ZFS vs hardware RAID

Daniel Traynor, QMUL, HEPsysman 2017
d.traynor@qmul.ac.uk

Aim

- Deploy ZFS on to storage hardware and evaluate performance. Compare to hardware raid with the same hardware. Use a modern system.

- For cheeper Lustre HA setup need to avoid using dedicated ZIL (ZFS intent log) for writes and L2ARC cache for reads.

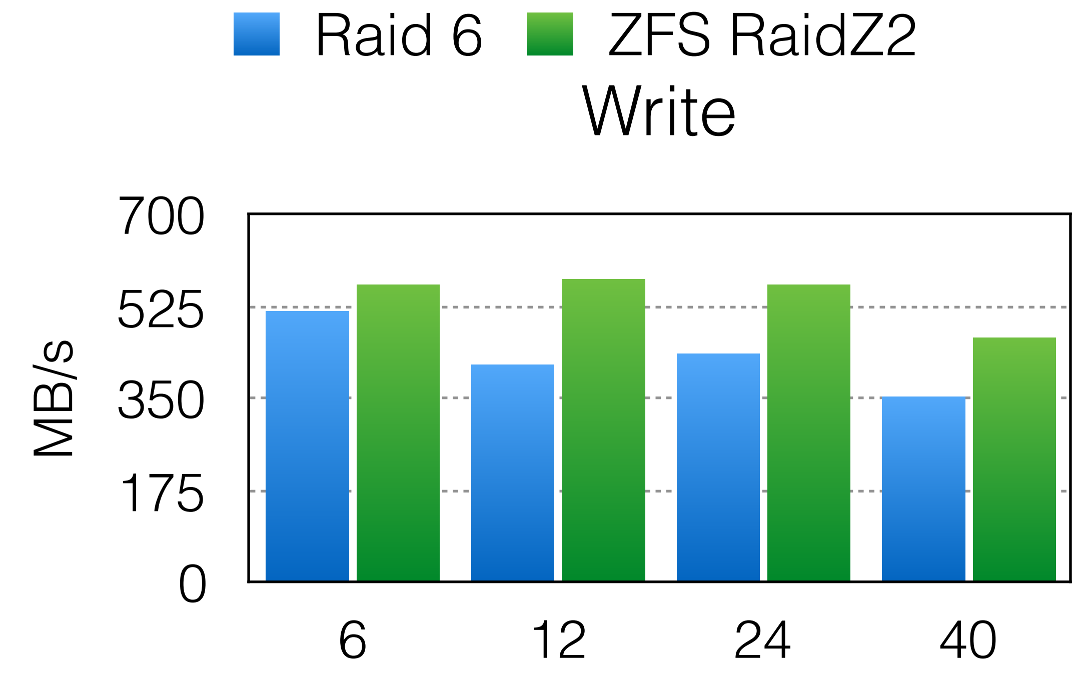
Tests setup

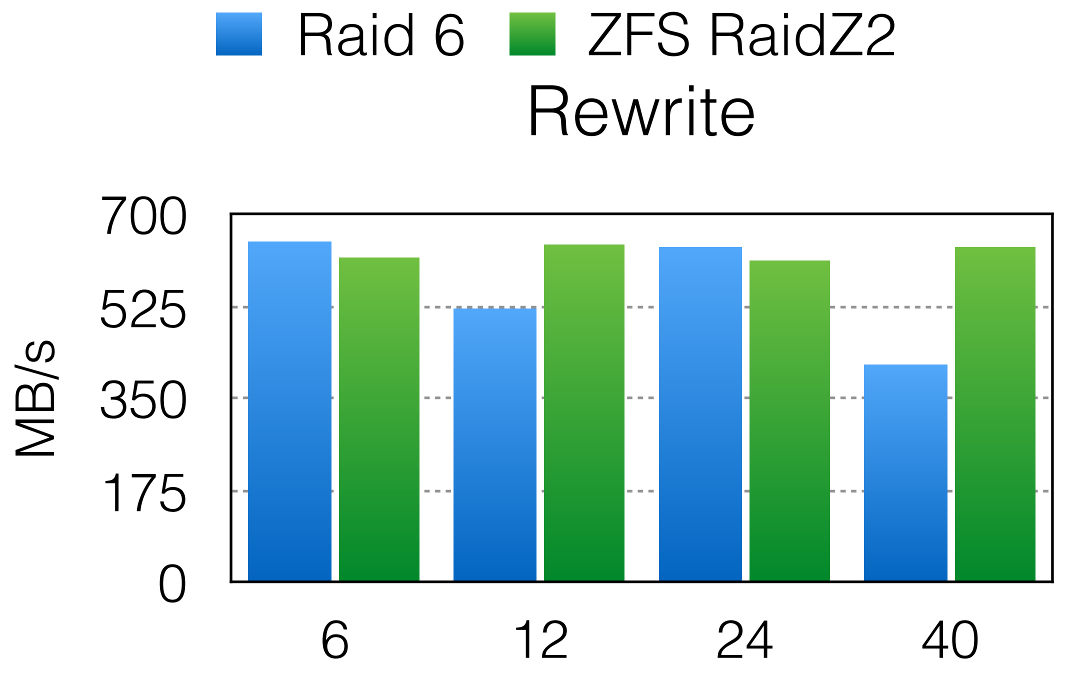
- Use “spare” HPE APPLO 4200.
- RAID card can be run as a HBA card.
- Compare 12 disk raid 6 with 12 disk raidZ2.
- 8TB disks, 128 GB RAM (2 or 16 GB used for performance tests), 2* E5-2609 V3 CPU (12 cores @ 1.9GHz). Raid card has 2GB cache in Raid mode but not HBA mode.
- SL6.7 with ZFS version 0.6.5.9.

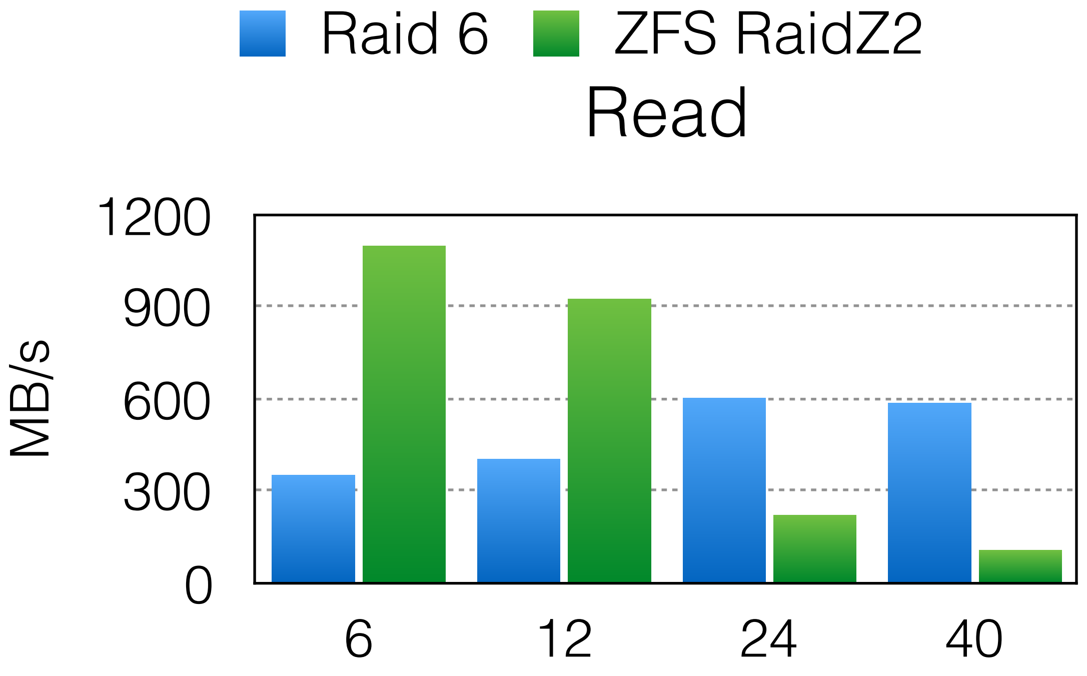
Short tests

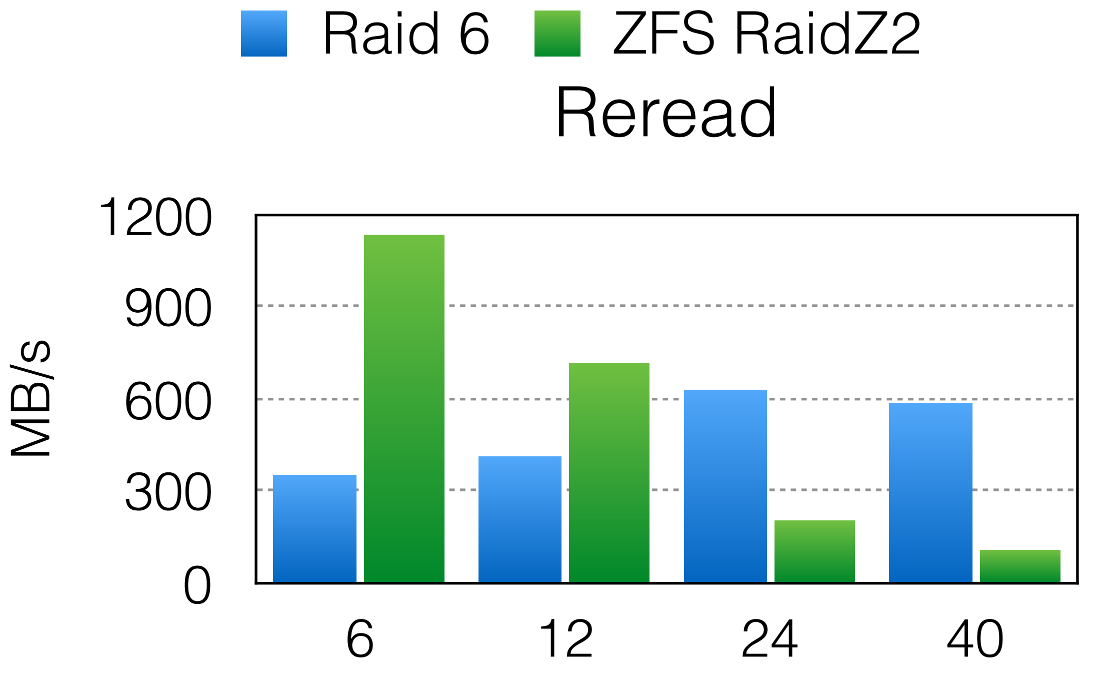
- Run “short” iozone test (streaming performance) with different thread count but same total data throughput (80GB) limit RAM to 2GB.
- Run “short” tests but with 16GB RAM.
- e.g. iozone -t 40 -r 1024k -s 2g -i0 -i1

Results (2G RAM)

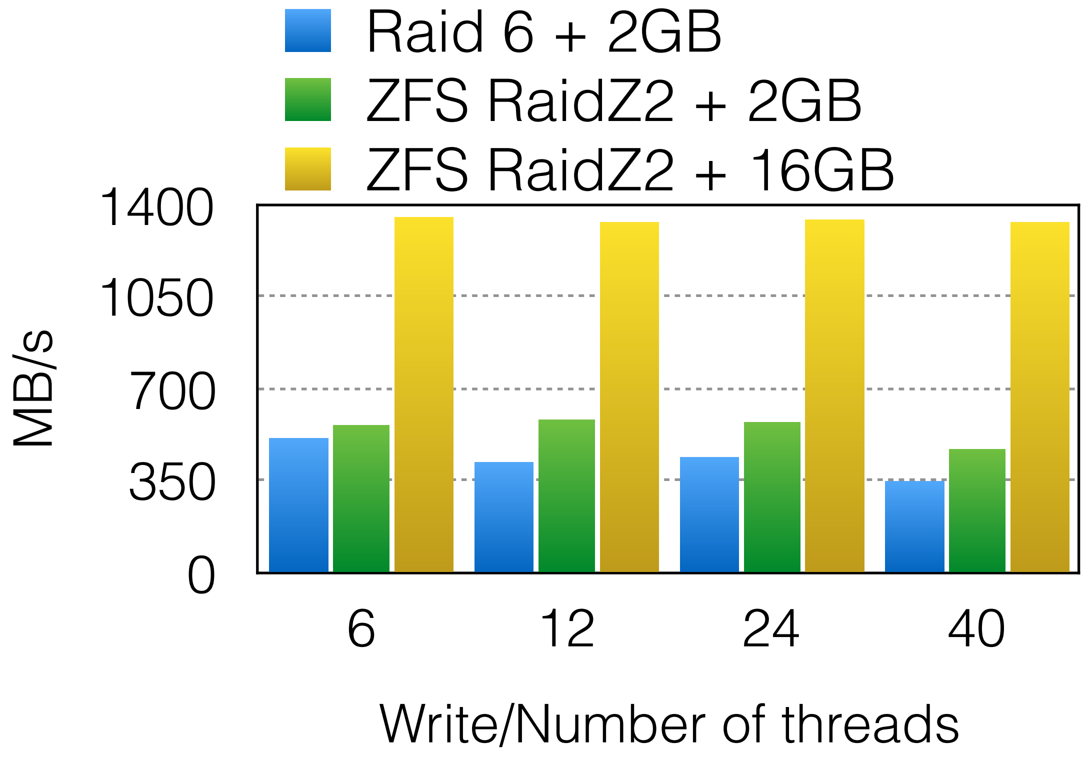


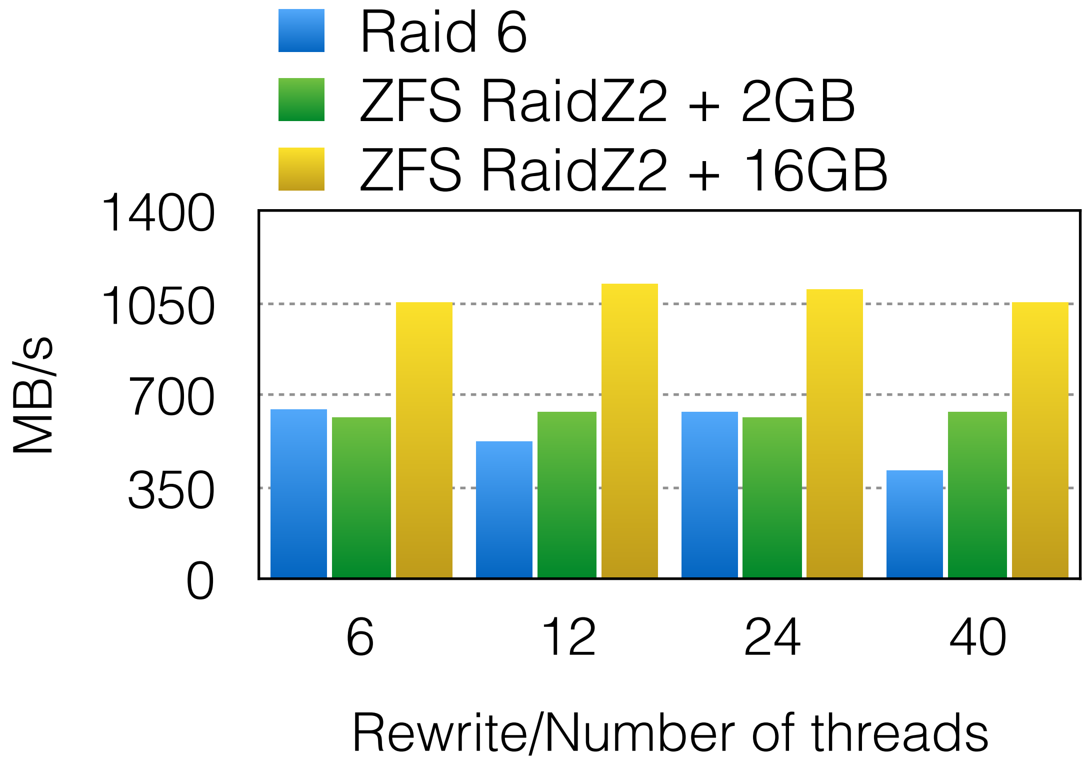


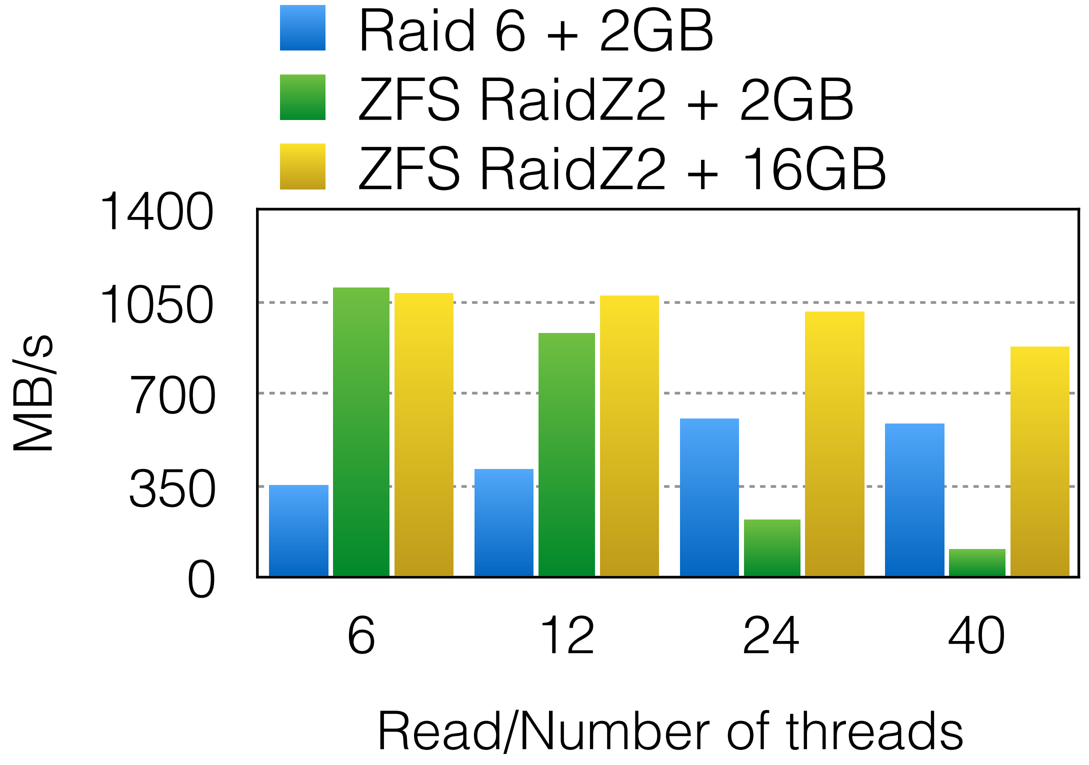


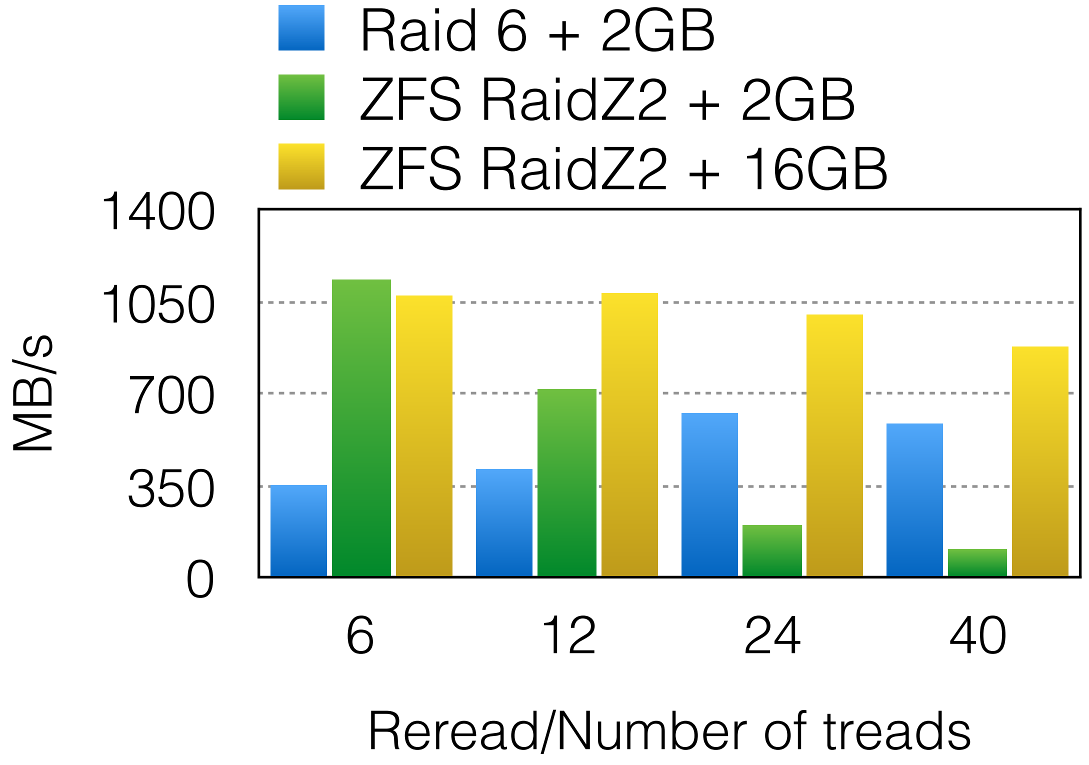


Results (16GB RAM)









Optimisation tests

- IOzone long test with 16GB RAM + different optimisations
  - RAID 6 use stranded EXT4 set taken from Lustre.
  - For ZFS
    - Subset of Lustre tunes
    - ZFS record size 64->128 k
    - ZFS IO operations per device 10 -> 12

-
  ```
  iozone -e -+u -t 12 -r 1024k -s 6.7g -i0 -i1 -i 2 -i 3 -i 5 -i 8
  ```

Optimisations

Ext4/Linux dev

```
echo deadline > /sys/block/sdb/queue/scheduler
echo 4096 > /sys/block/sdb/queue/nr_requests
echo 4096 > /sys/block/sdb/queue/read_ahead_kb
echo performance | tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor >/dev/null
```

Ext4/
Linux cache

```
echo madvise > /sys/kernel/mm/redhat_transparent_hugepage/enabled
echo madvise > /sys/kernel/mm/redhat_transparent_hugepage/defray
echo 1 > /proc/sys/vm/dirty_background_ratio
echo 75 > /proc/sys/vm/dirty_ratio
echo 262144 > /proc/sys/vm/min_free_kbytes
echo 50 > /proc/sys/vm/vfs_cache_pressure
```

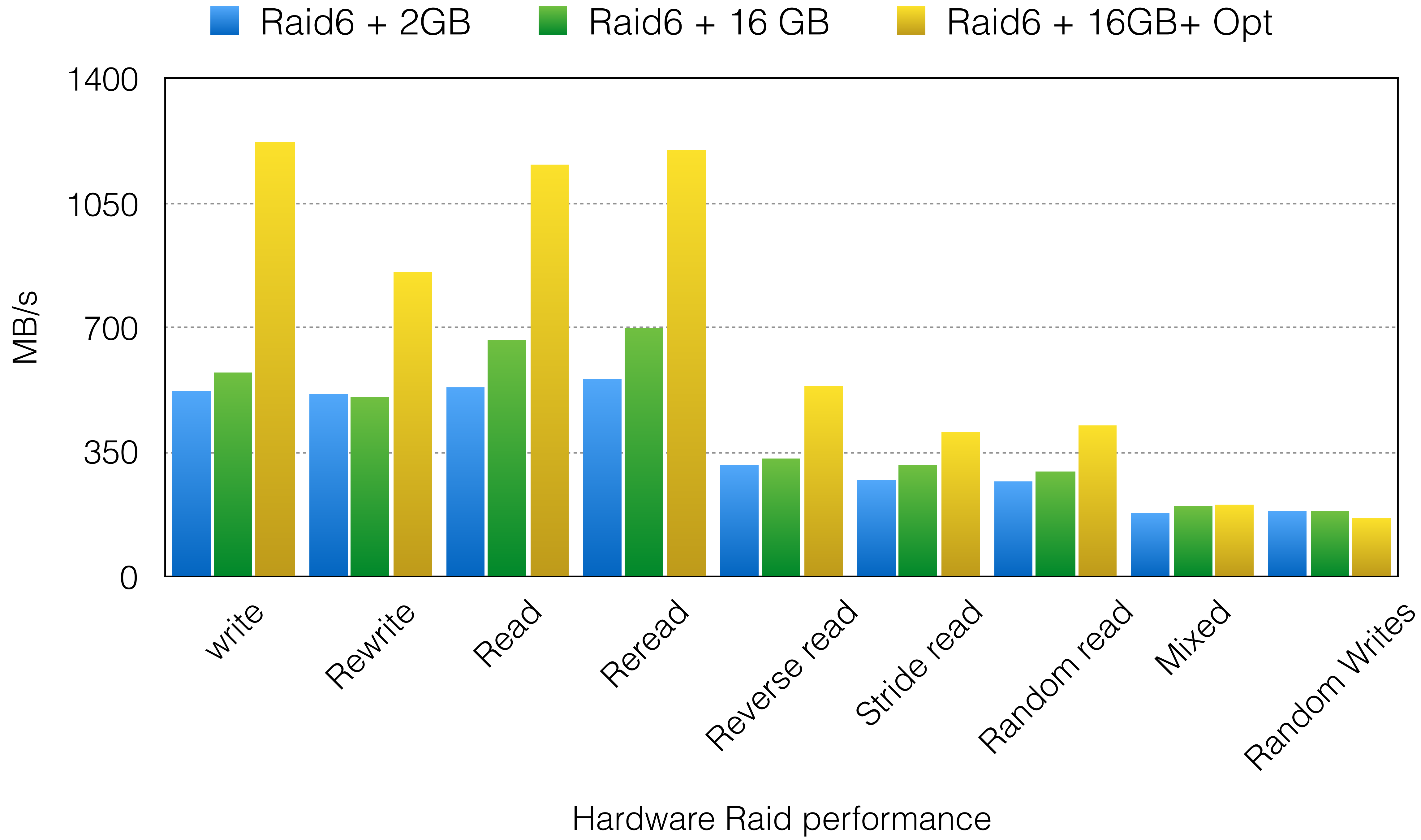
ZFS 128k
record size

```
options zfs zfs_vdev_cache_size=1310720
options zfs zfs_vdev_cache_max=131072
options zfs zfs_vdev_cache_bshift=17
options zfs zfs_read_chunk_size=1310720
```

ZFS
pending IO
per device

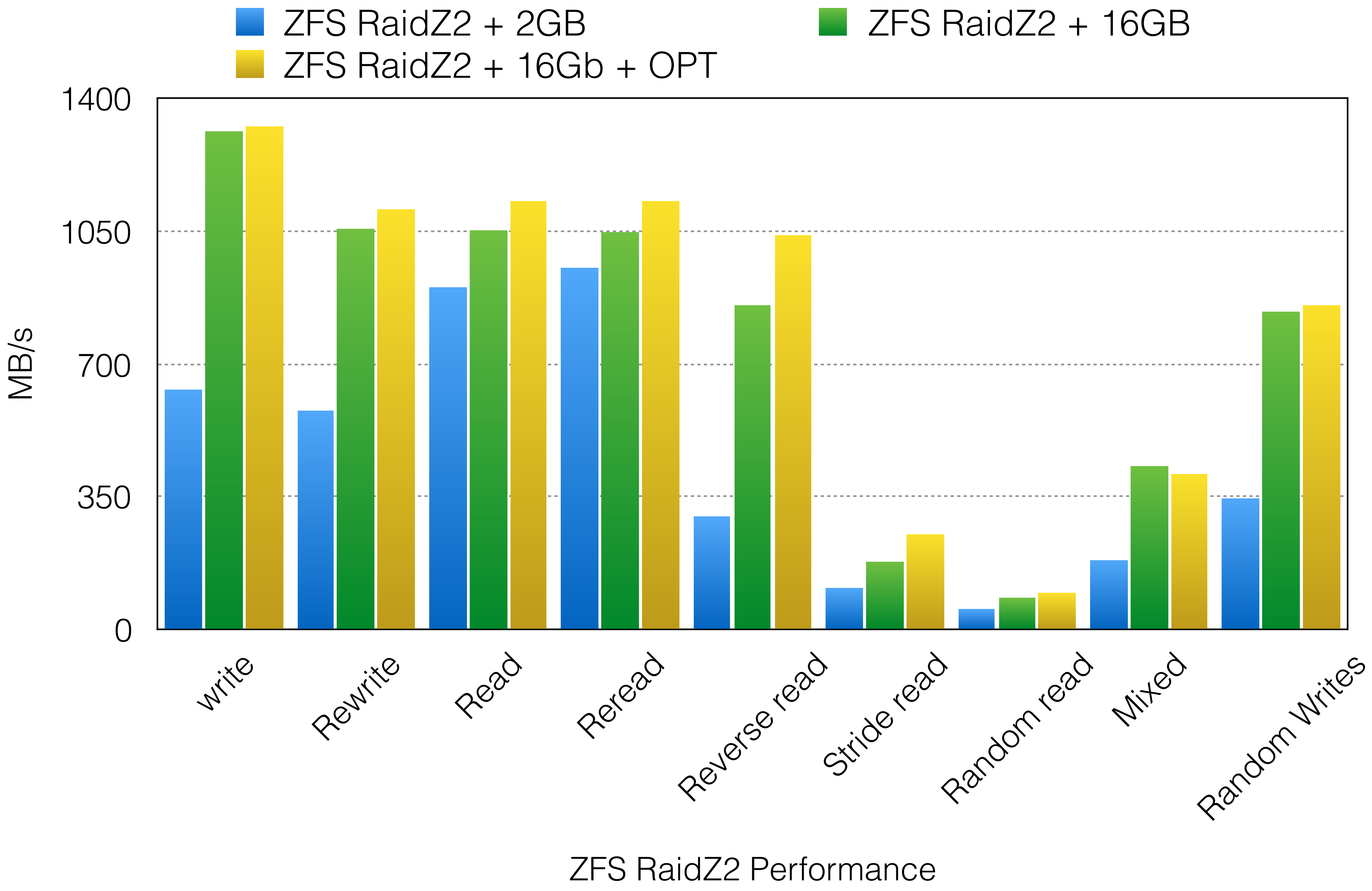
```
options zfs zfs_vdev_async_read_max_active=12
options zfs zfs_vdev_async_read_min_active=12
options zfs zfs_vdev_async_write_max_active=12
options zfs zfs_vdev_async_write_min_active=12
options zfs zfs_vdev_sync_read_max_active=12
options zfs zfs_vdev_sync_read_min_active=12
options zfs zfs_vdev_sync_write_max_active=12
options zfs zfs_vdev_sync_write_min_active=12
```

Optimisation results EXT4



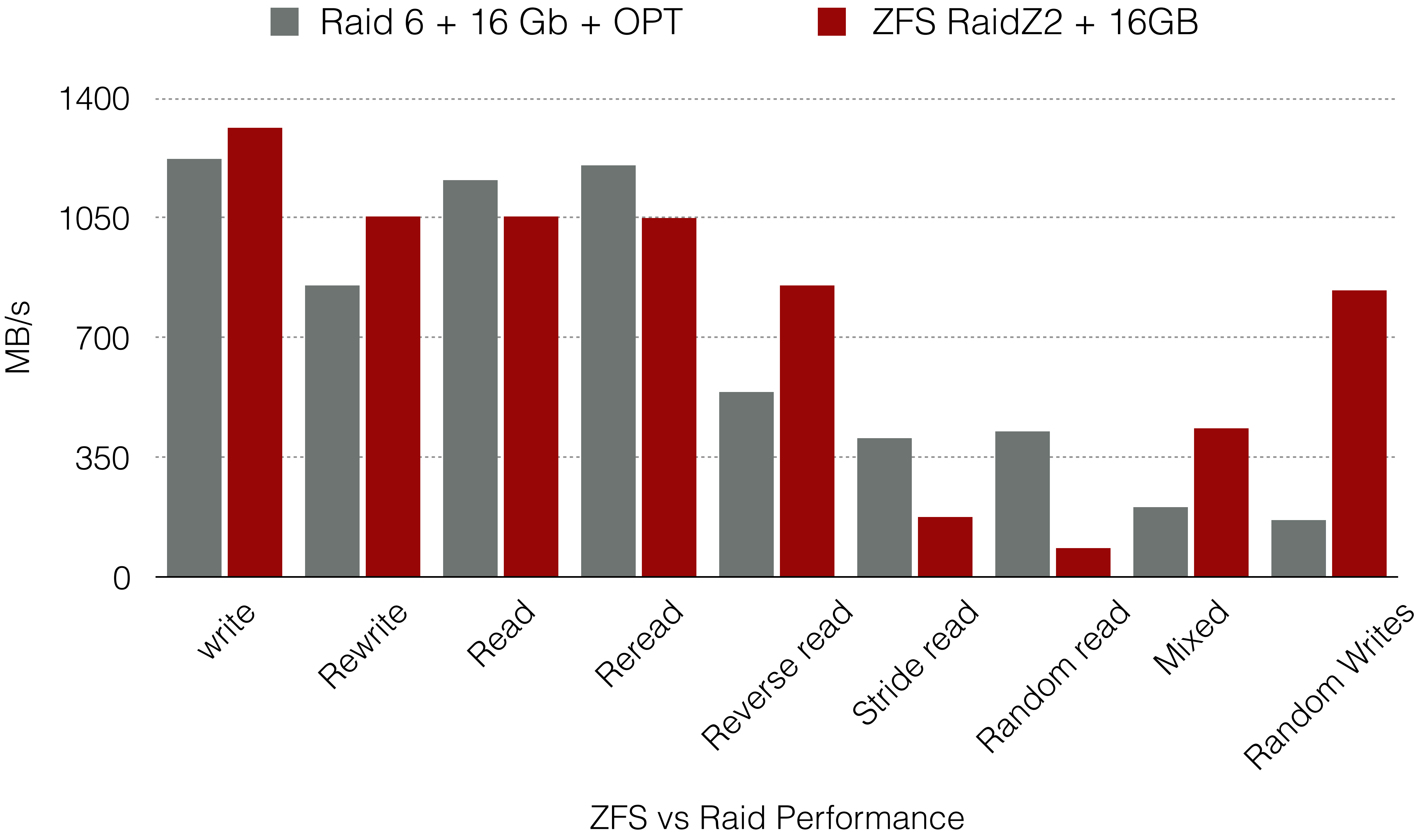
As expected significant improvement in Read and sequential write performance with Linux Optimisations

Optimisation results ZFS



Significant improvement with “small” amount of RAM.
Little improvement with ZFS optimisations

ZFS vs RAID



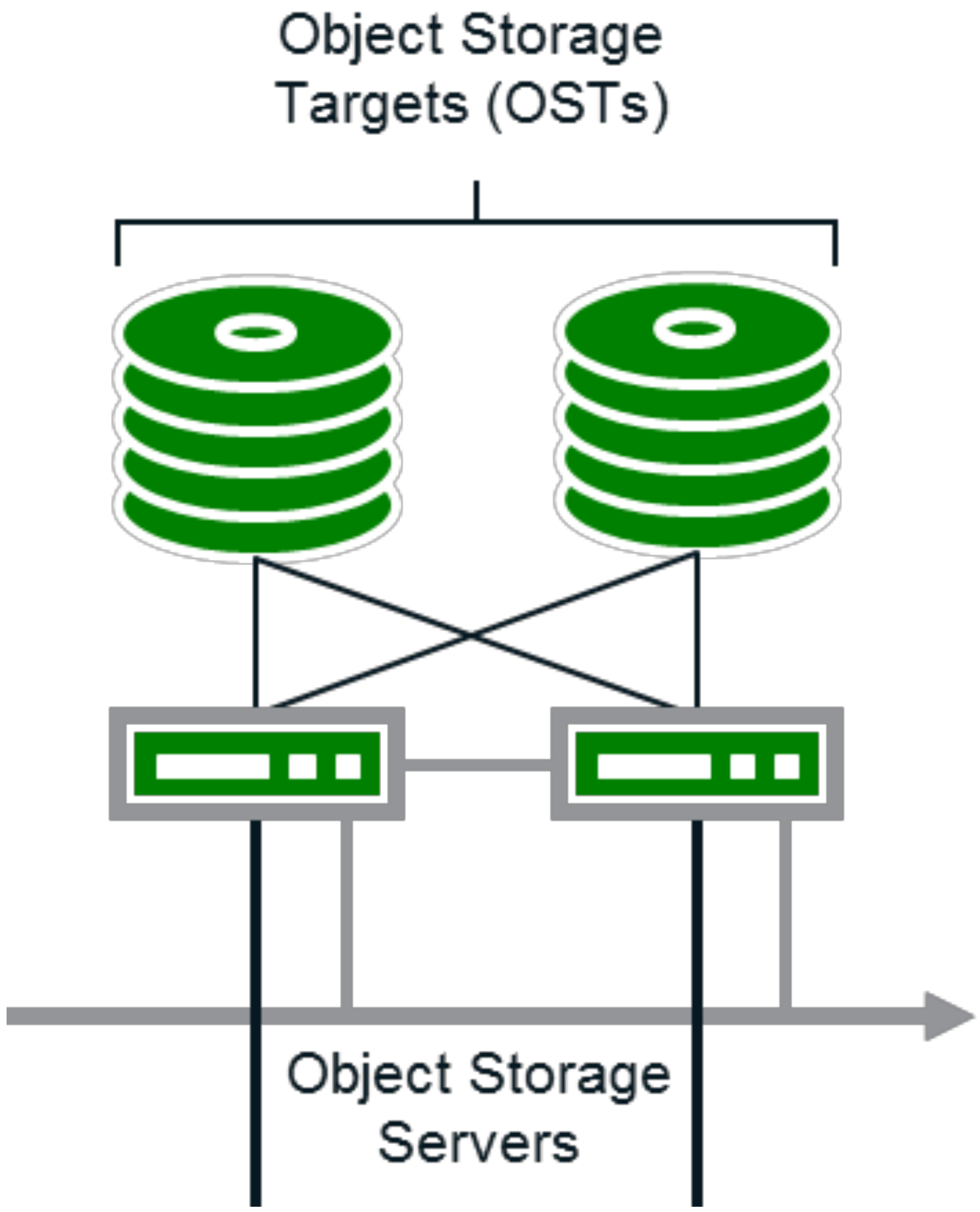
ZFS and RAID performance equal for sequential workloads
RAID better for random reads, ZFS better for random writes

Conclusions

- We were able to show that performance of ZFS (without dedicated ZIL or L2ARC cache) was able to match that of EXT4 + hardware RAID, when optimisation were applied to EXT4 and ZFS had a reasonable amount of RAM available (>>2GB). However these are simmilar to production conditions.

- Feel confident to use ZFS for our next storage purchase without dedicated caches. This will allow cost effective HA setup for Lustre.

Expected ZFS+Lustre setup



Don’t want ZIL on OSS due to failover requirements