Topics

• Lustre Wire Checksum Improvements
  – Cleanup
  – Portability
  – Algorithms
  – Performance

• End-to-End Data Integrity
  – T10DIF/DIX
  – Version Mirroring
Goals

• End user assurance that their data was written to disk accurately
• Protection against all RAID (single) failure modes
• Offload the heavy calculations from the servers
• Support a wider range of client/server hardware
Over-the-Wire Bulk Checksums
Bulk Checksum History

• Initially only software CRC-32 (IEEE, ANSI) 2007
• Adler-32 added in 1.6.5
  – easy to calculate
  – weak for small message sizes
• Shuichi Ihara added initial support for hardware CRC-32C (Castagnoli)
  – Intel Nehalem
  – bz 23549, landed in Lustre 2.2
• WC added multi-threaded ptlrpcd, and bulk RPC checksums moved into ptlrpcd context: parallelized checksums
  – LU-884, LU-1019, in Lustre 2.2
• sptlrpc implementation used a different set of functions
  – CRC-32, Adler, MD5, SHA1-512
Bulk Checksum Changes

• Cleanup of sptlrpc and bulk checksum algorithms to use the kernel crypto hash library
  – simplifies future additions
  – LU-1201

• Addition of Software CRC-32C support
  – eg. if server has HW support and clients don’t
  – LU-1201

• Implementation of Hardware CRC-32 using PCLMULQDQ
  – Intel Westmere
  – MRP-314, still testing
Bulk Checksum Speeds, MB/s

- **Zlib**
- **rsync**
- **IEEE 802.3**
- **ANSI X3.66**
- **gzip**
- **bzip2**
- **mpeg2**
- **png**
- **iSCSI**
- **Btrfs**

- **Adler**
- **CRC-32**
- **CRC-32C**

**SW**

**HW**
End-to-End Data Integrity with T10 and Version Mirroring
The guard tag protects the data portion of the sector. The application tag is simply opaque storage. And finally, the reference tag is being used to protect against out-of-order and misdirected write scenarios.

Standardizing the contents of the protection information enables all nodes in the I/O path, including the disk itself, to verify the integrity of the data block.

Comparison of I/O Paths

A typical I/O submission scenario in an enterprise configuration is illustrated in figure 2. The only entity capable of using the 8 bytes of protection information is the array firmware.

A similar DIF-enabled configuration will look like figure 3. The I/O controller generates and appends the protection information and every subsequent node in the I/O path can verify the data integrity.

Combining T10 DIF with the Data Integrity Extensions allows the protection information to be attached even higher up in the stack–either in the application or in the operating system. The entire I/O path is protected and true end-to-end data integrity protection is achieved.

- 8 bytes of PI appended to 512 byte sectors
- HBA and disk drives must support T10-DIF in hardware
Figure 3
DIF I/O write: Application writes byte stream to OS. Filesystem writes in logical blocks that are multiples of 512-byte sectors. HBA generates protection information and sends out 520-byte sectors. SAN switch can optionally check protection information. Array firmware verifies protection information, optionally remaps reference tags and writes to disk. Disk verifies protection information before storing request.

Figure 4
DIF I/O write: Application writes byte stream to OS, optionally including protection information. Filesystem writes in logical blocks that are multiples of 512-byte sectors. If no protection information has been generated, OS automatically does so and attaches it to the I/O. HBA verifies data integrity, merges data and protection scatterlists and sends out 520-byte sectors. SAN switch can optionally check protection information. Array firmware verifies protection information, optionally remaps reference tags and writes to disk. Disk verifies protection information before storing request.

Figure 5 illustrates the protection envelopes of the protection schemes mentioned above. The Normal I/O line illustrates the disjoint integrity coverage offered using a current operating system and standard hardware. The HARD line shows the protection envelope offered by the Oracle Database accessing a disk array with HARD capability.

T10-DIF
Figure 3

DIF I/O write: Application writes byte stream to OS. Filesystem writes in logical blocks that are multiples of 512-byte sectors. HBA generates protection information and sends out 520-byte sectors. SAN switch can optionally check protection information. Array firmware verifies protection information, optionally remaps reference tags and writes to disk. Disk verifies protection information before storing request.

Figure 4

DIX I/O write: Application writes byte stream to OS, optionally including protection information. Filesystem writes in logical blocks that are multiples of 512-byte sectors. If no protection information has been generated, OS automatically does so and attaches it to the I/O. HBA verifies data integrity, merges data and protection scatterlists and sends out 520-byte sectors. SAN switch can optionally check protection information. Array firmware verifies protection information, optionally remaps reference tags and writes to disk. Disk verifies protection information before storing request.

Figure 5 illustrates the protection envelopes of the protection schemes mentioned above. The Normal I/O line illustrates the disjoint integrity coverage offered using a current operating system and standard hardware. The HARD line shows the protection envelope offered by the Oracle Database accessing a disk array with HARD capability. DIF shows coverage...
Figure 3
DIF I/O write: Application writes byte stream to OS. Filesystem writes in logical blocks that are multiples of 512-byte sectors. HBA generates protection information and sends out 520-byte sectors. SAN switch can optionally check protection information. Array firmware verifies protection information, optionally remaps reference tags and writes to disk. Disk verifies protection information before storing request.

Figure 4
DIX I/O write: Application writes byte stream to OS, optionally including protection information. Filesystem writes in logical blocks that are multiples of 512-byte sectors. If no protection information has been generated, OS automatically does so and attaches it to the I/O. HBA verifies data integrity, merges data and protection scatterlists and sends out 520-byte sectors. SAN switch can optionally check protection information. Array firmware verifies protection information, optionally remaps reference tags and writes to disk. Disk verifies protection information before storing request.

Figure 5 illustrates the protection envelopes of the protection schemes mentioned above. The Normal I/O line illustrates the disjoint integrity coverage offered using a current operating system and standard hardware. The HARD line shows the protection envelope offered by the Oracle Database accessing a disk array with HARD capability. DIF shows coverage with Lustre.
Changes to Lustre

- Additional checksum data described or carried in brw RPC
- Add PI and checking to data path
- For mmap’ed pages, early GRD failure implies data has changed, recompute from OSC
- Disable bulk checksums
- Optional GRD checking on OSS can push all checksum load to HBA/disk hardware
RAID failure modes

- *Parity Lost and Parity Regained* - Andrew Krioukov
- Latent Sector (reliable read) errors
- Data Corruption
- Misdirected Writes
- Lost Writes
- Torn Writes
- Parity Pollution

- Outcomes
  - Data recovery
  - Data loss (detected)
  - Data corruption (silent)
RAID with Version Mirroring

- RAID stripe across disks
- Block (chunk) on one disk
- Multiple sectors per block
- Store block version in T10 APP field
  - sectors within a chunk store the same version
  - parity block contains version vector
Rebuild with Version Mirroring

Update C to C’, write C’ and P(ABC’)

Later, attempt to update A to A’
First, read B & C to prepare for constructing new P
But first read P to verify versions before writing P(A’BC’)
Version mismatch, latest is in P, so reconstruct C’ from P

Write C’
Calculate P(A’BC’)
Write A’ and P(A’BC’)

Friday, April 20, 12
RAID failure modes

- Latent Sector (reliable read) errors - drive detects
- Data Corruption - GRD, lightweight, partial reads
- Misdirected Writes - REF
- Lost Writes - parity block version vector
- Torn Writes - sector versions
- Parity Pollution - GRD + versions allow safe reconstruction
Fin