차세대 비휘발성 메모리를 이용한 플래시 메모리 파일시스템의 개선

Kim Hong, Jin Seong Jun, Sung Min, Hyeon Joon Kim, Joon Hoon Shin
Department of Electrical Engineering and Computer Science, Seoul National University

Enhancing Flash Memory File System Using Non-Volatile RAM

Kihong Kim*, Seongjun Ahn**, Jongmoo Choi†, Donghee Lee‡, Sam H. Noh§
Division of Information and Computer Science, Dankook University
Department of Electrical Engineering and Computer Science, Seoul National University
Department of Computer Science, University of Seoul

요 약
본 논문에서는 차세대 비휘발성 메모리를 이용하여 기존 플래시 메모리 파일시스템의 마운트 시간을 단축시키고 메인 메모리 사용량과 전력소모량을 감소시킬 수 있는 기법을 제시한다. 제안된 기법은 소량의 비휘발성 메모리를 사용하여 볼록의 상대 정보와 파일 메타데이터 및 데이터의 주소를 저장한다. 제안된 기법은 내장형 보도 드라이버로 구현되었으며, 실행 및 분석 결과는 마운트 시간을 크게 단축시키고 메인 메모리 사용량을 현저히 감소시켰음을 증명한다. 본 연구는 소량의 차세대 비휘발성 메모리를 내장형 시스템에서 어떻게 활용할 수 있는지에 대한 실험적인 방안을 제시하였으며, 그 효과를 실제로 구현하고 실험을 통해 정량적으로 검증하였습니다.

1. 서론

최근 내장형 시스템이 다양한 분야에서 널리 사용되고 있으며 특히 휴대형, MP3 재생기, PDA 등 모바일 장치의 수요가 증가하고 있다. 이러한 내장형 시스템은 다음과 같은 요구 조건을 가지고 있다. 첫째, 한정된 자원으로 동작이 가능해야 한다. 최근 디지털 통합이 이루어지면서 디지털 기기가 다양한 기능을 수행할 경우가 많아져서 따라 한정된 자원을 보다 효율적으로 사용할 수 있는 방법이 요구된다. 둘째, 제한된 전력을 공급하는 배터리 사용으로 전력 소모량이 적어야 한다. 셋째, 장치의 기능시작이 빨라야 한다. 일반적으로 내장형 시스템의 저장장치는 플래시 메모리가 많이 사용되고 있다. 플래시 메모리의 가용성과 크기로 인해 내구성이 뛰어난 특성을 가지고 있다. 그러나 적외선을 광속 할 수 있는 장치로 제작되는 것을 제약으로 얻어져서 이를 극복하기 위해 플래시 메모리에 특화된 파일시스템이 사용되고 있다. 그러나 현재의 플래시 메모리 파일시스템은 앞서 언급한 내장형 시스템의 요구조건을 만족시키지 못하고 있다. 가장 큰 문제점은 마운트 시간에 소요되는 시간이 길기 때문에 장치의 기동시각이 길어진다는 것이다.

본 논문에서는 차세대 비휘발성 메모리를 이용하여 기존 플래시 메모리 파일시스템의 마운트 시간을 단축할 수 있는 기법을 제시한다. 차세대 비휘발성 메모리(MEM, NVRAM)에 FeRAM, PRAM, MRAM 등이 있으며, 이러한 메모리는 ORAM과 유사한 성능을 가지고 있고 전력이 공급되지 않아도 저장된 데이터를 유지하는 특성을 가지고 있으며 플래시 메모리와는 다르게 볼록의 관리 기법을 필요로 하지 않다[1]. 경과에는 높은 가격으로 인해 보편적으로 사용되지 않았으나, 최근에는 반도체 기술의 발전으로 가격이 낮아지게 되면서 사용이 확대되고 있다. 본 연구의 접근 방법은 소량의 NVRAM에 플래시 메모리 파일시스템의 관리에 필수적인 정보를 저장함으로써 많은 가격으로 자원 사용의 효율성을 증대하는 것이다. 제안된 기법은 구체적으로 저장장치의 도움으로, 플래시 메모리 파일시스템의 마운트 시간을 크게 단축시킨다. 또한, 본 논문에서는 저장장치의 사용을 통해 전력 소모량을 감소시킬 수 있다.

본 연구에서는 리눅스와 윈도우에서 사용되는 플래시 메모리 파일시스템의 YAFFS[2]의 코드를 수정하여 제안한 기법을 구현하였다. 구현된 파일시스템을 사용하여 수행한 실험 결과는 마운트 시간이 크게 감소하였으며, 파일의 개수 및 용량에 관계없이 일정한 결과를 보였다. 또한 맥다운 데비아이크를 수행한 후 사용된 메인 메모리를 촉발한 것의 결과 기존의 YAFFS에 비해

본 연구는 한국과학재단특정기초연구(R01-2004-000-10188) 지원으로 수행되었음.
2. 기존 YAFFS 분석

NAND 유형의 플래시 메모리는 여러 개의 블록으로 구성되어 있으며, 블록은 다시 여러 개의 페이지로 구성되어 있다. 각 페이지는 데이터 영역과 소프트웨어 영역으로 나뉘어져 있다. YAFFS는 각 페이지의 데이터 영역에 파일의 메타데이터와 파일 데이터의 저장을 한다. 파일 데이터는 파일의 정적 정보를 포함하고 있어서 이를 통해 파일의 구조를 표시하고 있다. 각 페이지의 소프트웨어 영역에는 메인 데이터 영역에 저장된 정보의 유형, 데이터가 속한 파일의 식별자, 데이터가 포함한 파일의 부호에 대한 정보가 저장되어 있다. 파일 데이터에 저장된 정보는 파일의 크기, 위치, 속성 등에 대한 정보를 포함한다.

YAFFS는 모든 개체 간의 연관 관계를 역상(inverse mapping)의 형태로 표현하고 있다. 메인 데이터의 구조를 표현하면서 파일에서 파일의 데이터와 파일의 데이터를 표현한다. 또한 파일의 데이터가 저장된 파일의 데이터를 표현하는 데 사용된다. 역상 메커니즘은 보통의 메커니즘과 다른 구조를 가지고 있다. 그림 1은 레코더에서 사용되는 디스크 파일시스템의 방식으로 파일의 메타데이터와 데이터 사이의 관계를 표현하는 방법을 YAFFS의 비교적 보다 간단한 형태로 나타낸 것이다.

YAFFS에서는 역상 메커니즘의 형태를 사용하기 때문에 메니저에서 메모리가 소프트웨어 영역을 갖고 있는 데 사용된다. 역상 메커니즘은 메니저에서 메모리가 소프트웨어 영역에 저장된 정보를 사용하는 데 사용된다. 메니저에서 저장된 정보는 메니저에서 메모리가 소프트웨어 영역에 저장된 정보를 사용하는 데 사용된다.

3. 수정된 YAFFS의 설계 및 구현

기존 YAFFS에서는 역상 메커니즘을 사용한 것처럼 플래시 메모리 상의 최소의 성능을 유지하기 위해 이를 개선하기 위한 오프로드를 최소화하려는 목적이다. 그러나 NVRAM이 약간 느리기 때문에 메니저의 소프트웨어 구조가 변경될 수 있기 때문에 메니저의 설계가 변경되어야 한다. 이러한 변경은 메니저의 설계를 변경하기 위한 소프트웨어 구조의 변경과 관련이 있다. 메니저의 동작은 파일 데이터를 사용하여 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다.

기존 YAFFS에서는 NVRAM에 저장하는 정보를 사용하기 때문에 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다. 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다.

1) 플래시 메모리의 각 블록의 상태 정보: 기존 YAFFS에서는 주로 메니저의 메모리 영역을 사용하여 파일 데이터와 메니저의 메모리 영역에 저장된 정보를 기억한다. 그러나 수정된 YAFFS에서는 메니저의 상태 정보를 메니저의 메모리 영역에 저장할 수 있다. 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다.

2) 메니저 동작: 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다. 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다.

기존 YAFFS에서는 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다. 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다.

기존 YAFFS에서는 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다. 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다.

기존 YAFFS에서는 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다. 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다.

기존 YAFFS에서는 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다. 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다.

기존 YAFFS에서는 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다. 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다.

기준 YAFFS에서는 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다. 메니저의 동작을 변경하기 위해서 메니저의 동작을 변경하는 데 사용된다.
struct header_info {
    int flash_memory_address;
    struct header_info parent;
    struct header_info sibling;
    struct Node *Tnode; }

그림 3. 헤더 정보 저장하기 위한 자료구조

4.1 NVRAM 용량 분석

NVRAM에 저장되는 정보는 별도 헤더, 헤더 정보, Tnode이다. 한 블록의 크기로 표현하기 위해서는 8 바이트의 메모리가 필요하다.
따라서 4096개의 블록으로 이루어진 64MB의 플래시 메모리에 사용할 수 있는 별도 헤더의 개수는 32KB이다. 그러므로 헤더 정보 저장을 위한 자료구조는 20바이트로 이루어져야 한다. 64MB의 플래시 메모리 사용할 경우 생성할 수 있는 최대 파일의 개수는 63만 4천개이기 때문에 메모리에 저장되는 경우를 가장, 64MB의 파일의 헤더 정보를 저장하기 위해 필요한 NVRAM의 용량은 약 1.28MB이다. 또한 64MB의 파일 메타데이터를 위한 NVRAM의 용량은 약 300KB이다. 따라서 4096개의 헤더를 메모리에 저장할 때 필요한 NVRAM의 용량은 2MB이하로 플래시 메모리 용량의 약 3%에 해당한다.

4.2 마운트 시 소요 시간

그림 5. 마운트 시간 비교

마운트 5는 파일 시스템에 저장된 파일의 개수에 따른 마운트 시간의 변화를 보이고 있다. 각 파일의 크기는 512KB이다. 결과에서 확인할 수 있듯이 기존 YAFFS의 마운트 시간은 파일의 개수 및 파일의 크기와 비례하여 증가한다. 이는 메인 메모리에 오브젝트와 Tnode를 생성할 때 소요되는 시간을 반영하고 있다. 반면 수정된 YAFFS의 마운트 시간은 적합하다. 파일이 저장되지 않은 경우에도 기존 YAFFS는 별도의 시스템을 구성하기 위해 많은 별록의 첫번째 페이지의 사용을 검사하여 하므로 수정된 YAFFS보다 많은 시간이 소요된다. 이 경우 기존 YAFFS의 마운트 시간은 377ms, 수정된 YAFFS의 마운트 시간은 220ms로, 수정된 YAFFS의 마운트 시간이 약 42% 감소하였다.

4.3 메인 메모리 사용량

그림 4. 수정된 YAFFS의 NVRAM 자료구조

메인 메모리 사용량

또한 6은 512KB의 크기를 가지는 64개의 파일을 생성한 상태에서 오른쪽 파일의 개수가 증가할 때 각 파일시스템이 사용하는 메인 메모리 사용량을 보이고 있다. 기존 YAFFS는 마운트 시에 모든 파일의 오브젝트와 Tnode를 메인 메모리에 생성하므로 오른쪽 파일의 개수가 증가함에 따라 메인 메모리 사용량도 증가한다. 그러나 두 결과의 차이가 큰 것은 수정된 YAFFS에서 Tnode 정보가 메인 메모리에 저장되지 않기 때문이다.

4.4 앤드류 벤치마크 (Andrew benchmark) 수행 시간

수정된 YAFFS는 파일 연산 시 기존 YAFFS에 비해 빠르게 작동한다. 파일 생성 및 삭제 시에는 NVRAM에 저장된 헤더 정보를 검색하여 미리 파일의 위치를 찾고 플래시 메모리에서 헤더를 읽어 오브젝트를 생성한다. 따라서 파일 생성에 소요되는 시간이 기존 YAFFS에 비해 줄어든다. 이에 따라 오브젝트 생성 빠르게 관리된 엔드리 브랜치 마크의 1단계와 2단계를 수행하였다. 1단계에서는 20개의 디렉터리를 생성하는 작업을 수행하며, 2단계에서는 70개의 파일을 복사하는 작업을 수행한 결과, 메모리의 사용은 355KB이다. 표 1은 측정한 수행시간을 보이고 있다. 수정된 YAFFS의 경우 기존 YAFFS에 비해 빠르고 수행시간이 약 1.2% 줄어들었다. 그러나 메인 메모리 사용량은 기존 YAFFS가 12.3KB, 수정된 YAFFS가 2.1KB로 약 83% 감소하였다.

5. 결론

본 논문에서는 NVRAM을 사용하여 플래시 메모리 파일시스템의 마운트 시점을 단축하고 메인 메모리 사용량을 감소시킬 수 있는 기법을 제시하였다. 향후 제안된 기법이 가능하다고 판단할 수 있는 전체 사용량을 정량적으로 분석하고, NVRAM을 개선하여 사용하는 기법을 추가하여 사용할 수 있는 기법을 연구할 계획이다.

참고문헌

[1] 유영균, 류성욱, 윤성진, 유지Cliente 오비트 항목 메모리 기술, 전자통신정보학, 제 20권, 1호, 130~138, 2005