DIF/DIX Aware Linux SCSI HBA Interface

Martin K. Petersen, Oracle Linux Engineering
martin.petersen@oracle.com

July 16, 2008

1 Discovery

1.1 SCSI Host DIF Capability

A SCSI host bus adapter driver registers itself with the Linux SCSI layer by way of a `scsi_host_template`. The template includes information about the controller's capabilities in general (scatter-gather list lengths, etc.). After successful registration using the template, a `scsi_host` struct is created. This structure contains information used by the SCSI layer at runtime to represent a controller's state. A new field has been added this struct. An HBA driver can use the following call to indicate its DIF capabilities:

```
void scsi_host_set_prot(struct scsi_host *, unsigned int mask)
```

The valid values for the mask are:

Flag	Indicates
SHOST_DIF_TYPE1_PROTECTION	HBA supports DIF Type 1
SHOST_DIF_TYPE2_PROTECTION	HBA supports DIF Type 2
SHOST_DIF_TYPE3_PROTECTION	HBA supports DIF Type 3
SHOST_DIX_TYPE0_PROTECTION	HBA supports DIX Type 0
SHOST_DIX_TYPE1_PROTECTION	HBA supports DIX Type 1
SHOST_DIX_TYPE2_PROTECTION	HBA supports DIX Type 2
SHOST_DIX_TYPE3_PROTECTION	HBA supports DIX Type 3

Table 1: `scsi_host_prot_capabilities`

It is anticipated that some HBA vendors will produce controllers capable of doing DMA of protection data to and from host memory. The interface for doing so is described in a separate document[1]. DIX refers to the Data Integrity Extensions defined in that specification.

DIX Type 0–3 are identical with DIF with a few notable exceptions:

- DIX Type 0 is for use with legacy (non-DIF) devices and implies the use of `READ_INSERT` and `WRITE_STRIP`.
- DIX Type 0 is identical to DIF/DIX Type 1 except the contents of the `application` tag is undefined.

[1] I/O Controller Requirements for Data Integrity Aware Operating Systems

- The format of the guard tag can optionally be the IP checksum instead of the CRC mandated by T10 DIF.

Kernel subsystems can use the following call to determine whether an HBA supports DMA of protection information:

```
unsigned char scsi_host_get_prot(struct scsi_host *shost)
```

Kernel subsystems can use the following calls to determine whether an HBA supports a given DIF/DIX protection type:

```
unsigned int scsi_host_dif_capable(struct scsi_host *shost, unsigned
int target_type)

unsigned int scsi_host_dix_capable(struct scsi_host *shost, unsigned
int dix_type)
```

User applications can get the same information by reading:

```
/sys/class/scsi_host/hostN/prot_capabilities
```

For performance reasons controllers may support guard tag formats other than the T10 defined CRC. Another bitfield, prot_guard_type, has been introduced in the scsi_host structure which can be used to indicate the HBA's checksumming capabilities.

The value must be set using:

```
void scsi_host_set_guard(struct scsi_host *shost, unsigned char type)
```

Valid values are:

Flag	Indicates
SHOST_DIX_GUARD_CRC	HBA supports T10 DIF CRC
SHOST_DIX_GUARD_TCP	HBA supports TCP/IP checksumming

Table 2: scsi_host_guard_type

Kernel subsystems can inspect this to determine whether an HBA supports the given DIF protection type or not using the following call:

```
unsigned char scsi_host_get_guard(struct scsi_host *shost)
```

User applications can get the same information by reading:

```
/sys/class/scsi_host/hostN/prot_guard_type
```

1.2 Generic SCSI Device DIF Capability

When a SCSI host is being added or when the user explicitly requests it, a SCSI bus can be scanned. A standard INQUIRY will be sent to all devices.

1.3 SCSI Disk DIF Capability

If a `scsi_device` reports `DISK` in its `INQUIRY PERIPHERAL DEVICE TYPE` field it is passed on to the Linux SCSI Disk driver (`sd.c`). If the device has the `INQUIRY PROTECT` bit set, a `READ CAPACITY(16)` will be sent instead of the usual 10-byte command. The driver will look at the `PROT_EN` field in the returned parameter data to determine whether the disk is formatted with protection information or not. If it is, it will extract the type of protection from the `P_TYPE` field.

A series of checks will then be performed to verify that the HBA supports the right type of protection. This is done by matching the `P_TYPE` field to the `scsi_host`'s `prot_capabilities`.

Subsequently the `scsi_disk` struct is updated, setting the `protection_type` field to the type the drive is formatted with.

Kernel users can check the `protection_type` variable to determine whether a disk supports DIF or not. User applications can get the same information by reading:

`/sys/class/scsi_disk/`H`:`C`:`I`:`L`/protection_type`

The following constants are available within the kernel:

Flag
`SD_DIF_TYPE0_PROTECTION`
`SD_DIF_TYPE1_PROTECTION`
`SD_DIF_TYPE2_PROTECTION`
`SD_DIF_TYPE3_PROTECTION`

Table 3: `scsi_disk protection_type`

2 SCSI Command

T10 DIF is not defined for the 6-byte commands. So if the disk has a protection type > 0, we enforce the use of 10 byte or greater CDBs.

The Linux SCSI layer submits I/O to the HBA driver via the `queuecommand` function listed in the driver's `scsi_host_template`. The first argument to this function is a `scsi_cmnd` structure which among other things contains the CDB targeted for the device.

2.1 Protection Scatter-Gather

`scsi_cmnd` has been extended with a pointer to a struct `scatterlist` containing pointers to protection data tuples in 8-byte T10 DIF format. Each tuple is network endian.

If `scsi_prot_sg_count()` returns > 0 the driver will need to use this information when building the command bound for the HBA. The pages pointed to by `scsi_prot_sglist` will abide by the same DMA addressability and alignment constraints as the data scatter-gather list. The `scsi_for_each_prot_sg()` should be used to walk the list.

Unlike the data buffer there are no limits enforced at merge time on the number of elements in the protection scatter-gather list. In the common scenario there will be one protection scatter element for each physical page in the data buffer. If the HBA

has flagged that it supports clustering of adjacent data pages, that implies that the protection scatter-gather list could potentially be longer than the data list.

2.2 ATO

The ATO (App Tag Own) bit in the CONTROL MODE PAGE toggles whether the storage device or the host owns the contents of the app tag. If ATO is set, Linux expects to be able to store and retrieve values in the app_tag.

3 HBA Driver I/O Submission

If the target's protection_type is > 0, the SCSI disk driver will adjust the CDB accordingly, filling out the RDPROTECT and WRPROTECT fields.

Note that a DIF capable CDB will be sent in the scsi_cmnd *regardless* of whether the HBA supports protection info DMA (DIX). The CDB controls the communication between initiator and target.

Similarly, a valid protection scatterlist may be sent to the HBA even if the target has no DIF capability. The scatterlist controls the communication between Linux and initiator. Since there is no target-mandated protection type in this case, the format is defined to be identical Type 1 (See DIX Type 0 above) and the HBA must verify the guard and ref tags.

3.1 Helper Functions

A set of helper functions are provided to help the HBA driver submit the request correctly.

- scsi_prot_sglist(struct scsi_cmnd *) returns the scatterlist for the protection buffer.
- scsi_prot_sg_count(struct scsi_cmnd *) returns the number of scatterlist elements for the protection buffer.
- scsi_for_each_prot_sg(struct scsi_cmnd *, struct scatterlist *, int, int) walks the protection scatterlist taking chained lists into account.
- scsi_get_prot_op(struct scsi_cmnd *) returns the protection operation. See the following section.
- scsi_get_prot_type(struct scsi_cmnd *) returns the DIF type of the target device.
- scsi_get_lba(struct scsi_cmnd *) returns the target LBA for a READ/WRITE command.

3.2 DIX Commands

Each scsi_cmnd passed to the HBA driver will have the protection operation field set to one of the following values. See the DIX specification for more information.

Flag
SCSI_PROT_NORMAL
SCSI_PROT_READ_INSERT
SCSI_PROT_WRITE_STRIP
SCSI_PROT_READ_STRIP
SCSI_PROT_WRITE_INSERT
SCSI_PROT_READ_PASS
SCSI_PROT_WRITE_PASS
SCSI_PROT_READ_CONVERT
SCSI_PROT_WRITE_CONVERT

Table 4: scsi_prot_operations

3.3 Error Handling

The DIF specification adds three new Additional Sense Code/Additional Sense Code Qualifiers, namely:

ASC	ASCQ	Description
0x10	0x01	LOGICAL BLOCK GUARD CHECK FAILED
0x10	0x02	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
0x10	0x03	LOGICAL BLOCK REFERENCE TAG CHECK FAILED

Table 5: Additional Sense Codes + Qualifiers

SBC mandates that a storage device that experiences a protection information error shall return ABORTED COMMAND (0xB) with the appropriate additional sense codes from the table above. Given that the specification has no means for reporting errors between the operating system and the HBA, we will use the correct additional sense code paired with ILLEGAL REQUEST (0x5) to identify to the kernel that the HBA detected the corruption.

In case of a mismatch between CDB, data and protection data, the HBA driver must abort the command, fill out the sense buffer accordingly and set the driver status byte to DRIVER_SENSE and the host status byte DID_ABORT in scsi_cmnd's result. This indicates that bad data has been submitted and an error will be propagated up through the I/O stack to redrive the command.

The HBA driver must return a sense buffer with the appropriate ASC/ASCQ pair. The INFORMATION field in the sense data must contain the LBA where the error occurred,

3.3.1 Use Cases

Use cases for DIF-aware I/O:

Command	Protection Buffer	Condition	HBA to Disk	Disk to HBA	HBA to Host
READ	NULL	OK	-	NO SENSE + Data	NO SENSE + Data
READ	Provided	OK	-	NO SENSE + Data + Protection Data	NO SENSE + Data + Protection Data
READ	Provided	Disk or HBA: Sector data and crc tag mismatch	-	ABORTED COMMAND, GUARD CHECK FAILED	ABORTED COMMAND, GUARD CHECK FAILED
READ	Provided	Disk or HBA: app tag mismatch (undefined)	-	ABORTED COMMAND, APP. TAG CHECK FAILED	ABORTED COMMAND, APP. TAG CHECK FAILED
READ	Provided	Disk or HBA: CDB sector and ref tag mismatch	-	ABORTED COMMAND, REF. TAG CHECK FAILED	ABORTED COMMAND, REF. TAG CHECK FAILED

Table 6: Use Cases: Read

Command	Protection Buffer	Condition	HBA to Disk	Disk to HBA	HBA to Host
WRITE	NULL	OK	Data	NO SENSE	NO SENSE
WRITE	Provided	OK	Data + Protection Data	NO SENSE	NO SENSE
WRITE	Provided	Disk: Sector data and crc tag mismatch	Data + Protection Data	ABORTED COMMAND, GUARD CHECK FAILED	ABORTED COMMAND, GUARD CHECK FAILED
WRITE	Provided	Disk: app tag mismatch (undefined)	Data + Protection Data	ABORTED COMMAND, APP. CHECK FAILED	ABORTED COMMAND, APP. CHECK FAILED
WRITE	Provided	Disk: CDB sector and ref tag mismatch	Data + Protection Data	ABORTED COMMAND, REF. CHECK FAILED	ABORTED COMMAND, REF. CHECK FAILED
WRITE	Provided	HBA: Sector data and crc tag mismatch	Abort Command	ABORTED COMMAND	ILLEGAL REQUEST, GUARD CHECK FAILED, DID_ABORT
WRITE	Provided	HBA: app tag mismatch (undefined)	Abort Command	ABORTED COMMAND	ILLEGAL REQUEST, APP. CHECK FAILED, DID_ABORT
WRITE	Provided	HBA: CDB sector and ref tag mismatch	Abort Command	ABORTED COMMAND	ILLEGAL REQUEST, REF. CHECK FAILED, DID_ABORT

Table 7: Use Cases: Write

Changes

Date	Change
2008-07-16	Helper functions for protection op and type as well as start LBA
2008-07-09	Bring in sync with Linux kernel submissions
2008-03-31	Clarify protection scatterlist length
2008-01-10	Require protection info to be network-endian
2007-10-02	Initial version

Table 8: Document Revisions