

Protective Avionics Flash File System PAFFS

Pascal Pieper

26. April 2017

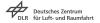




Table of contents

- 1 Introduction
- 2 Requirements
- 3 Konzept





Übersicht

- 1 Introduction
 - Overview
 - Idea
- 2 Requirements
- 3 Konzept



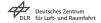


Overview

Computer guided spaceflight

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

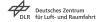




Use

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

- Wear ←→ RAM usage, fail safety
- lacktriangle Efficiency of data storage \longleftrightarrow RAM usage, fail safety

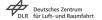




Übersicht

- 1 Introduction
- 2 Requirements
- 3 Konzept
 - Inodes und Tree Index
 - Superblock
 - Areas und Garbage Collection
 - Echlorkorrold
 - Fehlerkorrektur und Redundanz





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



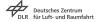


Tree Index

Structure

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





Difficulties

Change in a file

- → changes location of 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
- But still: how to find a ever changing root node?





Superblock

a

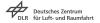




Areas







Fehlerkorrektur



