제28회 한국정보처리학회 추계학술발표대회 논문집 제14권 제2호 (2007. 11)

신속한 Mount 를 위한 YAFFS 에서의 블록 할당 방식

김석현*, 이재흥*, 오진하*, 민홍*, 구본철*, 이상호*, 조유근*
*서울대학교 컴퓨터공학부
e-mail : shkim@os.snu.ac.kr

A Block Allocation Scheme for Rapid Mount in YAFFS

Seok-Hyun Kim*, Jae-Heung Lee*, Jin-Ha Oh*, Hong Min*,
Bon-Cheol Gu*, Sang-Ho Yi*, Yoo-Kun Cho*
*Dept. of Computer Science & Engineering, Seoul National University

요 약

YAFFS 는 리눅스에서 사용되는 NAND 플래시 전용 파일 시스템이다. YAFFS 는 파일 시스템을 mount 할 때 전체 플래시를 scan 하여 메모리 상에 파일 시스템의 디렉토리 트리 구조를 만든다. 이 작업은 전체 메모리를 scan 하기 때문에 이 작업은 많은 시간을 필요로 한다. 이 논문에서는 YAFFS 의 블록 할당 방식을 개선하여 mount 시간을 크게 줄일 수 있는 방식을 제안한다.

1. 서론

PMP, MP3 Player, PDA 등의 기기들이 많이 사용되면서 플래시 메모리의 사용은 나날이 증가하고 있다. 플래시에 파일 시스템을 올리기 위해 FTL(Flash Translation Layer)을 이용하여 기존의 파일 시스템을 그대로 사용하는 방법과 JFFS, YAFFS 같은 플래시 전용 파일 시스템을 사용하는 방법이 있다.

YAFFS 는 현재 많이 사용되고 있는 플래시 파일 시스템으로서 리눅스를 사용하는 임베디드 기기에서 플래시 메모리를 사용할 경우 자주 이용된다. YAFFS 는 처음 파일 시스템을 mount 할 때 전체 디스크를 scan 하기 때문에 다소 긴 시간이 필요한 단점이 있다.

본 논문은 YAFFS(Yet Another Flash File System)가 플래시 메모리에 데이터 및 메타데이터를 기록되는 방법을 개선함으로써 플래시 파일 시스템의 mount 시간을 줄이는 방식에 대해 제안한다.

2. YAFFS 의 mount 과정

YAFFS 파일 시스템은 메모리 상에 현재 플래시 메모리에 저장되어 있는 모든 파일 및 디렉토리의 계층 구조를 유지한다. 이를 위해 yaffs_Object 구조체가 사용된다. yaffs_Object 는 파일, 디렉토리, 링크 등을 표현하며 yaffs_Object 의 트리 구조를 통해 전체 파일시스템의 구조를 메모리 상에 구축한다.

플래시 메모리는 블록과 페이지로 구성되어 있다. 플래시 내에 여러 블록들이 존재하고 각 블록은 같은 수의 페이지로 나누어진다. 특이한 점은 각 페이지에 여분의 영역이 존재한다는 것이다. 512 byte 크기의 페이지의 경우 각 페이지마다 16 byte 의 spare 영역이 존재한다. 이 영역은 주로 ECC(Error Corection Code)를 저장하는데 이용된다. YAFFS 의 경우 이 영역을 ECC 뿐만 아니라 파일 시스템의 정보를 담는 용도로써 적극적으로 사용한다. 16 byte 의 spare 영역 중 8 byte 는 ECC 에 사용하고 8 byte 는 청크 아이디, 오브젝트 아이디 등의 파일 시스템 고유 정보를 저장한다. YAFFS 에서 청크(chunk)는 페이지와 같은 의미로 사용된다.

YAFFS 는 mount 할 때 전체 페이지를 scan 하면서 각 페이지의 spare 영역도 읽어 들인다. spare 영역에 저장되어 있는 청크 아이디가 0 이면 해당 페이지는 메타데이터가 저장되어 있음을 뜻한다. 1 보다 크거나 같은 청크 아이디는 데이터가 저장되어 있는 청크를 의미한다. YAFFS 는 mount 를 하면서 메타데이터가 저장된 청크를 읽을 경우 이 정보를 바탕으로 yaffs_Object 구조체를 메모리 상에 만든다.

데이터가 저장된 청크를 읽어 들인 경우 이 청크가 소속된 object 의 아이디를 spare 영역에서 읽어 들임으로써 이 청크가 어느 object 에 소속되었는지 알 수 있다. 이 경우 메모리 상에 구성된 yaffs_Object 의 트리에서 읽어들인 object 아이디를 가진 object 를 찾아내어 이 object 에 해당 데이터 청크의 아이디를 추가한다.

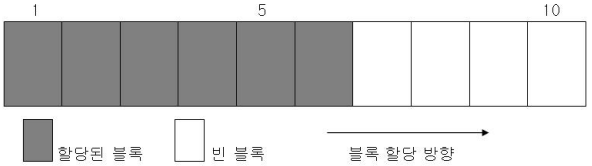
이렇게 데이터를 가지고 있는 object 는 파일을 의미한다. 파일을 의미하는 yaffs_Object 의 구조체에는 해당 파일에 속한 데이터들의 청크 아이디가 트리 형태로 기록된다. YAFFS 는 플래시에서 파일을 읽을 때에 이 구조체를 이용하여 데이터가 저장된 페이지들을 찾게 된다.

이상과 같은 과정을 통해 YAFFS 는 mount 시점에 NAND 플래시 전체를 검색하여 파일 시스템의 tree 구조를 메모리 상에 구축하고 이를 VFS(Virtual File System)를 통하여 커널에 등록하게 된다. [1]

3. Mount 시간 감소를 위한 블록 관리 방식

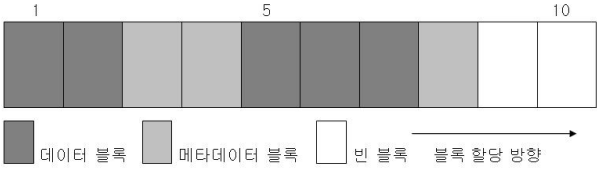
YAFFS 는 페이지를 할당 할 때 그림 1 과 같이 순

차적으로 할당해 나간다. 따라서 한 블록 내에 있는 각 페이지들은 데이터 page 일 수도 있고 메타데이터 일 수도 있다. 그렇기 때문에 YAFFS 는 mount 시 전체 플래시 메모리를 scan 해야만 메모리 상에 파일 시스템 tree 구조를 구축할 수 있다.



(그림 1) YAFFS 에서의 블록 할당 방식

이러한 단점을 극복하기 위하여 NAND 플래시 상에서 페이지를 할당할 때에 데이터 및 메타데이터에 따라 다른 블록을 할당할 것을 제안한다. 이렇게 메타데이터를 특정 블록에 집중하여 저장하면 mount 시점에 메타데이터가 저장된 블록들만 읽어 들임으로써 mount 시간을 감소할 수 있다. 그림 2 는 이와 같이 데이터와 메타데이터가 분리되어 저장된 블록들의 모습을 보여준다. 이렇게 메타데이터를 분리해서 저장하기 위해서 YAFFS 의 청크 allocation 방식을 수정하였다. YAFFS 내부에는 현재 청크 할당을 수행하는 블록 및 페이지를 관리하는 부분이 있다. 이 부분을 데이터와 메타데이터에 대해 각각 따로 관리하도록 수정 하였다.



(그림 2) 데이터, 메타데이터를 분리한 블록 할당

이렇게 데이터와 메타데이터가 저장되는 블록을 다르게 한 다음 mount 시점에 어떤 블록이 메타데이터를 위한 블록인지 알 수 있게 하기 위해 메타데이터 블록들의 위치를 NAND 플래시에 저장해야 한다. NAND 플래시는 특성상 overwrite 가 되지 않으므로 일반 디스크 기반 파일 시스템처럼 특정 위치에 super block 을 배치하는 방식을 사용할 수 없다. 이 점을 극복하기 위해 YAFFS 가 unmount 되는 시점에서 메타데이터가 저장된 블록들의 위치를 플래시의 가장 끝 부분에서부터 처음으로 비어있는 페이지에 기록한다. 그리고 mount 시점에서 플래시를 뒤에서부터 검색하여 메타데이터 블록 위치가 저장된 페이지를 읽어 들인다. 그러면 메타데이터들이 저장된 블록을 알 수 있고 해당 블록들만 읽어서 yaffs_Object 들의 tree 구조를 구축한다.

메모리 상에 yaffs_Object 들의 tree 구조를 구축할 때 해당 파일에 속해 있는 데이터 page 들의 아이디 역시 함께 기록되어야 한다. 이를 위해 플래시의 끝 부분에 메타데이터 블록 아이디를 기록한 다음 이어서 각 파일에 속해있는 데이터 page 들의 아이디를 함께 기록한다.

4. 성능 평가

NAND 플래시의 전체 블록 수를 N_b 라 하고 각 블록 내부의 페이지 수를 N_p 라 하자. 그리고 전체 블록 중 하나 이상의 page 에 정보를 저장한 블록의 비율을 p 라 하고 하나의 페이지를 읽는 시간을 T_p 라 하자. YAFFS 는 mount 할 때 정보가 들어있는 블록만 scna 한다. 따라서 기존 YAFFS 의 mount 시간은 다음 식으로 나타낼 수 있다.

$$T_{YAFFS\_MOUNT} = p \times N_b \times N_p \times T_p$$

논문에서 제안한 방식은 전체 블록이 데이터를 저장한 블록과 메타데이터를 저장한 블록으로 나뉜다. 이런 경우 mount 를 위해 메타데이터 저장 블록만 읽어 들이면 된다. 데이터 및 메타데이터 블록의 수를 각각 N_{bd}, N_{bm} 이라 하면 $N_b = N_{bd} + N_{bm}$ 이고 마운트 시간은 다음과 같다.

$$T_{MODIFIED\_YAFFS\_MOUNT} = p \times N_{bm} \times N_p \times T_p$$

따라서 기존 YAFFS 와 수정된 YAFFS 의 mount 시간 비율은 N_b/N_{bm} 이 된다. 이 비율을 구하기 위해서 먼저 메타데이터 블록의 수와 데이터 블록의 수의 관계를 구한다. 메타데이터 블록의 수에 블록당 페이지의 수를 곱하면 파일 시스템에 존재하는 전체 파일의 수가 된다. 여기에서부터 데이터 블록의 수를 구할 수 있다. S_f를 평균 파일 크기, S_b를 블록 크기라 하면 데이터블록의 수 N_{bd}는 다음과 같다.

$$N_{bd} = \frac{N_{bm} \times N_p \times S_f}{N_p \times S_b} = \frac{S_f}{S_b} N_{bm}$$

이 식과 $N_b = N_{bd} + N_{bm}$에서 성능 향상 비율 N_b/N_{bm}를 얻는다.

$$\frac{N_b}{N_{bm}} = 1 + \frac{S_f}{S_b}$$

평균 파일 크기가 200K 이고 블록 크기가 16K 인 플래시의 경우 N_b/N_{bm} 은 13.5 이다. 이는 수정된 YAFS 의 경우 기존 YAFFS 가 mount 를 위해 읽는 블록 수의 1/13.5 만큼의 블록 만을 읽음을 뜻한다. 따라서 논문에서 제안한 방법으로 많은 mount 시간의 개선을 기대할 수 있다.

5. 결론

YAFFS 는 NAND 플래시의 특성을 고려하여 만든 리눅스에서의 NAND 플래시 전용 파일 시스템이다. YAFFS 는 mount 시 전체 플래시 메모리 영역을 scan 하기 때문에 mount 시간이 오래 걸리는 단점이 있다.

본 논문에서는 YAFFS 에서 블록을 할당할 때에 데이터와 메타데이터가 할당되는 블록을 다르게 함으로써 mount 시간을 크게 감소 시킬 수 있는 방법을 제안 하였다.

참고문헌

[1] A Flash File System for Embedded Use, “http://www.yaffs.net/”