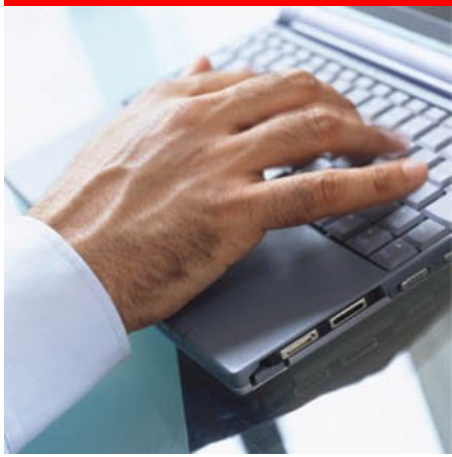

ORACLE®

ORACLE®

Linux Data Integrity

Martin K. Petersen
Consulting Software Developer, Linux Engineering

Topics



- DIF/DIX
  - Data Corruption
  - T10 DIF
  - Data Integrity Extensions
- Linux & Data Integrity
  - Block layer
  - Filesystems
  - User application Interfaces

DIF, DIX & Data Integrity

Data Corruption

- Tendency to focus on latent sector 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 admin / library / kernel / firmware errors
  - Examples: Bad buffer pointers, missing or misdirected writes
- Industry demand for end-to-end protection
  - Oracle HARD technology is widely deployed
  - Other databases and mission-critical business apps
  - Nearline/archival storage wants belt and suspenders

Data Corruption - Oracle HARD

- Hardware Assisted Resilient Data
  - Each database block has an internal checksum
  - Each database block also has an internal LBA check
  - Shipping since ~2001 on EMC/Hitachi/IBM Shark/NetApp
- Pro
  - Front-end of disk array can verify that Oracle logical blocks are intact
  - And in case of failure reject I/O on a logical block boundary
- Contra
  - Difficult to administer
  - Not all database I/O has a checksum
  - Proprietary and Oracle-specific ➔ Limited adoption
  - Expensive add-on, only very high-end arrays support it

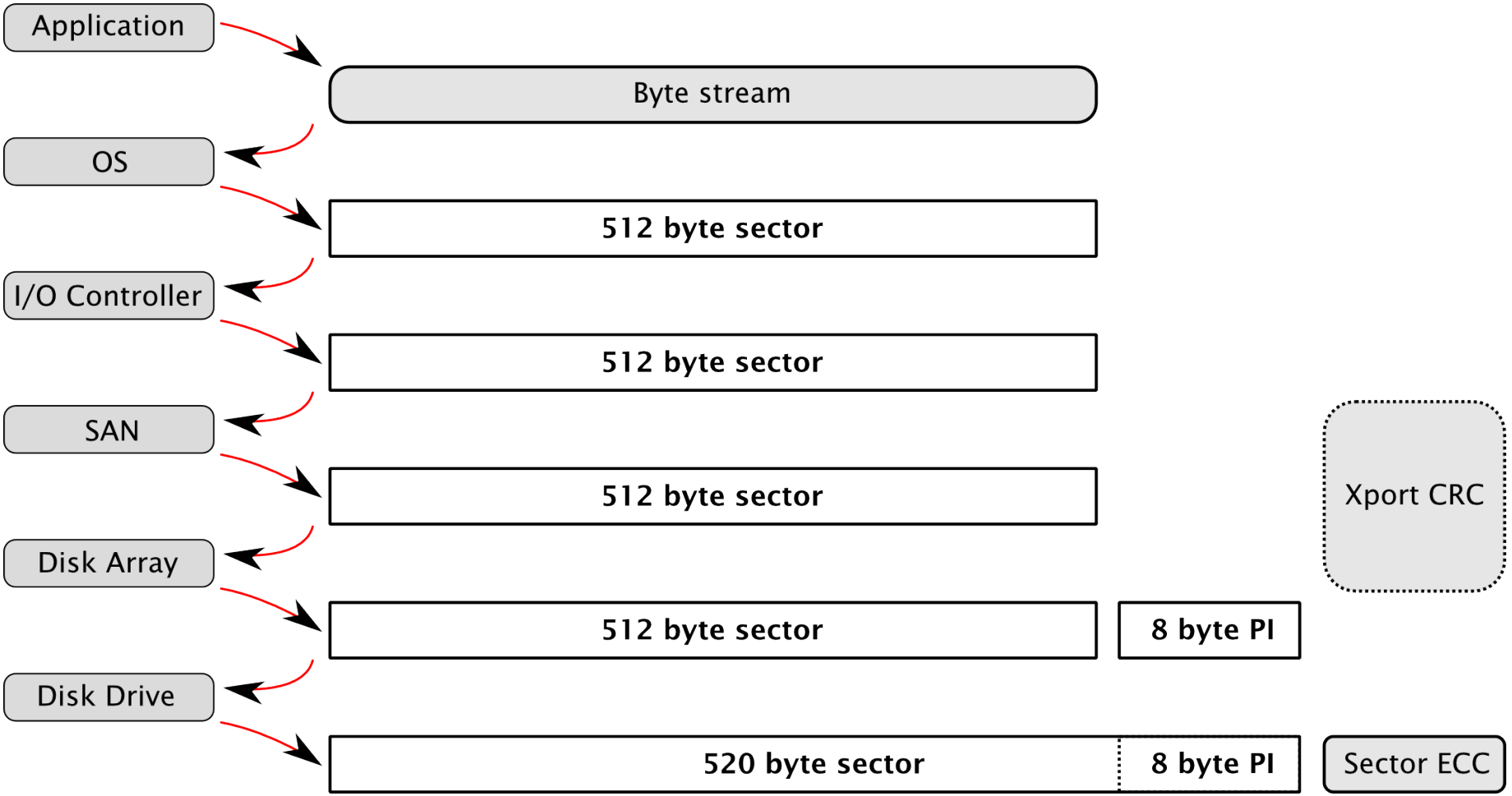
Data Corruption - DIF/DIX

- DIF/DIX are orthogonal to logical block checksums
  - btrfs/ZFS/Oracle database checksums are here to stay
  - Logical block checksums are used for *detection* of corrupted data at READ time
  - ... which could be months later ➔ Original, good buffer is lost
  - Logical block checksumming is a way to detect latent sector corruption
- DIF/DIX:
  - are about preventing *in-flight* corruption
  - tackle *content corruption* errors & *data misplacement* errors
  - allow us to detect problems before the original buffer is erased from memory
  - and before bad data ends up being stored on disk
  - Networks have had checksums for years. This is about time.

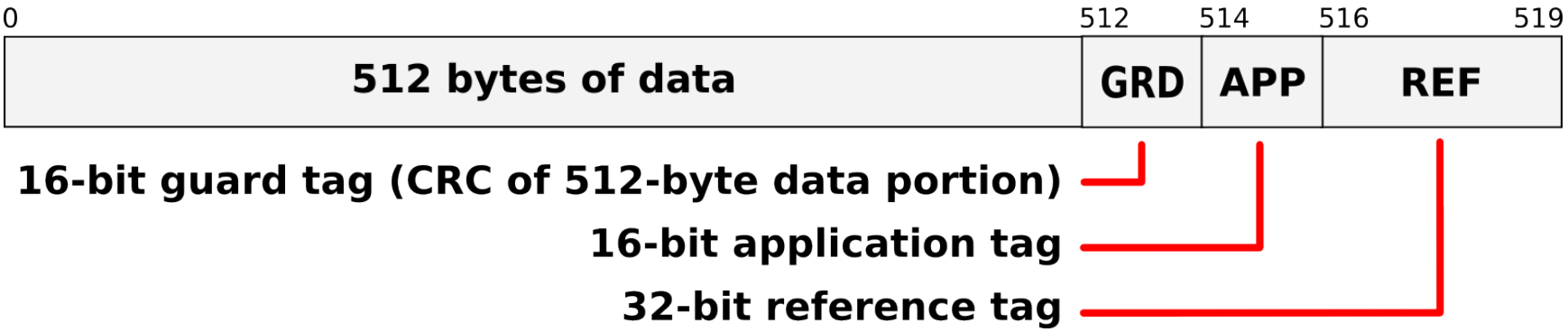
Disk Drives

- 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 ECC internal to the drive firmware
- 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 bytes
- RAID arrays make extensive use of fat sectors

Normal I/O

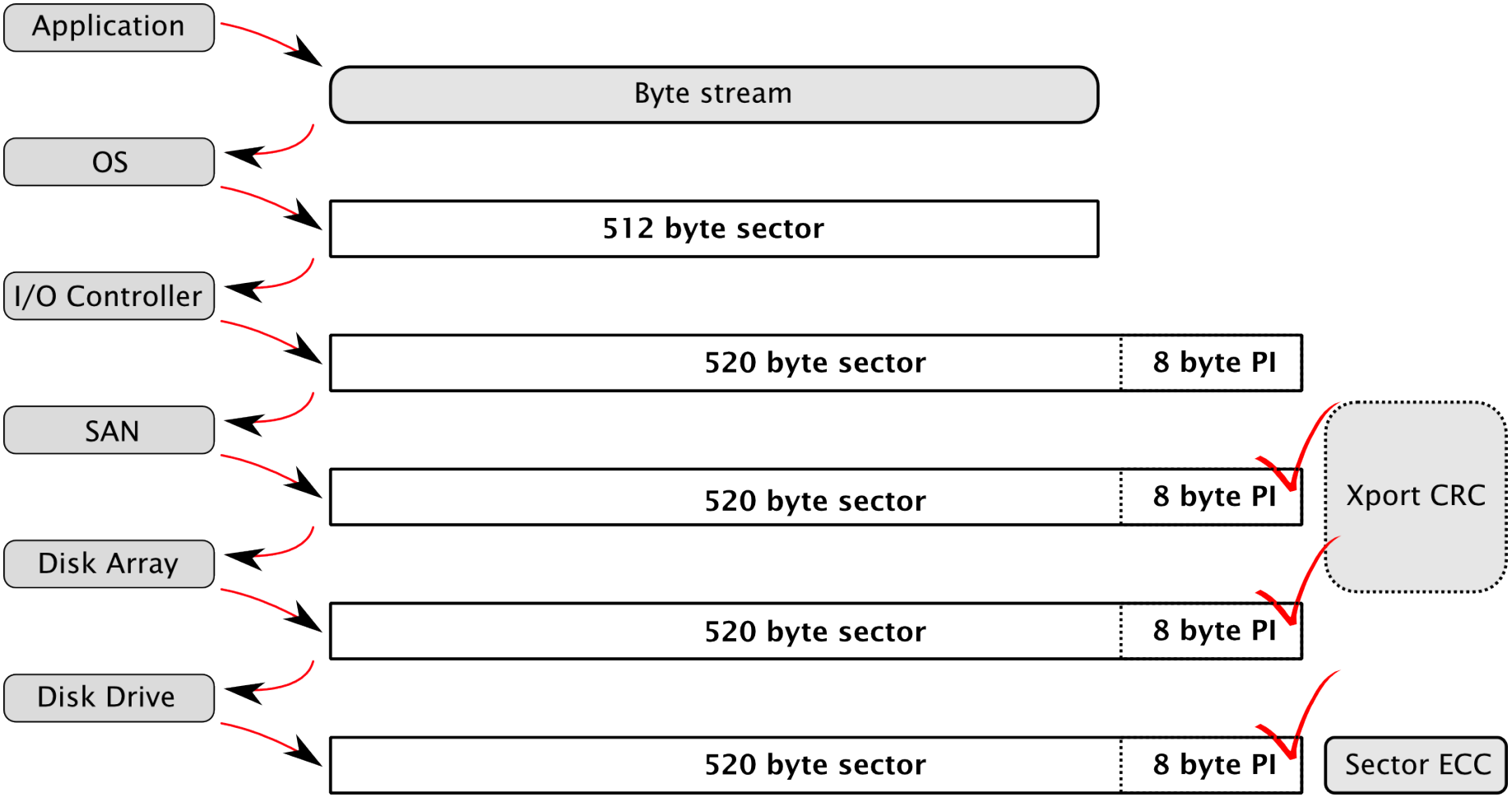

Application
Byte stream
OS
512 byte sector
I/O Controller
512 byte sector
SAN
512 byte sector
Xport CRC
Disk Array
512 byte sector
8 byte PI
Disk Drive
520 byte sector
8 byte PI
Sector ECC

T10 Data Integrity Field



- Standardizes those extra 8 bytes
- Prevents content corruption and misplacement errors
- Only protects path between HBA and storage device
- Protection information is interleaved with data on the wire, i.e. effectively 520-byte sectors
- SATA T13/External Path Protection proposal uses same protection information format
- SCSI tape proposal in the pipeline

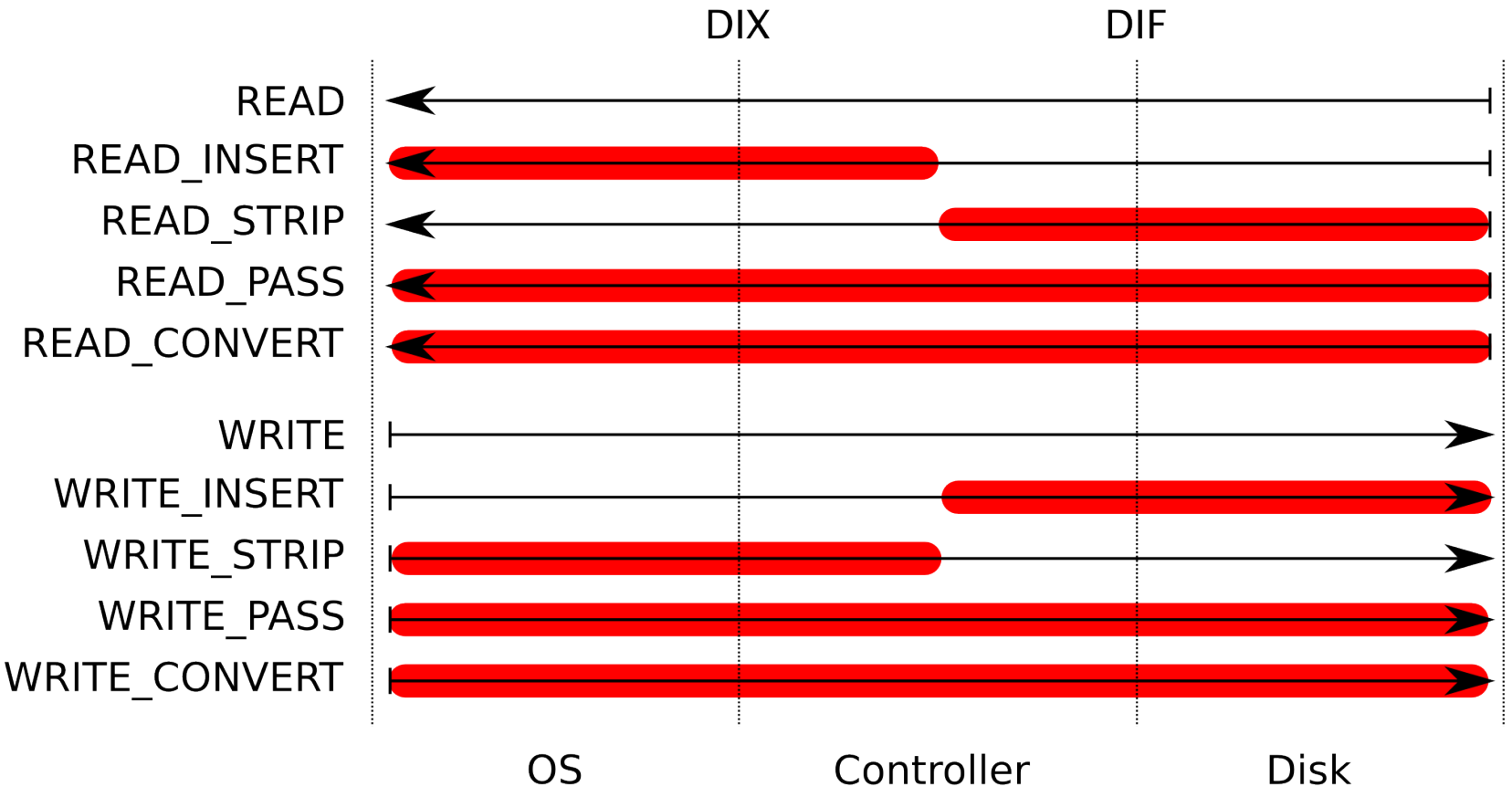
T10 Data Integrity Field I/O



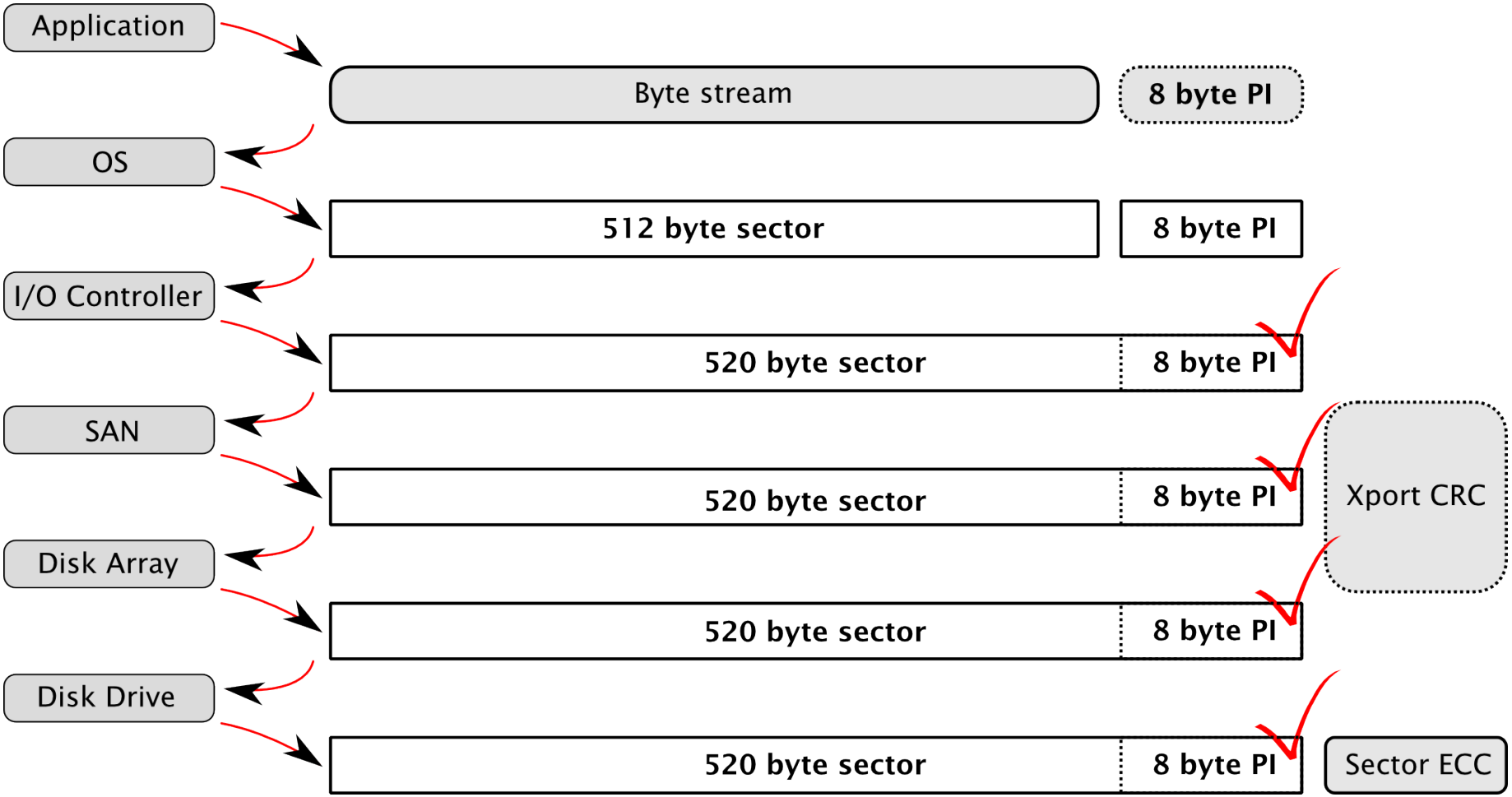
Data Integrity Extensions

- T10 DIF was a ratified, existing and open standard
- Attempt to extend DIF all the way up to the application, enabling true end-to-end data integrity protection
- Essentially a set of meta-commands for SCSI/SAS/FC controllers
- The Data Integrity Extensions:
  - Enable DMA transfer of protection information to and from host memory
  - Separate data and protection information buffers to avoid inefficient 512+8+512+8+512+8 scatter-gather lists
  - Provide a set of commands that tell HBA how to handle I/O:
    - *Generate, strip, pass, convert and verify*

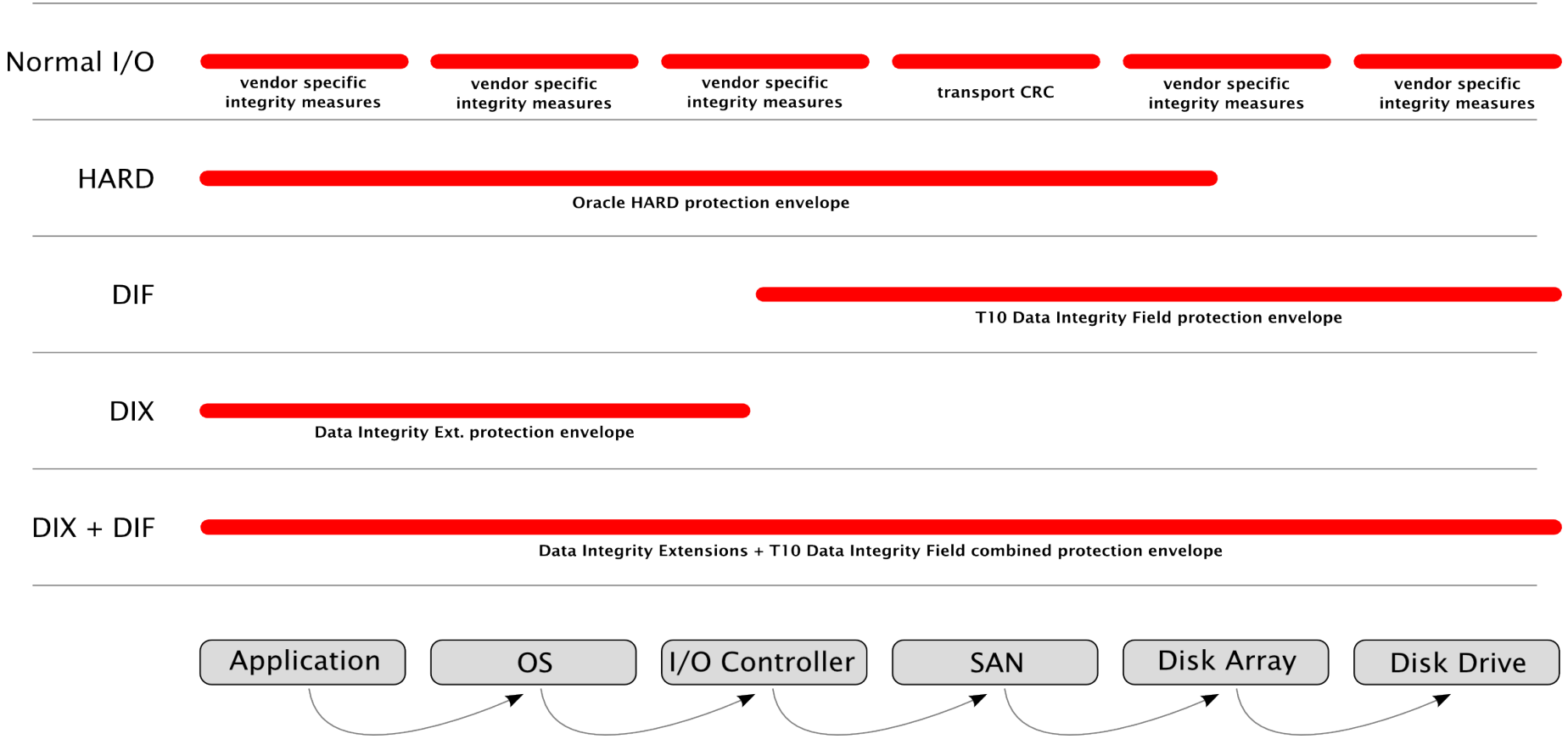
DIX Operations



Data Integrity Extensions + DIF I/O



Protection Envelopes



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
  - Product announcements soon

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

Linux & Data Integrity

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

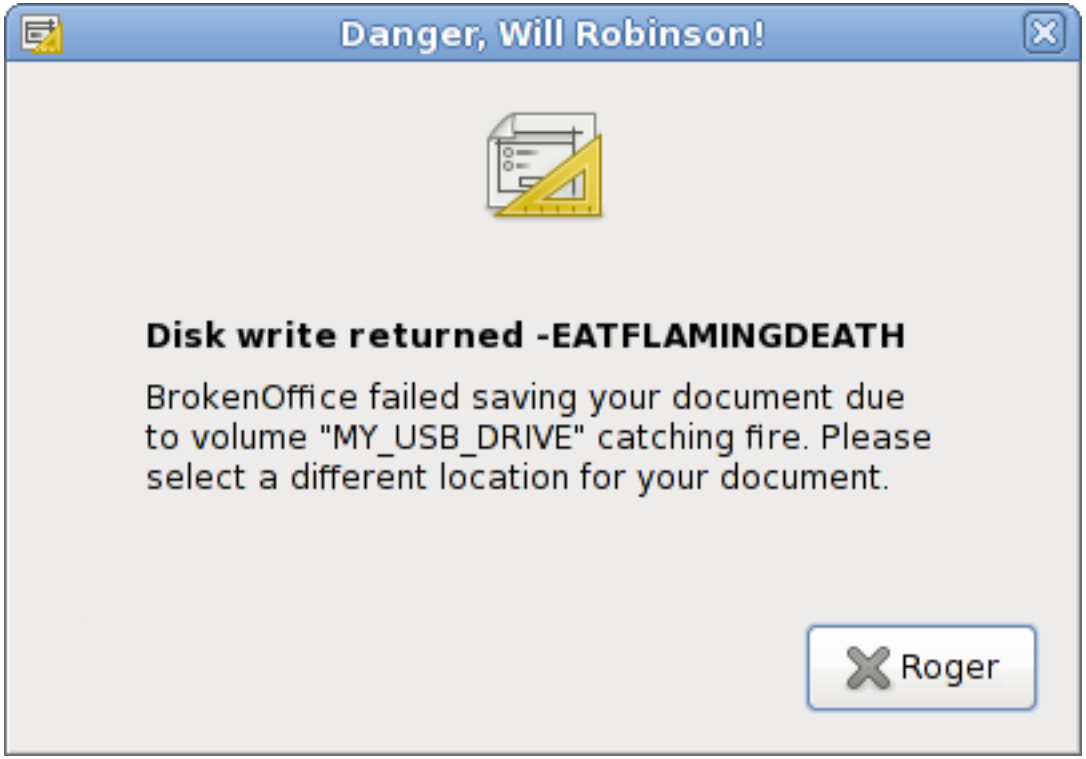
Linux Filesystems

- 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 things that will aid the recovery process (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
- Filesystem/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)

Wouldn't it be nice if...



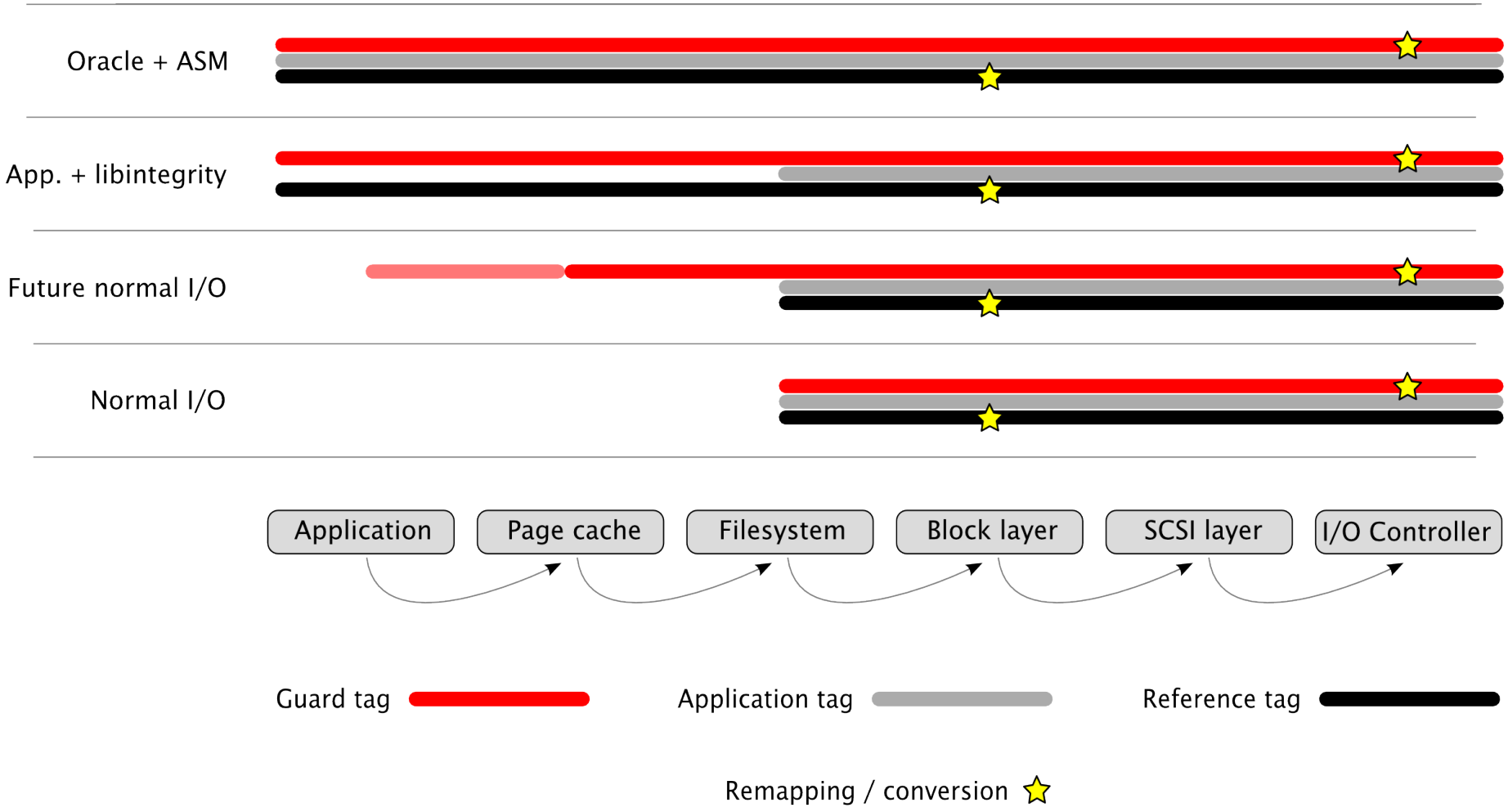
Our UNIX Heritage

- Then:
  - `cat foo | frob | mangle > bar`
  - Applications were short lived
  - -EIO meant that the pipeline broke and operator had to fix it
  - Input easily reproducible by restarting pipeline
- Now:
  - Oracle, mysqld, OpenOffice.org, firefox, etc.
  - Applications run forever
  - -EIO never gets to most applications thanks to buffered writes
  - Data mainly comes from user input and the network, often not reproducible
  - But we're still using the old API

Integrity Aware APIs

- POSIX asynchronous I/O interface
  - Not many applications use it
  - Linux implements POSIX aio poorly
  - Enables I/O completion status without resorting to blocking
  - Could potentially be augmented with a protection buffer
- Oracle ASM
  - Oracle's own swiss army knife I/O submission interface
  - Works with DIF/DIX today
- Generic interface in progress
  - Will allow normal applications to interact with protection information (in an opaque fashion)
- Worst case the filesystem or block layer will do the work for you

User API vs. Data Integrity



More Info

- http://oss.oracle.com/projects/data-integrity/
  - Documentation
  - DIX specification
  - Source code repository
  - Linux 2.6.27 has all the infrastructure
  - Software RAID/LVM support coming in 2.6.28