IRON for JFFS2

Presented by:
Abhinav Kumar
Raja Ram Yadhav Ramakrishnan

Motivation:

• Flash Storage is being increasingly used in the internal memory of Smart Phones

• Among the current flash file systems the foll two are dominant:

1. JFFS (Journaling Flash File system)
2. YAFFS(Yet Another Flash File system)

• IRON-like analysis has not been on Flash File Systems

Why an IRON-like analysis ?:

- Tests performed without modifying file system code

- Same pseudo-driver can be used to study other Flash File systems too

NAND Storage Basics:

- Memory is arranged as an array of pages
- A page consists of 256 / 512 Byte data and 8 / 16 Byte spare (out of band) area
- The spare area is used to store ECC (error correction code), bad block information and filesystem-dependent data
- Wear leveling is needed

Some Flash File system Basics:

- Flash doesn't match the description of either block or char devices
- Differences between flash and block devices
- A special device called “MTD” was created

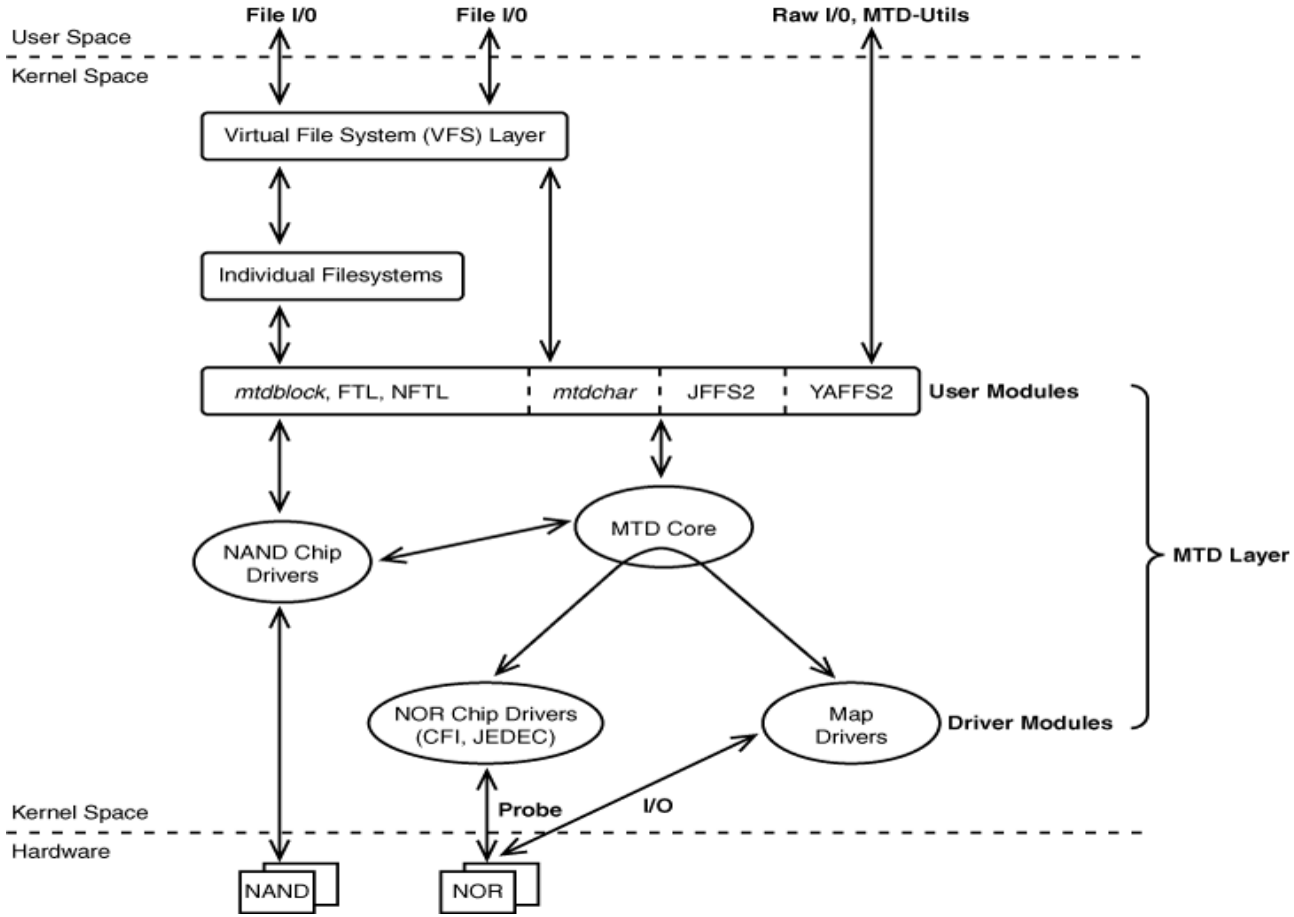
Flash File system Basics continued…

- USB Flash Drives, SD Cards also treated as Block Devices

- File Translation Layer(FTL) needed to use traditional File Systems like ext2 on MTD devices

- Why we shouldn’t use FTL…..

Flash Storage Stack:



About JFFS2:

- Log Structured File System to enable write leveling
- On-the-fly compression and decompression
- Uses Cleanmarkers to indicate whether a block was successfully erased

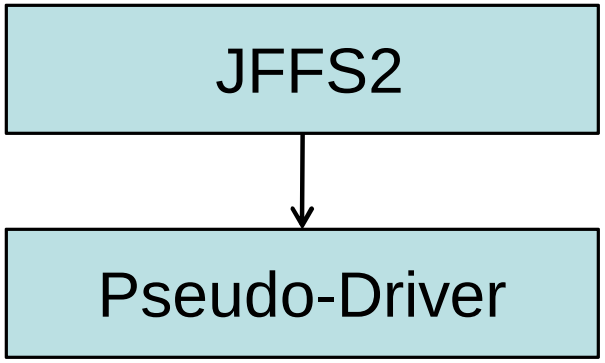
0x08	0x09	0x0a	0x0b	0x0c	0x0d	0x0e	0x0f
0x85	0x19	0x03	0x20	0x08	0x00	0x00	0x00

High-level implementation details:

- NAND simulator (nandsim)
- Provided us with a useful interface to simulate faults
- Used the interfaces to conduct several “IRON” like tests

Tests Performed:

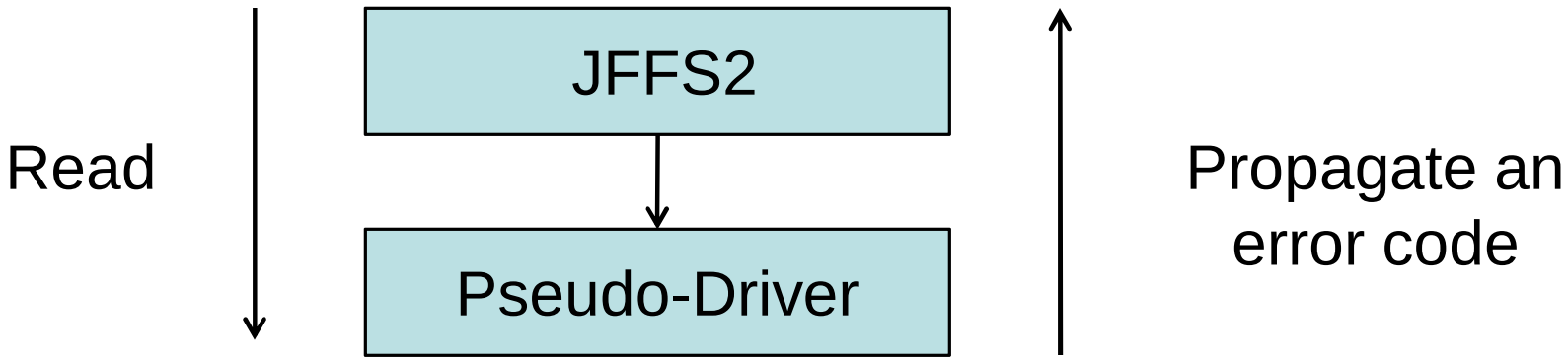
- Read Failure of entire block
- Read Failure (single bit)
- Read Failure (multiple bits)
- Write Failure
- Erase Failure
- Write Failure during GC



- Progress table

Test Performed	Error Detected?	Recovered from Error?	Comments

- Read Failure (entire page)



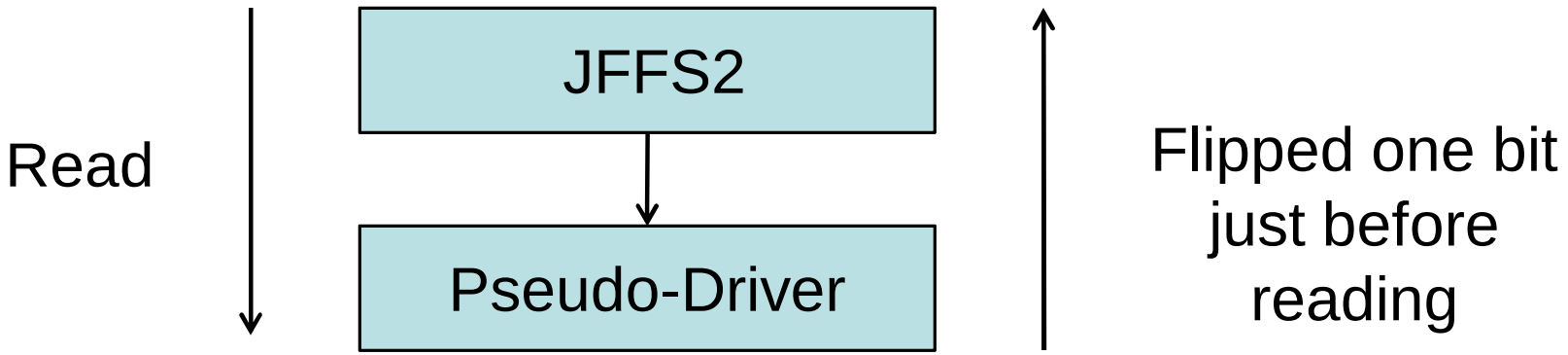
- Kernel Error Message Observed:

```
JFFS2 error: (10411) jffs2_get_inode_nodes: can not read 512 bytes
from 0x01ff8200, error code: -1.
JFFS2 error: (10411) jffs2_do_read_inode_internal: cannot read
nodes for ino 2, returned error is -1
Returned error for crccheck of ino #2. Expect badness...
```

- Progress table

Test Performed	Error Detected ?	Recovered from Error?	Comments
Read Failure (entire page)	✓	✓	Error Propagation but no retry

- Read Failure (single bit)



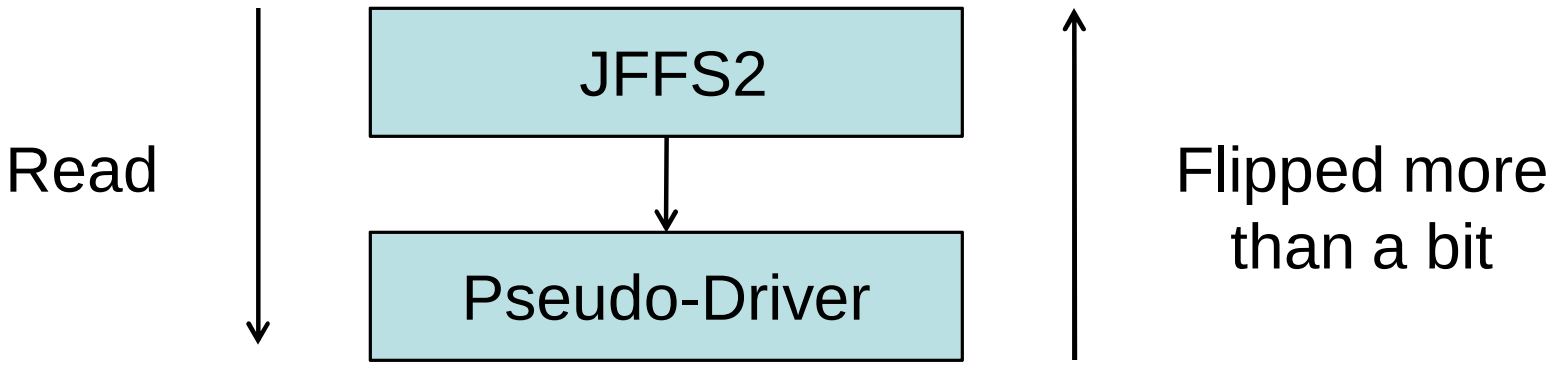
- Observation

Successful detection and correction

- Progress table

Test Performed	Error Detected ?	Recovered from Error?	Comments
Read Failure (entire page)	✓	✓	Error Propagation but no retry
Read Failure (single bit)	✓	✓	Error was detected and also corrected using checksum

- Read Failure (multiple bits)



- Kernel Error Message Observed:

```
JFFS2 notice: (8889) read_dnode: wrong data CRC in data node at
0x01ff8200: read 0x1aaee62e, calculated 0x6cae3da6.
```

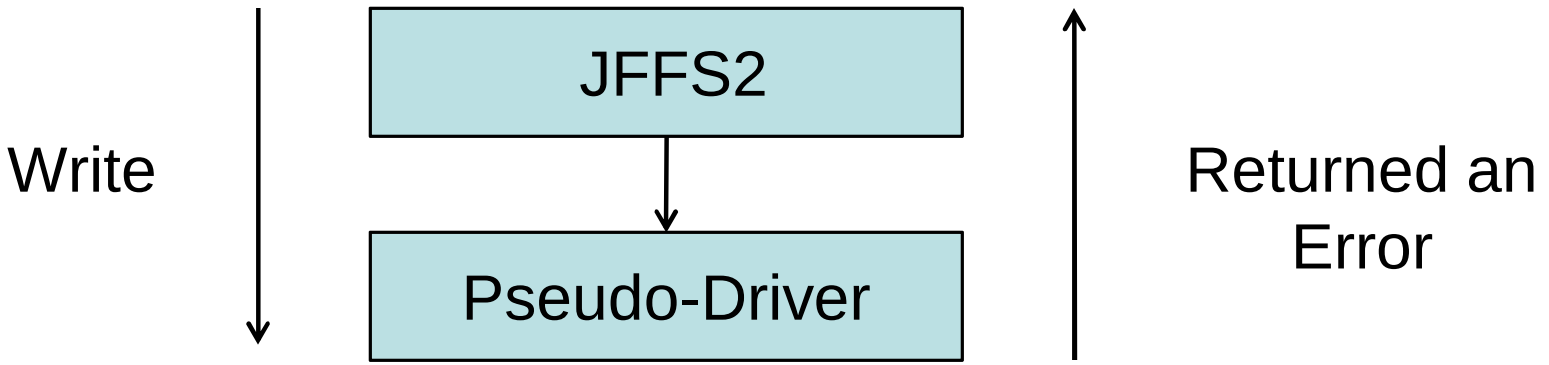
- Progress table

Test Performed	Error Detected?	Recovered from Error?	Comments
Read Failure (entire page)	✓	✓	Error Propagation but no retry
Read Failure (single bit)	✓	✓	Error was detected and also corrected using checksum
Read Failure (multiple bits)	✓	✗	Error detection No error correction No error propagation No retry

Why?:

```
static inline int read_dnode(struct jffs2_sb_info *c, struct jffs2_raw_node_ref
*ref,
                                     struct jffs2_raw_inode *rd, int rdlen,
                                     struct jffs2_readinode_info *rii)
{
          int ret = 0;
          ********
          ********
          if (len >= csize && unlikely(tn->partial_crc != je32_to_cpu(rd-
>data_crc)))
          {
                    JFFS2_NOTICE("wrong data CRC in data node at 0x%08x: read %#08x,
                    calculated %#08x.\n",ref_offset(ref), tn->partial_crc,
                    je32_to_cpu(rd->data_crc));
                    jffs2_mark_node_obsolete(c, ref);
                    goto free_out;
          }
          **********
          **********
          free_out:
                    jffs2_free_tmp_dnode_info(tn);
                    return ret;
}
```

- Write Failure



- Observation:

  Infinte retry leading to kernel crash

- Progress table

Test Performed	Error Detected?	Recovered from Error?	Comments
Read Failure (entire page)	✓	✓	Error Propagation but no retry
Read Failure (single bit)	✓	✓	Error was detected and also corrected using checksum
Read Failure (multiple bits)	✓	✗	Error detection No error correction No error propagation No retry
Write Failure	✓	✗	Error Propagation and infinite retry

Why?:

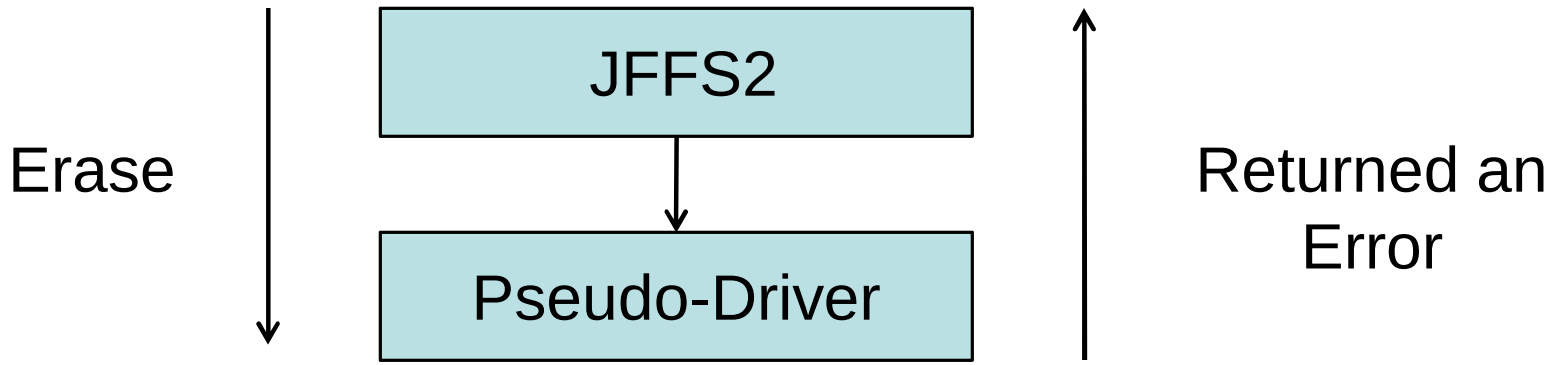
```
int jffs2_flush_wbuf_gc(struct jffs2_sb_info *c, uint32_t ino)
{
******
while (old_wbuf_len && old_wbuf_ofs == c->wbuf_ofs)
        {

                mutex_unlock(&c->alloc_sem);
                D1(printk(KERN_DEBUG "jffs2_flush_wbuf_gc() calls gc pass\n"));
                ret = jffs2_garbage_collect_pass(c);
                if (ret) {
                        /* GC failed. Flush it with padding instead */
                        mutex_lock(&c->alloc_sem);
                        down_write(&c->wbuf_sem);
                        ret = __jffs2_flush_wbuf(c, PAD_ACCOUNTING);
                        /* retry flushing wbuf in case jffs2_wbuf_recover
                           left some data in the wbuf */
                        if (ret)
                                ret = __jffs2_flush_wbuf(c, PAD_ACCOUNTING);
                        up_write(&c->wbuf_sem);
                        break;
                }
*****
```




- Progress table in GC path

Test Performed	Error Detected?	Recovered from Error?	Comments

- Erase Failure



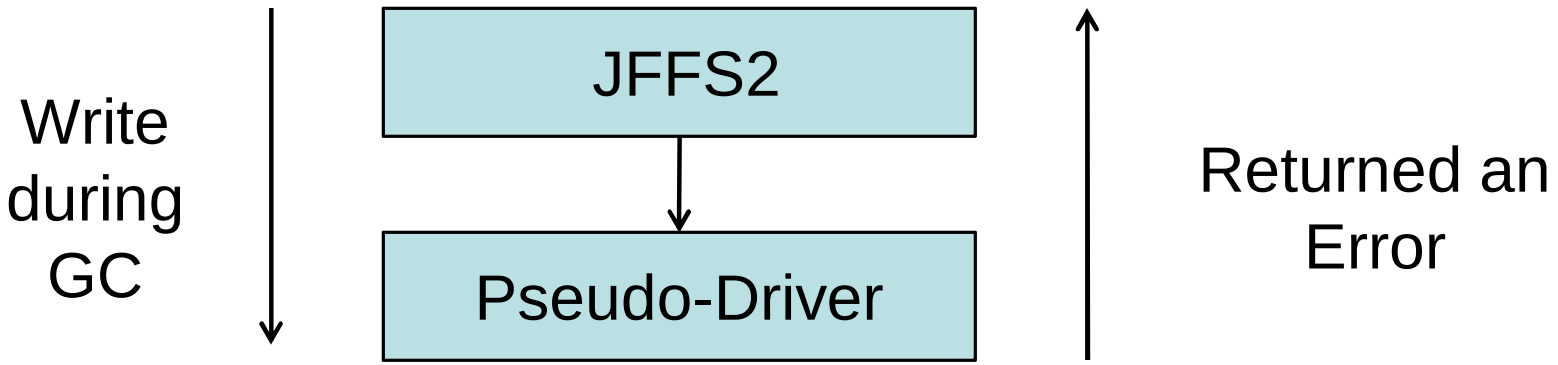
- Kernel Error Message Observed:

```
 Erase at 0x01ffc000 failed immediately: errno -1
No space left on device
```

- Progress table in GC path

Test Performed	Error Detected?	Recovered from Error?	Comments
Erase Failure	✓	✓	Error propagation but no retry

- Write Failure in GC path:



- Observation:

  Infinite retry observed because of the same reason

- Progress table in GC path

Test Performed	Error Detected?	Recovered from Error?	Comments
Erase Failure	✓	✓	Error propagation but no retry
Write Failure	✓	✗	Infinite retry with kernel crash

Test Performed	Error Detected?	Recovered from Error?	Comments
Read Failure (entire page)	✓	✓	Error Propagation but no retry
Read Failure (single bit)	✓	✓	Error was detected and also corrected using checksum
Read Failure (multiple bits)	✓	✗	Error detection No error correction No error propagation No retry
Write Failure	✓	✗	Error Propagation and infinite retry
Erase Failure	✓	✓	Error Propagation but no retry
Write Failure in GC path	✓	✗	Infinite retry with kernel crash

Conclusion:

- Several tests were performed on JFFS2
- It has many favorable characteristics:
    1. On-the-Fly Compression and Decompression
    2. Checksumming
    3. Good Recovery techniques like Error propagation, retry on most cases

- However the recovery technique isnt apt in some cases like Read Failure(multiple bits), Write failure

Future Work:

- Further exploration of the GC path
- Failure of specific metadata blocks such as i-node, i-node map etc

References:

- **JFFS : The Journalling Flash File System**
  By David Woodhouse, Red Hat Inc.

- **IRON File Systems**
  By Vijayan Prabhakaran, Nitin Agrawal, Lakshmi Bairavasundaram, Haryadi Gunawi, Andrea C. Arpaci-Dusseau and Remzi H. Arpaci-Dusseau

- **Essential Linux Device Drivers**
  by Sreekrishnan Venkateswaran

- **Linux Device Drivers, Third Edition**
  Jonathan Corbet, Alessandro Rubini,Greg Kroah-Hartman

Acknowledgements:

• Special thanks to Prof. Remzi Arpaci-Dusseau for giving us a chance to work on this project

Questions?