xyratex

Advancing Digital Storage Innovation

An Exploration of New Hardware Features for Lustre

Nathan Rutman

Motivation

- Open-source
- Hardware-agnostic Linux

↓

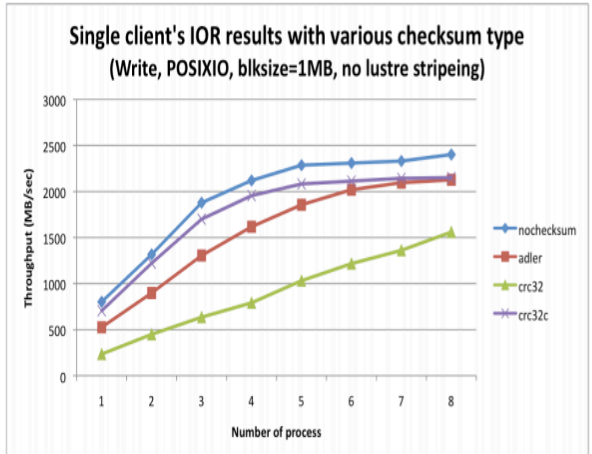
- Least-common-denominator hardware

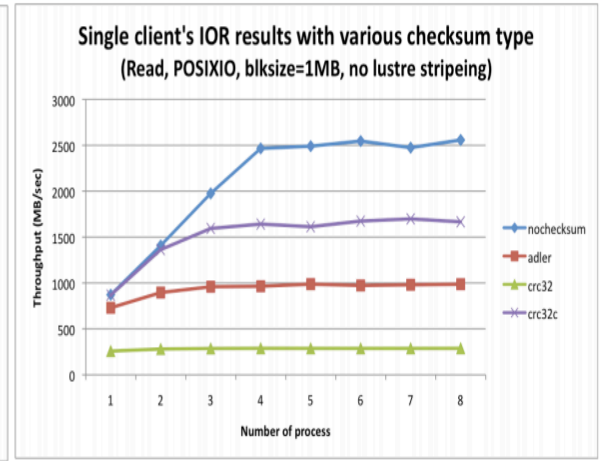
Contents

- Hardware CRC
- MDRAID
- T10 DIF
- End-to-end data integrity
- Flash drive use
- Hybrid volumes
- HA and faster failover

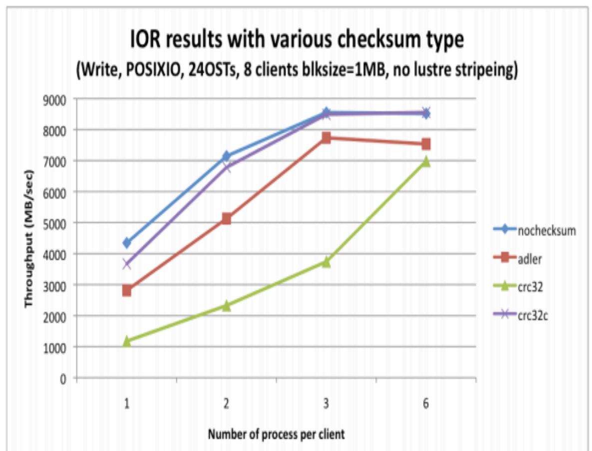
Hardware CRC

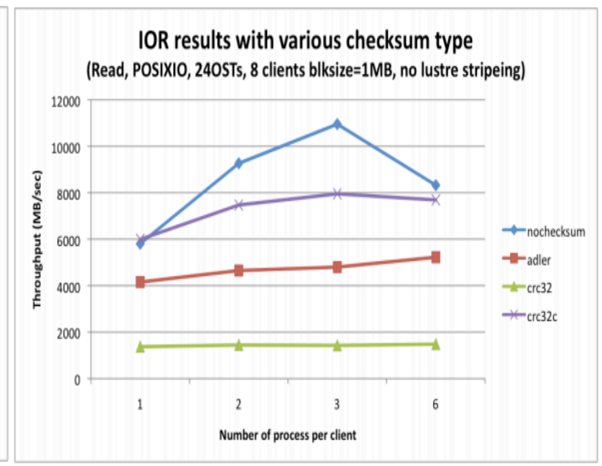
- Nehalem CRC-32C
  - BZ23549
  - LU-241





Shuichi Ihara's graphs from BZ23549





- Intel Westmere, AMD Bulldozer (2011)
  - PCLMULQDQ (64bit carryless multiply)
  - speed up CRC32, Adler

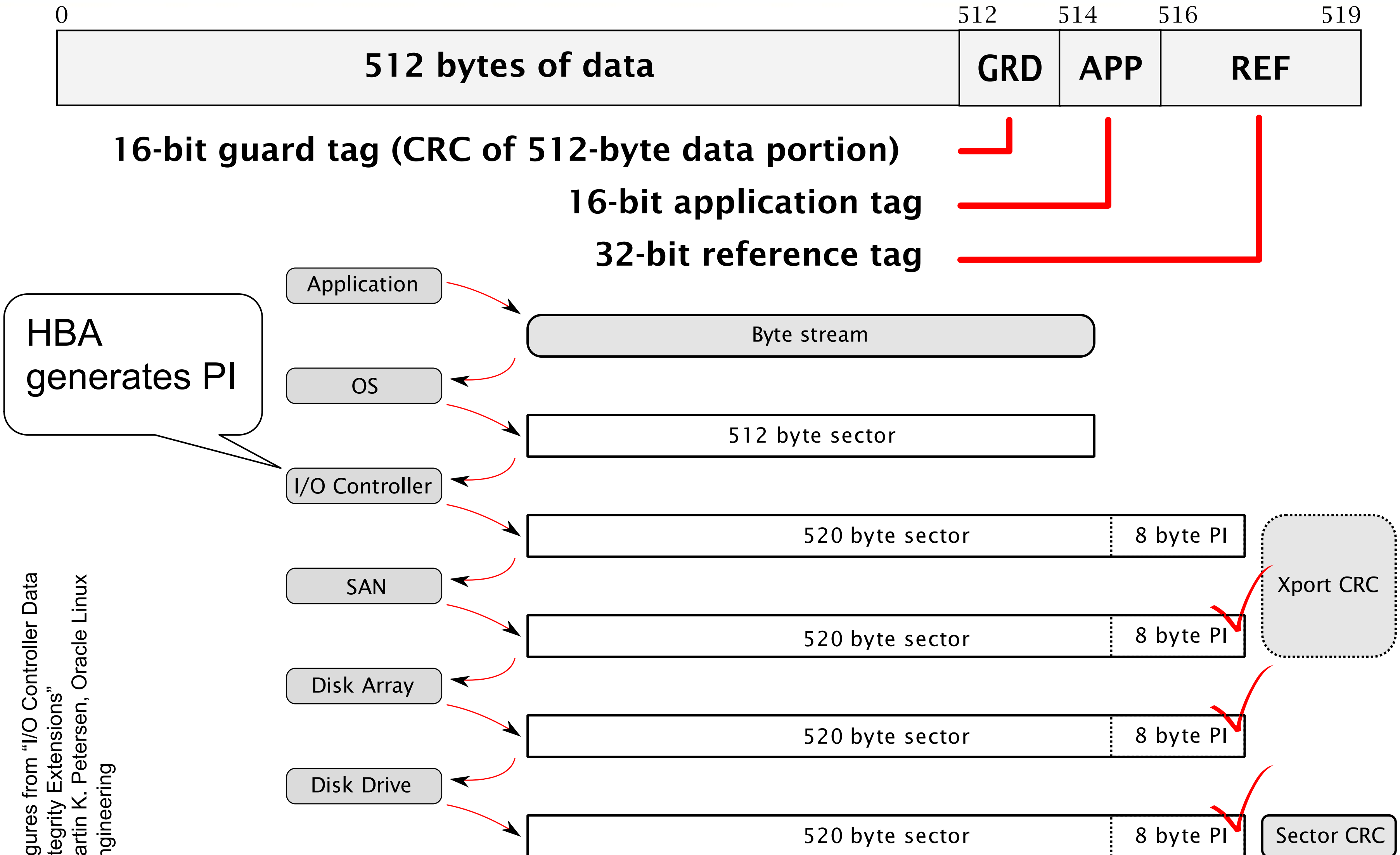
xyratex

MDRAID

- Ongoing improvements in Linux SW RAID
  - Hardening
  - Zero copy writes
  - Performance
  - RAID 6, 6E, 10, etc
- Still to do
  - Zero copy reads
  - PDRAID
  - Hardware parity math acceleration

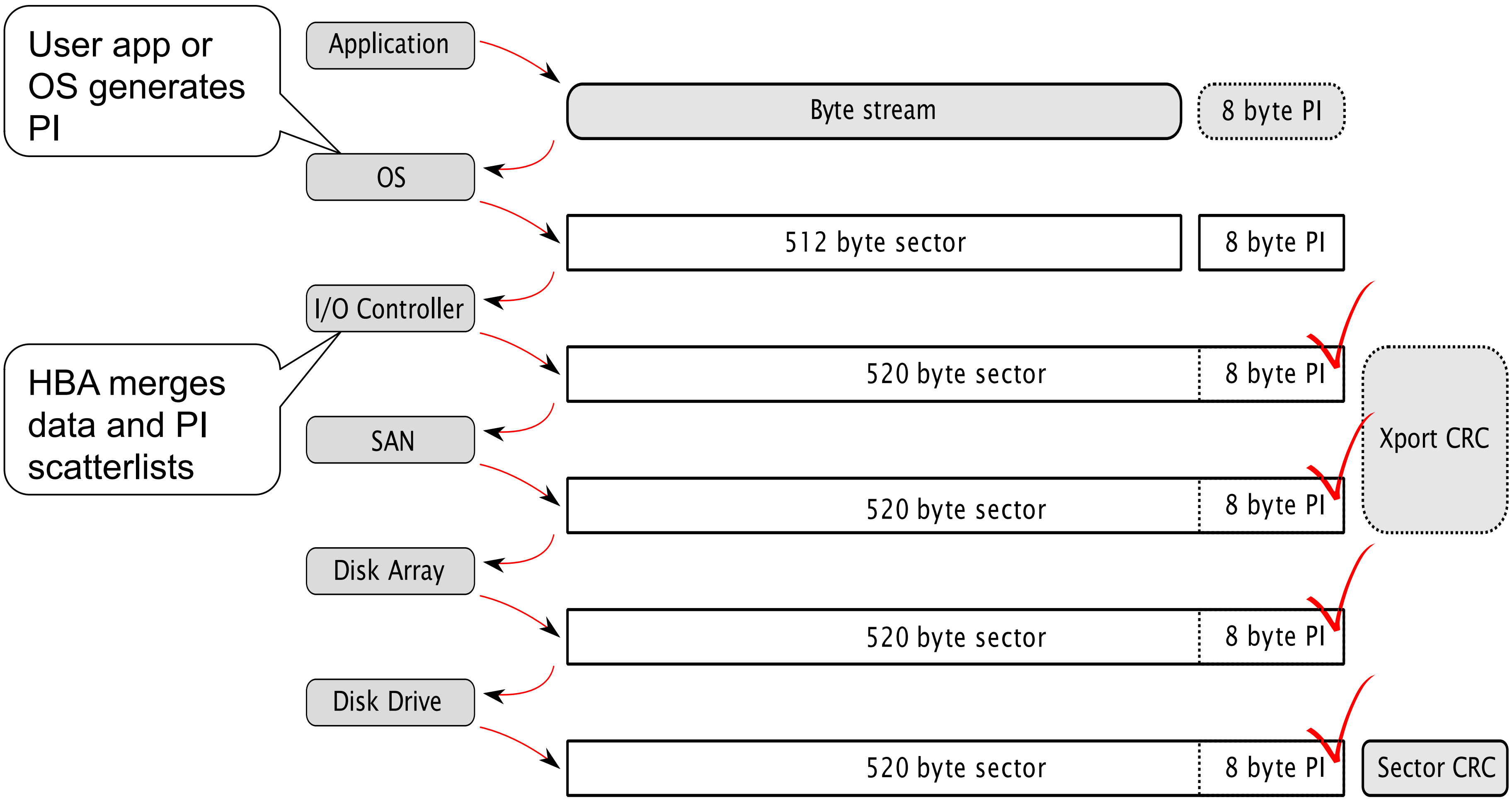
T10DIF and End-to-End Data Integrity

T10 DIF



Figures from “I/O Controller Data Integrity Extensions”
Martin K. Petersen, Oracle Linux Engineering

T10 DIX



xyratex

T10 End-to-End Data Integrity

Write

- Userspace computes GRD per sector (optional)
- Client recomputes GRD after kernel copy
- Client adds GRD to BIO bulk descriptor
- OST pulls BIO
- OST passes PI to ldiskfs
- MDRAID maps REF tags
- HBA takes SG lists of data and PI
- HBA recalculates GRD (retries from client)
- Disk verifies REF and GRD (retries from HBA)

Read

- Client sets up PI and data buffers, sends BIO
- OST sets up PI and data buffers
- MDRAID requests data and parity blocks
- Disk verifies REF and GRD
- HBA maps data and PI to buffers
- MDRAID verifies parity, reconstructs corrupted data
- OST sends data
- Client verifies GRD
- Userspace reverifies GRD after kernel copy (optional)

Flash Drives

Flash Drives

- SSDs are fast, but not large

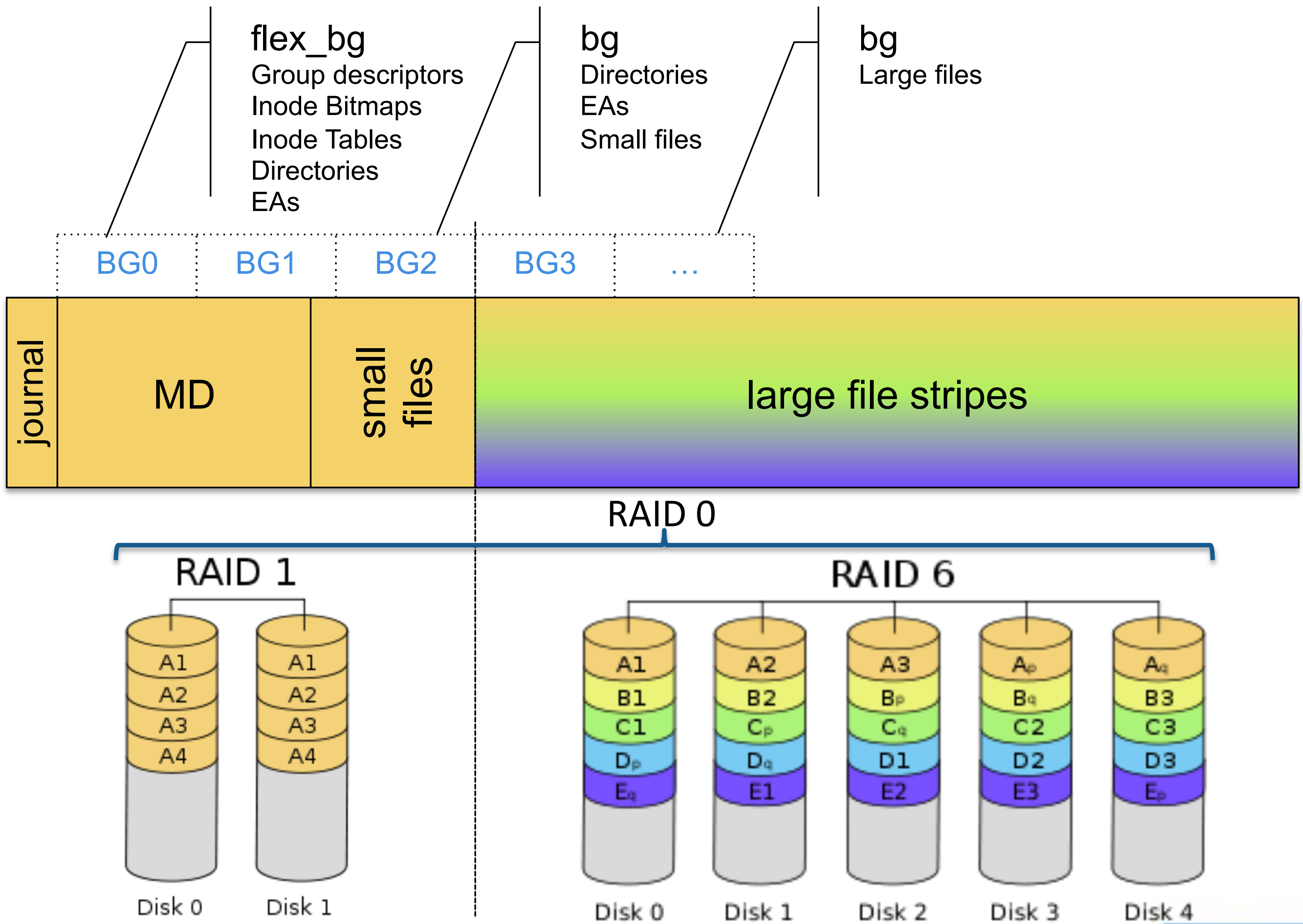
- Caching
- Lustre persistent data files
  - last_rcvd, last_objid, open_write
- Metadata
  - journals
  - WIBs
  - Lustre FS metadata
- Data
  - Small files

EXT4 Hybrid Volumes

EXT4 Hybrid Volumes

- Problems today
  - Metadata distributed around disk, breaking up large disk chunks and slowing fsck
  - Small files treated the same as large files (e.g. same RAID level)
- Hybrid Volume
  - A single filesystem spanning a group of local devices with different RAID striping, speeds, hardware, or access patterns
  - Single device loss will kill the FS, so each device must be safe (RAID)
- Change allocator
  - Put metadata together, all on fast RAID1
  - Put small files after this on RAID1
  - Put large files aligned on 1MB boundaries on RAID6
  - EXT4 online defragmenter can migrate them

EXT4 Hybrid Volume Layout



xyratex

EXT4 Hybrid Volume Advantages

- Larger transfer sizes and reduced seek time for each type
  - Don’t need to skip over data to get to metadata, and vice-versa
- Eliminate seek time between types
  - Leave the data volume read head waiting at the next 1MB boundary
  - Leave the MD head waiting at the allocator bitmaps
- Avoid RAID6 parity recompute penalty for small files
- Smaller metadata volume makes SSD, RAID1 practical
- FSCK times reduced
  - location implies content type (don’t need to read the whole disk for each pass)
  - ordered metadata is faster to read
- Multiple data volumes
  - Can sleep volumes as needed

HA and Failover

- HW Monitoring and Imperative Recovery
  - don’t wait for Lustre timeouts
  - more important in larger clusters
- Persistent RAM
  - RAMdisk MDT
  - Quick flush to SSD on power loss

Thank You