ZFS and Lustre at LLNL

Presented to Joint Facilities User Forum on Data-Intensive Computing

June 18, 2014

Richard Hedges

Lawrence Livermore National Laboratory

LLNL-PRES-655826

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Today’s topics

- Motivation of LLNL ZFS Strategy
- ZFS features and advantages
- Implementation efforts
- ZFS based Lustre expectations and experiences

Motivation of ZFS Strategy

- Existing Situation for Lustre:
  - previous servers use ext4 (ldiskfs)
  - Random writes bound by disk IOPS rate, not disk bandwidth
  - OST size limits a limitation
  - fsck time is unacceptable
  - Expensive hardware required to make disks “reliable”

- Late 2011 requirement (Sequoia)
  - 50PB, 512 gigabytes/sec to 1 terabyte/sec
  - At a price we can afford

ZFS Features and benefits

- Copy on Write (COW) serializes random writes
  - Performance no longer bounded by drive IOPS
- Single volume size limit of 16 exabytes
- Zero fsck time. On-line data integrity and error handling
- Expensive RAID controllers are unnecessary
- Data is always checksummed and self repairing to avoid silent corruption.
- Easy aggregation of multiple devices in to a single OST.
- A 256 zettabytes (2^78 bytes) OST size limit enables larger servers.
- Snapshot the Lustre file system prior to maintenance for worry free updates.
- Transparent compression increases your total usable capacity.

Implementation efforts

- Began in 2007 by Bryan Bellendorf
- Using ZFS with Lustre was a major undertaking. Native kernel support for ZFS on Linux was not available, so LLNL undertook the significant effort required to make that a reality.
- The Lustre code itself had grown tightly coupled to ldiskfs, and another significant programming effort was funded by LLNL to create an abstracted Object Storage Device (OSD) in Lustre.
- This allowed Lustre to to interact with any file system for which an OSD layer is created, and allowed Lustre to intially support both ldiskfs and ZFS.
- Lustre support for ZFS first appeared in Lustre version 2.4.0, released in May of 2013

ZFS and Lustre Expectations

- New performance issues were anticipated in general
- The BG/Q system was different enough from previous generations, due to the system software, if nothing else.
- the Lustre client was all new code.
- ZFS as a back end file system for storage and the MDS would have an unknown (to us) performance profile.
- We changed storage hardware vendors.

ZFS and Lustre Experiences

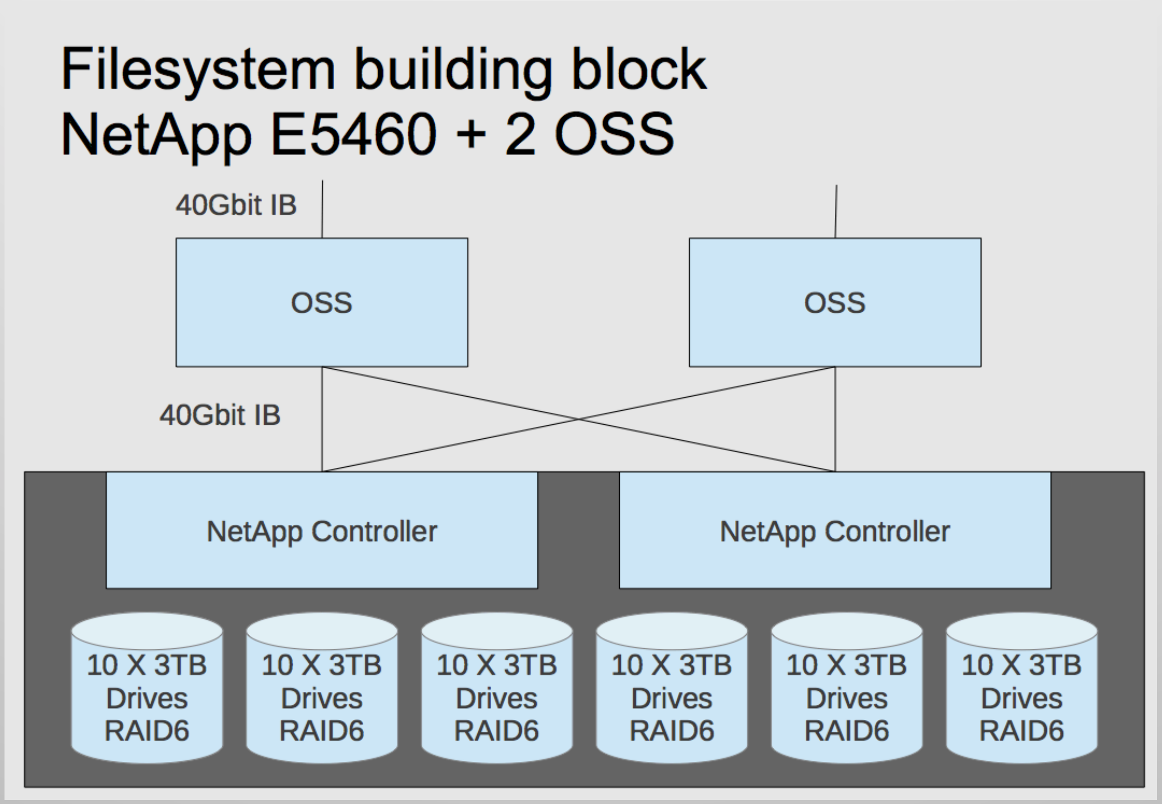
- Most BG specific Lustre development was repeated due to new system software and Lustre client
- Our benchmarks missed some factors that were relevant in production
- Interactive responsiveness initially not up to previous
- Very different workload on OSTs vs. MDS initially troublesome w ZFS
- Hardware reliability dramatically improved

Questions/ Discussion?

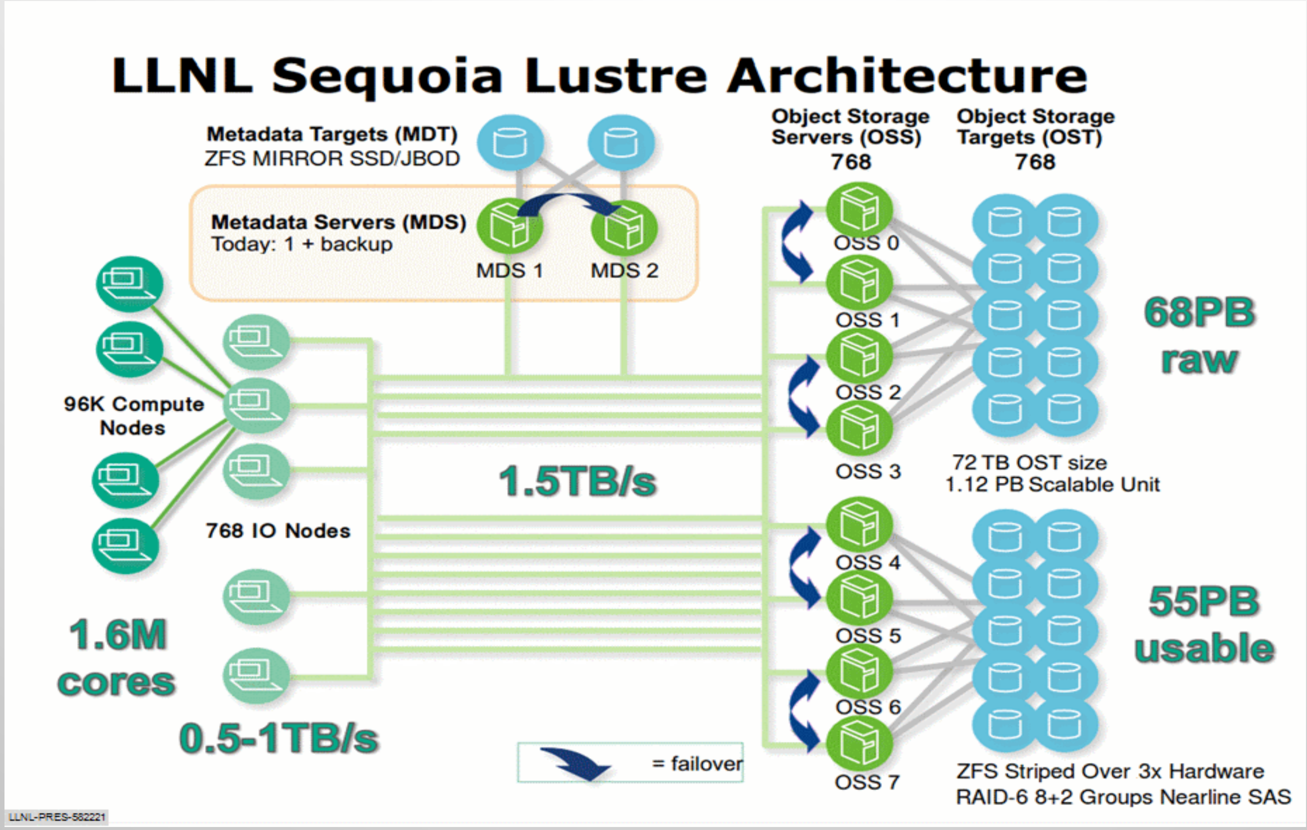
Some Configuration Details

- The OSS design consists of:
- Appro GreenBlade dual socket board
- Intel Xeon 8 core processors
- 64 gigabytes of RAM
- QDR Mellanox ConnectX-3 IB (LNET to Lustre)
- DualPort QDR ConnectX-2 (to disks).
- This hardware is configured into a file system building block consisting of a NetApp E5460 and 2 OSS nodes. The file system building block incorporates 2 OSS nodes, 2 Netapp controllers and 6 RAID6 sets consisting of ten 3 terabyte drives

Block diagram representing the hardware configuration of an OSS pair in the instantiation of the ZFS based Lustre file system for Sequoia at LLNL.



Eight of these are integrated into a rack (RSSU = rack scalable storage unit) and then 48 of them make up the resulting Lustre file system



55 petabyte storage
850 gigabytes/sec measured sustained write throughput
768 OSSs & OSTs
Each OST is 72 terabytes