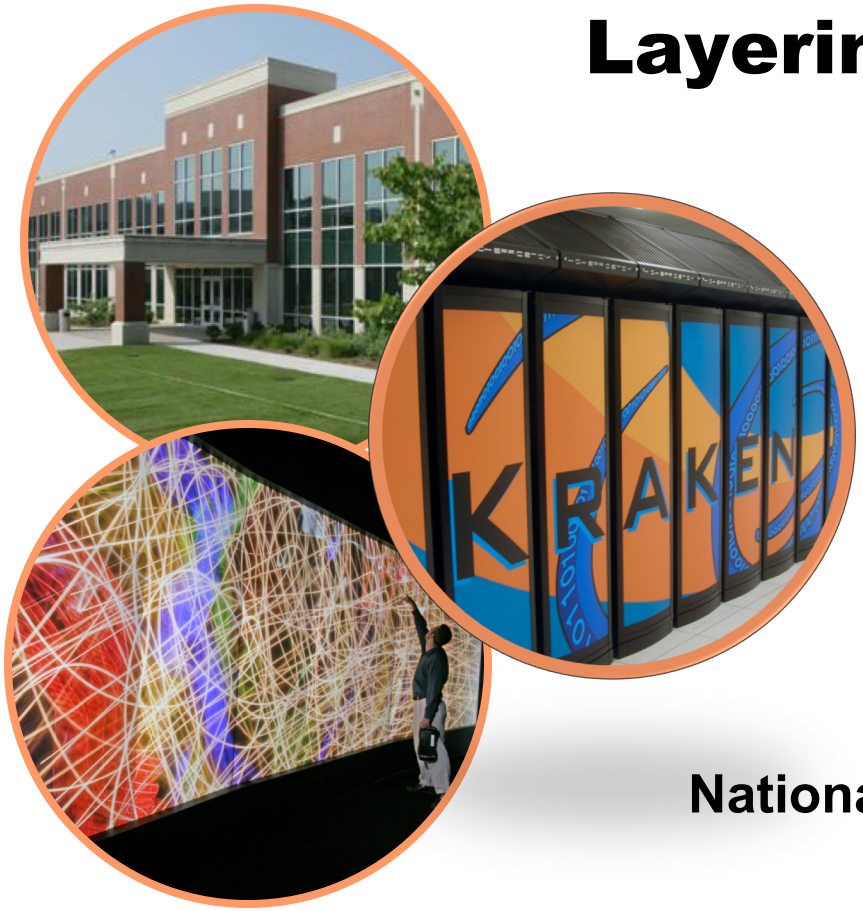


Layering ZFS Pools on Lustre

Rick Mohr

Adam Howard

National Institute for Computational Sciences
University of Tennessee

Introduction

- Lustre is a good choice for providing high-bandwidth access to shared storage
  - Excels at large sequential reads/writes
  - Other operations (like small and/or random I/O) may not perform as well
- Sites may deploy other storage (like NFS) for home directories, software builds, etc.
  - Doesn't leverage existing investment in Lustre
- Could layering another file system (like ZFS) on top of Lustre provide some benefit?
  - Improve sub-optimal use cases(?)
  - Provide additional tools for system admins

Use Cases

- Reduction in inode usage
  - One researcher created ~500 million small files and consumed ~70% of all inodes
  - Workflow did not require parallel access across nodes
- Quota management
  - Project uses a small cluster of compute nodes in an isolated enclave
  - Lustre is the only storage available, so it serves as fast scratch space and home directory space
  - Would like to restrict home directory usage while allowing unlimited use of scratch space

Use Cases

- Backups
  - If home directories are on Lustre, it would be nice to backup files without walking the directory tree (i.e. - snapshots)
  - And even nicer if there was a convenient way to send the backups to a remote host (i.e. – zfs send/receive)
- Encryption
  - Provide a place for encrypted files without needing to encrypt the entire file system
- I/O conditioning
  - Make “bad” I/O patterns more palatable to Lustre

ZFS on Lustre

- Why ZFS?
  - Can use files as VDEVs
  - Has lots of nice admin tools
  - Easily expands as needed
- Is it useful?
  - Run some tests to gauge performance
  - Gain understanding of benefits and limitations of this approach
- Keep in mind that the goal is to supplement Lustre, not replace it

File System Configuration

- DDN SFA7700 w/ single SS8642 expansion enclosure
  - 80x 10TB 7.2K RPM drives configured as 8 pools (RAID6 8+2) for OSTs
  - 4x 1TB 10K RPM drives in RAID10 config for MDT
- Two VMs (CentOS 7.5) on SFA7700 controller
  - First VM mounts MDT and 4 OSTs
  - Second VM mounts 4 OSTs
  - FDR Infiniband for both
- Client system (CentOS 7.4) w/ EDR Infiniband
- Lustre 2.10.3 and ZFS 0.7.11

ZFS Configuration

- Used two different ZFS configurations
  1. Single Lustre file with stripe_count = 8
  2. Eight Lustre files each with stripe_count = 1 (chosen on different OSTs)
- Will refer to these configurations as ZFS(1v8s) and ZFS(8v1s) respectively
- A partition on the client system's internal drive was available for use as a ZIL
  - Any configuration using a ZIL will have "+ZIL" appended to name

Test #1: Code Compilation

- Test code compilation with two benchmarks
  1. kcbench – Compiles Linux kernel
  2. LAPACK (v3.8.0) build – Measure time needed to run “make lib”
- Ran benchmarks on:
  - Lustre
  - Two ZFS configurations without ZIL
  - Two ZFS configurations with ZIL
  - XFS on local drive
- Each test was run three times

Code Compilation Times

	kcbench	LAPACK
XFS	24.95	112.16
Lustre	170.90	122.09
ZFS(1v8s)	28.08	112.25
ZFS(1v8s)+ZIL	28.21	112.40
ZFS(8v1s)	27.41	112.76
ZFS(8v1s)+ZIL	27.36	113.50

Average Compilation Times (in secs)

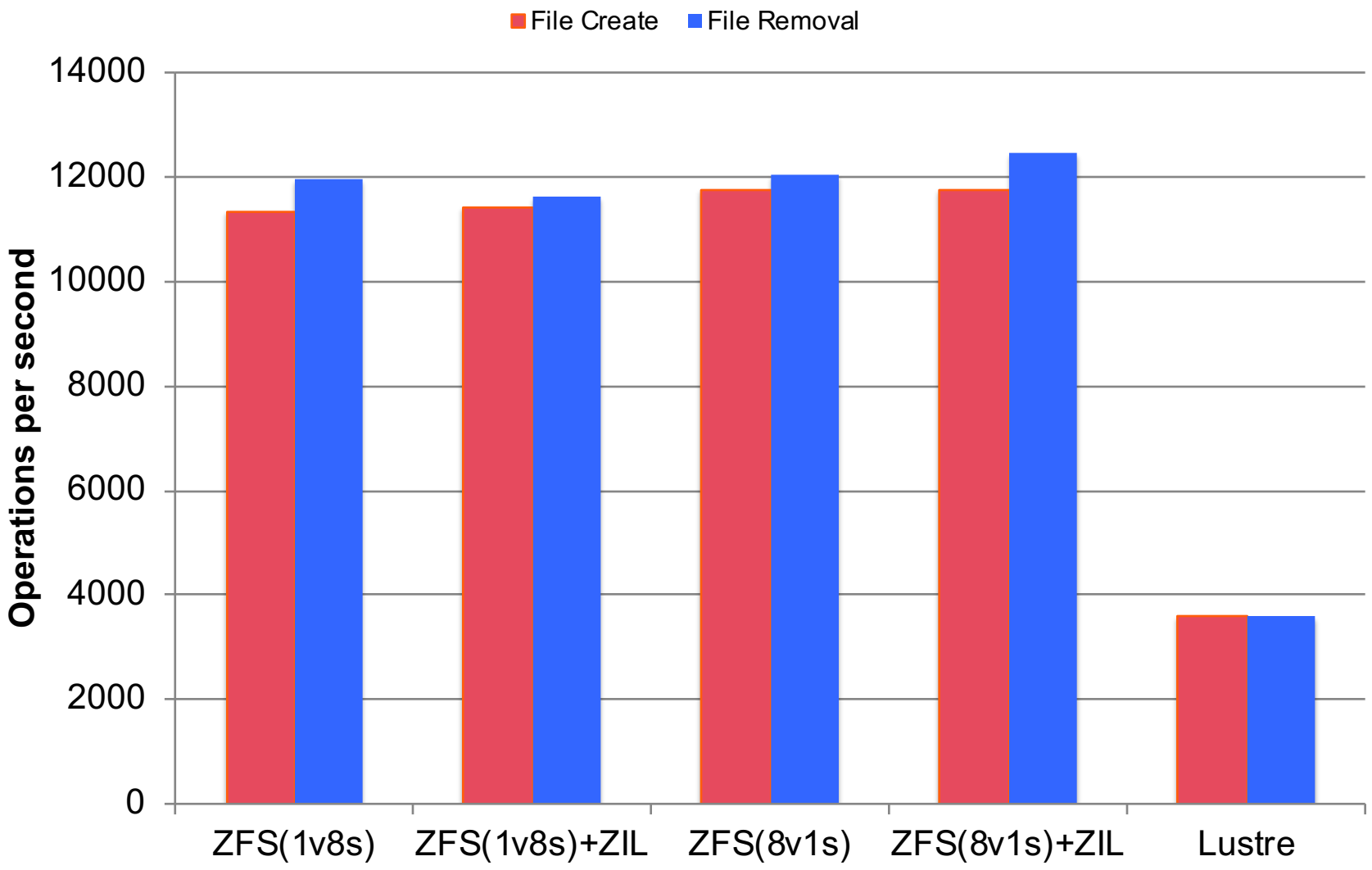
Test #2: Metadata Rates

- Use `mdtest` to measure file create/remove/stat operations per second
- Create about 340,000 files

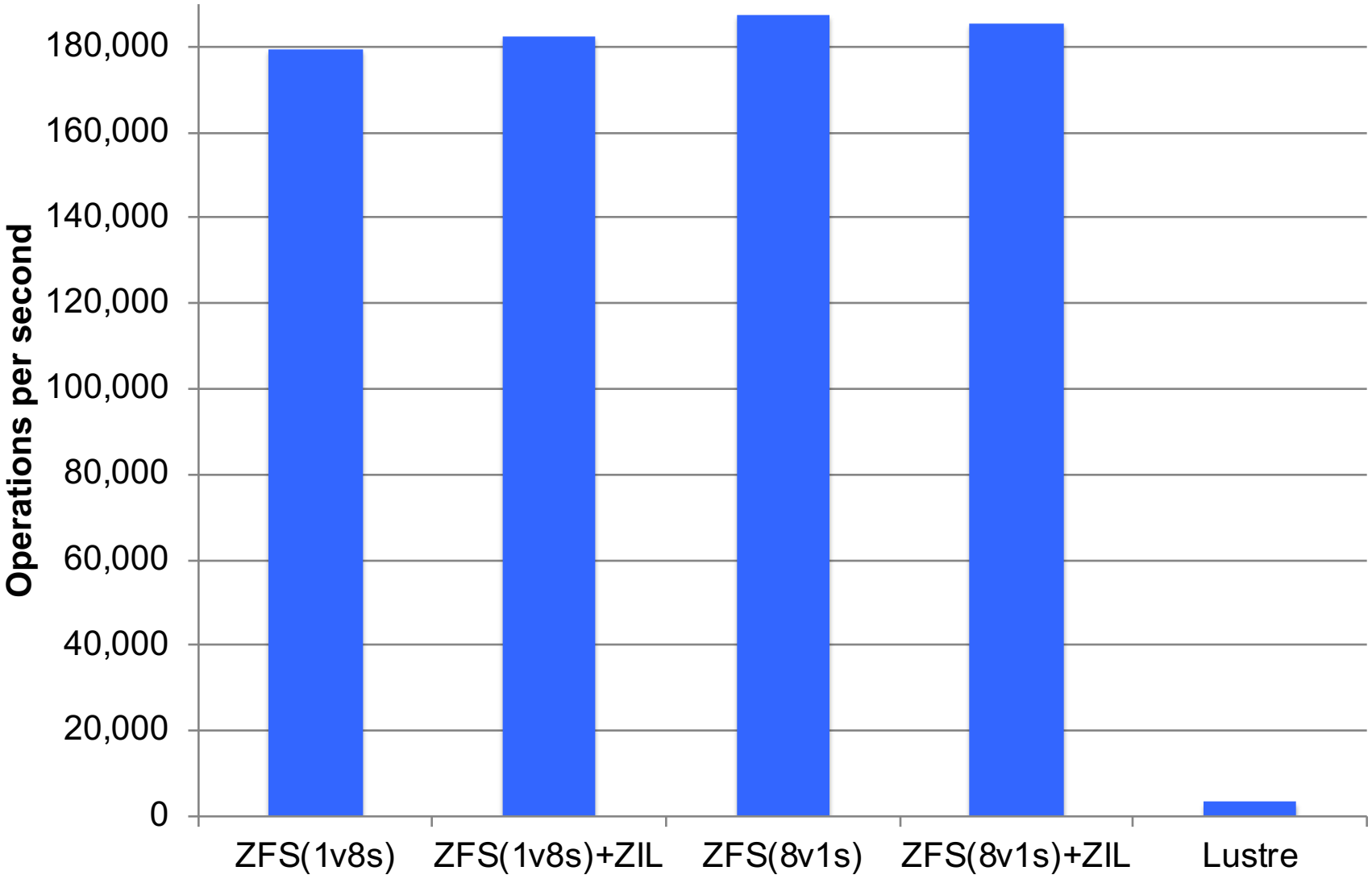
```
mdtest -F -b 4 -z 4 -u -I 1000 \
    -i 3 -d <target_dir>
```

- Ran test against four ZFS configurations and Lustre (3 times each)

mdtest: File Create/Remove



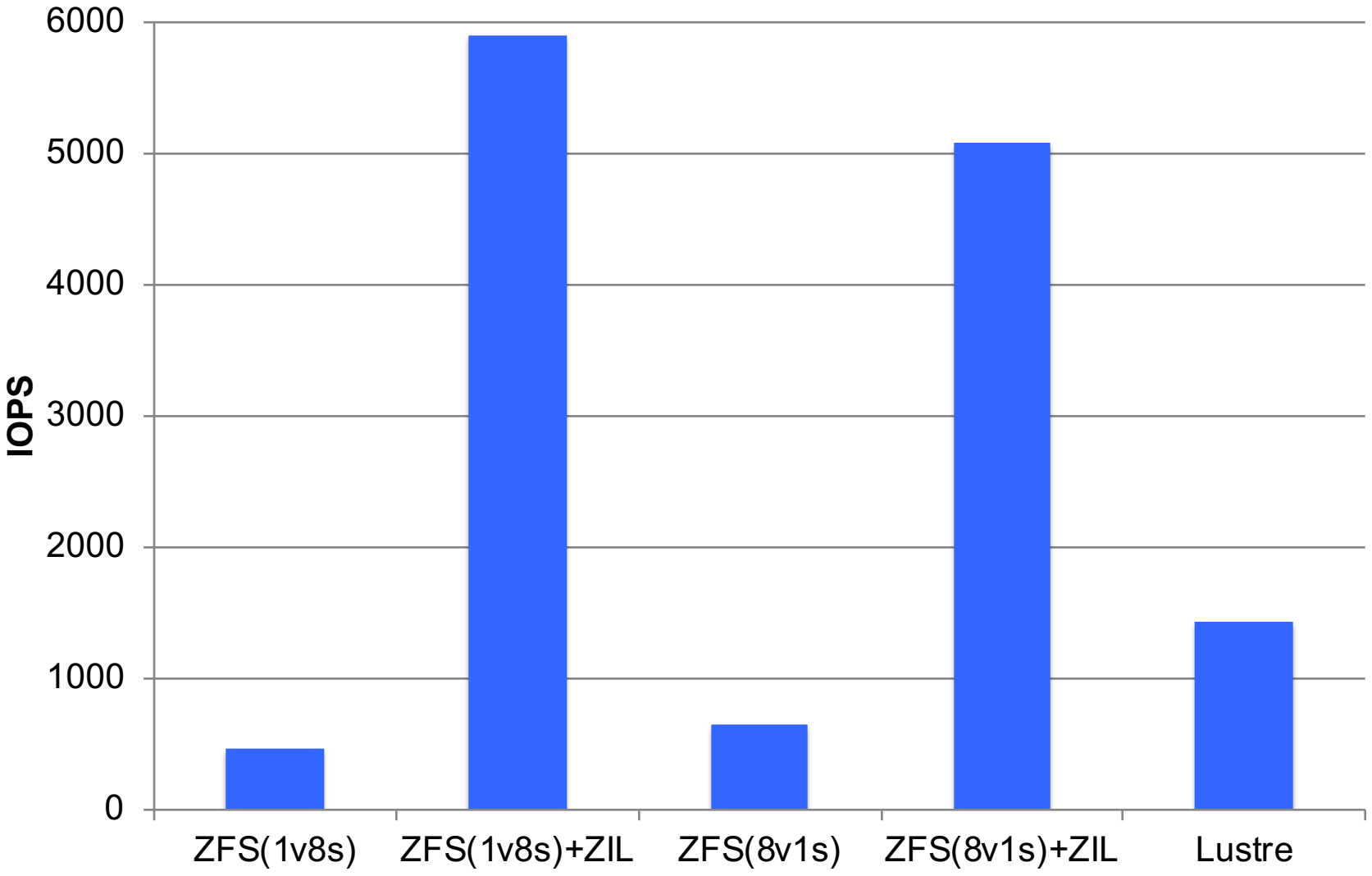
mdtest: File Stat



Test #3: Small Random Sync Write

- Use `fio` to generate random 4K synchronous write requests
- Options for `fio`:
  - bs = 4K
  - runtime = 60
  - size = 1G
  - rw = randwrite
  - iodepth = 1
  - sync = 1
- Test all ZFS configurations and Lustre (3 times each)

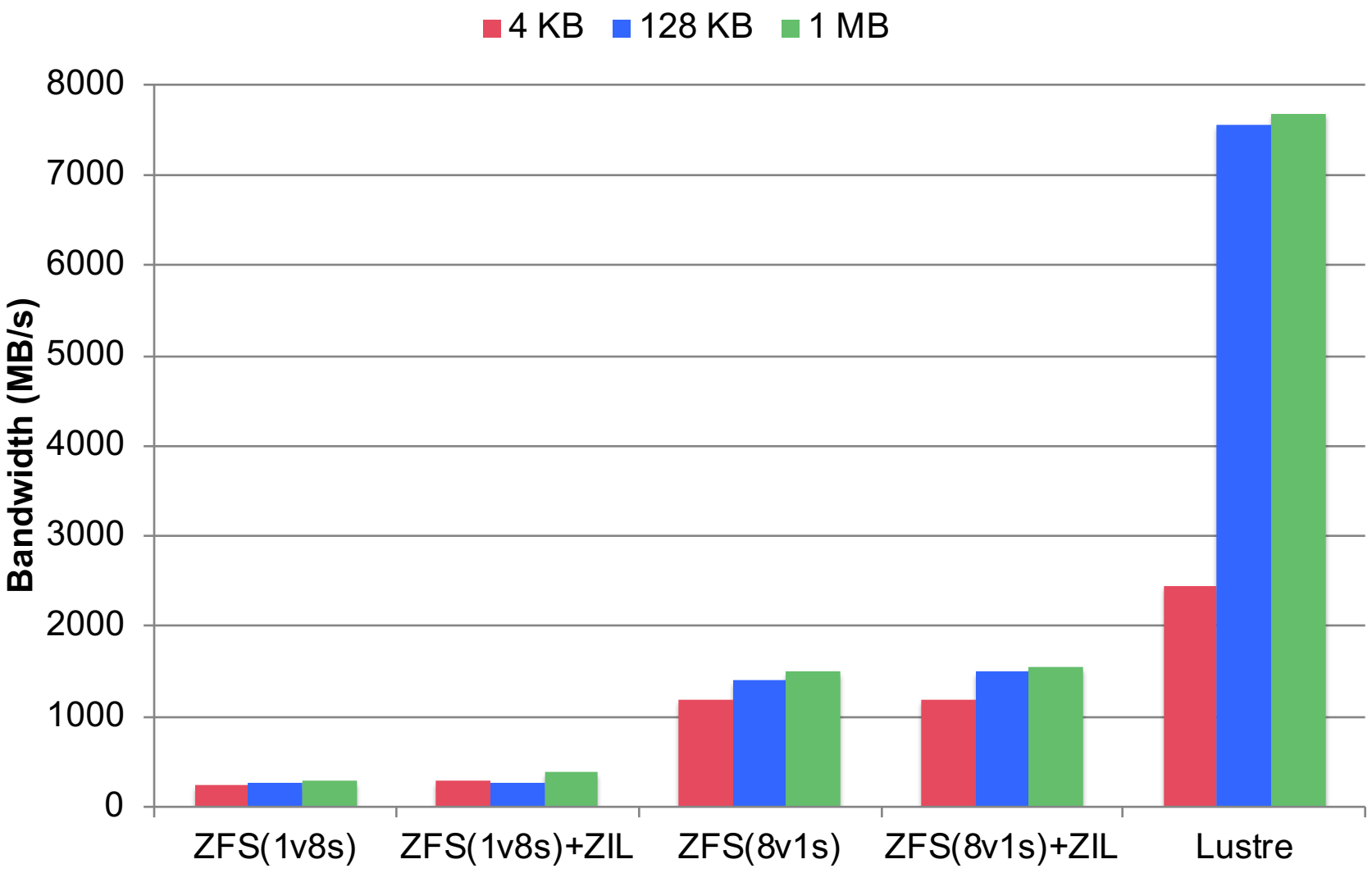
IOPS (4KB Sync Random Write)



Test #4: Sequential Write

- Use `fio` to generate sequential write operations for various block sizes (4KB, 128KB, 1MB)
- Options for `fio`:
  - bs = 4K
  - runtime = 60
  - size = 20G
  - rw = write
  - fallocate = none
- Test all ZFS configurations and Lustre (3 times each)

Sequential Write Bandwidth



NFS Export

- **Using ZFS-on-Lustre for home directories requires making it available via NFS**
  - **Need to also consider NFS client performance**
- **Some initial results:**

	kcbench	LAPACK	Create	Remove	Stat
Lustre	170 s	122 s	3,577 ops	3,607 ops	3,692 ops
ZFS(8v1s)	27 s	113 s	11,748 ops	12,055 ops	187,212 ops
ZFS/NFS	152 s	128 s	584 ops	694 ops	17,985 ops

- **More testing needed**

Conclusions

- ZFS-on-Lustre looks like it may be a viable option for certain workloads
  - Seems to compliment Lustre in some areas
  - Could be particularly useful for home directories
  - Opens up possibilities for alternate use cases
- Number of VDEVs and their stripe count influences performance on some tests
  - More testing needed to identify optimal configuration
- May allow sites to leverage Lustre for other storage needs

Future Work

- Investigate ZFS tuning parameters
- Optimal Lustre striping for VDEVs
- Lustre tuning parameters
- SSD instead of HDD for ZIL
- Analysis of I/O patterns on the Lustre servers
- Other benchmarks

Questions?