Clément DUTU
Simon AUTECHAUD

File System: YAFFS and YAFFS2

I. Introduction

1. What is YAFFS

YAFFS (Yet Another Flash File System), created in 2002 is a file system specifically designed for NAND-type flash memory devices. It was created to improve on raw NAND flash memory, especially in embedded systems. But can also work with other solid-state memory such as NOR flash.

Thanks to its architecture, this file system has a very fast mount speed and uses very little RAM. As such, we can find Yaffs in mobile devices, cameras or even satellites.
Finally it needs special handling as it does not have a built-in controller like traditional block storage (ex: HD).

2. Flash memory: NOR vs NAND

To understand how this file system works, we have to understand the difference between NOR and NAND memory.

NOR memory uses transistors in parallel, which allow a fast random access memory.
On the other hand, NAND uses memory cells connected in series, resulting in very fast sequential memory access but slower random access memory. As such, NAND memory is ideal for systems such as Hard Drives or thumb drives.

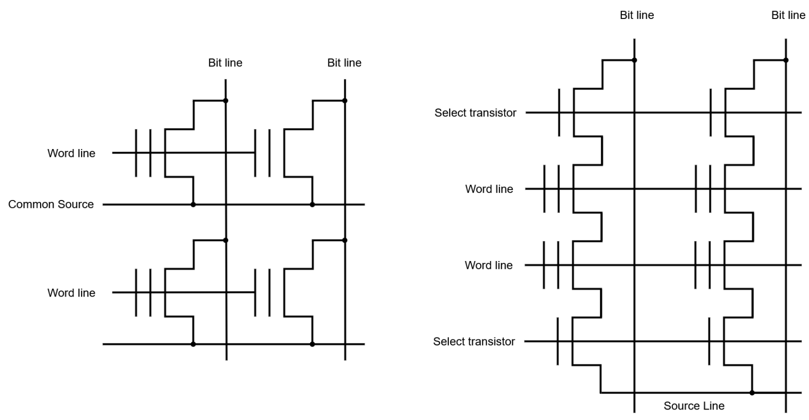

Fig 1: NOR (left) vs NAND (right) architecture

II. Yaffs architecture

1. File system structure

Yaffs stores data into areas called Pages, Blocks and OOB like so:

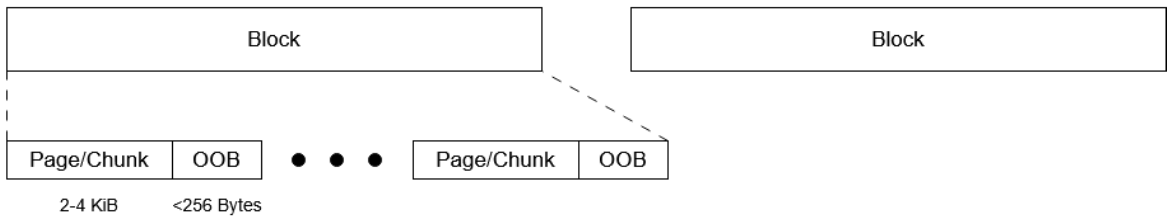

Fig 2: Diagram of the areas used in YAFFS

Blocks:
A block is made up of a group of pages, generally around 64 to 256. They are the smallest unit that can be written and erased by the system.

Pages:
They are the smallest unit that can be read by the system. They are of a fixed size (2 KiB to 4 KiB) and contain either the data of an object or the metadata describing one, in that second case they are called object chunk headers. All pages are followed by an OOB area.

OOB (Out-Of-Bounds):
Chunks containing by default the metadata of a page such as but not limited to the object ID, the chunk index (offset of the object), the chunk type (ex: header/data/deleted marker) and a sequence to identify the latest copy.
They also keep track of bad blocks tags for the system to keep track and ignore.

It is using these structures that YAFFS stores data. It is also good to note that all files and directories are considered objects in the file system and are assigned a unique 32-bit ID, allowing for a O(1) lookup. In addition, YAFFS implements a double linked list to have a fast lookup by name.

Note:
The difference between YAFFS1 and YAFFS2 is that YAFFS2 extended the size of the pages from 512 B to 2 Kib/4 KiB.

Here is an example of the tree structure used in YAFFS:

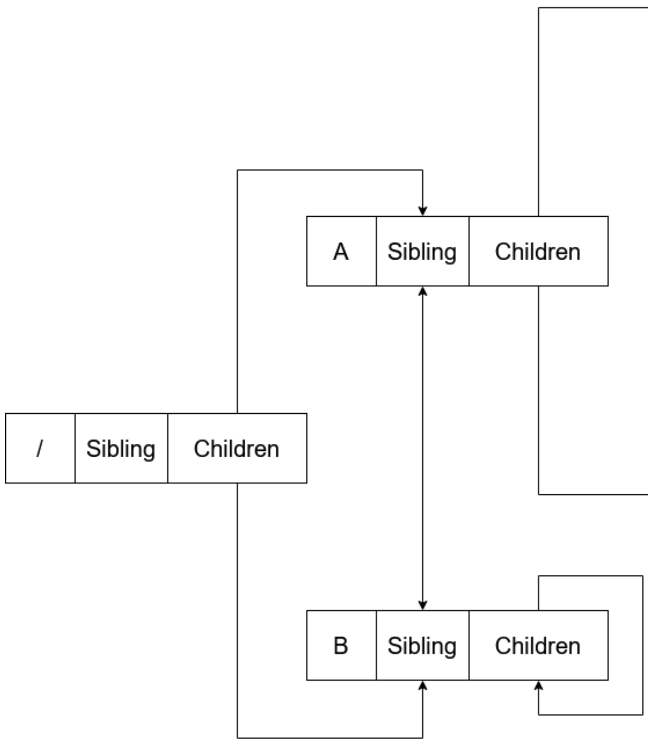

Fig 3: Tree structure of the objects in YAFFS

/	Root directory
/a	Directory
/b	Empty directory
/a/c	Empty directory
/a/hello	Data file

III. Behavior

YAFFS is subject to a few rules, such as:

- “Write-once”: Writing operations can only be done for entire blocks. In addition writing can only go from all‑1’s by erasing, to some 0’s. A 0 cannot be turned into a 1 without erasing the entire block.
- Limited cycles: Each block has a life span and can only be written/erased a certain number of times (around 10 000 to 100 000 times).
- Log structured: Meaning that every time data is modified, all of it is written to a new location while invalidating the previous location.
- Bad block managing: The system keeps track of the bad blocks

Writing/modifying data:

Being built around these constraints, YAFFS is never modifying pages in place. Instead, it writes new copies (“out‑of‑place”) and invalidates old ones, then reclaims invalid pages through garbage collection.

When a read/write/modify action is performed, a new metadata header chunk AND complete data chunks are created. Previous data is discarded by the file system. New data is never overwritten on old data

Garbage Collection

YAFFS implements a garbage collection to reclaim memory. It iterates through blocks to find deleted chunks, if it does, a thread copies the valid chunks and then reclaims the block by erasing it. This way of doing also allows for wear management of the memory cells.

Journaling

- Regularly, YAFFS uses checkpoints to save the current page tree structure that allows a fast mount using these checkpoints
- Checkpoints are stored as chunks in the log (storing hierarchy, free blocks, IDs, etc)