Data Integrity in the Storage Stack

Or, it's 1:00 AM and Do You Know the Integrity of Your Data?

Jim Williams, Oracle Corporation, James.A.Williams@oracle.com
Martin Petersen, Oracle Corporation, Martin.Petersen@oracle.com

Agenda

- What is data corruption
- Dealing with data corruption
- Protection Information model
- Data Integrity Extensions and DMA of protection information
- Making Linux data integrity aware

What Is Data Corruption

- *Defined as the non-malicious loss of data resulting from component failure or inadvertent administrative action*
- Frequency and impact
  - Frequency low
  - Cost very high!
- Causes of data corruption
  - Hardware
  - Software
  - Administrative error

What Is Data Corruption

- At the storage level, there are two types of data corruption
  - Latent sector errors (application cannot read once valid data)
  - Silent data corruption (data read by application is not what was last written)
- Silent data corruption returns invalid data on a read operation, rather than a "failed I/O operation"
- SNIA's Data Integrity TWG focus is silent data corruption

What Is Data Corruption (What)

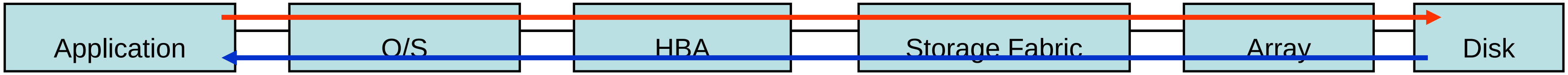
- There are four general types of data corruption
  - *Data Misplacement Errors*
    - Data is stored or retrieve from the wrong location or device
  - *Data Content Errors*
    - Data content is changed during its life
  - *Lost I/O Operations*
    - An apparent write operation is lost, but signaled complete
  - *Administrative Errors*
    - Sysadmin makes an error leading to destroyed data

What Is Data Corruption (When)

- The event of data corruption occurs at one of three stages in the life of data
  - Corruption can occur during the process of writing data
  - Corruption can occur during the process of reading data
  - Or corruption can occur while data is at rest
- It is usually not possible to know when and where corruption occurred

What Is Data Corruption (Where)

- Data corruption can occur at many places in the storage stack?
  - Application layer
  - Operating System
  - Host Bus Adapter (or any storage interface)
  - Storage Fabric
  - Storage Array
  - Hard Disk Drive



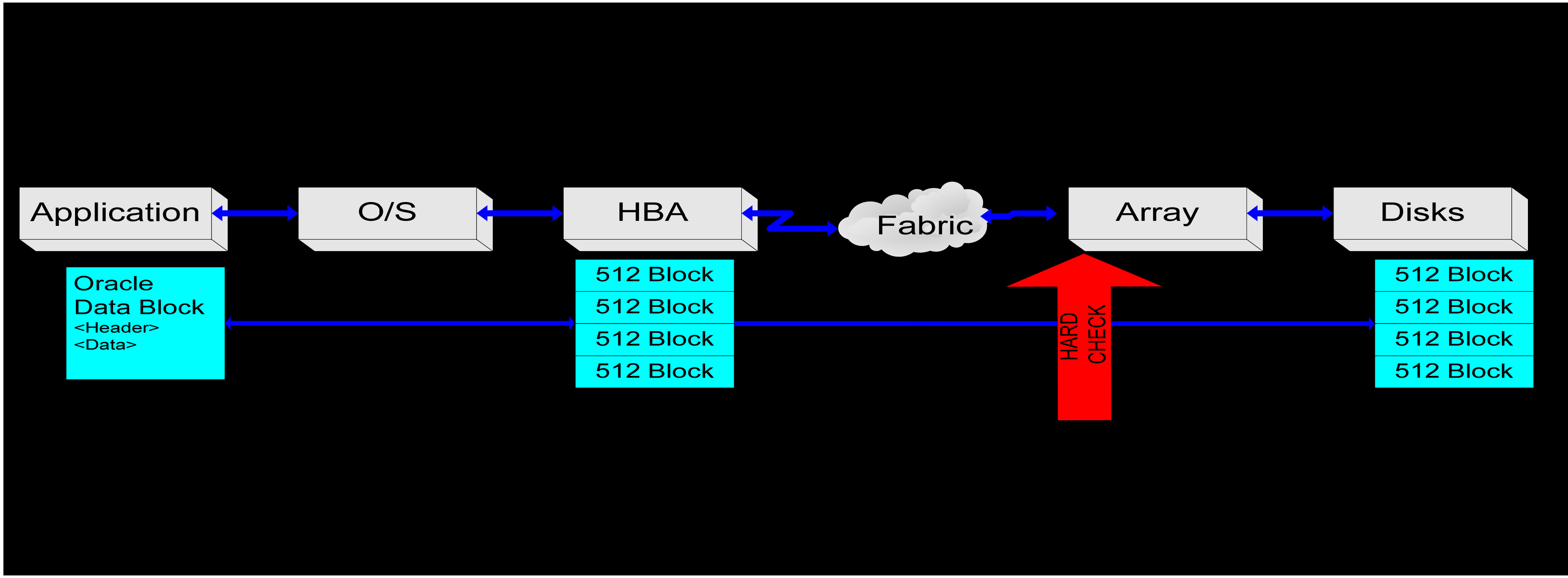
What Is Data Corruption (examples)

- Examples
  - O/S memory map failure leading to a data going to the wrong LBA
  - Lost write caused by storage array firmware
  - Admin error formatting wrong volume
  - O/S failure writing dump to wrong device on system crash
  - O/S memory mapping failure leading to a data being read from the wrong device

Dealing With Data Corruption

- Detection versus prevention (early detection)
  - An example of detection mechanism is the checksum residing in Oracle RDBMS data blocks. By itself, the checksum only enables the RDBMS to detect, during a read operation, when the data block has been corrupted somewhere in the storage stack.
  - An example of prevention is if the storage array understood the Oracle RDBMS data block structure and prevented corrupt data from being written to permanent storage.
- This is the concept behind Oracle HARD.
- Both prevention and detection are useful together.

Oracle HARD E2E Data Protection



On write operations, storage array validates written data. Data detected as invalid is rejected. It is up to the Oracle RDBMS to recover from the failed write operation.

Questions?

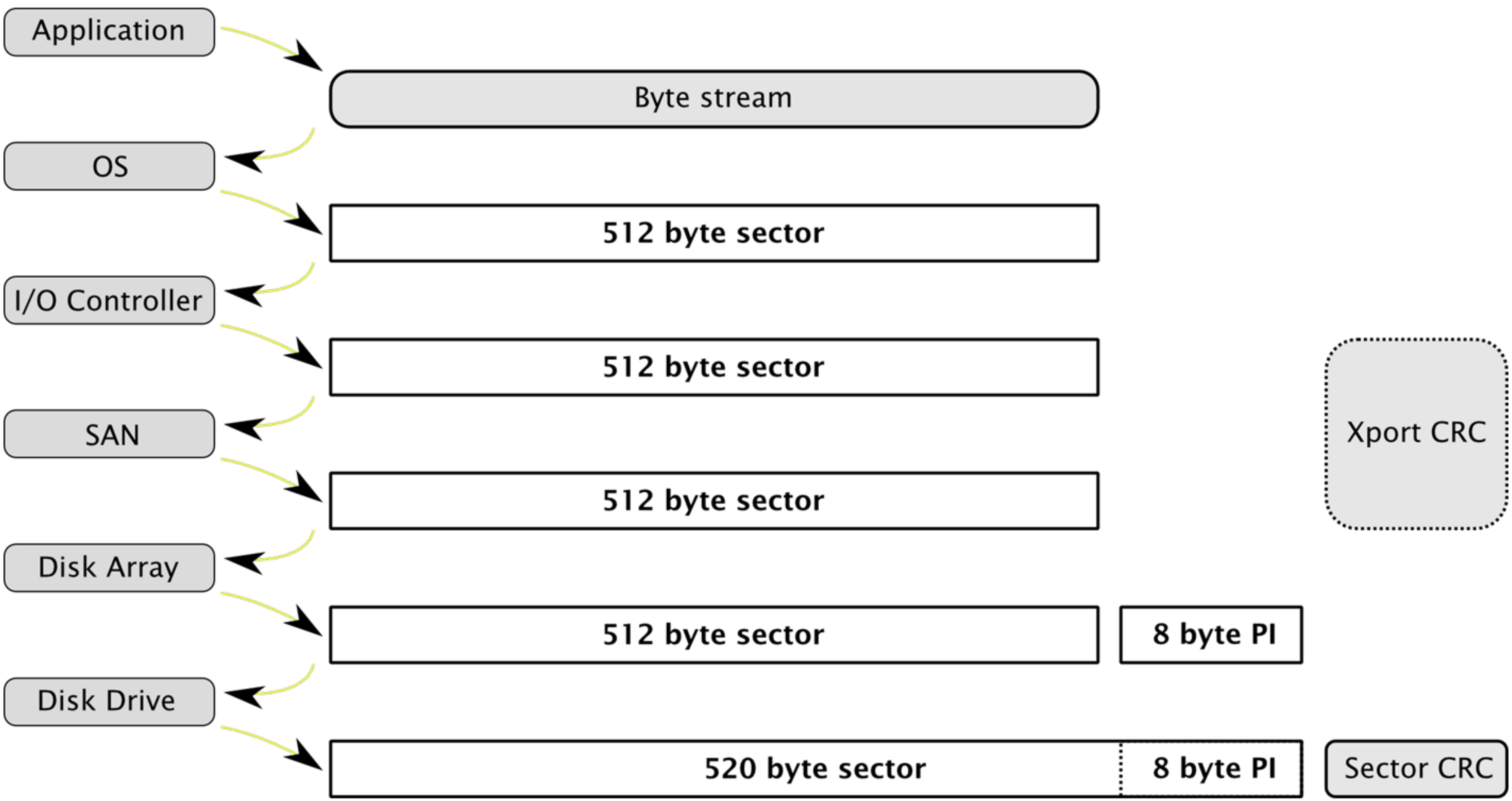
DIF and Data Integrity Extensions

Making Linux data integrity aware

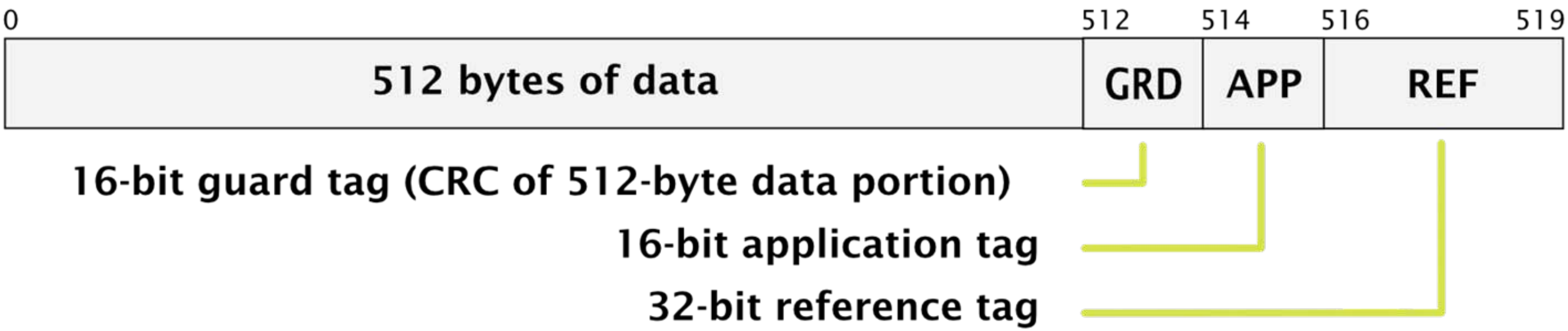
Disk Drives

- Most drives use 512-byte sectors although 4096-byte sectors are coming
- Each sector is protected by a proprietary cyclic redundancy check internal to the drive firmware
- Enterprise drives support 520/528 byte "fat" sectors
- Sector sizes that are not a a multiple of 512 have seen limited use because operating systems deal with units of 512
- RAID arrays make extensive use of "fat" sectors

Normal I/O

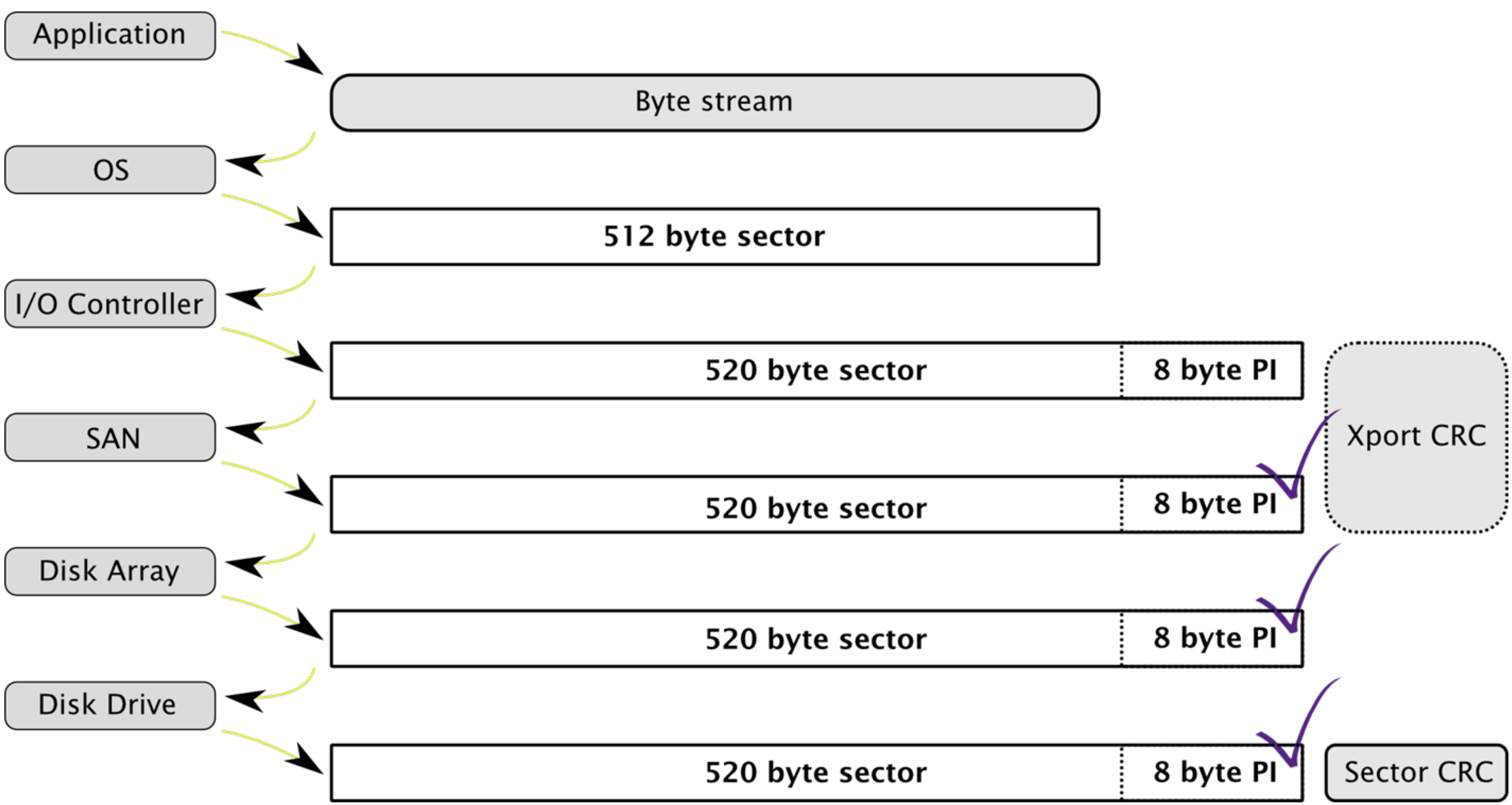

Application
Byte stream
OS
512 byte sector
I/O Controller
512 byte sector
Xport CRC
SAN
512 byte sector
Disk Array
512 byte sector
8 byte PI
Disk Drive
520 byte sector
8 byte PI
Sector CRC

T10 Data Integrity Field



- Only protects between HBA and storage device
- PI interleaved with data sectors on the wire
- Three protection schemes
  - All have a 16-bit CRC guard tag
  - Type 1 reference tag is lower 32 bits of target sector
  - Type 2 reference tag seeded in CDB
- SATA T13/EPP uses same format
- SCC tape proposal is different

T10 Data Integrity Field I/O



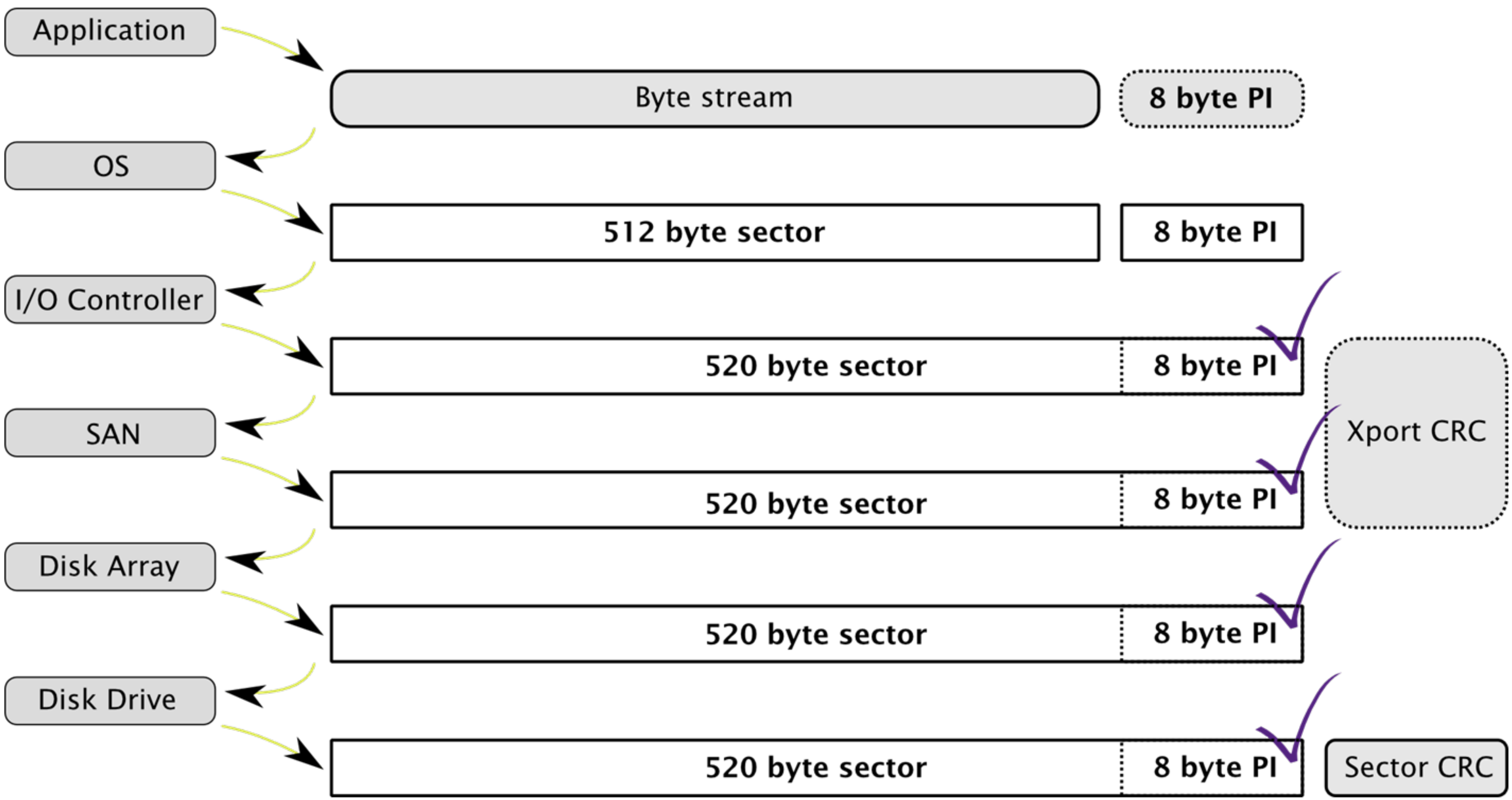
Data Integrity Extensions

- 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 commands for SCSI/SAS/FC controllers
- 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*

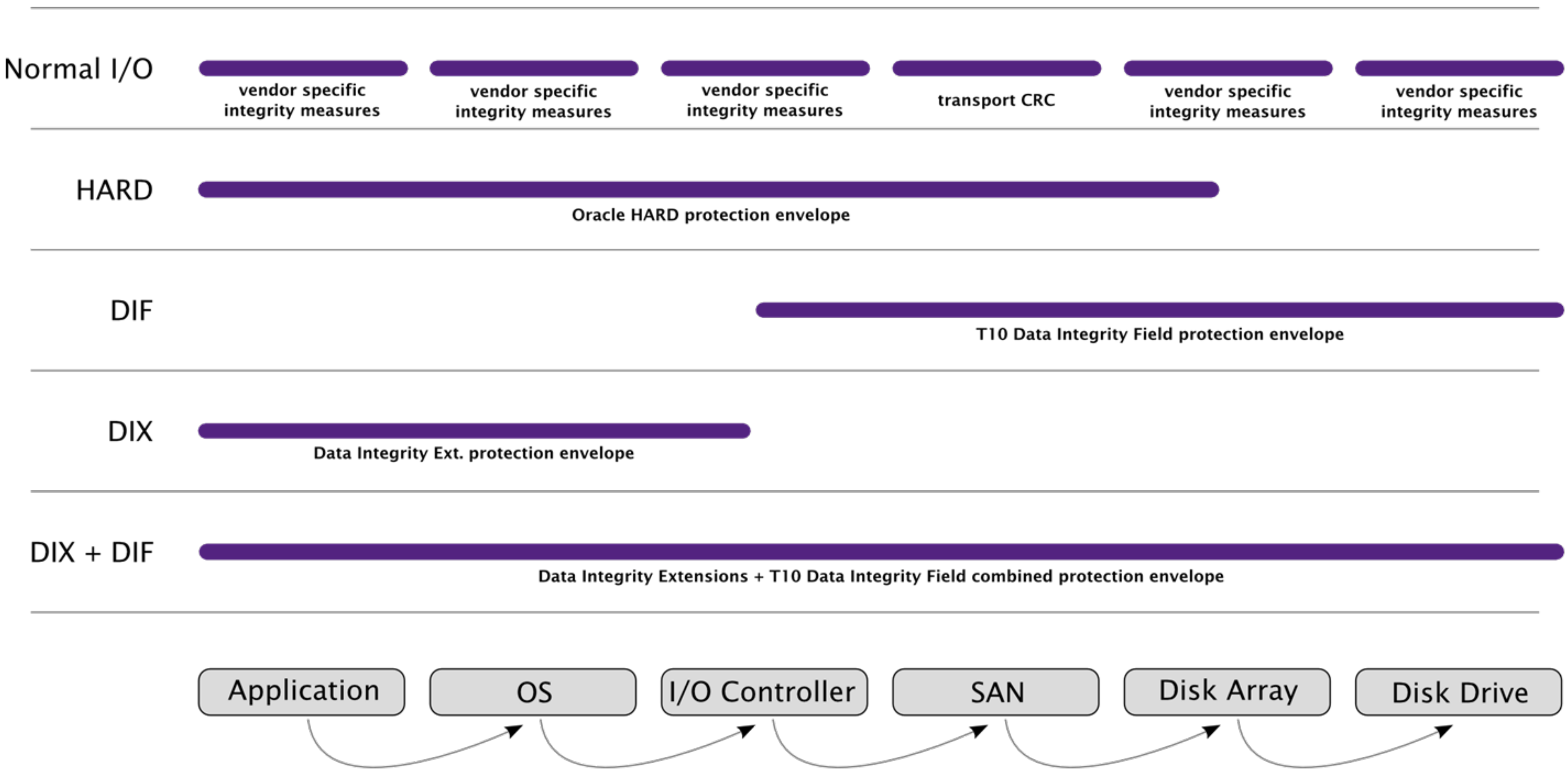
Data Integrity Extensions

- Separate protection scatter-gather list
  - 520-byte sectors are hard to deal with in a general purpose OS
  - <512, 8, 512, 8, 512, 8, …> does not perform well
- Checksum conversion
  - CRC16 is slow to calculate
  - IP checksum is fast and cheap
  - Optional feature
  - Strength is in data and protection information buffer separation

Data Integrity Extensions + DIF I/O



Protection Envelopes



Linux SCSI Layer

- Storage device discovery
  - DIF enabled?
  - Which protection type?
  - Application tag available (ATO bit)?
  - Protects path between initiator and target. CDB prepared accordingly.
- HBA registers DIX capability
  - Checksum formats supported
  - DIF and DIX modes supported
  - Allows exchange of protection information
  - SCSI requests will be submitted with a DIX operation telling HBA how to handle I/O

Linux Block Layer

- Basic I/O container extended with a separate scatter-gather list describing protection buffer
- Merge and splitting constraints
- Each block device has an integrity profile describing protection information must be prepared or verified (guard type, sector size, etc.)
- Filesystems can issue requests with protection information attached

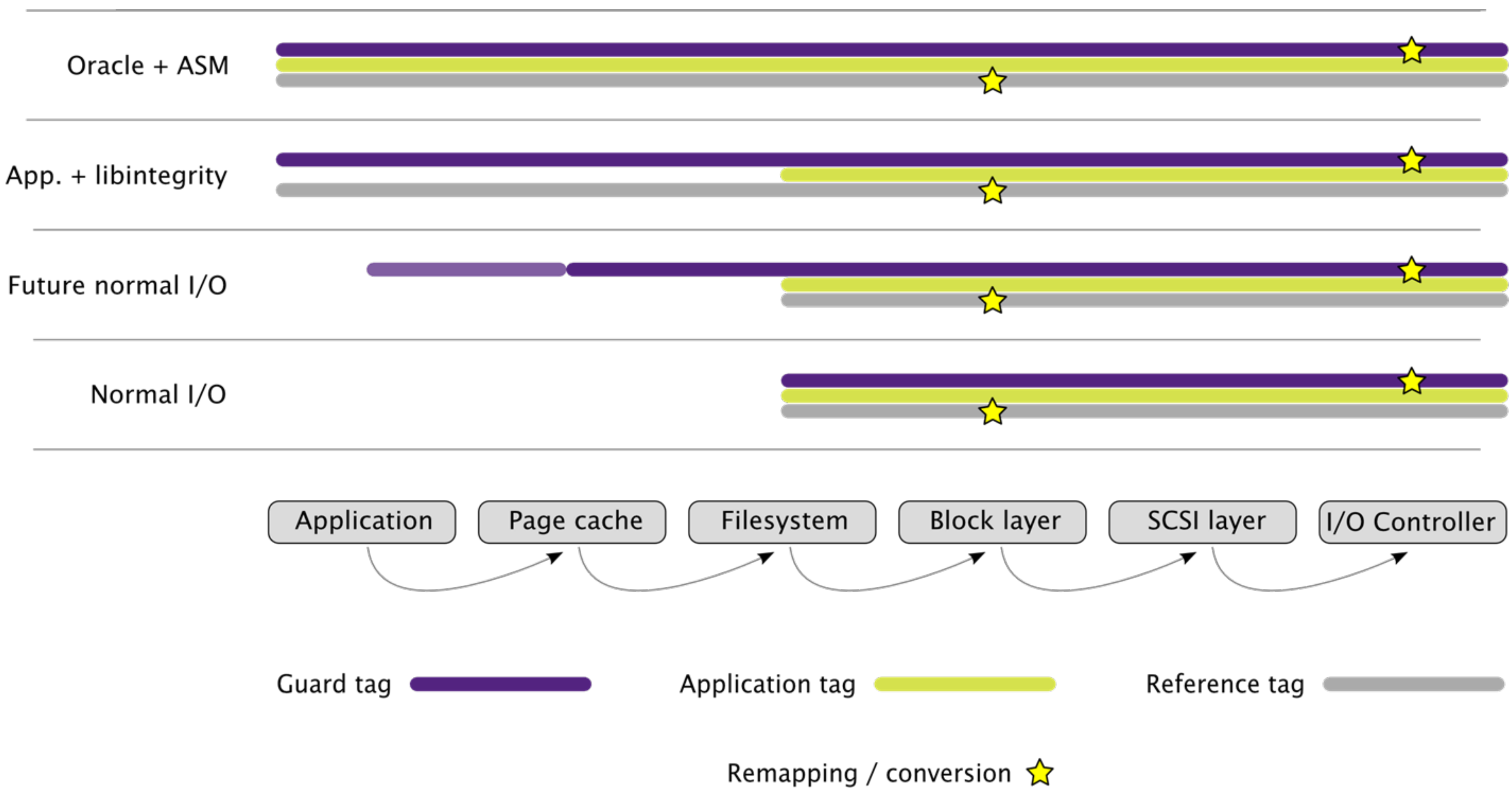
Linux Filesystems

- Can prepare protection information for WRITE commands and verify it for READs
- Details of the format are opaque to filesystem. Callback functions used to prepare and verify.
- Filesystems can use interleaved application tag space to implement checksumming without changing on-disk format
- Another possibility is to use the application tag space for back pointers, inode numbers, etc.

User Application Interfaces

- Any layer can add PI if not already present
- Owner of PI is responsible for re-driving failed requests
- FS/block layer transparently protects and verifies unprotected application I/O
- Most applications are not block oriented but deal with byte streams
- UNIX API poses some challenges (memory mapped I/O)

User Application Interfaces



Questions?