Using Linux block integrity in building and testing storage systems

Mikhail Malygin
Principal Engineer
YADRO
What we are doing?

Storage appliance

• 550PB installed, up to 18PB per system
• OpenPOWER storage controllers

Data integrity is vital
Problem statement

Data integrity for distributed RAID

- detect stealth data corruption
- no external metadata storage
Naive approach

- HDD extended sector
  - 520/528
  - 4104/4112/4160/4224
- FORMAT UNIT (took 16 hours for 12TB)
- and...
Naive approach

- HDD extended sector
  - 520/528
  - 4104/4112/4160/4224
- FORMAT UNIT (took 16 hours for 12TB)
- and... nothing works:
  - unsupported sector size 4160
  - in sd.c sector must be equal to $512 \times 2^n$
RTFM!

• 15 years ago
• T10 PI (Protection Information)
  • 8b per sector
  • format defined by T10
• See also:
  • DIF, DIX, EEDP
T10 PI (theory)

For each logical block

- guard tag – checksum
- app tag – application meta
- ref tag – application meta (sometimes)
T10 PI (theory)
Kernel blk_integrity API

- register device as integrity capable:
  ```c
  int blk_integrity_register(gendisk, blk_integrity_profile);
  ```
- attach metadata:
  ```c
  struct bip * bio_integrity_alloc(bio, gfp_mask, nr_pages);
  int bio_integrity_add_page(bio, page, len, offset);
  ```
- supports custom metadata model
T10 PI (practice)

- SCSI FORMAT UNIT
  - `sg_format --format --size 4096 --fmtinfo=2 -vvv /dev/sda`
    - Type 1 “--fmtinfo=2”
    - Type 3 “--fmtinfo=3 --pfu=1”
    - 6 or 10 bytes?
  - No Type 3 support for NL SAS drives
  - 16 hours later...
T10 PI (practice)

Nothing happens :( however:

- lower capacity (-0.7%)
- SCSI stack considers device as Type 1 DIF
- still no integrity on block level:
  /sys/block/sda/integrity/device_is_integrity_capable -> 0
  /sys/block/sda/integrity/tag_size -> 0
sg_readcap -v --long /dev/sdc

read capacity(16) cdb: 9e 10 00 00 00 00 00 00 00 00 00 00 00 20 00 00

Read Capacity results:

Protection: prot_en=1, p_type=0, p_i_exponent=0 [type 1 protection]

Logical block provisioning: lbpme=0, lbprz=0

Last logical block address=2424569855 (0x9083ffff), Number of logical blocks=2424569856

Logical block length=4096 bytes

Logical blocks per physical block exponent=0

Lowest aligned logical block address=0

Hence:

Device size: 9931038130176 bytes, 9470976.0 MiB, 9931.04 GB
T10 PI (HBA)
T10 PI (HBA)

- SAS HBA may require explicit DIF enablement:
  example for LSI: options mpt3sas prot_mask=63
- block layer detects integrity format T10-DIF-TYPE1-CRC
- looks good, but tag_size is still zero:
  /sys/block/sda/integrity/tag_size -> 0
T10 PI (ATO)

- `tag_size` – application metadata size
- Control Mode Page (0x0A), ATO bit
  - `sg_wr_mode -v -p 0a -c 00,00,00,00,00,80,00,00,00,00,00 -m 00,00,00,00,80,00,00,00,00,00,00,00 /dev/sda`
  - use `--s` for persistence
T10 PI (driver)

• Everything in place but still nothing works:
  mpt3sas_cm0: log_info(0x31120309): originator(PL), code(0x12), sub_code(0x0309)
  mpt3sas_cm0: log_info(0x31120434): originator(PL), code(0x12), sub_code(0x0434)

• Driver version may be important (or vendor)
T10 PI (all together)

• It works!
  • Just need a correct format, drive settings, HBA settings and HBA driver
  • /sys/block/sda/integrity/device_is_integrity_capable -> 1
  • /sys/block/sda/integrity/tag_size -> 2
  • /sys/block/sda/integrity/format -> T10-DIF-TYPE1-CRC
• We can read and write data
NVMe PI

A copy of T10 PI

• optional
• configured as namespace format
• different namespaces with different formats
• firmware issues
• noop profile
nvme id-ns -H /dev/nvme1n1

... 
dpc  : 0x1f

[4:4] : 0x1 Protection Information Transferred as Last 8 Bytes of Metadata Supported
[3:3] : 0x1 Protection Information Transferred as First 8 Bytes of Metadata Supported
[2:2] : 0x1 Protection Information Type 3 Supported
[1:1] : 0x1 Protection Information Type 2 Supported
[0:0] : 0x1 Protection Information Type 1 Supported

...
dps  : 0x3

[3:3] : 0 Protection Information is Transferred as Last 8 Bytes of Metadata
[2:0] : 0x3 Protection Information Type 3 Enabled

...
LBA Format 0 : Metadata Size: 0 bytes - Data Size: 512 bytes - Relative Performance: 0x1 Better
LBA Format 1 : Metadata Size: 8 bytes - Data Size: 512 bytes - Relative Performance: 0x3 Degraded
LBA Format 2 : Metadata Size: 0 bytes - Data Size: 4096 bytes - Relative Performance: 0 Best
LBA Format 3 : Metadata Size: 8 bytes - Data Size: 4096 bytes - Relative Performance: 0x2 Good (in use)
How to code with PI?

• Without specific hardware
  • even on laptop
• Run tests in CI
scsi target

data file
  + metadata file
  + fileio backend
  + loopback frontend

= scsi device with DIF on any hardware
loopback

```
data/md files
  ext4
  /dev/sda
/dev/sdb
  tcm_loop
  target_core_file
```
loopback

Block device

DIF enabled block device

data/md files

ext4

/dev/sda

target_core_file

tcm_loop

/dev/sdb

Linux VM
Tests in CI

• T10 PI devices for VMs
• component tests
  • NVMe/SCSI devices with PI support
• integration tests
  • VMs with shared drives
• external configuration
VM: NVMe PI

- PCI passthrough
- one VM per disk
  - dual port drives may help
  - or SR-IOV (in future?)
VM: SCSI PI

SAS disk -> SAS HBA -> /dev/sda

Host
VM: SCSI PI

SAS disk → SAS HBA → /dev/sda → target_core_iblock → tcm_vhost → virtio_scsi → /dev/sda

Host → VM
VM: SCSI PI

Host

SAS disk ➔ SAS HBA ➔ /dev/sda ➔ dm_linear ➔ target_core_iblock ➔ tcm_vhost ➔ virtio_scsi ➔ /dev/sda

VM1

VM2

Kernel 4.12+
vhost_scsi

• no DIF/DIX again:(
• virtio_scsi/vhost_scsi seem to support T10 PI
  • VIRTIO_SCSI_F_T10_PI feature bit
vhost_scsi

- no DIF/DIX again:(
- virtio_scsi/vhost_scsi seem to support T10 PI
  - VIRTIO_SCSI_F_T10_PI feature bit
  - qemu 3.1+
<domain>
... 
<devices>
...
<hostdev mode='subsystem' type='scsi_host' managed='no'>
  <source protocol='vhost' wwpn='naa.50014052ccc30dc0'/>
  <alias name='hostdev2'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
</hostdev>
</devices>

<qemu:commandline>
  <qemu:arg value='-set'/>
  <qemu:arg value='device.hostdev2.t10_pi=on'/>
</qemu:commandline>
</domain>
vhost_scsi

VIRTIO_SCSI_F_T10_PI negotiated and nothing works:

- Bad block number requested
- 4096b bio => 4040b
- 8192b bio => 8080b
- 56b stolen from each sector
commit cdcdaaee8450a975e7d07e1b6e21f9b8c0156d0c
Author: Greg Edwards <gedwards@ddn.com>
Date:   Thu Jul 26 15:52:54 2018 -0400

scsi: virtio_scsi: fix pi_bytes{out,in} on 4 KiB block size devices

When the underlying device is a 4 KiB logical block size device with a
protection interval exponent of 0, i.e. 4096 bytes data + 8 bytes PI, the
driver miscalculates the pi_bytes{out,in} by a factor of 8x (64 bytes).

This leads to errors on all reads and writes on 4 KiB logical block size
devices when CONFIG_BLK_DEV_INTEGRITY is enabled and the
VIRTIO_SCSI_F_T10_PI feature bit has been negotiated.
What else?

SPDK:

- NVMe bdev supports metadata
- vhost does not
- ppc64le memory issues
blk_integrity API (practice)

Common error: - EILSEQ, Logical block reference tag check failed
  • in case of incorrect reference tag
  • LBA lower bits for PI Type 1
  • Linux 4k block issues
## blk_integrity API (practice)

### 512b sector (bio)

<table>
<thead>
<tr>
<th></th>
<th>data (LBA x)</th>
<th>guard</th>
<th>app</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>data (LBA x + 1)</td>
<td>guard</td>
<td>app</td>
<td>x + 1</td>
</tr>
<tr>
<td></td>
<td>data (LBA x + 2)</td>
<td>guard</td>
<td>app</td>
<td>x + 2</td>
</tr>
<tr>
<td></td>
<td>data (LBA x + 3)</td>
<td>guard</td>
<td>app</td>
<td>x + 3</td>
</tr>
</tbody>
</table>

### 512b sector (SCSI)

<table>
<thead>
<tr>
<th></th>
<th>data (LBA x)</th>
<th>guard</th>
<th>app</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>data (LBA x + 1)</td>
<td>guard</td>
<td>app</td>
<td>x + 1</td>
</tr>
<tr>
<td></td>
<td>data (LBA x + 2)</td>
<td>guard</td>
<td>app</td>
<td>x + 2</td>
</tr>
<tr>
<td></td>
<td>data (LBA x + 3)</td>
<td>guard</td>
<td>app</td>
<td>x + 3</td>
</tr>
</tbody>
</table>
### blk_integrity API (practice)

#### 4k sector (bio)

<table>
<thead>
<tr>
<th>guard</th>
<th>app</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>data (LBA x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data (LBA x + 8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x + 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data (LBA x + 16)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x + 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data (LBA x + 24)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x + 3</td>
</tr>
</tbody>
</table>

#### 4k sector (SCSI) \( y = x/8 \)

<table>
<thead>
<tr>
<th>guard</th>
<th>app</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>data (LBA x/8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x/8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data (LBA x/8 + 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x/8 + 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data (LBA x/8 + 2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x/8 + 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data (LBA x/8 + 3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x/8 + 3</td>
</tr>
</tbody>
</table>

*reftag remap*
blk_integrity API (practice)

Common error: - EILSEQ

• request greater than 126 64k pages in virtio_scsi
blk_integrity API (practice)

Common error: - EILSEQ
- request greater than 126 64kB pages in virtio_scsi
- or more than 32 pages in nvme
  - request split in scheduler
  - /sys/block/sda/queue/max_segments
blk_integrity API (practice)

4k sector (bio)

<table>
<thead>
<tr>
<th>data (LBA x)</th>
<th>guard</th>
<th>app</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>data (LBA x + 8)</td>
<td>guard</td>
<td>app</td>
<td>x +1</td>
</tr>
<tr>
<td>data (LBA x + 16)</td>
<td>guard</td>
<td>app</td>
<td>x +2</td>
</tr>
<tr>
<td>data (LBA x + 24)</td>
<td>guard</td>
<td>app</td>
<td>x +3</td>
</tr>
</tbody>
</table>

4k sector (bio) split

<table>
<thead>
<tr>
<th>data (LBA x)</th>
<th>guard</th>
<th>app</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>data (LBA x + 8)</td>
<td>guard</td>
<td>app</td>
<td>x +1</td>
</tr>
<tr>
<td>data (LBA x + 16)</td>
<td>guard</td>
<td>app</td>
<td>x +2</td>
</tr>
<tr>
<td>data (LBA x + 24)</td>
<td>guard</td>
<td>app</td>
<td>x +3</td>
</tr>
</tbody>
</table>

4k sector (SCSI) y = x/8

<table>
<thead>
<tr>
<th>data (LBA x/8)</th>
<th>guard</th>
<th>app</th>
<th>x/8</th>
</tr>
</thead>
<tbody>
<tr>
<td>data (LBA x/8 + 1)</td>
<td>guard</td>
<td>app</td>
<td>x/8 +1</td>
</tr>
<tr>
<td>data (LBA x/8 + 2)</td>
<td>guard</td>
<td>app</td>
<td>x/8 +2</td>
</tr>
<tr>
<td>data (LBA x/8 + 3)</td>
<td>guard</td>
<td>app</td>
<td>x/8 +3</td>
</tr>
</tbody>
</table>
blk_integrity API (practice)

Common error: - EILSEQ

- request greater than 126 64k pages in virtio_scsi
- or more than 32 pages in nvme
  - request split in scheduler
  - or multiple metadata pages (nvme)
  - single MD page for PRP
  - /sys/block/sda/queue/max_integrity_segments
blk_integrity API (practice)

Common error: - EILSEQ

- request greater than 126 64k pages in virtio_scsi
- or more than 32 pages in nvme
  - request split in scheduler
  - or multiple metadata pages (nvme)
  - single MD page for PRP
- or segment size >4k (nvme)
Performance

Guard tag must contain checksum

• CRC16 or IP
• NVMe supports only CRC16
• crct10dif – 200MB/s per Power8 core, alternatives:
  crct10dif-pclmul (x86_64)
  crct10dif-vpmsum 4.12+ (ppc64)
  crct10dif-arm64-neon 5.1+
  crct10dif-arm64-ce 4.10+
Summary

Works:

- T10 DIF/DIX на SCSI, NVMe PI (carefully)
- Qemu/KVM: virtio_vhost/virtio_scsi
- Device mapper (kernel 4.12+)

All above helped to deliver software with block-level integrity

Nice to have:

- NVMe PI (complex setups),
- userspace DIF API, DIF support in SPDK vhost
Questions?