ZFS* as backend file system for Lustre* the current status, how to optimize, where to improve

Gabriele Paciucci – Solution Architect

Agenda

The objective of this presentation is to identify the areas where development is focused in order to fill gap in performance or functionalities.

- Benefit of ZFS and Lustre implementation
- Performance
- Reliability
- Availability
- Serviceability
- How to design
- How to tune

Benefit

ZFS is an attractive technology to be used as backend for Lustre

- Copy on Write improves random or misaligned writes
- Zero offline fsck time
- Rebuild time based on the HDD utilization
- Compression (can also improve I/O performance)
- Efficient snapshotting
- Checksum and on block data corruption protection
- Enabling efficient JBOD solutions
- Integrated flash storage hierarchal management

intel Software

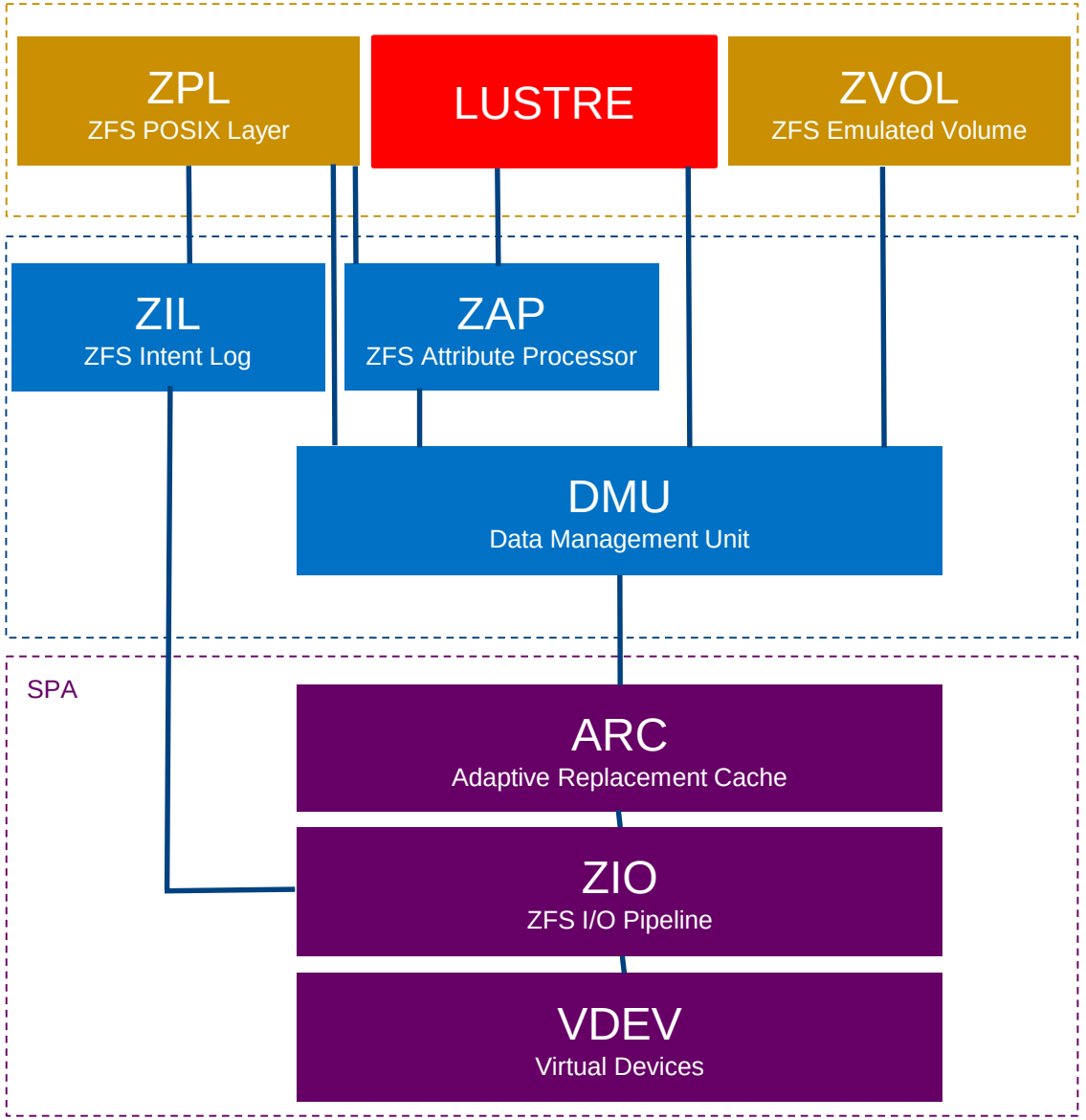
How do Lustre and ZFS interact ?

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
- Targets can be different types (for example LDISKFS as MDTs and ZFS as OSTs)
- Lustre Clients are unaffected by the choice of OSD file system
- ZFS as backend is functional since Lustre 2.4
- ZFS fully supported by Intel

intel Software

ZFS I/O Stack



intel Software

Performance

- Sequential I/O
  - Writes are as good as LDISKFS (or better due the clustering provided by CoW)
  - Reads are affected by the small block size (128K) - LU-6152 + ZFS #2865 planned for 0.6.5
  - Several other minor improvement expected
- Random I/O
  - Random reads I/O can be enhanced using L2ARC devices
  - Writes can be improved by COW but fsync() is a problem without ZIL – LU-4009
- Metadata
  - ZFS Attribute Processor used for all Lustre index files.
  - Increase indirect and leaf block size can help – LU-5391
  - Active OI ZAP blocks should be cached as much as possible to avoid repeated I/Os – LU-5164/LU-5041

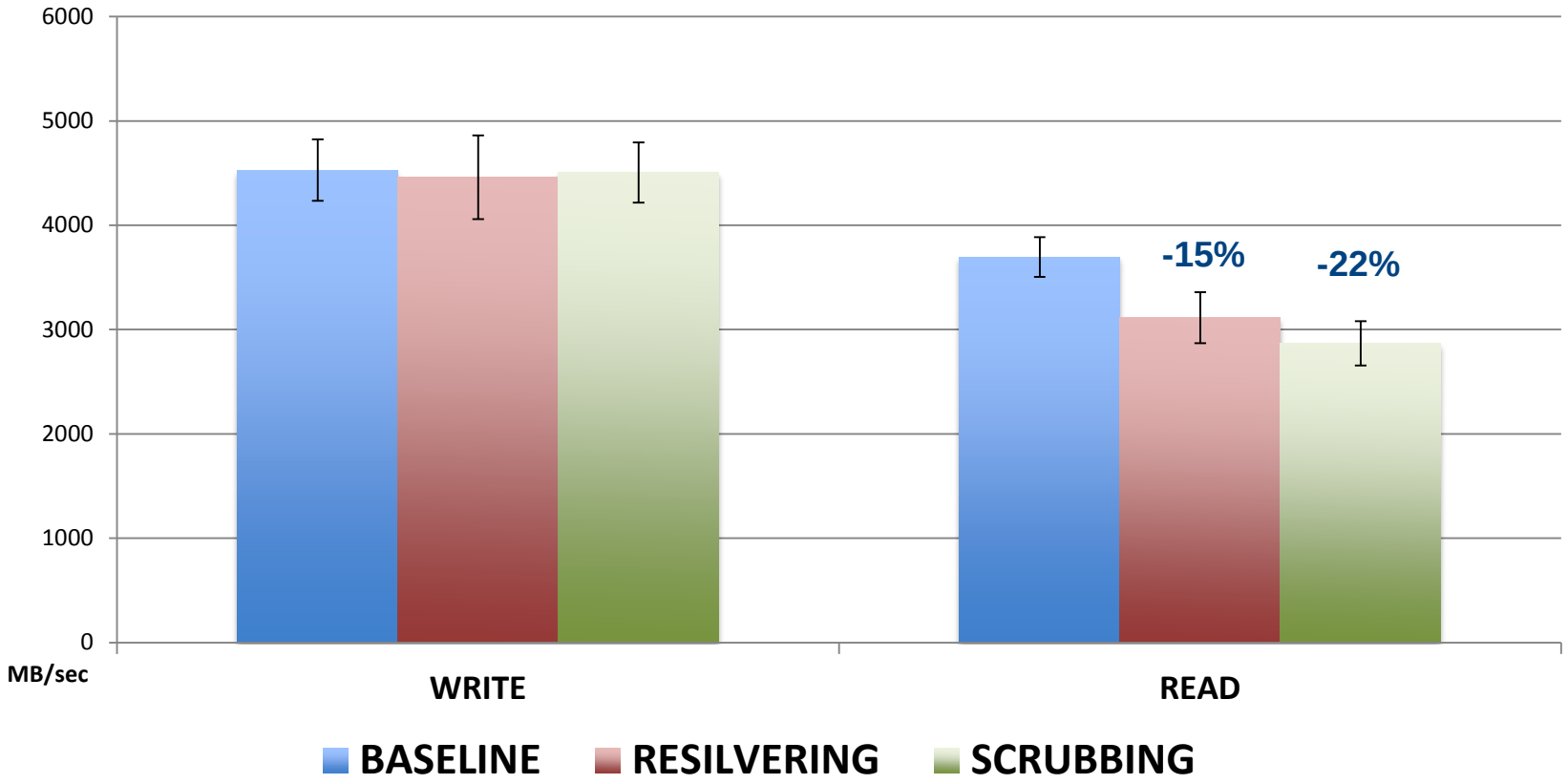
intel Software

Reliability

ZFS is a robust file system, really designed for high reliability

- Lustre+ZFS is end-to-end checksummed
  - but not integrated with the RPC checksum
- Resilvering = Rebuild based on the utilization of the failed disk
  - Increasing resilvering bandwidth using declustered ZFS – future work
- Corrupted blocks on disks are automatically triggered and scrubbed

Performance regression during repair



Only READS are affected during repair.

Resilvering and scrubbing are autotuned by the ZFS I/O scheduler.

This experiment was conducted using the IOR benchmark on 16 compute nodes across 8 OSTs on 4 OSS.

1 OST was impacted by the repair activity during all the IOR run.

intel Software

Availability

- ZFS pools are imported in a persistent manner. Set the "cachefile" property to "none" while creating the pool disable local persistent file.
- Pacemaker’s scripts can be easily developed to:
  - import/export pools
  - mount/umount Lustre targets
- Multi Mount Protection is not yet available – future development

Serviceability

ZFS and online LFSCK can move the serviceability of Lustre to a higher level

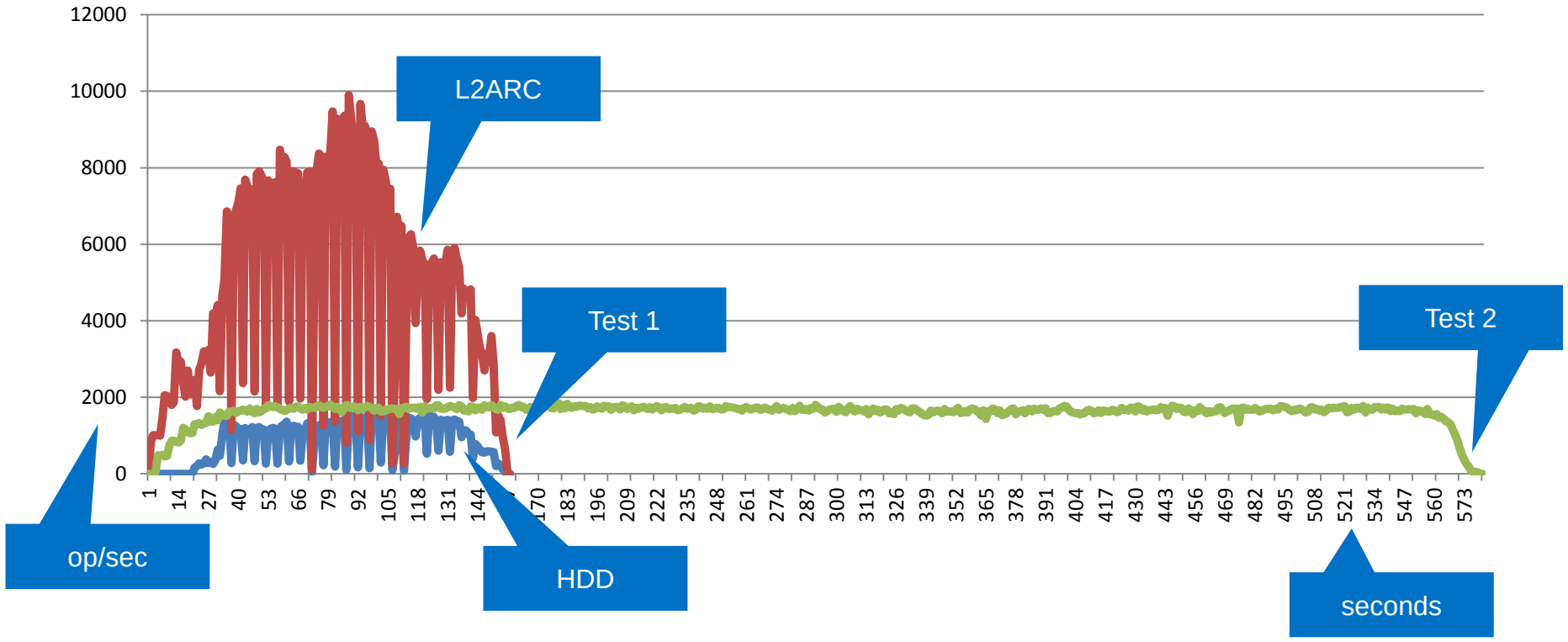
- Zero offline fsck time always consistent on disk
- *lctl LFSCK* can repair logical lustre errors online
- Sys Admin challenges with ZFS, JBODs and RAIDZ
  - Multipath configuration. RedHat* version can't allow to set priority, SuSE does. Alignment between RAIDZ array, SAS expander, SAS HBA and OST is critical.
  - How to find a failed disk? sdparm can help with sas_disk_blink script or SCSI Enclosure Services (SES)
- ZFS Event Daemon – can be trigger by errors and make actions/notifications
- DKMS simplify upgrade and management

How to design a ZFS storage server

- CPU
  - Higher performance single core CPU is critical with RAIDZ2
  - If ZFS is used on top of HW RAID (LLNL's mode), CPU doesn't matter
- RAM
  - ECC memory to better detect in-memory corruption
    - If you have corruption in memory, you can get corruption in ZFS (CoW)
  - The faster memory is better for performance reasons
  - A lot of memory (128GB+)
- Storage
  - Design for storage is still 8+2 also for RAIDZ2
  - Lowering the number of OSTs striping together several RAIDZ2 vdev
- We can take advantage of L2ARC devices (and ZIL devices in the future) for OST
  - L2ARC can be installed locally on the server (cache dropped at every import/export)
  - Great **read** IOPS improvement on OST
  - L2ARC device can be attached to a single pool (not shared between pools)

intel Software

L2ARC benefit



L2ARC device can improve by 8x n. IOPS of a single OST

Test 1 with L2ARC completes in 157 sec

Test 2 without L2ARC completes in 581 sec

Reading 3.84M files each 64K from 16 clients in parallel. Lustre configured using ZFS on 4 OSS. On each OSS, Intel configured 1 OST using 16 HDD and 1 Intel DCP 3700 SSD as L2ARC device. In the chart: operations/sec during both experiments from a single OST

intel Software

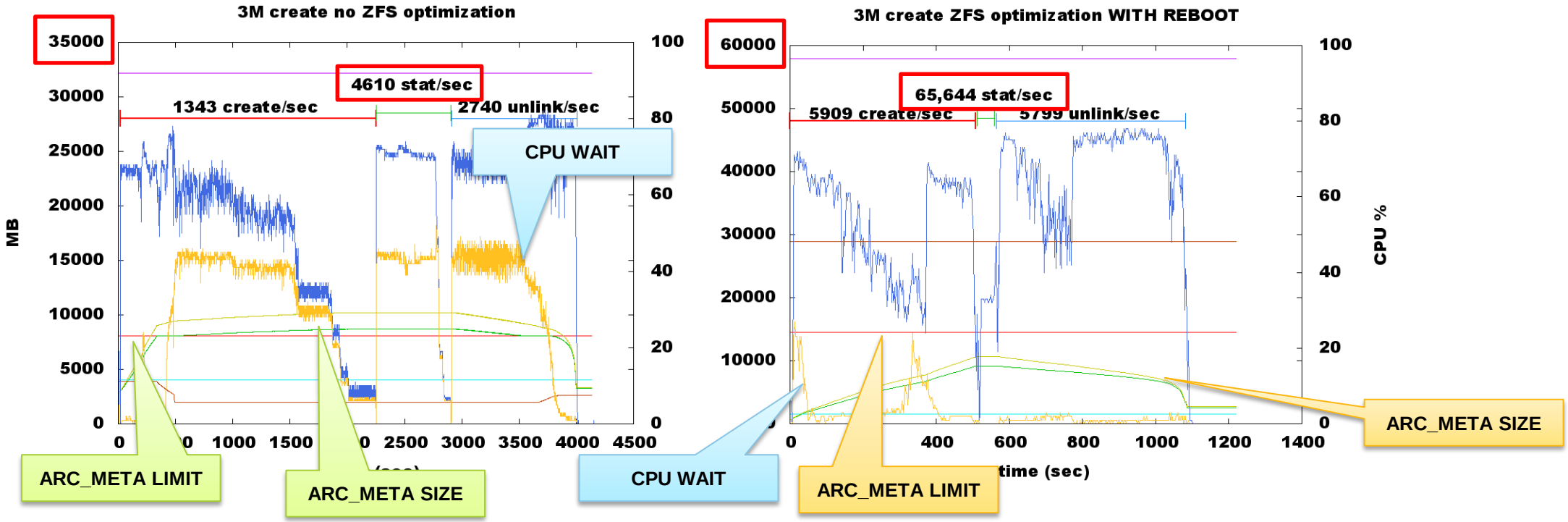
How to tune – General parameters

Parameter	Notes	Default	Suggested
osd_object_sync_delay_us	To improve fsync() performance until ZIL device, it is possible disable the code which causes Lustre to block waiting on a TXG to sync.		Wait for ZIL support
zfs_prefetch_disable	ZFS's prefetch algorithm was designed to handle common server and desktop workloads.	0	1
metaslab_debug_unload	This option prevents ZFS from unloading the spacemap from a metaslab once it is read in.	0	1
spl_kmem_cache_slab_limit	The SPL slab is known to suffer from a significant level of fragmentation under certain workloads.		16384
spl_kmem_cache_reclaim	By default only a single pass is made over the SPL slabs when reclaim is needed.	1	0

Out-of-the-box ZFS is not tuned for Lustre. Various parameters should be modified for Lustre and reviewed for the specific workload.

intel Software

How to tune – ZFS memory management



Parameter	Notes	Default	Suggested
zfs_arc_max	Maximum size of the ARC.	½ RAM	¾ RAM
zfs_arc_meta_limit	Increasing this value will improve performance if the workload involves operations on a large number of files and directories, or frequent metadata operations, at the cost of less file data fitting in the ARC.	¼ ARC	¾ ARC for the MDS only

intel Software

How to tune – ZFS I/O scheduler

Parameter	Notes	Default	Suggested
zfs_dirty_data_max	Amount of dirty data on the system, Able to absorb more workload variation before throttling	10% RAM	1 - 4GB
zfs_vdev_async_write_min_active	Minimum outstanding writes per VDEV	1	5
zfs_vdev_async_write_max_active	Maximum outstanding writes per VDEV	10	15
zfs_vdev_async_write_active_min_dirty_percent	Minimum of dirty data to start delay write operations	30	20
zfs_vdev_scheduler	VDEV scheduler	noop	deadline

When dirty data is less than 30% of zfs_dirty_data_max, ZFS keeps 1 outstanding writes per VDEV. Then dirty data would build up very quickly, since there's only 1 outstanding write per disk, ZFS would start to delay or even halt writes.

The zfs_dirty_data_max should ideally match the backend storage capability. The code simply uses 10% of system memory as the default.

Conclusion and future directions

There’s a lot of interesting work already and more to come

- Performance
  - Increasing ZFS block size, Metadata performance, optimizing Lustre code for ZFS
  - Supporting the ZFS Intent Log device
  - Implementing ZFS’s tunables for Lustre during mkfs and mount
- Reliability
  - Implementing De-clustering ZFS
- Availability
  - Implementing Multi Mount Protection for ZFS
- Serviceability
  - ZFS Event Daemon specific scripts for Lustre

Legal Information

- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
- This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.
- The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.
- Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.
- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at http://www.intel.com/content/www/us/en/software/intel-solutions-for-lustre-software.html.
- Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase.
- Test and System Configurations are conducted by Intel in the Intel's HPC Swindon Lab, UK all the technical details are available
- For more complete information about performance and benchmark results, visit http://www.intel.com/performance.
- Intel and the Intel logo, are trademarks of Intel Corporation in the U.S. and/or other countries.

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

Software


intel
Software