ETH zürich

Shifting a Lustre filesystem to flash: challenges and caveats

Diego Moreno

LAD 2021

Sept, 30th 2021

Agenda

- Lustre @ ETH Zürich
- Flash storage 101
- Data life cycle as a mandatory duty
- fstrim or a quarter of performance
- Final notes

ETH zürich

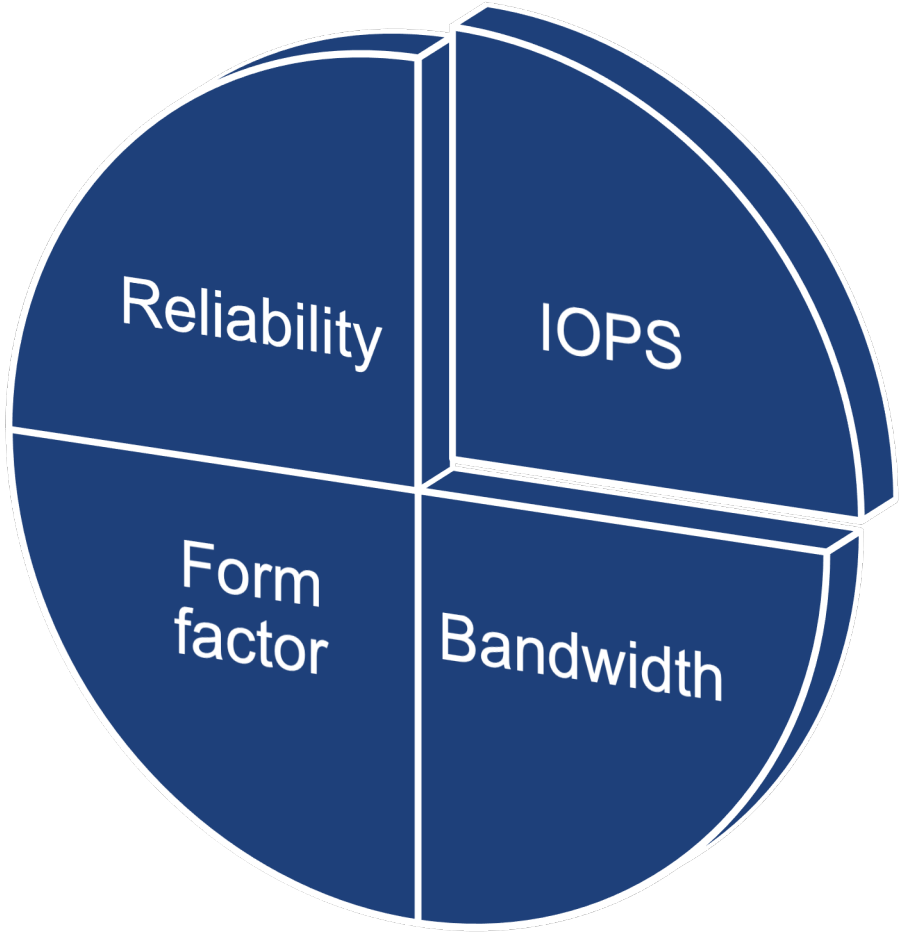
Lustre @ ETH Zürich

- 2 sites, 3 Lustre filesystems:
  - **Leonhard** @ Zürich (3.2 PiB - Lustre 2.10)
    - Multi-tenancy cluster (presented @ LAD'19)
  - **Euler** @ Lugano (CSCS):
    - 2.4 PiB – Capacity filesystem – Lustre 2.12 – HDDs storage
    - **240 TiB – Scratch filesystem – Lustre 2.12 – All NVRAM**

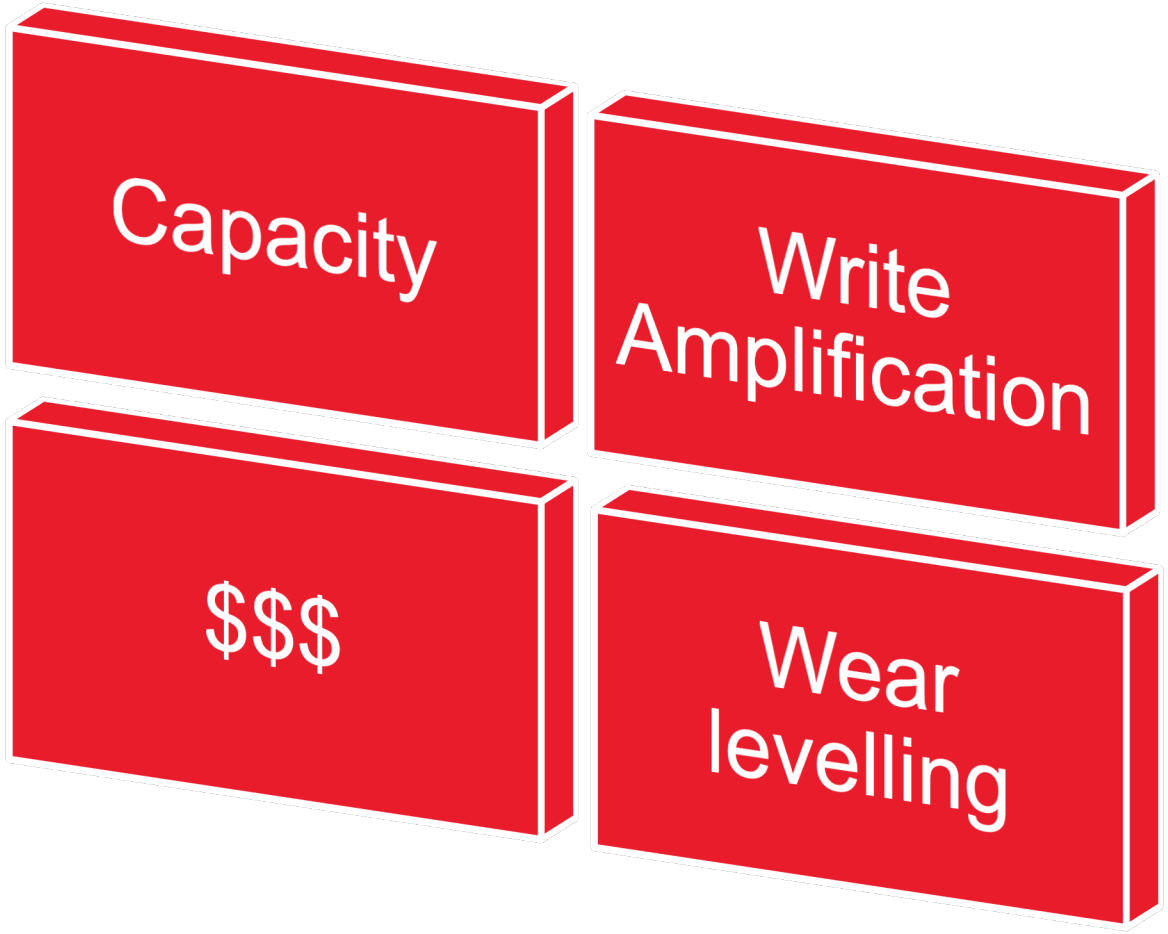
All NVRAM Lustre Scratch

- 240 TiB – Scratch filesystem – Lustre 2.12 – All NVRAM
- Based on DDN's ES400NV:
  - 24 x 16TB NVRAM
  - 8 OSTs + 4 MDTs on 4 hybrid OSS/MDS VMs

Flash storage: disruptive advantages



Flash storage: and a few caveats

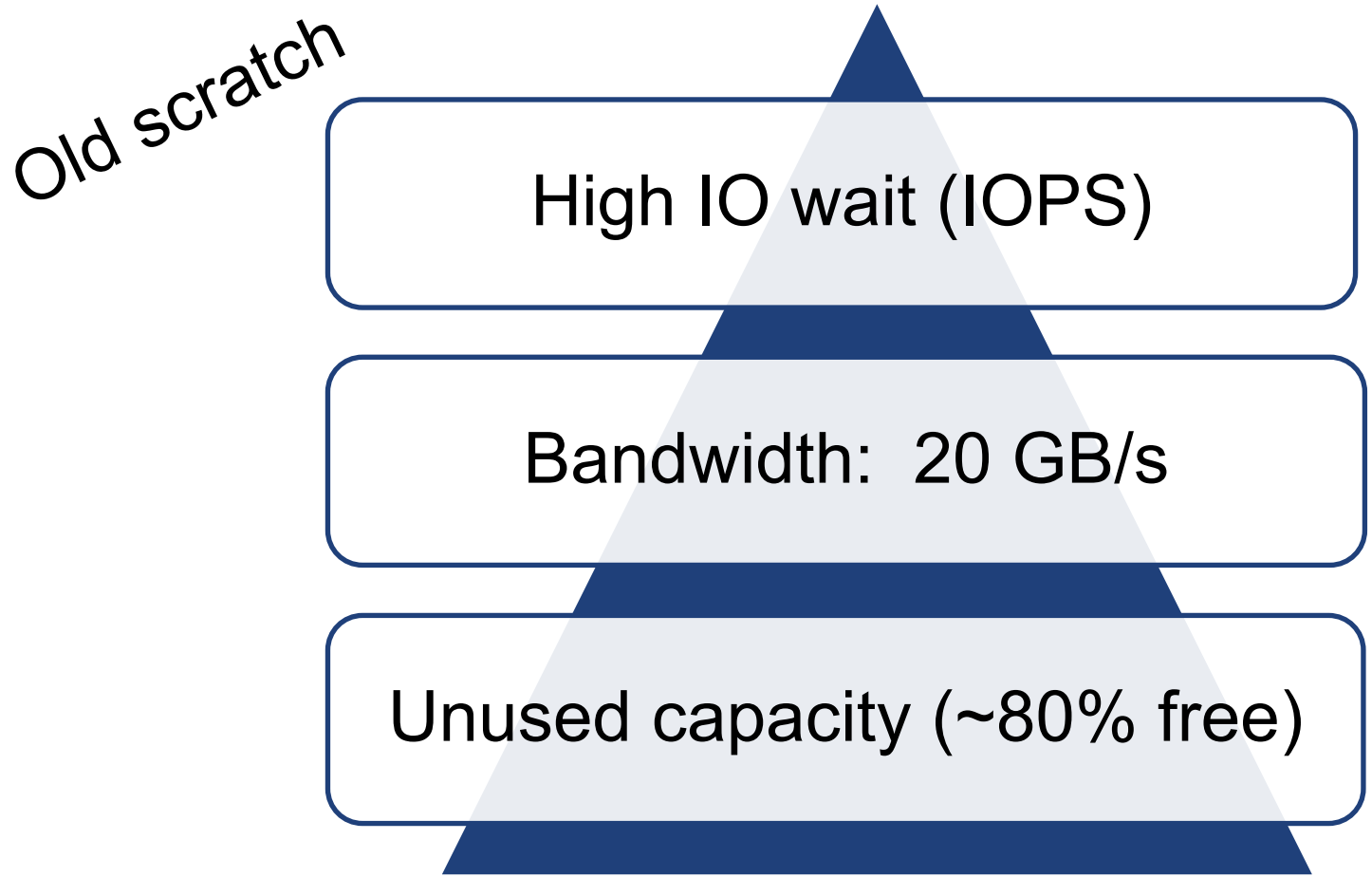


Flash storage: this talk focus on these caveats

The rules of our scratch filesystem

- Soft quota: 2 TB
- Hard quota: 2.5 TB
- Grace period: 7 days
- Purge files with atime > 15d

The evolution of our scratch filesystem: the idea



"Flash should improve all areas with just less capacity"

The evolution of our scratch filesystem: first month

"Just purge scratch from time to time (we have lots of free capacity). Cross your fingers and fine tuning to handle periods of high IOPS load."

Previous scratch

New flash scratch

"Scratch is down, please fix it ASAP"

First month with a flash scratch filesystem

IOR write bandwidth per OST	Current: 3 to 100 MB/s	Reference: 10 GB/s
High %sys on servers	Low %iowait	Low load
No purges triggered yet	Manually run by admins	Capacity still at 80%

Datalife cycle as a mandatory duty

Diagnostic (credits to *J. Peyrard*, DDN): ldiskfs block allocator problem

```
for i in /sys/fs/ldiskfs/*/mb_c3_threshold ; do echo 40 > $i ; done
```

- Already discussed at LAD'19 (ldiskfs block allocator and aged file system, *A. Blagodarenko*)
- Defaults were simply not good for us => tune mb_c*_threshold on the OSTs
- But, above all:

*Trigger nightly automatic scratch purges to keep the filesystem **capacity** usage low*

ldiskfs tuning, and then what?

- With the ldiskfs block allocator tuning and the nightly scratch purge the system is responsive
- Let's check the IOR bandwidth for the whole system:

Benchmark reference on perfect conditions

Write	Read
36 GB/s	45 GB/s

IOR after "fixing" our ldiskfs block allocator issue

Write	Read
9.5 GB/s	38 GB/s

Low write performance - Context

Users removing data

- It's scratch
- Quota
- The purge policy encourages user data management

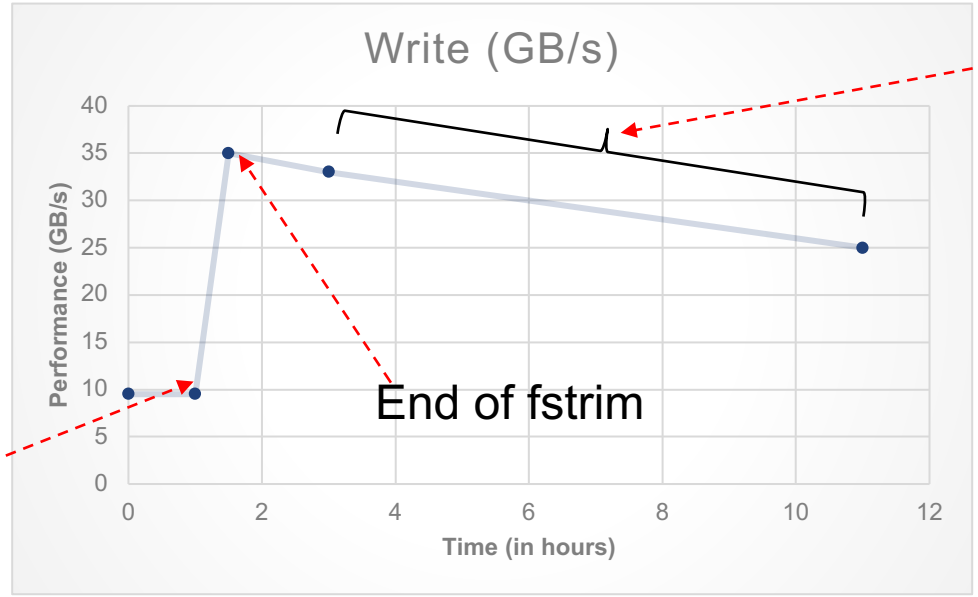
Admins removing data

- In the last 30 days: 161 TB were purged
- 66% of the fs capacity

Perfect write amplification scenario

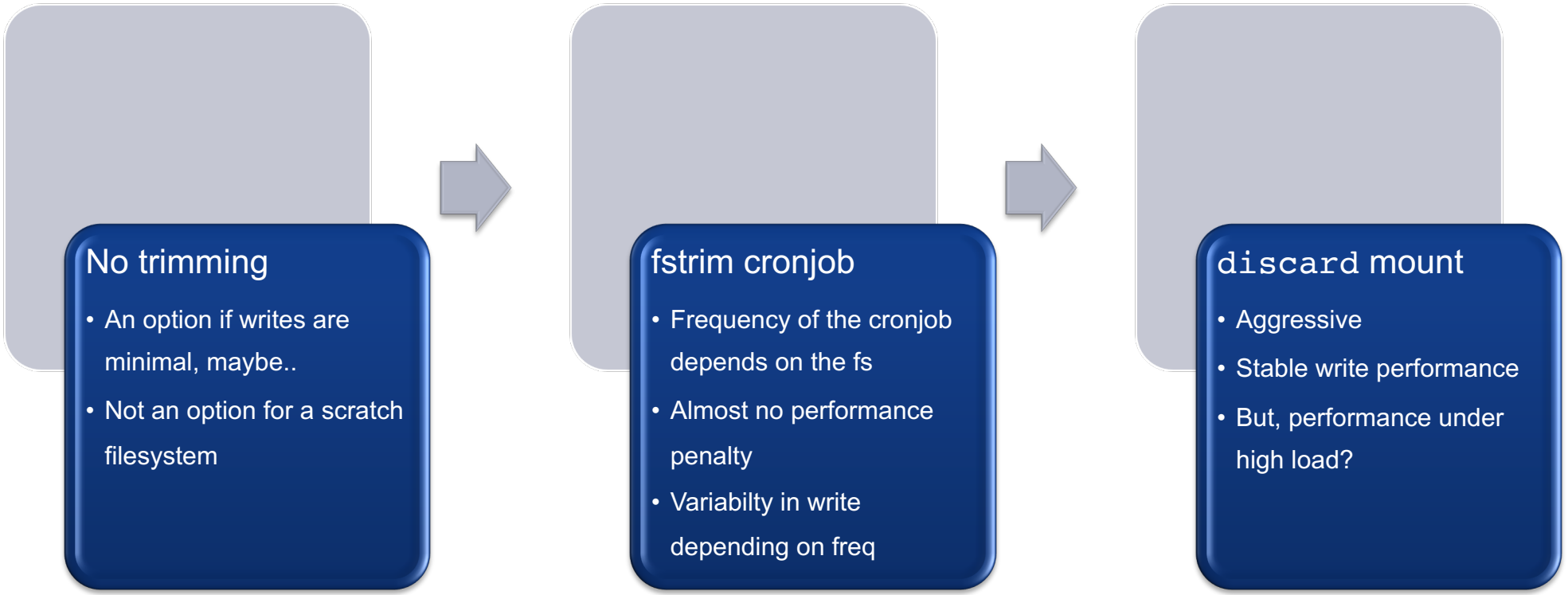
Upgrade your flash storage every day: `fstrim` or `-o discard`

- Already covered on other LAD talks (e.g: S. Ihara and G. Delbary at LAD'19)
- Write performance before, during and after `fstrim`:




```
# fstrim -v /lustre/euscrat/ost0004 ; fstrim -v /lustre/euscrat/ost0005
/lustre/euscrat/ost0004: 29.9 TiB (32849356800000 bytes) trimmed
/lustre/euscrat/ost0005: 29.9 TiB (32889314746368 bytes) trimmed
```

Upgrade your flash storage every day: `fstrim` or `-o discard`



Final notes

- Take special care of ldiskfs tuning with a flash filesystem
- Be strict with your data management to keep data capacity at good levels
- For highly changing data, trimming might be essential
- Open questions:
  - fstrim (manual trimming) vs `-o discard` mount option
  - Wear levelling on a scratch filesystem on long term (after 6 months: 99% life remaining)