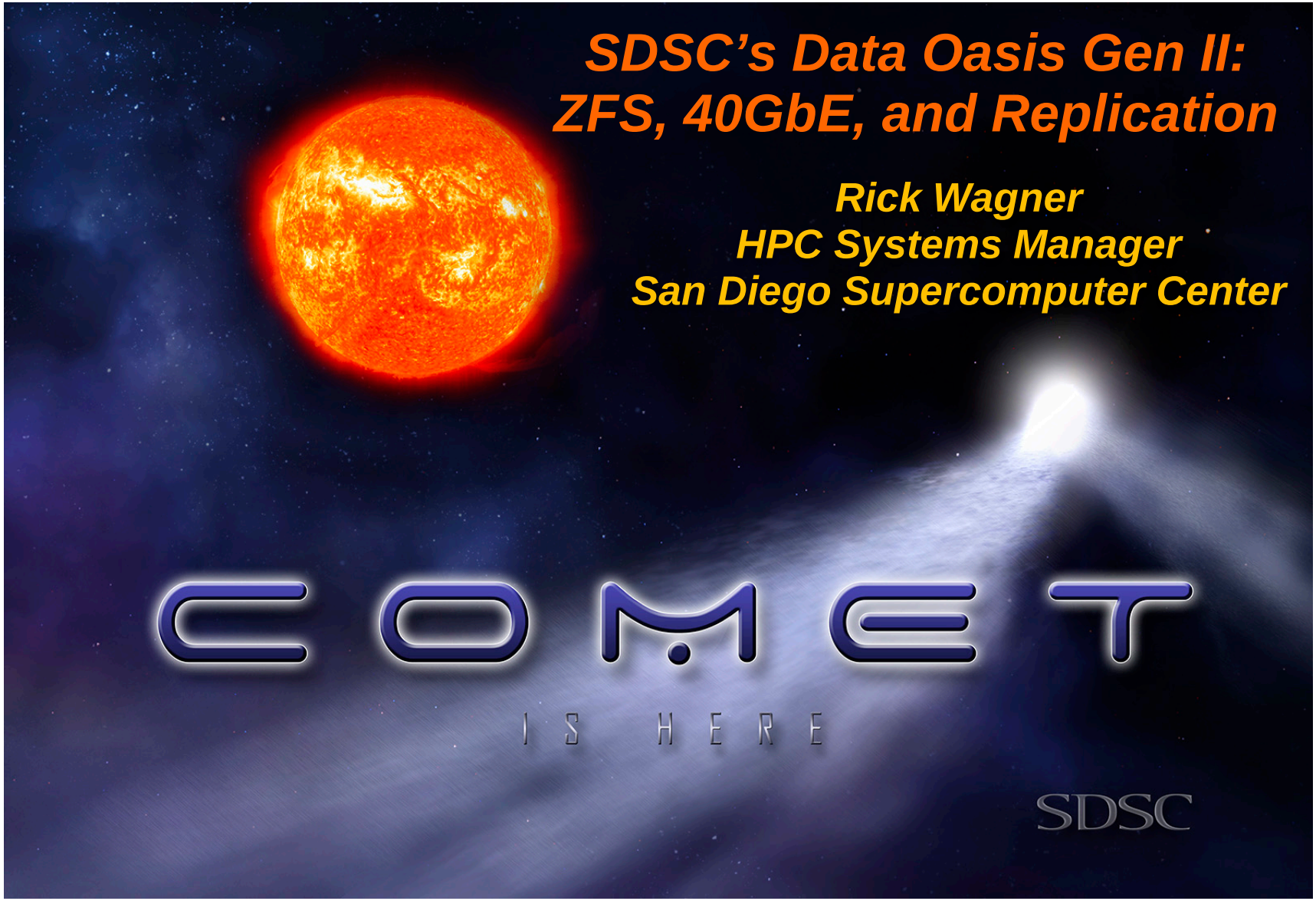
SDSC's Data Oasis Gen II: ZFS, 40GbE, and Replication

Rick Wagner
HPC Systems Manager
San Diego Supercomputer Center



Comet
"HPC for the long tail of science"

iPhone panorama photograph of 1 of 2 server rows

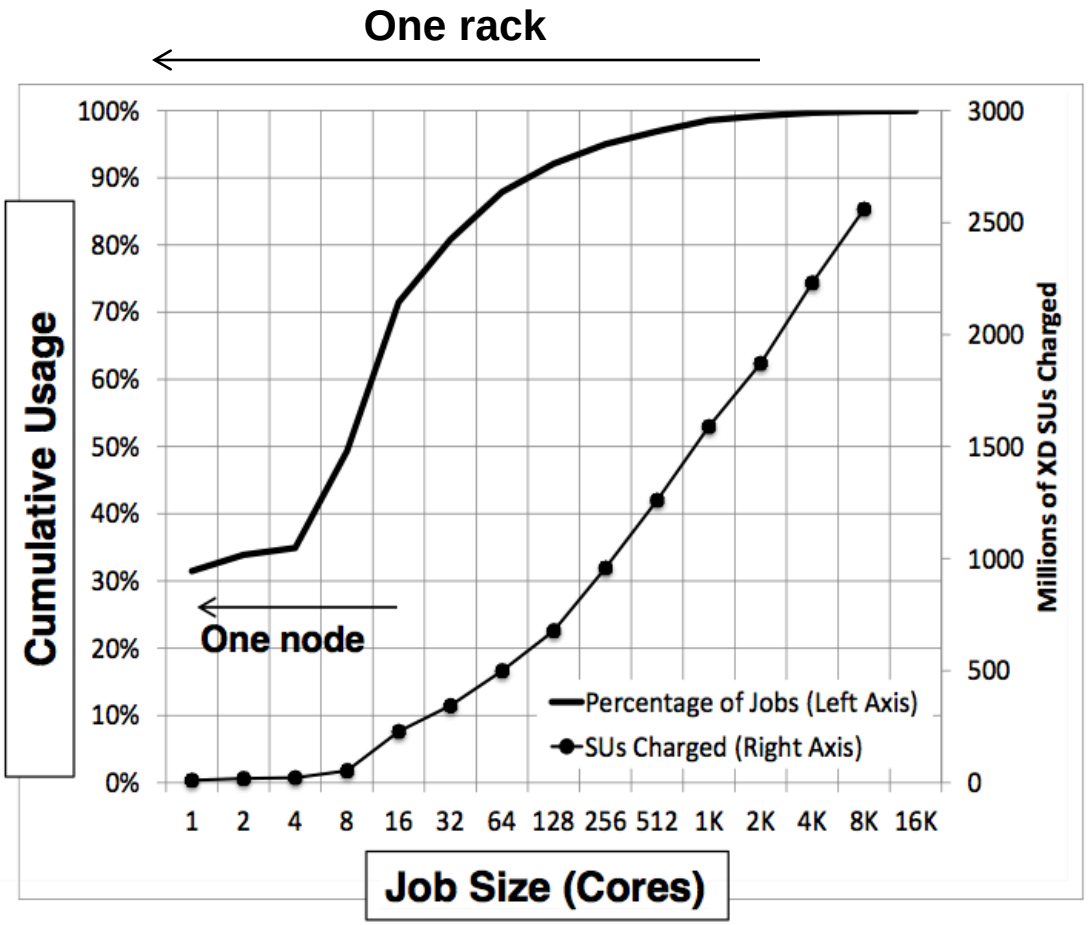
Comet is in response to NSF's solicitation (13-528) to

- **"… expand the use of high end resources to a much larger and more diverse community**
- **… support the entire spectrum of NSF communities**
- **... promote a more comprehensive and balanced portfolio**
- **… include research communities that are not users of traditional HPC systems."**

The long tail of science needs HPC

HPC for the 99%

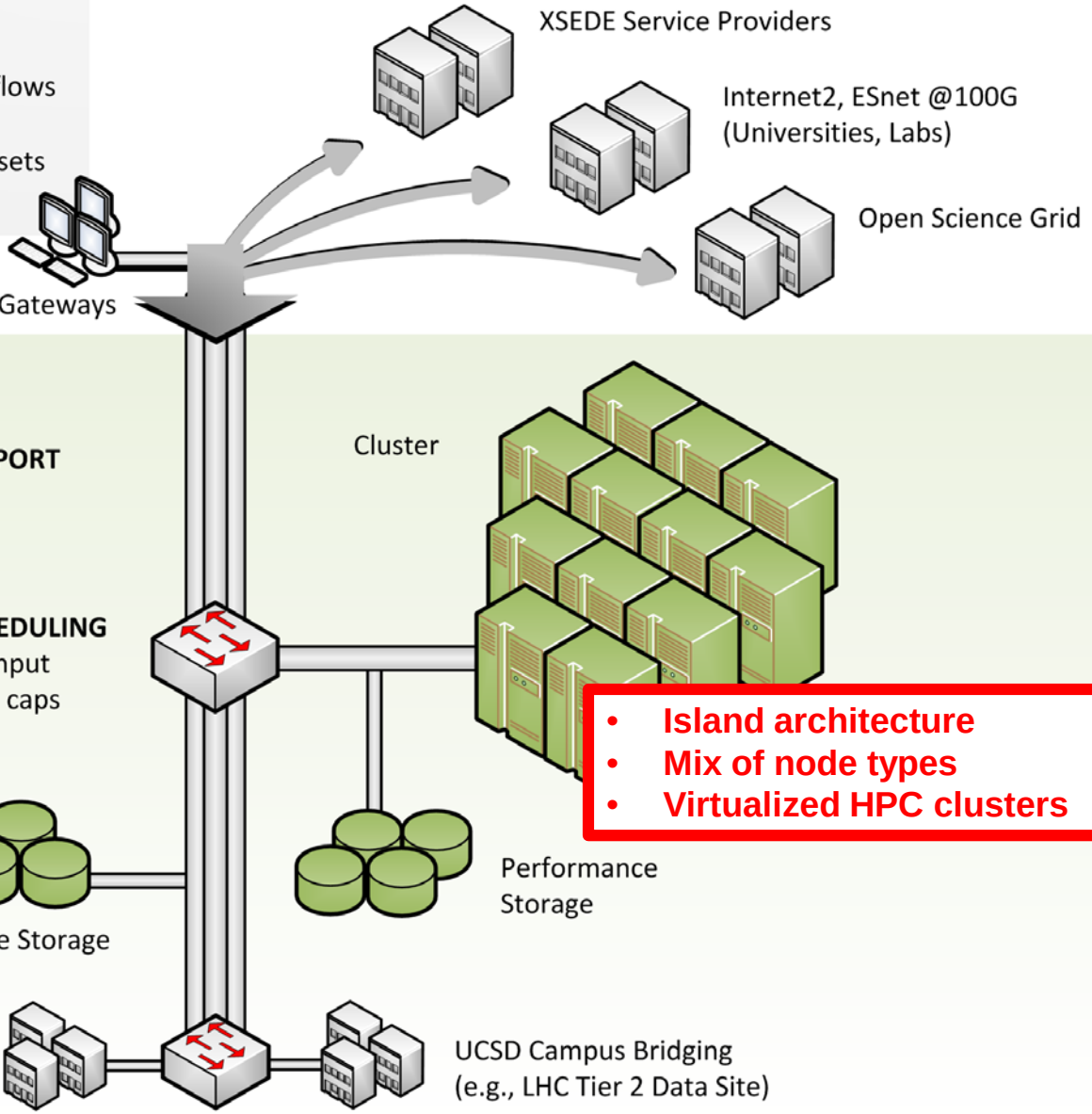
- 99% of jobs run on NSF's HPC resources in 2012 used <2,048 cores
- And consumed >50% of the total core-hours across NSF resources



Comet Will Serve the 99%

CHALLENGES OUR PROPOSAL ADDRESSES

- ✓ Attract new users and communities
- ✓ Support diverse applications with complex workflows
- ✓ Ensure responsiveness for thousands of users
- ✓ Transfer, store, analyze, and share massive data sets
- ✓ Integrate with XSEDE



COMET COMPUTE SYSTEM

Cluster architecture
Fast standard nodes
Large-memory nodes
GPU-accelerated nodes
FDR InfiniBand

Storage architecture
Performance Storage
Durable Storage

Software
Science Gateways
Rich base of installed apps
Virtualization

USER & SYSTEM SUPPORT
New user orientation
XSEDE collaborations
FutureGrid

ALLOCATIONS & SCHEDULING
Optimized for throughput
Per-project allocation caps
Per-job core limits

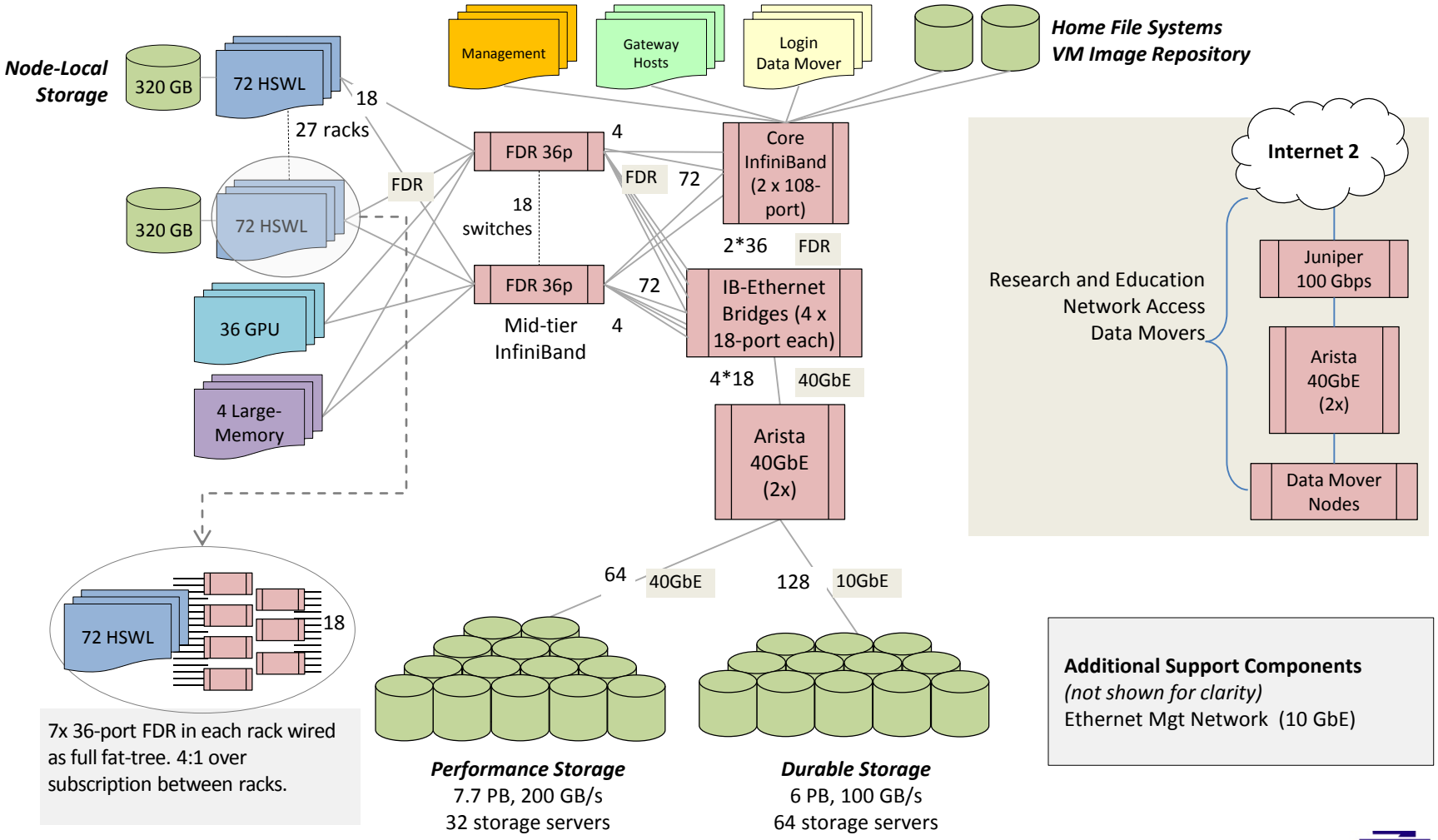
- **Island architecture**
- **Mix of node types**
- **Virtualized HPC clusters**

Comet: System Characteristics

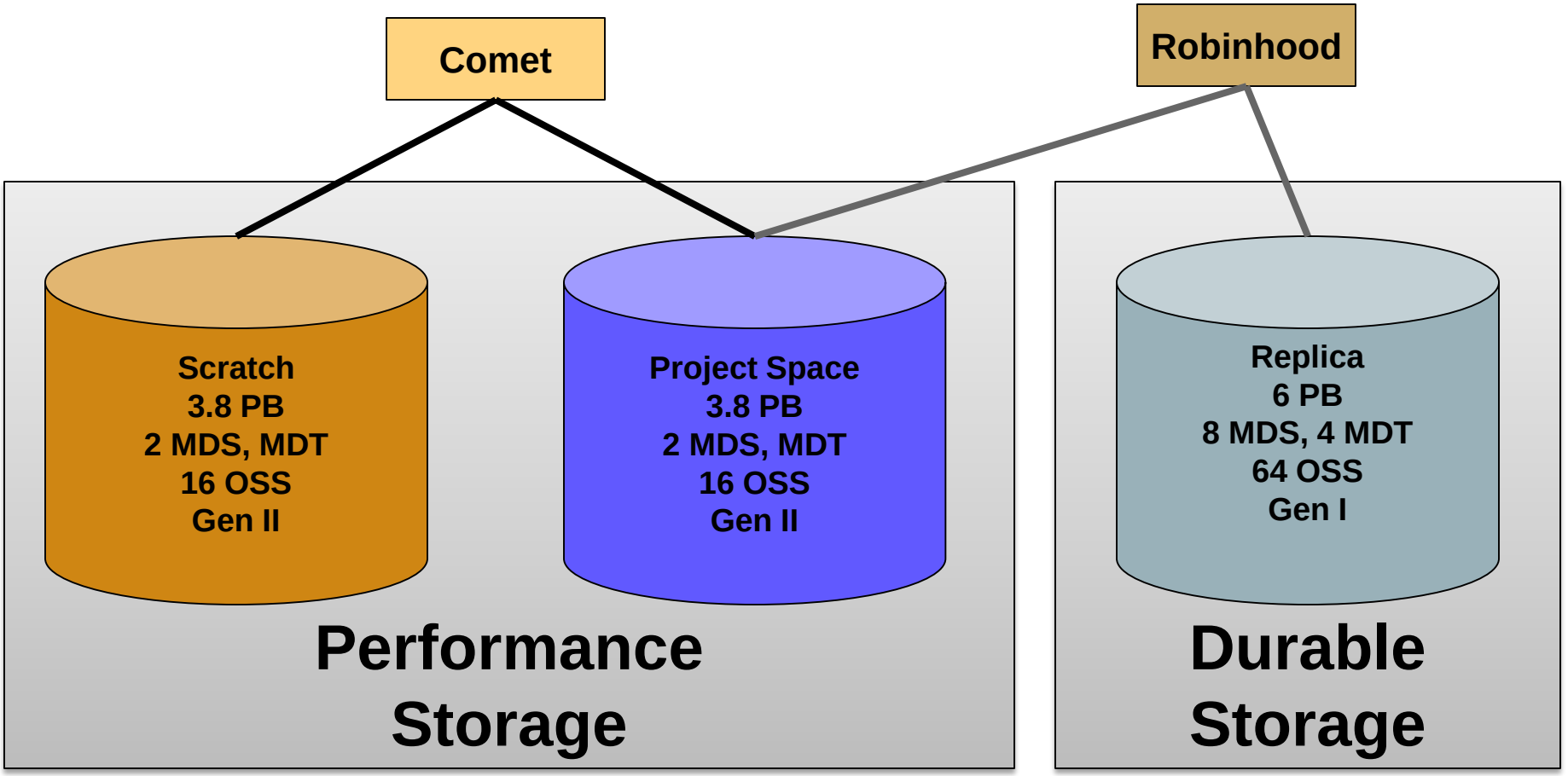
- **Total peak flops ~2.1 PF**
- **Dell primary integrator**
  - Intel Haswell processors w/ AVX2
  - Mellanox FDR InfiniBand
- **1,944 standard compute nodes (46,656 cores)**
  - Dual CPUs, each 12-core, 2.5 GHz
  - 128 GB DDR4 2133 MHz DRAM
  - 2*160GB GB SSDs (local disk)
- **36 GPU nodes**
  - Same as standard nodes *plus*
  - Two NVIDIA K80 cards, each with dual Kepler3 GPUs
- **4 large-memory nodes (June 2015)**
  - 1.5 TB DDR4 1866 MHz DRAM
  - Four Haswell processors/node
- **Hybrid fat-tree topology**
  - FDR (56 Gbps) InfiniBand
  - Rack-level (72 nodes, 1,728 cores) full bisection bandwidth
  - 4:1 oversubscription cross-rack
- **Performance Storage (Aeon)**
  - 7.6 PB, 200 GB/s; Lustre
  - Scratch & Persistent Storage segments
- **Durable Storage (Aeon)**
  - 6 PB, 100 GB/s; Lustre
  - Automatic backups of critical data
- **Home directory storage**
- **Gateway hosting nodes**
- **Virtual image repository**
- **100 Gbps external connectivity to Internet2 & ESNet**

Comet Network Architecture

InfiniBand compute, Ethernet Storage



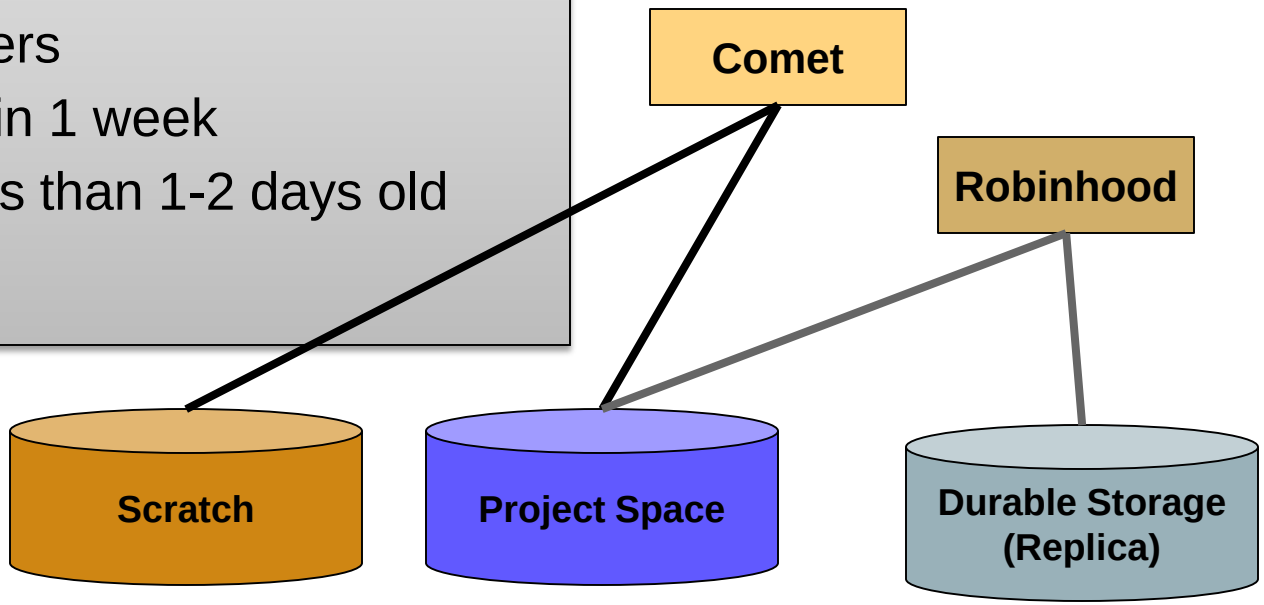
File System Breakdown



Replication & Migration

Durable Storage

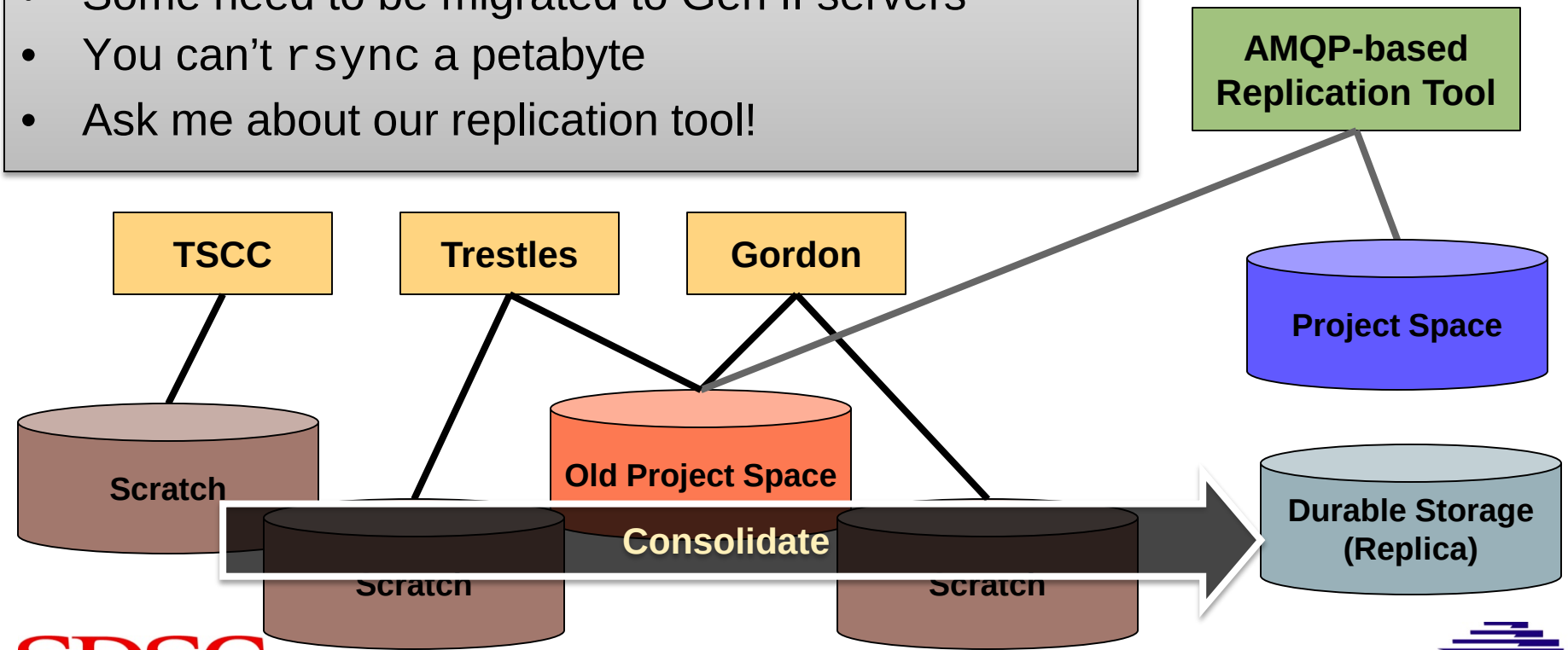
- Reuse current Data Oasis servers as slightly stale replica
- Replicates allocate project space
- Think “disaster recovery” not “backup”
- Not accessible to users
- Goal is full sync within 1 week
- Exclude changes less than 1-2 days old
- Using Robinhood



Replication & Migration

Migration

- Building up Durable Storage requires consolidating several production file systems
- Some need to be migrated to Gen II servers
- You can't `rsync` a petabyte
- Ask me about our replication tool!



Lustre & ZFS Performance

Goal: 6.25 GB/s per server

- ZFS (array) performance
- Lustre performance
- LNET performance

Resolutions:

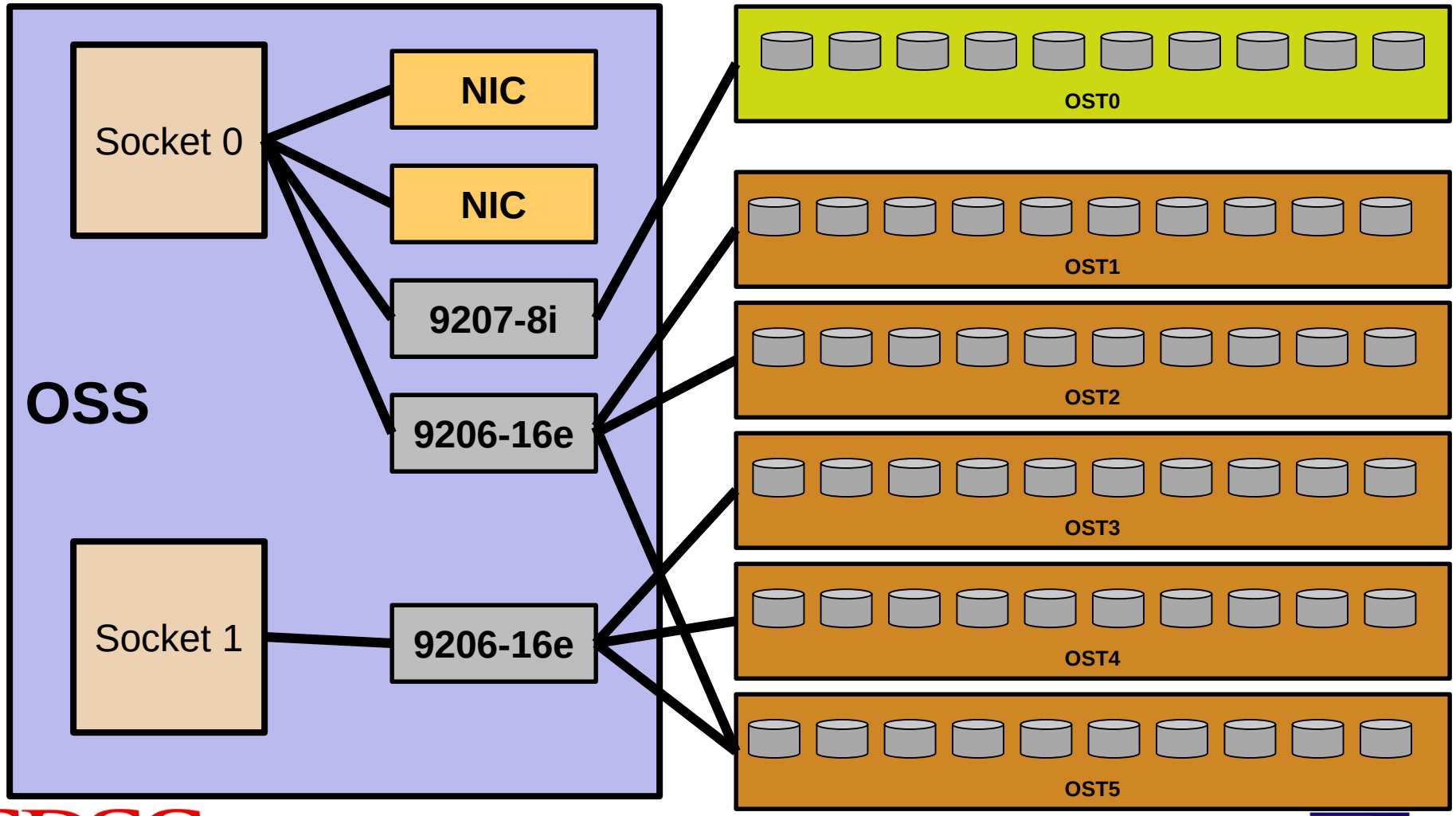
- **Linux**
  - Kernel 3.10 (better IO)
- **ZFS**
  - Version including large block (1024k recordsize) support
  - Tuning for prefetch
    - `zfs_vdev_cache_size`
    - `zfs_vdev_cache_max`
- **Lustre**
  - Patch to utilize zero copy buffers
- **LNET**
  - SMP affinity for network tasks
  - Free cores for NIC IRQ handling
- **Hardware**
  - Attach both 40GbE NICs to same NUMA domain

Aeon OSS

Ground Up Design for ZFS

- 2 @ LSI 9206-16e HBA (to JBOD)
- 1 @ LSI 9207-8i HBA (to front of chassis)
- 2 @ MCX313A-BCBT 40GbE NICs
- 2 @ Xeon E5-2650v2 @ 2.60GHz (8 core/processor)
- 128 GB DDR3 1867 MHz DRAM
- 60 @ ST4000NM0034
  - 4TB
  - 4K native, 512 emulation (ashift=12)
  - 10 chassis
  - 50 JBOD
  - Rated 225 MB/s peak BW
- **2 servers per chassis**
- **Multipath for chassis and JBODs**

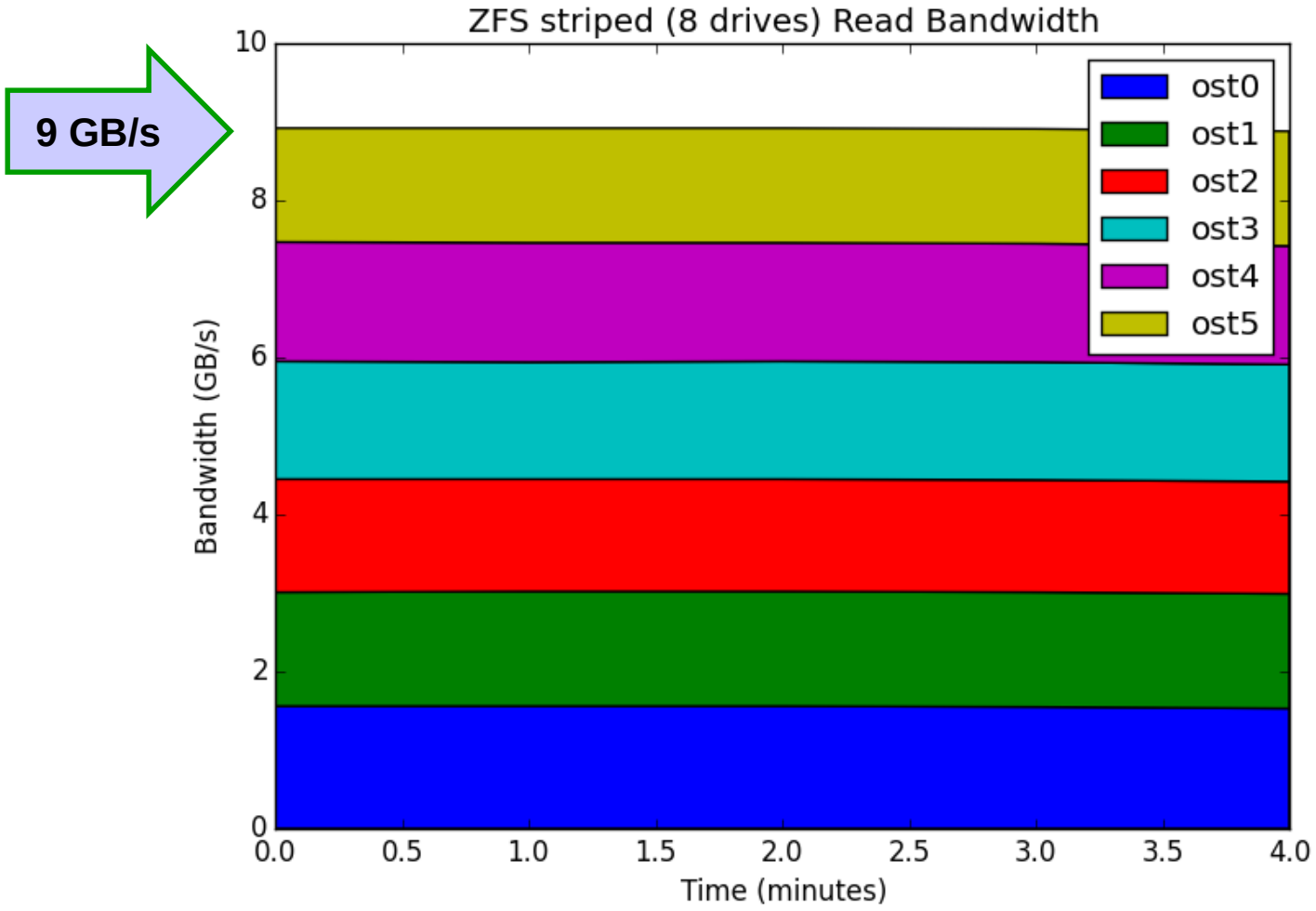
OSS and Native ZFS Perfomance



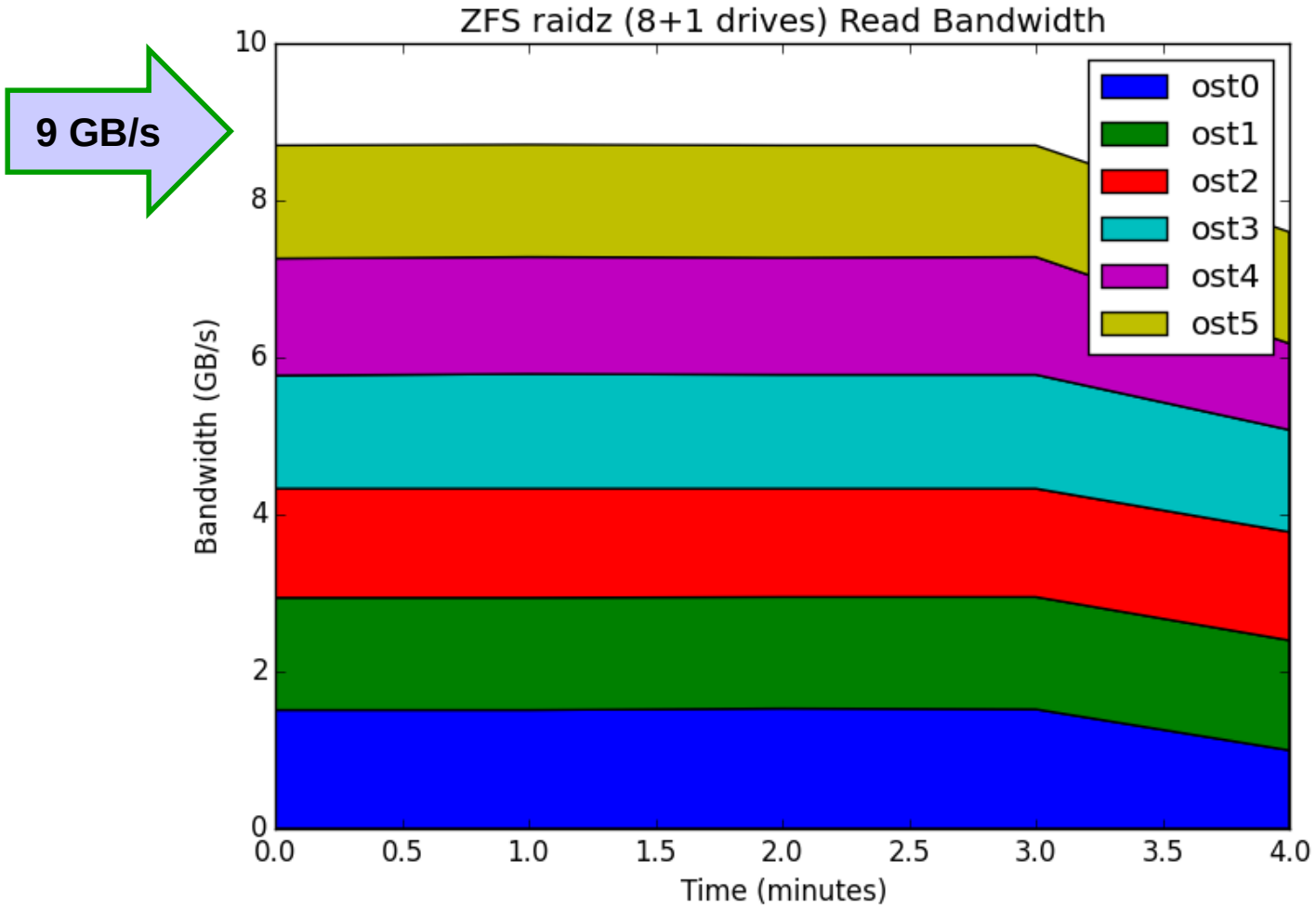
ZFS Performance: Read

```
[root@panda-mds-19-5 tests]# cat readdata.sh
#!/bin/bash

zpool iostat 60 > /share/apps/bin/wombat/tests/results/zpool-read-$(hostname -a).txt &
iostatpid=$!

for ost in 0 1 2 3 4 5
do
    for blob in 0 1 2 3
    do
          dd of=/dev/null if=/mnt/ost${ost}-zfs/randblob${blob} bs=16M count=$((8 * 1024)) &
    done
done

for p in $(pidof dd)
do
    wait $p
done

kill $iostatpid
```

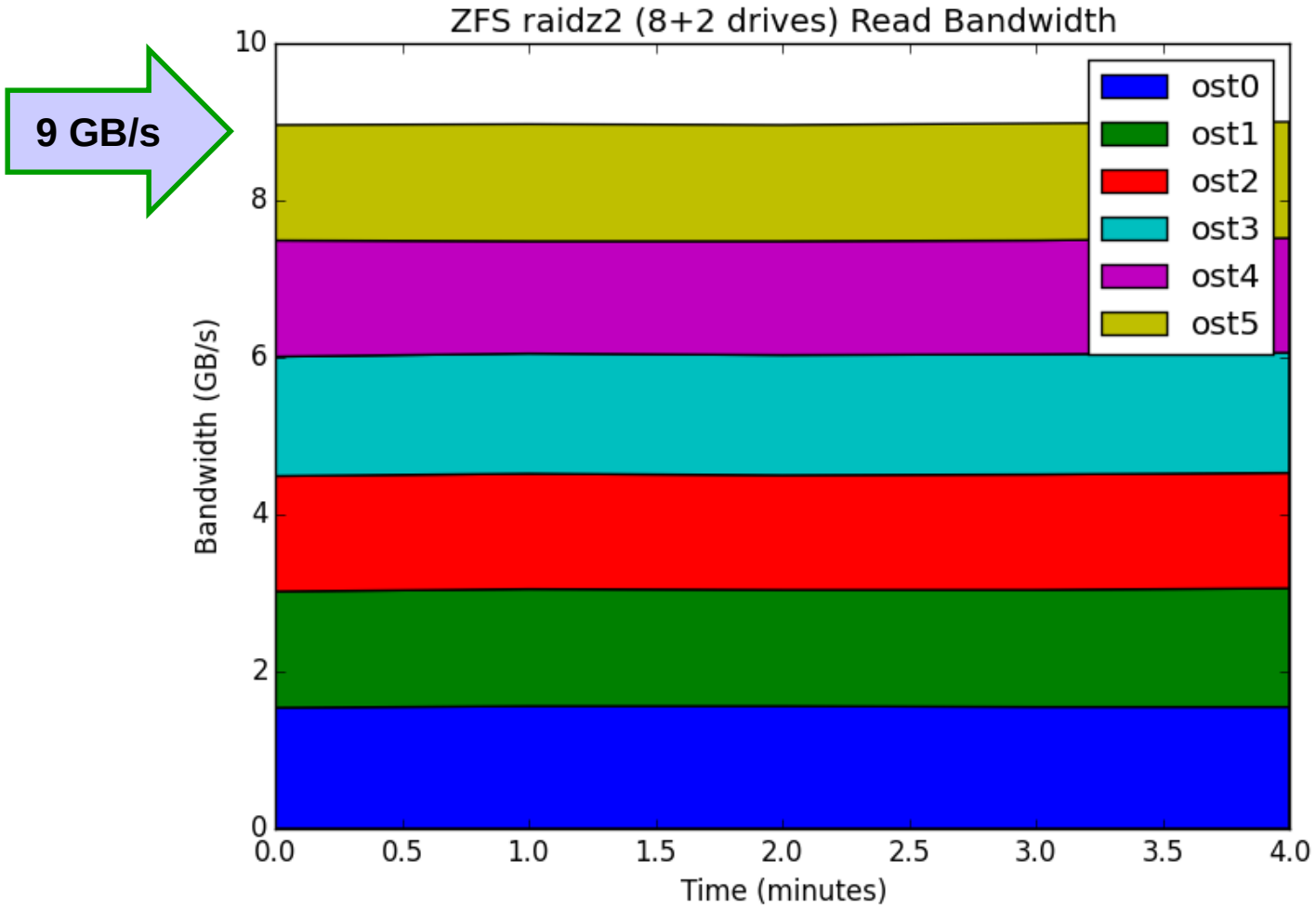
ZFS Performance: Read



ZFS Performance: Read



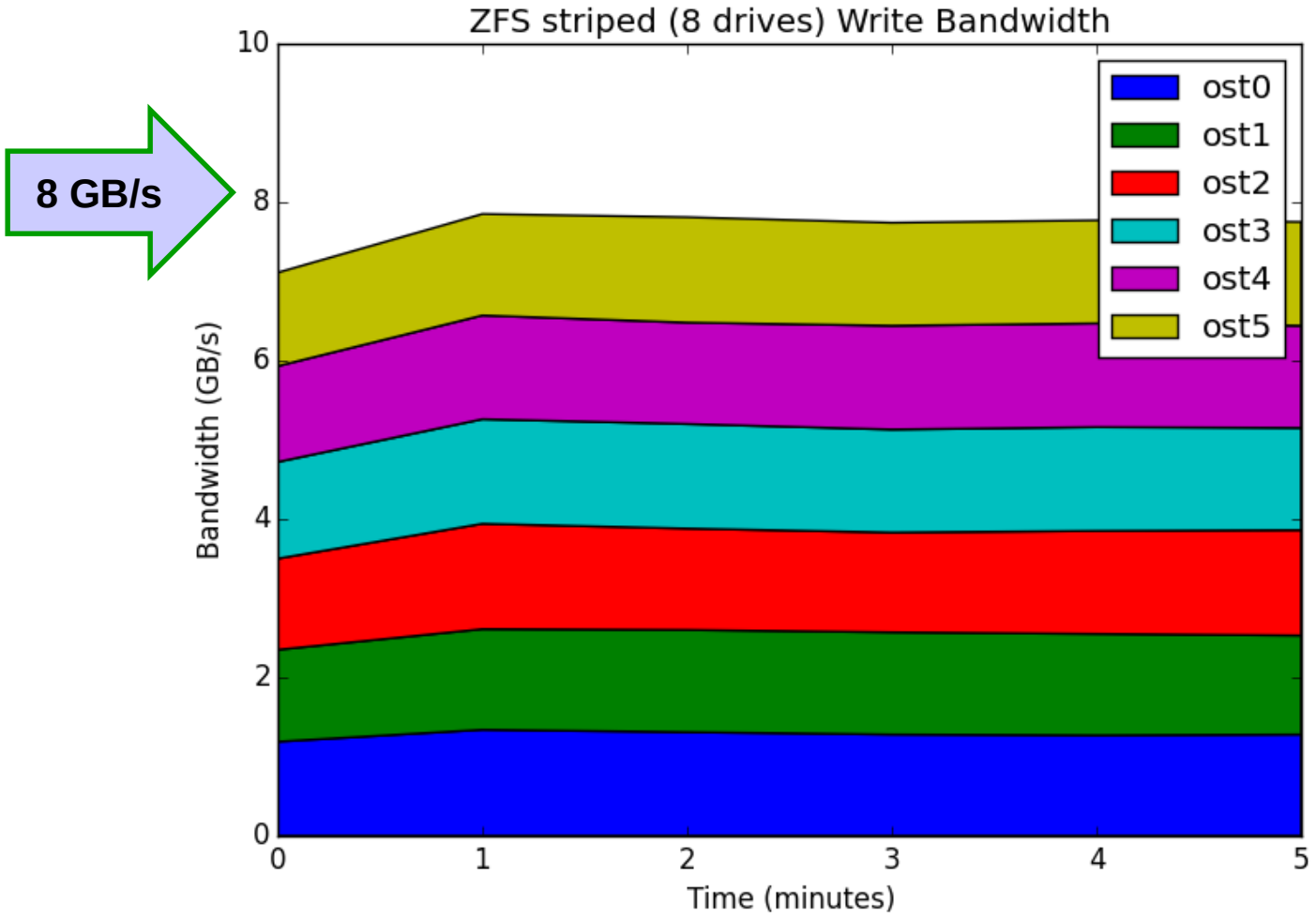
ZFS Performance: Read



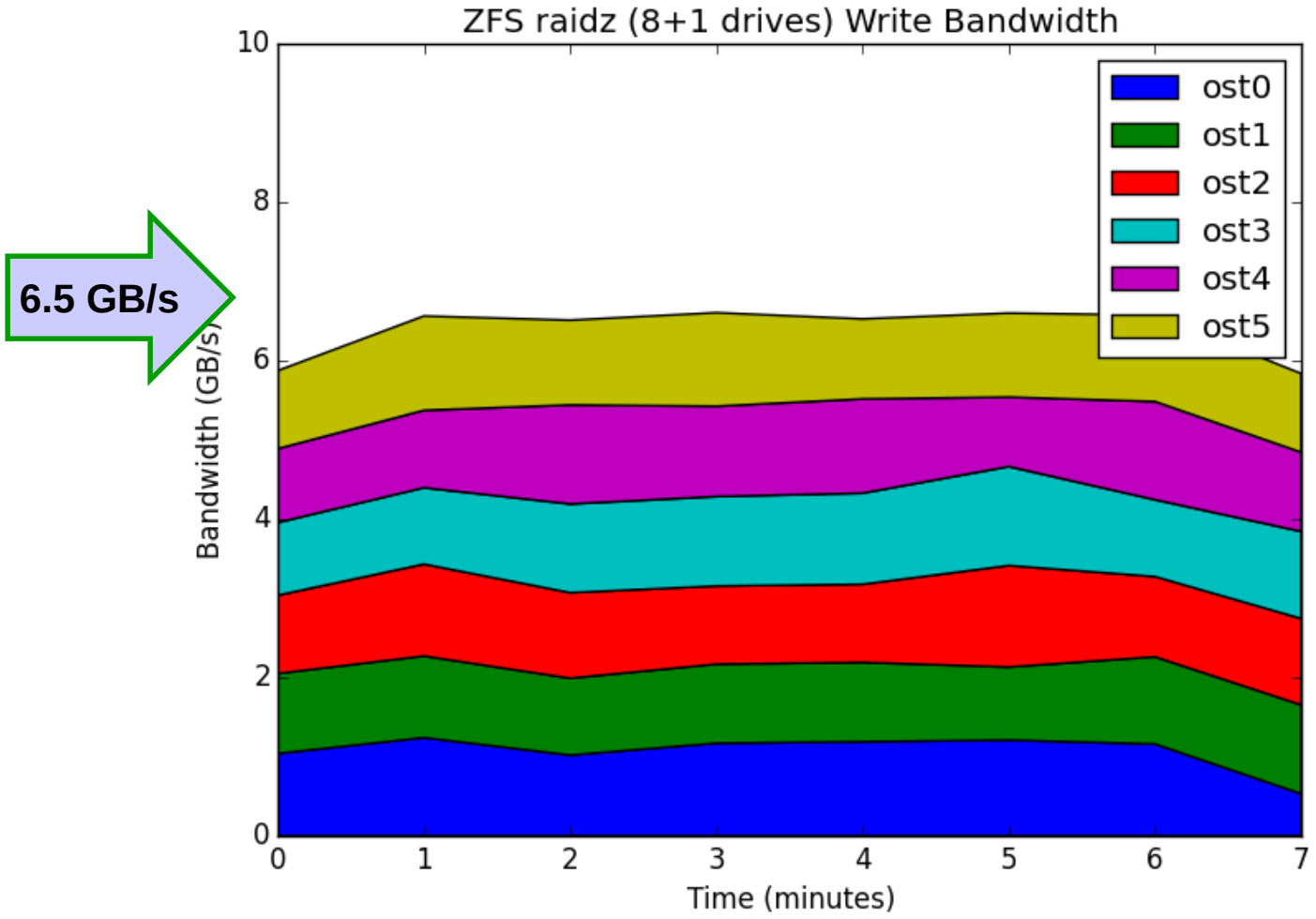
ZFS Performance: Write

```
[root@panda-mds-19-5 tests]# cat gendata.sh
#!/bin/bash

# mkrandfile generates 128GB files using drand48
mkrand=/share/apps/bin/wombat/tests/mkrandfile

zpool iostat 60 > /share/apps/bin/wombat/tests/results/zpool-write-$(hostname -a).txt &
iostatpid=$!

for ost in 0 1 2 3 4 5
do
    for blob in 0 1 2 3
    do
          $mkrand /mnt/ost${ost}-zfs/randblob${blob} &
    done
done

for p in $(pidof mkrandfile)
do
    wait $p
done

kill $iostatpid
```

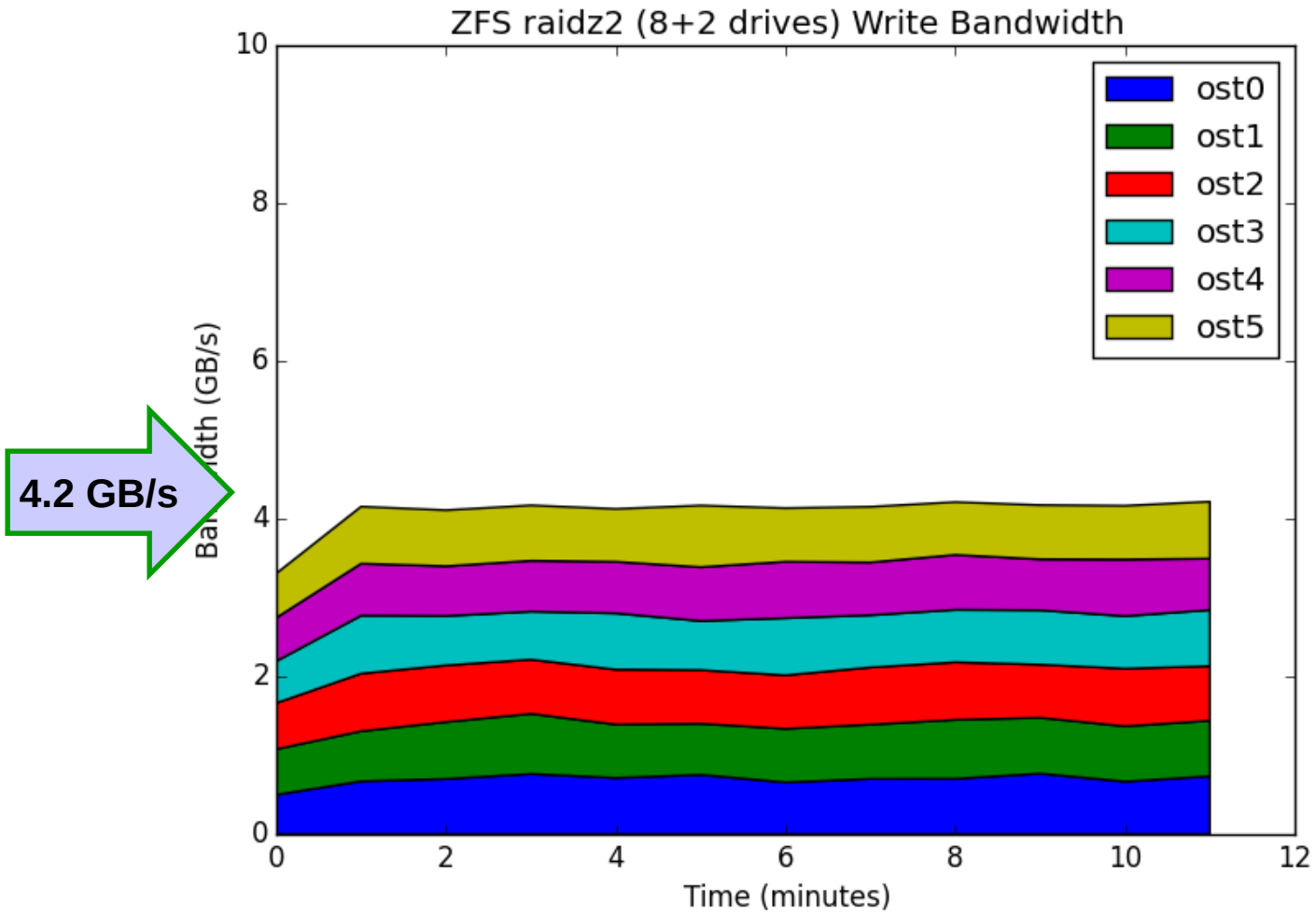
ZFS Performance: Write



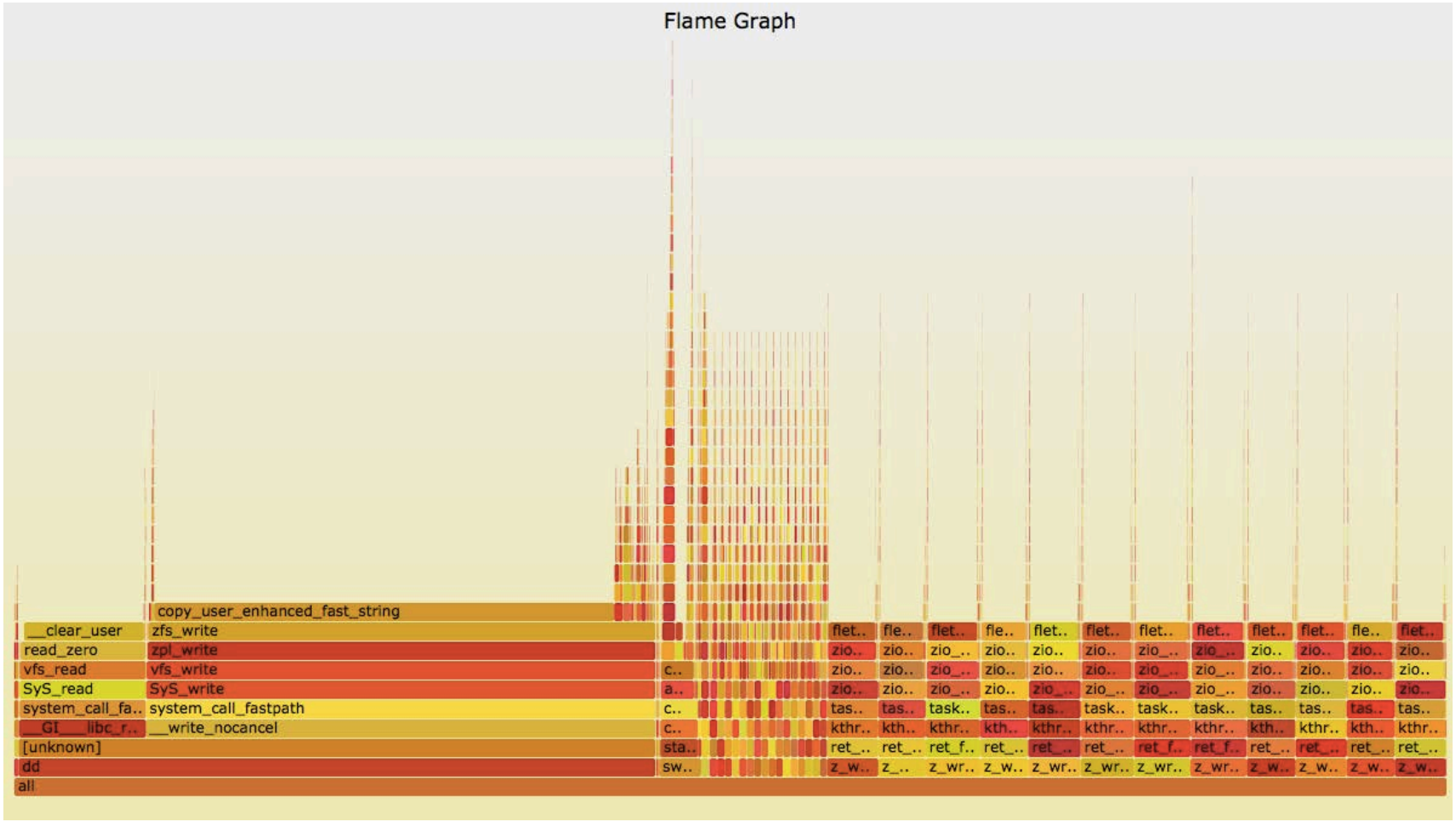
ZFS Performance: Write



ZFS Performance: Write

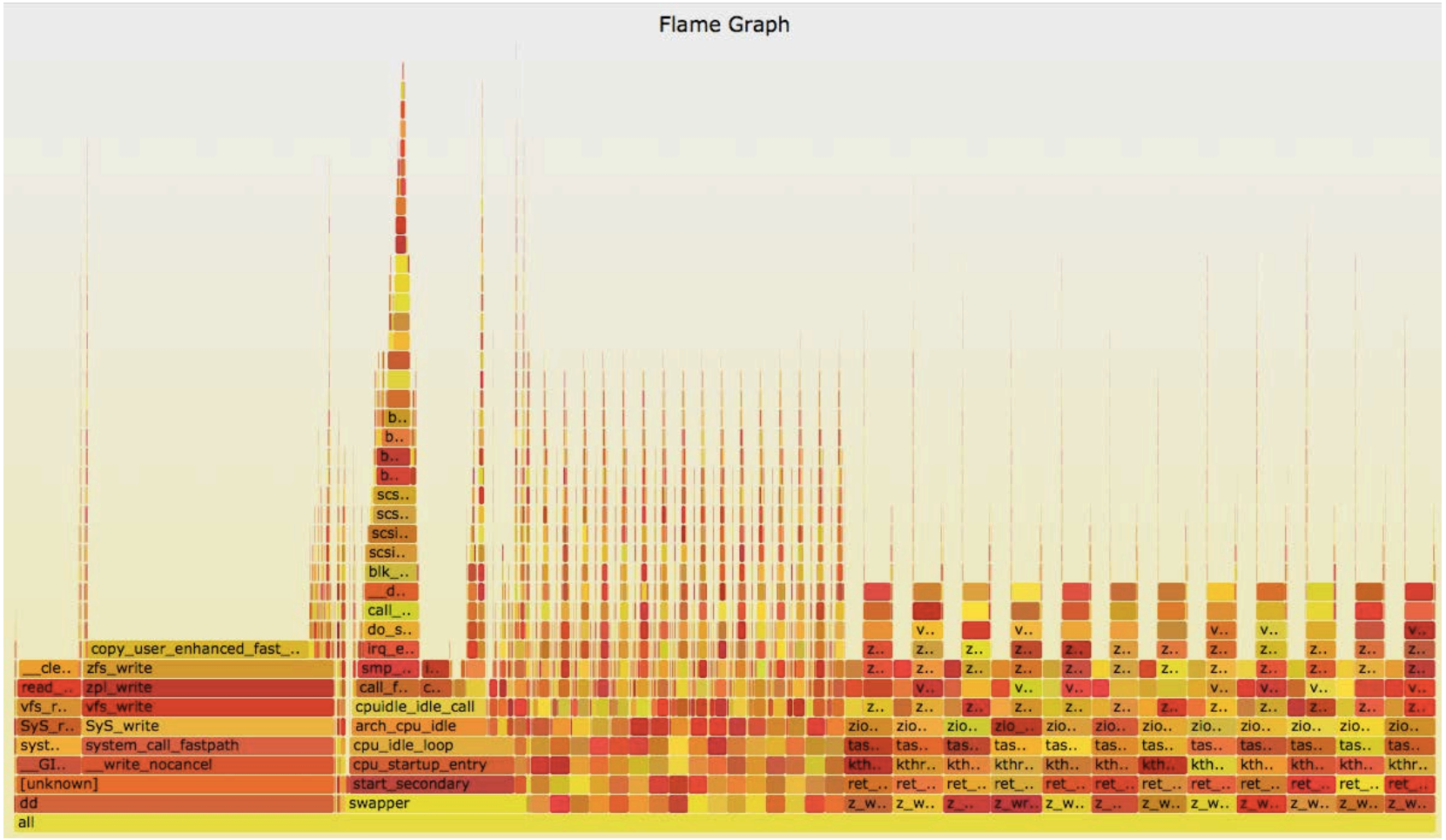


perf Counters Striped zpool



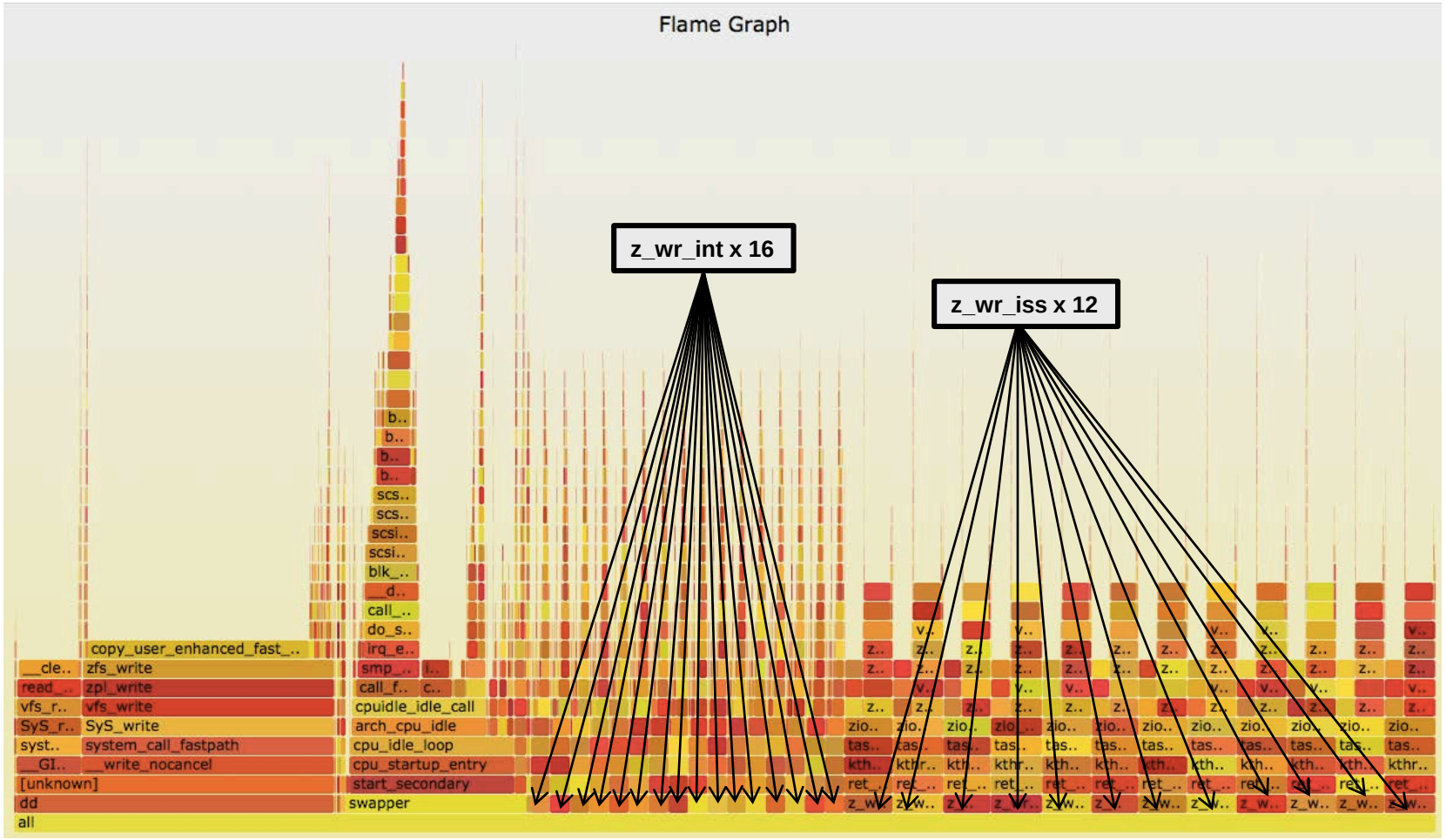
http://users.sdsc.edu/~rpwagner/perf-kernel-6ost-dd-stripe.svg

`perf` Counters RAIDZ2 zpool



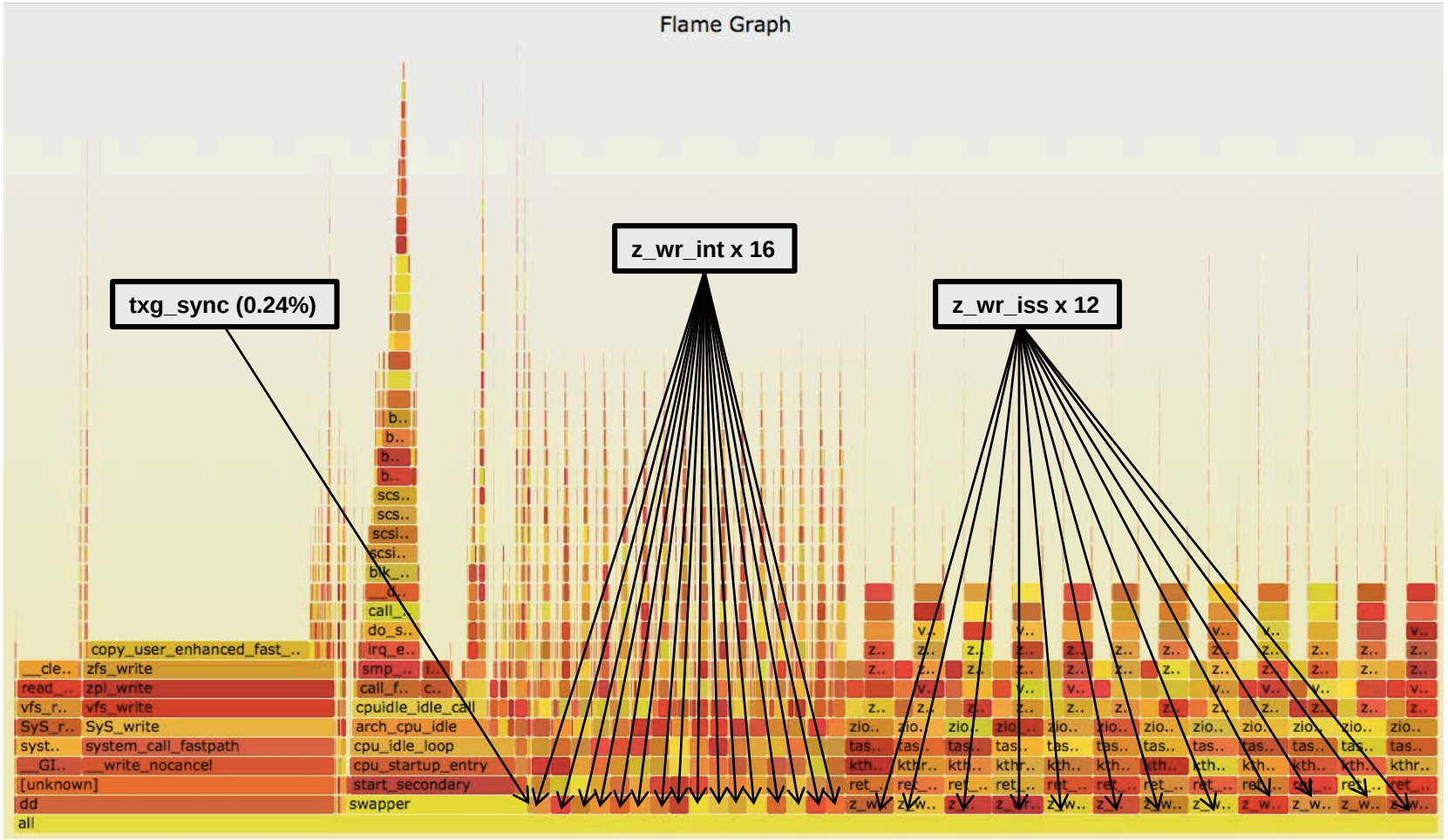
`http://users.sdsc.edu/~rpwagner/perf-kernel-6ost-dd-raidz2.svg`

perf Counters RAIDZ2 zpool

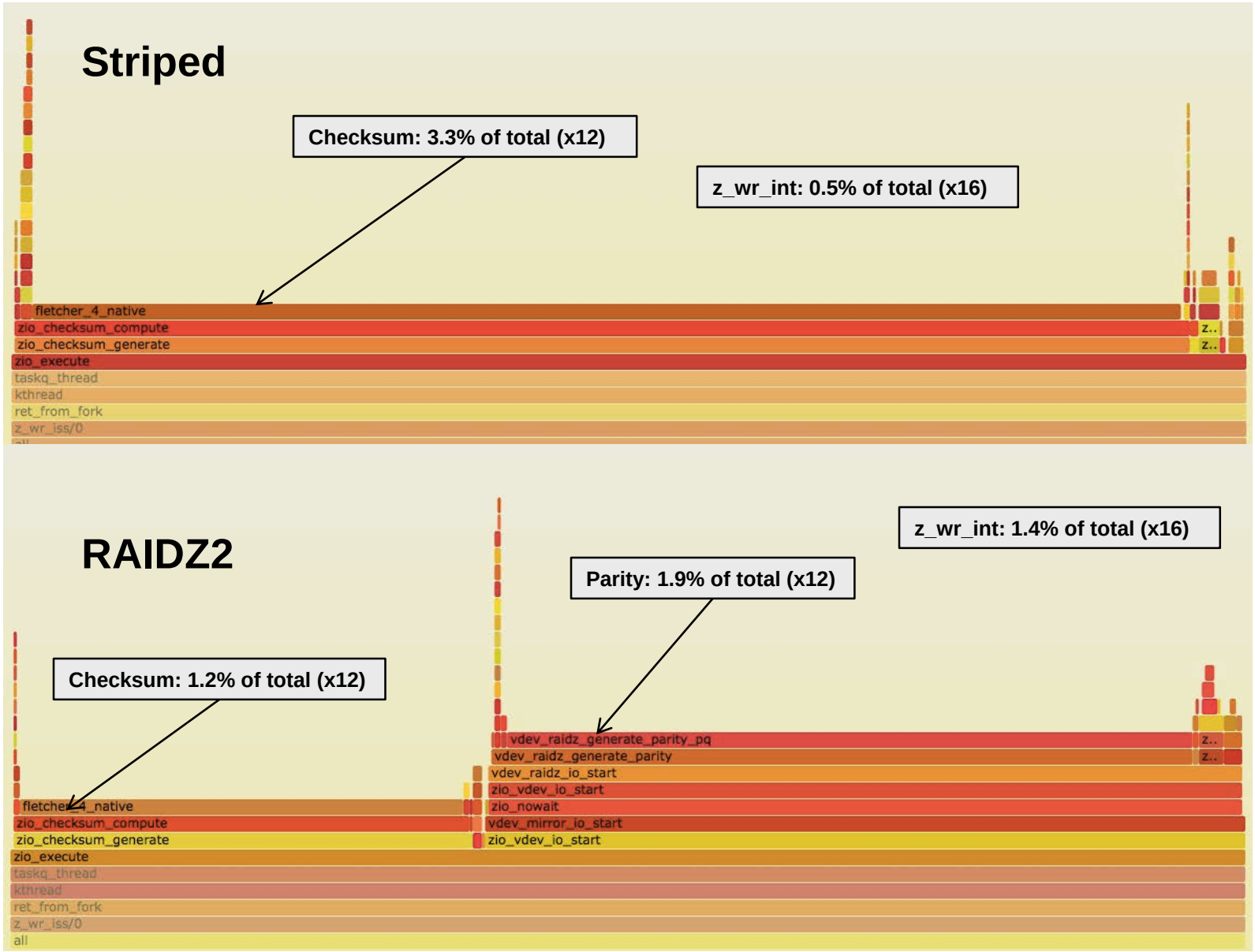


http://users.sdsc.edu/~rpwagner/perf-kernel-6ost-dd-raidz2.svg

perf Counters RAIDZ2 zpool



http://users.sdsc.edu/~rpwagner/perf-kernel-6ost-dd-raidz2.svg


Striped
Checksum: 3.3% of total (x12)
z_wr_int: 0.5% of total (x16)
fletcher_4_native
zio_checksum_compute
zio_checksum_generate
zio_execute
taskq_thread
kthread
ret_from_fork
z_wr_iss/0
RAIDZ2
z_wr_int: 1.4% of total (x16)
Parity: 1.9% of total (x12)
Checksum: 1.2% of total (x12)
vdev_raidz_generate_parity_pq
vdev_raidz_generate_parity
vdev_raidz_io_start
zio_vdev_io_start
zio_nowait
vdev_mirror_io_start
zio_vdev_io_start
fletcher_4_native
zio_checksum_compute
zio_checksum_generate
zio_execute
taskq_thread
kthread
ret_from_fork
z_wr_iss/0
all

ZFS & AVX

```
>>> dt_striped = 100
>>> stuff = dt_striped - (16*0.5 + 12*3.3)
>>> dt_raidz2 = stuff + 16*1.4 + 12*3.3*(1 + 1.9/1.2)
>>> slowdown = dt_striped/dt_raidz2
>>> print "%0.2f GB/s" % (slowdown*8)
4.52 GB/s
```

Not far off the 4.2 GB/s observed.

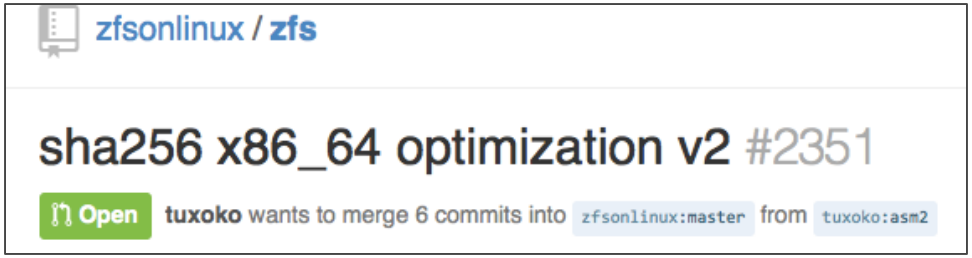
ZFS & AVX

```
>>> dt_striped = 100
>>> stuff = dt_striped - (16*0.5 + 12*3.3)
>>> dt_raidz2 = stuff + 16*1.4 + 12*3.3*(1 + 1.9/1.2)
>>> slowdown = dt_striped/dt_raidz2
>>> print "%0.2f GB/s" % (slowdown*8)
4.52 GB/s
```

Not far off the 4.2 GB/s observed.

Help is on the way!

- Work started on AVX(2) optimizations for checksums
- Hoping to see this extended to parity



https://github.com/zfsonlinux/zfs/pull/2351

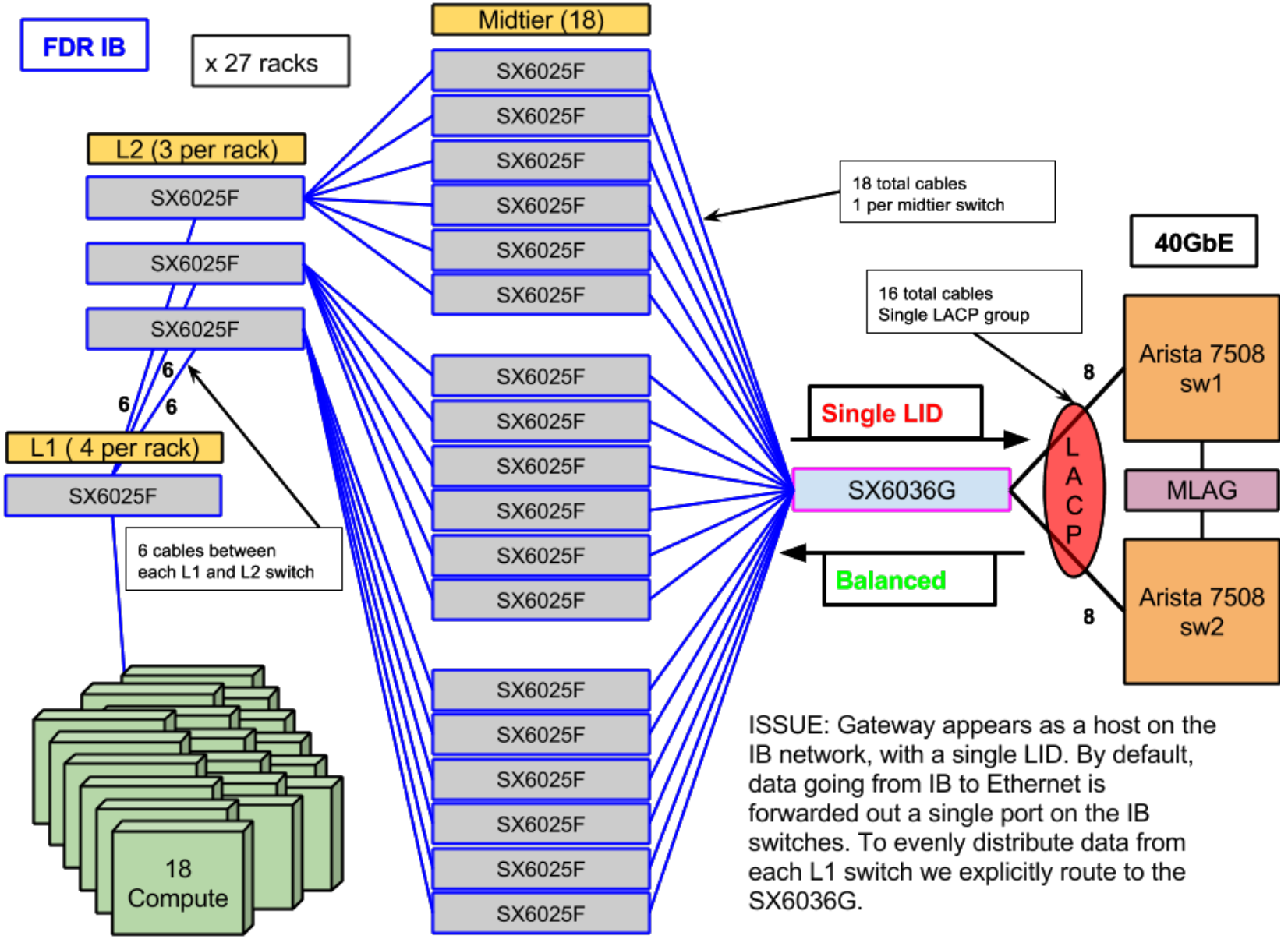
IB-Ethernet Bridges

Challenge: 64 x 40 Gb/s bidirectional

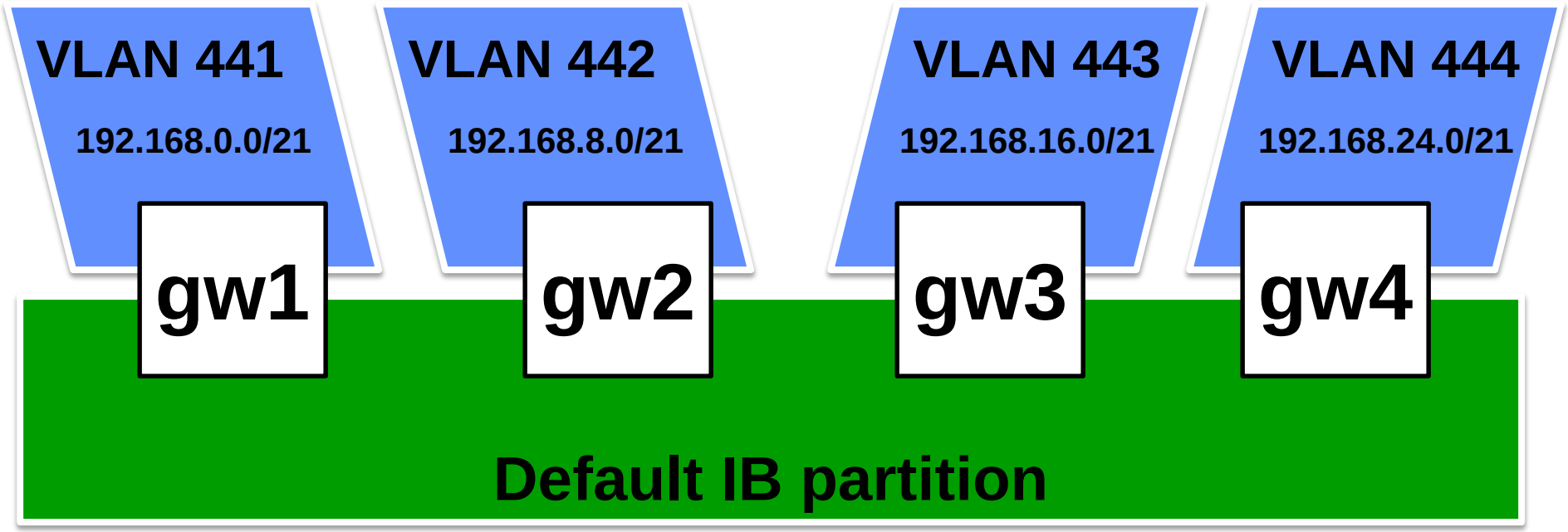
- Single LID per gateway
- Proxy-ARP performance
- LNET

Resolutions:

- **Proxy-ARP**
  - 1 VLAN per gateway
  - 4 VLANs map to default IB partition
  - Core Aristas routing between IP subnets
  - 4 IP addresses per client
- **IB Subnet**
  - Explicit routing from leaf and spine switches to gateways
  - Max 12 links per rack outbound
- **LNET**
  - Single network “tcp”
- **Hardware**
  - Migrated gateway IB connections to midtier switches

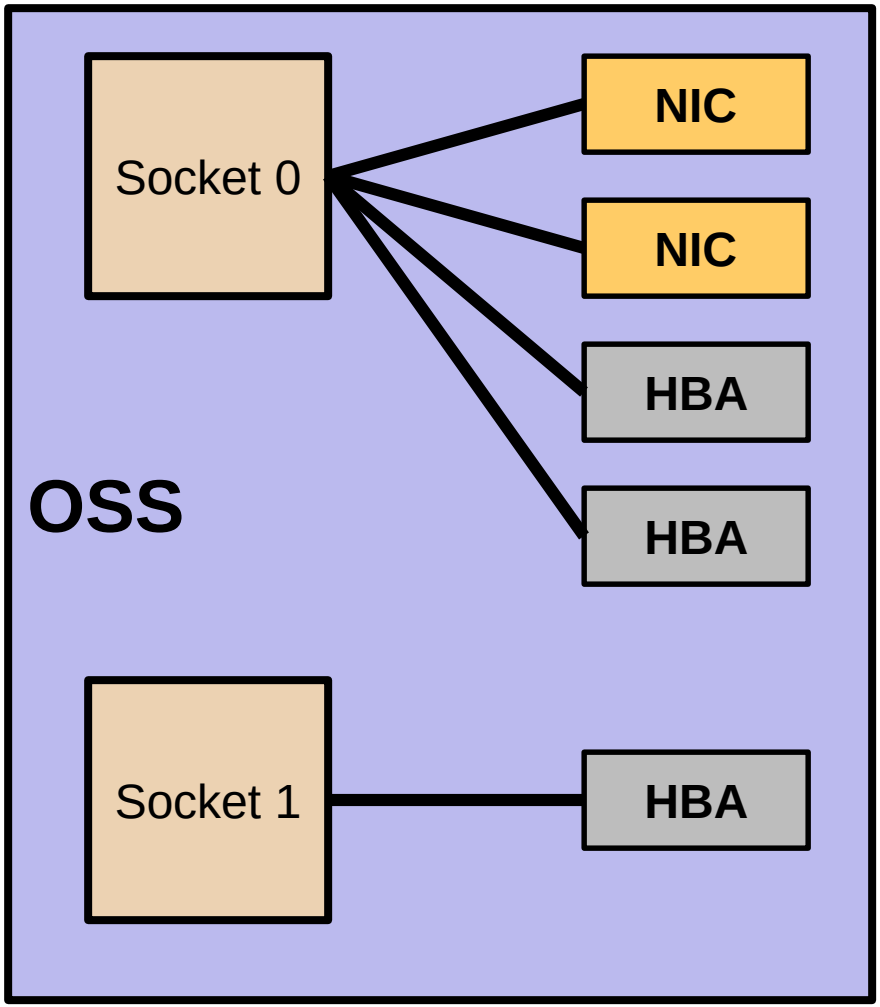

FDR IB
x 27 racks
Midtier (18)
SX6025F
SX6025F
SX6025F
SX6025F
SX6025F
L2 (3 per rack)
SX6025F
SX6025F
SX6025F
6
6
6
L1 (4 per rack)
SX6025F
6 cables between each L1 and L2 switch
SX6025F
SX6025F
SX6025F
SX6025F
SX6025F
SX6025F
SX6025F
SX6025F
SX6025F
SX6025F
SX6025F
SX6025F
18 Compute
18 total cables
1 per midtier switch
40GbE
16 total cables
Single LACP group
Arista 7508 sw1
8
Single LID
SX6036G
LACP
MLAG
Balanced
Arista 7508 sw2
8
ISSUE: Gateway appears as a host on the IB network, with a single LID. By default, data going from IB to Ethernet is forwarded out a single port on the IB switches. To evenly distribute data from each L1 switch we explicitly route to the SX6036G.

IB-Ethernet Bridges



- 9 servers (1 MDS, 8 OSS) per VLAN
- Each client has 4 IPoIB aliases (ib0:1, ib0:2, ib0:3, ib0:4)
- Single LNET NID using “tcp(ib0:1)”
- Allows gateways to handle ARP requests appropriately

40GbE NICs & NUMA

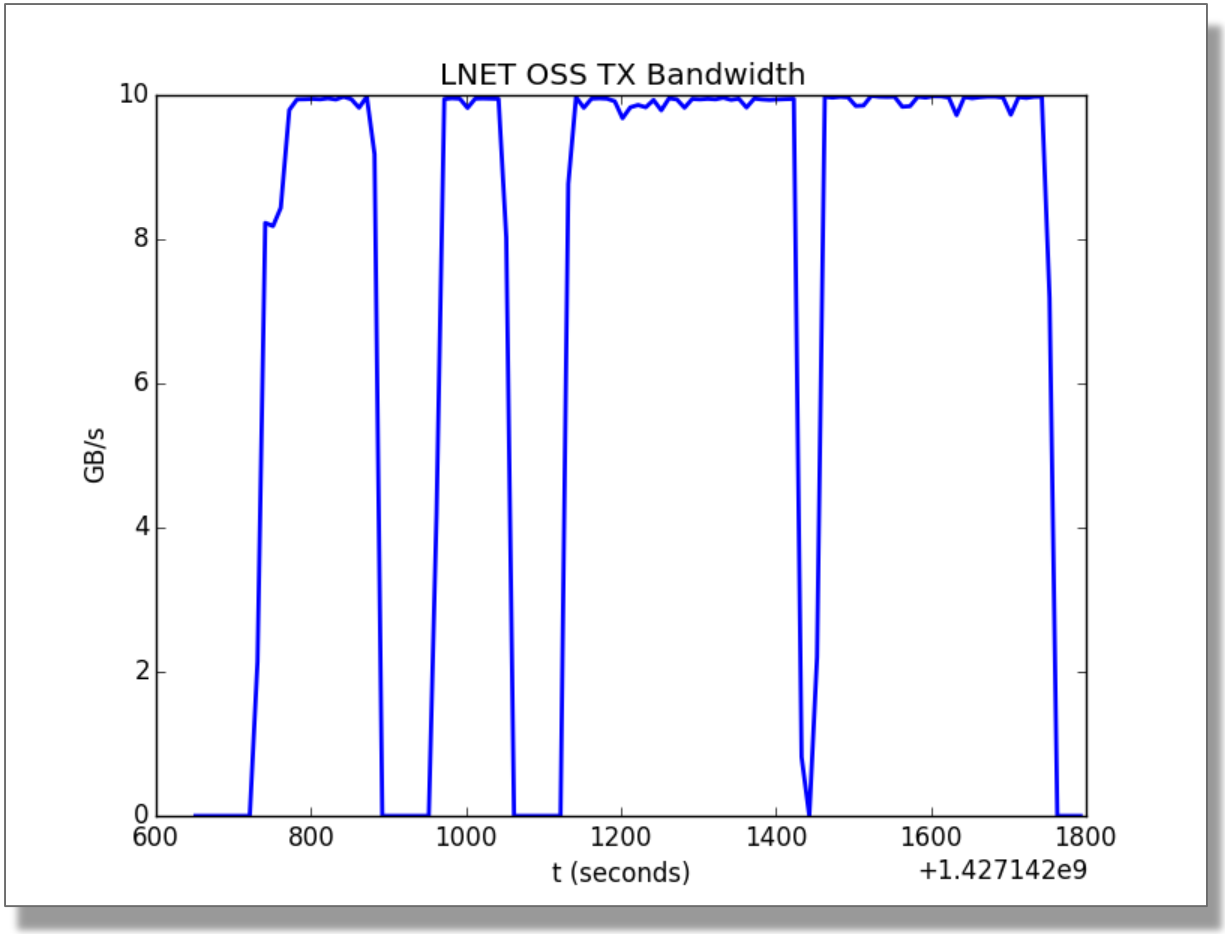


- Critical to have both NICs in same NUMA domain (think bonding & tcp)
- Good to leave some cores free for IRQ handling
- irqbalance and setting affinity not helpful
- See LU-6228 for a discussion

```
[root@wombat-oss-20-1 ~]# for i in lnet libcfs ksocklnd
> do
> echo $i.conf
> cat /etc/modprobe.d/$i.conf
> done
lnet.conf
options lnet networks="tcp(bond0)"
libcfs.conf
options libcfs cpu_pattern="0[2-7] 1[8-15]"
options ost oss_io_cpts="[0,1]" oss_cpts="[1]"
ksocklnd.conf
options ksocklnd nscheds=6 peer_credits=48 credits=1024
```

https://jira.hpdd.intel.com/browse/LU-6228

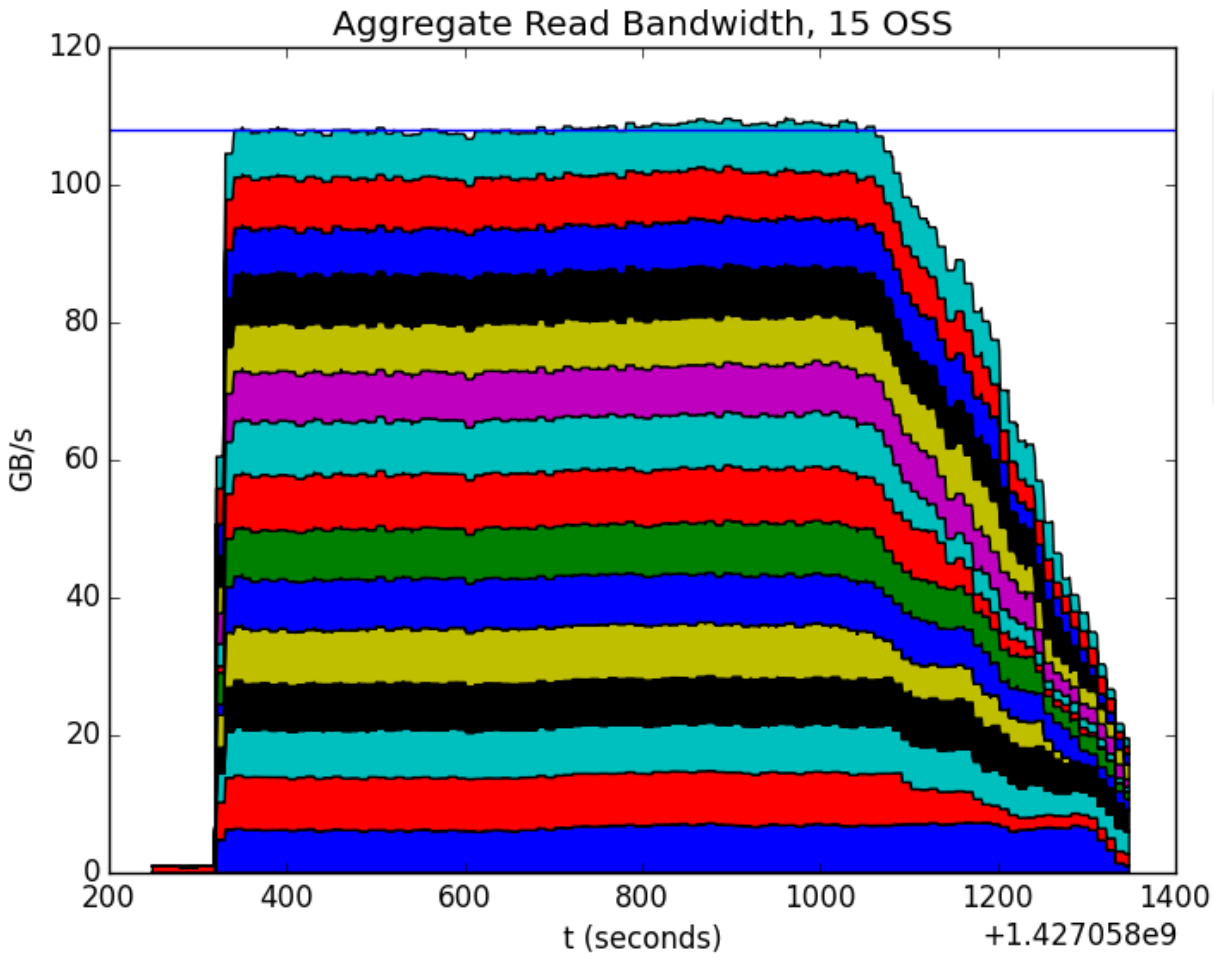
LNET Performance Excellent



LNET Self Test
8 clients, 1 OSS
concurrency=16
check=simple
size=1M

Client reads at 10 GB/s
Client writes at 8 GB/s

Putting it All Together: Wombat



- IOR: 12 clients per OSS
- 108 GB/s total
- 7.2 GB/s average
- Individual OSS go to 8+ GB/s

Fine print: This should have been 16 servers at 115+ GB/s. I screwed up my striping and didn't discover it until making this plot.

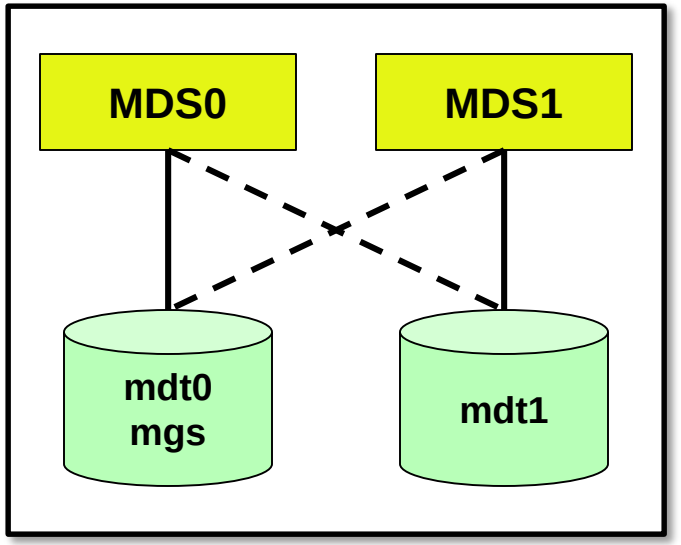
ZFS & Lustre Metadata

MDS

- Same base hardware as OSS
- No JBOD
- Dual 10GbE NICs
- Intel® Xeon® E5-2670v2 (2.50GHz, 10 c)

MDT Configuration

- 12 SSDs per MDS
- Intel® SSD DC S3500 Series 300GB
- RAID10
  - Stripe of over 5 mirrored pairs, 2 global spares
- DNE
- Failover



ZFS & Lustre Metadata

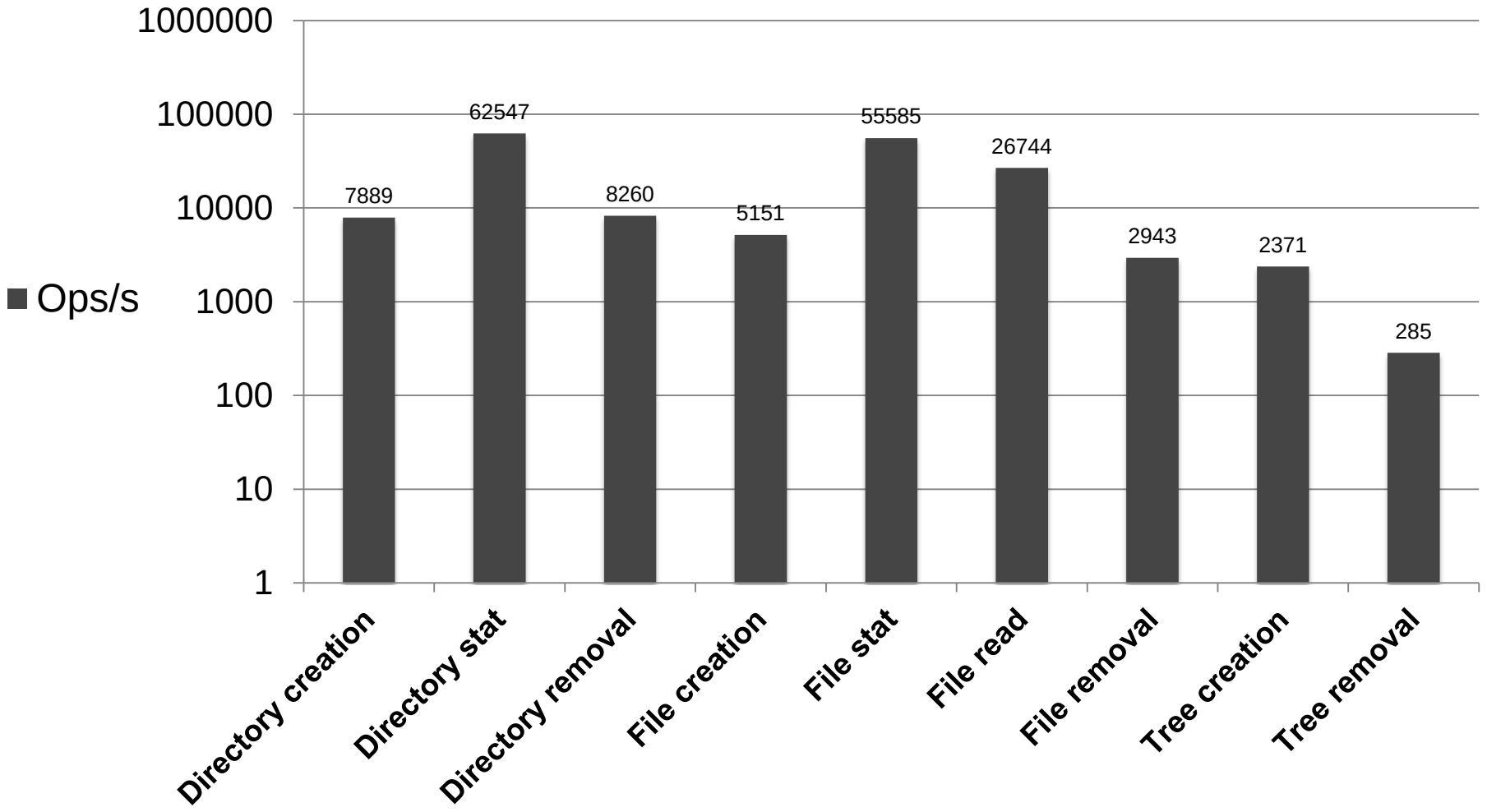
On MDS0:

```
zfs set recordsize=4096 wombat-mdt0
zfs create wombat-mdt0/wombat-mdt0

mkfs.lustre --reformat --mdt --mgs --failover=<NID of MDS1> \
  --fsname=wombat --backfstype=zfs --index=0 wombat-mdt0/wombat-mdt0
```

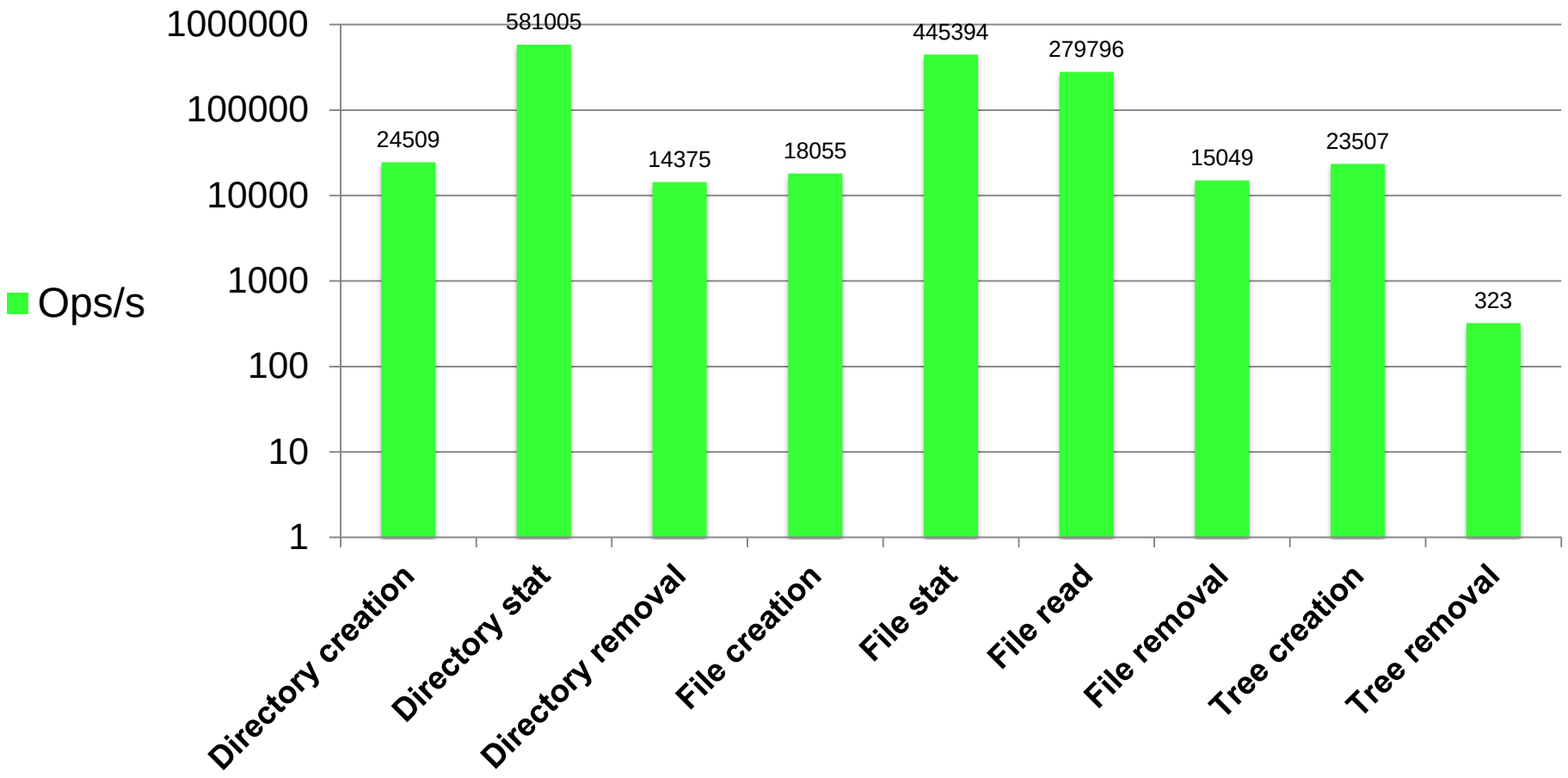
On MDS1:

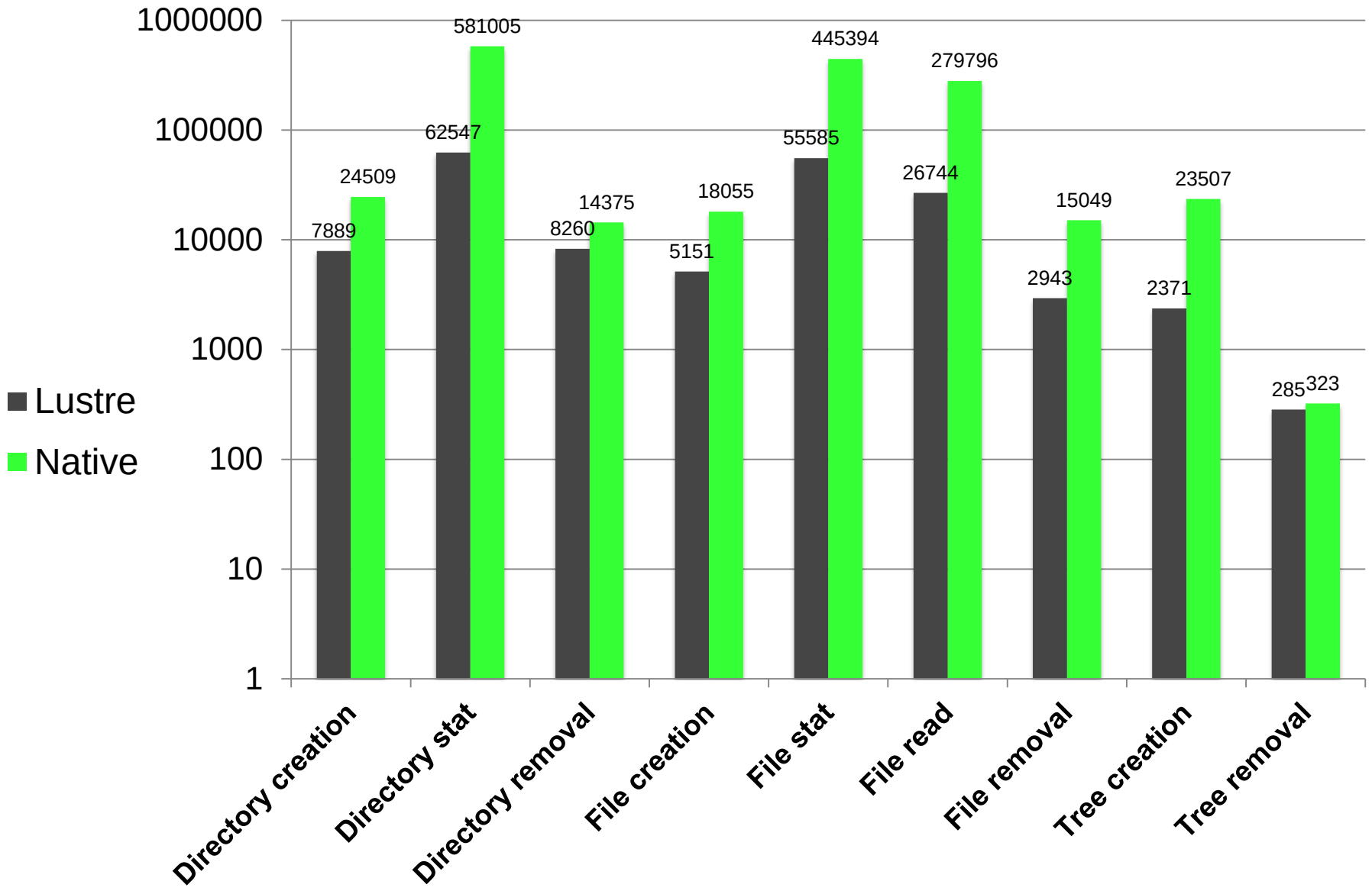
```
zfs set recordsize=4096 wombat-mdt1
zfs create wombat-mdt1/wombat-mdt1

mkfs.lustre --reformat --mdt --failnode=<NID of MDS0> \
 --mgsnode=<NID of MDS0>:<NID of MDS1> --fsname=wombat \
 --backfstype=zfs --index=1 wombat-mdt1/wombat-mdt1
```

mdtest (Lustre)
1152 tasks, 48 nodes, 1.8M files/directories
`mdtest -I 25 -z 5 -b 2`



mdtest (ZFS)
16 tasks, 1 nodes, 250k files/directories
`mdtest -I 250 -z 5 -b 2`




1000000
100000
10000
1000
100
10
1
Lustre
Native
7889
24509
62547
581005
8260
14375
5151
18055
55585
445394
26744
279796
2943
15049
2371
23507
285
323
Directory creation
Directory stat
Directory removal
File creation
File stat
File read
File removal
Tree creation
Tree removal

Lustre Stack Notes

Linux 3.10.65 kernel.org
SPL: GitHub master
ZFS: GitHub master and pull 2865

- `https://github.com/behlendorf/zfs/tree/largeblock`

Lustre: master (~v2.6.92) and the following patches:

- LU-4820 osd: drop memcpy in zfs osd
- LU-5278 echo: request pages in batches
- LU-6038 osd-zfs: Avoid redefining KM_SLEEP
- LU-6038 osd-zfs: sa_spill_alloc()/sa_spill_free() compat
- LU-6152 osd-zfs: ZFS large block compat
- LU-6155 osd-zfs: dbuf_hold_impl() called without the lock

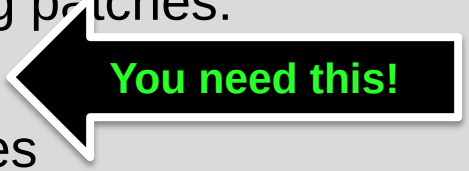
Lustre Stack Notes

Linux 3.10.65 kernel.org
SPL: GitHub master
ZFS: GitHub master and pull 2865

- `https://github.com/behlendorf/zfs/tree/largeblock`

Lustre: master (~v2.6.92) and the following patches:

- LU-4820 osd: drop memcpy in zfs osd
- LU-5278 echo: request pages in batches
- LU-6038 osd-zfs: Avoid redefining KM_SLEEP
- LU-6038 osd-zfs: sa_spill_alloc()/sa_spill_free() compat
- LU-6152 osd-zfs: ZFS large block compat
- LU-6155 osd-zfs: dbuf_hold_impl() called without the lock



This work supported by the National Science Foundation, award ACI-1341698.