# A triad-based architecture for a multipurpose Lustre filesystem at /rdlab

Iván Couto Vivas, Iván Balañá Jiménez and Gabriel Verdejo Álvarez
Universitat Politècnica de Catalunya (UPC), Computer Science Department – Research and Development Lab
ivanc@cs.upc.edu, ivanb@cs.upc.edu, gabriel@cs.upc.edu
https://rdlab.cs.upc.edu

**L**ustre **U**ser **G**roup Conference, May 2021

***Abstract* – In this presentation we are going to explain how our ten years' experience using Lustre for our HPC services, lead us to break its traditional usage silos and vouch for Lustre as the core filesystem for all the /rdlab research services.**

**We will detail how we evolved from a traditional Lustre storage setup [1][2][3] to a triad-based disk architecture in order to fulfill multipurpose requirements: (hard) multi-vendor compatibility, mixed disks technologies, cost-effective/budget constraints and (soft) cloud services and virtualization or general database systems.**

**This idea was first discussed in 2015 with other fellow IT research teams and was developed, tested and improved through these years, taking a definitive boost in 2018 with the ZFS Lustre support.**

**In a nutshell, our proposal defines entities as a group of several 3-disks members (ZFS mirror) from 3 different physical servers. These entities are assembled to create 3 different OST that will be served by an OSS server triad in an active-active-active HA scheme. This architecture can also be used for MDS/MDT components, but usually on active-passive-passive HA design.**

**We have successfully deployed this idea as our main filesystem (+400TBytes, Lustre 2.12.5 + ZFS) providing HPC and Cloud services for 18 research groups, over 160 researchers and several fellow research centers and universities.**

## CONTEXT AND MOTIVATION

Lustre is a high-performance parallel filesystem that has been consolidated through these years in research niches and in HPC mainly, despite of their robustness, flexibility and scalability. Unlike other parallel well-known file systems like *Ceph* [4], *GlusterFS* [5] or *BeeGFS* [6], it is very unusual to find Lustre as the main file system for cloud services, virtualization, databases or any general-purpose initiative regardless of their multiple benefits.

When this topic is openly discussed, we find most of the times that there are no real disadvantages or technical problems about why not to use a Lustre solution for their core storage. Usually, just bunch of misunderstandings (a very complex architecture, an expensive setup, maintenance costs…) and unfounded fears (integration with their existing hardware/vendors, the early adopter risk, user support…).

However, the same question always shines: "*OK, but who is using Lustre as core filesystem*?". Well, we are, and this is how we did it.

## A TRIAD-BASED ARCHITECTURE FOR LUSTRE

Common high performance Lustre deployments (see figure 1) rely mainly in two classical architecture categories:

- Type A: Several n-disk volumes OST governed by a single dedicated OSS.
- Type B: A multiple-disk OST pair (2-tuple) attached to a couple of OSS in order to provide high availability.

Our triad-based approach tries to evolve this well-known architecture, taking advantage of the Lustre ZFS support and the iSCSI Extensions for RDMA (*iSER* [8]) in order to provide a good balance between economic cost, scalability, performance and trustworthiness.

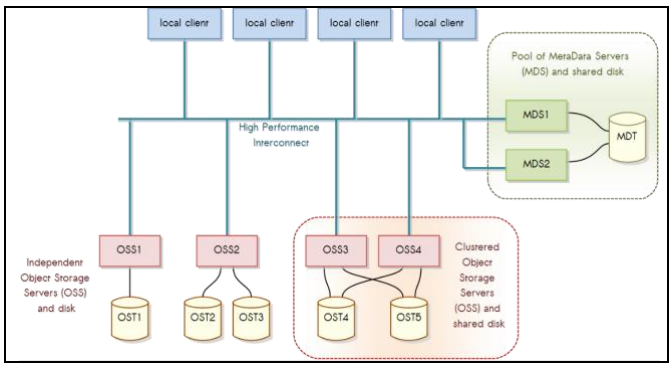


Fig. 1: Classical Lustre layout using Type A and Type B [7]

On the one hand, we keep the essence of the flexible architecture provided by Lustre and their outstanding performance. On the other hand, we can easily integrate a very cost-competitive hardware that is commonly used in

general purpose filesystems. Therefore, we can expand the existing boundaries of traditional Lustre usage silos and open our systems and services to fulfill nowadays requirements and new opportunities.

In order to create our initial triad base setup, we will need 3 disk servers (JBOD or external arrays can also be used) with, at the minimum, the equal number of disks and the same technology. Keep in mind that is possible to mix different disk types, technology or capacities (ex: 2 x SATA3, 6 x SSD…) but all 3 servers must have the same layout in order to build a coherent triad (see figure 2).

We also need a dedicated low latency network (typically infiniband based) to attach the triad members directly using a point-to-point connection to offer the "same" performance for every disk regardless if it is local or not. Using a direct connection to attach the triad members, also avoids the Single Point of Failure (*SPOF*) problem and provides a fault-tolerant architecture.

We share all disk devices through this high-speed low-latency network to ensure their availability in every triad member. The *iSER* provides this service over infiniband easily, but you can use any device export method that ensures a reliable access to the local and remote disks.

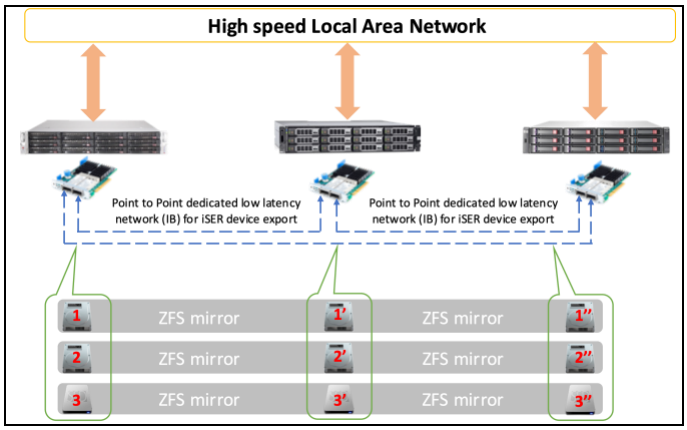

Fig. 2: Triad-based architecture connection (general)

In our experience, this dedicated network must be good enough to keep overall performance. Therefore, at least, the remote disk access and input/output must be >90% of local disk access time and performance (MB/s).

When this basic setup is completed, we need to create a 3 disk/device ZFS mirror using one disk of every member of the triad (see figure 2). We will repeat this process for every available disk in the triad in order to get as many ZFS mirrors as needed.

We can use different disk capacities or technologies in a triad-server, but you must group them into the same ZFS mirror to avoid further problems.

As you can see, this model can be extended or reduced easily just adding or removing disks to the ZFS mirror in order to create n-tuple arrays.

When all the 3-disk ZFS mirrors had been created, we can scatter these mirrors across 3 ZFS stripped pools (see figure 3) that belong to OST0, OST1 and OST2.

We decided to use a ZFS triad base architecture in our services after a careful assessment of the following facts:

1. ***Performance***: Remote disks, at least, must deliver >90% of the local disk output.

2. ***Data cost vs. redundancy***: 2 data copies could not be enough for sensitive data in most environments, whereas, 4 data copies are too expensive for large setups.

3. ***Reliability***: ZFS has a checksum-CRC system and some self-healing capabilities in order to verify data integrity [9][10]. If an error is detected, an odd number of copies will help to untie undesirable situations.

4. ***Standalone***: Isolate the impact of a total disaster in a group or a mandatory maintenance action. Bigger server configurations maximize the damage.

5. ***Rebuild impact***: Using ZFS triad groups ensures that the worst-case scenario (disk replacement) only affects locally, not even to the whole OST. Besides, ZFS, unlike most commonly used systems (ex: arrays with raid-controllers or Linux mdadm) only copies the effective used data sectors to the new disk, avoiding the "mandatory" need to sync all disk sectors and stress the entire system.

   Keep in mind that disks/storage devices, regardless their technology, will increase their capacity every year. Using many disk configurations in a group like common JBOD or vendor arrays setups using 12/24/48 disks RAID5/6/50/60+spare, will increase the failure likelihood of another disk while rebuilds are in progress due the long time required and the big amount of system stress [11][12].

6. ***Quorum***: In order to resolve safely any High Availability (HA) issues regarding OST servers or data checksum divergences, we need an uneven number of members [13][14].

Moreover, building three different pools (OST0/1/2) instead of a single big group, let us assign one pool for OSS taking advantage of the enhance performance offered in an active-active-active architecture. Additionally, if big-file support is required, Lustre provides file-stripping over OST [15].

The HA of this system is granted through a customized *Corosync* configuration. By default, every OSS will mount and deliver a single OST and provides backup for the other triad members as well as keep data integrity for every 3-disk ZFS mirror.

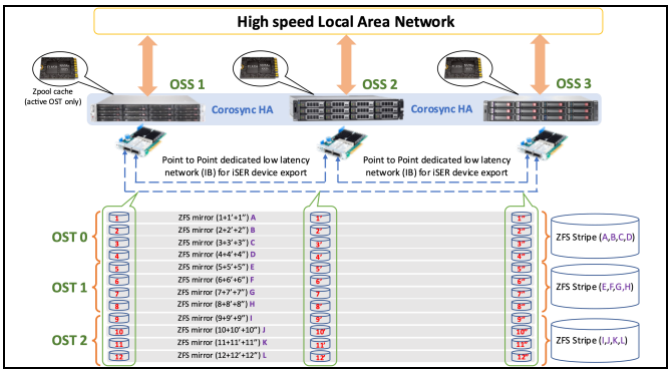


Fig. 3: Triad-based architecture (detail)

This High Availability management service could also be used along with the ZFS pool cache feature [16][17] to increase the overall pool output using a high-speed device (see figure 3). Currently, only the synchronous writes are cached (*SLOG*) [18]. Therefore, we recommend to use a read-only cache configuration.

Lustre supports ZFS natively, so you can tune your ZFS pool to take advantage of the multiple options provided [19][20] (ex: transparent compression, block size, cache…) but you must be compliant and ensure the Lustre ZFS compatibility attributes [21][22].

## CONCLUSIONS

We have been using Lustre for over 10 years in our HPC services, and this triad-based architecture idea (to be honest, with lots of prototypes/tests) since 2016. We successfully completed its full deployment as our main storage system in 2019 with noteworthy results.

Our positive experience using Lustre as our core filesystem to provide research support, see figure 4, inspire us to share our knowledge with the Lustre community to make a small contribution to this great project.

Our proposal tries to evolve the "orthodox" Lustre structure in order to provide a general-use filesystem that expands its utilization out the known boundaries.

The main advantages of this triad-based architecture are:

- **Flexibility**: This Lustre triad-based layout lets you combine, easily, different disk technologies in the same storage servers. It can also be integrated with any other existing setup, and escalates as much as needed without any further trouble.

- **Performance**: Using an active-active-active setup maximizes the Input/Output efficiency, splitting all the filesystem requests among the 3 OSS-OST ZFS stripped pools.

  In addition, *NVME* or *RAM* disks could be added to cache the pool reads effortlessly.

- **Reliability**: All data are always synced over the 3-disk mirror, granted by the ZFS filesystem itself, avoiding other software layers (ex. Linux mdadm, LVM) or any hardware-controller dependencies.

  Moreover, if an extra security for your data is required, you can simply add more disks to the ZFS mirror in order to create multiple data copies in every triad member.

- **Cost-effective**: You can extend your current Lustre solution to offer new general-purpose services with a tight budget, or even reuse old storage servers.

  Besides, transparent compression or deduplication provided by ZFS [24][25] (among other properties) can make the most of your storage platform.

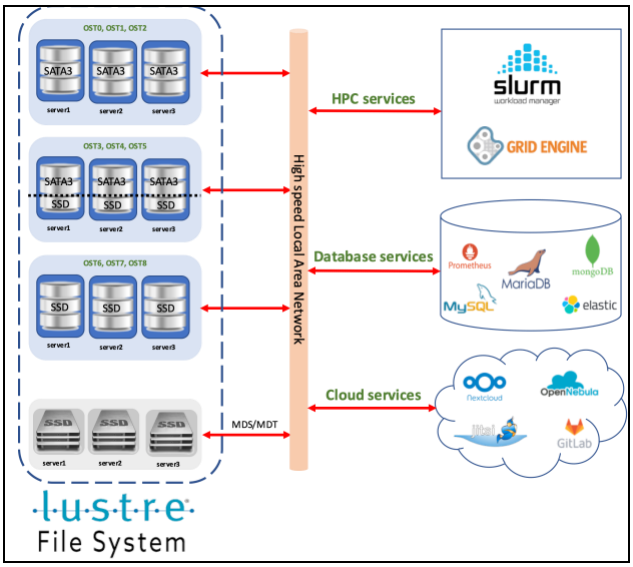


Fig. 4: Multi-purpose Lustre filesystem (example)

## ACKNOWLEDGMENT

We must deeply thank, honestly, to the IT staff at the /rdlab (current and former) and the Computer Science Department chairmanship. Our fellow IT and research members at UPC, and to RedIRIS (Spanish Academic and Research Network) for their support and patience through these years. And last, but not least, to the Lustre community for their efforts to foster this initiative and for the wiki/documentation ☺.

## ABOUT THE SPEAKER

Mr. Gabriel Verdejo Alvarez is the IT manager at the Research and Development Lab (/rdlab) since 2010, and full-time staff at Universitat Politècnica de Catalunya (UPC), Computer Science Department (CS), after 2003. Previously, worked as a senior IT consultant for innovation and database processes.

The /rdlab provides specific IT support for the UPC research groups, fellow universities and research centers in their national and European projects in order to foster their technology transfer initiatives. Nowadays, over 160 researchers and 18 research groups make use of our High Performance Computing (HPC) and cloud services.

Further information is available at his homepage:

https://www.cs.upc.edu/~gabriel/
https://rdlab.cs.upc.edu

## REFERENCES


[1] https://wiki.lustre.org/Lustre_Object_Storage_Service_(OSS)
[2] https://www.snia.org/sites/default/files/RogerGoff_SDC2014_LustreFS_Evolution.pdf
[3] https://www.intel.com/content/dam/www/public/us/en/documents/training/lustre-ost-pools.pdf
[4] https://ceph.io/
[5] https://www.gluster.org/
[6] https://www.beegfs.io
[7] https://www.jiscmail.ac.uk/DELL04.Handover_training.pdf
[8] https://en.wikipedia.org/wiki/ISCSI_Extensions_for_RDMA
[9] https://openzfs.github.io/openzfs-docs/Basic%20Concepts/Checksums.html
[10] https://docs.oracle.com/cd/E19253-01/819-5461/6n7ht6r6i/index.html
[11] https://blog.westerndigital.com/hyperscale-why-raid-systems-are-dangerous/
[12] https://www.digistor.com.au/the-latest/Whether-RAID-5-is-still-safe-in-2019/
[13] https://www.systutorials.com/docs/linux/man/5-votequorum/
[14] https://en.wikipedia.org/wiki/High-availability_cluster
[15] https://wiki.lustre.org/Configuring_Lustre_File_Striping
[16] https://wiki.archlinux.org/index.php/ZFS#SSD_Caching
[17] https://docs.oracle.com/cd/E53394_01/html/E54801/gfxtd.html
[18] https://wiki.archlinux.org/index.php/ZFS#SLOG
[19] https://docs.oracle.com/cd/E19253-01/819-5461/6n7ht6r3f/index.html
[20] https://openzfs.github.io/openzfs-docs/
[21] https://wiki.lustre.org/ZFS_OSD
[22] https://wiki.lustre.org/ZFS_OSD_Storage_Basics
[23] https://openzfs.org/wiki/Features#lz4_compression
[24] https://openzfs.org/w/images/4/4d/Compression-Saso_Kiselkov.pdf
[25] https://openzfs.org/w/images/8/8d/ZFS_dedup.pdf
[26] https://www.upc.edu
[27] https://www.cs.upc.edu
[28] https://www.rediris.es/
[29] https://rdlab.cs.upc.edu