T10PI End-to-End Data Integrity Protection for Lustre

Shuichi Ihara, Li Xi

DataDirect Networks, Inc.
Why is data Integrity important?

▶ Data corruptions is painful!
  • Frequency is low, but cost is very high.
  • A lot of unusual operations and step by step procedures to recover.

▶ What causes data corruptions?
  • Facility
  • Hardware include network
  • Software
  • Human errors
Type of data corruption

▶ Two types of data corruption
  • Latent sector/block errors
    ◦ Application can’t read sector/block and return an error.
  • Silent data corruption
    ◦ Application can read sector/block, but it’s NOT expected data and NOT valid data.

▶ Silent data corruption causes another corruptions
  • Application read data as expected and write new data based on it, but it’s wrong!

▶ Where/Why this happens?
  • All storage stacks(App, OS, HBA, Storage Fabric/Array, Disk)
  • Lack of integrity check, each storage stack trusts upper/lower comportment.
Data Integrity of Lustre

▶ Lustre checksum
• Checksum on between OSCs and OSTs.
• Prevent server/client wrong RPC handling if it’s corrupted.
• No store checksums into Disks.

▶ Backend Storage
• metadata checksum is available in Ext4, but not supported in Lustre.
• ZFS has very strong mechanism for data Integrity
  o CoW, Transaction based, End-to-End checksum, Scrub, etc..
  o Data integrity inside ZFS.

▶ Is this enough?
• Still missing guarantee on some places.
  o After sever received RPCs (e.g. Memory corruptions, OS to HBA to Storage Array, etc)
• There was Lustre End-to-End Data Integrity discussion(LU-2584)
  o Proposed T10 PI/DIX support and submitted patches by Xyratex
  o Required to replace whole Lustre checksum with new T10PI/DIX checksum
T10PI(DIF) and DIX(Data Integrity Extensions)

The standard specify an additional 8 byte field designated for data integrity/protection for each data block.

- **Application or OS add PI.**
  - Application: 512 or 4096 byte stream
  - OS: 512 or 4096 byte sector

**HBA/HCA**
- HBA verify Integrity and merge PI into data send pass it down
- HBA automatically calculate PI and store

**HBA/HCA Storage Controller**
- T10PI aware Storage Array

**HBA**
- T10PI

**HDD/SSD**
- T10PI

**GRD: 2 byte guard tag (CRC of data)**
**APP: 2 byte application tag**
**REF: 4 byte reference tag**
Proposed Design of Lustre End-To-End Data Integrity

- Fully transparent End-to-End Data integrity from Lustre client to disk.
- Relies on open standard format T10PI/DIX and any T10PI/DIX supported hardware work.
- Don’t change Lustre RPC format and extends current Lustre checksum framework.
- Consider minimum performance impacts.
- Keep compatibility for old Lustre version or non-T10PI supported hardware.
## Basic flow of Lustre End-to-End Data Integrity

<table>
<thead>
<tr>
<th>Lustre Client</th>
<th>Lustre Server</th>
<th>OS</th>
<th>HBA</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device Setup</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Check if T10PI/DIX enabled</td>
<td></td>
<td>Load disk with enabling T10PI</td>
<td></td>
</tr>
<tr>
<td><strong>Lustre Setup</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Check if T10PI/DIX is available</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Select checksum algorithm with T10PI/DIX.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Lustre IO</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Calculate T10PI/DIX and send it as Lustre checksum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify Lustre checksum and Re-calculate PI.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pass Data and PI down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify PI and store them into disk as well as data.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- Check if T10PI/DIX is enabled.
- DIX capability.
- Load disk with enabling T10PI.
Today’s Lustre checksum(Write)

- **Write()**
  - Protect
  - User space
  - Lustre Client
  - Serer Bulk Data
  - Network
  - Lustre Sever
  - BIO
  - Data
  - Storage

- **CRC**: Calculate Lustre Checksum per RPC with either adler, crc32c or crc
- **Server RPC**: Send checksum as part of RPC struct
- **CRC’**: Calculate Lustre checksum based on arrived RPCs and verify if CRC’ = CRC

© 2017 DataDirect Networks, Inc. * Other names and brands may be claimed as the property of others. Any statements or representations around future events are subject to change.
Today’s Lustre checksum (Read)

Calculate Lustre checksum based on arrived RPCs and verify if CRC' = CRC

Server RPC
Send checksum as part of RPC struct

Calculate Lustre checksum based on BIO

RCP Request

Serer Bulk Data

User space
Lustre Client

Read()

BIO

Data

Protect

Network
Lustre Sever

RPC Request

Server RPC

Send checksum as part of RPC struct

CRC

CRC'

Lustre
Server

Lustre
Client

Storage

© 2017 DataDirect Networks, Inc. * Other names and brands may be claimed as the property of others. Any statements or representations around future events are subject to change.
Lustre checksum with T10PI/DIX for Enabling End-to-End Data Integrity (Write)

1. **Write()**
   - User space
   - Lustre Client
   - User sends data to Lustre server via User space
   - Data passed to Lustre client

2. **Sector Size**
   - PIs (per sector size)
   - Serer Bulk Data
   - Data is segmented into blocks

3. **Calculate PI per sector size**
   - Client calculates checksum per RPC based on all PIs in a RPC
   - CRC is calculated from PIs

4. **Server RPC**
   - Data sent to server via RPC struct
   - CRC is sent as part of RPC struct

5. **Network**
   - Lustre Server
   - Data is received and processed
   - CRC' is calculated from PIs

6. **CRC**
   - Data is verified if CRC' = CRC

7. **Storage**
   - Data is stored
   - HBA Verify PI merge data and PI scatter list, Pass it down

© 2017 DataDirect Networks, Inc. * Other names and brands may be claimed as the property of others. Any statements or representations around future events are subject to change.
Lustre checksum with T10PI/DIX for Enabling End-to-End Data Integrity (Read)

1. **User space Lustre Client**
   - User space Lustre Client
   - **Server RPC**
   - Send checksum as part of RPC struct
   - **Verify Lustre checksum if CRC' = CRC**

2. **Network Lustre Server**
   - Network Lustre Server
   - **CRC is calculated from PIs**
   - **Verify Lustre checksum if CRC' = CRC**

3. **Lustre Client**
   - **Server Bulk Data**
   - **RPC Request**
   - **Calculate PI from BIO**
   - **HBA**
   - **HBA Verify data with PI.**
   - **CRC**
   - **CRC'**
   - **X**

4. **Storage**
   - **Data + PI**
   - Protect

**Diagram Notes:**
- Protect: Data + PI
- Calculate PI from BIO
- HBA Verify data with PI.
- CRC is calculated from PIs
- Server RPC: Send checksum as part of RPC struct
- Verify Lustre checksum if CRC' = CRC
- User space Lustre Client: User space Lustre Client, Server RPC: Send checksum as part of RPC struct, Verify Lustre checksum if CRC' = CRC
Status

▶ Task is tracked under LU-10472
  • Patch being to submit for review
    o T10PI support for BIO https://review.whamcloud.com/#/c/31513
    o T10PI support for Lustre checksum (https://review.whamcloud.com/#/c/30980)
    o T10PI support for page cache (https://review.whamcloud.com/#/c/30792)
  • Cleanup and optimization are ongoing to finalize patches

▶ Started function test and benchmark
  • Adding test codes
  • Fault injection
  • Comparing performance against today’s Lustre checksum
Test Environment

► 1 x MDS
  • 2 x E5-2640v3, 256GB Memory, 1 x EDR Infibanind
  • 1 x LSI SAS3008(Enabled T10PI/DIX)

► 1 x OSS
  • 2 x E5-2640v3, 256GB Memory, 1 x EDR Infibanind
  • 1 x LSI SAS3008(Enabled T10PI/DIX)

► 1 x SS8462
  • 8 x NL-SAS and 2 x SAS disks connected to OSS/MDS with SAS

► 6 x Client
  • 2 x E5-2660v3, 128GB Memory, 1 x EDR Infibanind

► Use IOR with Lustre Fake-IO
Performance Comparison – Single Client (FPP, Sequential, Write)

Single Client Performance (Write)

- Bandwidth (MB/sec) vs. Number of Thread
- Log scale for Bandwidth
- Different line styles and colors for each configuration:
  - default (crc32c)
  - T10PI (crc32c)
  - default (adler)
  - T10PI (adler)

Client CPU Usage

- CPU Usage (%) vs. Number of Thread
- Different bar colors for each configuration:
  - default (crc32c)
  - T10PI (crc32c)
  - default (adler)
  - T10PI (adler)
Performance Comparison - Single Client (FPP, Sequential, Read)

Single Client Performance (Read)

- Bandwidth (MB/sec) vs. Number of Thread
- Comparing default(crc32c), T10PI(crc32c), default(adler), T10PI(adler)

Client CPU Usage

- CPU Usage (%) vs. Number of Thread
- Comparing default(crc32c), T10PI(crc32c), default(adler), T10PI(adler)
Performance Comparison - Multi Client/Single Server (FPP, Sequential, Write)

**Single Server Performance (Write)**

- Bandwidth (MB/sec) vs. Number of Thread

**Server CPU Usage**

- CPU Usage (%) vs. Number of Thread

### Details

- **Bandwidth (MB/sec)**
  - default (crc32c)
  - T10PI (crc32c)
  - default (adler)
  - T10PI (adler)

- **Number of Thread**
  - 6, 12, 24, 48, 96, 120
Performance Comparison - Multi Client/Single Server (FPP, Sequential, Read)

Single Server Performance (Read)

Server CPU Usage

Bandwidth (MB/sec) vs. Number of Thread

CPU Usage (%) vs. Number of Thread

- default (crc32c)
- T10PI (crc32c)
- default (adler)
- T10PI (adler)
Conclusions

▶ Designed Lustre End-to-End Data integrity
  • Reused current Lustre checksum design and expended with T10PI/DIX.
  • Flexible and adaptable to any T10PI/DIX supported hardware and software.
  • Very minimum performance impacts.

▶ Further Work
  • Cleanup and shape the codes and add additional test codes.
  • Continue benchmark and test many failure scenarios on entire End-to-End comportment.