Lustre* Features In Development

Fan Yong

High Performance Data Division, Intel

CLUG 2016 @Shanghai

Legal Information

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.

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. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

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.

Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

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.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

Intel, the Intel logo and Intel® Omni-Path 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.

Ongoing performance and functional improvements

- ZFS* feature and performance improvements (Intel, LLNL)
  - Performance improvement, global snapshot, …
  - MMP, large dnode, …
- Composite File Layout for performance and ease of use (Intel, ORNL)
  - Low stat overhead for small file, high IO bandwidth for large file
  - Infrastructure for others: DoM, FLR, HSM partial restore, …
- Multi-Rail LNet for network performance and reliability
- Data-on-MDT for small file performance/latency
- File Level Redundancy for reliability and performance
- Miscellaneous features and researches

Statements regarding future functionality are estimates only and are subject to change without notice

Multi-Rail LNet (Intel, SGI*)

Allow LNet across multiple network interfaces

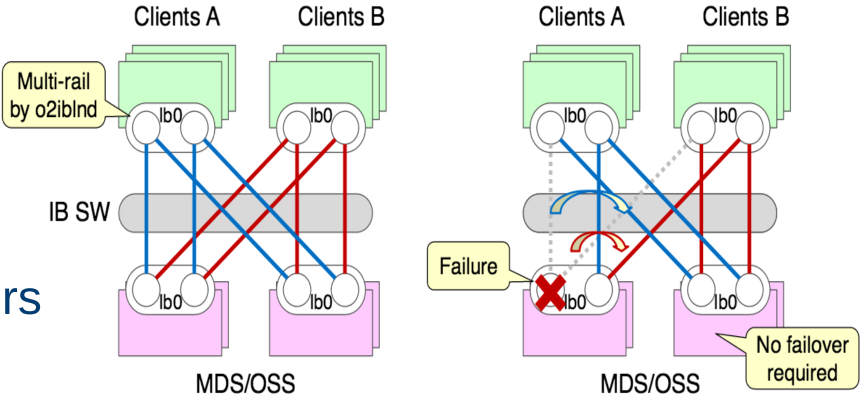
- Supports all LNet networks – LNet layer instead of LND layer
- Allows concurrent use of different LNDs (e.g. both TCP and IB at one time)

Scales performance

- Aggregates multiple network interfaces

Improves reliability

- Active-active network links between peers



Statements regarding future functionality are estimates only and are subject to change without notice

Improve small file performance (Intel)

Data-on-MDT (DoM) optimizes small file IO

- Avoid OST RPC overhead (data and lock RPCs)
- Use high-IOPS MDT storage (mirrored SSD vs. RAID-6 HDD)
- Pre-fetch file data with metadata
- Size on MDT for regular files
- Manage MDT space usage by quota

Client RPC	MDT reply	OST reply
open + truncate	layout, lock (DOM), full attributes (including size), data (pre-fetch)	N/A
write	N/A	-

T1: partial write small file with DoM

Client RPC	MDT reply	OST reply
open	layout, partial attributes	N/A
truncate	N/A	-
glimpse lock	N/A	size + time
extent lock	N/A	lock
read	N/A	data
write	N/A	-

T2: partial write small file without DoM

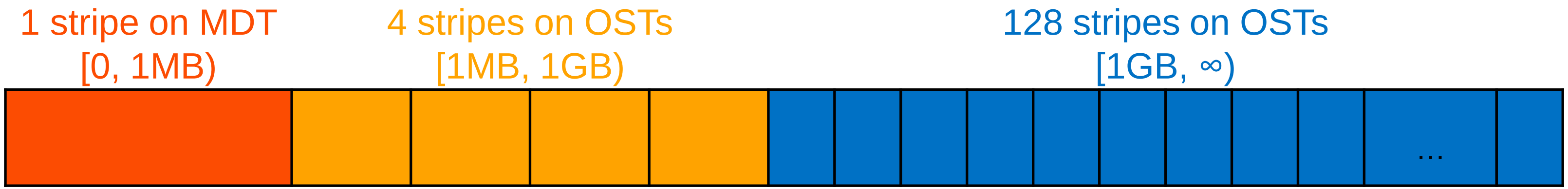
Statements regarding future functionality are estimates only and are subject to change without notice

Improve small file performance (con’t)

Use PFL to handle enlarging small file

- Extend larger file from MDT to OST(s)

FPL example for extending enlarged DoM file with 3 components



1 stripe on MDT
[0, 1MB)

4 stripes on OSTs
[1MB, 1GB)

128 stripes on OSTs
[1GB, ∞)

Complementary with DNE 2 striped directory

- Scale small file IOPS with multiple MDTs

Statements regarding future functionality are estimates only and are subject to change without notice

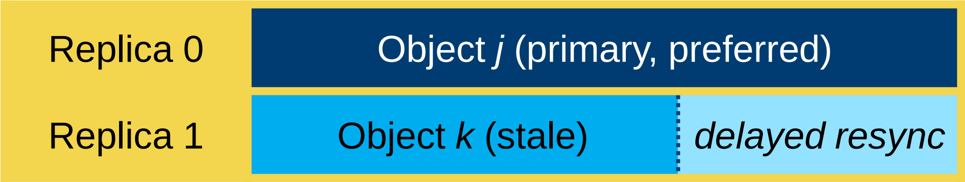
File Level Redundancy (FLR) (Intel)

Provides significant value and functionality for both HPC and Enterprise use

- Select layout on a per-file/dir basis (e.g. mirror all input data, one daily checkpoint)
- Higher availability for server/network failure - finally better than HA failover
- Robustness against data loss/corruption - mirror or M+N erasure coding for stripes
- Increased read speed for widely shared files - mirror input data across many OSTs

Replicate/migrate files between storage classes

- NVRAM->SSD->HDD
- Local vs. remote replicas



Replica 0 | Object *j* (primary, preferred)

Replica 1 | Object *k* (stale) | *delayed resync*

Statements regarding future functionality are estimates only and are subject to change without notice

FLR phased implementation approach

Phases 2/3/4 can be implemented in any order

Phase 0: Composite Layouts from PFL project (Intel, ORNL)

- Plus OST pool inheritance, Project/Pool Quotas

Phase 1: Delayed read-only mirroring – depends on Phase 0

- Manually replicate and migrate data across multiple tiers

Phase 2: Integration with policy engine/copytool - with/after Phase 1

- Automated migration between tiers based on admin policy/space

Phase 3: Immediate write replication – depends on Phase 1

Phase 4: Erasure coding for striped files - with/after Phase 1

Statements regarding future functionality are estimates only and are subject to change without notice

FLR with erasure coding

Erasure coding provides redundancy without 2x or 3x overhead of mirrors

Add redundancy component to existing striped files *after* write is finished

- Can add parity component to any existing RAID-0 file

Suitable for striped files - add N parity per M data stripes (e.g. 16d+3p)

- Parity declustering avoids IO bottlenecks, CPU overhead of too many parities
- Should take failure domains into account (avoid data and parity on same OSS)
  - e.g. split 128-stripe file into 8x (16 data + 3 parity) with 24 parity stripes

dat0	dat1	...	dat15	par0	par1	par2	dat16	dat17	...	dat31	par3	par4	par5	...
0MB	1MB	...	15M	p0.0	q0.0	r0.0	16M	17M	...	31M	p1.0	q1.0	r1.0	...
128	129	...	143	p0.1	q0.1	r0.1	144	145	...	159	p1.1	q1.1	r1.1	...
256	257	...	271	p0.2	q0.2	r0.2	272	273	...	287	p1.2	q1.2	r1.2	...

Statements regarding future functionality are estimates only and are subject to change without notice

Miscellaneous features and researches

Client DLM lockahead (Cray*)

- Allow libraries/apps to pre-fetch locks for striped or arbitrary IO patterns

Code cleanups (ORNL, Intel, Cray)

- Lustre* code kernel stylization, port patches to/from kernel
- RHEL weak symbol versioning, patchless server kernels

Uni Hamburg + German Client Research Centre (DKRZ)

- Client-side data compression
- Adaptive optimized ZFS* data compression

Lawrence Berkeley National Laboratory

- Spark* and Hadoop* on Lustre

Statements regarding future functionality are estimates only and are subject to change without notice

Potential development proposals for the future...

DNE enhancements

- Dynamic metadata migration among shards of striped directory
- Metadata Redundancy via DNE2 distributed transactions

Tiered storage with Composite Layouts and File Level Redundancy

- Integration with RobinHood to manage migration between tiers, rebuild replicas

Local persistent cache on client with fscache or local OSD

- Use FLR to ensure availability in case of client failure

Statements regarding future functionality are estimates only and are subject to change without notice