Writing your own file system is easier than you think

Andrzej Jackowski, MSST 2019

9LivesData

- I’ve been working for 9LivesData since 2014
    - > Advanced software R&D since 2008
    - > 75 developers / scientist with MSc/Phd in CS
    - > Clients in the USA and Japan
    - > Millions of C/C++ LOCs
    - > Products used by thousands of corporations worldwide
    - > Specializations: file systems, software defined storage, scalable distributed systems, deduplication

- I’m also a PhD student at University of Warsaw

- In 9LD, one of our projects is a backend of HYDRAstor

> distributed secondary storage
> global deduplication
> massive linear scalability from 1 to 165 nodes
> capacity up to 11.88PB (158.4PB effective)
> high performance (up to 5.2PB / hr)
> erasure-coding, self-healing
> 5th generation on the market
> Veritas NetBackup™ OpenStorage integration
> deduplication client fs

- More details available in our publications

>"HYDRAstor – A Scalable Secondary Storage"
The 7th USENIX Conference on File and Storage Technologies (FAST '09)
San Francisco, California, USA, February 2009

> "A High-Throughput File System for the HYDRAstor Content-Addressable Storage System"
The 8th USENIX Conference on File and Storage Technologies (FAST '10),
San Jose, California, USA, February 2010

> "Concurrent Deletion in a Distributed Content-Addressable Storage System with Global Deduplication"
11th USENIX Conference on File and Storage Technologies (FAST '13)
San Jose, California, USA, February 2013

Why we needed a custom file system?

- We needed a file system in the deepest level of HYDRAstor

- We started with *ext3* – most common fs at that time

- Very high fragmentation, even if disk wasn't full

- Out of space protection was very tricky

  E.g. *statfs* show num of blocks, but a single block append can use 1-4 blocks; directory size changes on file creation

- Double journaling affected performance

  We needed own journal to perform transactions on multiple disks

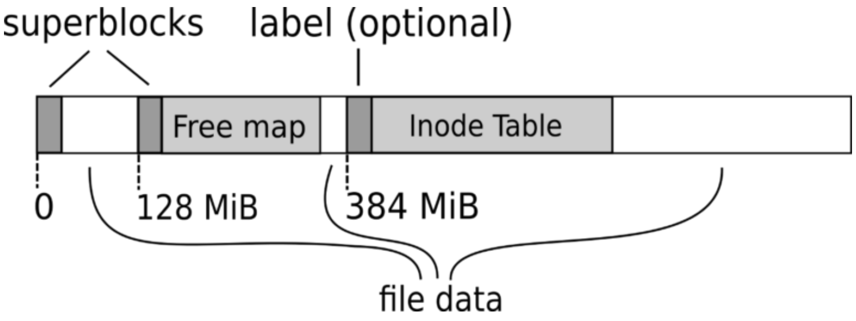
There were a lot of different problems

- Many others performance issues
  > Inefficient fsync / O_SYNC
  > Inode lock (only one outstanding operation on each inode)
  > io_submit blocked thread sometimes
  > Needless features affecting critical path
  (e.g. small files support; directories)

- General lack of control
  > Difficult integration with our resource manager
  > We needed to develop custom features
  (e.g. on-demand shredding of data)

We kept design of our fs as simple as possible

- Only two data structures (and two superblocks + optional label)
  > free map – keeps information about unused disk blocks
  > inode table – keeps simplified inodes, we don't even need filenames

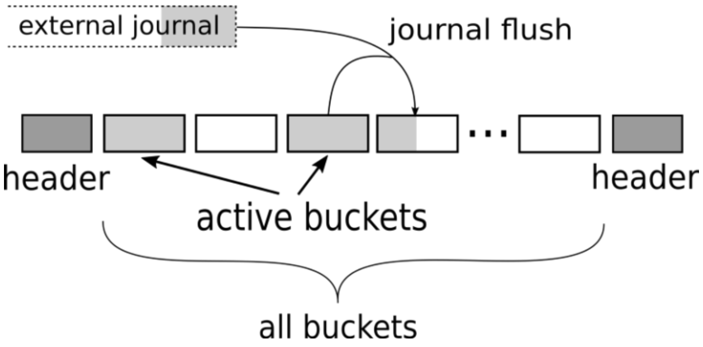


Disk layout of our file system

- Allocation algorithm that solved fragmentation issue

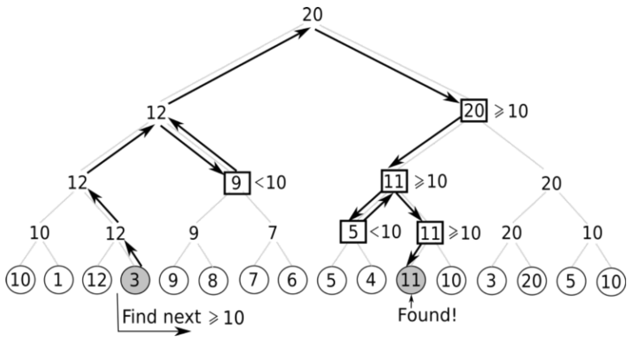
1. Find the closest extent after the current "allocation pointer" of size at least 1024 blocks, within initial disk part of size = total occupied space * 120%.
2. If it fails, try the same within the whole partition.
3. Then seek the closest free extent within the whole partition, of minimal size 256 blocks, then 32 block and finally 1 block.
4. Once you've found some extent, remember to update the allocation pointer to its end.

Inode table design

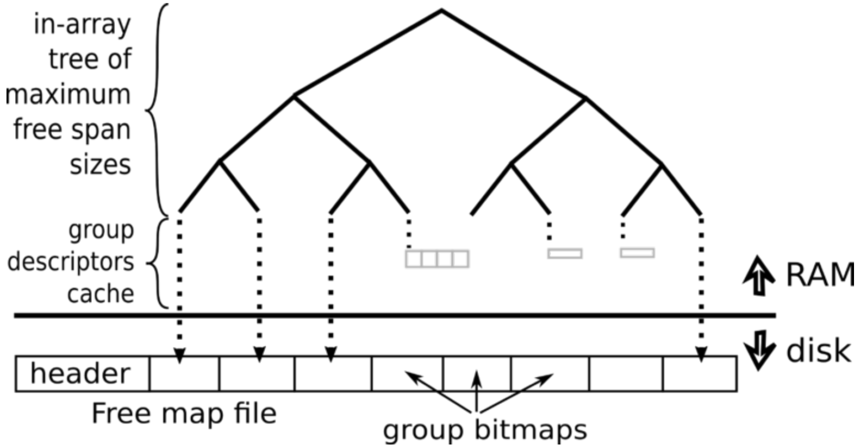


Inode table design

FreeMap design



Logarithmic search for the closest next group with a given minimum free extent size



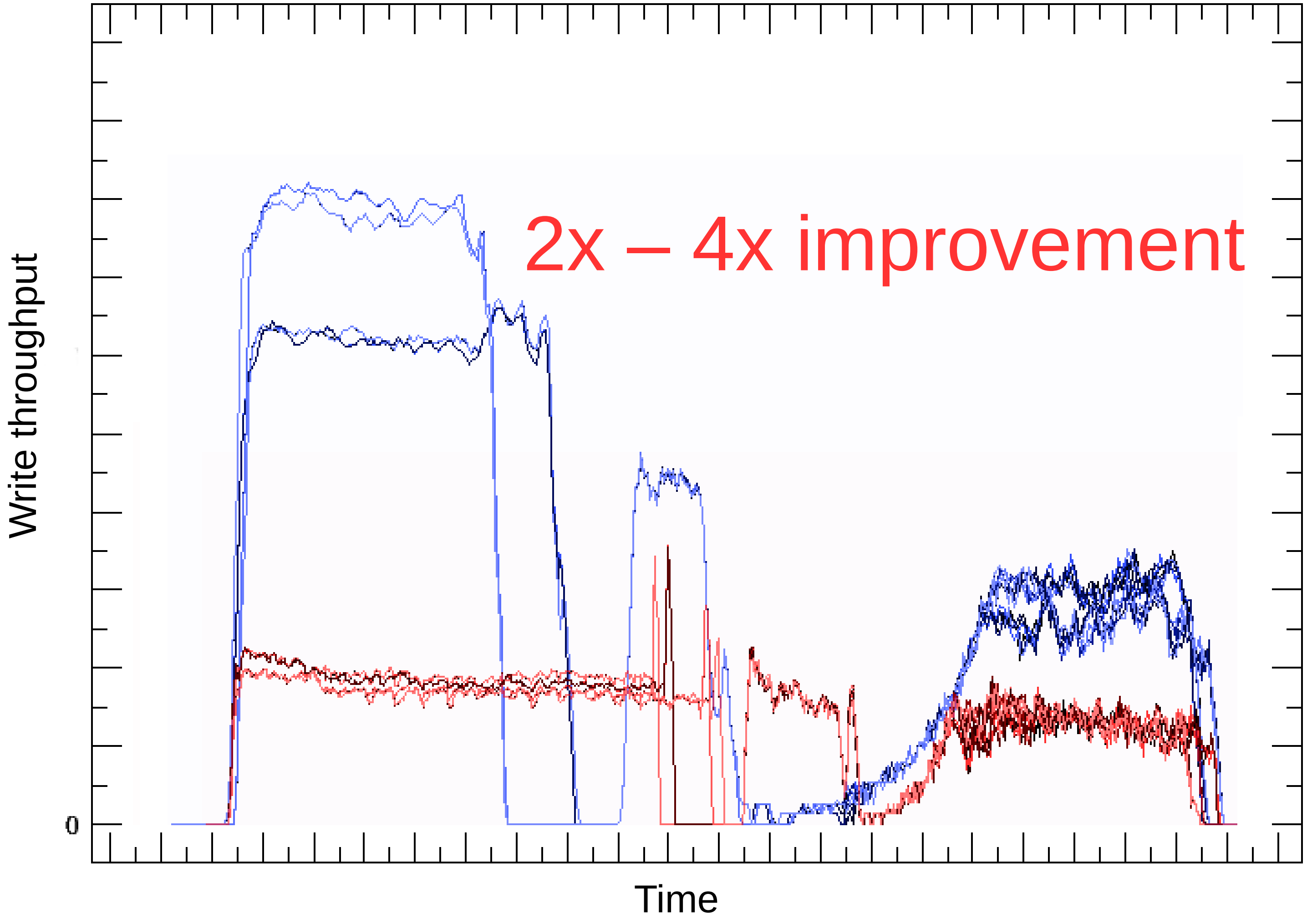
High-level design of Free map

Explained with details by Marian Kędzierski in a blog post:

http://9livesdata.com/writing-your-own-file-system-is-easier-than-you-think/

Performance evaluation

Write throughput during different loads



2x – 4x improvement

- Our filesystem
- EXT3

The After Years

- Our file system is easy to develop, understand and maintain
  Man-hours spent on creating file system from scratch paid off

- Performance is great, but now the gap is smaller
  Especially XFS caught up in our recent benchmarks

- *Ceph* adopted similar strategy by introducing *BlueStore*

Takeaway

Sometimes it is easier to implement dedicated solution from scratch than adjust a complex general-purpose code

Thanks for watching!

Any feedback is appreciated!!!

✉ jackowski@9livesdata.com

in linkedin.com/in/ajackowski/

Visit our blog for more details

www.9livesdata.com/blog/