Ash FS: A Flash Drive File System

Daniel Băluţă

Faculty of Computer Science and Engineering
Politehnica University of Bucharest
Romania, Bucharest
Email: daniel.baluta@gmail.com

Gabriel Sandu

Faculty of Computer Science and Engineering
Politehnica University of Bucharest
Romania, Bucharest
Email: gabrim.san@gmail.com

Abstract—**The recent increase in USB Flash Drive capacities has triggered a need for a special file system that can handle large files such as the ones encountered in a NTFS environment, while keeping a good error correction and access speed.**

In Ash FS, we are trying to reduce the wear levelling of the flash drive by storing such large quantity of data in an uniform way and provide the possibility of using compression, crypting and error correction techniques.

I. INTRODUCTION

A. Flash

Flash memory is non-volatile computer memory that can be electrically erased and reprogramed. During last years it become a common storage medium in embedded devices, because it provides solid state storage with high reliability, at a relatively low cost.

Flash is a specific type of EEPROM (Electrically Erasable Programmable Read-Only Memory) that is erased and programmed in large blocks;in early flash the entire chip had to be erased at once.It is available in two major types - the traditional NOR flash which is directly accessible, and the newer, cheaper NAND flash which is addressable only through a single 8-bit bus used for both data and addresses, with separate control lines. This types of flash share their most important characteristics - each bit in a clean flash chip will be set to a logical one , and can be set to zero by a write operation.

Flash chips are arranged into blocks which are typically 128KB on NOR flash and 8KB on NAND flash.Resetting bits from zero to one cannot be done individually, but only by reseting or erasing a complete block.The lifetime of a flash chip is measured in such erase cycles, with the typical lifetime being 100,000 erases per block.To ensure that no erase block reaches this limit before the rest of the chip, most users of flash chips attempt to ensure that erase cycles are evenly distributed around the flash; process known as "wear levelling".

Aside from the difference in erase block sizes, NAND flash chips also have other differences from NOR chips.They are further divided into "pages" which are typically 512 bytes in size, each of which has an extra 16 bytes of "out of band" storage space, intended to be used for metadata or error correction.NAND flash is written by loading the required data into an internal buffer one byte at a time , then issuing a write command. While NOR flash allows bits to be cleared individually until thare are none left to be cleared, NAND flash allows only ten such write cycles to each page before leakage causes contents to become undefined until the next erase of the block in which the page resides.

B. Flash Translation Layers

The majority of applications of flash for file storage have involved using the flash to emulate a block device with standard 512-byte sectors, and then using standard file systems on that emulated device.

The simplest method of achieving this is to use a simple 1:1 mapping from the emulated block device to the flash chip, and to simulate the smaller sector size for write requests by reading the whole erase block, modifying the appropriate part of the buffer, erasing and rewriting the entire block.This approach provides no wear levelling, and is extremely unsafe because of the potential power loss between the erase subsequent rewrite of the data. However, it is acceptable for use during development of a file system which is intended for read-only operation in production modles.The mtdblock Linux driver provides this functionality, slightly optimised to prevent excessive erase cycles by gathering writes to a single erase block and only performing the erase/modify/writeback procedure when a write to a different erase block is requested.

To emulate a block devicde in a fashion suitable for use with a writable file system, a more sophisticated approach is required.

To provide wear levelling and reliable operation, sectors of the emulated block device, are stored in varying locations on the physical medium, and a "Translation Layer" is used to keep track of current location of each sector in the emulated block device.This translation layers is effectively a form of journalling file system.

The mos commmon such translation layer is a component of PCMCIA specification, the "Flash Translation Layer"

(FTL). More recently, a variant designed for use with NAND flash chips has been in widespread use in the popular DiskOnChip devices produced by M-Systems.

Unfortunately, both FTL and the newer NFTL are encumbered by patents. M-Systems have granted a license for FTL to be used on all PCMCIA devices, and allow NFTL to be used only on DiskOnChip devices.

Linux supports both of these translation layers, but thier use is deprecated and intended for backwards compatibility only.Not only are there patent isues, but the practice of using a form of journalling file system to emulate a block device, on which a "standard" journalling file system is then used, is unnecessarily inefficient.

A far more efficient use of flash technology would be permitted by the use of a file system designed specifically for use on such devices, with no extra layers of translation in between.

Ash File System is a block device based file system and tries to take in consideration the particular characteristics of flash memory.

II. ASH FILE SYSTEM

A. Storage Format

Add text for Storage Format

B. Mounting

Add text for Mounting

C. Operations

Add text for operations

D. Cryptography

Add text for cryptography

E. Compression

Add text for compression

III. FUTURE DEVELOPMENT

Subsection text here.

REFERENCES

[1] David Woodhouse, *JFFS: The Journaling Flash File System*, Ottawa Linux Symposium,2001
[2] Han-Joon Kim, Sang-Goo Lee, *A new flash memory management for flash storage system*, COMPSAC '99
[3] Charles Manning, *YAFFS: the NAND-specific flash file system*, Linuxdevices.org, September 20th 2002
[4] Eran Gal, Sivan Toledo, *A Transactional Flash File System for Microcontrollers*, USENIX '05
[5] Seung-Ho Lim, Kyu-Ho Park, *An efficient NAND flash file system for flash memory storage*, IEEE Transaction on Computers, 55th Vol., July 2006