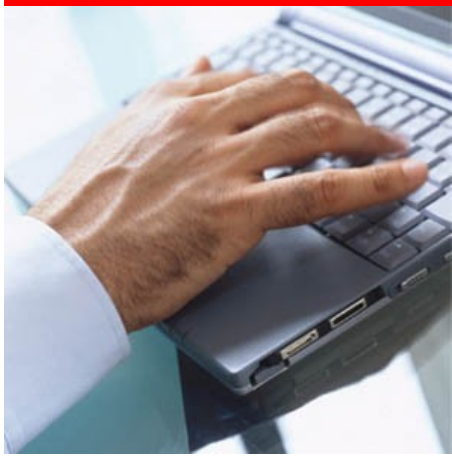

ORACLE®

ORACLE®

SNIA Data Integrity TWG: True E2E with T10 DIF

Martin K. Petersen
Software Developer, Linux Engineering

Topics



- What are we trying to achieve?
- Proof of concept at Spring SNW '07
- DII & DITWG
- Interoperability and T10 DIF
  - 4KB sectors
  - Reference tag
  - Write atomicity

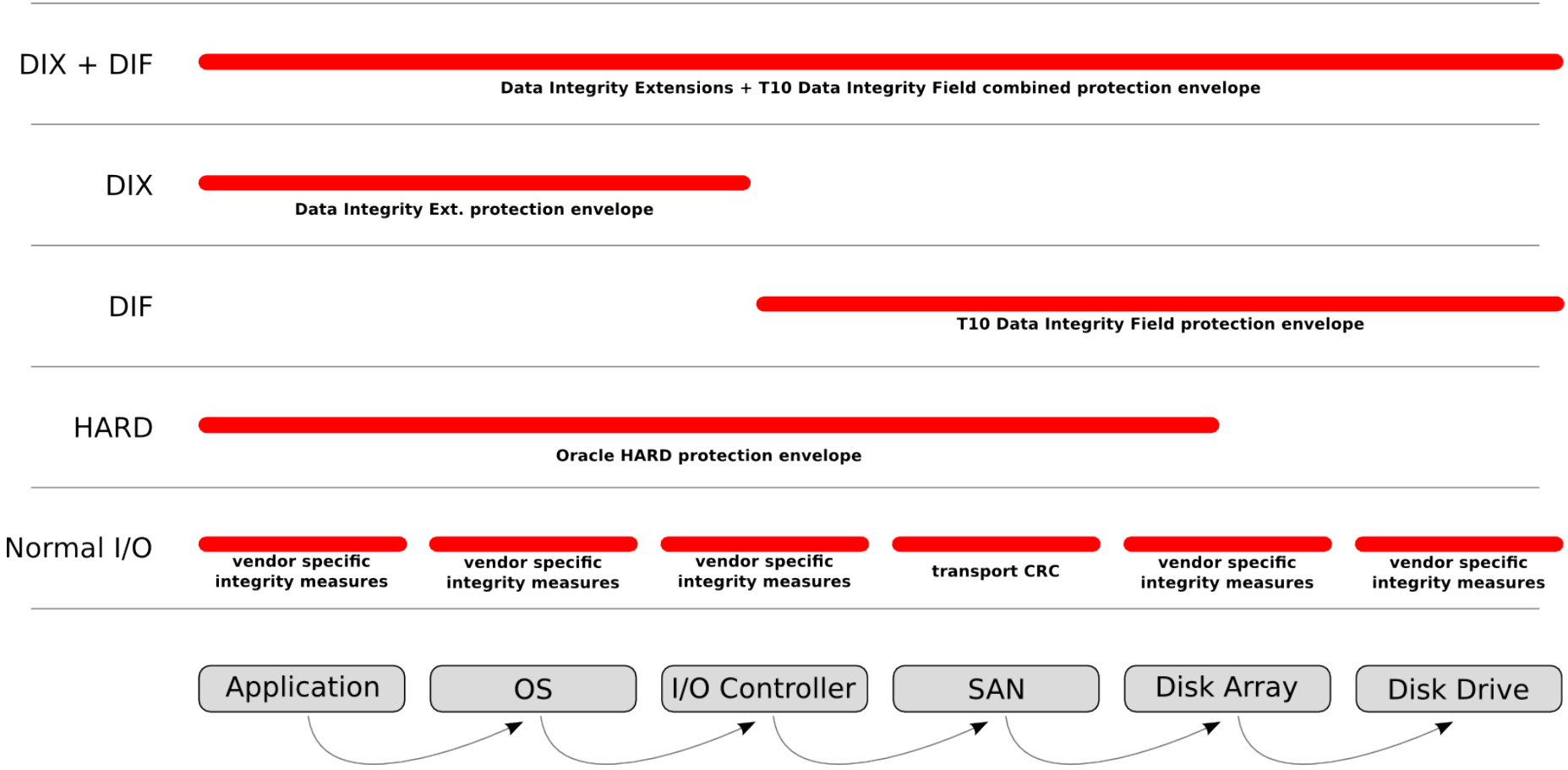
What are we trying to achieve?

- Industry demand for end to end data integrity
- Oracle HARD is widely deployed
  - Array firmware recognizes and verifies Oracle database logical blocks at the front end
  - Only available on high-end gear (Symmetrix class)
  - Doesn't protect path between array front end and physical disk
- Oracle would like to provide true end to end data integrity protection at the mid range and below
  - Decided to leverage an industry standard: T10 DIF
  - Want to provide integrity protection for non-database I/O

Data Integrity Initiative → DITWG

- Oracle/Emulex/LSI/Seagate
  - Proof of concept at SNW Spring 2007
  - Oracle DB running on Linux with Emulex HBA, LSI JBOD and Seagate disks
  - Linux reference implementation plus requirements document for generic interface for exchange of protection information between operating system and HBA (Data Integrity Extensions)
- Decided to broaden participation
  - Approached SNIA which lead to the formation of this TWG
  - Goal is to provide data integrity interfaces on several operating systems
  - Better means for data integrity will require interacting with T10

Data Integrity Extensions



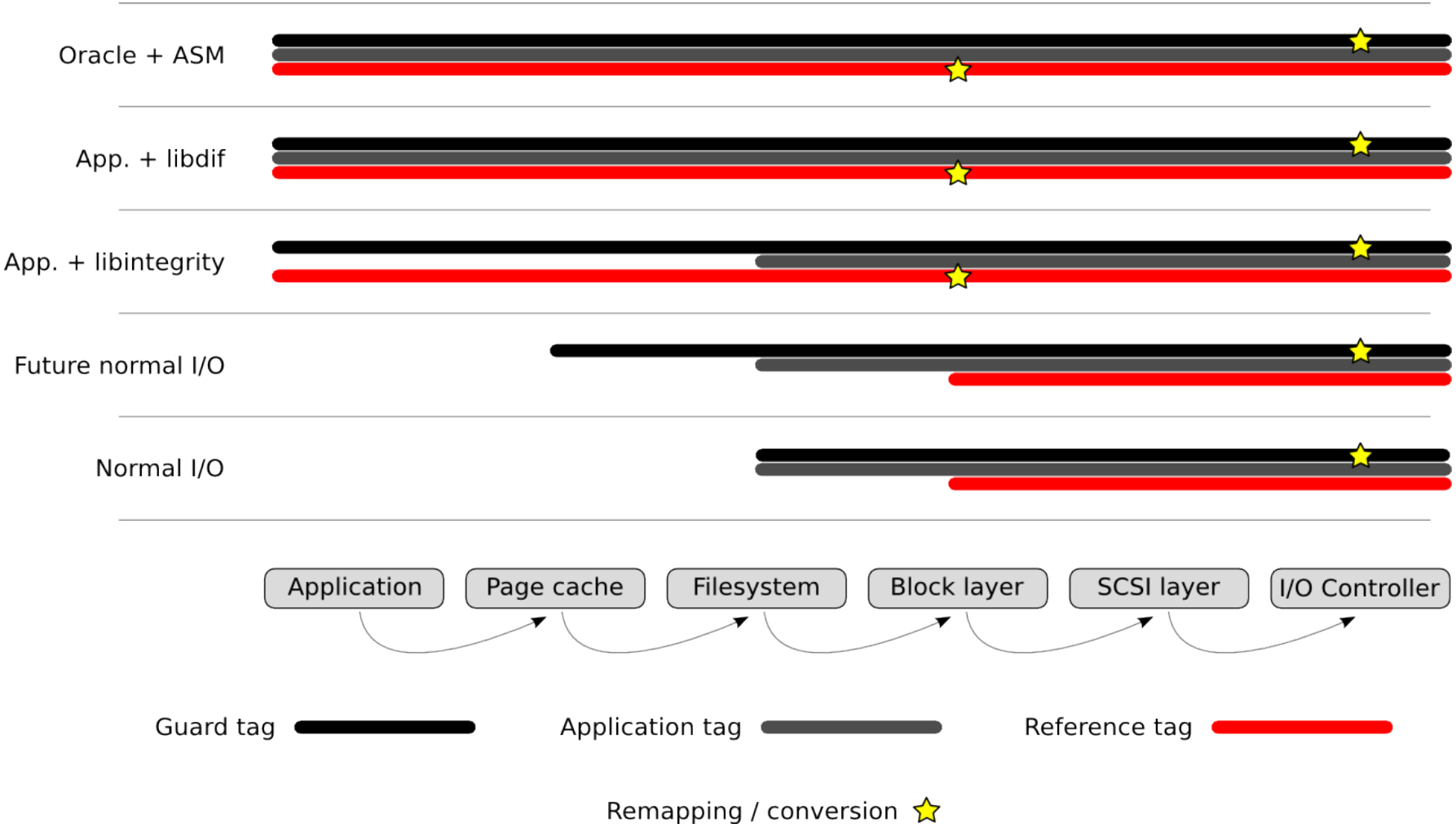
Where are we now?

- DITWG is obviously a medium to long term effort
- “Verbatim” DIF exchange via controller extensions pretty much good to go
  - Linux data integrity infrastructure ready from block layer down
  - Also supports current T13 External Path Protection proposal
  - Oracle ASM ready
  - Aiming for inclusion in upstream kernel release 2.6.27 and general availability in EL6
  - Application interfaces are work in progress

Application Interfaces

- Explicit - `libdif`
  - `mkfs`/`fsck` accessing DIF on storage device directly
  - Allows us to tag logical blocks
  - Portions may be within the scope of this TWG
- Opaque - `libintegrity`
  - “Protect this buffer”
  - “Let me know if everything got written ok”
  - Within the scope of this TWG
- Transparent - `libc`
  - Standard POSIX/SUS `read()`/`write()` style calls
  - `mmap()`, direct I/O pose challenges but doable
  - OS specific, not within the scope of this TWG

Application Interfaces



T10 DIF – 4KB sectors

- Interoperability with 4KB hardware sector devices:
  - Software RAID / Logical Volume Manager stacking can be arbitrary and infinite. Add to this snapshotting and live migration. This means that we can't recurse to map out an I/O ahead of submission time. We really need to be able to prepare **1** type of integrity metadata regardless of underlying storage topology and sector size.
  - Access patterns within a device are not homogenous. Not even for block-based applications like databases that access the storage device via direct/raw I/O.
  - Heterogeneous block sizes or access constraints are not acceptable. We must be able to read/write in arbitrary multiples of logical block length. Otherwise existing applications will break.

T10 DIF – 4KB sectors

- T10 DIF somewhat biased towards “application client” being array firmware
  - LSI's proposal assumes that the reader of a virtual logical block is the same (or has access to the same information) as the original writer. That is not a valid assumption in the OS space.
  - LSI's proposal also assumes that a virtual logical block will be aligned on a virtual logical block boundary which may not be the case.
- We need:
  - Access to protection information on any logical block sized boundary.
  - To be able to read and write an arbitrary number of logical blocks starting at that boundary.

T10 DIF – Reference Tag

- Current reference tag is 32 bits.
- Most operating system/library/application errors will happen on a 32-bit boundary.
- 32-bit LBA and 512 byte sectors = 2TB. Problem today.
- 32-bit LBA and 4KB sectors = 16 TB.
- Would like to propose a Type 4 as an extension of Type 1 in which the application tag space is used to extend reference tag to 48 bits.

T10 – Write Atomicity

- At the top of our wish list is a means to prevent partial writes from happening. All or nothing at a virtual block boundary.
- This virtual block size would be in the 1KB – 128KB range.
- Not a requirement that this blocking persists after I/O completion.
- Checksum to protect entire virtual block would be nice. Hard to do given the existing protocol constraints, though. Also makes OS stack much harder.
- Something akin to VLBAE would do the trick.