Lustre & ZFS Go to Hollywood
Lustre User Group 2013

Josh Judd, CTO

Q2-2013

Case Study: Lustre+ZFS in M/E

- Media/Entertainment workflow for F/X
- Customer information anonymized
  - Do not have customer permission to go into more detail
- How Lustre provides superior solution
- How Lustre is combined with ZFS in production

ZFS+Lustre: Open Storage Layers

- ZFS:
  - Volume management layer (RAID)
  - Reliable storage (checksums)
  - Feature rich (snap, compression, replication, etc.)
  - Accelerators (HDD + SSD + NVRAM hybrid)
- Lustre:
  - Can sit on top of ZFS
  - Linux-centric scale-out filesystem, but...
  - Can support other OSs with some help

WARP MECHANICS®

M/E doesn't just use Linux

- Of course, NFS/SMB are options
- But how about...
- Native Linux/Lustre layer running KVM hypervisor
- Definitely works for Windows clients
- Also *works* for Mac OS *technically*, but...
- Legal issues with that approach put Apple back into SMB

WARP MECHANICS®

What does ZFS do for M/E?

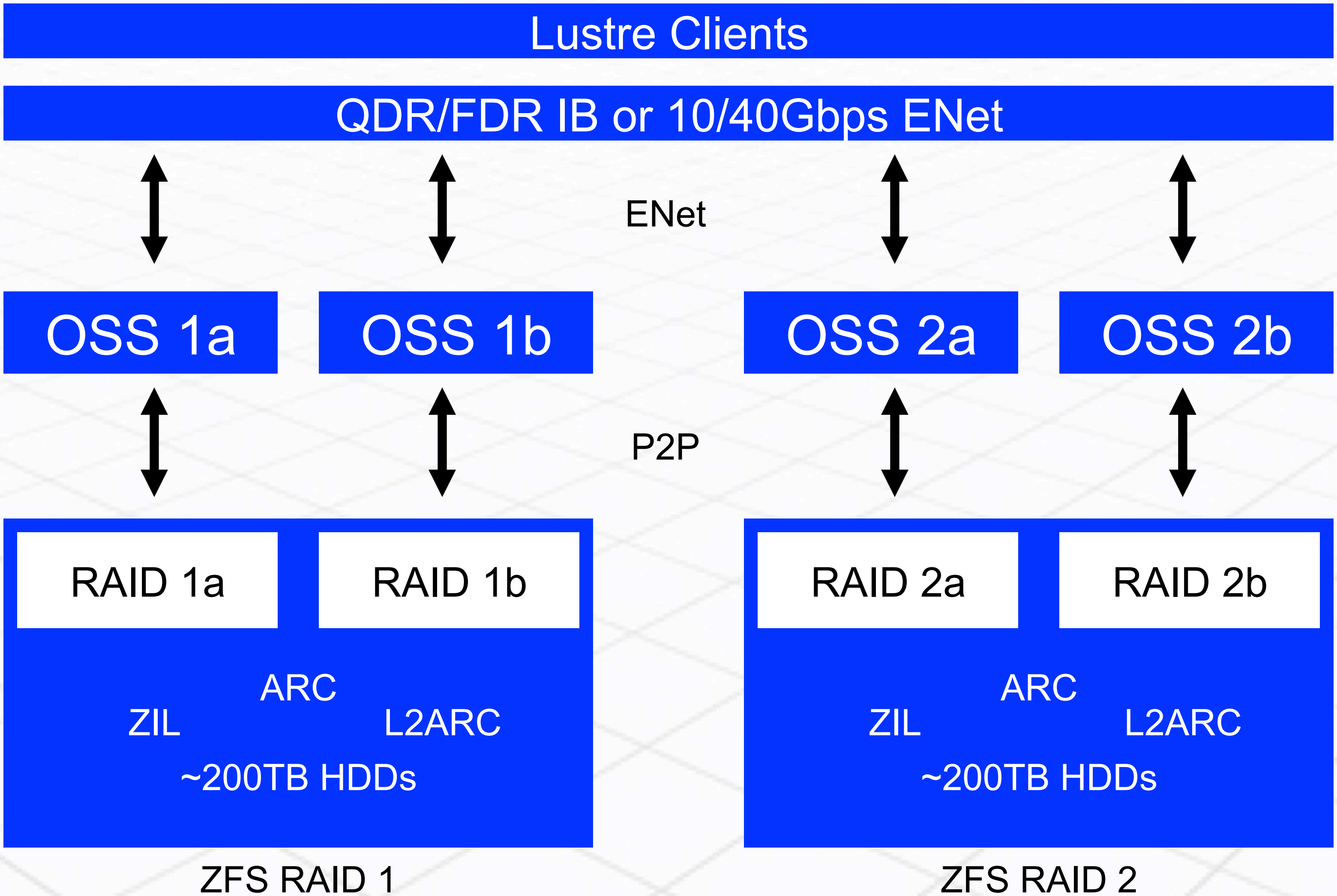
- M/E relevant features:
  - Multi-layer cache combines DRAM, NVRAM, SSDs, & HDDs
  - Copy-on-write eliminates holes and accelerates writes
  - Checksums eliminate silent data corruption and bit rot
  - Snap, thin provisioning, compression, de-dupe, etc. built in
- Same software/hardware supports NAS and RAID
  - One management code base to control all storage platforms
- Open storage software lowers costs

WARP MECHANICS®

How do they combine?

- ZFS is supportable on Solaris. Lustre is supportable on Linux. So... How do they mix?
- In theory, three ways:
  - Port Lustre to Solaris – not so much
  - Port ZFS to Linux –replace MD/RAID and EXT4
  - Use ZFS on Solaris as a RAID controller under Lustre on Linux
- WARP Mechanics focuses on the third option
  - Is supportable in production now
  - Maintains full separation of code
  - Still allows ZFS to replace EXT4 down the road, while performing volume management on separate controllers – which aids performance and scalability

Architecture for M/E Lustre



Example Architecture (cont.)

16 ZFS-based WARP*raid* systems =

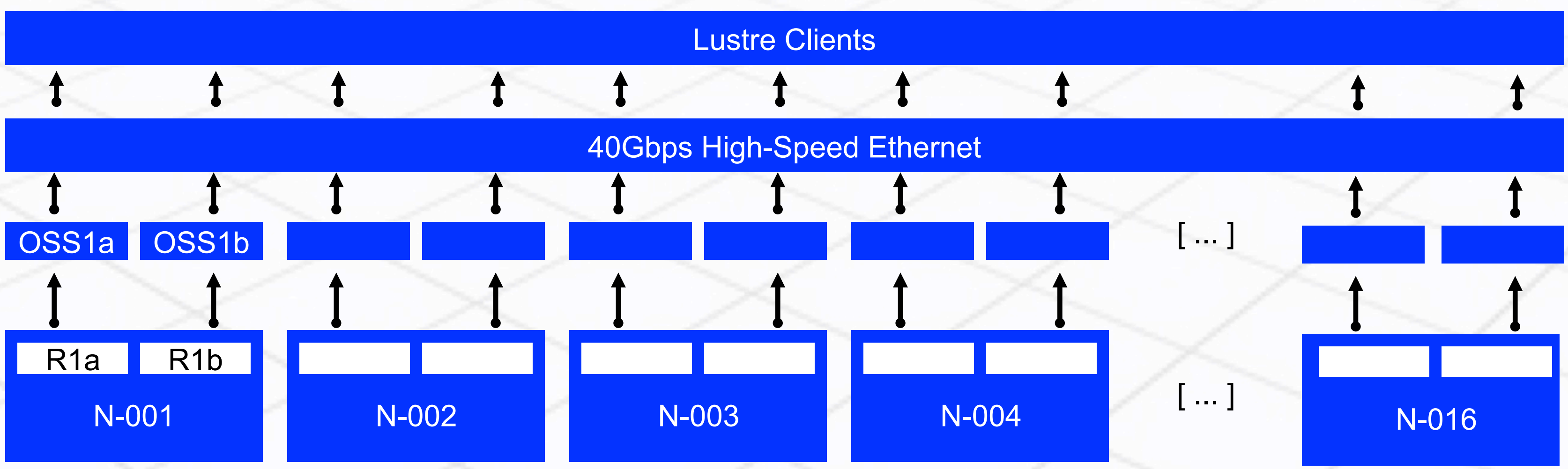
32 controllers = ~256 GBytes/sec

800x NL-SAS HDDs = ~3PB storage

80x NVRAM write accelerator modules for ZIL

160 TB "working data set" on SSD via L2ARC

SSDs in OSSs can bring cache up to 700TB+

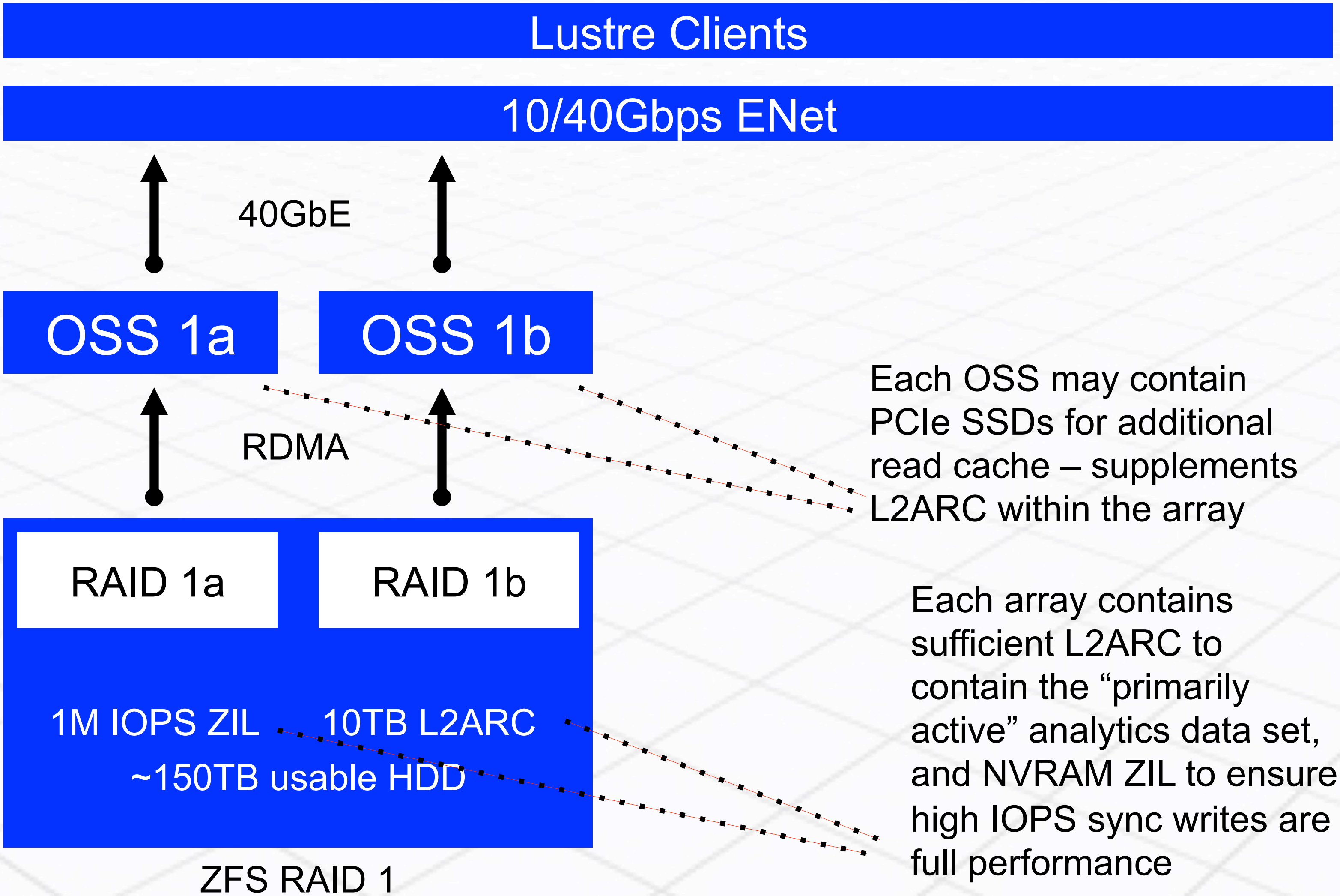


WARP MECHANICS®

Analysis of M/E Use Case

- Media workflow challenges
  - Continual ingest of large files at high speed
  - Simultaneous read of *current working set* from many workers
  - Working set has random high speed IO requirement
  - Each project is 10s of TB and growing
    - E.g., 54MB/frame * 24fps = 1.2GB/s
    - 1.2 * 60 * 120 = ~8TB/movie *before* F/X etc.
    - Then move to higher def, 60fps, and 3D...
    - ~200MB/frame * 60 * 2 * 60 * 120 = ~<u>175TB</u>/movie
- Lustre over ZFS with NVRAM solves high speed writes
- But what about reads?

Analysis of Use Case (cont.)



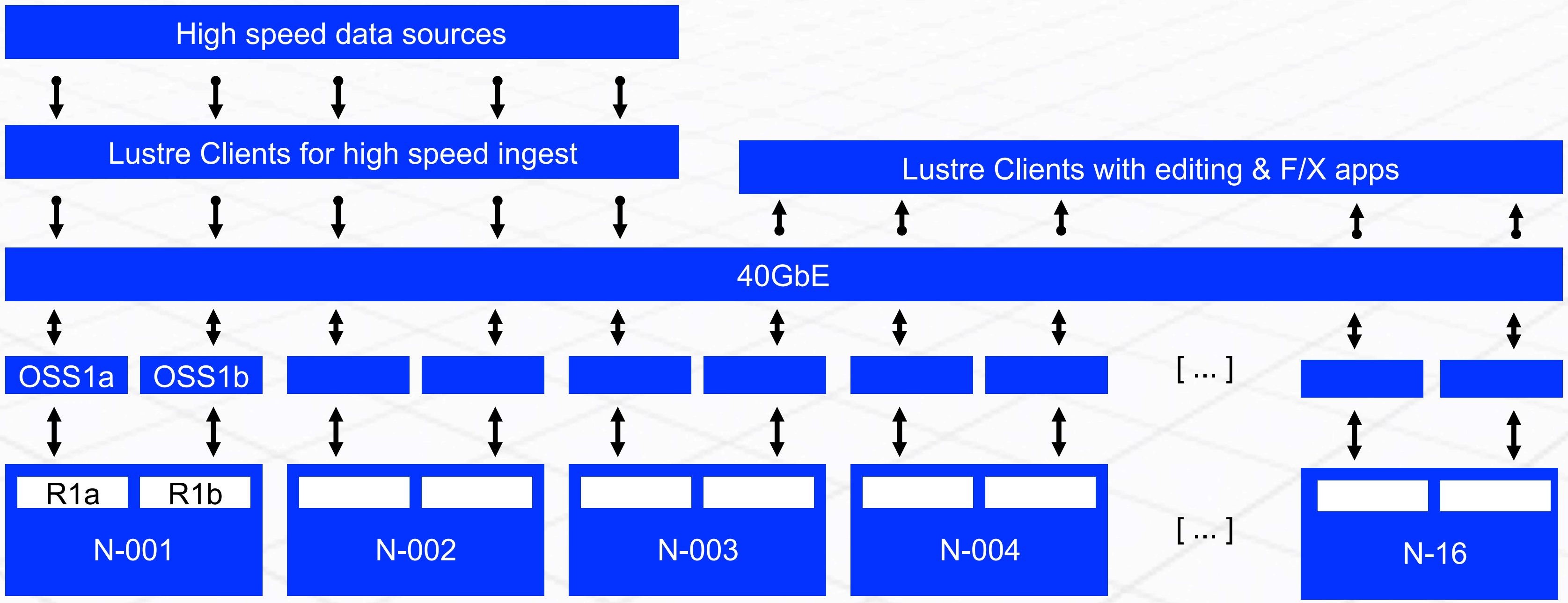
WARP MECHANICS®

Analysis of Use Case (cont.)

- Lustre is generally not very good at random reads
  - Especially if the same FS is still being hit with writes
  - Recipe for maximum head contention
- Each WARP*raid* array can have up to 10% of its usable capacity staged on SSD via ZFS L2ARC feature
- By adjusting ZFS kernel params to change fill rate, the SSDs can always have the active data set ready to go

WARP MECHANICS®

Analysis of Use Case (cont.)



Each OST has an amount of L2ARC scaled to keep the active data automatically on high speed SSDs. As any given object is read less often it automatically ages off but is already on HDD.

How does this make movies better?

- Non-ZFS arrays with similar features are *massively* more expensive – open storage software lowers costs
- Non-ZFS arrays *without* similar features are orders of magnitude slower for reads of active data sets
- Combining Lustre with ZFS allows much lower cost at both the array and scale-out layers
- The saved software budget can be applied to SSD and NVRAM, thus *vastly* accelerating workflow

WARP MECHANICS®

Q&A