내부 단편화 크기를 소스 페이지 크기로 나눈 값을 뜻한다. 따라서, 실질 압축률은 평균 압축률과 평균 내부 단편화 비율의 합으로 구할 수 있다. 실질 압축률이 구해지면 쓰기 대역폭의 확장 비율과 기억 공간의 확장 비율은 각각 1/CR-1과 1/ECR-1를 통해 구할 수 있다.

그림 9는 벤치마크 프로그램의 압축 단위에 따른 압축률을 나타내고 있다. 그림을 통해 큰 압축 단위가 더 우수한 압축률을 나타내지만, 압축 단위가 2K 바이트를 넘어서면 압축률이 특정값으로 수렴함을 알 수 있다. 따라서, 압축률 상의 이득을 극대화하기 위해서는 압축 단위가 최소 2K 바이트 이상이어야 함을 알 수 있다. 그러나, 압축 단위가 커지면 입출력 시간이 길어지며 하드웨어 설계도 더 복잡해진다. 현재 널리 사용되는 NAND형 플래시메모리의 경우 플래시 페이지 크기와 입출력 단위가 모두 512 바이트이기 때문에 본 논문에서는 내부 그룹화 기법의 압축 단위를 512 바이트로 설정하였다. 한편, 큰 단위 압축 기법에서는 압축률과 내부 단편화 비율 측면에서 이득을 취할 수 있도록 압축 단위를 1K 또는 2K 바이트로 설정하였다. 이는 각각 플래시 페이지 크기의 2배와 4배에 해당하는 값이다.

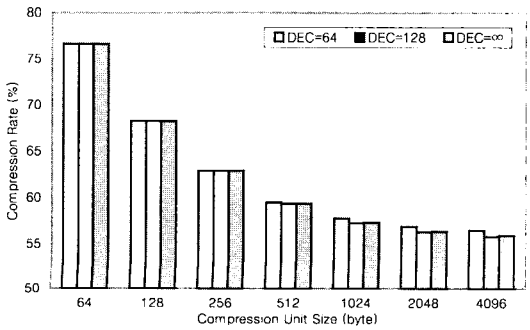

그림 9 압축 단위와 압축기의 사전 크기(DEC)에 따른 압축률

또한 그림 9는 압축기의 사전 크기에 따른 압축률을 보이는데, 여기서 사전은 X-RL 압축기가 원 데이타와 압축 데이타 사이의 연관성을 찾는 과정에서 사용하는 임시 저장소이다. 실험 결과 사전 크기가 128 엔트리인 경우 최적의 압축률을 얻음을 알 수 있다. 하나의 사전 엔트리 크기가 4 바이트이기 때문에 전체 사전의 크기는 512 바이트가 되어 이를 SRAM을 사용해 구성하는 경우에도 설계 비용을 크게 증가시키지 않음을 알 수 있다.

페이지 단위 입출력만을 지원하는 NAND형 플래시메모리에서 압축 데이타를 효율적으로 관리하기 위해서는 내부 단편화를 최소화해야 한다. 그림 10은 큰 단위 압축 기법을 사용한 경우 입출력 단위와 플래시 페이지 크기에 따른 내부 단편화 비율을 보여주고 있다. 그림을 통해 플래시 페이지의 크기가 통상적인 512 바이트이고 입출력 단위가 플래시 페이지 크기의 2배와 4배인 경우 내부 단편화 비율은 각각 26.2%와 20.9%임을 알 수 있다.

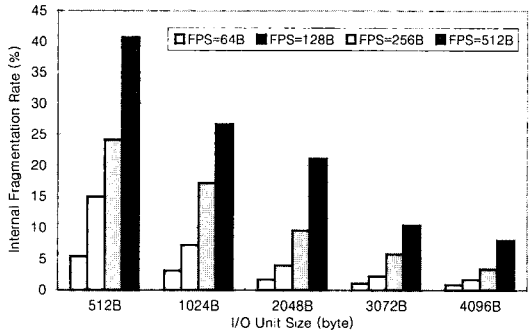

그림 10 큰 단위 압축기법(LCS)의 입출력 단위와 플래시 페이지 크기(FPS)에 따른 내부 단편화 비율

그림 10을 통해 입출력 단위가 커지거나 플래시 페이지 크기가 작아질 경우 내부 단편화를 줄이는데 효율적임을 알 수 있다. 그러나, 이 두 설계 요소를 변경하는 것은 심각한 문제를 야기할 수 있다. 예를 들어 큰 입출력 단위는 긴 입출력 시간과 인접한 페이지에 대한 불필요한 입출력 연산을 야기하며, 작은 플래시 페이지 크기는 플래시메모리 회로 설계와 플래시 변환 계층의 주소 변환 테이블을 더 복잡하게 만든다.

이러한 관점에서 볼 때 내부 그룹화 기법은 위의 두 설계 요소를 변경하지 않고도 내부 단편화를 효율적으로 줄일 수 있는 기법이다. 그림 11은 내부 그룹화 기법의 내부 단편화 비율을 쓰기 버퍼의 크기와 그룹화 정책을 변경해 가며 측정한 것이다. 그림에서 굵은 실선은 큰 단위 압축 기법의 입출력 단위가 각각 1K와 2K 바

이트인 경우의 내부 단편화 비율이다. 이 실험 결과를 통하여 내부 그룹화 기법이 상대적으로 작은 크기의 입출력 단위를 사용함에도 불구하고 큰 단위 압축 기법보다 항상 적은 양의 내부 단편화를 발생시킴을 알 수 있다. 그룹화 정책별로는 'Best-Fit' 정책이 항상 가장 좋은 성능을 보임을 알 수 있다. 한편, 내부 그룹화 기법의 쓰기 버퍼 크기는 160K 바이트로 설정한다. 그 이유는 내부 그룹화 기법의 내부 단편화 비율이 이 크기를 경계로 수렴하기 때문이다.

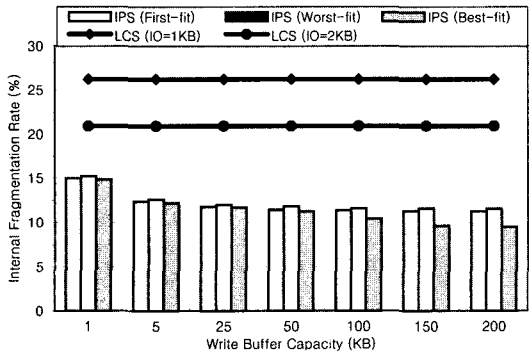

그림 11 그룹화 정책과 쓰기 버퍼의 크기에 따른 내부 그룹화 기법(IPS)의 내부단편화 비율

그림 12는 큰 단위 압축기법과 내부 그룹과 기법의 벤치마크 프로그램 별로 살펴본 내부 단편화 비율이다. 대부분의 벤치마크 프로그램에 대해 내부 그룹화 기법이 큰 단위 압축기법보다 적은 내부 단편화를 발생시킨다. 특히, 내부 그룹화 기법은 벤치마크 프로그램의 압축률이 낮을 때 보다 적은 내부 단편화를 발생시키는데, fax 벤치마크의 경우 16% 대의 낮은 압축률로 인하여 내부 단편화를 거의 발생시키지 않음을 알 수 있다.

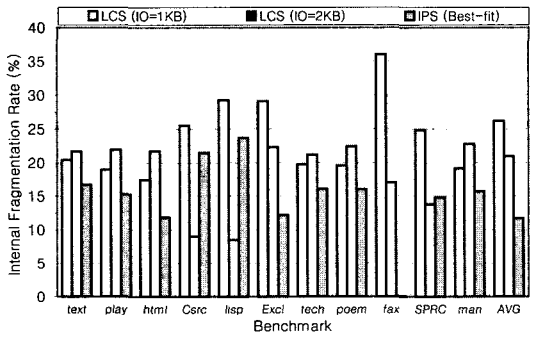

그림 12 큰 단위 압축기법(LCS)과 내부 그룹과 기법(IPS)의 벤치마크 프로그램에 따른 내부 단편화 비율

표 2는 제안하는 플래시 압축 계층의 성능을 요약한 것이다. 내부 그룹화 기법은 큰 단위 압축 기법보다 작은 입출력 단위를 사용하기 때문에 압축률 측면에서는 다소 떨어지지만, 내부 단편화를 크게 줄이게 된다. 그 결과 전체적으로 실질 압축률은 큰 단위 압축기법보다 내부 그룹화 기법이 더 우수함을 알 수 있다. 이를 바탕으로 계산한 내부 그룹화 기법의 저장 공간의 확장 비율은 40.8%로 큰 단위 압축기법의 그것보다 우수함을 알 수 있다.

일반적으로 쓰기 연산은 크게 두 가지 분류로 나뉜다. 하나는 임의 연산으로 비교적 작은 단위 이뤄지며 시공간적 지역성을 갖는 특성이 있으며, 다른 하나는 순차 연산으로 비교적 큰 단위로 이뤄진다. 임의 쓰기 연산의 경우 일반적으로 데이타의 양이 적기 때문에 쓰기 버퍼만을 사용해 연산을 수행하고 호스트 시스템에 작업 완료 신호를 보낼 수 있기 때문에 제안하는 플래시 압축 계층의 쓰기 연산 시간을 쓰기 버퍼의 연산 시간과 같이 크게 단축할 수 있다. 반면에 순차 쓰기 연산의 경우 큰 단위로 이뤄져 쓰기 버퍼의 크기를 초과해 실제로 플래시메모리에 기록해야 하는데 이때 제안하는 기법의 경우 벤치마크 데이타에 대한 실질 압축률이 71%로 플래시메모리에 쓰기 연산 횟수가 29% 감소함을 알 수 있다.

표 2 제안하는 플래시 압축계층의 성능 요약

설계 기법	압축률	내부 단편화 비율	실질 압축률	기억 공간 확장비율
큰 단위 압축기법 (IO=1KB)	57.3%	26.2%	83.5%	19.8%
큰 단위 압축기법 (IO=2KB)	56.3%	20.9%	77.2%	29.5%
내부 그룹화 기법 (Best-fit)	59.3%	11.7%	71.0%	40.8%

5. 결 론

페이지 단위의 입출력만 지원하는 NAND형 플래시메모리에서 내부 단편화는 데이타 압축 기법의 이득을 심각하게 저해시킨다. 본 논문에서는 NAND형 플래시메모리에서 내부 단편화를 최소화함으로써 압축 효과를 극대화할 수 있는 플래시 압축 계층을 제안하였다. 세부적으로 본 논문에서는 입출력 단위를 증가시키는 큰 단위 압축 기법과 작은 압축 페이지를 쓰기 버퍼를 사용해 그룹화하는 내부 그룹화 기법을 설계하였다. 실험 결과 내부 그룹화 기법에 기반한 플래시 압축 계층은 플래시메모리의 저장 공간을 40% 이상 확장하며 쓰기 대역폭을 크게 개선함을 알 수 있었다.

참 고 문 헌

[1] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and J.A. Tauber, "Storage Alternatives for Mobile Computers," *In Proceedings of the 1st USENIX Symposium on Operating System Design and Implementation*, pp. 25-37, 1994.

[2] Intel Corporation, "3 Volt Synchronous Intel StrataFlash Memory," http://www.intel.com/.

[3] Samsung Electronics, "128M x 8 Bit / 64M x 16 Bit NAND Flash Memory," http://www.samsung-electronics.com/

[4] M. Kjelso and S. Jones, "Memory Management in Flash-Memory Disks with Data Compression," *Lecture Notes in Computer Science*, Springer Verlag, Vol. 986, pp. 399-413, 1995.

[5] S. Wells and D. Clay, "Flash Solid-State Drive with 6MB/s Read/Write Channel and Data Compression," *In Proceedings of the 40th IEEE International Conference on Solid-State Circuits*, pp. 52-53, 1993.

[6] M. Wu and W. Zwaenepoel, "eNVy: A Non-Volatile, Main Memory Storage System," *In Proceedings of the 6th Symposium on Architectural Support for Programming Languages and Operating Systems*, pp. 86-97, 1994.

[7] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A Space-Efficient Flash Translation Layer for CompactFlash Systems," *IEEE Transactions on Consumer Electronics*, Vol. 48, No. 2, pp.366-375, 2002.

[8] Intel Corporation, "Understanding the flash translation layer (FTL) specification," http://developer.intel.com/.

[9] A. Kawaguchi, S. Nishioka, and H. Motoda, "A Flash Memory Based File System," *In Proceedings of the USENIX 1995 Winter Technical Conference*, pp. 155-164, 1995.

[10] MTD, "Memory Technology Device (MTD) subsystem for Linux," http://www.linux-mtd.infradead.org/.

[11] L.-P. Chang and T.-W. Kuo, "An Efficient Management Scheme for Large-Scale Flash Memory Storage Systems," *In Proceedings of the ACM Symposium on Applied Computing*, pp. 862-868, 2004.

[12] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to Flash Memory," *In Proceedings of the IEEE*, Vol. 91, No. 4, pp. 489-502, 2003.

[13] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, 2nd Ed., Addison-Wesley, 1994.

[14] D. Woodhouse, "JFFS: The Journaling Flash File System," *In Proceedings of the Ottawa Linux Symposium*, 2001. (Availble at http://sources.redhat.com/jffs2/)

[15] The Yet Another Flash Filing System Project, http://www.aleph1.co.uk/yaffs/

[16] M. Rosenblum and J.K. Ousterhout, "The Design and Implementation of a Log-Structured File System," *ACM Transactions on Computer Systems*, Vol. 10, No. 1, pp. 26-52, 1992.

[17] N. K. Edel, D. Tuteja, E. L. Miller, and S. A. Brandt, "MRAMFS: A Compressing File System for Non-Volatile RAM," In Proceedings of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 596-603, 2004.

[18] A. Silberschatz, P.B. Galvin, and G. Gagne, *Operating System Concepts*, 6th Ed., pp. 285-287, John Wiley & Sons Inc., 2003.

[19] M. Kjelso, M. Gooch, and S. Jones, "Design and Performance of a Main Memory Hardware Data Compressor," *In Proceedings the 22nd Euromicro Conference*, IEEE Computer Society Press, pp. 422-430, 1996.

[20] R. Arnold and T. Bell, "A Corpus for the Evaluation of Lossless Compression Algorithms," *In Proceedings of the 7th IEEE Data Compression Conference*, pp. 201-210, 1997.

임 근 수

2003년 연세대학교 컴퓨터과학 학사 및 전기전자공학 학사. 2005년 서울대학교 컴퓨터공학 석사. 현재 삼성 종합기술원 컴퓨팅 연구실 연구원. 관심분야는 고성능 메모리 구조, 플래시메모리 저장장치, 센서 네트워크, 운영체제 등임

반 효 경

1997년 서울대학교 계산통계학과 학사 1999년 서울대학교 전산과학과 석사 2002년 서울대학교 컴퓨터공학부 박사 현재 이화여자대학교 컴퓨터학과 조교수 관심분야는 웹 시스템, 스토리지 관리 및 캐싱 기법, 유비쿼터스 컴퓨팅 등임

고 건

1974년 서울대학교 응용물리학 학사 1979년 Univ. of Virginia 전산학 석사 1981년 Univ. of Virginia 전산학 박사 현재 서울대학교 컴퓨터공학부 교수 관심분야는 운영체제, 컴퓨터 구조, 컴퓨터 시스템 성능평가, 분산 컴퓨터시스템임