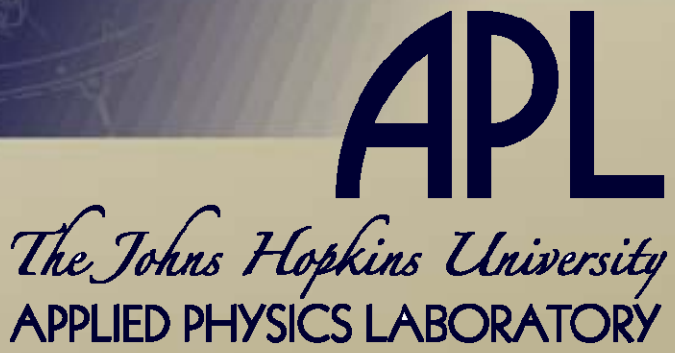
Flash File System for a “Smart” SSR:

Modifying YAFFS for Onboard Usage

David Edell

David.Edell@jhuapl.edu
443-778-2053



Overview

- Introduction
- Unique characteristics of NAND Memory
- Flash file systems
- YAFFS
- YAFFS modifications
- Future work
- Conclusion

Introduction

- **Spacecraft solid state recorders (SSRs) are increasingly using Flash memory.**

- **Flight applications impose requirements on Flash to provide:**
  - **High throughput**
  - **High reliability**
  - **Large disk sizes**
    - **Potential support for 128Gbit or larger devices**
  - **Low memory footprint**
    - **A requirement shared to a lesser degree by mobile devices in the commercial world**

Flash File System

- A file system is required to minimize software complexity and maximize effective usage of system resources.
- Characteristics of NAND Flash impose additional requirements on the selected file system
  - NAND Flash devices consist of write-once, read-many pages. A page is the smallest read/write block.
  - Pages are grouped into erase blocks.
    - A page cannot be edited until the entire block is erased.
  - Each block has a limited number of write cycles in its lifetime
    - To optimize disk life, it is necessary to use wear-leveling to distribute disk usage across the device

Trade Study

A trade study was conducted to evaluate several candidate file systems including:

- **VxWorks True Flash File System (TFFS)**
  - **TFFS is an abstraction layer allowing native DOS, FAT, or RAW file systems to be used on a Flash device**
- **YAFFS**
- **JFFS**
- **UBIFS**

Trade Study Context

- **Basic file system functionality**
- **Special file system functionality**
  - **Requirements unique to space systems**
- **Derived technical requirements**
  - **Compatibility with real-time operating systems (vxWorks, RTEMS)**
- **Performance, resource utilization and capacities**
  - **Based on expected mission requirements**

Trade Study Functionality

- **Basic file system functionality**
  - Capability to perform standard file I/O
  - Capability to manage files and directories (i.e.: move, rename, delete)
  - These functions support flight software development and can reduce complexity in mission operations

- **Special file system functionality**
  - Requirements unique to space systems including:
    - Explicit or deterministic garbage collection
    - Chip level power management
    - Efficient wear-leveling strategy
    - Power management concerns
    - Incorporation of Reed-Solomon error detection and correction

Trade Study Requirements/Performance

- **Derived technical requirements**
  - Compatibility with real-time operating systems (vxWorks, RTEMS)
- **Performance, resource utilization and capacities**
  - Expected mission requirements for:
    - Disk capacity
    - Number and size of files
    - Mount and file I/O response times
    - Memory (RAM) usage
    - Reliability, particularly in the event of a sudden power failure

Evaluation Results (1/2)

- **VxWorks TFFS**
  - **Relies on a translation layer to simulate traditional magnetic drives on solid state disks**
  - **Limited to a maximum 2 GB disk size**
  - **No adapters exist for NAND memory.**
  - **Existing NOR adapters may not function correctly for NAND flash.**

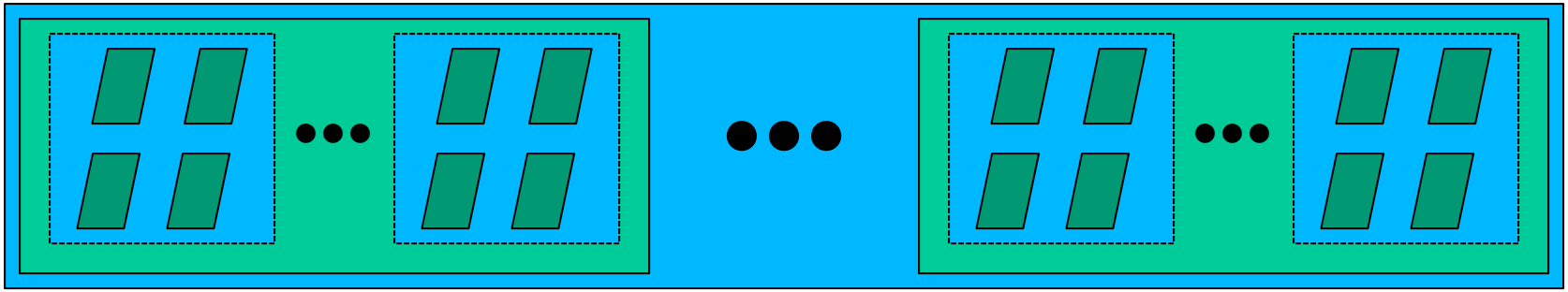
Evaluation Results (2/2)

- **UNIX/Linux Derived File Systems**
  - **These are non-native to VxWorks and must be ported**
  - **May not be optimized for usage in a Real-time OS**
  - **Of the file systems investigated, only YAFFS:**
    - **Has been used in a real-time OS**
    - **Is designed with a direct interface for simplified integration into diverse environments**
    - **Is designed to use minimal system resources**
    - **Is specifically designed for usage with NAND Flash**
    - **Handles sudden power outages in a simple and straight-forward way**
      - **Provides a test suite for verifying such events.**

Yet Another File System (YAFFS): Overview

- YAFFS was designed specifically for NAND Flash memory
- It is currently used by Google's Android OS for mobile devices (ie:SmartPhones)

- YAFFS utilizes three units for addressing the disk:
  - The physical page
  - The chunk, a group of N sequential pages treated as a single allocation block. This is an artificial abstraction added by YAFFS to optimize usage of larger disks
  - A block, the smallest erase unit supported by the NAND device



YAFFS: Memory Usage

Predicted RAM requirements for YAFFS:

128 Gbit disk with 1:1 mapping of page to chunk

SSR Size (bits)	1.37439E+11	(Bits)
SSR Size (M Bytes) (hex)	4000	HEX
SSR Size (M Bytes) (dec)	16384	
SSR Size (G Bytes) (hex)	10	G Bytes HEX
SSR Size (G Bytes) (dec)	16	G Bytes DEC
SSR Size (Bytes) (dec)	17179869184	
SSR Size in Flash Pages	8388608	
SSR Size in Flash Erase Units	65536	
Chunk size in flash pages	1	High Density Flash Pages (2048 Bytes per Cluster.)
Bits Per Cluster Address in T-Nodes	16	(2 Byte Addresses)
SSR Size in Clusters	8388608	(23 bit Adresses = 128 cluster hash buckets = 128 flash pages)
Cluster size (hex)	800	
Cluster size (bytes)	2048	
Bits	16384	
Device Temporary Cluster Buffers	1	
Max Open Files	50	(3 per instrument + 10 for spacecraft & playback)
Directories + CFDP	10	
Max Files on SSR	5000	
RAM for file T-Nodes	16384	K Bytes
RAM for Erase Block Info	512	K Bytes
RAM for Chunk Allocation Bits	1024	K Bytes
RAM for File / Directory Structure	406	K Bytes
RAM for Temporay Device (Cluster) Buffers	2	K Bytes
RAM required for key data structures:	**18328**	K Bytes

YAFFS: Memory Usage

Predicted RAM requirements for YAFFS:

128 Gbit disk with mapping of 16 pages per chunk

SSR Size (bits)	1.37439E+11	(Bits)
SSR Size (M Bytes) (hex)	4000	HEX
SSR Size (M Bytes) (dec)	16384	
SSR Size (G Bytes) (hex)	10	G Bytes HEX
SSR Size (G Bytes) (dec)	16	G Bytes DEC
SSR Size (Bytes) (dec)	17179869184	
SSR Size in Flash Pages	8388608	
SSR Size in Flash Erase Units	65536	
Chunk size in flash pages	16	High Density Flash Pages (32768 Bytes per Cluster.)
Bits Per Cluster Address in T-Nodes	16	(2 Byte Addresses)
SSR Size in Clusters	524288	(19 bit Adresses = 8 cluster hash buckets = 128 flash pages)
Cluster size (hex)	8000	
Cluster size (bytes)	32768	
Bits	262144	
Device Temporary Cluster Buffers	1	
Max Open Files	50	(3 per instrument + 10 for spacecraft & playback)
Directories + CFDP	10	
Max Files on SSR	5000	
RAM for file T-Nodes	1024	K Bytes
RAM for Erase Block Info	512	K Bytes
RAM for Chunk Allocation Bits	64	K Bytes
RAM for File / Directory Structure	406	K Bytes
RAM for Temporay Device (Cluster) Buffers	32	K Bytes
RAM required for key data structures:	**2038**	K Bytes

YAFFS Memory Usage

- **In this case, changing the chunk size results in an 8x reduction in memory size**
  - **From approximately 18,000k to 2,000k**

- **YAFFS defines and documents this concept**
  - **YAFFS2, the latest released version, does not implement this feature.**
  - **Existing YAFFS versions assume a 1:1 mapping of chunk to page**
    - **Often inconsistent terminology within the codebase.**
    - **The resulting memory optimizations described in the YAFFS descriptions are therefore not available for larger disks**

YAFFS Guts

We decided to implement this enhanced capability

The first task was to gain a working understanding of the YAFFS codebase

- YAFFS is an open source project released under the GPL
- Documentation, inside and outside source, was sparse.
- As functions and variables were identified, I annotated the code accordingly for our own reference.

YAFFS Guts

Next, Variables had to be cleaned up

- One of the most time consuming tasks was to rename all references to pages and chunks
  - YAFFS 1/2 uses page and chunk interchangeably
  - To create our YAFFS3, this distinction must be made explicit.
  - A simple find and replace served as a basis, but a line-by-line analysis of each variable was required to differentiate between when a page vs. a chunk was needed.

YAFFS3

New Functional Layers were required to perform chunk-level operations

- Existing functions are renamed to reflect page-scope, and removed where no longer applicable

YAFFS Addressing changes from Page-Level to Chunk-Level

- Allocation reserves a block of N-sequential pages per chunk
- Setting N=1 is functionally equivalent to legacy YAFFS

YAFFS3

- **Read and write functions continue to operate on pages at a lower level**
  - **The file-system determines the desired chunk Id (address)**
  - **I/O operations use offset to determine the actual page-in-chunk to operate on**

- **File cache operates on 1-page segments**
  - **At most one page per chunk is cached during normal operations**
  - **Future revisions will allow additional caching options in support of large edit operations within a file.**
    - **This may include adding a pointer to the original chunk to the cache information, such that the remaining pages within the chunk can be automatically copied over as necessary at the end of the edit operation.**

Future

- **Additional work is needed to further optimize our additions to the YAFFS file system in a flight environment.**
  - **Optimizations for file editing, an operation not generally used in flight systems.**
  - **Enhancements to reduce duplicate page header information within a chunk, providing a marginal increase in available disk space.**
  - **Enhancements to prevent file descriptors from requiring a full chunk to be allocated.**

- **We are beginning the evaluation of YAFFS as an integrated VxWorks file system with a commercial NAND Flash Card.**
  - **Preliminary testing was conducted using the RAM emulation and the direct C interface**

APL