ZFS on RBODs

Leveraging RAID Controllers for Metrics and Enclosure Management

International Workshop on The Lustre Ecosystem: Challenges and Opportunities. Annapolis, MD

March 4, 2015

Marc Stearman
Parallel File Systems Operations Lead

Lawrence Livermore National Laboratory

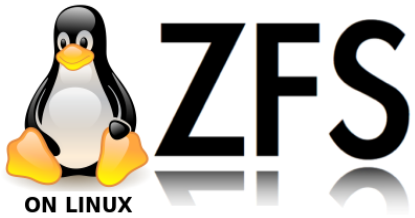
LLNL-PRES-668264
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Abstract

- Lustre requires reliable storage
  - Traditionally: ldiskfs with RAID controllers
- ZFS, added with Lustre 2.4 has RAID built into the software
  - Designed to work with JBODs (Just a Bunch of Disks)
- RAID controllers have value beyond RAID
  - Drive/Enclosure/Fault management
  - Metrics
  - Monitoring
- Can we leverage the strengths of both and run an RBOD in a JBOD style mode?

History Lesson

- Prior to Lustre 2.4
  - One backend FS: ldiskfs, based off Linux ext3 journaled file system.
  - Needed RAID controllers to provide fault tolerance
  - File System Checks took a long time offline
- The Sequoia Model
  - Lustre 2.4 added ZFS support
  - Risk mitigation was to put it on RAID Controllers
  - Not an optimal strategy for ZFS



RAID Controllers

Redundancy Schemes

T10-DIF and Parity Check on Read for Integrity

Firmware Management

Drive/Enclosure Services

Predictive Failure Analysis

Logging

Metrics and Reporting

Improved Reliability

ZFS+RAID = Redundancy + Integrity

Redundancy schemes

Stores integrity checksum in the parent block

Self healing data – mitigates silent data corruption

Dynamic stripe sizes – avoids read-modify-write

Copy-on-write transactional model

Online file system checking

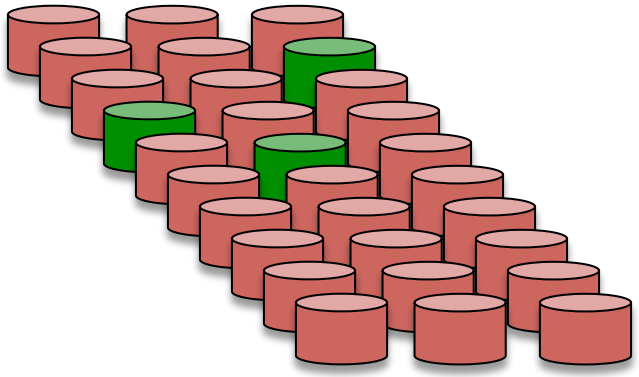
Snapshots, clones, compression, dedup, etc.

RAID Controller Value Add

RAID Controller less than 10% of the cost

Rich API

- read/write bytes
- histograms
- fault analysis
- firmware updates
- drive latencies
- Predictive Failures



The Green Drives are slow.
How do I know?

Work already done. No need to reinvent the wheel

ZFS on RBODs – The Sequoia Decision

Use the NetApp E5400 RAID engine

Stripe ZFS across RAID6 volumes

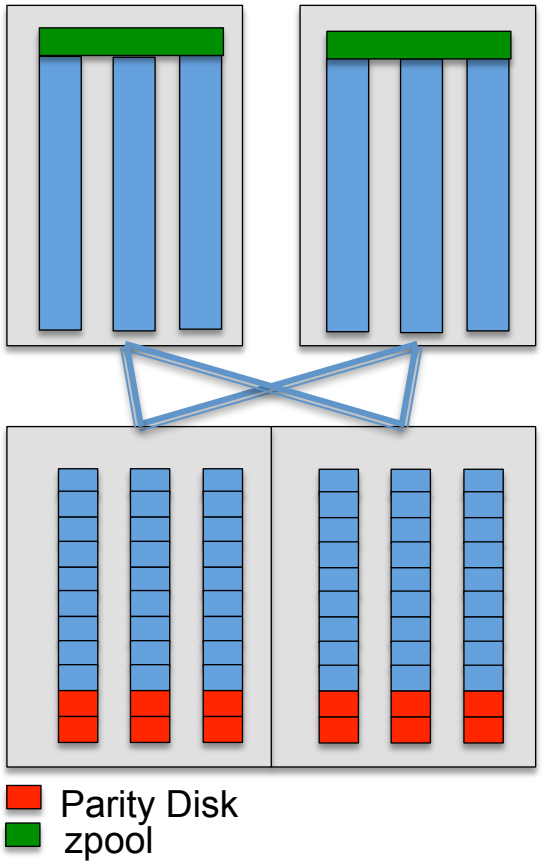
Mitigates risk

ZFS cannot self heal

Present Production Configuration

- NetApp E5400, with 60 drives using RAID6

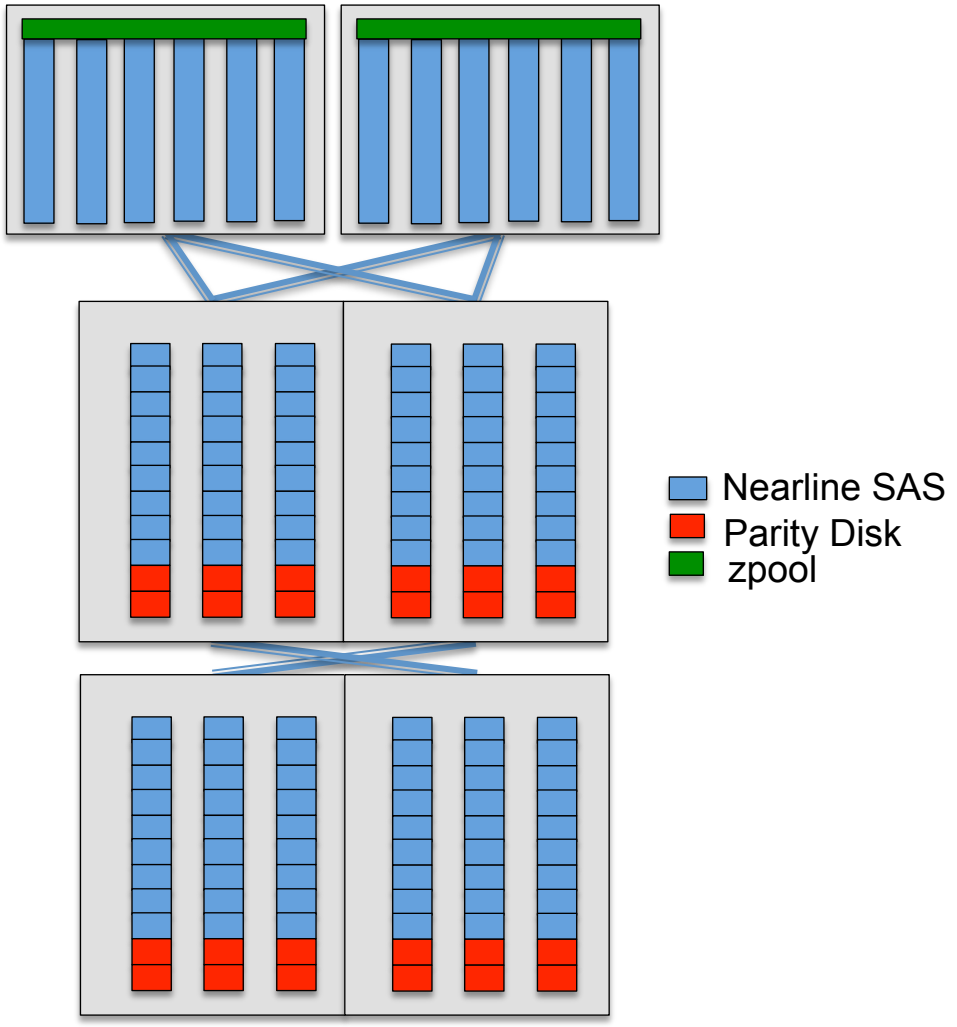
Three RAID6 (8+2) LUNS presented to each host.



A zpool is created by striping across the three LUNs

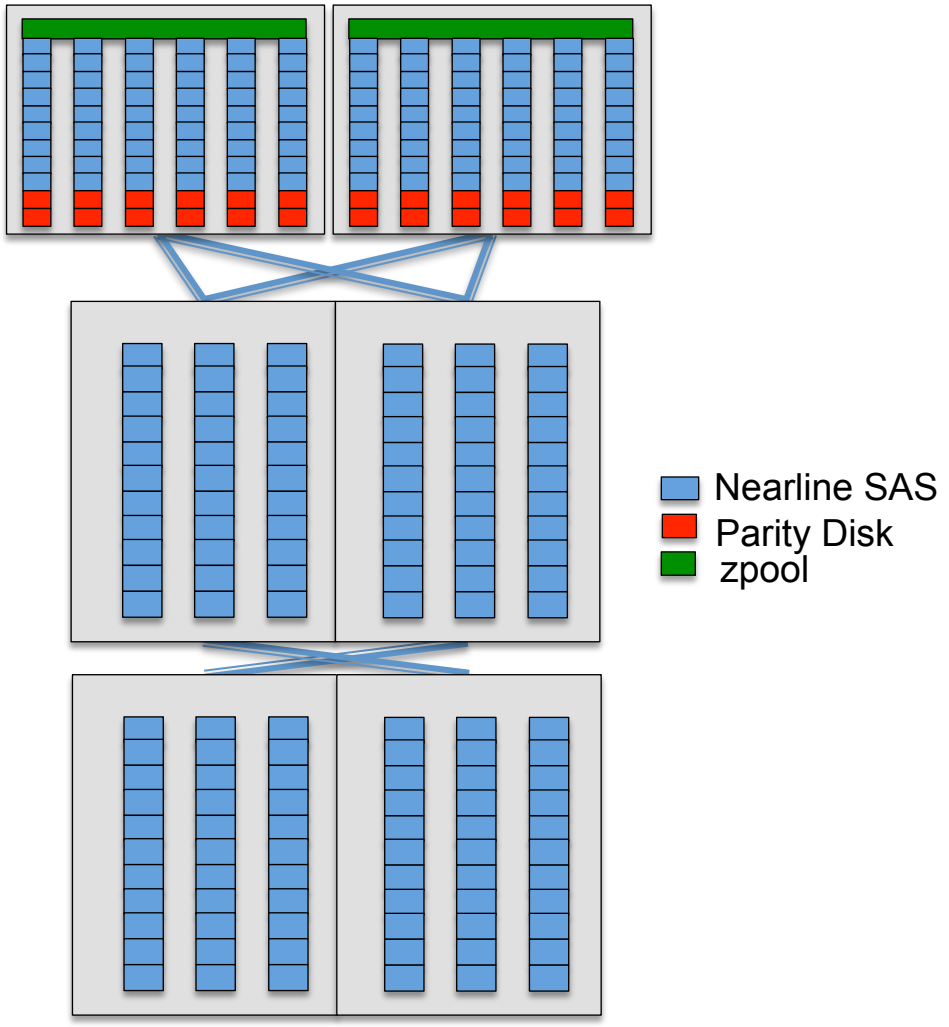
Test Configuration – RAID6

- NetApp E5500, with 120 drives using RAID6
- Six RAID6 (8+2) LUNS presented to each host.
- A zpool is created by striping across the six LUNs



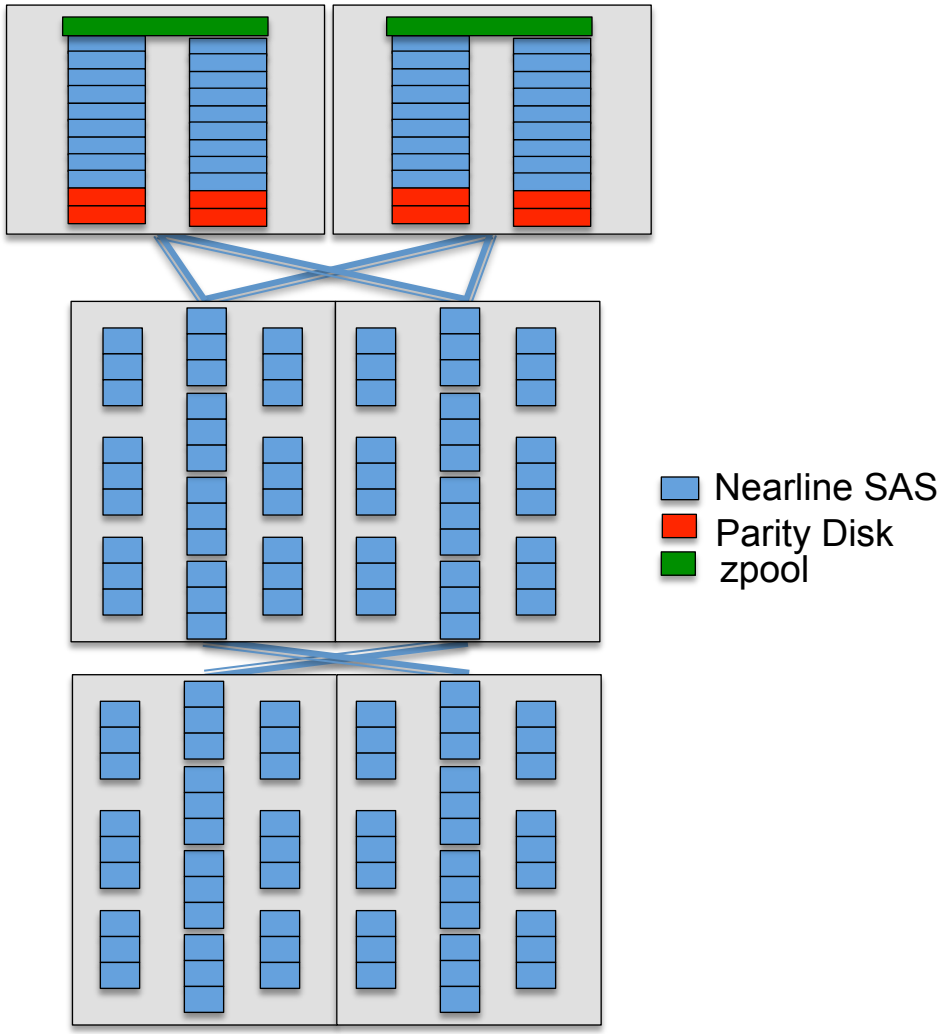
Test Configuration – 120 LUNs

- NetApp E5500, with 120 drives created as individual RAID0
- Six RAIDZ2 (8+2) vdevs created on each host as a zpool.
- Tested with and without the RAID controller



Test Configuration – 40 LUNs

- NetApp E5500, with 120 drives created as RAID0 LUNs in three drive groups
- Two RAIDZ2 (8+2) vdevs created on each host as a zpool.

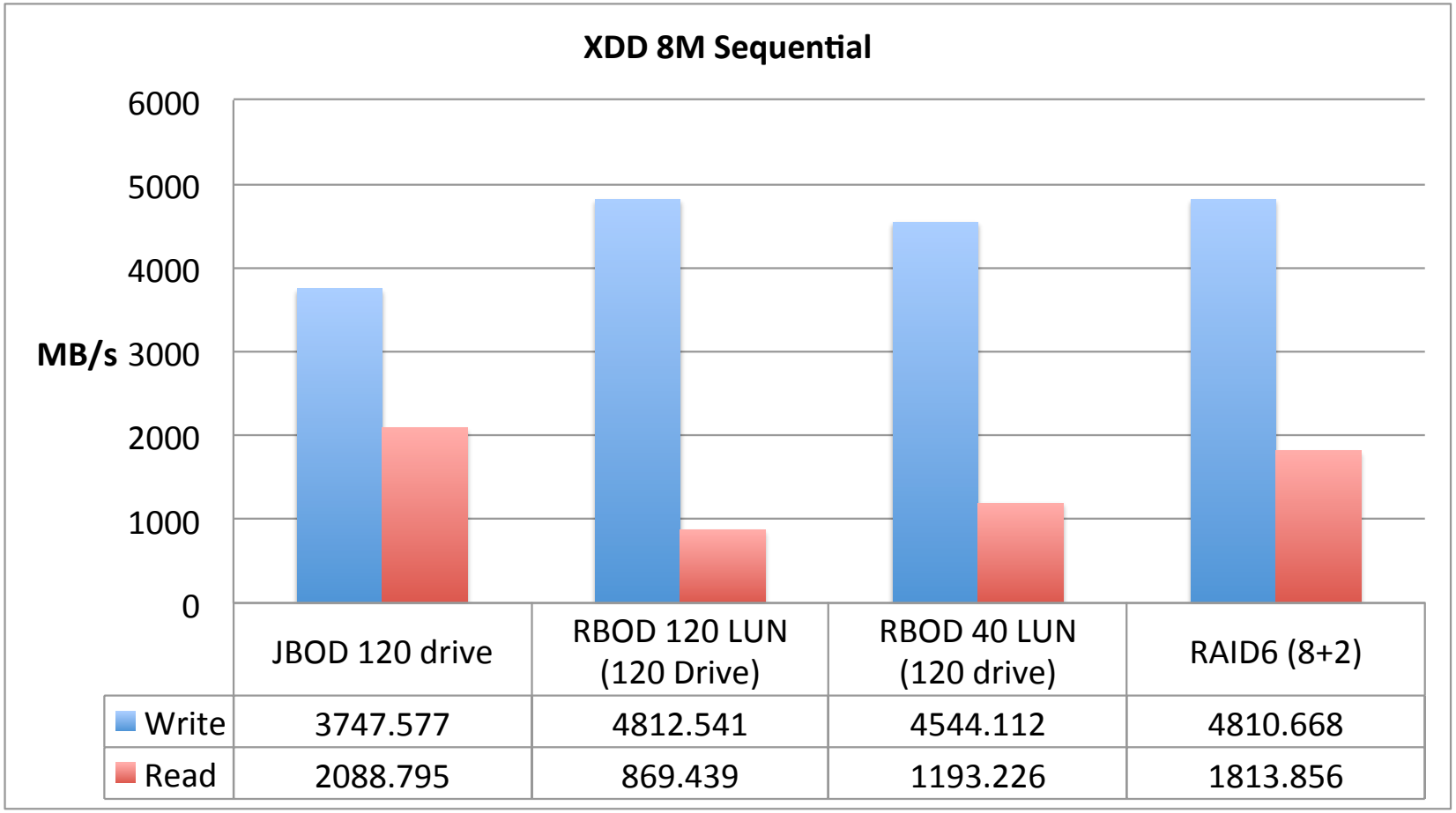


I need a Hero!!

- Hardware and ZFS have limits. XDD and FIO were used to find them
- XDD – Sequential Read/Write tests at 8M BS
- FIO – 2M block size Random Read/Write test modeled after the iometer example
  - size=64g
  - rwmixread=50
  - nrfiles=16
  - ioengine=posixaio
  - random_distribution=zipf:1.2
  - iodepth=32
  - fallocate=none
  - direct=0

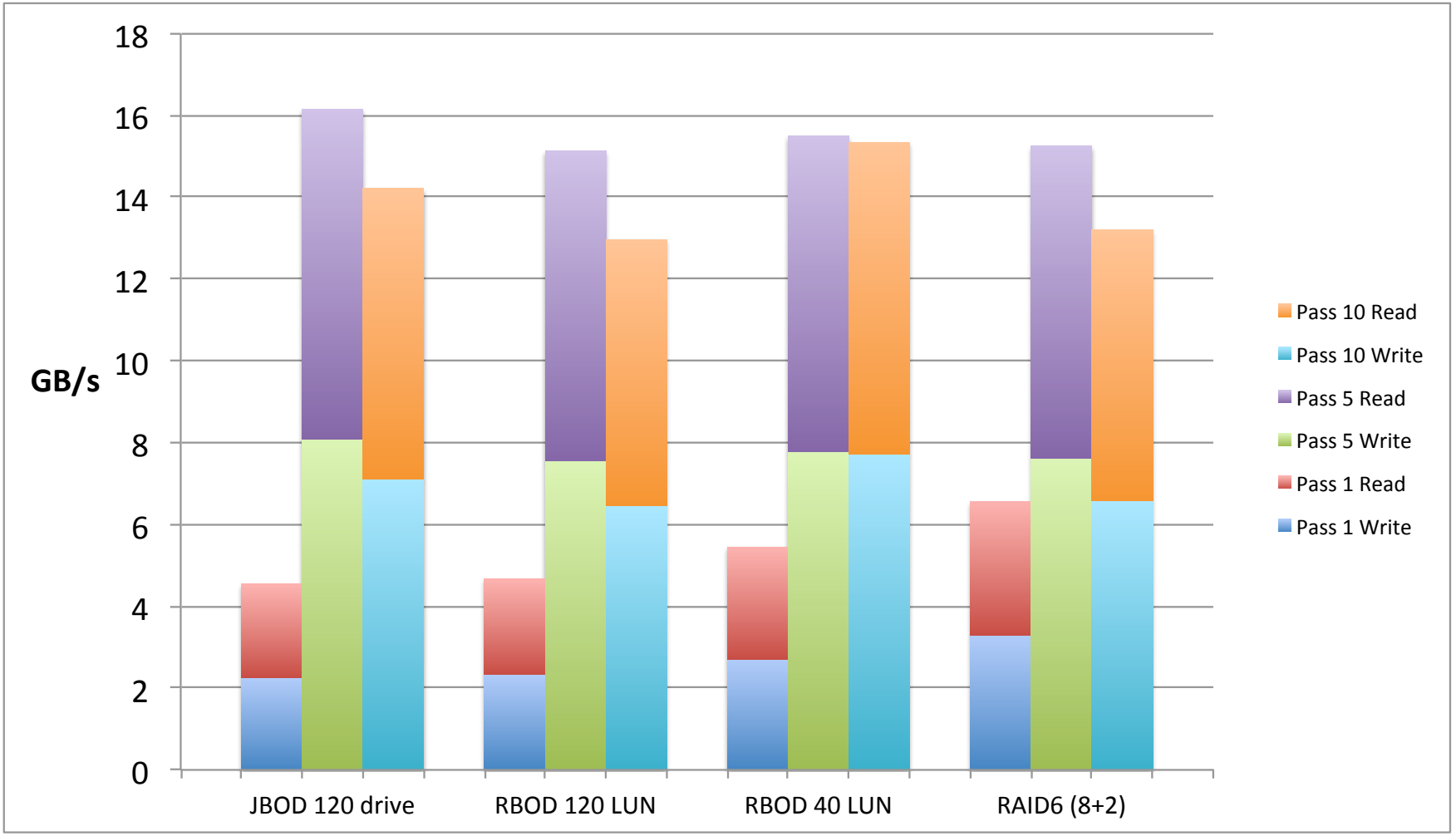
XDD writing to 10 files through the posix layer of ZFS.

Amount of data defeats the ZFS cache

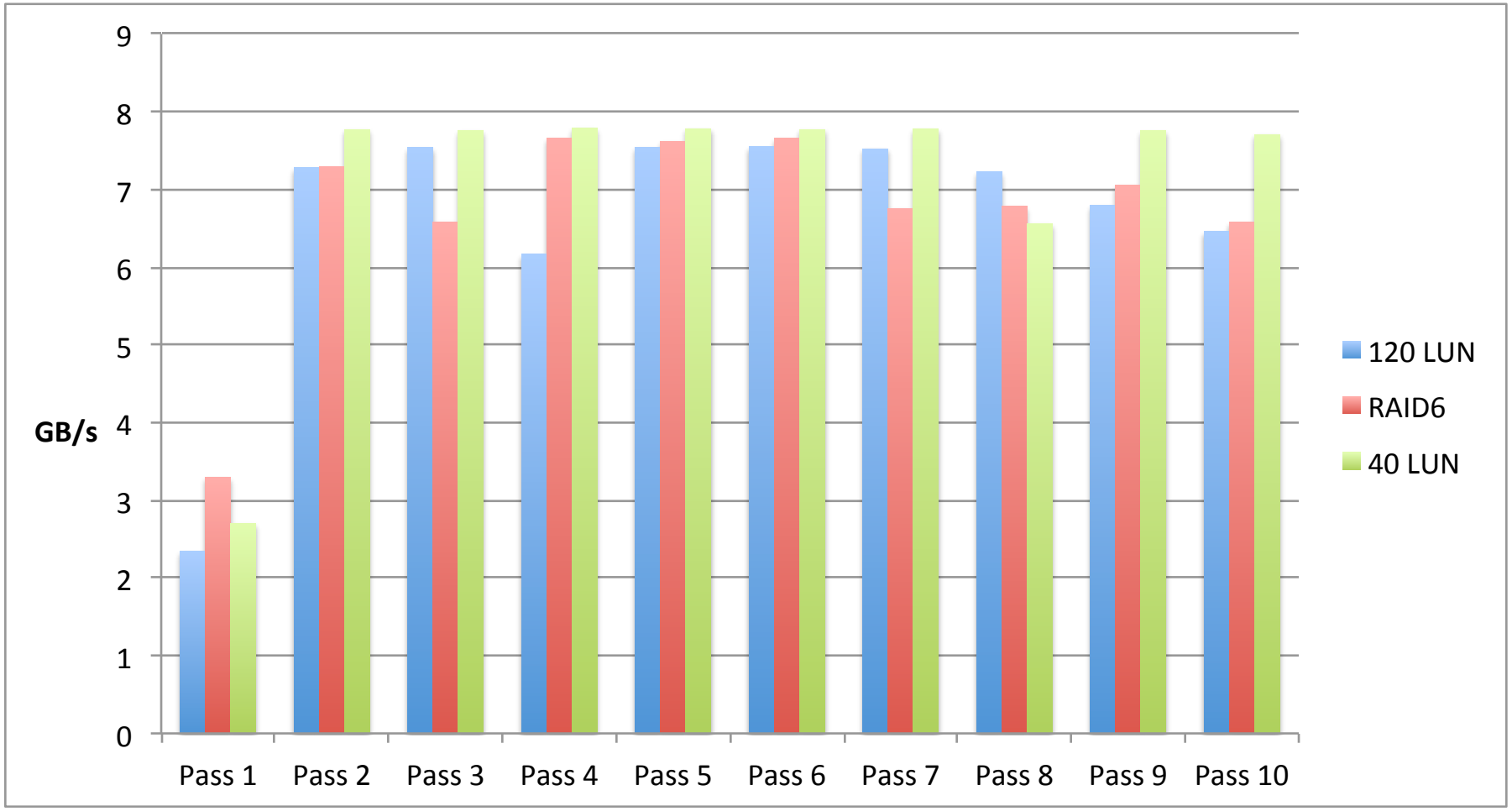


	JBOD 120 drive	RBOD 120 LUN (120 Drive)	RBOD 40 LUN (120 drive)	RAID6 (8+2)
Write	3747.577	4812.541	4544.112	4810.668
Read	2088.795	869.439	1193.226	1813.856

FIO shows what ZFS can do when you utilize the cache



FIO write progression as cache warms up



User Job Simulation

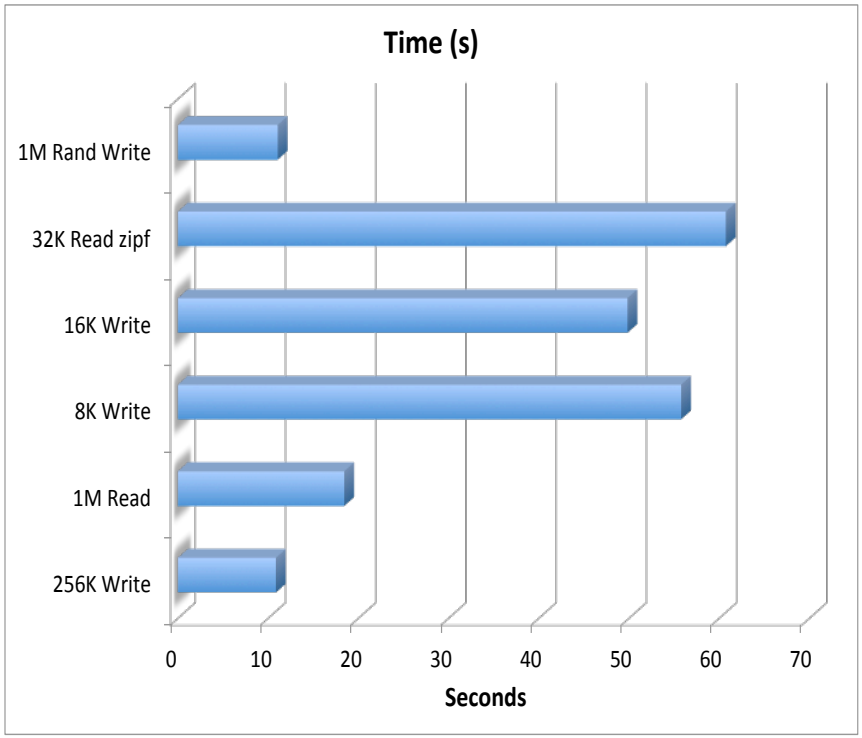
- Simulate something more realistic that is not a hero number
- ZFS is an intelligent file system and very hard to benchmark
- 92% of files on one of our Lustre file systems are smaller than 32KB in size
- How to defeat the ZFS cache, but avoid re-write workloads which are not realistic?

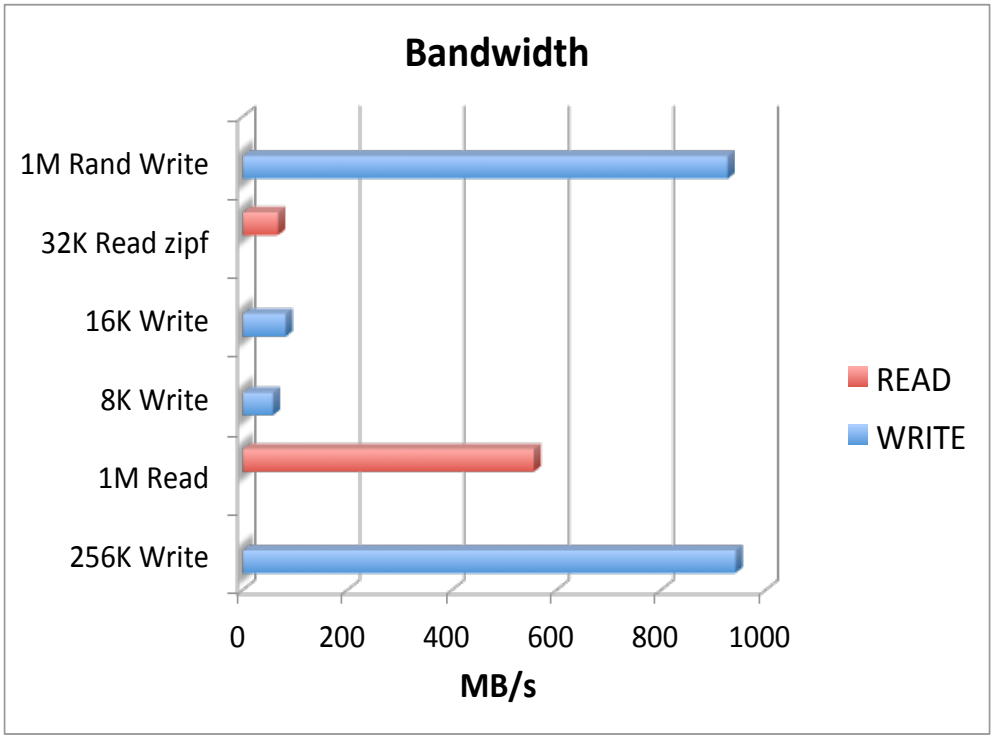
Job Mix – FIO, 6 jobs in parallel

- Global Settings: ioengine=posixaio, iodepth=32, create_on_open=1, direct=0, fallocate=none
- 1M Rand Write – 10,000 2MB files
- 32K Read – 100,000 64KB files
- 16K Write – 100,000 64KB files
- 8K Write – 100,000 32KB files
- 1M Reads – 10,000 3MB files
- 256K Write – 10,000 2MB files

Job Mix

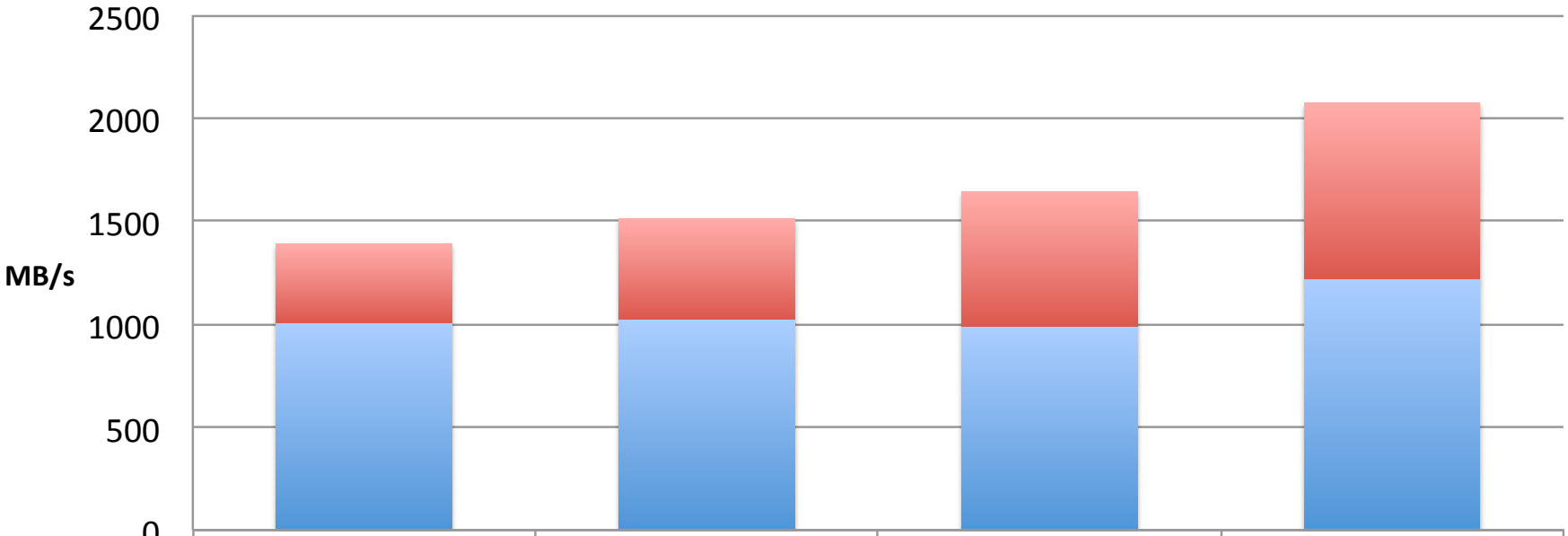
Smaller block sizes take longer to complete. Lager block sizes still get good bandwidth





Mixed workload helps defeat ZFS cache and simulates realistic work.

Avoiding the Hero numbers focuses on the workload.



	JBOD 120 Drive	RBOD 120 LUN (120 drive)	RBOD 40 LUN (120 drive)	RAID6 (8+2)
Read	381.964	489.714	657.859	852.72
Write	1007.884	1024.193	987.303	1219.862

Analysis

- RAID6 is about 20% faster on Writes and almost double on Reads
  - NetApp does not optimize for RAID0, but it is more likely due to the number of I/Os
  - ZFS with RAIDZ2 sends ten I/Os to the NetApp. With the NetApp doing the RAID, ZFS only sends one I/O.
  - The NetApp has a write cache and optimized RAID engines
  - RBOD mode dependent on drive path configuration

Analysis – Risk vs. Reward

- The RAID controllers add approximately 10% to the cost of the system.
- ZFS can fix bad data if it is responsible for RAID
- RAID controllers add value
  - Metrics – drive and channel heuristics, performance counters
  - Monitoring – Enclosure services, Predictive Failure Analysis
  - Firmware – Vendors work closely with drive manufacturers
  - Allow fine tuning of I/O paths
- Considering the RAID6 mode is fastest, why not just use it? You paid for it.

Future Discussion

- Running in RBOD mode is viable. Can the system be tuned for better performance based on workloads?
- Test ZFS large block patch. Does everyone win?
- Test at the Lustre layer. Lustre may introduce bottlenecks that even the playing field.
- Test a variety of JBODs to see performance differences
  - Did a small comparison with a Seagate box with 80 drives, and a Warp Mechanics box with 60 drives. The tests were comparable to the NetApp using extrapolation, but need more investigation.

Questions?

Marc Stearman

stearman2@llnl.gov

925-423-9670

Thanks to:

Brian Behlendorf, ZFS Guru, LLNL

John Miller, NetApp