Lustre User Group 2012

April 23, 2012

Brian Behlendorf, Sequoia's 55PB Lustre+ZFS Filesystem

Lawrence Livermore National Laboratory

LLNL-PRES-551671

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract

DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Outline

- Why ZFS
- Status Update
  - Sequoia Storage Hardware
  - Functionality
  - Stability
- Performance
  - Meta Data (MDTEST)
  - Write / Read (IOR)
- Open Questions
- Summary

Why ZFS

- Scalability
  - Massive storage capacity
    - 2^64 bytes per object
    - 2^78 bytes per pool
  - Dynamic Striping
  - Single OST per OSS
- Cost
  - Combined RAID+LVM+FS
  - Built for inexpensive disk
  - No vendor lock in
  - All open source
- Data Integrity
  - Copy-on-Write
  - Checksums
    - Meta data and block data
    - Checksums verified on read
    - Automatically repairs damage
  - Multiple copies of meta data
    - Small amount of storage
    - Spread over different disks
  - Ditto Blocks
  - Redundancy – Stripes, Mirrors, RAIDZ1, RAIDZ2, RAIDZ3

Why ZFS

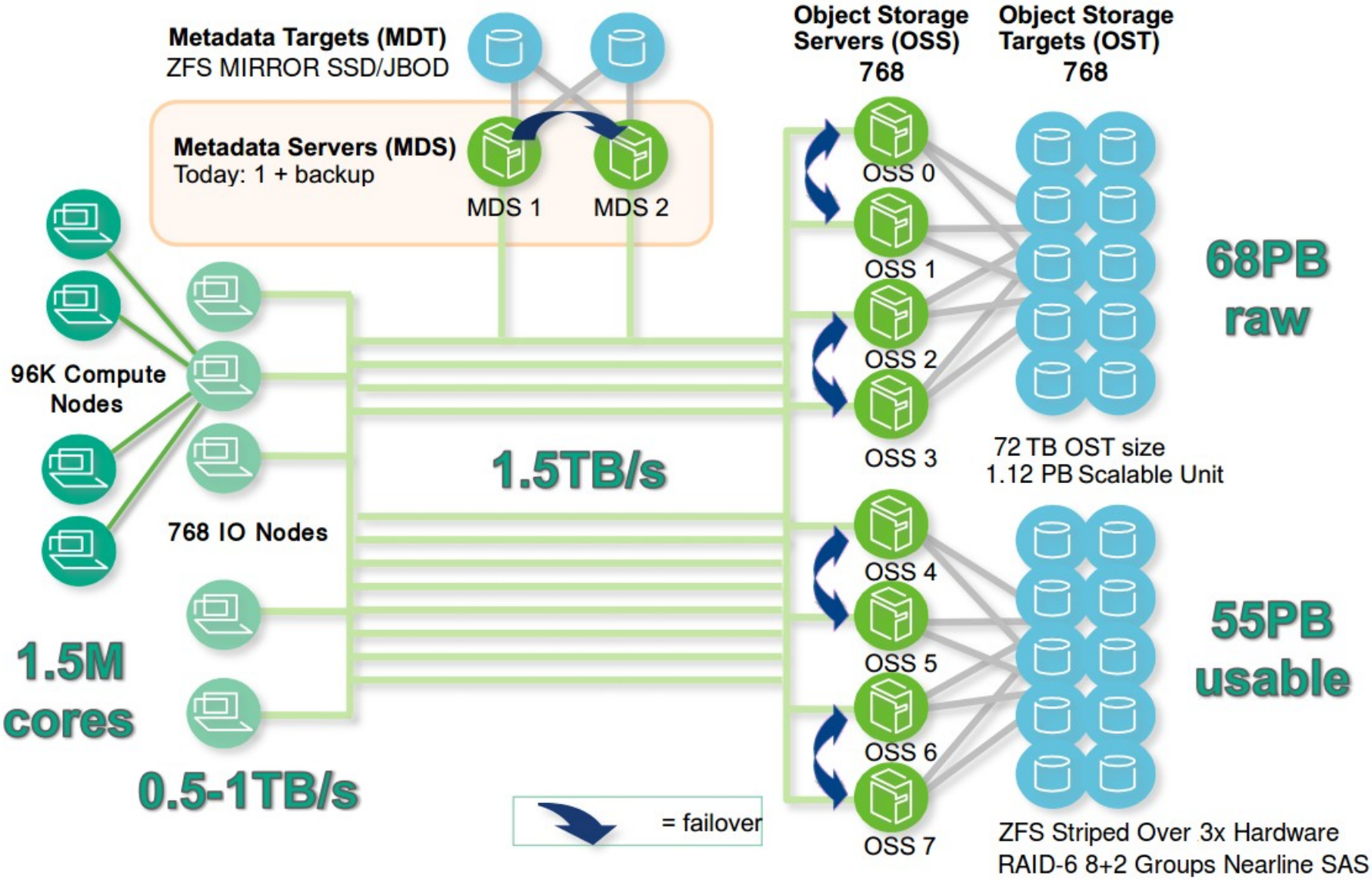
- Manageability
  - Online everything
    - Scrubbing
    - Resilvering
    - Pool expansion
    - Configuration changes
  - Fast filesystem creation
  - High quality utilities

- Features
  - Snapshots
  - Clones
  - Compression
  - Deduplication
  - Dataset send / receive
  - Advanced Caching
    - ZFS Intent Log (ZIL)
    - L2ARC
  - Adaptive Endianness
  - Quotas

Status Update

- 55PB Lustre+ZFS File system for Sequoia
  - Storage hardware has arrived
  - Lustre+ZFS file system configured
  - File system available to Sequoia users
  - Development still under way
  - Performance work started
- Contract with Whamcloud
  - Development scheduled to be complete in September 2012
  - Available in the Lustre 2.4 release

LLNL Sequoia Lustre Architecture



Sequoia Storage Hardware (OSS)

- NetApp E5400 (LSI)
  - 60-bay 4U Enclosure
  - Dual RAID Controllers
  - 3TB Nearline SAS Disks
  - 180TB RAW Capacity
  - IB Host Attached

- Appro GreenBlade
  - Intel Xeon E5-2670 (Sandy Bridge)
  - Dual Socket, 8 Core @ 2.60GHz
  - 64GB RAM
  - QDR Mellanox ConnectX-3 IB
  - Dual Port QDR ConnectX-2

Sequoia Storage Hardware (MDS)

- JBOD
  - 24-bay Enclosure
  - Dual Controllers
  - 40 - 1TB OCZ Talos 2 SSDs
  - 40TB RAW Capacity
  - SAS 6GB/s Host Attached

- Supermicro X8DTH
  - Intel Xeon X5690 (Westmere)
  - Dual Socket, 6 Core @ 2.47GHZ
  - 192GB RAM
  - QDR Mellanox ConnectX-3 IB
  - Dual Port LSI SAS Adapter

Functionality

- Critical Components Largely Complete
  - OSD API
    - LDISKFS OSD
    - ZFS OSD
  - OFD API
  - LOD API
  - LLOG restructuring
  - MGS/MDT/OST over OSD
  - Patchless ZFS Servers

- Under Development
  - Quota over OSD
  - Changelog over OSD
  - ZIL Integration
  - Linux Drive management

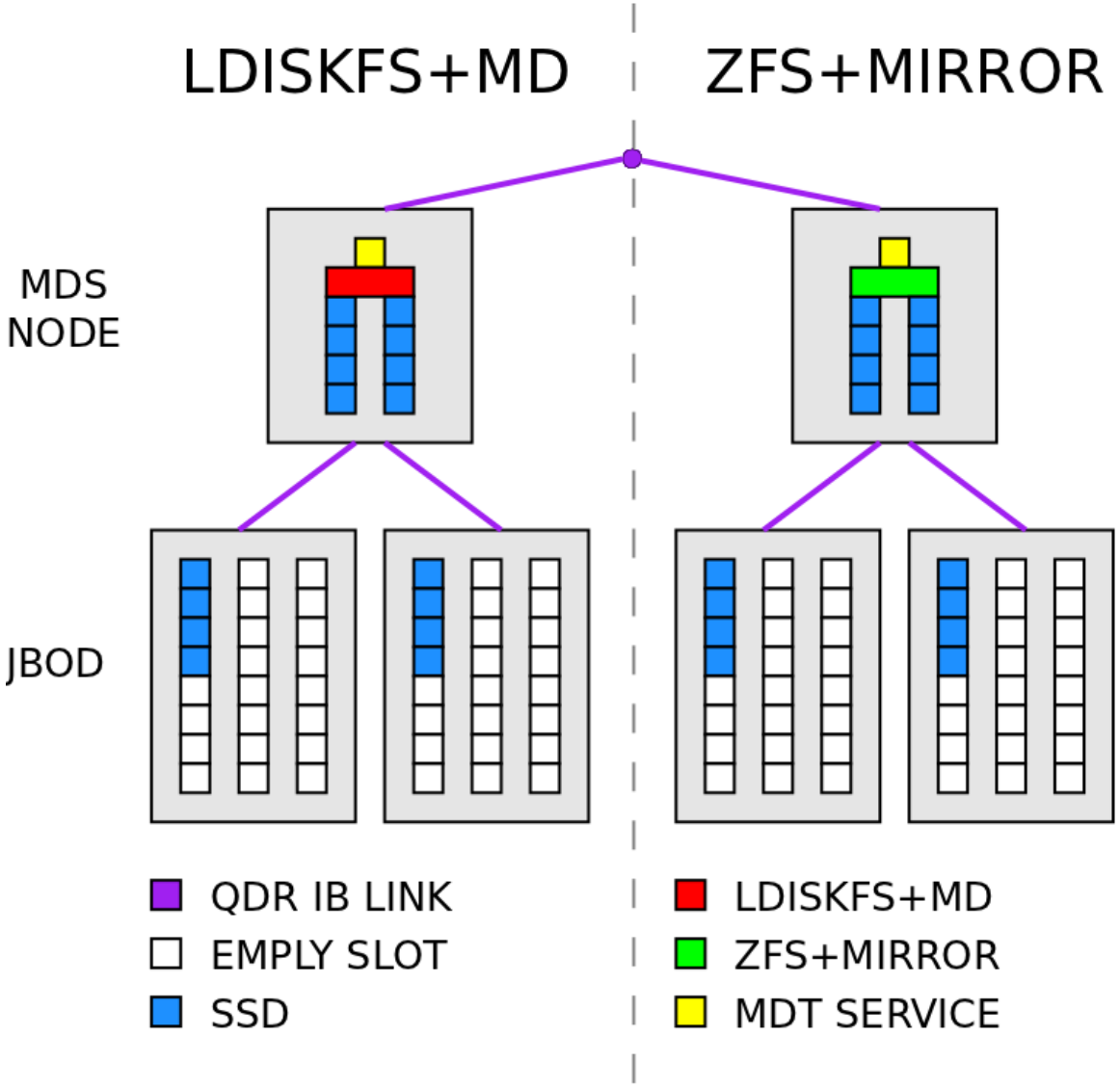
Stability

- Reasonably Stable
  - Passes most of the Lustre Test Suite
  - Stable under moderate test loads
  - Issues observed under heavy testing
    - Memory pressure can cause OOMs
    - Lustre and ZFS deadlocks
    - Relatively few panics
  - Working to resolve all issues
  - Real usage on Sequoia will expose any issues missed during testing

Performance

- NOT one of Livermore's main reasons for ZFS
- BUT performance must be reasonably good
- Focused on expected Sequoia workloads
  - Meta data - creates / stats / unlinks
  - Defensive I/O - checkpoint restart
- Caveats
  - Just started performance analysis
  - Minimal tuning has been done
  - Expect to be able to make significant improvements

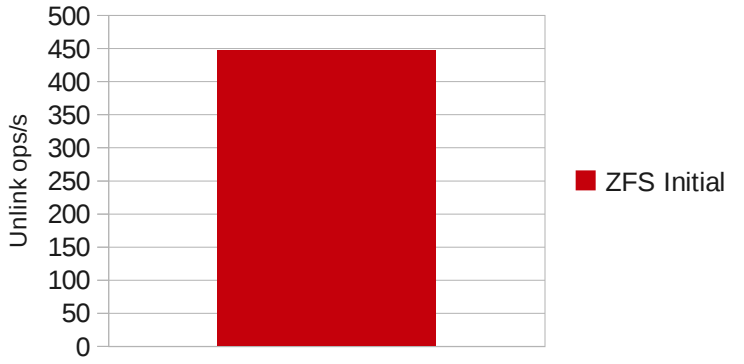
MDS Configurations

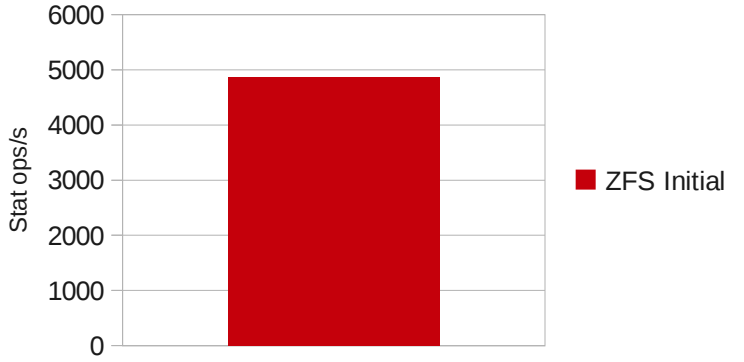


- SSDs used for increased IOPs
- All Linux disk management

Meta Data Performance

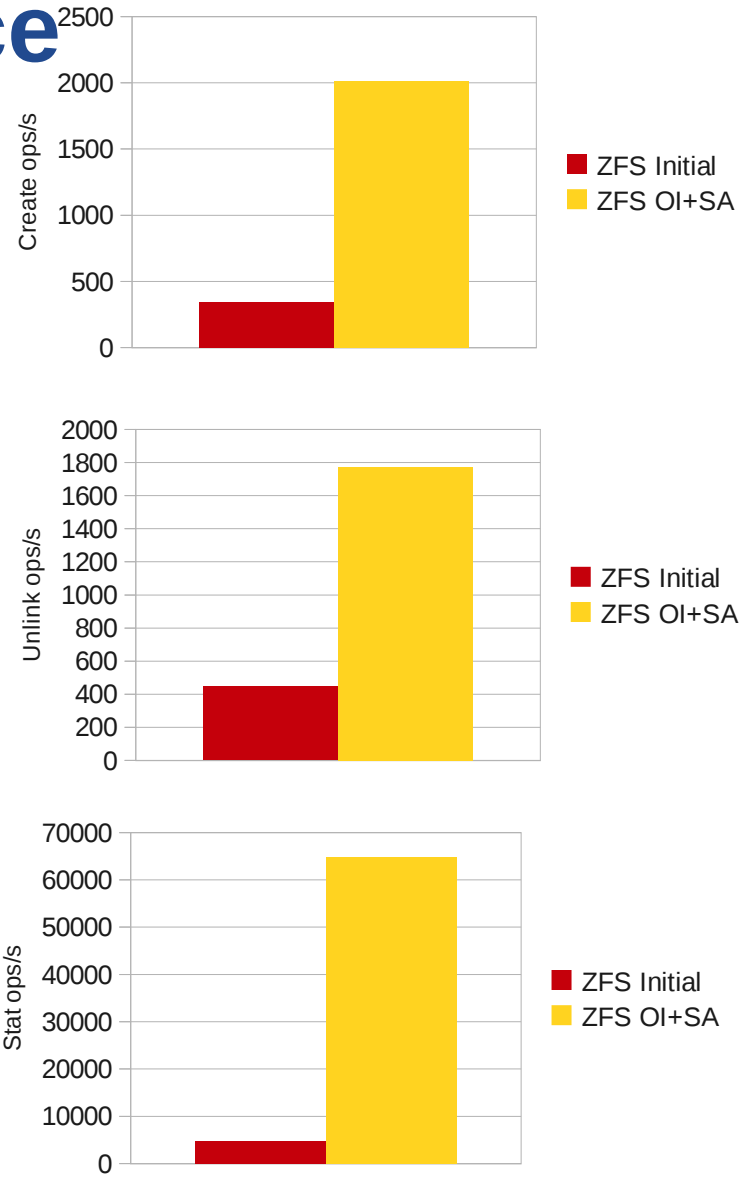
- MDTEST
  - Parallel create / stat / unlink
  - 1,000,000 files
  - Single directory
  - 52 Clients
- FID Hashing
  - Very poor hash distribution
  - Fixed upstream
- Deadlocks
  - ZFS+Lustre VM Integration
  - Lock inversions





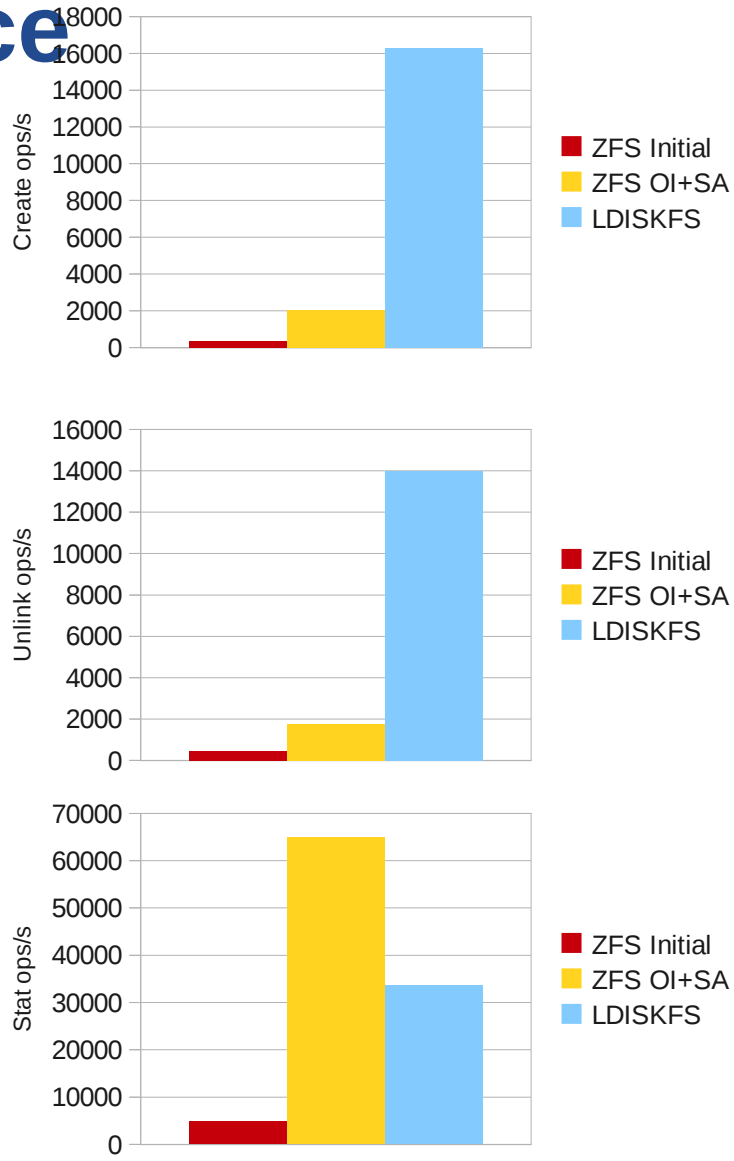
Meta Data Performance

- Multiple Object Indexes
  - Stores FID to object mapping
  - Implemented with a ZAP
  - ZAPs
    - Designed to scale for capacity
    - Concurrent updates contend
    - Insertion may require a disk I/O when leaf block is on disk
- System Attributes
  - Lustre relies heavily on xattrs
  - ZFS xattrs are flexible but slow
  - Store the xattr with the dnode

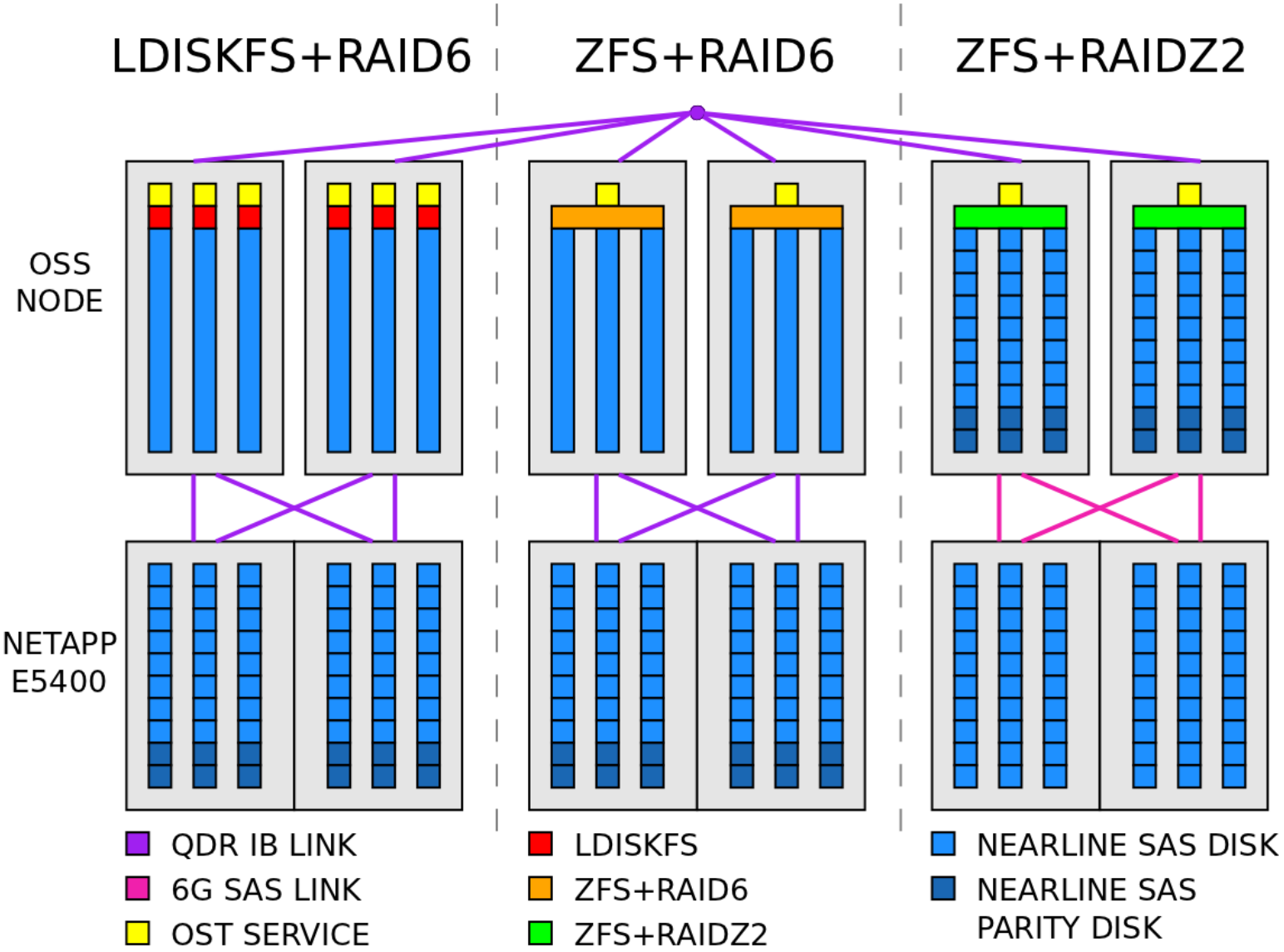


Meta Data Performance

- LDISKFS
  - Parallel directory improvements
  - ptlrpc improvements
- ZFS
  - Lock contention on MDT limiting creates and unlinks
  - Higher level Lustre layers are clearly capable
  - Improvements targeted for the ZFS layers

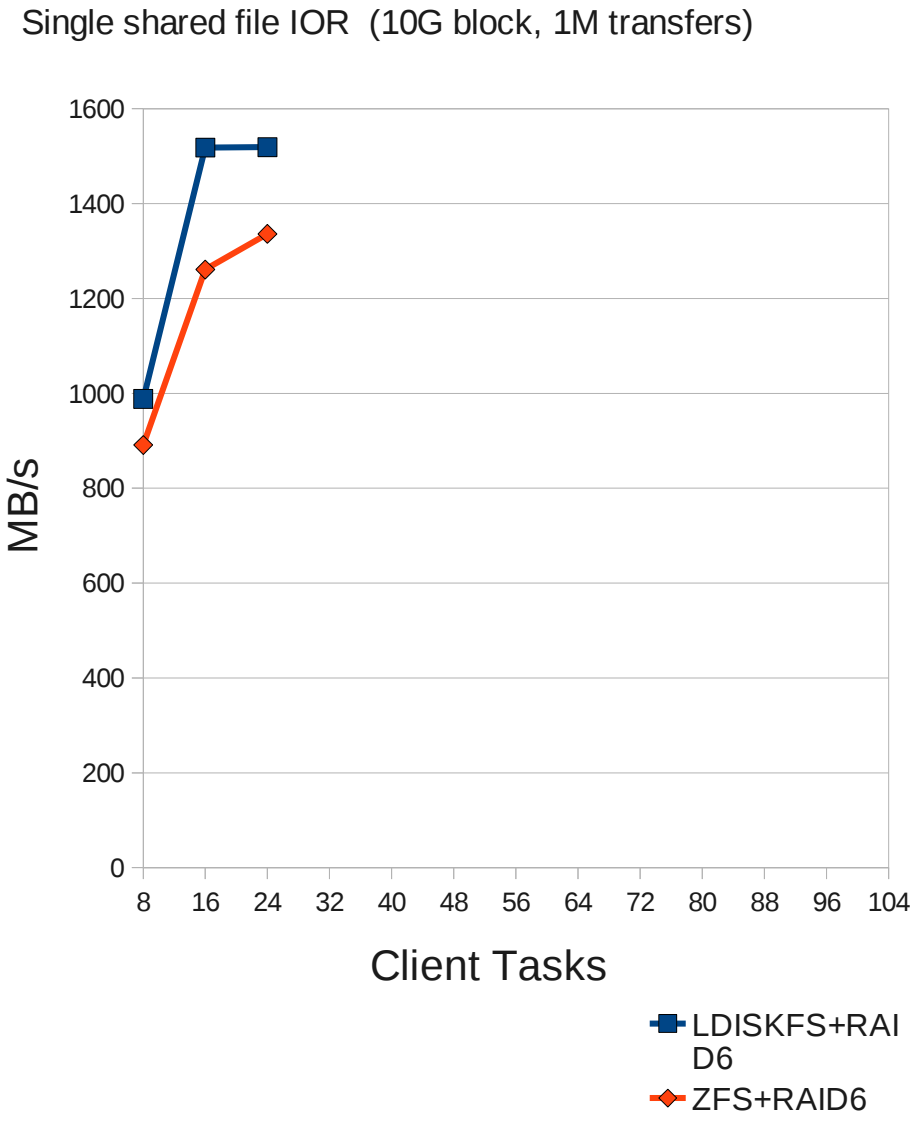


OSS Configurations



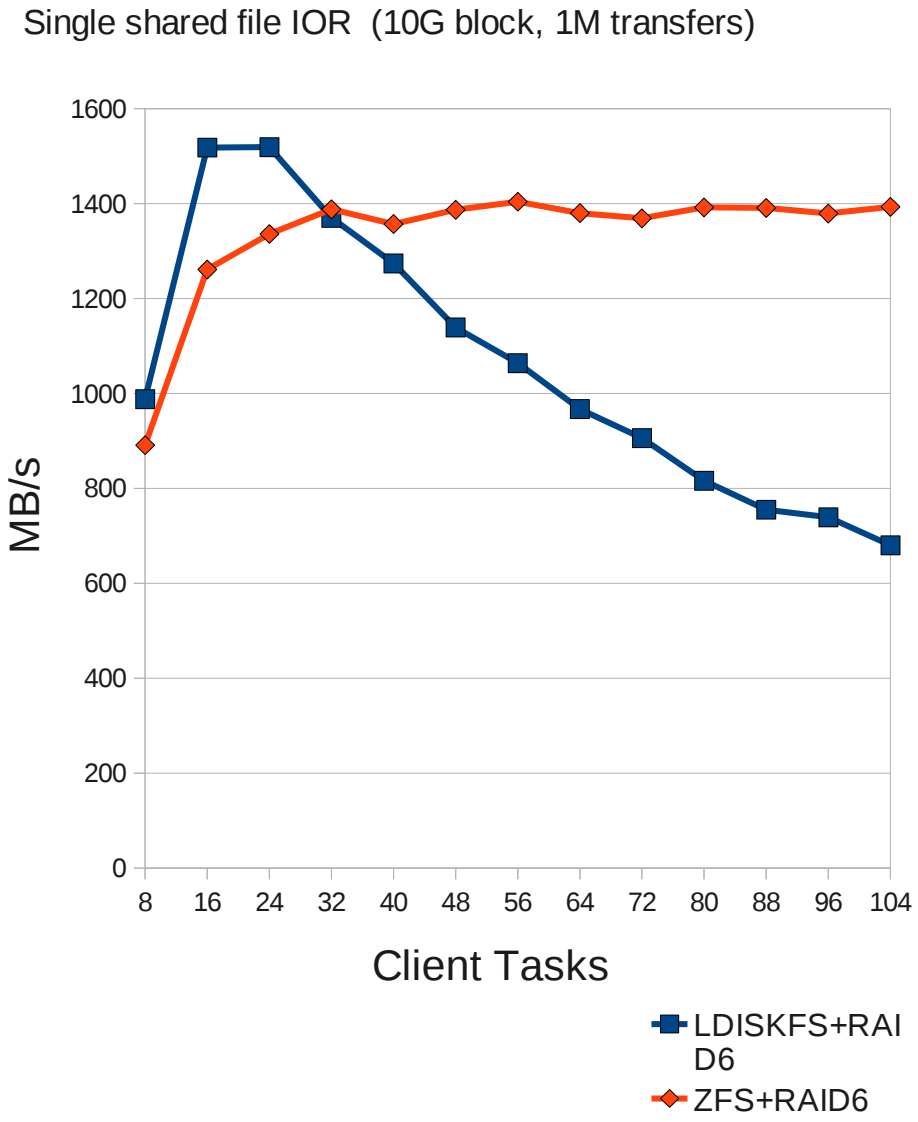
Write Performance

- IOR Test Case
  - Single Shared File
  - Single OSS
  - 30 Nearline SAS drives
- LDISKFS
  - Designed for this workload
  - Years of optimization
- ZFS
  - New Linux implementation
  - Unoptimized
  - Additional Overhead



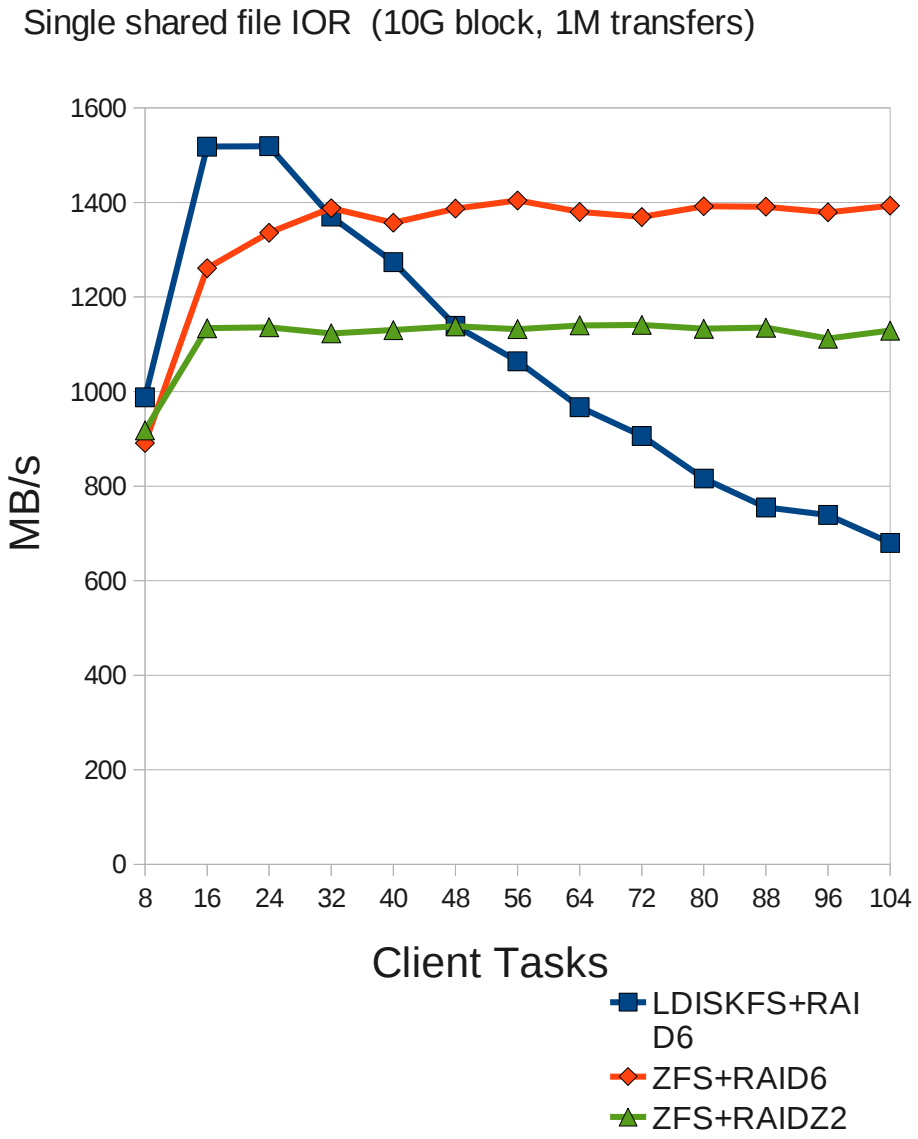
Write Performance

- Sequoia Workload
  - Defensive checkpoint I/O
  - 1,572,864 Compute Cores
  - 768 OSS Nodes
  - 2048 Tasks per OSS
- LDISKFS
  - Increasing tasks per OSS degrades performance
- ZFS
  - Constant performance



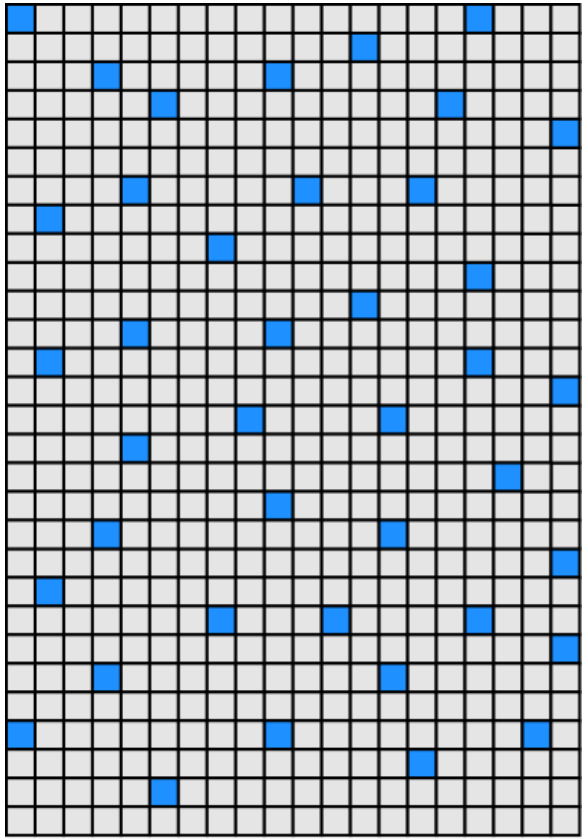
Write Performance

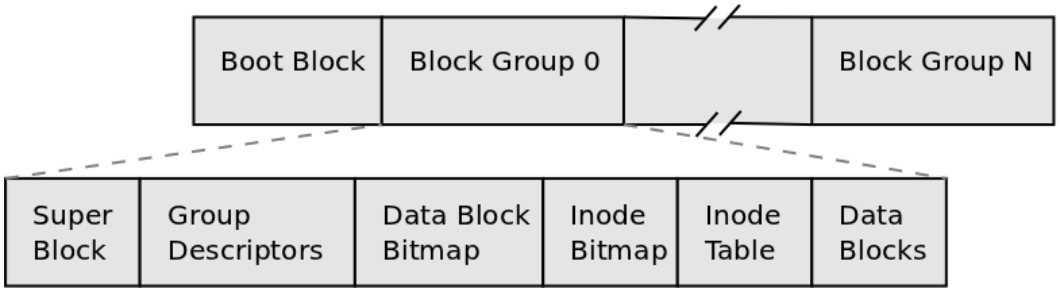
- Sequoia Workload
  - Defensive checkpoint I/O
  - 1,572,864 Compute Cores
  - 768 OSS Nodes
  - 2048 Tasks per OSS
- LDISKFS
  - Increased tasks per OSS degrades performance
- ZFS
  - Constant performance
  - Increase I/O size for RAIDZ2



LDISKFS On Disk Format

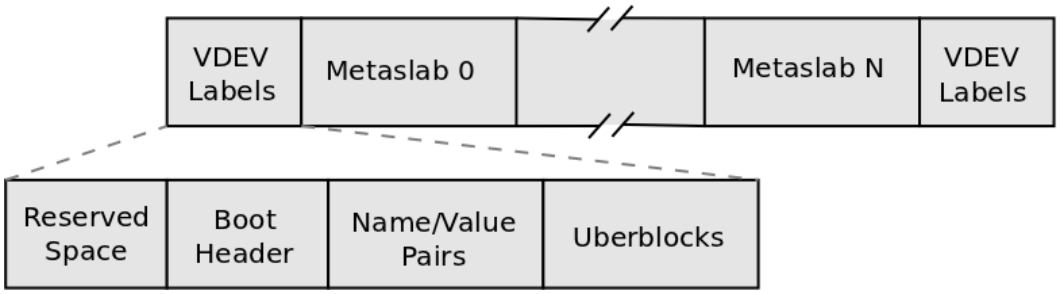
- Device is divided in to N statically allocated block groups
- Allocator has less flexibility over which inodes and blocks must be updated.
- Desirable for small file workloads
- Concurrent I/O degrades to random I/O

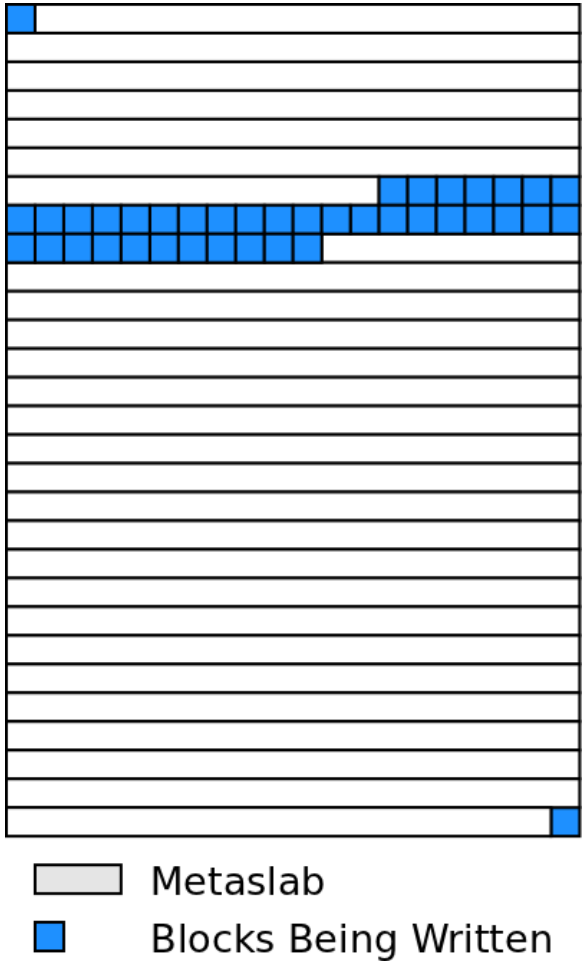




ZFS On Disk Format

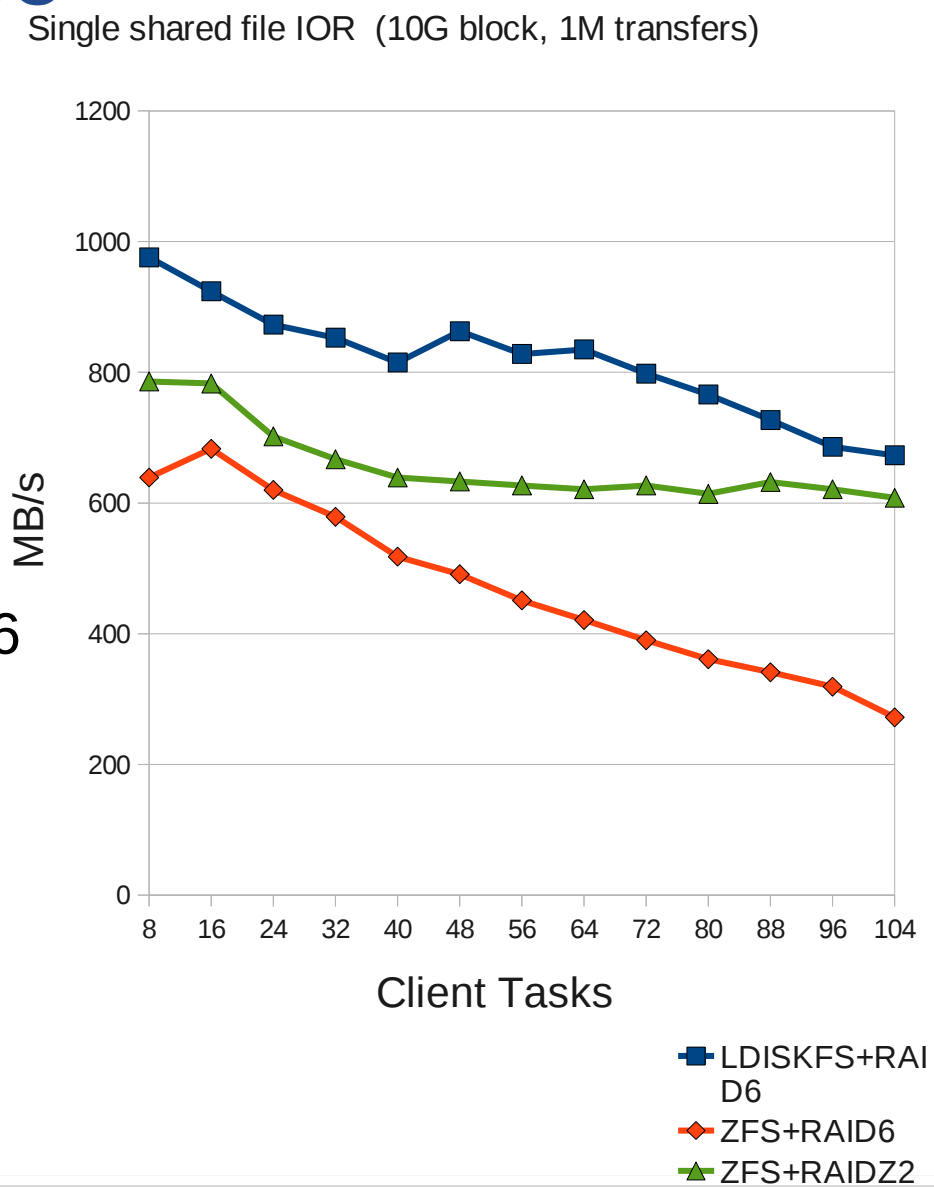
- Only vdev labels are statically allocated at beginning/end
- Copy-on-Write
  - New dnodes and blocks are written
  - Allocation decisions done at txg sync
  - All writes can be done sequentially
- Concurrent I/O can be sequentialized
- Increase I/O size





Read Performance

- Random I/O
- LDISKFS
  - mballoc allows larger I/O
- ZFS
  - 128K maximum block size
  - IOPs limited for ZFS+RAID6
- Network Request Scheduler
- Lustre aware prefetching



Open Questions

- Cache Devices
  - Only make sense for OSSs
  - ZFS Intent Log (ZIL)
    - For small synchronous I/O
    - Speed up lock cancels
  - L2ARC
    - Cache the object indexes
    - Generic read cache
- Compression
- Ditto blocks
- Snapshots

Summary

- Lustre+ZFS is usable
- But there are trade offs
  - ZFS - data integrity, scalability, manageability, extra features
  - LDISKFS – performance
- Expected in Lustre 2.4
- Website
  - http://zfsonlinux.org/lustre.html

ZFS on Linux

- Stable Release Candidate
  - ZFS 0.6.0-rc8
- Community
  - Maintained packages:
    - Ubuntu PPA
    - Gentoo ebuilds
    - Generic RPMs
  - Website
    - http://zfsonlinux.org
  - Mailing Lists
    - zfs-discuss@zfsonlinux.org
    - zfs-devel@zfsonlinux.org