LAD 2022

ldiskfs Snapshots

Andreas Dilger

adilger@whamcloud.com

ddn

Snapshot Options Evaluated

ZFS Integrated Snapshots

- \+ ZFS also allows compression, checksum, dedupe
- \+ Works well today, reduced core ***development*** effort
- \- Major changes to EXA configuration, ***support*** effort
- \- ***Much*** slower than ldiskfs for high-IOPS usage
- \- No upgrade path for existing customers

Fujitsu Ext4 DLSnap Patches

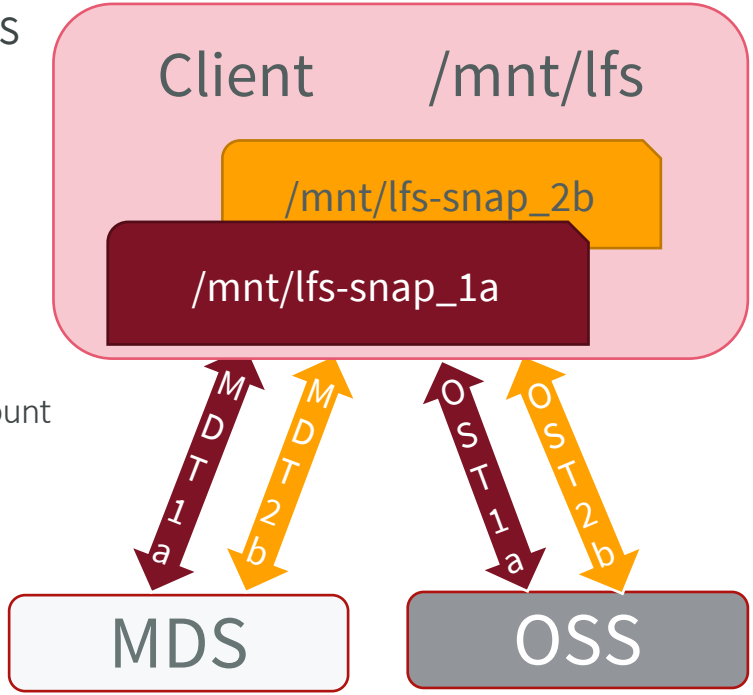
- \+ Previously used in large-scale production
- ▪ File level snapshot, not whole filesystem level
- ▪ Allows upgrade path for Fujitsu customers
- \- *Very* old Lustre/kernel versions used without update
- \- Major effort needed to update to new Lustre/ext4
- \- Need to process ***every*** file when creating snapshot
- \- N copies of ***each*** inode for N snapshots (=slow, large)
- \- Issues with code complexity and Lustre interaction
- \- Incompatible with ext4, upstream unlikely to accept

Ext4 Snapshot (CTerra) patches

- \+ Whole-filesystem snapshots
- \+ Available e2fsck/e2fsprogs patches
- \+ Reserved fields in upstream ext4
- \+ Upgrade path for existing DDN customers
- \+ Lots of developer experience with ldiskfs
- \+ Efficient space usage for multiple snapshots
- ▪ Some production usage at CTerra
- ▪ Newer kernel/e2fsprogs vs. DLSnap
- \- Needs development to work for newer ldiskfs
- \- Unproven with Lustre and large filesystems

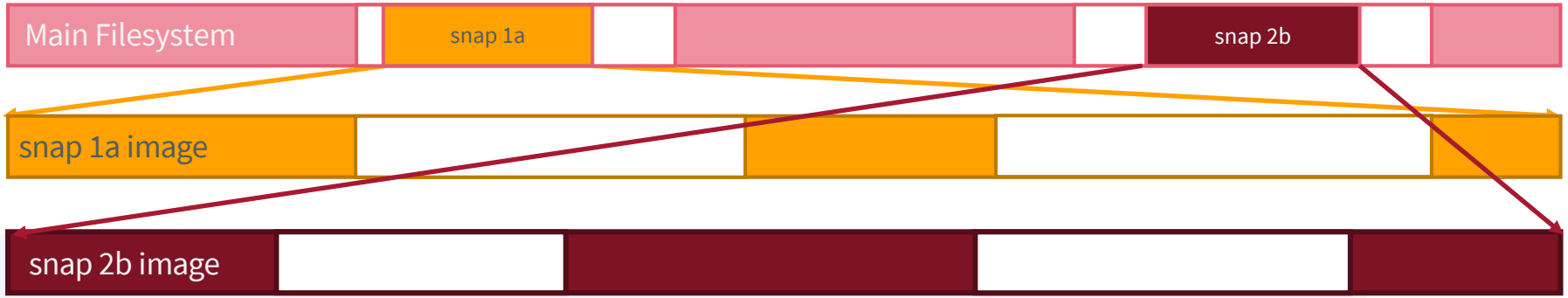
Overall Snapshot Architecture

- Snapshot is created within `ldiskfs` for each MDT and OST
- Same "`lctl snapshot`" and `lsnapshot` commands as ZFS
  - Added additional status commands, snapshot space usage
- Appears as a virtual ldiskfs/lustre filesystem image
  - Can mount image with either ldiskfs or osd-ldiskfs
- Snapshots strictly read-only after creation
  - Snapshot filesystem name modified while snapshot is created
  - Small changes needed to allow read-only ldiskfs filesystem mount
  - Lustre clients must read-only mount snapshot
- Remove snapshots without modifying main filesystem



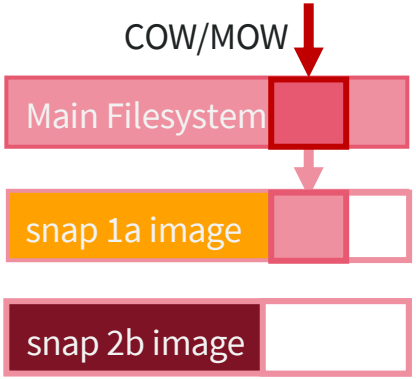
Internal ldiskfs Snapshot Implementation

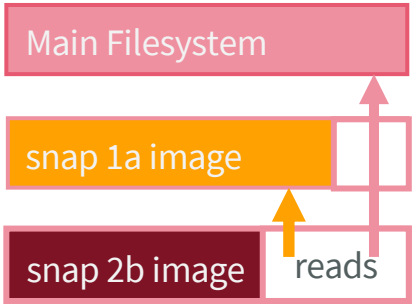
- Each snapshot is a sparse **image file** (inode) stored inside `ldiskfs` filesystem
  - Image file exported via **kernel loopback driver** to userspace
  - Snapshot inode looks like full copy of filesystem at time of creation
- Must handle ldiskfs filesystems much larger than normal 16TB file limit
  - Snapshot inodes are very sparse and have randomly-allocated blocks
  - New 64-bit 5-level indirect block format ("`5ind`") only used for large snapshot inodes
  - Could update/replace `5ind` format by making new snapshots, if needed
- Mostly standalone patches applied via ldiskfs series during build
  - 10kLOC patch for snapshots, 5kLOC patch for large inode format



ldiskfs Snapshot Creation and Update

- Create sparse inode with size of main filesystem
  - Copy of superblock and core filesystem metadata
- Metadata (inode, dir, bitmap) copy on write (COW) to snapshot inode
  - Copy from *physical* block to *logical* block in snapshot inode on first change
- File data (allocated blocks) moved on write (MOW) to snapshot inode
  - All file data writes *already* overwrite whole block today
    - Copy-on-write done by Lustre client and/or ldiskfs local mount
  - Avoids data copy overhead/fragmentation for majority of large writes
  - Avoids multiple copy operations when multiple snapshots
- Do not process blocks belonging to snapshots when modified
- Reads of sparse holes fall through to previous snapshot or physical block
- Oldest snapshot removal is (mostly) deleting the snapshot inode
- Intermediate snapshot removal moves blocks to next older snapshot





Detailed Test Workloads

Test System

- SFA18KX with 32 x 540TB OSTs (8 OST/OSS), 4x NVMe MDTs (1 MDT/MDS)
- 50% of main filesystem filled up at start (inodes/capacity)
- 8 worker nodes (1 x 80-core ARM CPU, 512GB Memory, 100Gbps RoCE)

Workloads

- W1: 320 x `mdtest` jobs (`ppn=1`, random create order, 5K, 17K, 66K, 133KB file size)
- W2: 320 x `IOR` jobs (`ppn=1`, random write/read, 50GB files)
- W3: Delete random files including pre-created files

Test Cases

- Endless repeat workloads W1, W2, W3
- Take new snapshot every hour up to 14 snapshots
- Mount oldest snapshot, check and destroy
- Stop system in middle of tests, run `e2fsck` on OSTs/MDTs to verify consistency

Current Snapshot Status

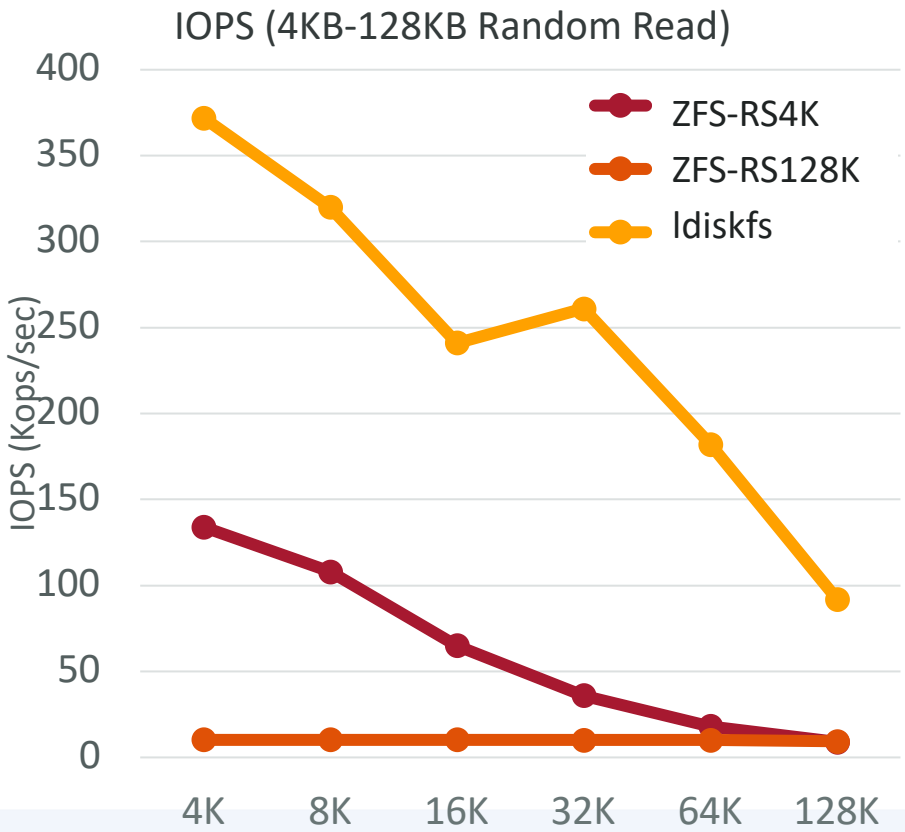
- Originally developed and tested on el7.9 3.10 kernel
  - Patches +11kLOC for snapshots, +5kLOC for `5ind` file format
- Core Lustre changes mostly isolated to utils and osd-ldiskfs
  - +1.6kLOC for `lustre/osd-ldiskfs` (loopback setup, credits, management)
  - +2.4kLOC for `lustre/utils` (`zfs`/`ldiskfs` separation, status reporting)
- Performs well on NVMe OSTs/MDTs
  - High IOPS storage handles COW/MOW/read IOPS easily, relatively smaller devices
- Slower performance on very large HDD OSTs
  - Originally took over 30 minutes to create/mount/remove "full" snapshot, now tens of seconds
  - Needed additional create/access/remove space/performance optimizations
  - Will no longer prellocate all group descriptors at snapshot creation
- e2fsprogs with `e2fsck` to handle snapshots and `5ind` file format
  - Delete all snapshots with "`e2fsck -x`" (clears snapshot inodes, all blocks reclaimed)
  - Can run e2fsck on snapshot image to verify correctness, no snapshot repair today
- Working on port to el8.6 4.18 kernel

Thank You

Andreas Dilger

adilger@ddn.com

Benchmark ldiskfs and ZFS on NVMe - IOPS



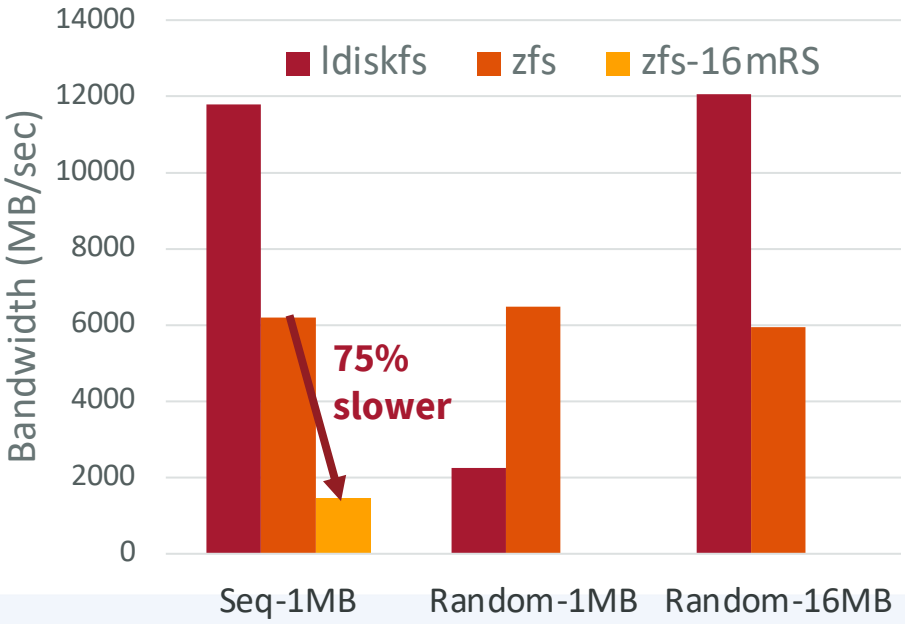
- AI200 (24x NVMe), ldiskfs ZFS 0.8.1, Lustre 2.13
  `# zfs set recordsize=4K scratch5/ost0`
- 4KB `recordsize` vs. 128KB faster ZFS IOPS but…
  - `recordsize` is only tunable per filesystem today
  - Must set file `recordsize` before first file write
  - Small `recordsize` causes slow write performance
- ldiskfs is still much better for flash IOPS than ZFS
  - ZFS must still read whole block to verify checksums
  - IOPS limited by CPUs power within VM
  - Even for NVMe OST size is getting larger
  - IOPS gap large, need to compensate with hardware
    - Need 2-15x more storage to match ldiskfs performance

Small `recordsize` helps small random IO somewhat, but still a large gap vs. ldiskfs

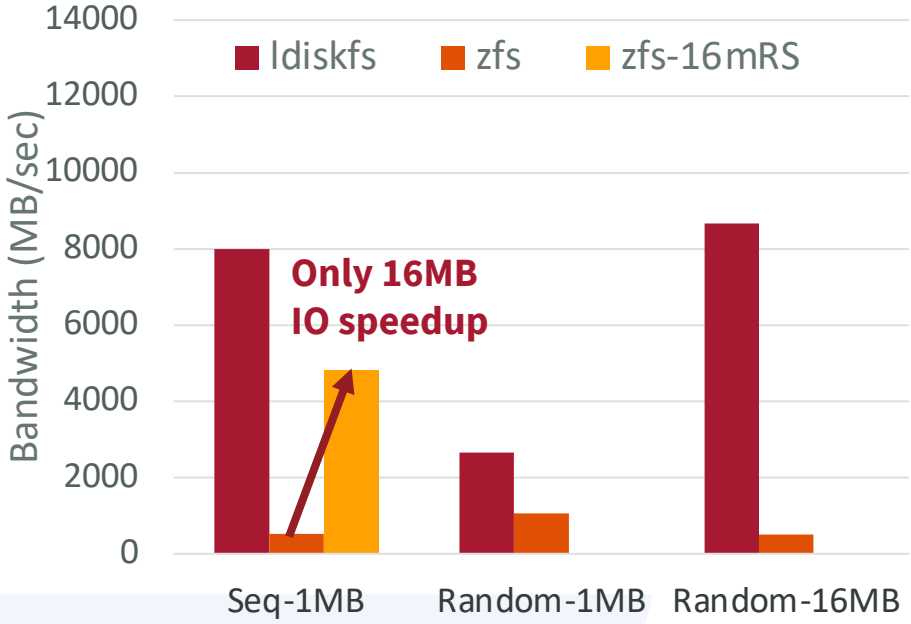
Benchmark ldiskfs and ZFS on SFA HDD - Bandwidth

- Setup SFA7990 and installed both ldiskfs and ZFS 0.8.1 with Lustre 2.13
  `# zfs set recordsize=16M scratch7/ost0` (changed 16MB record size from default 1MB)

IOR (FPP, 1MB, Write)



IOR (FPP, 1MB, Read)



Large recordsize hurts many IO patterns, hard to improve ZFS performance


ddn