모바일 멀티미디어 기기를 위한 낸드 플래시 메모리 파일 시스템의 인덱싱 구조[1]

이현철° 김성조
중앙대학교 컴퓨터공학부
abudd@konan.cse.cau.ac.kr° sjkim@cau.ac.kr

An Indexing Structure of NAND Flash File System for Mobile Multimedia Devices

Hyunchul Lee Sungjo Kim
School of Computer Science & Engineering Chung-ang University

요 약

낸드 플래시 메모리 파일 시스템에 관한 연구는 빠른 마운트, 효율적인 가비지 컬렉션 그리고 저널링을 중심으로 이루어지고 있다. 하지만 기존의 파일 시스템은 읽기와 쓰기 성능이 낮고 파일의 개수와 크기가 증가함에 따라서 메모리 사용량이 크게 늘어난다. 멀티미디어 파일의 크기와 비트율이 계속 증가하고 있기 때문에 기존의 파일 시스템은 모바일 멀티미디어 기기에 적합하지 않다. 본 논문은 데이터를 가변적인 크기로 기록함으로써 이 문제를 해결한다. 실험을 통해 프로토타입 파일 시스템인 NAMU(NAnd flash MUltimedia file system)의 메모리 사용량은 YAFFS2에 비해 28%, JFFS2에 비해 2673% 감소 하였음을 보였다. 그리고 읽기 성능은 YAFFS2에 비해 약 28%, JFFS2에 비해 약 75% 증가했고 쓰기 성능은 YAFFS2에 비해 약 43%, JFFS2에 비해 약 120% 증가 하였음을 보였다.

1. 서 론

모바일 기기는 휴대성 때문에 크기가 작고 내구성이 뛰어난 저장 장치를 필요로 한다. 그리고 안정적으로 전원 공급을 받을 수 없기 때문에 전력 소비가 낮은 저장 장치를 요구한다. 낸드 플래시 메모리는 이러한 조건을 만족 시키기 때문에 많은 모바일 기기가 저장 장치로 낸드 플래시 메모리를 사용하고 있다[1].

멀티미디어 모바일 기기는 파일 시스템에 대해서 기존의 기기와는 다른 요구사항을 갖는다. 첫 번째로 멀티미디어 모바일 기기에 저장되는 파일은 용량이 크기 때문에 파일 시스템은 큰 파일을 지원할 수 있어야 한다. 두 번째로 멀티미디어의 품질이 계속 증가하고 있기 때문에 멀티미디어 파일 특히 동영상 파일을 원활하게 재생하고 저장하기 위해서는 더 높은 읽기와 쓰기 성능이 필요하다.

낸드 플래시 메모리는 1년에 2배씩 용량이 증가하고 읽기와 쓰기 성능이 꾸준히 향상되고 있기 때문에 멀티미디어 모바일 기기의 요구사항을 만족시킬 수 있을 것으로 예상된다[2]. 그러나 기존의 낸드 플래시 메모리 파일 시스템은 멀티미디어 파일의 특성을 고려하지 않아 멀티미디어 모바일 기기에 적합하지 않다. 특히 파일 시스템이 큰 파일을 지원하기 위해서 많은 메모리를 사용하는 문제점이 해결 되어야 하고 읽기와 쓰기 성능의 향상이 필요하다.

[1] 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 (홈네트워크연구센터) 지원사업의 연구결과로 수행되었음(IITA-2006-C1090-0603-0035)

2. 관련 연구

낸드 플래시 메모리는 페이지 단위로 입출력이 일어나는 블록 장치로써 동일한 개수의 페이지들이 묶인 블록들로 구성된다. 낸드 플래시 메모리는 고유한 2가지 특성을 가진다. 첫 번째로 낸드 플래시 메모리는 블록을 지우기 전에 덮어쓰기를 할 수 없다. 두 번째로 블록의 최대 지움 횟수가 있어서 최대 지움 횟수를 초과하여 블록을 지우면 블록은 마모될 수 있다.

낸드 플래시 메모리의 고유한 특성 때문에 하드 디스크 기반의 파일 시스템을 낸드 플래시 메모리에서 직접 사용할 수 없다. 이러한 문제를 해결하기 위해서 낸드 플래시 메모리를 하드 디스크로 가상화하는 FTL(Flash Translation Layer)[3] 또는 NFTL[4] 계층을 사용한다. 하지만 그 계층은 파일 시스템의 성능을 떨어뜨리고 낸드 플래시 메모리를 효율적으로 사용하지 못한다[5].

이 점을 보안하기 위해서 낸드 플래시 메모리를 위한 파일 시스템들이 개발되고 있다. 하드디스크 기반의 파일 시스템은 파일의 논리 주소를 물리 주소로 바꿀 수 있는 데이터 인덱싱 정보를 저장장치에 저장한다. 하지만 낸드 플래시 메모리의 특성상 특정 위치에 데이터 인덱싱 정보를 저장할 수 없기 때문에 초기 낸드 플래시 메모리 파일 시스템들은 로그 구조 파일 시스템[6]으로 개발되었다. JFFS2(Journaling Flash File System 2)[5]는 덮어쓰기가 불가능한 낸드 플래시 메모리의 특성을 고려하여 쓰기가 일어나면 파일 시스템의 변경 내용만 기록한다. JFFS2는 메타 정보의 쓰기 부하를 줄이고 공간을 효율적으로 사용하기 위해서 메타 정보와 데이터를 페이지 크기로 압축하여 낸드 플래시 메모리에 기록한다. 만약 새로운 쓰기가 이전 노드의 데이터를 포함한다면 이전 노드는 무효화된다. JFFS2는 마운트를 할 때 유효한 페이지의 위치들을 메모리에 저장하고 파일이 접근될 때마다 캐시 또는 낸드 플래시 메모리의 페이지들을 읽어서 파일의 메타 정보와 데이터 인덱싱 정보를 구성한다. JFFS2는 데이터를 특정 크기로 정렬하여 저장하지 않기 때문에 이진 균형 트리를 사용하여 파일의 논리 주소와 물리 주소를 대응시킨다. JFFS2는 마운트를 할 때에 모든 유효한 페이지를 다 읽어야 하기 때문에 마운트가 느리고 메모리 사용량이 많으며 읽기 및 쓰기에서 최대 지연시간이 길다. YAFFS2(Yet Another Flash File System 2)[7]는 파일의 메타 정보를 데이터와 다른 페이지에 쓰고 페이지의 잉여 영역을 활용하여 마운트 속도를 개선하였다. 메타 정보가 저장된 페이지와 데이터가 저장된 페이지의 잉여 영역만을 읽으면 마운트를 할 수 있기 때문이다. 그리고 YAFFS2는 데이터를 페이지 단위로 정렬하여 낸드 플래시 메모리에 기록한다. 따라서 데이터 인덱싱 정보의 크기가 JFFS2보다 작아서 모든 메타 정보 및 데이터 인덱싱 정보를 메모리 상에 저장하기 때문에 파일 시스템의 지연 시간이 JFFS2보다 짧다.

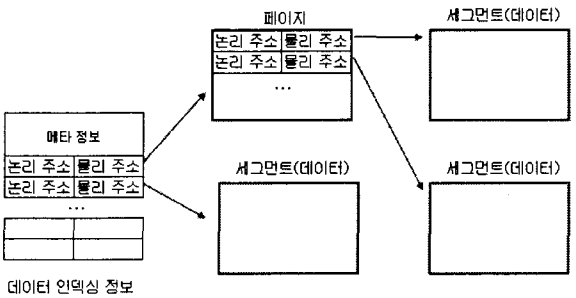
JFFS2와 YAFFS2는 파일의 개수가 증가하고 파일 크기가 커짐에 따라서 메모리의 사용량이 크게 증가하거나 파일 시스템의 성능이 떨어진다. 그리고 데이터를 읽거나 쓸 때 페이지 크기 단위 마다 페이지의 물리 주소를 얻기 위한 작업이 필요하고 낸드 플래시 메모리의 순차적 읽기 및 쓰기 연산을 이용하지 못한다. 낸드 플래시 메모리의 순차적 읽기와 쓰기 성능은 임의 읽기와 쓰기 성능보다 20~30% 높다[8].

본 논문은 이러한 문제들을 해결하기 위한 방법을 제안하고 프로토타입 파일 시스템인 NAMU를 구현하여 제안한 방법의 적절함을 입증한다. 이 문제를 해결하기 위해서 본 논문은 멀티미디어 파일이 주로 순차적으로 참조되고 수정이 잘 일어나지 않는 특성을 이용하여[9], 모바일 멀티미디어 기기에 적합한 데이터 인덱싱 구조를 제안한다.

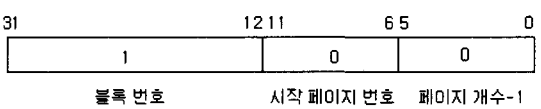
본 논문은 아래와 같이 구성된다. 3장에서 메모리 사용량을 줄이고 읽기와 쓰기 성능을 높일 수 있는 데이터 인덱싱 구조를 제안한다. 그리고 4장에서 기존 파일 시스템과 메모리 사용량 그리고 읽기와 쓰기 성능을 비교하여 제안한 방법이 적절함을 입증하고 마지막으로 5장에서 결론을 내린다.

3. 데이터 인덱싱 구조

NAMU는 쓰기 요청을 버퍼에 모았다가 파일의 데이터를 세그먼트 단위로 낸드 플래시 메모리에 기록한다. 세그먼트는 낸드 플래시 메모리에서 한 블록 보다 크지 않고 한 블록 내에서 연속적인 페이지들로 이루어진 영역이다. 이렇게 세그먼트 단위로 데이터를 기록함으로써 읽기 및 쓰기 성능을 높이고 작은 메모리와 플래시 메모리의 사용으로 큰 파일을 지원할 수 있다. 세그먼트의 크기가 크면 클수록 위의 장점이 분명해지기 때문에 파일의 처음과 마지막을 제외한 세그먼트는 가능한 블록이 되게 한다. 처음과 마지막 세그먼트를 제외한 이유는 낸드 플래시 메모리의 공간을 효율적으로 사용하기 위해서이다. NAMU는 세그먼트가 가능한 블록이 되도록 하기 위해서 빈 블록을 선택하여 데이터를 기록한다. 그리고 선택된 블록을 예약을 하여 다른 데이터가 그 블록에 써지지 않게 하고, 파일이 닫히면 사용 중인 블록을 파일 시스템으로 반환한다. 멀티미디어 파일은 수정이 거의 일어나지 않으므로 이런 방식으로 데이터를 기록하면 낸드 플래시 메모리의 블록은 대부분 비거나 가득 차 있기 때문에 파일의 처음과 마지막을 제외한 대부분의 세그먼트가 블록이 되는 것을 보장할 수 있다.



(그림 1) 데이터 인덱싱 구조



(그림 2) 세그먼트의 물리 주소

파일은 한 페이지에 저장되는 아이디, 이름, 크기, 수정 시간 등의 메타 정보를 가진다. 그리고 그 페이지는 추가로 데이터 인덱싱 정보를 세그먼트의 시작 논리 주소, 세그먼트의 물리적 주소 쌍의 배열 형태로 갖는다(그림 1). 메타 정보와 데이터 인덱싱 정보를 저장하는 페이지를 1차 인덱싱 페이지라고 한다.

세그먼트의 물리 주소는 4 바이트로 상위 20비트는 낸드 플래시 메모리에서 0부터 시작하는 블록 번호, 중간 6비트는 그 블록에서 세그먼트가 시작하는 페이지 번호, 마지막 6비트는 세그먼트를 이루는 페이지 개수 - 1을 나타낸다(그림 2). 페이지 크기가 2KB인 낸드 플래시 메모리의 한 블록은 64개의 페이지를 갖기 때문에 위의 물리 주소 표현 방법으로 최대 한 블록을 가리킬 수 있다.

만약 파일의 크기가 1차 인덱싱 페이지가 가리킬 수 있는 세그먼트들의 크기 합보다 커지면 데이터 인덱싱 정보만 저장하는 페이지를 생성하고 그 페이지가 저장하는 첫 번째 세그먼트의 시작 논리 주소, 페이지의 물리 주소 쌍을 1차 인덱싱 페이지의 인덱싱 정보에 저장한다(그림 1). 데이터 인덱싱 정보만이 저장되는 페이지를 2차 인덱싱 페이지라고 한다.

데이터를 가변적인 크기 단위로 기록하면 논리적 주소를 물리적 주소로 변환하기 위해서는 데이터 인덱싱 정보를 검색해야 하기 때문에 파일의 임의 접근 성능이 좋지 않다. 하지만 멀티 미디어 파일은 주로 순차적으로 접근되기 때문에 제안한 데이터 인덱싱 구조는 멀티 미디어 파일에 적합하다. 반면 YAFFS2는 다음 페이지의 물리 주소를 얻기 위해서 루트에서 단말 노드까지 트리를 순회해야 하고 JFFS2는 트리의 노드 크기가 32바이트로 메모리 사용량이 많다.

4. 성능 측정

NAMU를 리눅스 2.6.10에서 구현하고 YAFFS2 그리고 JFFS2와 성능을 비교했다. 성능 측정은 192MHZ ARM9과 32MB DDR RAM을 가진 보드에서 수행했다. 실험에서 사용한 낸드 플래시 메모리는 삼성 전자의 K9F2G08U0M이고 256MB 크기이다.

먼저 파일의 크기와 파일의 개수 따른 메모리 사용량을 측정했다. 그리고 읽기와 쓰기의 성능을 측정했다.

4.1 메모리 사용량

<표 1> 파일 크기에 따른 메모리 사용량(byte)

	NAMU	YAFFS2	JFFS2
1MB	2048	704	22680
2MB	2048	1296	45428
4MB	2048	2464	90704
8MB	2048	4800	181388
16MB	2048	9488	362756
32MB	6148	18848	718760
64MB	8182	37568	1450700
128MB	10240	75024	2901424

<표 1>은 파일 크기에 따른 메모리 사용량을 비교한 결과이다. 메모리 사용량으로 파일 시스템의 메타 정보와 데이터 인덱싱 정보의 크기만을 측정했다. NAMU는 1차 인덱싱 페이지를 메모리에 유지하고 파일의 크기가 커지면 2차 인덱싱 페이지들을 메모리에 유지한다. 2KB 페이지의 낸드 플래시 메모리에서 1차 인덱싱 페이지는 218개의 주소들을 저장할 수 있고 2차 인덱싱 페이지는 256개의 주소들을 저장할 수 있다. NAMU는 1차 인덱싱 페이지만으로 27.25MB를 저장할 수 있다.

JFFS2는 파일의 크기가 증가함에 따라서 메모리 사용량이 급격하게 증가하는 것을 알 수 있다. 그리고 NAMU의 메모리 사용량은 파일 크기가 4MB 이상일 때부터 YAFFS2보다 작아져서 크게 차이가 나는 것을 알 수 있다. JFFS2과 YAFFS2는 파일의 메타 정보와 데이터 인덱싱 정보로 큰 메모리 공간을 요구하기 때문에 제한적인 메모리를 갖고 멀티미디어 파일을 주로 저장하는 멀티미디어 모바일 기기에서 적합하지 않다는 것을 알 수 있다.

<표 2> 낸드 플래시 메모리에 기록한 파일

	1MB	2MB	4MB	8MB	16MB
개수	2	10	25	13	1

<표 3> 낸드 플래시 메모리가 <표 2>와 같이 파일들을 가질 경우 메모리 사용량

	NAMU	YAFFS2	JFFS2
사용량(MB)	104448	147856	2792196

실제로 디스크에 저장되는 MP3 파일의 크기 분포 확률[10]과 실험에 사용된 낸드 플래시 메모리의 256MB 크기를 고려하여 <표 2>와 같이 낸드 플래시 메모리를 파일로 채우고 메모리 사용량을 측정했다. <표 3>은 파일 시스템의 메모리 사용량을 나타낸다. NAMU는 YAFFS2에 비해서 약 29% 감소했고 JFFS2에 비해서 약 2673% 감소했다.

4.2 읽기 및 쓰기 성능

멀티미디어 파일은 주로 순차적으로 읽히거나 써지고 수정이 잘 일어나지 않기 때문에 순차적 읽기와 쓰기의 성능이 모바일 멀티미디어 기기를 위한 파일 시스템의 성능에 큰 영향을 미친다. 따라서 수정 없는 순차적 읽기와 쓰기 성능을 측정했다.

대표적인 파일 시스템 벤치마크 프로그램인

Iozone[11]을 사용하여 4KB부터 16MB까지 크기가 2배씩 늘어나는 버퍼를 사용하여 64KB에서 32MB까지 크기가 2배씩 증가하게 파일을 생성하고 생성된 파일을 읽는 방식으로 성능을 측정했다. 그리고 리눅스의 페이지 캐시 효과가 사라진 결과들에서 평균을 구하여 순차적 읽기와 쓰기 성능을 구했다

<표 4> 읽기 성능(KB/sec)

	NAMU	YAFFS2	JFFS2
KB/S	802	624	458

<표 5> 쓰기 성능(KB/sec)

	NAMU	YAFFS2	JFFS2
KB/S	371	258	162

. Iozone은 쓰기 버퍼를 쓰기가 제대로 되었는지 확인하기 위한 코드와 0으로 채우기 때문에 데이터의 압축률이 높아서 JFFS2가 높은 성능을 보였다. 실제로 응용프로그램은 그런 형태의 데이터를 사용하지 않기 때문에 그 코드를 제외한 나머지를 임의 내용으로 채웠다. <표 4>과 <표 5>는 리눅스의 페이지 캐시 효과가 없을 때의 평균적인 순차적 읽기와 쓰기 성능을 나타낸다. NAMU의 읽기 성능은 YAFFS2보다 약 28%, JFFS2보다 약 75% 향상 되었으며 쓰기 성능은 YAFFS2보다 약 43%, JFFS2보다 약 120% 향상 되었다. 결과는 세그먼트 단위로 데이터를 읽거나 쓰기 때문에 NAMU의 읽기와 쓰기 성능이 향상 되었음을 보여준다.

5. 결론 및 향후 연구

기존의 파일 시스템은 저장되는 파일의 개수와 크기가 증가하면 메모리 사용량이 크게 증가하기 때문에 작은 메모리를 가진 모바일 멀티미디어 기기에 부적합하다. 그리고 멀티미디어의 품질은 계속 향상 되고 있는 반면에 데이터를 페이지 단위로 읽기와 쓰기를 수행하기 때문에 성능이 낮은 문제점을 가진다. 본 논문은 데이터를 세그먼트 단위로 기록하여 성능을 높이고 주로 순차적으로 참조되는 멀티미디어 파일의 특성을 고려하여 데이터 인덱싱 정보를 세그먼트의 시작 논리 주소, 세그먼트의 물리 주소의 배열 형태로 저장하여 데이터 인덱싱 정보의 크기를 줄였다.

그 결과로 메모리 사용량은 YAFFS2에 비해 28%, JFFS2에 비해 2673% 감소였다. 그리고 읽기 성능은 YAFFS2에 비해 약 28%, JFFS2에 비해 약 75% 증가했고 쓰기 성능은 YAFFS2에 비해 약 43%, JFFS2에 비해 약 120% 증가했다.

하지만 제안한 데이터 인덱싱 구조는 파일의 임의 위치에 접근하기 위해서 배열을 검색해야 한다. 그리고 만약 파일의 중간에 수정이 일어나면 수정이 일어나는 부분에 대응하는 데이터 인덱싱 구조의 엔트리의 수정과 그 뒤의 엔트리들의 복사가 필요하다. 따라서 임의 접근 및 수정의 부하에 대한 보완이 필요하다.

참고 문헌

[1] 박정태, 최문선, 김성조, "NAND 플래시 메모리의 효율적 사용을 위한 접근계층의 설계 및 구현", 한국정보과학회 2004년 춘계학술대회, Vol. 31, No. 1, pp.178-180, 2004.

[2] Needham & Company, LCC, "NAND vs. Hard Disk Drives: Hype, Myth and Reality", 2005.

[3] Intel Corporation, "Understanding the Flash Translation Layer (FTL) Specification", Application Note 648, 1998.

[4] A. Ban, "Flash File system", US Patent, No. 5,404,485, 1995.

[5] David Woodhouse, "JFFS: The Journaling Flash File System", Technical Paper of RedHat inc., 2001.

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

[7] YAFFS Spec, http://www.aleph1.co.uk.

[8] CommDesign, "Flash memory 101: An Introduction to NAND Flash", 2006.

[9] T. Niranjan, T. Chiueh, and G. Schloss, "Implementation and Evaluation of a Multimedia File System", In Proceedings of ICMCS'97, pp.269-296, 1997.

[10] K.M. Evans and G.H. Kuenning, "A Study of Irregularities in File-Size Distributions", In Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2002.

[11] Iozone Filesystem Benchmark, http://www.iozone.org.