DIF, DIX and Linux Data Integrity

Martin K. Petersen
Consulting Software Developer, Linux Engineering
Topics

• Data Integrity Technologies
  • Data Corruption
  • T10 DIF
  • Data Integrity Extensions

• Linux Data Integrity Infrastructure
  • SCSI Layer
  • Block Layer
  • Filesystems
  • User Application Interfaces
Data Corruption

• Tendency to focus on corruption inside disk drives
  • Media developing defects
  • Head misses

• However, corruption can - and often does - happen while data is in flight
  • Modern transports like FC and SAS have CRC on the wire
  • Which leaves library / kernel / firmware errors
  • Bad buffer pointers
  • Missing or misdirected writes

• Industry demand for end-to-end protection
  • Oracle HARD is widely deployed
  • Other databases and mission-critical business apps
  • Nearline/archival storage wants belt and suspenders
Data Corruption

• **DIF/DIX** are orthogonal to logical block checksums
  • We still love you, btrfs!
  • Logical block checksum errors are detected at READ time
  • ... which could be months later, original buffer is lost
  • Redundant copy may also be bad if buffer was incorrect

• **This is about:**
  • Proactively preventing bad data from being stored on disk
  • ... and finding out before the original buffer is erased from memory
  • Plus using the integrity metadata for forensics when logical block checksumming fails

• **It's an insurance policy. Must be cheap!**
Disk Drives

• Most disk drives use 512-byte sectors
• A sector is the smallest atomic unit the drive can access
• Each sector is protected by a proprietary cyclic redundancy check internal to the drive firmware
• 4096-byte sectors are coming
• Enterprise drives (Parallel SCSI/SAS/FC) support 520/528 byte “fat” sectors
• Sector sizes that are not a multiple of 512 bytes have seen limited use because operating systems deal with everything in units of 512, 1024, 2048 or 4096 bytes
• RAID arrays make extensive use of fat sectors
Normal I/O
## T10 Data Integrity Field

<table>
<thead>
<tr>
<th>512 bytes of data</th>
<th>GRD</th>
<th>APP</th>
<th>REF</th>
</tr>
</thead>
</table>

- **16-bit guard tag** (CRC of 512-byte data portion)
- **16-bit application tag**
- **32-bit reference tag**

- Only protects between HBA and storage device
- PI interleaved with data sectors on the wire
- Three protection schemes
  - All have guard tag defined
  - Type 1 reference tag is lower 32-bits of target sector
  - Type 2 reference tag is seeded in 32-byte CDB
- SATA T13/EPP uses same PI format
- SSC tape proposal is different (guard only)
T10 Data Integrity Field I/O

Application → Byte stream
OS
I/O Controller → 512 byte sector
SAN
Disk Array
Disk Drive

520 byte sector | 8 byte PI

Xport CRC
Sector CRC
Data Integrity Extensions

- Attempt to extend T10 DIF all the way up to the application, enabling true end-to-end data integrity protection
- Essentially a set of extra knobs for SCSI/SAS/FC controllers
- The Data Integrity Extensions:
  - Enable transfer of protection information to and from host memory
  - Separate data and protection information buffers
  - Provide a set of commands that tell HBA how to handle I/O:
    - Generate, strip, pass, convert and verify protection information
DIX Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>OS</th>
<th>Controller</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ_INSERT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ_STRIP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ_PASS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ_CONVERT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE_INSERT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE_STRIP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE_PASS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE_CONVERT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Integrity Extensions

- Separate protection scatter-gather list
  - 520-byte sectors are inconvenient for the OS
  - A <512, 8, 512, 8, 512, 8, ...> scatterlist is also crappy

- DIF tuple endianness
  - Application tag must be portable across little- and big-endian systems

- Checksum conversion
  - CRC16 is somewhat slow to calculate
  - IP checksum is cheap
  - Strength is in data and protection information buffer separation
Data Integrity Extensions + DIF I/O
Protection Envelopes

**DIX + DIF**
Data Integrity Extensions + T10 Data Integrity Field combined protection envelope

**DIX**
Data Integrity Ext. protection envelope

**DIF**
T10 Data Integrity Field protection envelope

**HARD**
Oracle HARD protection envelope

**Normal I/O**
- vendor specific integrity measures
- vendor specific integrity measures
- vendor specific integrity measures
- transport CRC
- vendor specific integrity measures
- vendor specific integrity measures

**Application** — **OS** — **I/O Controller** — **SAN** — **Disk Array** — **Disk Drive**
Data Integrity Extensions + T10 DIF

- Proof of concept last summer
  - Oracle DB, Linux 2.6.18, Emulex HBA, LSI array, Seagate drives
  - Error injection and recovery
  - Showed Oracle DB crash and burn without DIX+DIF

- Product availability
  - Some hardware shipping
  - Emulex, LSI, Seagate, Hitachi
SNIA Data Integrity Technical Workgroup

• TWG just dropped provisional status
• Aims to broaden participation
• Aims to standardize data integrity terminology
  • Think RAID levels
• Aims to standardize OS-agnostic API and/or common methods for applications to interact with integrity metadata
• Companies at first face 2 face
  • Emulex, Oracle, LSI, Seagate, Qlogic, Brocade, EMC, PMC Sierra, HP, Teradata, IBM, Sun, Microsoft, Symantec
What Is Now?

• SNIA DITWG is obviously a long-term effort
• “Verbatim” DIF exchange via DIX is pretty much good to go
• Block layer changes are in 2.6.27
• SCSI changes partially merged
• Hoping for GA in next generation enterprise distributions
Linux vs. Data Integrity
SCSI Layer Changes

• Mid level
  • INQUIRY and READ CAPACITY(16) during scan
  • Extra scsi_data_buffer in scsi_cmnd
  • Protection operation and target type in scsi_cmnd
  • Protection scatter-gather list mapping

• sd.c
  • CDB prep
  • Block integrity profile registration
  • Virtual sector remapping

• sd_dif.c
  • Callbacks for generation / verification of protection information
Block Layer Changes

- **struct bio**
  - `bio_integrity_payload`
    - Integrity `bio_vec` + housekeeping hanging off of `bio`
    - Filesystem can explicitly attach it...
    - ... or block layer can auto-generate on WRITE
    - Block layer can verify on READ
  - Format of protection information opaque to block layer

- **struct block_device**
  - Has an integrity profile that gets registered by ULD
  - Layered devices must ensure all subdevices have same profile
Block Layer Changes

- `struct request`
  - A few merging constraints
  - Protection buffer ordering is important
Filesystems

• DIF application tag:
  • 2 bytes per sector for Type 1 + 2
  • 6 bytes per sector for Type 3
• FS can attach arbitrary structures which will be interleaved between the available tag space in an I/O
• Essentially allows logical (filesystem) block tagging
• FS can use tags to implement checksumming without changing on-disk format
• Another option is to write stuff that will aid recovery (back pointers, inode numbers, etc.)
User Application Interfaces

• Explicit - libdif
  • mkfs/fsck accessing DIF on block device directly

• Opaque - libintegrity
  • “Protect this buffer”
  • Akin to POSIX async I/O

• Transparent - libc
  • standard read() / write() style calls
  • mmap() => bonghit bonanza
User Application Interface Challenges

- Oracle + ASM
- App. + libdif
- App. + libintegrity
- Future normal I/O
- Normal I/O

Application | Page cache | Filesystem | Block layer | SCSI layer | I/O Controller

Guard tag | Application tag | Reference tag

Remapping / conversion
More Info

  - Documentation
  - DIX specification
  - Patches
  - Source repository