Protective Avionics Flash File System
PAFFS

Pascal Pieper

2. Mai 2017

Table of contents

Übersicht

Overview

Computer guided spaceflight

- Consisting of many subparts
  - Focus of this work is mass storage
- Experiment data (payload)
- Instructions
- Program images

  → Many applications for long term memory

Overview

Negative influences on memory

- Vibrations
- Radiation
- Rapid temperature changes
- Hard heat dissipation

Overview

Solution

- Radiation tolerant and robust memories
  - High cost

Cheap memories

- Compensate error rate with filesystem
- Its logic optimizes lifetime and reliability

Use

Cheap memory in space

NAND Flash

- Can write a page only once (512-4096 Bytes)
- Can delete only a whole block (16-512 Pages)
- Deletions can only happen rarely (100.000-100 Erases)

Requirements

- Take care of NAND specialities
  - → Especially the low lifetime
- Manage multiple redundant chips
- Tolerate bit errors as well as total loss of single chips
- Show minimal RAM footprint while being able to scale with increasing size memories
- Offer POSIX related file interface
- Minimize loss of data after unexpected power failure

Requirements

Tradeoff

- Read-/Writespeed $\longleftrightarrow$ RAM usage
- Wear $\longleftrightarrow$ RAM usage, fail safety
- Efficiency of data storage $\longleftrightarrow$ RAM usage, fail safety

Übersicht

Inodes

- Represent an Object
  - → File, directory or softlink
- Point to data chunks containing each objects contents
- ... and some other metadata such as an unique ID and size

Tree Index

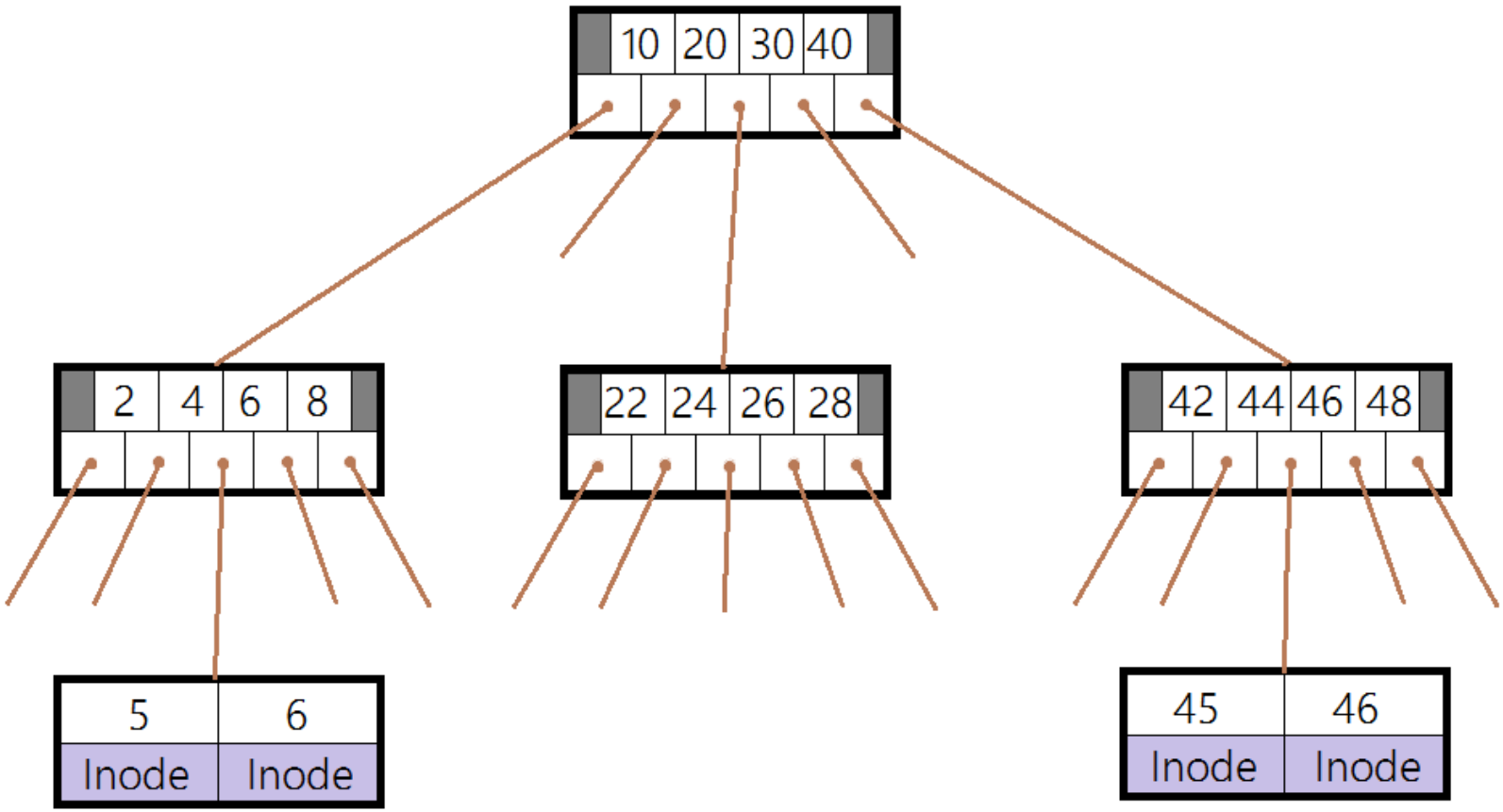
Structure

- Contains all Inodes
- Is ordered by Inode ID in a B+Tree
- Branches contain pointers to branches or leaves
- Leaves contain Inodes

Advantages

- All Operations $\mathcal{O}(\log n)$
- Only one tree path is changed upon write, not the whole index (see tables etc)

Tree Index


10 20 30 40
2 4 6 8
22 24 26 28
42 44 46 48
5
6
Inode
Inode
45
46
Inode
Inode

Difficulties

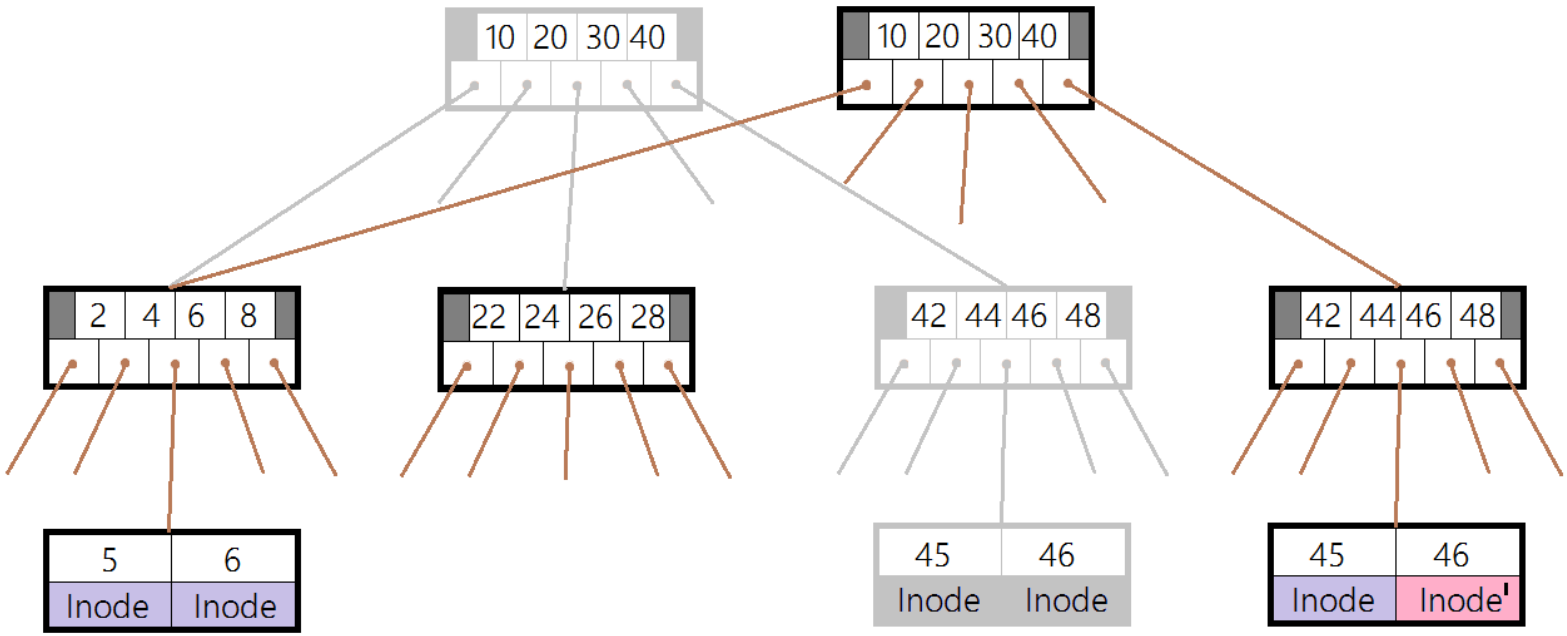
Change in a file

- → changes location of the files data
- → changes location of Inode
- → changes location of corresponding leave
- → changes location of every parent branch including root node

How to approach

- Reduce wear by caching a subset of tree index and root node address
- But still: how to find a ever changing root node?

Tree Index


10 20 30 40
10 20 30 40
2 4 6 8
22 24 26 28
42 44 46 48
42 44 46 48
5
6
Inode
Inode
45
46
Inode
Inode
45
46
Inode
Inode'

Superblock

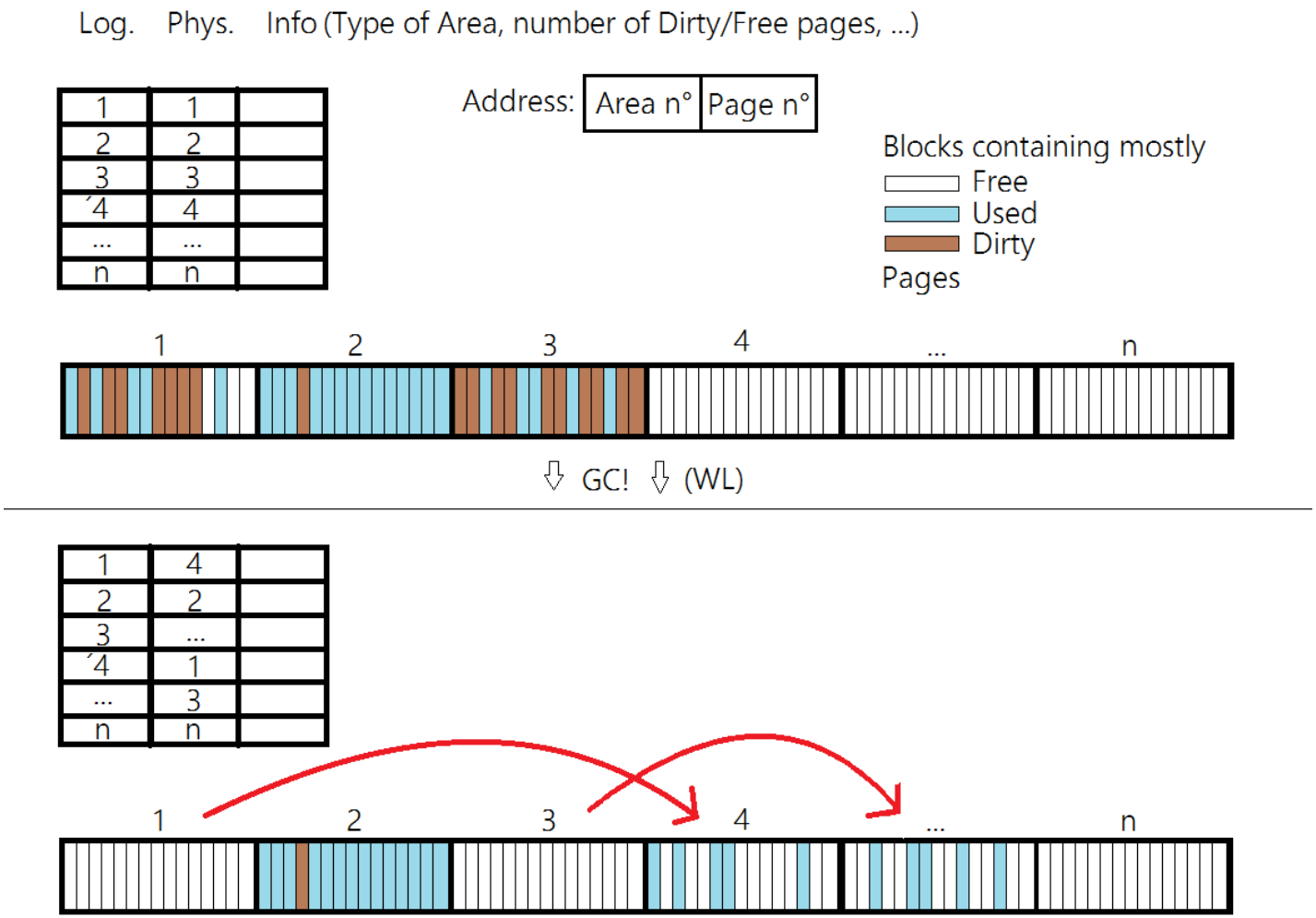
Chaining

- First two valid Areas contain anchor entries pointing to jump pads
- Jump pads point to other pads until final super page is reached
- Super page contains address of root node and uncommitted area summaries

Areas

- Combine erase blocks to a logical group
- Act as a single erase block
- Abstract logical and physical position on flash
- Contain only data of one type of `superblock`, `index`, `data` and `journal`
- *Areasummary* and its cache

Areas


Log. Phys. Info (Type of Area, number of Dirty/Free pages, ...)
1 1
2 2
3 3
´4 4
... ...
n n
Address: Area n° Page n°
Blocks containing mostly
Free
Used
Dirty
Pages
1 2 3 4 ... n
⇩ GC! ⇩ (WL)
1 4
2 2
3 ...
´4 1
... 3
n n
1 2 3 4 ... n

Error correction

Idea

- Every flash chip maintains its own, valid filesystem
- *OOB* is fully used for ECC (other fs's use the OOB-area to store other metadata as well)
- Check happens on read actions and during periodic scrubbing
- Bit errors are, if possible, corrected on the fly by rewriting the page to another location
- If bit error is not correctable, a valid copy from another image is taken

Journaling

Idea

- Keeps the filesystem sane when a sudden powerloss occurs
- Logs the intention to modify flash before actual conduct
- For every Action that is performed only in RAM (caching)
  - And for write operations on flash
- If filesystem is interrupted, it can either revert changes or continue the last operation