I/O Controller Requirements for Data Integrity Aware Operating Systems

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

April 1, 2008

1 Introduction

In order to facilitate true end to end data integrity between application and storage device, integrity metadata must be made accessible to the operating system.

We have developed a model in which the T10 Data Integrity Field protection data can be transferred to and from the operating system using standard DMA.

While the functionality of the OS-level interfaces can be implemented on top of most currently available controllers, the interfaces have been designed using some features that drastically improve performance. Most of these features are optional but if they are not present they put additional load on the system processor. We encourage vendors to implement the full feature set described below.

1.1 Scatter-Gather List Separation

On the wire between initiator and target T10-SBC mandates that the protection information is interleaved with the data sectors. For instance 512 bytes of data[1] followed by 8 bytes of protection information. This is very inconvenient for the operating system which usually stores application data in 4KB pages. Doing memory copies to interleave data and protection data wastes system resources. Our benchmarking shows that interleaving by way of scatterlists also impacts system performance.

Consequently, in our model interleaving of data and protection information has been delegated to the I/O controller. The controller will receive a request with two scatter-gather lists attached. One containing the data in/out buffer as usual and one containing the matching protection data in/out buffer.

The scatter-gather list elements of the protection buffer will honor the same alignment constraints as those of the data buffer.

1.2 Protection Data Endianness

The DIF protection data format is network-endian as per T10-SBC. This includes the protection data exchanged with the host.

[1] or 4KB with the upcoming support for bigger sector sizes

1.3 Guard Tag Format

The format of the guard tag between initiator and target is specified in the T10 standard as a cyclic redundancy check using a well-defined polynomium. A CRC is very expensive to calculate in software and benchmarks show that it has a significant impact on the performance of a system.

Various algorithms were tested and the IP checksum was selected for the first implementation of this. The IP checksum is cheap to calculate and also has a few other properties that made it suitable for this purpose.

It should be noted that the T10 CRC checksum is mandatory and the alternate guard tag format is an optional feature. An I/O driver, utility or management tool can indicate to the operating system whether to use the T10 DIF CRC, the IP checksum or (in the future) another format. The operating system is responsible for negotiating the best performing algorithm, in the case of for instance multipathing, using controllers with different checksum capabilities.

This negotiation capability is also used to pick a suitable checksum format when a software RAID device spans devices with different hardware sector size or guard tag capabilities.

The guard tag type is a per-request property, not a global setting.

1.4 Error Detection and Isolation

For error isolation purposes the I/O controller is expected to verify data integrity before passing it on. This is an optional feature when the T10 CRC is used. However, it is mandatory when any guard tag conversion is taking place.

In case of mismatch in the protection data the I/O controller must return a suitable error to the operating system. The error notification must include in which request, at which LBA the mismatch occurred, and whether it was a guard, reference or an application tag failure.

If a storage device returns a value of $0xFFFF$ in the application tag and the device is formatted with DIF Type 1 or 2 protection, the I/O controller must disable integrity checking for that sector.

Similarly, if a storage device returns a value of $0xFFFF$ in the application tag, a value of $0xFFFFFFFF$ in the reference tag and the device is formatted with DIF Type 3 protection, integrity checking must be disabled for that sector.

For all other errors the controller is expected to abort the I/O, report the error to the OS, and let the OS deal with (SPC/SBC-level) retries. Physical transport (SPI, SAS, FC) retries are still the responsiblity of the I/O controller.

1.5 The CDB and Request Routing

In the case of both READ and WRITE requests, the CDB passed to the I/O controller has a RDPROTECT/WRPROTECT field which indicates to the target whether to perform integrity verification or not. It also indicates whether to transfer protection data between initiator and target. Given the OS to I/O controller interface is outside the scope of T10, there is no similar controls for this piece of the I/O path.

The I/O controller also doesn't know, or care, whether a target supports protection information, which type of protection it is formatted with, etc.

The OS, which has this knowledge, will always prepare a CDB with appropriate RDPROTECT/WRPROTECT information, depending on target format and capabilities.

The request will also include information about which DIF protection type the target has been formatted with.

Finally, the OS will also provide the I/O controller driver with an operation code which tells the controller which type of I/O to perform. The operations are as follows:

Flag	OS-Controller	Controller-Target
NORMAL	Unprotected	Unprotected
READ_INSERT	Protected	Unprotected
WRITE_STRIP	Protected	Unprotected
READ_STRIP	Unprotected	Protected
WRITE_INSERT	Unprotected	Protected
READ_PASS	Protected	Protected
WRITE_PASS	Protected	Protected
READ_CONVERT	Protected	Protected
WRITE_CONVERT	Protected	Protected

Table 1: DIF I/O Operations

- NORMAL is for normal (unprotected) I/O
- READ_INSERT Read data from target, generate protection data and transfer both to OS
- WRITE_STRIP Transfer data and protection data from OS, verify data integrity, discard protection data and write data to target
- READ_STRIP Read data and protection data from target, verify data integrity, discard protection data and transfer data to OS
- WRITE_INSERT Transfer data from OS, generate protection data and write both data and protection data to target
- READ_PASS Read data + protection data from target, optionally verify protection data, transfer both data and protection data to OS
- WRITE_PASS Transfer data + protection data from OS, optionally verify protection data, write both data and protection data to target
- READ_CONVERT Read data + protection data from target, verify and convert the guard tags, transfer both data and protection data to OS
- WRITE_CONVERT Transfer data + protection data from OS, verify and convert the guard tags to T10 DIF CRC, write both data and protection data to target

The I/O driver can use these codes in combination with the guard tag format to prepare an appropriate request to the controller firmware.

Changes

Date	Change
2008-03-31	Clarify handling of uninitialized DIF data (0xFFFF in the application tag)
2008-01-10	Require protection info to be network-endian
2007-10-02	Initial version

Table 2: Document Revisions