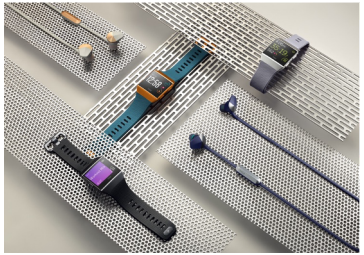


A bit of SmartFS talk | Iulian-Răzvan Mateșică

fitbit

Agenda

- What is SmartFS? How does it work?

- File system check
- Bad blocks management
- Partial mapping
- Logical defragmentation
- Sector Cache

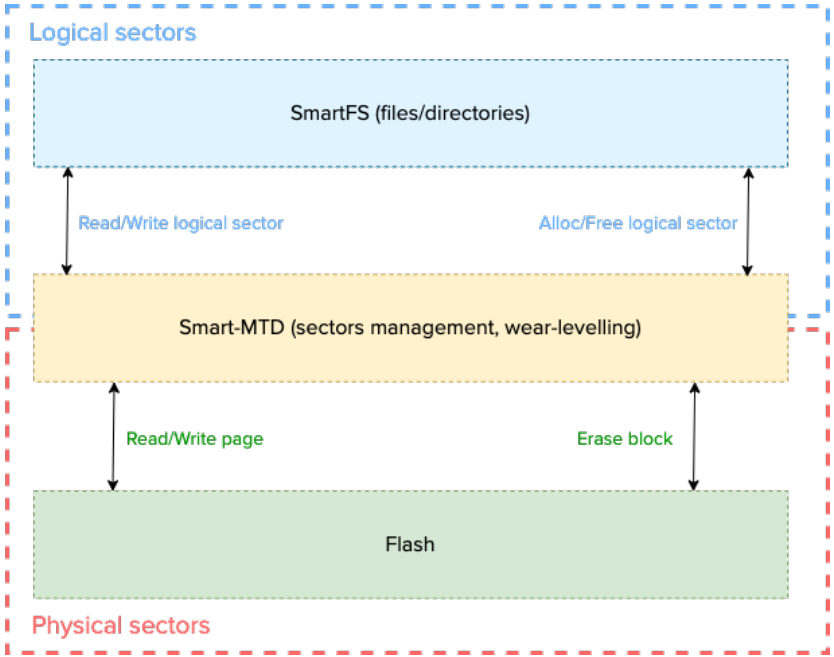
- Conclusion

Why

- Hardware constraints (limited RAM)
  (Partial Mapping)
- Reliable storage for high complexity applications
  (Bad Blocks Management)
- Flat build -> memory corruptions -> data corruptions
  (File System Check)
- Deterministic peak RAM usage
  (Sector Cache)

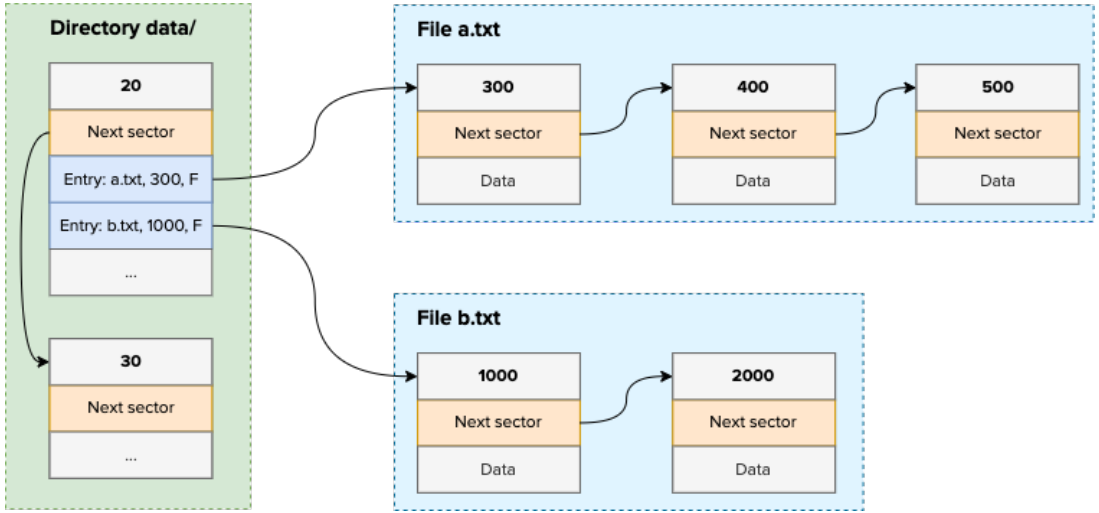
What is SmartFS? How does it work?

- SmartFS stands for Sector Mapped Allocation for Really Tiny flash
- Designed to be used with NOR flashes
- Built-in Wear Levelling
- Configurable sector size (256, 1K, 4K)
- Uses a 1:1 mapping between Logical and Physical sectors



SmartFS layer - How does it work?

- A file/directory is a linked list of logical sectors

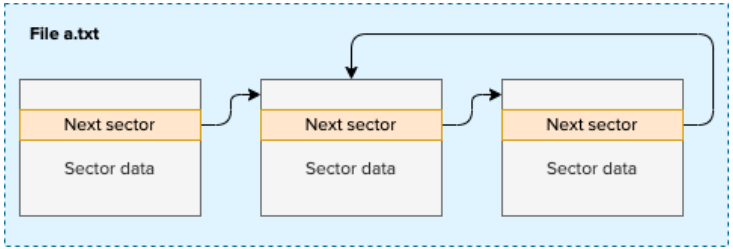


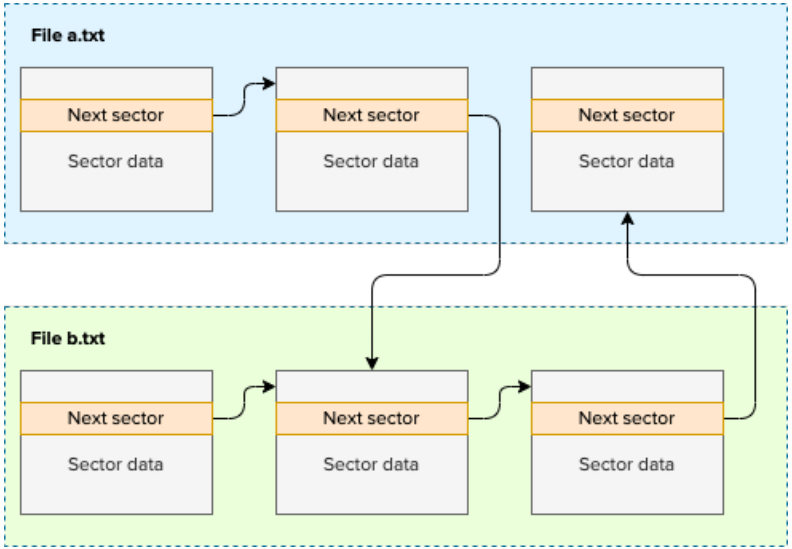
File system check

- The system always boots up with a valid file system
- Corrupted files are removed at boot-up (power loss/crashes)

It can detect:

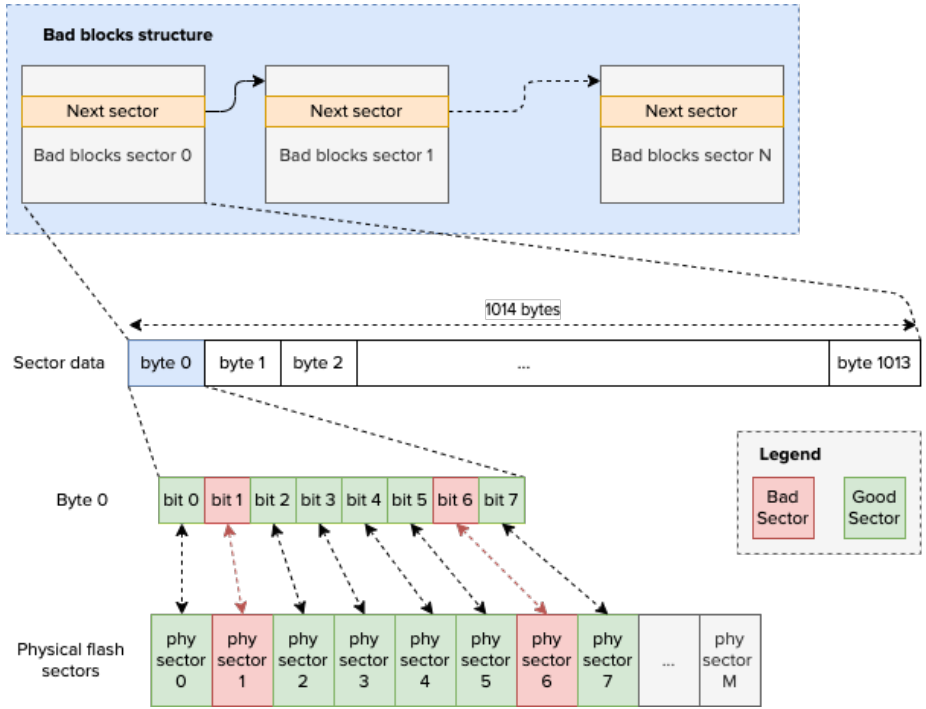
- Invalid sector and chain headers
- File loops
- Cross-file loops





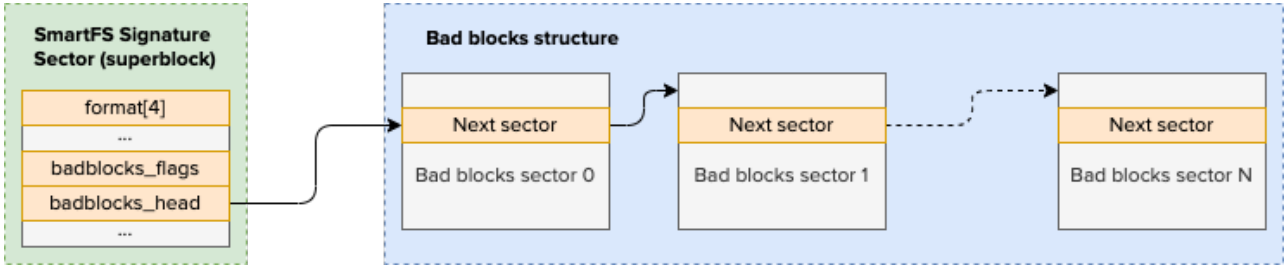
Bad blocks management - 1

- User data integrity and device reliability represent top priorities for Fitbit
- NOR flashes do go bad (or come bad from factory)
- Uses CRC for detection (read after write)



Bad blocks management - 2

- It reuses sector read/write/allocate functions
- The granularity is configurable, smallest one being 1 sector (1KB)
- The head of the list is stored in the SmartFS Signature Sector (superblock)

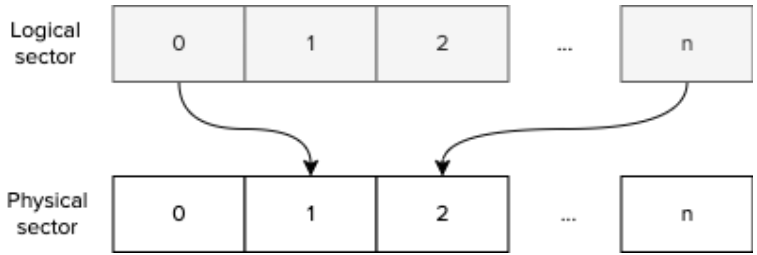


Partial mapping - 1

- 1:1 mapping between logical sectors and physical sectors
- This information is stored in an array (needs RAM)

```
uint16_t map[];
map[log sector] = phy sector;
```

- 8MB flash would need 16KB of RAM just for this map (sectors of 1KB)



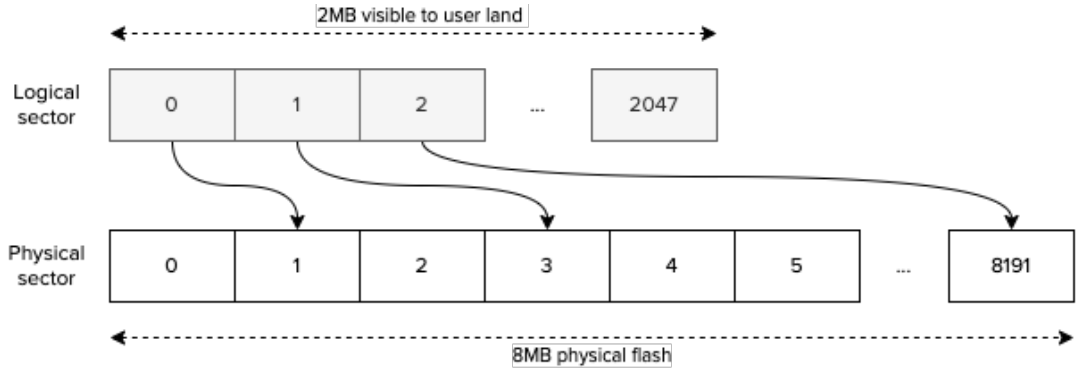
Partial mapping - 2

Why allocate memory to access all 8MB if we only use 2MB in normal usage?

(8MB are used for staging during firmware update)

Partial mapping - 3

- Possible solution: 2 partitions – 2MB and 6MB (disadvantages: increased wear and discontinuity)
- Better solution: map just enough logical sectors to cover 2MB



Partial mapping – Good & Bad

Advantages:

- Partial mapping can be enabled/disabled per partition
- Wear leveling still uses the entire 8MB space
- Only 4KB of RAM needed for logical-physical mapping
- We've saved 5% from the entire RAM (*Charge 3)
- Switching between modes at runtime:
  - **partial mapping** (2MB)
  - **full mapping** (8MB)

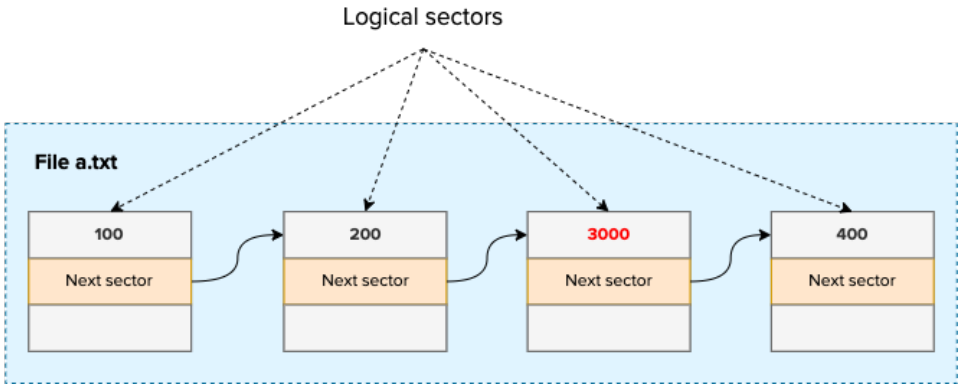
Disadvantages:

- Some files may not be accessible from partial mapping

Partial mapping – New problem

- File ‘a.txt’ is created in **full mapping**
- In **partial mapping** (2MB) only logical sectors 0…2047 can be accessed

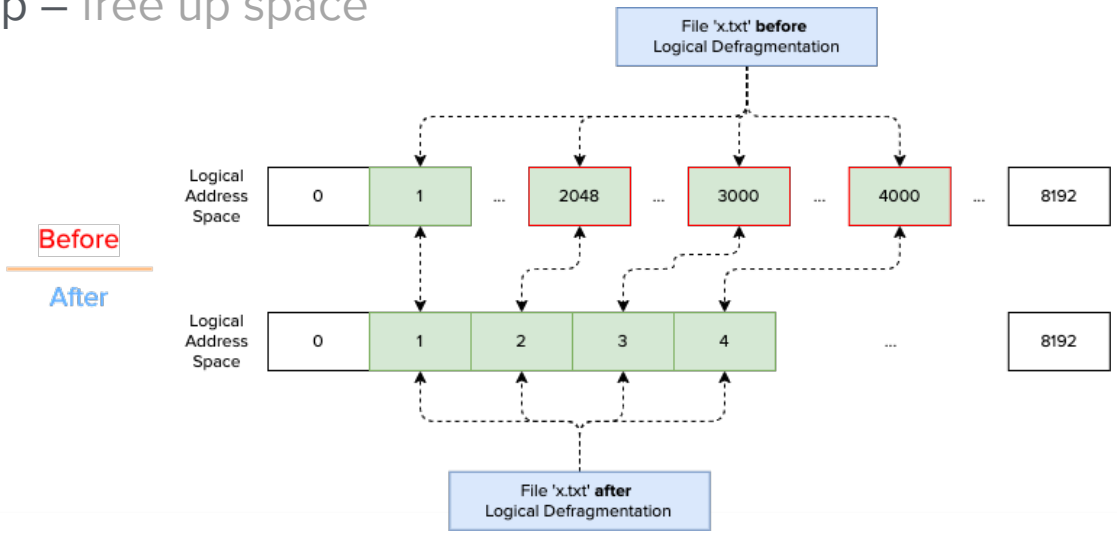
- Q: What if we need to access ‘a.txt’ in Partial Mapping?



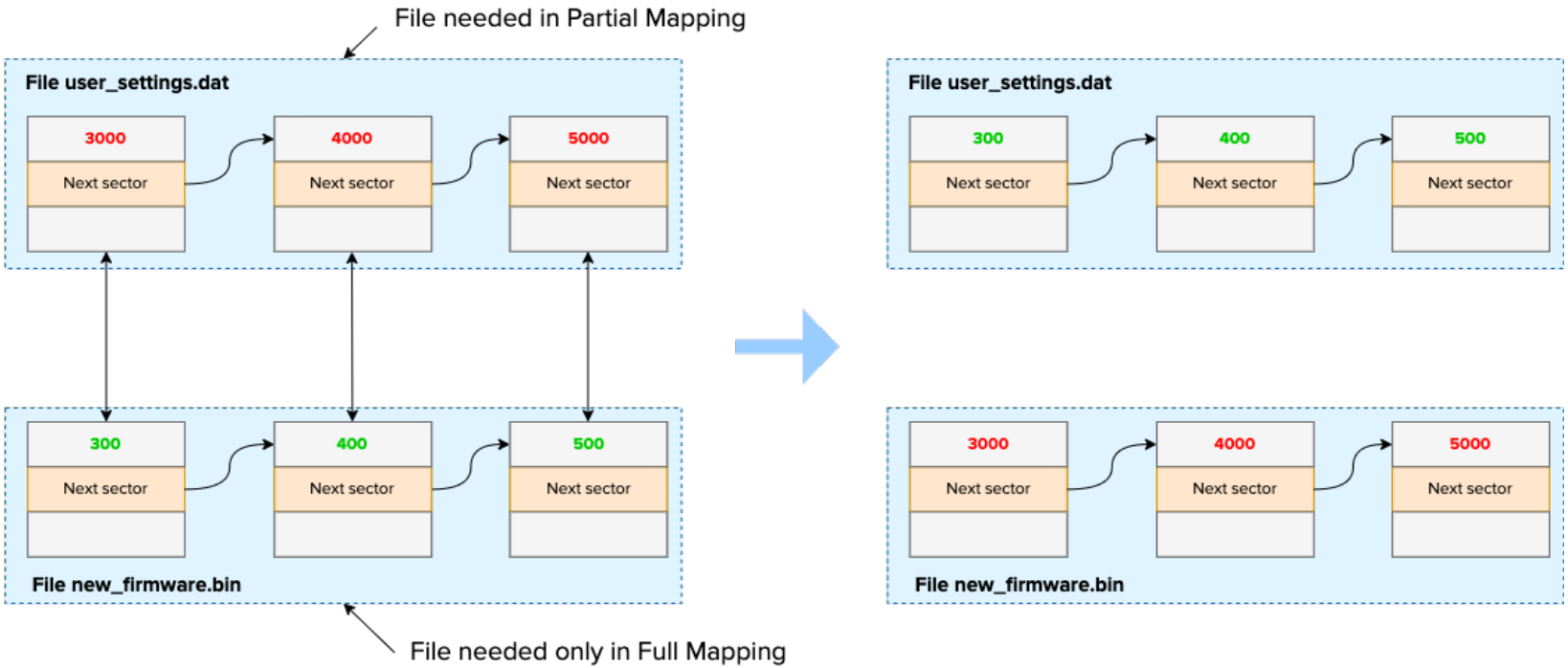
Logical defragmentation

- Replaces logical sectors >=2048 with logical sectors <2048

- Two steps:
  1. Exchange logical sectors – 1:1 logical sectors exchange
  2. Clean-up – free up space

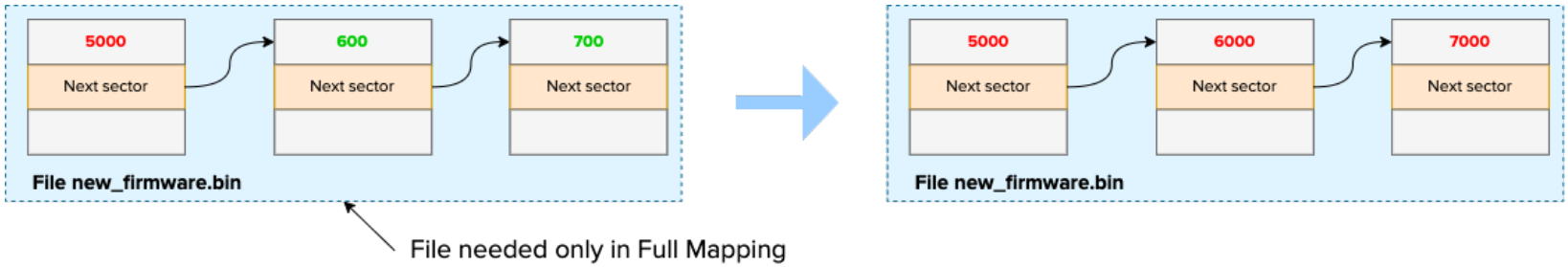


Logical defragmentation – Exchange

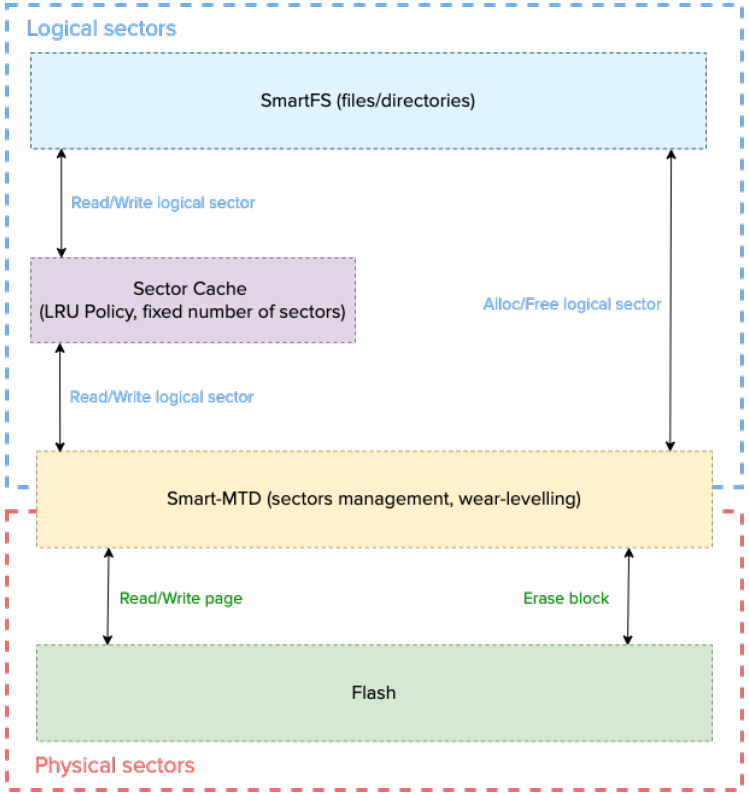


Logical defragmentation – Clean-up

- Move all files that are not needed outside of partial mapping
- **Partial mapping** range: 0 – 2047 (2MB)
- **Full mapping** range: 0 – 8191 (8MB)



Sector Cache



Why Sector Cache?

- Previously, each open file would allocate 1KB of RAM (current sector buffer)
  e.g.: 15 open files == 15KB of RAM

- Peak RAM usage couldn't be controlled

Features of Sector Cache:

- LRU Cache Policy
- Fixed RAM usage, configurable
- Also acts as a read cache
- Addresses concurrency issues

Conclusion

- Fitbit's top priorities: user data integrity, device reliability:
  - FS Check to ensure that we always boot to a sane file system
  - Bad blocks management
- RAM optimizations:
  - Partial mapping (2MB/8MB), freed up 5% from the entire RAM
  - Sector Cache – deterministic peak RAM usage
- Downside – increased complexity:
  - Logical defragmentation + clean-up
  - Increased code space usage (ROM)

THANK YOU