LA-UR-20-26141

Approved for public release; distribution is unlimited.

Title:	Hunting for Bottlenecks in ZFS Failure Recovery using NVMe Drives
Author(s):	Bautista, Trevor Scott Parga, Alex Manno, Dominic Anthony
Intended for:	2020 HPC Intern Showcase, 2020-08-13 (Los Alamos, New Mexico, United States)
Issued:	2020-08-11 (Draft)

The Limits of ZFS Redundancy

Hunting for Bottlenecks in ZFS Failure Recovery using NVMe Drives

Trevor Bautista

13 August 2020

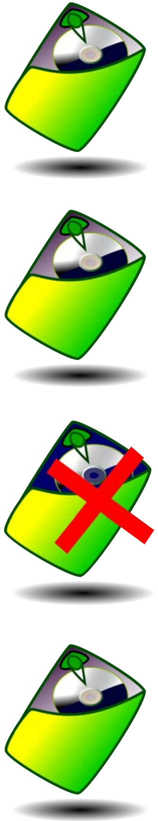
Managed by Triad National Security, LLC for the U.S. Department of Energy's NNSA

Background: Redundancy and Performance

- Data redundancy is important
  - Disks failure scales linearly with quantity
  - Data is important; takes time to compute (snapshotting)
- ZFS filesystem implements redundancy in software
  - Often is a “backbone” to other distributed filesystems
- Challenges to data redundancy
  - Resilver operations are costly, especially with capacious disks
  - HDDs are slow
    - Data recovery time depends on disk bandwidth (bottleneck)
- NVMe SSDs: Ideally eliminate disk bandwidth as a bottleneck
  - So… is there still a bottleneck?
- **Goal: Find bottlenecks in ZFS resilver performance**

Method: Testing ZFS Resilver Times in Different Scenarios

1. Create RAIDZ1 zpool (`zpool create ...`)
2. Set desired ZFS-related tunables
3. Take one drive offline, forcing drive fault (`zpool offline -f ...`)
4. Format offlined drive (`nvme format /dev/nvme1n1`)
5. Optional: start I/O load with fio
6. Re-online drive (`zpool online ...`)
7. Initiate resilver by replacing drive (`zpool replace ...`)
8. Gather CPU/RAM/disk metrics until resilver completes
    a. sar, pidstat, iostat
9. Clean up (fio remnants, background processes)
10. Store data
11. Go to step 2



Values: The Baseline

What to measure:

- Time to resilver
- Amount of data resilvered
- Disk utilization
- CPU & Memory usage

Important constants:

- Zpool filled to 60%
- 9x 1.5T NVMe SSDs
- Record size: 1M

Values: The Baseline

What to measure:

- Time to resilver/rebuild
- Amount of data resilvered/rebuilt
- Disk utilization
- CPU & Memory usage

No background I/O load

Important constants:

- Zpool filled to 60%
- 10x 1.5T NVMe SSDs
- Record size: 1M
- Zpool type: RAIDZ1

What to vary:

- Resilver type
  - Sequential, legacy
- Tunables related to resilver
  - zfs_resilver_min_time_ms
  - zfs_vdev_max_active
  - zfs_vdev_async_write_max_active

Values: Simulated I/O Workloads

What to measure:

- Time to resilver/rebuild
- Amount of data resilvered/rebuilt
- Disk utilization
- CPU & Memory usage

I/O load variations:

- 1M sequential read
  - Varying read behaviors
- 1M sequential write
- 4K random read / write
- 1M mix (20% read / 80% write)

Important constants:

- Zpool filled to 60%
- 10x 1.5T NVMe SSDs
- Record size: 1M

What to vary:

- Zpool type
  - RAIDZ, dRAID
- Resilver type
  - Sequential, legacy
- Tunables related to resilver
  - zfs_resilver_min_time_ms
  - zfs_vdev_max_active
  - zfs_vdev_async_write_max_active

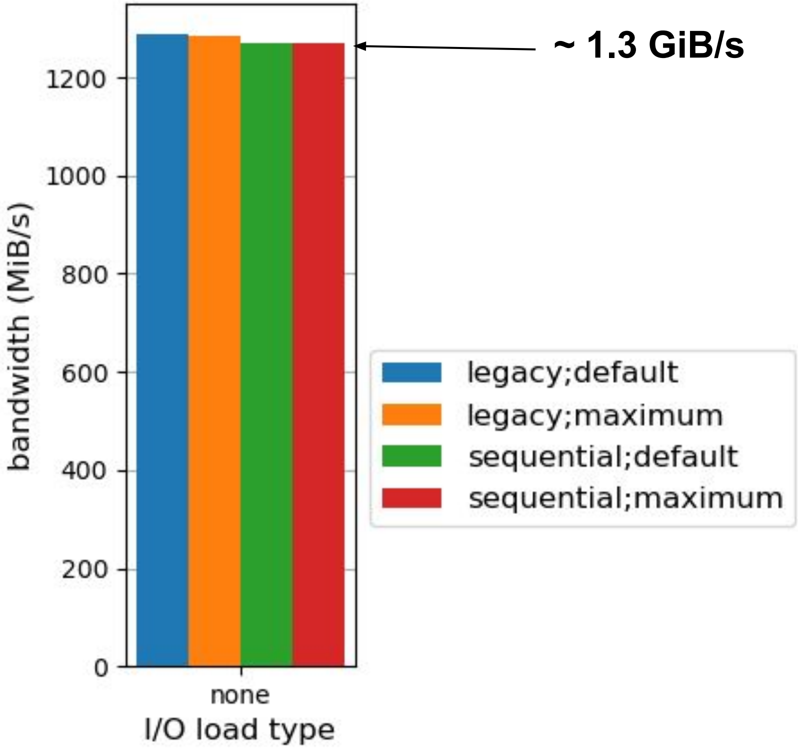
Data Analysis

Disk Benchmarks

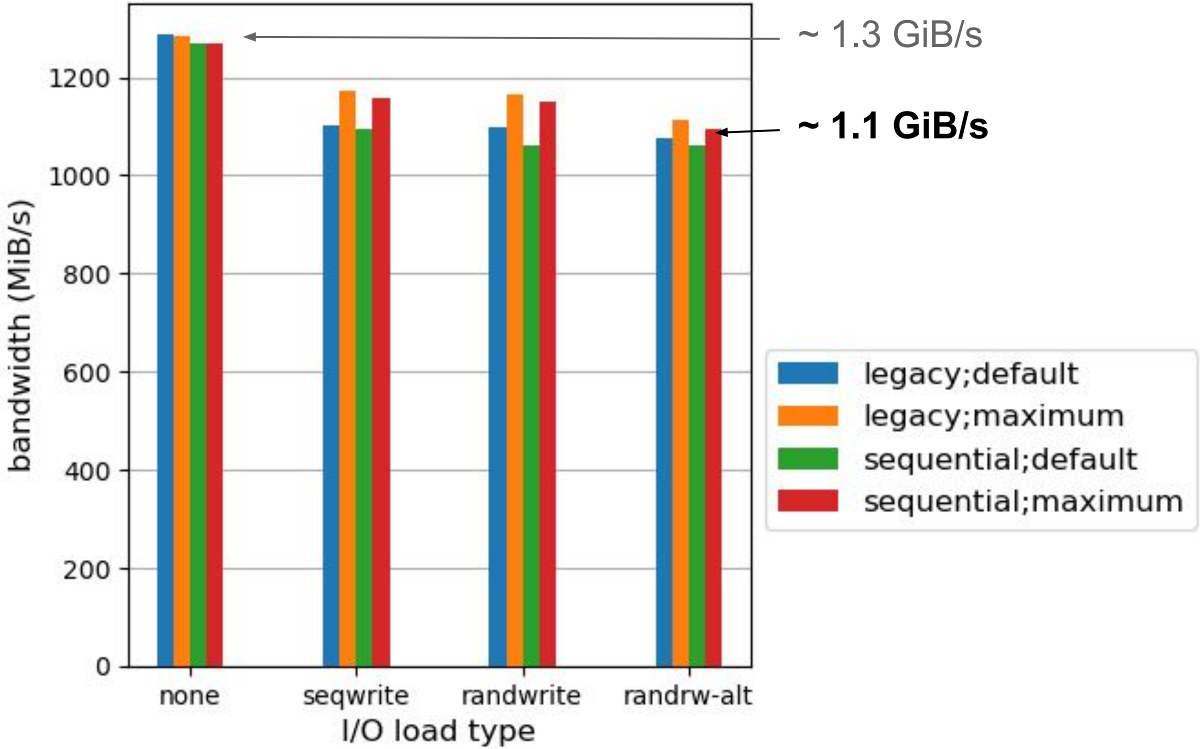
- **Best:** sequential read (~3.5 GiB/s)
- **Worst:** random read/write (~710 MiB/s)

		Multiple Test (loops=5, 1M seq, 4K rand, numjobs=32, iodepth=16)								
		Drive								
		nvme0n1	nvme1n1	nvme2n1	nvme3n1	nvme4n1	nvme5n1	nvme6n1	nvme7n1	nvme8n1
Seq Read	Avg IOPS	3,414	3,414	3,414	3,414	3,414	3,414	3,414	3,414	3,414
	Avg BW (MiB/s)	3,414.50	3,414.74	3,414.77	3,414.66	3,414.78	3,414.71	3,414.33	3,414.05	3,414.88
Seq Write	Avg IOPS	1,794	1,905	1,842	1,841	1,820	1,838	1,802	1,812	1,798
	Avg BW (MiB/s)	1,794.96	1,905.59	1,842.38	1,841.07	1,820.68	1,838.67	1,802.65	1,812.40	1,798.20
Rand Read	Avg IOPS	697,216	691,372	690,914	689,801	695,298	694,390	681,698	690,185	685,381
	Avg BW (MiB/s)	2,723.50	2,700.67	2,698.88	2,694.54	2,716.01	2,712.46	2,662.88	2,696.04	2,677.27
Rand Write	Avg IOPS	373,978	428,880	376,811	374,890	373,143	375,793	373,826	373,955	373,015
	Avg BW (MiB/s)	1,460.85	1,675.31	1,471.92	1,464.42	1,457.59	1,467.94	1,460.26	1,460.76	1,457.09
Rand Read/Write	Avg IOPS R/W	182,210/ 182,135	190,788/ 190,709	182,732/ 182,656	182,657/ 182,581	182,175/ 182,099	183,416/ 183,340	182,651/ 182,576	182,081/ 182,006	182,038/ 181,963
	Avg BW (MiB/s) R/W	711.76/ 711.46	745.27/ 744.96	713.80/ 713.50	713.50/ 713.21	711.62/ 711.33	716.47/ 716.17	713.48/ 713.19	711.26/ 710.96	711.09/ 710.79

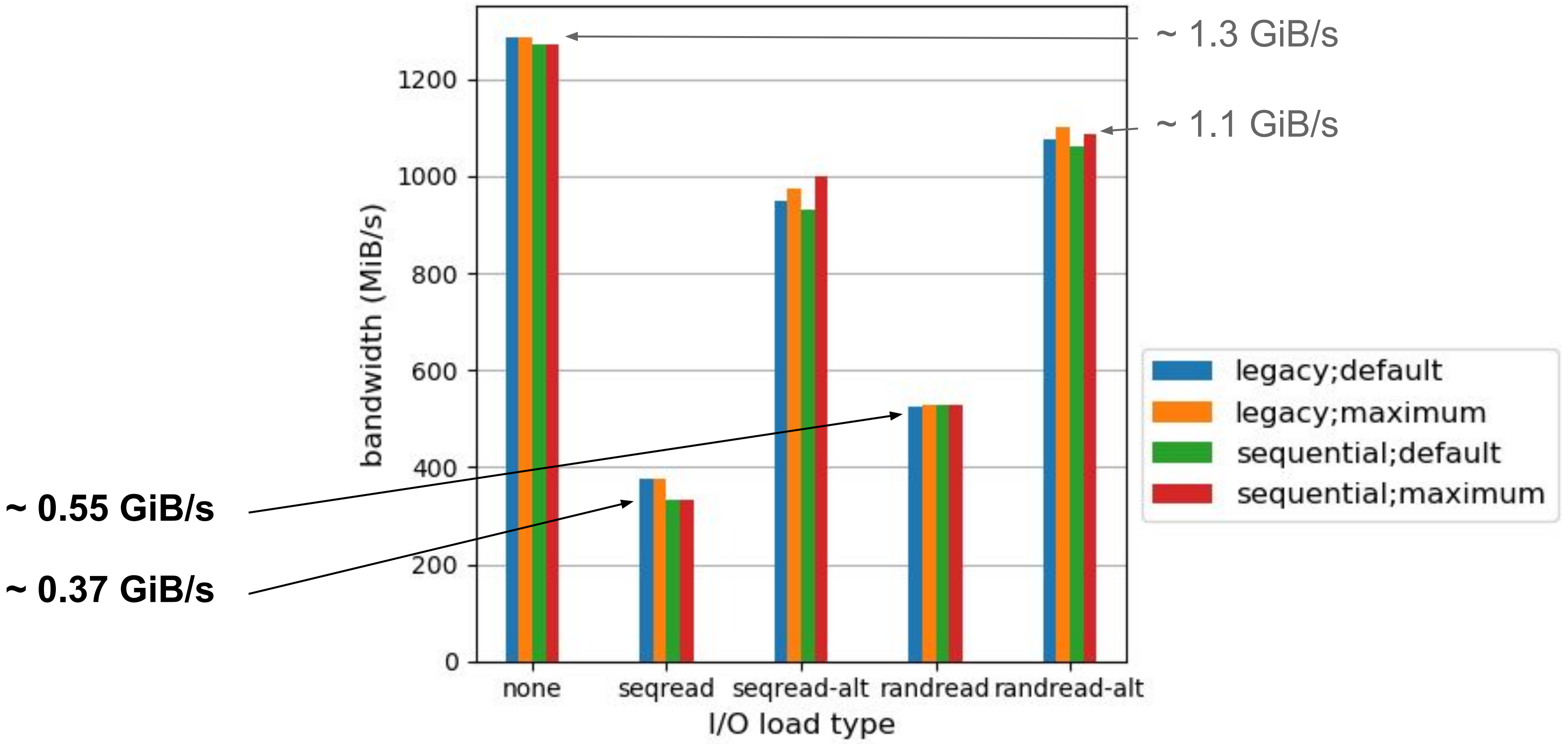
Reported Average Resilver Bandwidths: Baseline



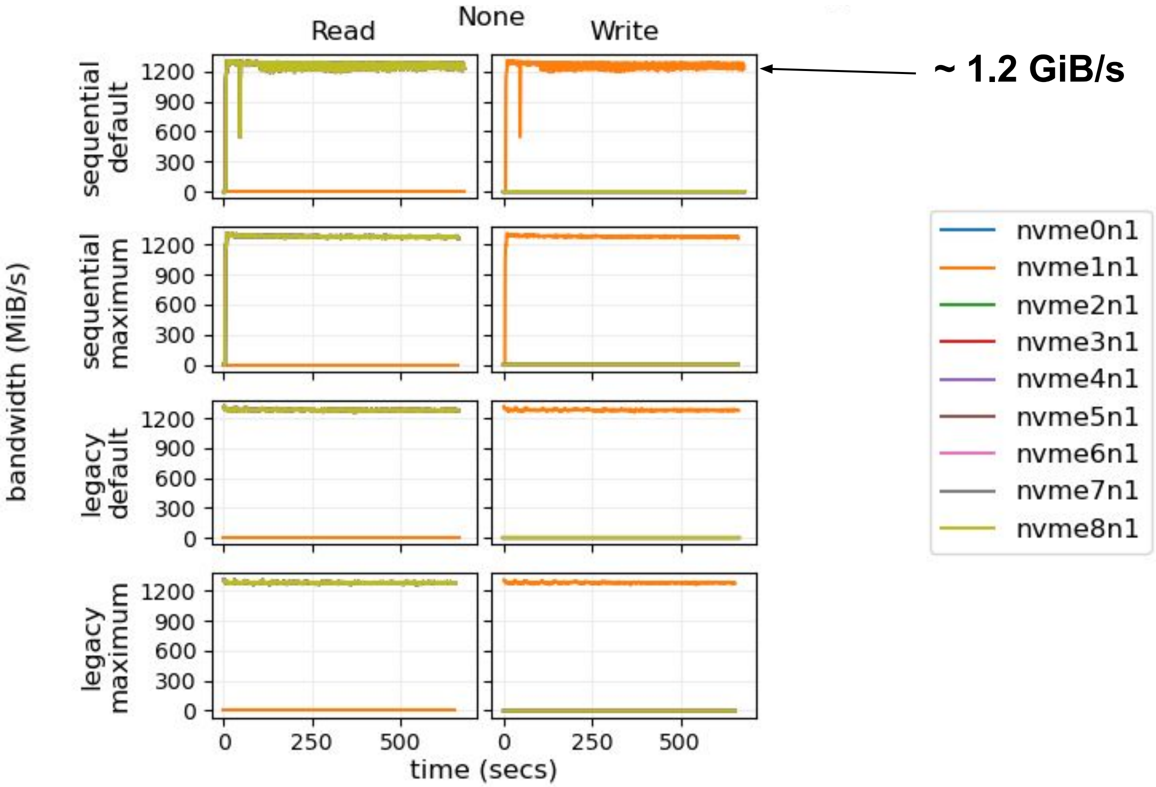
Reported Average Resilver Bandwidths: Write I/O



Reported Average Resilver Bandwidths: Read I/O

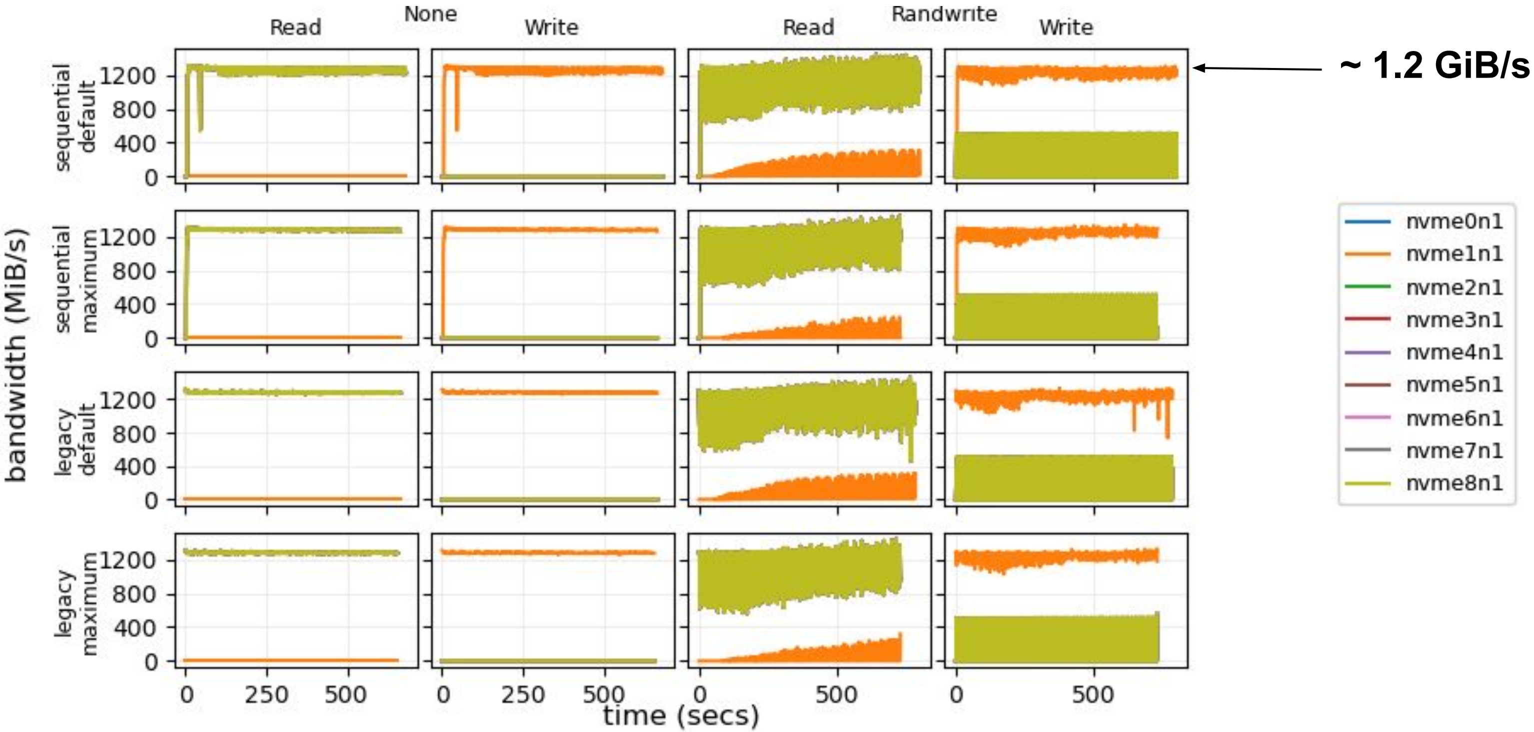


Reported Disk Bandwidths: Baseline



Measured ZFS Resilver (write): ~1.3 GiB/s

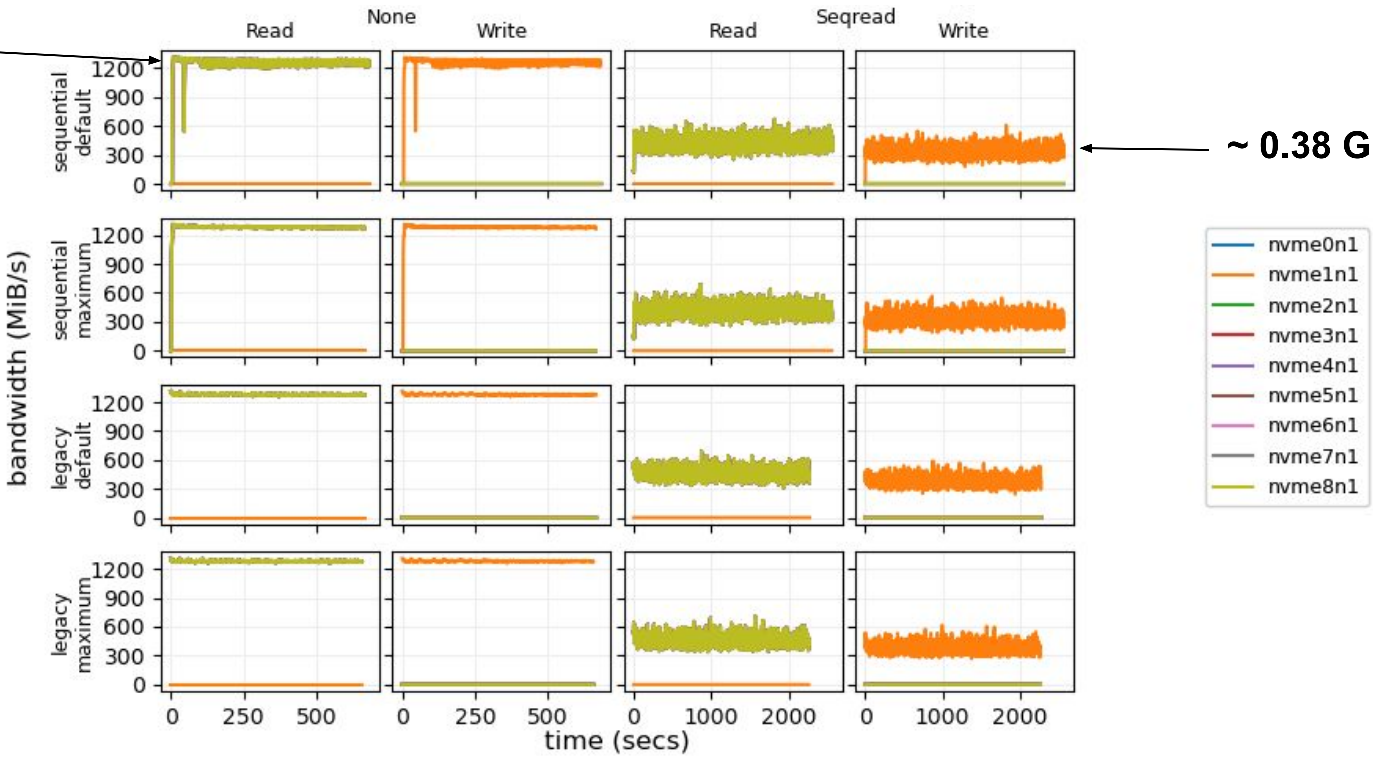
Reported Disk Bandwidths: Write I/O



Disk Write Benchmark: ~ 1.5 GiB/s

Reported Disk Bandwidths: Seq Read I/O

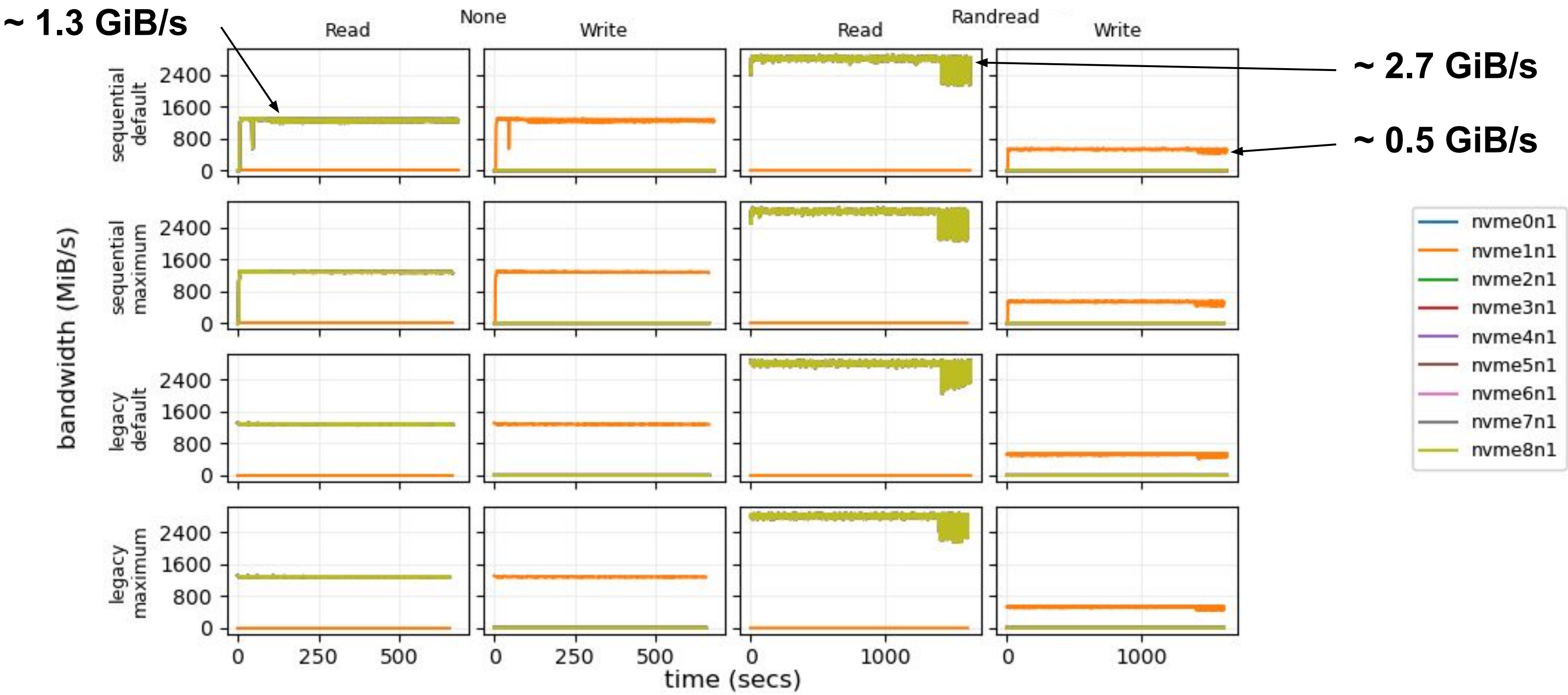
~ 1.3 GiB/s



~ 0.38 GiB/s

Disk Seq Read Benchmark: ~ 3.4 GiB/s

Reported Disk Bandwidths: Rand Read I/O



Disk Rand Read Benchmark: ~ 2.7 GiB/s

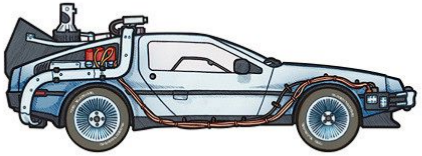
What can we conclude?

Interpreting the Data

- No I/O load: drives still do not reach benchmarked values
  - Resilver read bandwidth limited by write bandwidth of replaced drive
  - Presence of mixed I/O types (read/write) per disk
- Given these workloads: varying tunables/resilver does not vary resilver time by much
- **Bottleneck** in RAIDZ: dependence on write bandwidth of replaced drive
  - Still a reasonable performance target (minor bottleneck)
- **Bottleneck** in ZFS resilver during read workload
  - Reading un-resilvered data interferes with resilver
    - Mechanism yet to be explored

Looking Ahead

- What about dRAID?
  - Issues during resilver tests (kernel panic during rebuild)
  - Explore rebuild/reprotect operations
- Improving ZFS behavior in certain scenarios
  - Read-during-resilver mechanism
- Run more tests for more variation with the same workloads
  - Eliminate possible outlier data
- Compare to ZFS mirror performance
  - Parity vs. copy



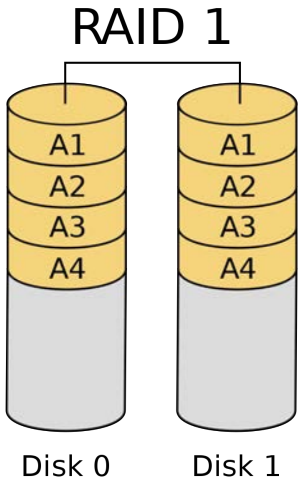
Questions?

Backup Slides

Background

What is Redundancy?

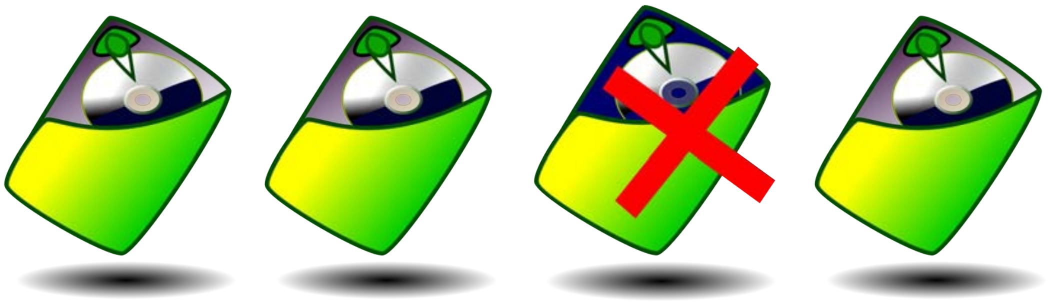
- Storage: The presence of extra information
  - Used to reconstruct existing information after a failure
  - Erasure codes
  - Can be full data copies or parity
- Examples
  - ECC Memory
  - Erasure codes
  - RAID
  - ZFS



Source: Wikimedia Commons https://upload.wikimedia.org/wikipedia/commons/thumb/b/b7/RAID_1.svg/800px-RAID_1.svg.png

Why Redundancy?

- Disks fail.
  - Many disks = high probability of failure
- Data is important.
  - Most tasks are mission critical
- Data can take time to compute.
  - E.g., time-intensive simulations



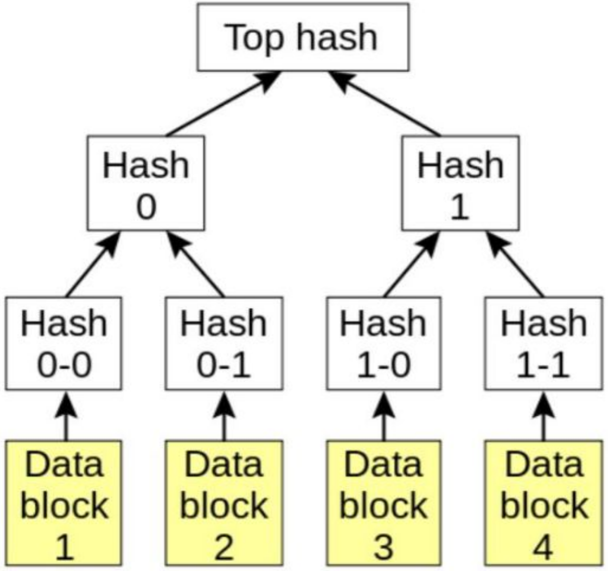
Common Implementation: ZFS

ZFS: Filesystem/Logical Volume Manager

- Allows redundancy in software
  - RAIDZ
  - dRAID
  - Mirroring
- Often is a “backbone” to other distributed filesystems
  - Lustre OSTs, MDTs
- Open source
- “Resilver”/“rebuild” operations for RAIDz/dRAID
  - How do these operations affect I/O performance?

ZFS: Relevant Terminology

- **Scrub**
  - traverse block pointers, comparing checksum to existing one
- **Resilver**
  - Reconstruct data by traversing block pointers
  - Scrub data during operation
  - Two types: sequential and legacy
- **RAIDZ**
  - RAID-like zpool configuration
    - 1, 2, or 3 parity units
  - Parity not limited to single disk



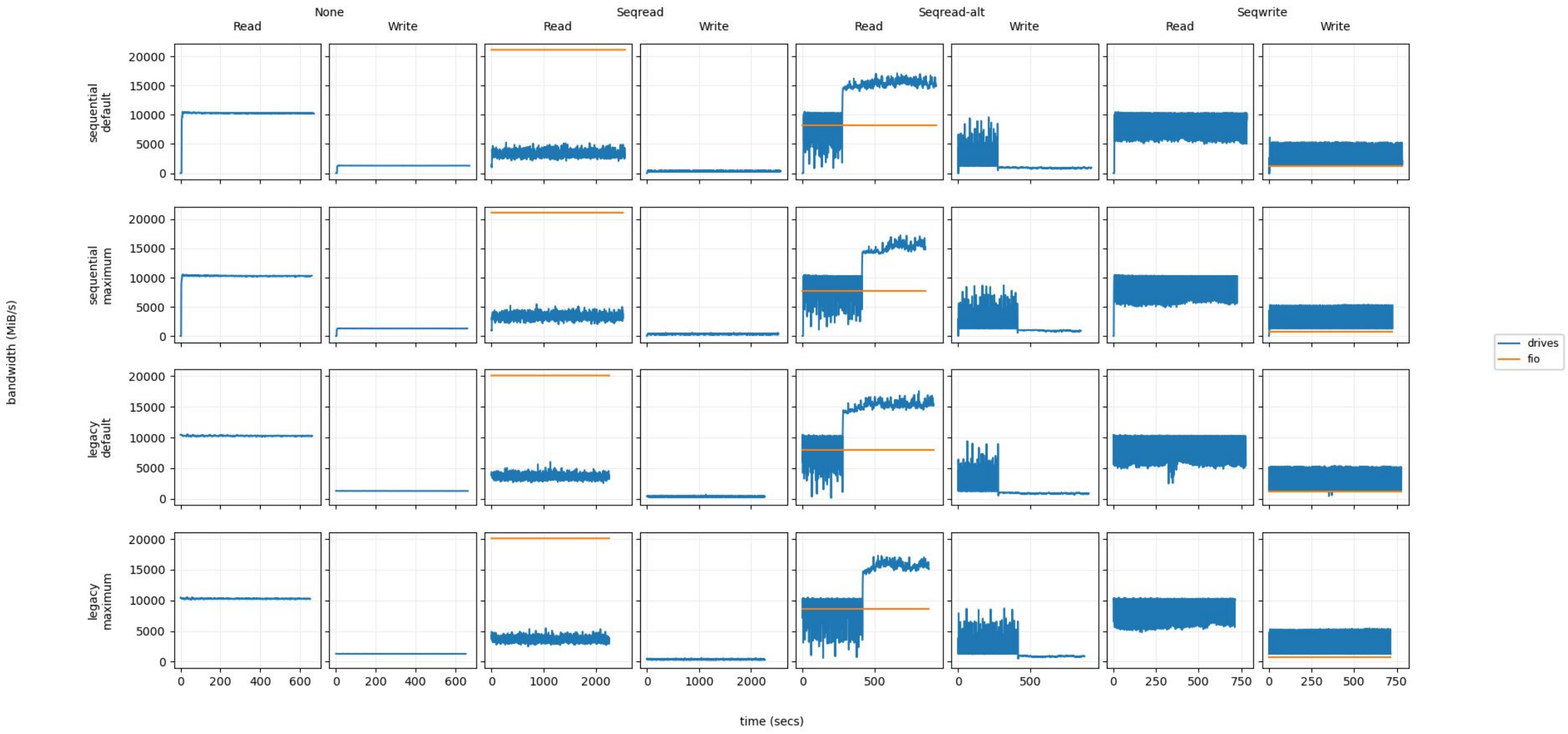
Challenges With ZFS

- Resilver operations are costly
  - Parity calculation involves many I/Os
- Conventional HDDs are slow...
  - Data recovery can take a long time
  - Disk bandwidth is a significant bottleneck for data recovery
- NVMe SSD drives are fast!
  - Ideally eliminates disk bandwidth as a bottleneck
  - So… is there still a bottleneck?
    - How can we find out?

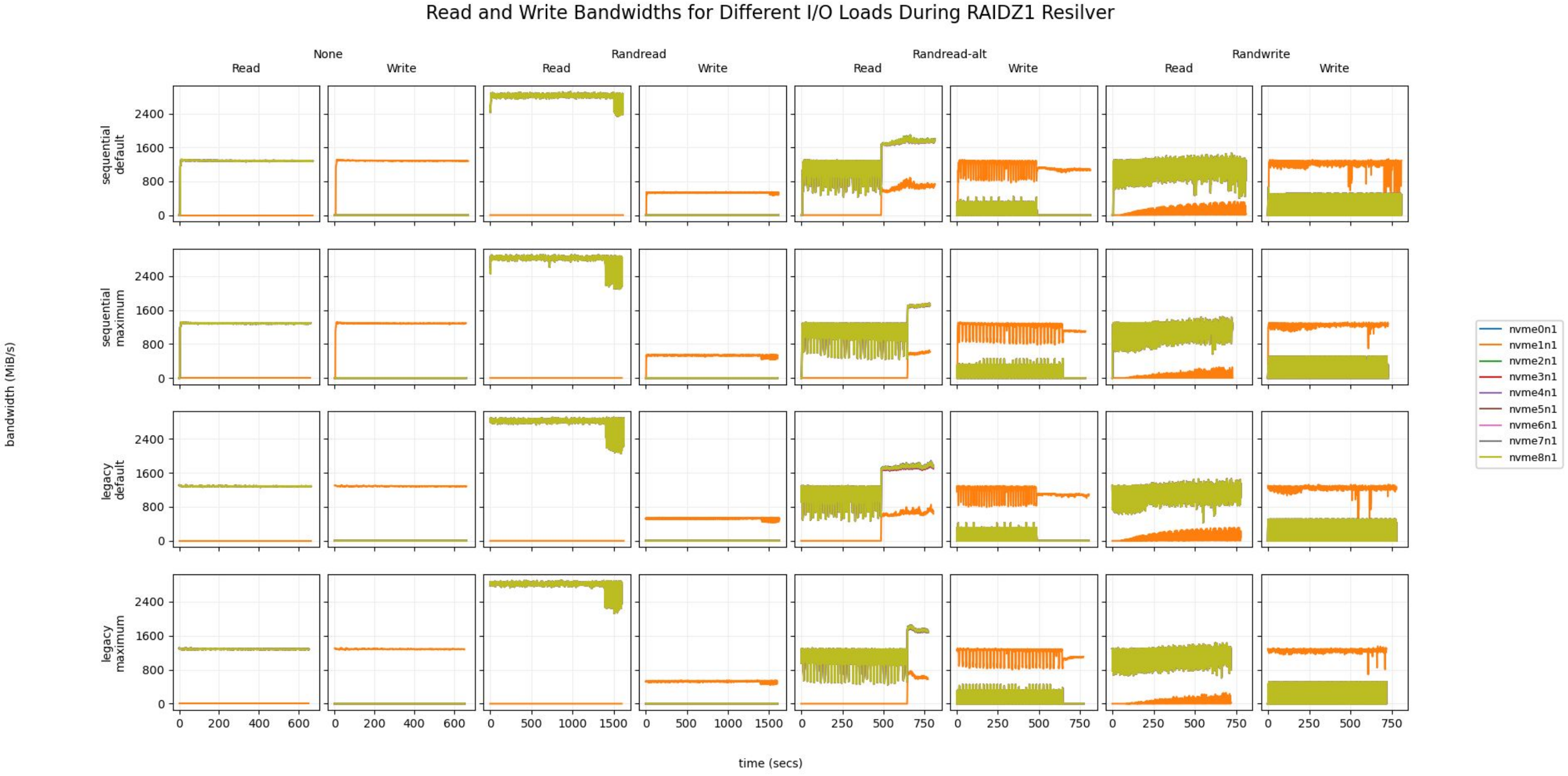
Goal: Find bottlenecks in ZFS rebuild/resilver performance!

The Setup

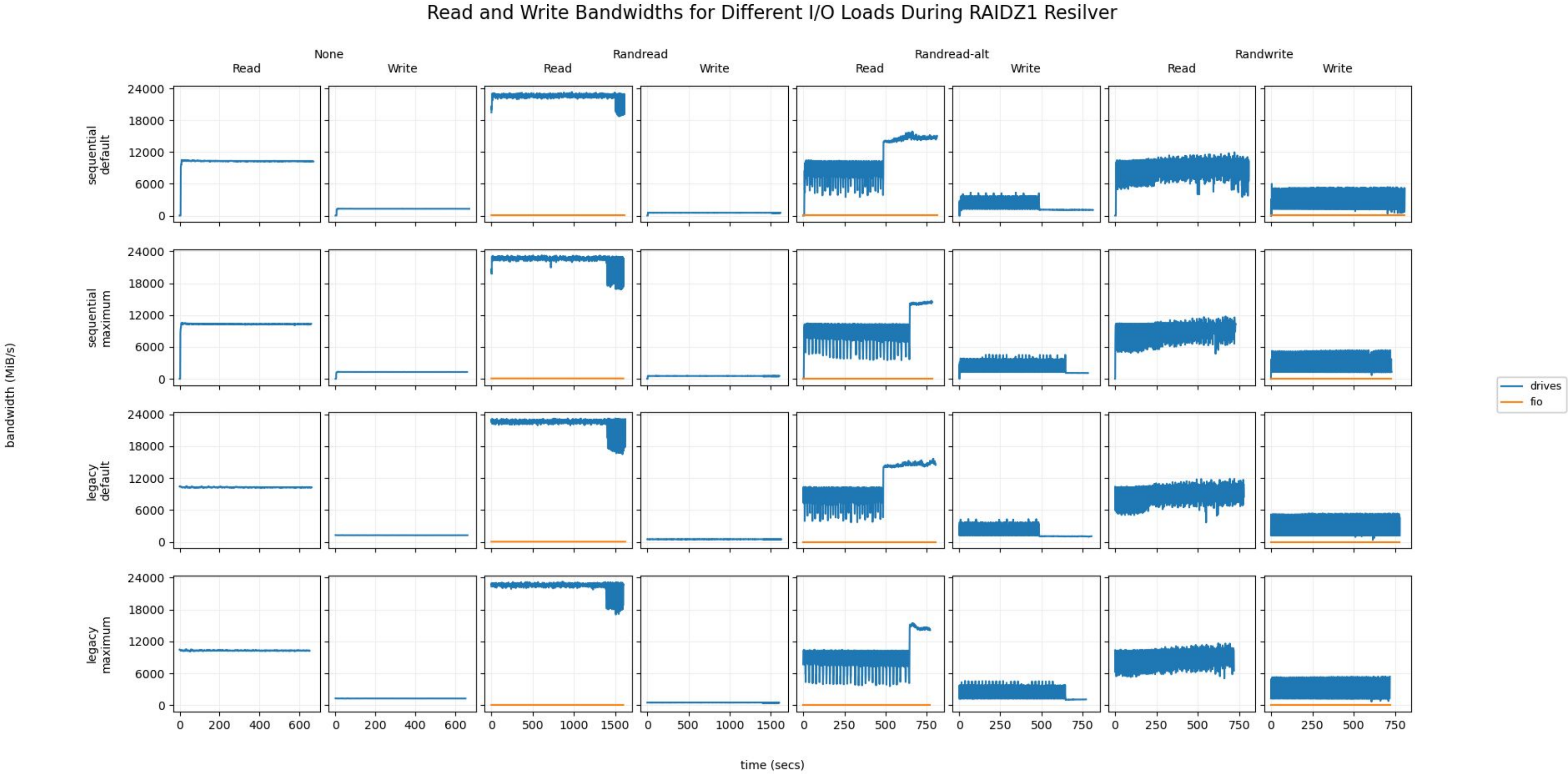
Reported Aggregate Disk Bandwidths (Sequential)



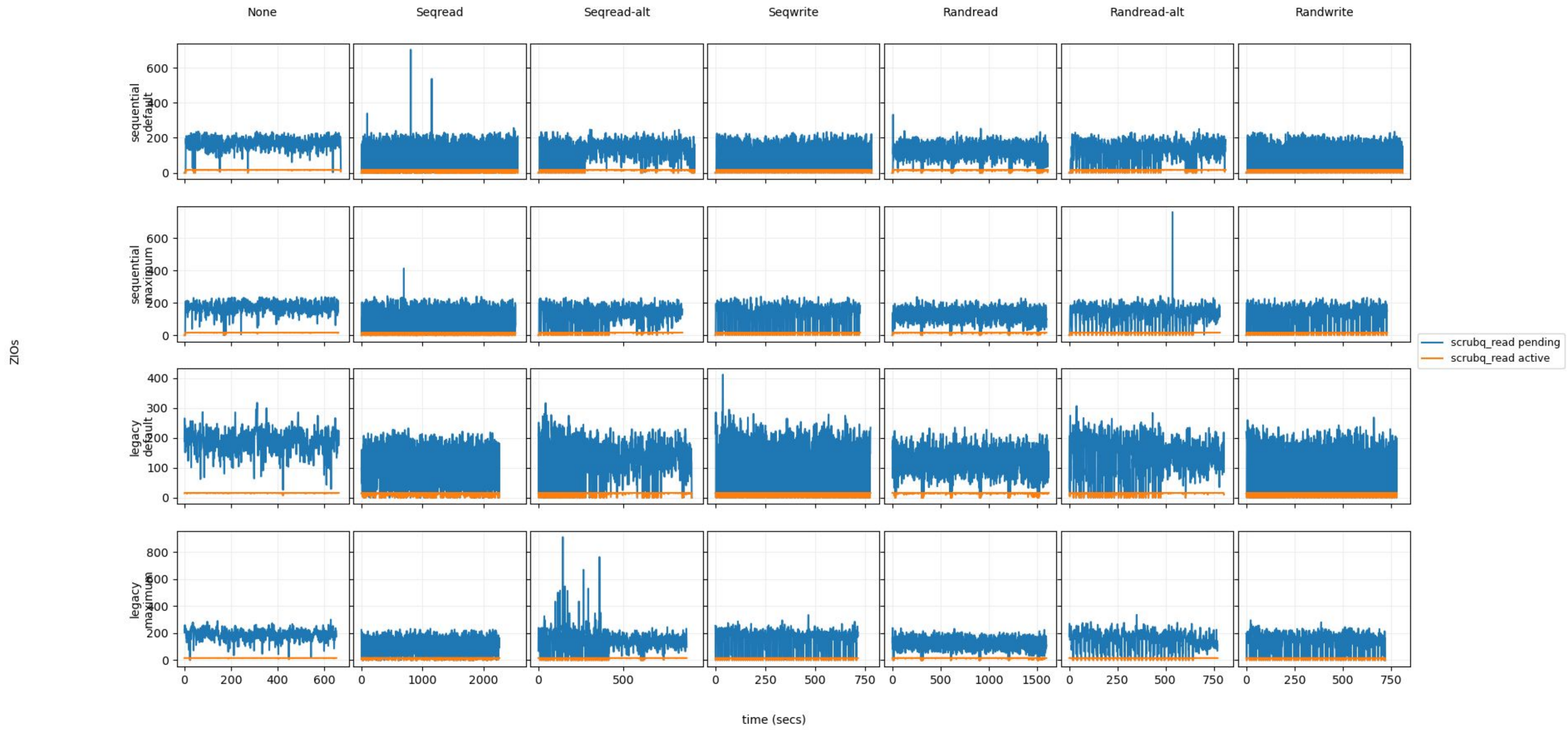
Reported Disk Bandwidths (Random)



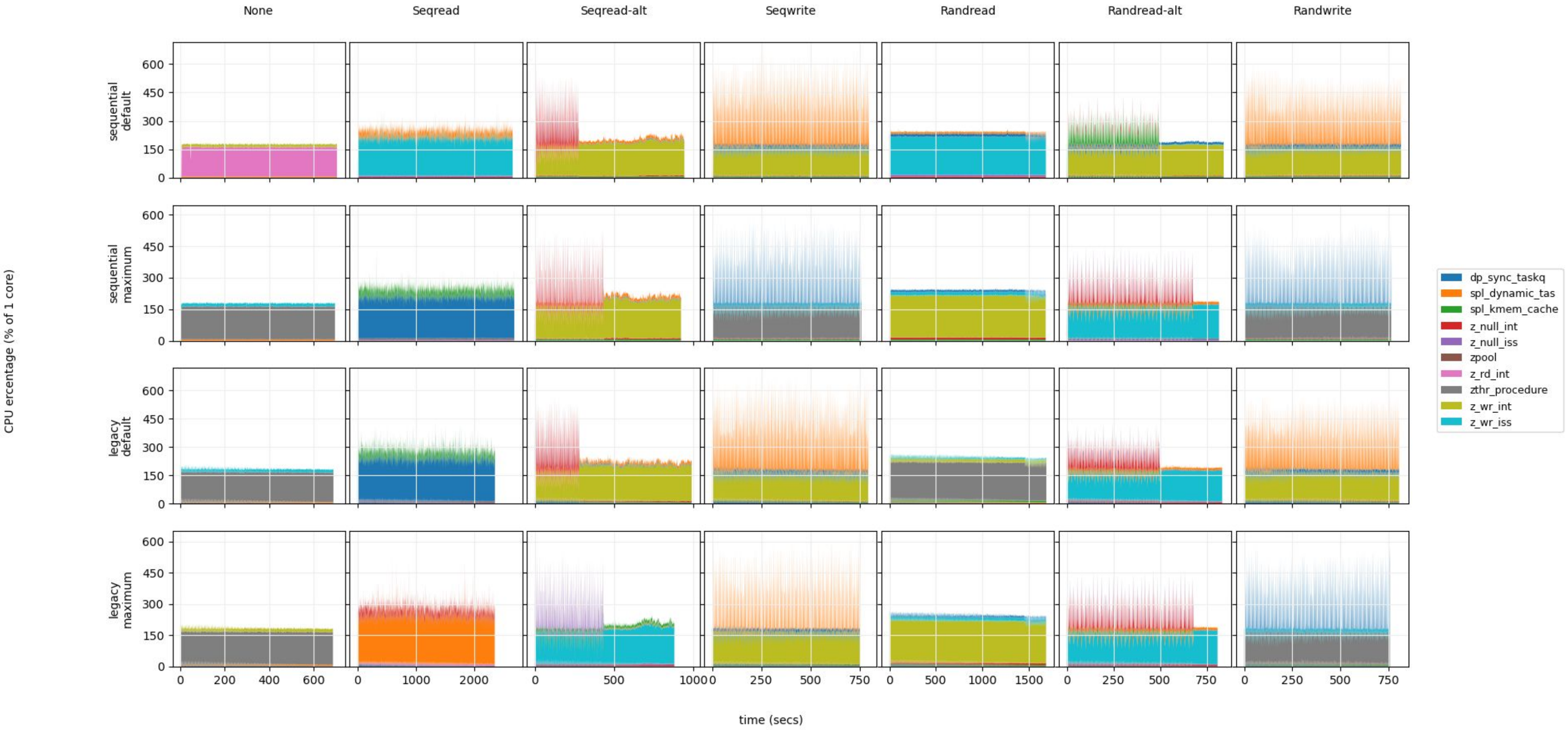
Reported Aggregate Disk Bandwidths (Random)



Scrub Queue Depths



CPU Usage by ZFS



Interpreting the Data

- Continually reading unresilvered data is a worst case.
  - Exact ZFS mechanism yet to be studied
- Even with no I/O load, the drives do not reach benchmarked values.
  - Most likely due to mixed I/O types
- Varying scrub tunables and resilver type does not vary resilver time by much.
- Baseline cases: not at NVME per drive performance
  - Still a reasonable performance target (no major bottleneck)