SSDs und LogFS

Jörn Engel

Lazybastard.org

November 12, 2009

The good

- Lower random access times
- More IO/s
- More robust against mechanical failures
- Less noise
- Lower power consumption possible

The bad

- Large blocksize
- Erase before write
- Wear-out
- Higher bit-price

The ugly

- Soft errors
- Write disturb
- Read disturb
- Leakage?

2007 Numbers

- Cheapest Tape: .10€/GB
- Cheapest 3.5" disk: .18€/GB
- Cheapest 2.5" disk: .51€/GB
- Cheapest 1.8" disk: 1.57€/GB
- Cheapest 1.0" disk: 7.50€/GB
- Cheapest flash SSD: 12.0€/GB
- Cheapest USB stick: 8.75€/GB

2009 Numbers

- Cheapest Tape: .05€/GB
- Cheapest 3.5" disk: .06€/GB
- Cheapest 2.5" disk: .14€/GB
- Cheapest 1.8" disk: .60€/GB
- Cheapest 1.0" disk: .00€/GB
- Cheapest flash SSD: 1.86€/GB
- Cheapest USB stick: 1.72€/GB

2007 predictions

- Flash capacity will double every year
- Hard disk capacity will double every two years
- Flash will be cheaper than 1.0" disks in 2008
- Flash will be cheaper than 1.8" disks in 2012
- Flash will be cheaper than 2.5" disks in 2016
- Flash will be cheaper than 3.5" disks in 2019

Large Blocksize

- OS writes smaller than blocksize
- Requires Read Modify Write

Erase before write

- Interrupted writes cause data loss
- Never overwrite old data
- New data constantly moves around

Wear-out

- Maximum lifetime depends on write load
- Local wear-out must be prevented
- Wear leveling
- Throttling

Soft errors

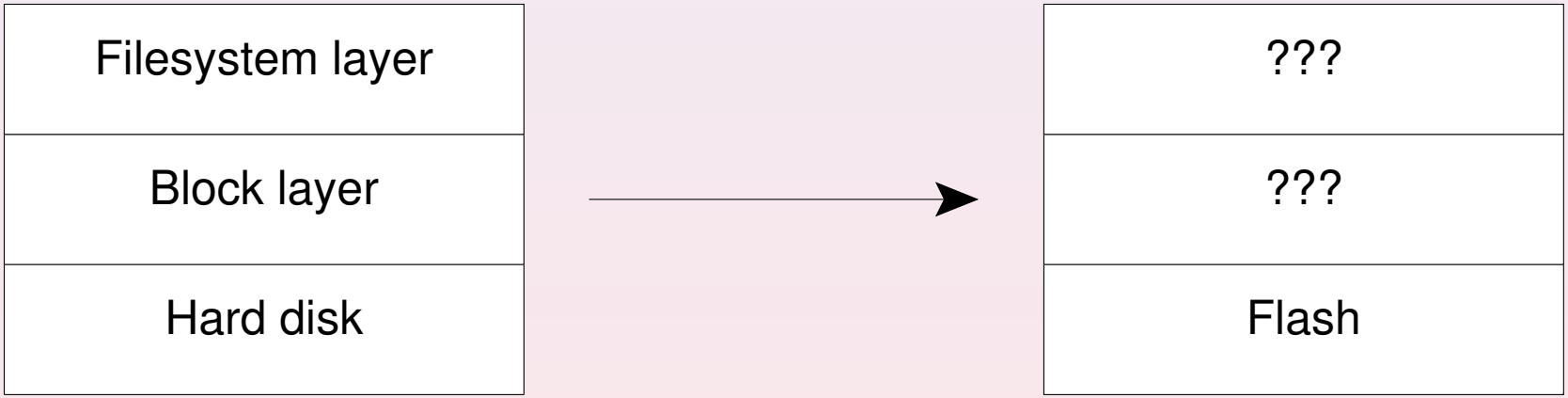
- Error correction codes necessary

Read/write disturb

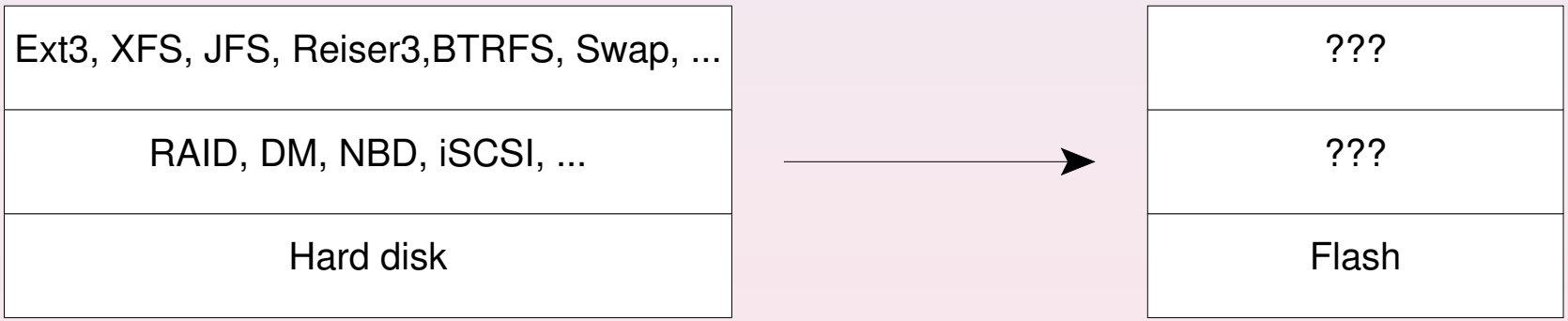
- Scrubbing necessary

Now what?

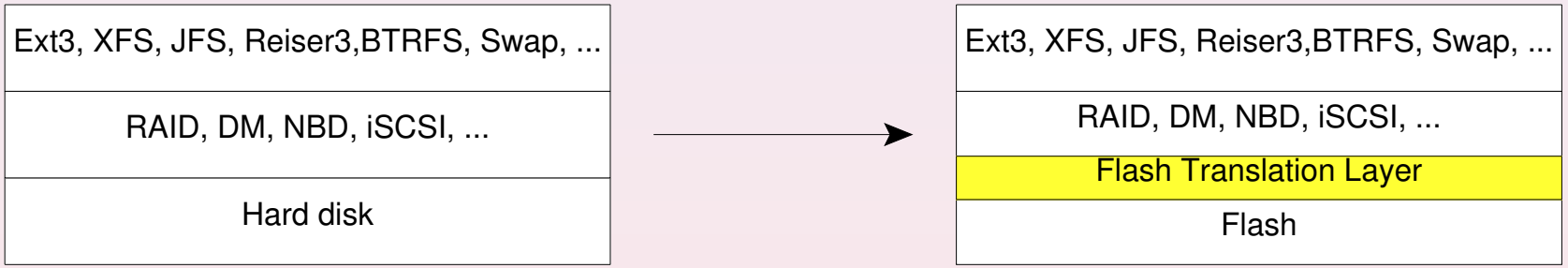
Now what?



Now what?


Ext3, XFS, JFS, Reiser3,BTRFS, Swap, ...
RAID, DM, NBD, iSCSI, ...
Hard disk
???
???
Flash

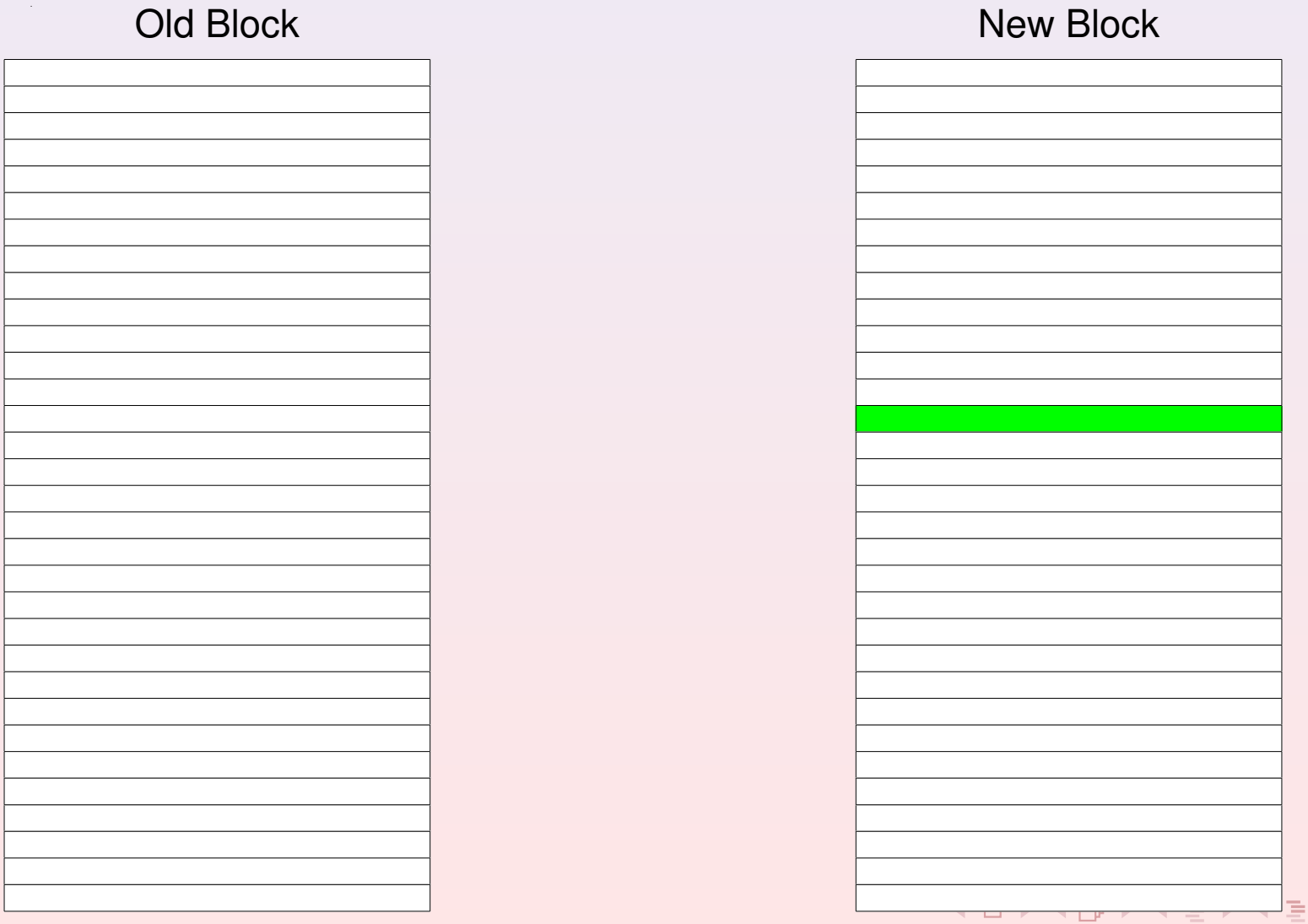
Now what?


Ext3, XFS, JFS, Reiser3,BTRFS, Swap, ...
RAID, DM, NBD, iSCSI, ...
Hard disk
Ext3, XFS, JFS, Reiser3,BTRFS, Swap, ...
RAID, DM, NBD, iSCSI, ...
Flash Translation Layer
Flash

Least smart of all approaches

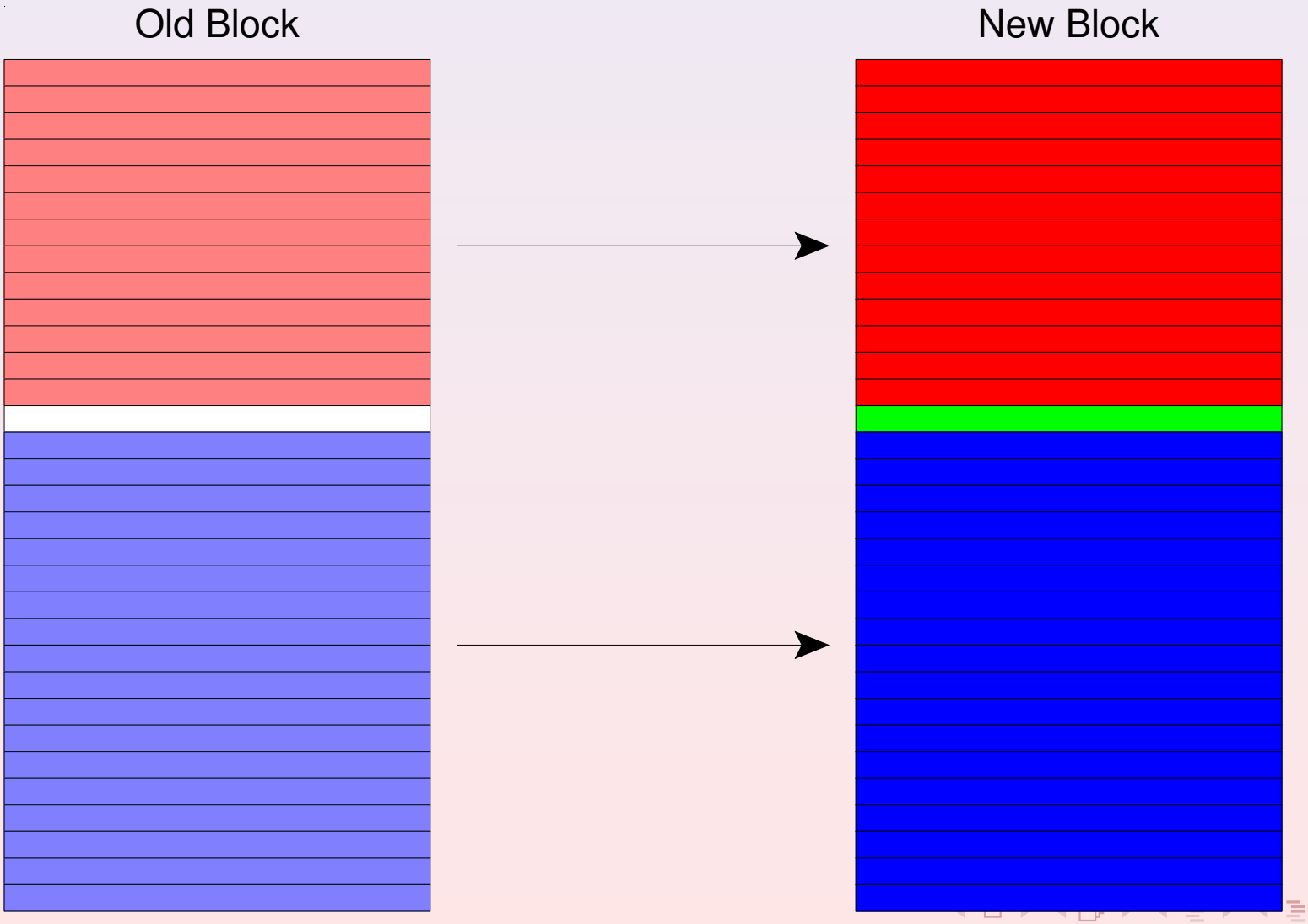
- Block mapping
- Logical->Physical indirection
- Read-Modify-Write (-Erase)
- Scan limited with areas

Writes



Writes

Writes



Least smart of all approaches

- Block mapping
- Logical->Physical indirection
- Read-Modify-Write (-Erase)
- Scan limited with areas

But at least it is documented

- Specification available for personal information

Most likely the following

- Sector mapping
- Simplified Logstructured Filesystem
- Garbage Collection
- Plenty of (bogus) Patents

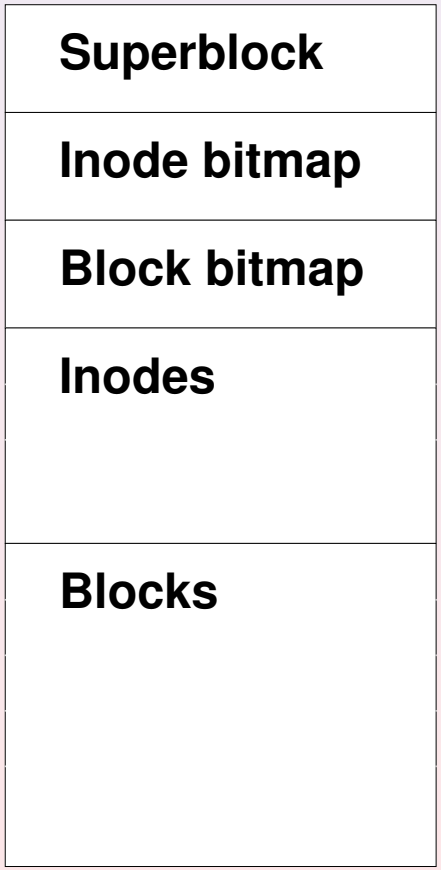
Some math

- 31,536,000 seconds per year
- 10,000
- N years life expectancy
- max writerate 1/3000N medium size

Some math

The worst-case performance of a medium with N% free space is N% of the best-case performance.

FFS Blockgroup



COW

- FFS writes blocks in-situ
- Flash prohibits in-situ writes
- Solution: Log-structured design

Segments

- Basic write unit
- Matches flash erasesize (or multiple)
- Must be completely empty before reuse (GC)

Segments

- Blocks can be compressed
- Blocks are checksummed
- Per-block rmap

Superblock

- FFS frequently updates superblocks
- Flash would wear out
- Solution: keep superblocks (mostly) read-only

Superblock

- Two superblocks at beginning and end of medium
- Usually kept read-only
- Dynamic parts moved to journal

Journal

- Circular buffer
- Newest entry wins
- Contains dynamic parts of superblock
- and more...

Inodes

- FFS writes inodes in-situ
- Flash prohibits in-situ writes
- Solution: write inodes to special file

Inode file

- Master inode stored in journal
- Contains inodes instead of blocks

Inode bitmap

- FFS writes inode bitmap in-situ
- Flash prohibits in-situ writes
- Solution: SEEK_HOLE and SEEK_DATA

SEEK_DATA

- Trivial: first valid pointer

SEEK_HOLE

- Top bit of pointer indicates hole

Block bitmap

- FFS writes block bitmap in-situ
- Flash prohibits in-situ writes
- Solution: segment file

Segment file

- 8 bytes per segment
- Contains:
  - erase count
  - valid bytes
  - level

Segment file

- Only contains segment summary
- Use rmap to verify blocks

Cow writes

- Writes change data address
- Immediately updating indirect block inefficient

Cow writes

- Each write changes segment file
- Including writes to segment file
- Segment file is notoriously outdated

Aliases

- Small updates written to journal
- (addr, data) tupel

Garbage collection

- Only completely empty segment usable
- Solution: Garbage collection

GC deadlock

- GC moves data
- Pointer changes consume space
- Solution: Levels

Levels

- Data blocks on level 0
- Pointers on higher levels than target

Wear-out

- Not all segments written equally
- Some will wear out sooner than others
- Solution: Wear leveling

Wear leveling

- Erase count stored in segment file
- GC forced for stale blocks

Journal wear-out

- Journal written relatively often
- Solution: move journal
- Requires writing superblocks

Detect data corruption

- Checksums on everything
- But no second copy

Fast fsync

- Don't wait for other files
- Don't wait for transactions
- Don't wait - full stop

Device-agnostic

- Works on MTD
- Works on block devices