모바일 장치를 위한 데이터베이스 기반 플래시 메모리 파일 시스템

문명진° 노홍찬 박상현
연세대학교 컴퓨터과학과
{psiwind, fallsmal, sanghyun}@cs.yonsei.ac.kr

Database-based Flash Memory File System for Mobile Devices

Myungjin Moon° Hongchan Roh Sanghyun Park
Dept. of Computer Science, Yonsei University

요 약

플래시 메모리는 하드 디스크에 비하여 전력 소모가 적고 공간 대비 저장 효율이 우수하여 모바일 기기들을 비롯한 다양한 기기에 사용되고 있다. 일반적으로 플래시 메모리는 데이터 관리를 위하여 파일 시스템을 사용한다. 그러나 파일 시스템은 데이터가 중복되거나 통일성을 잃어버리기 쉽고, 데이터 고립화 현상이 나타나고, 트랜잭션의 처리가 어려우며, 무결성과 보안성이 떨어진다는 한계로 인하여 플래시 메모리에 DBMS(Database Management System)를 탑재하려는 연구들이 진행되었다. DBMS는 구조화된 데이터와 메타데이터의 저장 및 관리에 매우 효율적이므로, 이러한 데이터를 주로 다루는 모바일 장치에서 특히 유용하다. 그러나 파일 시스템과 DBMS는 서로 다른 버퍼링 및 캐싱 정책, 저장 공간 관리 전략을 사용하는 경우가 많기에, 파일 시스템 상에서 DBMS를 구동하는 것만으로는 데이터베이스의 전반적인 기능을 효율적으로 활용하기 어려워진다. 또한 파일 시스템과 DBMS가 중복적으로 포함하고 있는 기능들이 존재하므로 낭비되는 자원 역시 늘어난다. 이러한 문제를 해결하기 위하여 본 논문에서는 기존 플래시 메모리의 파일 시스템과 DBMS를 통합하는 새로운 구조를 제안한다.

1. 서론

플래시 메모리는 전원 공급에 관계없이 데이터의 유지가 가능한 비활성 저장 매체이다. 플래시 메모리는 별도의 지연 시간 없이 일정한 시간 내에 데이터 접근이 가능하며 외부의 충격에 강하고, 기계적 동작에 필요한 전력 소모가 없어서 적은 전력으로 동작이 가능하며 크기에 비하여 저장 용량이 크다는 장점이 있다. 또한 최근에는 용량이 빠르게 증가하고 있고, 이에 반해 가격은 하락하는 추세를 보이고 있다. 이러한 이유로 플래시 메모리를 이용하는 장치들이 급속히 증가하고 있다. 일반적으로 플래시 메모리는 데이터의 관리를 위하여 파일 시스템을 사용하고 있다. 그러나 파일 시스템은 정보 관리의 편의성 데이터 접근성, 제약 조건 설정의 용이성 보안성 등에 있어서 DBMS(Database Management System)에 비해 부족함을 보인다. 이러한 이유로 플래시 메모리에 DBMS를 탑재하는 연구들이 진행되기 시작하였다.

DBMS를 이용할 경우 파일들 간의 복잡한 관계를 다룰 수 있고, 검색 및 정렬을 빠르게 수행할 수 있다. 또한 플래시 메모리가 주로 사용되는 모바일 장치에서는 파일의 정보에 대한 다양한 메타데이터를 효율적으로 처리할 필요성이 있으며, 데이터를 체계적으로 구조화하여 관리해야 한다. 이러한 측면에서 DBMS는 매우 효율적이다.

† .본 연구는 한국학술진흥재단의 2008년도 기초 연구지원 기초과학사업(D00849)의 지원을 받아 수행되었습니다.

기존에 진행된 연구는 파일 시스템을DBMS로 대체하는 것이 아닌, 파일 시스템 상에서 DBMS를 동작시키는 것이라는 한계를 가지고 있다. 이러한 시스템에서는 DBMS의 연산이 파일 시스템을 거쳐서 수행된다. 파일 시스템은 기본적으로 버퍼링 전략에서 DBMS와 차이를 보이기에 두 시스템을 혼용하게 되면 성능 저하를 가져오게 된다. 또한 순수한 데이터베이스의 사용이 이루어지지 않으므로 데이터베이스의 ACID(Atomicity, Consistency, Isolation, Durability) 속성을 제대로 만족할 수 없게 한다. 따라서 플래시 메모리 상에서 DBMS를 보다 효율적으로 사용하기 위해서는 파일 시스템과 DBMS를 밀접하게 통합할 필요가 있다. 이러한 통합된 시스템은 구조화된 데이터의 저장 및 메타데이터의 관리에 용이하므로, 모바일 장치에서의 활용도가 특히 높다.

본 논문에서는 모바일 장치를 주 대상으로 하는 플래시 메모리의 데이터베이스 기반 파일 시스템을 제안한다. 2장에서는 데이터베이스를 파일 시스템에 적용한 기존의 연구들에 대하여 살펴본다. 3장에서는 제안하는 시스템의 구조와 이를 위해 고려해야 할 사항들을 제시한다. 마지막으로 4장에서는 본 논문의 결론과 향후 연구 방향을 기술한다.

2. 관련 연구

DBMS를 파일 시스템에 부분적으로 적용하기 시작한 연구로는 LISFS(Logic File System)[1]과 LiFS(Linking File System)[2]이 있다. LISFS는 Semantic file systems[3]의 연구 결과를 보다 정형화한 모델로 기존의 파일 시스템을 최

대한 유지하면서 데이터베이스를 적용하는 모델을 제시하였다. LiFS는 메타데이터로 부터 얻은 정보를 통해 비슷한 메타데이터를 공유하는 파일들 간에 연결 관계를 형성함으로써 기존의 단순한 계층적 패러다임보다 유기적인 파일 시스템을 구성하였다. 이들 두 파일 시스템은 기능의 차이는 있으나, 추출한 메타데이터 정보를 데이터베이스에 저장하고 사용자 레벨 파일 시스템 인터페이스를 이용하여 파일 시스템을 구현했다는 측면에서 유사하다.

데이터베이스 기반의 파일 시스템으로는[4]가 있다. 그러나 [4]는 NFS(Network File System)의 사용자 레벨 파일 시스템 인터페이스를 이용함으로써 데이터 캐싱 과정에서 ACID 속성을 유지하지 못한다는 문제점이 있다. 파일 시스템에 ACID속성들을 지원하기 위해 데이터베이스에 기반하여 파일 시스템을 설계한 최초의 시도로는IvFS(Inversion File System)[4]가 있다. IvFS는 ACID 속성을 지원하는 파일 시스템이나, POSIX(Portable Operating System Interface) 파일 시스템 인터페이스를 거치지 않고Postgres 데이터베이스에 대한 질의 언어로만 파일 시스템에 접근이 이루어지는 특징을 가진다. 반면, Amino[6]는 POSIX 인터페이스와 ACID 속성을 동시에 지원한다. 기존의 데이터베이스 기반 파일 시스템과 달리 사용자 레벨 파일 시스템 인터페이스를 사용하는 대신 리눅스 커널 내에 파일 입출력 관련 시스템 콜에 대한 호출을 추적하는 방법을 사용하고 시스템 콜을 수행하는 동안ACID 속성을 유지하기 위해 복구 가능한 가상 메모리 및 일관성의 유지를 위한 다수의 규칙을 사용한다. 이와 유사한 파일 시스템으로 오라클이 2008년에 발표한 Oracle SecureFiles 파일 시스템[7]이 있다. Oracle SecureFiles는 성능이 검증된 Oracle database server를 활용한 데이터베이스를 활용한 파일 시스템으로 기존 연구가 목표로 하였던ACID 속성의 지원과 POSIX 인터페이스를 지원 할 뿐 아니라 최근에 업계 및 정부에서 주목하고 있는 대용량 분산 파일 시스템까지 지원한다.

그러나 현재까지 모바일 장치를 주 대상으로 하여 데이터베이스와 파일 시스템을 통합하려는 시도는 없었다. 본 연구는 모바일 장치 상에서 파일 시스템과 데이터베이스 시스템을 통합하여 여러 가지 성능 상의 이점을 제공하고자 한다. 이에 대해서는 3장에서 자세히 기술한다.

3. 시스템 제안

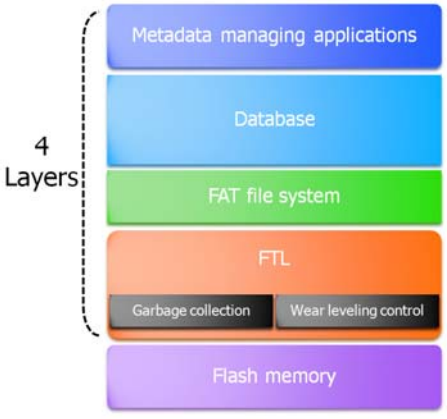

그림 1. 블록 매핑 파일 시스템

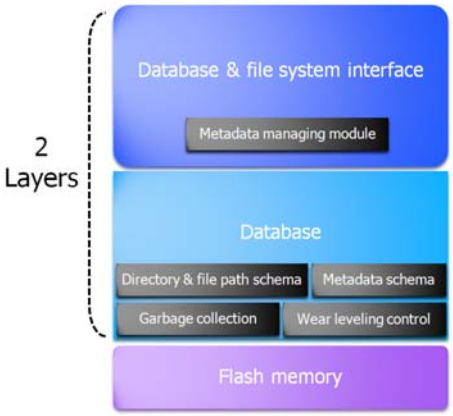

그림 2. 데이터베이스 기반 파일 시스템

일반적으로 파일 시스템은 그림1과 같은 구조를 가진다. 이는 기존의 하드 디스크에서 사용되었던 운영체제 및 응용 프로그램에 대한 수정을 최소화하기 위한 구조로 블록 매핑 방식으로 불린다.[8] 플래시 메모리 상에 FAT와 같은 파일 시스템이 존재하며 이러한 파일 시스템의 요청을 재해석하여 플래시 메모리에 전달하는 중간 계층인 FTL(Flash Translation Layer)이 탑재된다. 모바일 장치에서 사용되는 대부분의 데이터베이스 시스템들은 이러한 파일 시스템 위에서 동작한다.

그림 2는 본 논문에서 제안하는 데이터베이스 기반 파일 시스템을 보여준다. 이는 DBMS가 파일 시스템과 하나로 통합된 형태로, 기존의 FTL이나 플래시 메모리의 파일 시스템이 담당하던 가비지 컬렉션(garbage collection)이나 웨어 레벨링(wear leveling)까지 DBMS가 담당하게 된다.

그러나 이러한 통합된 시스템을 구성하기 위해서는 추가적으로 고려해야 할 요소들이 있다. 일반적으로 파일 시스템은 플래시 메모리의 커널 레벨에 존재한다. 반면에 DBMS는 사용자 레벨의 환경에서 작동하게 된다. 이러한 문제를 해결하기 위해서는 사용자 레벨에서 파일 시스템의 인터페이스를 사용할 필요성이 있다. 이를 위해 커널 인터페이스를 사용할 수 있도록 교량 역할을 해주는 사용자 레벨의 파일 시스템 인터페이스를 제공할 필요가 있다. 이러한 사용자 레벨 파일 시스템 인터페이스로는 대표적으로 FUSE(Filesystem in Userspace)[9]가 있다.

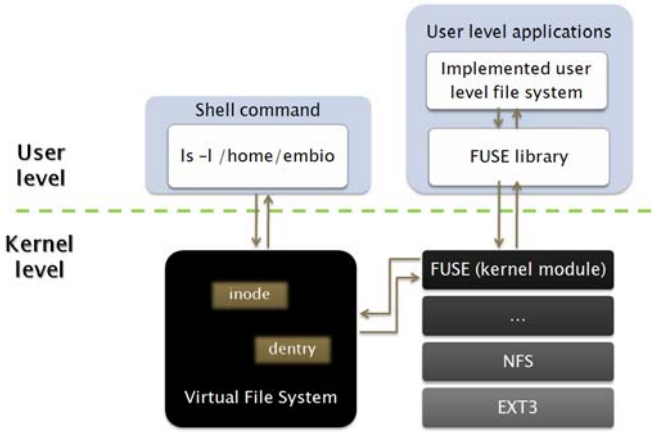

그림 3. FUSE의 동작 구조

그림 3에서 볼 수 있듯이, FUSE는 사용자 레벨에 존재하면서 사용자가 커널 레벨의 파일 시스템에 접근할 수 있도록 인터페이스를 제공한다. 다시 말해, 사용자 영역에 있는 DBMS가 직접적으로 파일 시스템을 제어할 수 있도록 연결해주는 역할을 수행한다. FUSE는 다양한 운영 체제와 프로그래밍 언어를 지원하며, 구현 역시 매우 간단하여 사용자 레벨 파일 시스템 인터페이스로서 상당히 널리 사용되고 있다. 그러나 커널 레벨에 존재하는 장치들에 대한 접근이 항상 FUSE를 통하여 이루어지므로 약간의 오버헤드는 발생하게 된다.

통합된 시스템을 구성하기 위해 고려해야 할 또 하나의 사항은 기존의 파일 시스템 구조를DBMS의 구조에 맞게 변경하는 것이다. 일반적으로 파일 시스템은 계층형 구조인 트리 형태를 사용하는 반면, DBMS는 관계형 구조를 사용한다. 따라서 파일 시스템을 통합하는 DBMS를 사용하기 위해서는 그림 4와 같이 계층형 구조를 관계형 구조로 변경할 필요가 있다.

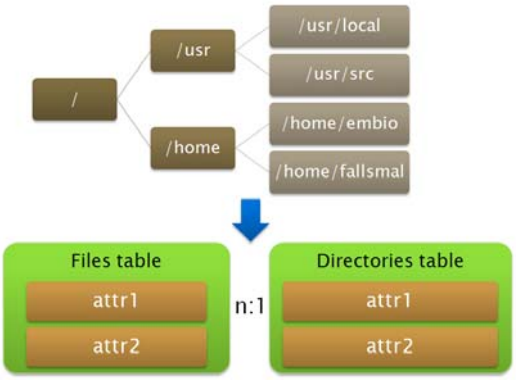

그림 4. 계층형 구조의 관계형 구조로의 변화

표 1은 본 논문에서 제안하는 관계형 파일 테이블이다. 폴더명이나 파일명은 부모 폴더가 동일하지 않으면 같은 이름을 가질 수 있으므로, 이름뿐 아니라 ID를 통하여 이들을 구분한다. 그리고 상위 경로에 대한 추적이 필요하므로 부모의 ID에 대한 정보를 보관한다. 추가적으로 파일인지 폴더인지를 구분할 수 있는 속성을 따로 설정하여 검색을 한 번에 수행할 수 있도록 하였다.

표 1. 제안하는 관계형 파일 구조 테이블의 예 (/home/embio/file1, /usr/embio/file1)

ID	Name	ParentID	isFile
1	/	0	0
2	home	1	0
3	embio	2	0
4	usr	1	0
5	embio	4	0
6	file1	3	1
7	file1	5	1

위에서 제안한 것처럼 DBMS를 기존의 파일 시스템과 통합할 경우 얻을 수 있는 이점은 다양하다. 기본적으로 DBMS가 FTL이나 YAFFS, JFFS와 같은 파일 시스템을 거치지 않고 저장 장치(raw device)를 직접 제어할 수 있으므로 접근 오버헤드가 줄어들고 보안성이 향상된다. 또한, DBMS가 직접 플래시 메모리를 저장장치로 접근하고 파일 시스템의 기능을 대신하게 되면 DBMS가 파일 시스템 상위에서 동작할 때 발생하는 두 시스템의 로깅, 버퍼링, 캐싱 알고리즘 및 저장 방법에 대한 충돌 문제가 자연히 해결될 수 있다. 이로 인해 전반적인 I/O 처리 성능의 향상을 기대할 수 있다. 플래시 메모리는 블록 단위의 삭제만을 지원하며, 쓰기 횟수에 제한이 있기 때문에 저장 공간을 효율적으로 관리하는 것이 매우 중요하다. 기존의 파일 시스템들에서는 대부분 논리적 블록에 물리적 주소를 매핑하는 FTL 매핑 테이블을 관리하는데, 이는 플래시 메모리가 마운트될 때마다 FTL 매핑 테이블이 매번 새로 구성되어야 한다는 비용을 발생시킨다. 그러나 제안하는 파일 시스템에서는 DBMS의 테이블을 사용하여 이러한 매핑 테이블을 효율적으로 저장할 수 있기에, 기존 파일 시스템들과는 달리 저장되어 있는 테이블을 단순히 읽어오는 것이 가능해져 소요되는 시간 및 공간을 줄일 수 있다.

4. 결론 및 향후 연구 계획

기존의 플래시 메모리 상의 파일 시스템과DBMS를 통합한 구조를 제시하였으며, 사용자 영역에 존재하는 DBMS가 커널 영역에 존재하는 파일 시스템에 접근할 수 있는 인터페이스를 제공하기 위하여 FUSE를 사용한 구조를 제시하였다. 추가적으로 계층형 구조로 이루어진 기존의 파일 구조를 관계형 구조로 변환하는 방법을 제안하였다. 이러한 시스템을 통하여 데이터베이스의 ACID 속성을 보장할 수 있고, 저장 장치의 직접적인 제어가 가능해져 접근에 대한 오버헤드를 줄일 수 있다. 또한 파일 시스템과 DBMS의 정책들의 차이에서 오는 충돌을 피할 수 있고, DBMS의 테이블을 통하여 저장 공간을 효율적으로 관리할 수 있게 되어 시간적·공간적인 성능 향상을 가져오게 된다. 이러한 이점들은 특히 구조화된 데이터의 저장과 메타데이터의 관리가 중요한 모바일 장치에서 잘 드러나게 된다.

차후에는 FUSE에 대한 분석을 추가적으로 진행하고, FUSE와 함께 공개 데이터베이스인 SQLite를 이용하여 이러한 구조를 구현할 계획이다.

5. 참고 문헌

[1] Y. Padioleau et al., "A Logic File System", In Proceedings of the 2003 USENIX Annual Technical Conference, pp. 99-112, 2003.
[2] A. Alexander et al., "Richer File System Metadata Using Links and Attributes", In proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies, pp. 49-60, 2005.
[3] D. K. Gifford et al., "Semantic file systems", In proceedings of the 13th ACM Symposium on Operating Systems Principles, pp. 16-25, 1991.
[4] N. Murphy et al., "The Design and Implementation of the Database File System", www.eecs.harvard.edu/~vernal/learn/cs261r/index.shtml, 2002.
[5] M. A. Olson, "The Design and Implementation of the Inversion File System", In Proceedings of the USENIX Winter 1993 Technical Conference, pp. 205-218, 1993.
[6] M. Niloy et al., "Oracle SecureFiles System", In Proceedings of the VLDB Endowment, Vol. 1 No. 2, pp. 1301-1312, 2008.
[7] C. P. Wright et al, "Extending ACID Semantics to the File System", ACM Trans on Storage, Vol. 3, No. 2, pp. 1-42, 2007.
[8] E. Gal et al, "Algorithms and Data Structures for Flash Memories", ACM Computing Surveys, Vol. 37, No. 2, pp. 138-163, 2005.
[9] M. Szeredi, "File System in User Space", http://fuse.sourceforge.net, 2006.