ZFS

Steven Lee

Features

- **Hardware Agnostic**
  Runs on servers with directly attached storage. No fancy RAID cards or SAN.

- **Pooled Storage**
  No volumes. All capacity is available to all file systems.

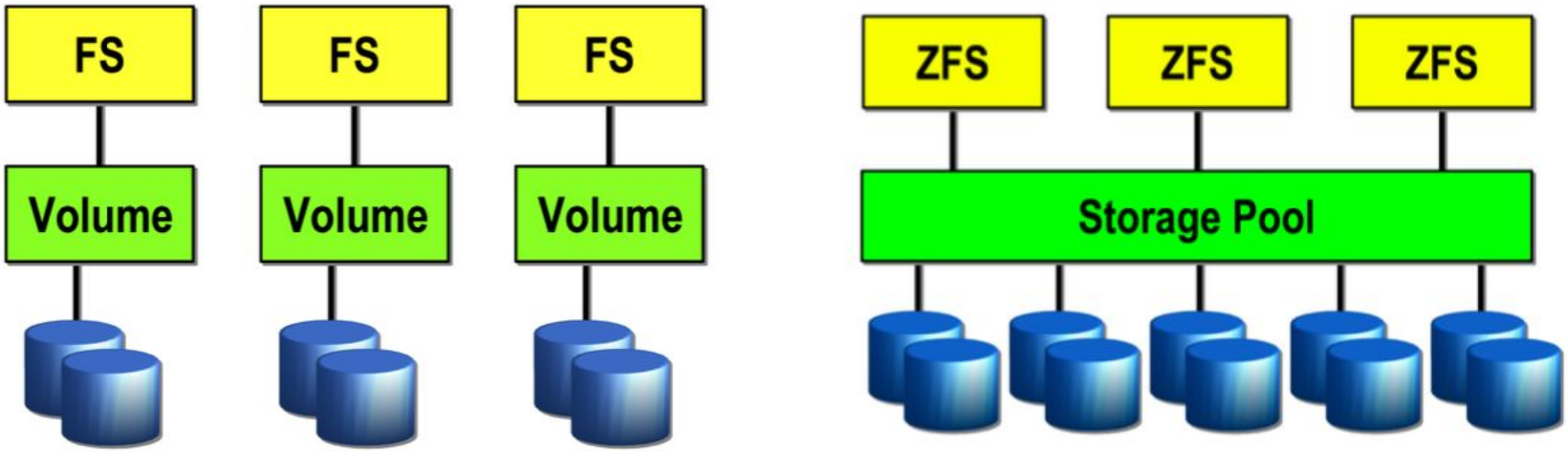
- **Transactional**
  Always consistent on disk. No fsck. Ever.

- **Data Integrity**
  Detects silent data corruption. Automatic corrects detected errors.

- **Simple Administration**

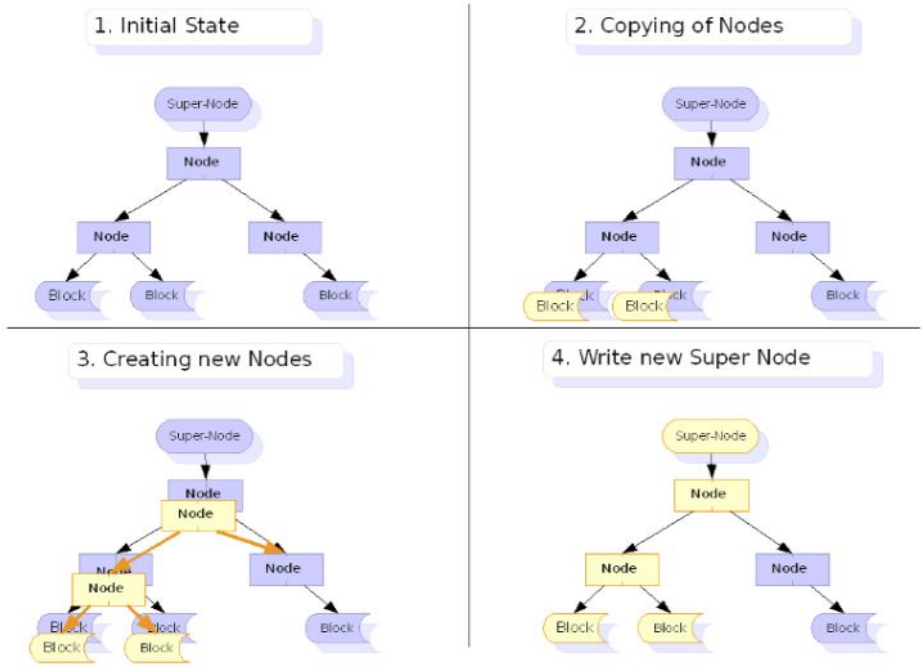
ZFS Storage Pool (zpool)

- **Abstraction: malloc/free**
- **No partition to manage**
- **Share all available *capacity* and *bandwidth***
- **Grow / shrink automatically: Unusual "df" output**



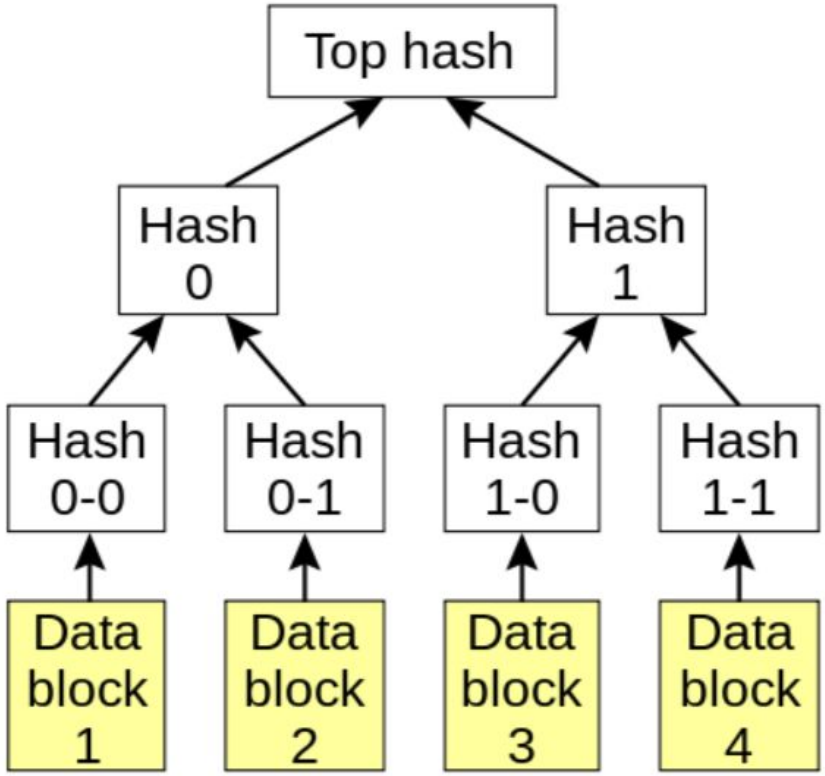
Copy-On-Write Transactions

- **Never overwrite block when modifying data:**
  - Write modified data on new block
  - Update pointers to new block
- **Free versioning / snapshots with old blocks & pointers**
- **Increased disk fragmentation**
  - Mitigating with read/write caching features



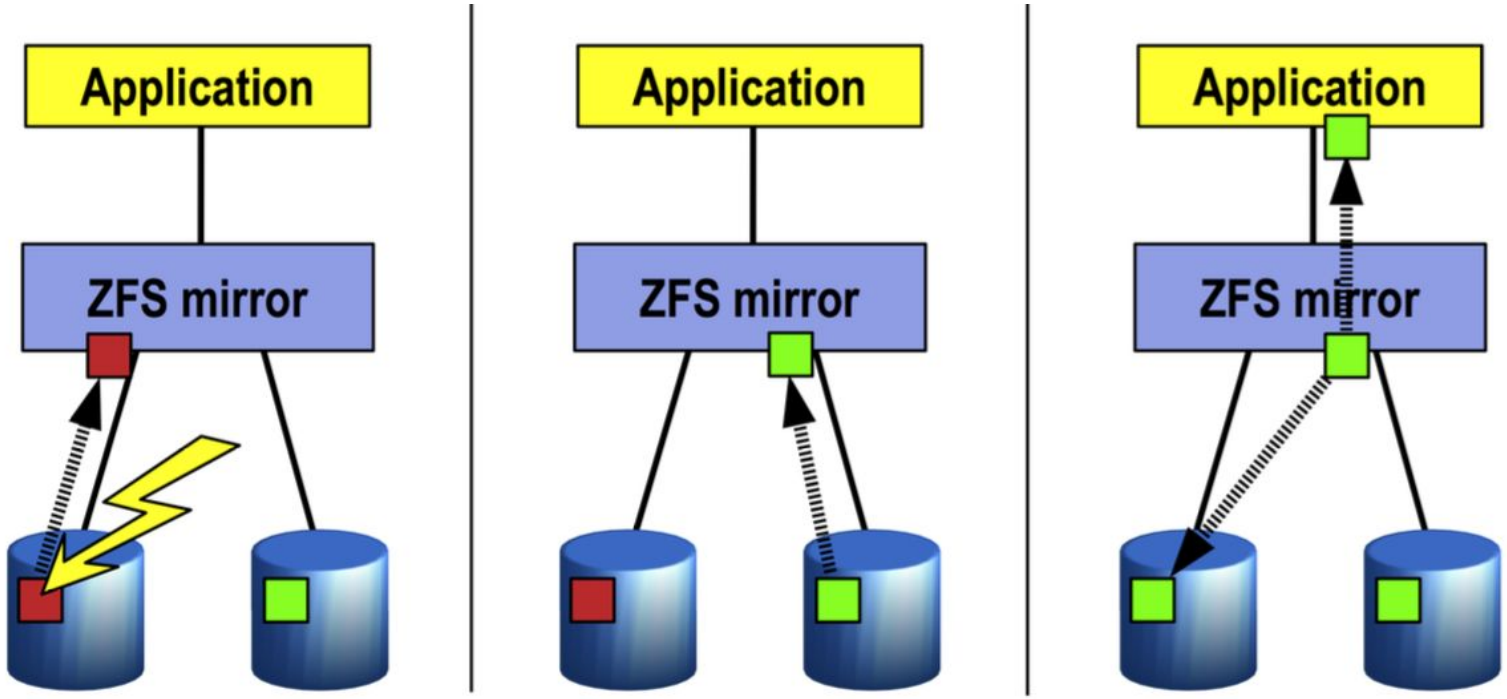
Bit-Rot Detection

- **Merkle Tree (Hash Tree)**
  - Each block of data is hashed
  - Parent hashes are computed from child hashes
  - Corruption/changes to data is detected when traversing the tree



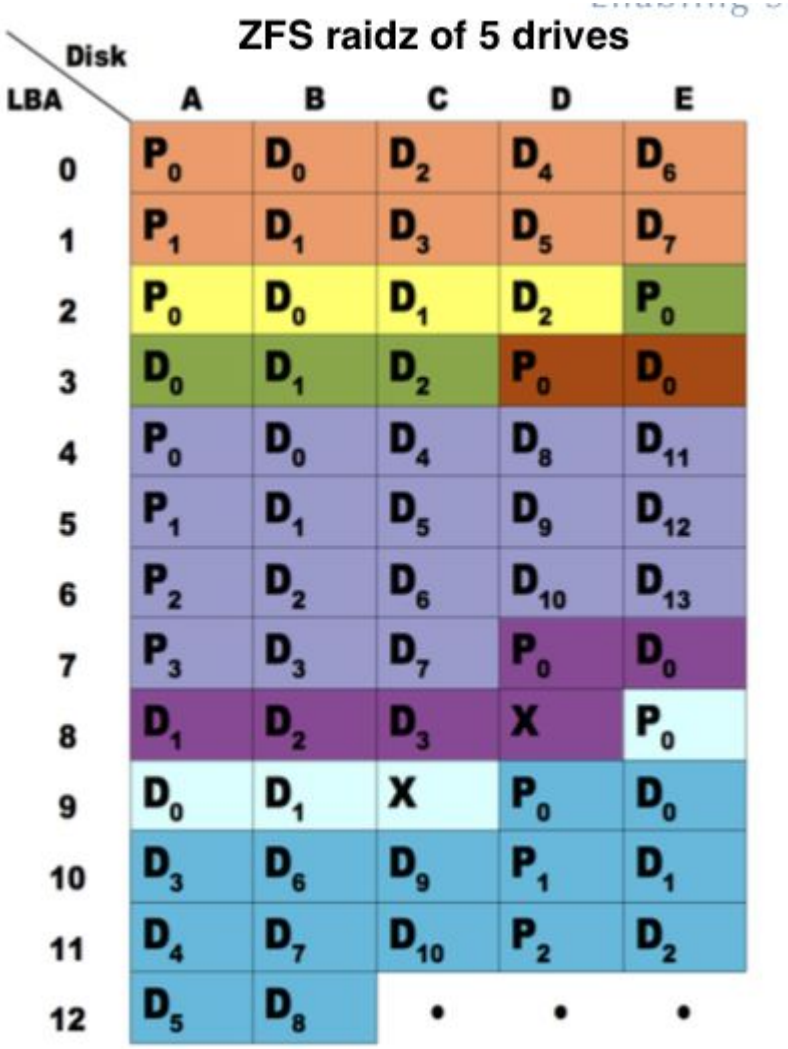
Data Error Correction

- **If RAID is configured, ZFS automatically corrects data corruptions as they are detected.**



RAID-Z

- **Nomenclature**
  - Mirror
  - raidzX where X = number of parity drives
    - raidz = RAID5
    - raidz2 = RAID6
    - raidz3: 3 parity drives
- **Dynamic Stripe Width**
  - Each logical block is its own stripe
  - ZFS decides disk placement at write time
- **No more read-modify-write!**
  - Faster as write operations don't require read
  - No need for NVRAM



Resilvering: RAID Creation & Repair

- **Top-down: ZFS resilvers the block tree from the root down**
  - Most important blocks first!
  - Every block copy increases recoverable data
- **Live blocks only**
- **Dirty time logging for transient outages**
  - ZFS walks the tree and updates where birth time < DTL

Scrubbing

- **Walks the entire tree and correct all the errors**
- **Live block only**
- **Minimal performance impact**
- **Should be a scheduled task performed periodically to discover silent errors/bit rots.**

ARC and L2ARC

- **Two levels of read cache**
  - ARC: implemented in RAM
  - L2ARC: implemented in fast SSDs
- **L2ARC is optional. ZFS continues to operate, albeit at reduced performance, if L2ARC (SSD) fails.**

ZIL & SLOG

- **ZIL (ZFS Intent Log)**
  - Used to bundle POSIX synchronous writes into larger writes
  - Help reducing file fragmentation
  - Not in data path: read only when recovering from system crashes
  - Implemented in zpool: slower as zpool gets busy
- **SLOG (Separate intent Log)**
  - SLOG implements ZIL in separate SSDs
  - SSDs should be mirrored to protect against SSD failures
  - SLOG is optional. ZFS will continue to operate at slower performance if SLOG is removed from the zpool.

Miscellaneous Features

- **Compression**
  lz4 saves space, increases performance, and should be enabled.

- **Scalability**
  zpools are scalable by adding more devices. Workload is automatically distributed.

- **Snapshots (.zfs/snapshot in root of each file system)**
  - Instantaneous creation, unlimited number
  - No additional space used
  - Ideal uses: incremental backups/self-served restores, sharing read-only data

ZFS on Linux

- **Owned by Oracle; development lead by Intel under OpenZFS and ZFS-on-Linux as part of Intel Lustre**
- **Implemented as a kernel module**
- **Oracle Licensing Restriction**
  Vendors can distribute source code but not binaries
- **Workarounds**
  - dkms (dynamic kernel module service): Detects new kernel and compiles kernel module at boot time. Automates kernel upgrade but slows boot time.
  - kmod: Pre-compiles kernel module and deploys with the new kernel.

ZFS at CAC

- **Dexter: a ZFS-based file server**
  http://www.cac.cornell.edu/syswiki/index.php?title=Dexter

- **TheCube: Intel Lustre**
  - Intel Lustre uses ZFS for object store