The current status of the adoption of ZFS* as backend file system for Lustre*: an early evaluation

Gabriele Paciucci

EMEA Solution Architect

Outline

The goal of this presentation is to update the current status of the ZFS and Lustre* implementation in a Sys Admin prospective. Johann will cover the development status.

- Benefits of ZFS and Lustre* implementation
- Performance
  - Sequential I/O
  - Metadata
- Reliability
- Availability

ZFS benefits

ZFS is a robust, scalable file system with features not available in other file systems today. ZFS can enable cheaper storage solution for Lustre* and increase the reliability of data on the next generation of fat HDDs.

- **Hybrid Storage Pool** (ARC+L2ARC+ZIL)
- **Copy on Write (COW)**
- **Checksum**
- **Always consistent on disk**
- **Snapshot and replication**
- **Resilvering**
- **Manageability**
- **Compression and deduplication**

How do Lustre* and ZFS interact ?

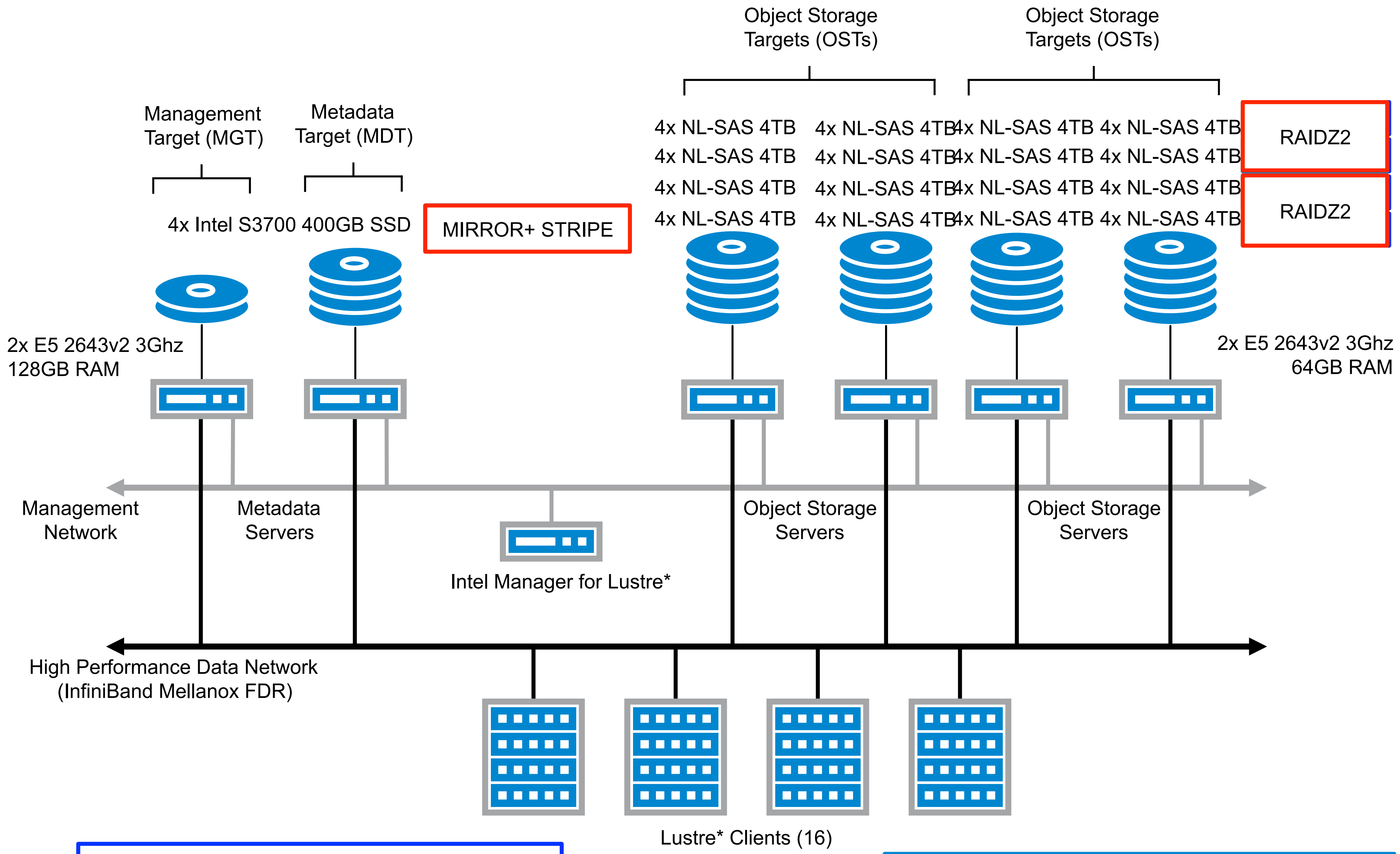
Lustre* depends on the "ZFS on Linux" implementation of ZFS

- Lustre* targets run on a local file system on Lustre* servers. Object Storage Device (OSD) layer supported are:
  - ldiskfs (EXT4) is the commonly used driver
  - ZFS is the 2nd use of the OSD layer based on OpenZFS implementation
- Lustre* targets can be different types (hybrid ldiskfs/ZFS is possible)
- Lustre* Clients are unaffected by the choice of OSD file system
- Lustre* ZFS is functional since Lustre* 2.4

Our LAB and partner's PoC

There is a strong interest in our partners and end user to test the Lustre* + ZFS combination.

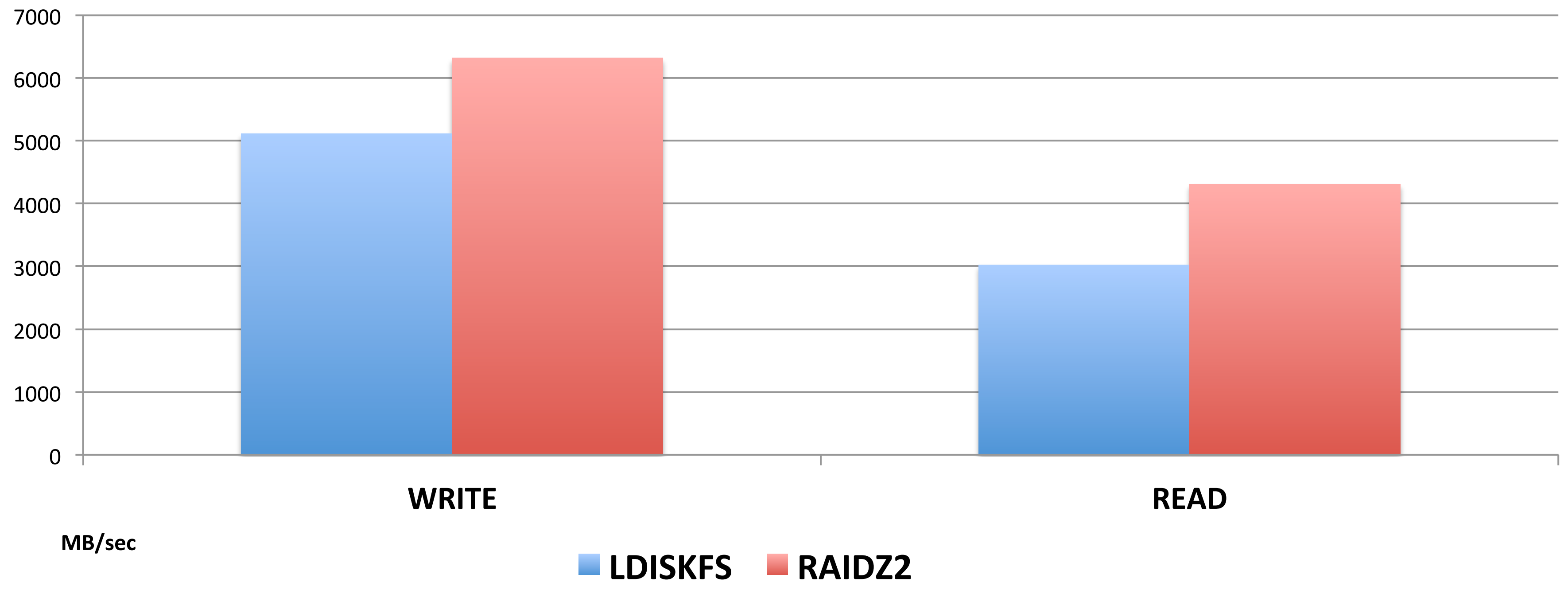
- This work try to summarize the results from benchmarks run on the Intel's HPC LAB in Swindon (UK) managed by Jamie Wilcox and early results from partner's proof-of-concept
- Intel is actively improve the ZFS + Lustre* integration (see Johann deck)
- Intel Enterprise Edition for Linux will support ZFS in Q4 2014



- Intel RAID cards for LDISKFS
- RAID-Z2 for ZFS

Lustre* server version 2.6.50
ZFS version 0.6.3
IEEL 2.0 as Lustre* client based on 2.5

Sequential I/O – LDISKFS vs RAID-Z2



- IOR results from 384 threads and an aggregate file size of 3TB
- Theoretical performance is 6.7 GB/sec (48x 4TB NL-SAS at 140MB/sec)
- 16x OSTs for ldiskfs using Intel RAID cards and RAID5 with 4 HDD
- 8x OSTs for RAID-Z2 using 8 HDD
- ldiskfs pays the penalty of fragmentation and misalignment confirmed by brw_stats for this uncommon configuration.
- ZFS's write performance is in line with the expectations and take advantage of COW.
- ZFS's read is limited by the small "record size" and concurrency

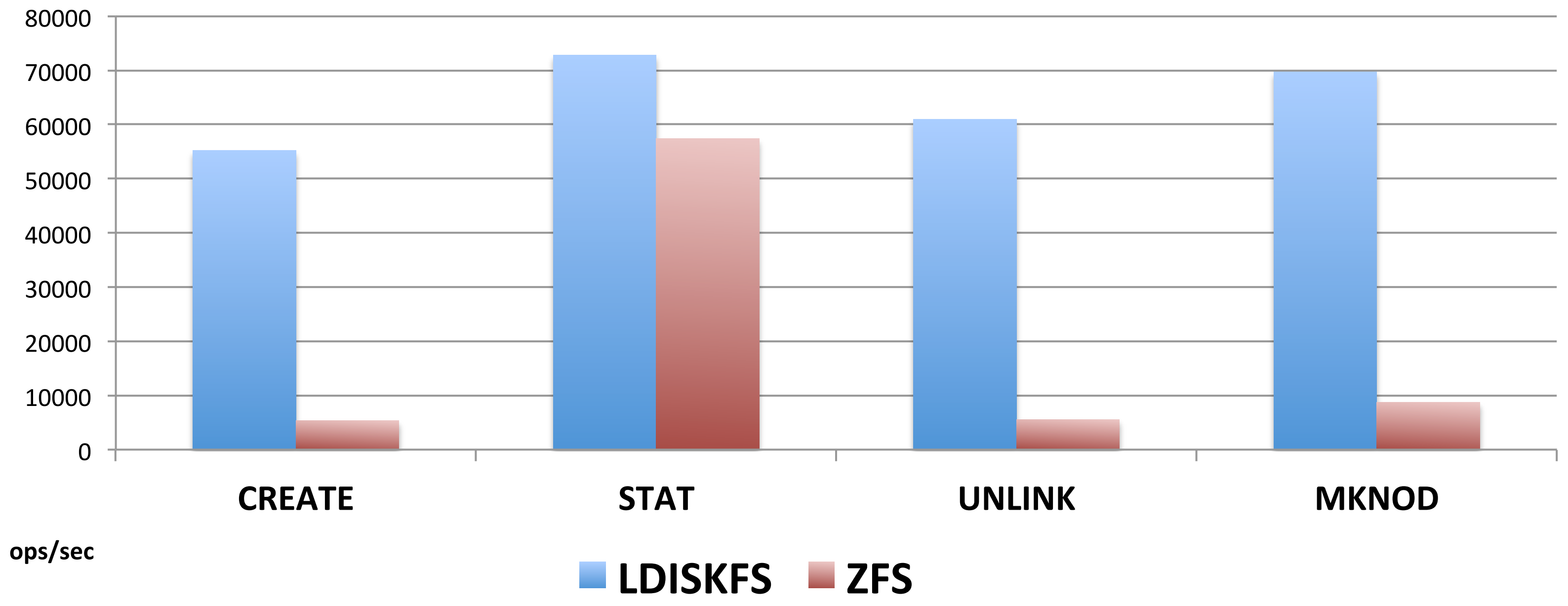
Sequential I/O – Where to improve?

- CPU frequency on OSS matter. Accurate design is necessary (+3Ghz)
- Increasing the record size to 16M. A record size of 128K doesn't fit Lustre's workload!
- Potentially the FIEMAP work on the Network Request Scheduler (NRS) can efficiently schedule a large number of requests better than what the ZFS code can do itself.
- Increase the number of concurrent reads in the OSD-ZFS layer
- Optimize ZFS I/O scheduler for Lustre*

Parameter	Default MIN	Default MAX
zfs vdev sync READ	10	10
zfs vdev sync WRITE	10	10
zfs vdev async READ	3	1
zfs vdev async WRITE	10	1
zfs vdev resilvering scrubbing	2	1
zfs vdev scheduler	noop	

* Other names and brands may be claimed as the property of others.

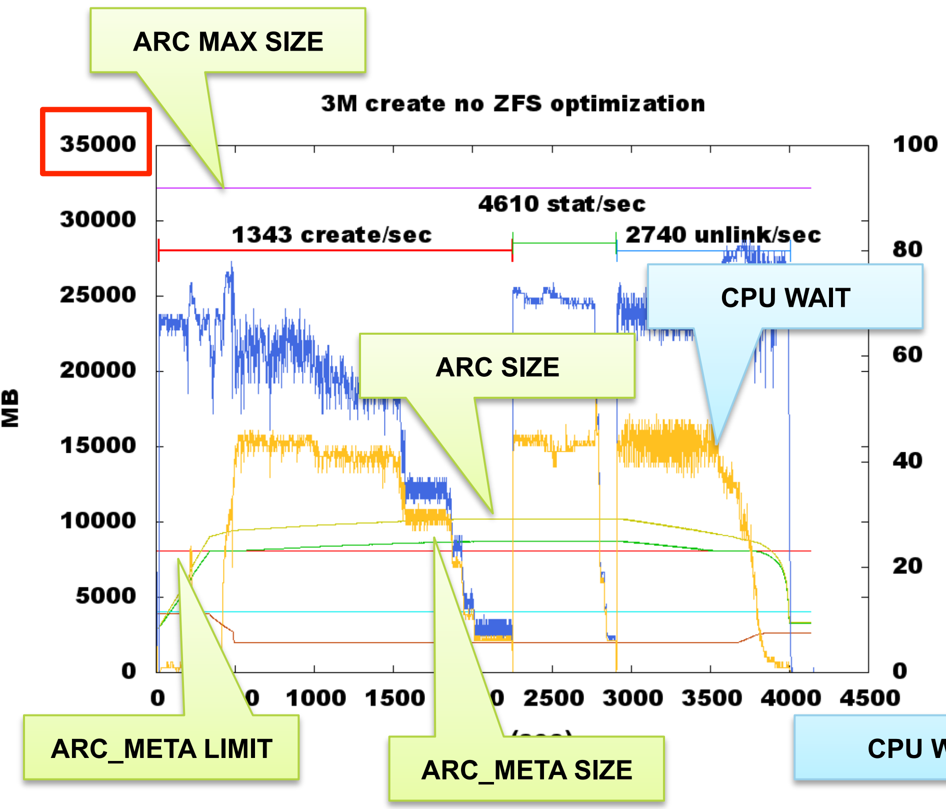
Metadata – LDISKFS vs ZFS

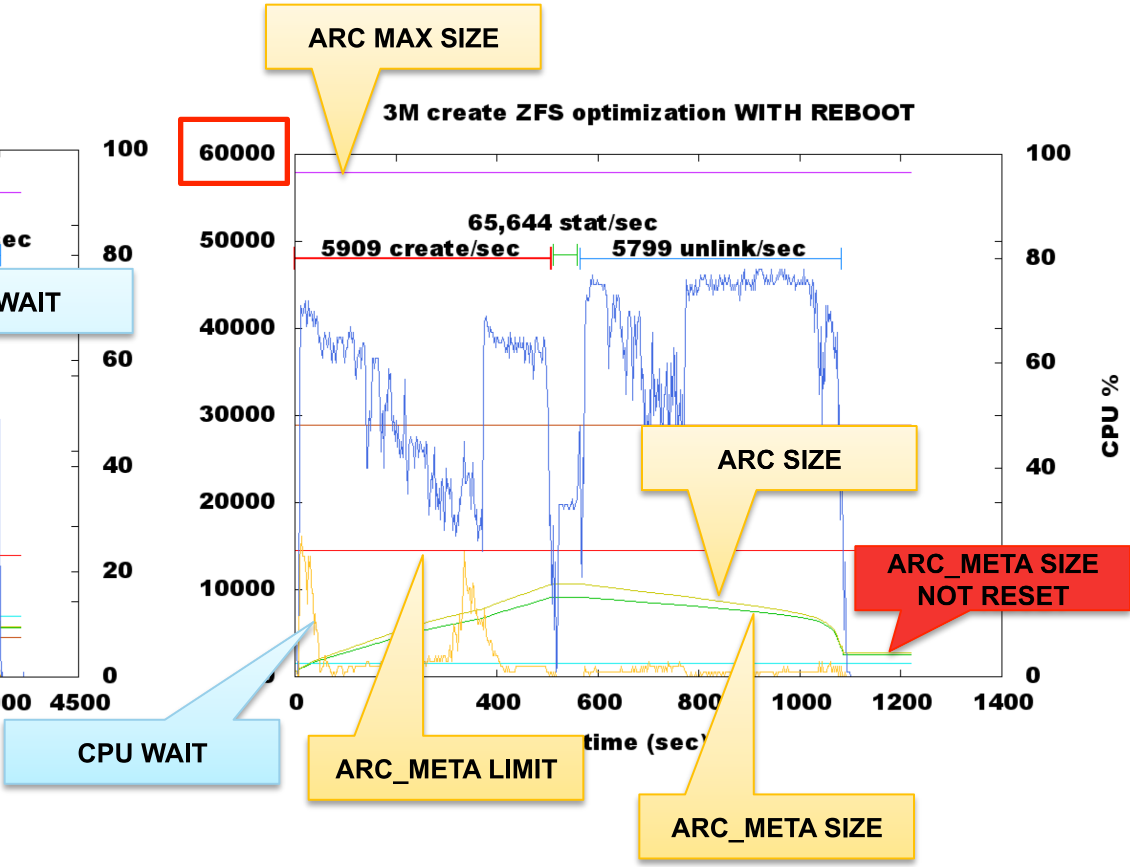


The Metadata performance is comparable only on stats

- MDSRATE results from 32 threads, 3.2M files in 32 dirs
- ldiskfs using Intel RAID card with FastPath technology. RAID10 array with 4x Intel S3700 SSD 400GB
- ZFS (0.6.2) using a striped mirrored vdev with 4 Intel S3700 SSD 400GB

Metadata – ZFS's Memory Management





It's critical tune the ZFS's ARC_META_LIMIT parameter on the MDS in order to achieve decent results.

Metadata – Where to improve ?

A lot of work is needed to improve the metadata performance.

- Increase ZAP indirect and leaf block size
- Better ARC_META memory management for Lustre’s workload
- Improve ZAP caching

Compared to ldiskfs, ZFS has some benefits:

- Not limited to 4 billions of Lustre’s inode
- Better scalability in the code. ldiskfs metadata performance is limited by the size of the directory: leaf block updates will become more random as the directory gets larger.

Reliability – Is RAID dead?

- MTTDL is safe using RAID with double parity
- Hard Error Rate (HER): when the disk can’t read a sector, disk failed and causes a RAID rebuild
- Probability of a second HER during rebuild is high with larger NL-SAS
- Because there are cables and electronics, storage channels are susceptible to Silent Data Corruption (SDC)

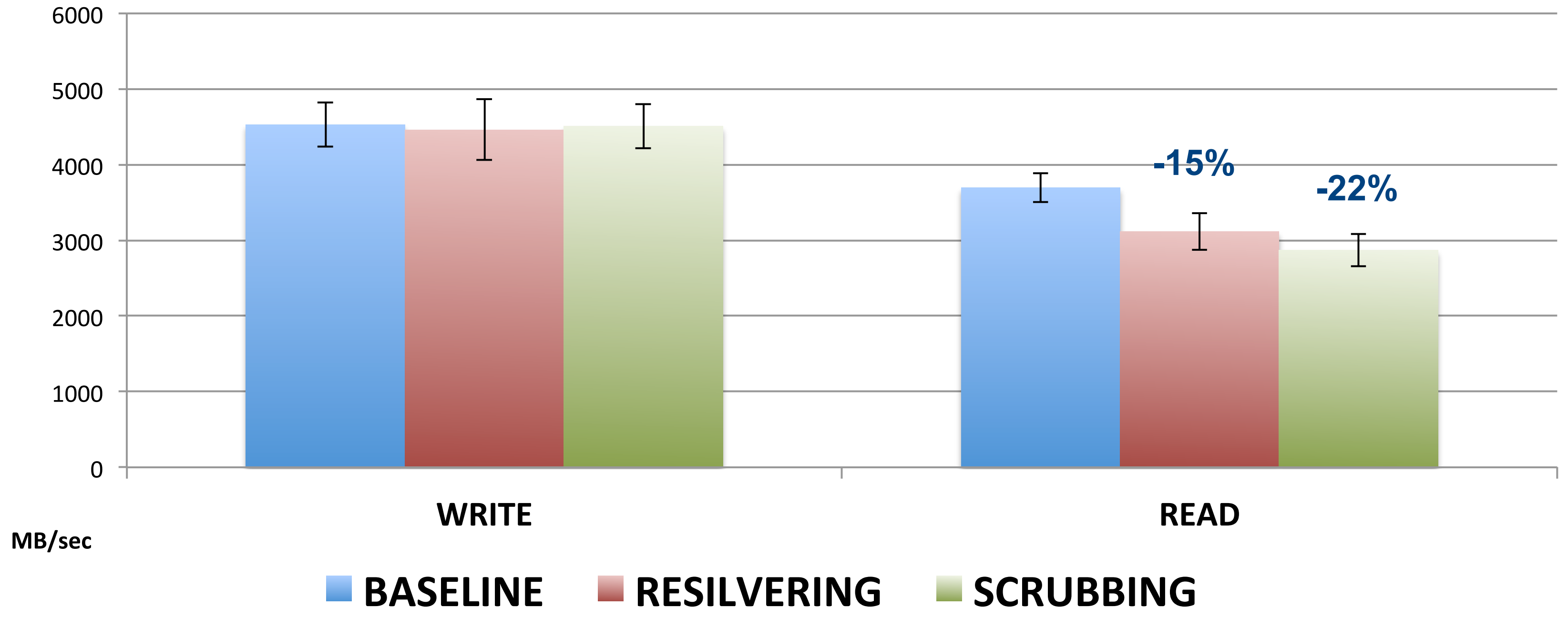
Device	Hard Error Rate in bits	Equivalent in PB’s
SATA Consumer	10E14	0.01
SATA/SAS Nearline Enterprise	10E15	0.11
Enterprise SAS/FC	10E16	1.11
LTO and some Enterprise SAS SSD’s	10E17	11.10
Enterprise Tape or greater	10E19	1110.22

		Sustained Transfer Rate per Second for a Year			
	SDC Rate	10 GiB/s	100 GiB/s	1 TiB/s	10 TiB/s
SAS/FC	10E21	0.0	0.0	0.3	2.7
	10E20	0.0	0.3	2.7	27.1
	10E19	0.3	2.7	27.1	270.9
	10E18	2.7	27.1	270.9	2,708.9
SATA/IB Standard	10E17	27.1	270.9	2,708.9	27,089.2
	10E16	270.9	2,708.9	27,089.2	270,892.2
	10E15	2,708.9	27,089.2	270,892.2	2,708,921.8

Reliability with ZFS

- ZFS only copies “live,” or relevant, blocks of data when creating mirrors or RAID groups. This means that it takes nearly zero time to create and “initialize” new RAID sets.
- Always consistent on disk (**software bug or massive corruption?**):
  - FSCK is a challenge for a RAID6 using 10x 6TB NL/SAS
  - MDRAID need a rescan of the RAID array at each reboot
- Scrubbing:
  - ZFS can do background scrubbing of data by reading all of the blocks and doing checksum comparison and correction.
- Resilvering
  - ZFS addresses the rebuild resilvering “top-down” from the most important blocks in its tree to the least—only reconstructing blocks that matter and writing those on the new drive, and it verifies the validity of every block read using its checksums, stored in the “parent” blocks along the way. This is extremely significant for both efficiency and data integrity—especially as drives continue to grow.

Performance regression during repair



Only READS are affected during repair. Resilvering and scrubbing are autotuned by the ZFS I/O scheduler.

- IOR results from 384 threads and an aggregate file size of 1.5TB
- RAID-Z2 using 7 HDD. 8 OSTs are available.
- Resilvering and Scrubbing on one OST during all the IOR run

Reliability – Where to improve ?

- Declustering ZFS
  - Randomize distribution of RAIDZ redundancy groups and spares to have full bandwidth during resilvering.
- ZFS Event Daemon
  - ZED can trap events and make actions.
  - ZED is implemented in the latest version (0.6.3) of ZFS
  - ZED can help to add hot spare disks automatically

Availability

Lustre* depends on the “ZFS on Linux” implementation of ZFS

- Integration with Pacemaker/Corosync is not a problem
  - Pools should be imported in a non-persistent way
  - One script to import/export the pools
  - One script to mount/unmount Lustre*

Conclusion

ZFS can enable a safe and efficient software RAID solutions using JBODs.

ZFS can guarantee a robust data protection without any special protocol (T10-PI).

Sequential write performance are inline with the expectations. Intel and the ZFS community is working to improve performance.

Evaluation of large (1TB+) SSD device on OST for L2ARC could be a nice next step for this work.

Evaluation of “Enterprise” functionalities like DeDup and Compression.

intel®