Exploring Future Storage Options for ATLAS at the BNL/SDCC facility

Qiulan Huang, Vincent Garonne, Robert Hancock, Douglas Benjamin, Carlos Gamboa, Shigeki Misawa, Zhenping Liu

Scientific Data and Computing Center (SDCC)/BNL

Brookhaven National Laboratory

U.S. DEPARTMENT OF ENERGY

@BrookhavenLab

CHEP 2023, Norfolk, Virginia - May 08 - May 12, 2023

Motivation & Challenges

- Storage "Ecosystem" has evolved over the years
  - New/changes in protocol and storage software in WLCG
    - e.g., dCache, XRootD, EOS, Lustre, Ceph, MinIO
  - New data protection schemes (e.g., distributed RAID, erasure coding)
  - Hardware capabilities have increased
    - Network bandwidth
    - Server capability
    - HDD bandwidth/capacity
  - ATLAS Storage Environment and requirements have changed
    - Migration to new transfer protocols(~~GRIDFTP~~, WebDAV/XRootD), , storage tokens, …
    - *ATLAS storage requirement: Space token, ADLER32, TPC Pull, ...*
- BNL provides large scale storage service for large projects: **ATLAS**, Belle II, DUNE, sPHENIX, STAR, NSLS-II, etc
  - Disk storage: **151.2PB** (~87.2 PB dCache, 64.12PB Lustre, GPFS, NFS NetAPP)
  - Tape storage: **~221.5PB** HPSS

An opportunity to revisit current implementation in view of forthcoming requirements for HL-LHC

Brookhaven National Laboratory

Storage Components: Evaluation

The complete storage system may be implemented by one software package or a set of software packages working in concert

1. Access Layer Frontend
Client access protocol support

2. Unified Storage System Layer
Organizes the storage blocks provided by the backend into a coherent and unified storage space for storing data

3. Backend Storage Layer
Creates the storage "blocks" (space) used by the storage system to store user data

Evaluated components

dCache | XRootD
(dCache is software that supports multiple access protocols,XRootD is both software and a protocol)

dCache | XRootD + Lustre

OS level: Linux Software RAID (MDRAID), OpenZFS
Software defined: **Ceph**, Lustre

dCache or XRootD are recommended storage technologies that meet the ATLAS requirements

Early studies showed that CEPH was not considered for ATLAS. The main reason (at that time) was the I/0 performance below US T1 requirements

Three tested configurations to evaluate the stacks

1. **dCache with ZFS pools**
2. **dCache with Lustre storage pools**
3. **XRootD with Lustre backend storage**

Brookhaven National Laboratory

Write/Read Stress Tests for TPC(Third Party Copy)

Source
- Production ATLAS dCache

TPC write →

Testbeds
- dCache w/ZFS pools
- dCache w/Lustre pools
- XRootD w/ Lustre backend storage

← TPC read

Destination
- Production storage

Goal: Saturate the different storage configurations and sustain the peak rates with production data

FTS(File Transfer Service)

- Controlled test with FTS used to simulate realistic load
- Bulk FTS transfers
- Files: 500K, Max active limit (FTS): 1200

Testbed(cf. slides 12,13 and 14) same hardware

➔ Large scale test **5 PB**

➔ Simultaneous test of two configurations

Lustre vs dCache: TPC-Write(per door)

Davs TPC	XRootD w/ Lustre	dCache w/ ZFS	dCache w/ Lustre
traffic per door	3.1GB/s per door	+2.0GB/s per door	+3.8 GB/s per door
CPU Usage	<10% per door	~40% per door	~68%
Success rate	>98.5%	>99.4%	>98%

- ➔ **IO traffic of XRootD w/ Lustre is ~1.5 times of dCache w/ ZFS**
- ➔ **Important difference in checksum calculations (see next slide)**

Thanks to XRootD team's help with Lustre(e.g., configurations, tpc, checksum)
Thanks dCache develop team's suggestions for tuning(e.g., HTTP encryption), the gap between XRootd/Lustre and dCache/ZFS reduced from ~2 times to ~1.5times

Checksum calculation in dCache and XRootd

- dCache calculates dynamically checksum as the file is received or written to disk
- XRootD calculates checksum after the file has been written to disk
  - File read from backend storage cause extra I/O traffic
  - Increase load on network and backend storage servers(CPU, disk, etc)
  - Needs more gateway and tunings to saturate the backend storage performance
- Observed errors during TPC-write tests(slide 6), most of which are checksum related issues
  - Checksum timeout: happen while there are bulk of active requests on FTS
  - HTTP 500 error: Can be fixed by increasing the maximum number of checksum calculations that may run at the same time

Error Description	XRootd w/Lustre	dCache w/ZFS	Comments
Recoverable error: [110] DESTINATION CHECKSUM timeout of 1800s	✗	✗	Checksum timeout on FTS side while there are bulk of active requests(e.g.,1200)
Recoverable error: [5] DESTINATION CHECKSUM HTTP 500 : Unexpected server error: 500	✓	✓	Fixed the error by Increasing maximum number for checksum calculations for XRootd **max>=512(According to tuning tests)**

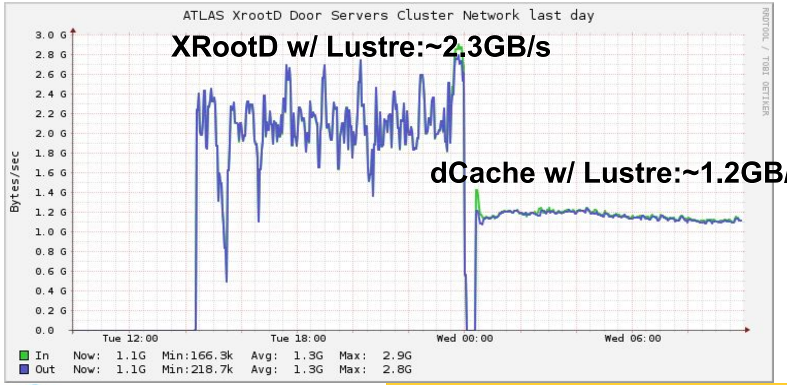
✗ **: Exist**
✓: **Fixed**

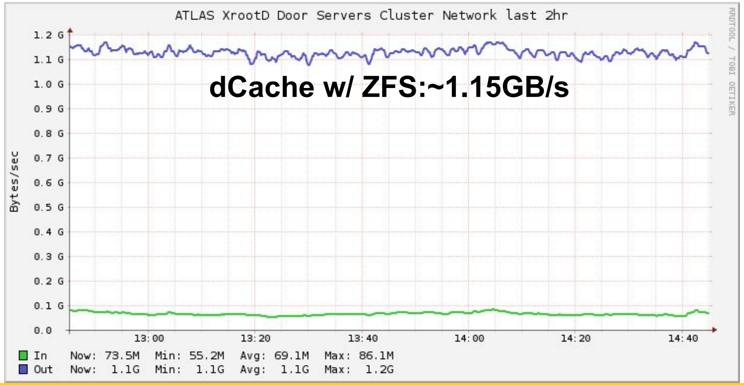
- **dCache checksum with dynamic calculates behaves better compared to XRootD**

Brookhaven National Laboratory

Lustre vs dCache: TPC-Read(per door)

Davs TPC	XRootD w/ Lustre	dCache w/ ZFS	dCache w/ Lustre
Aggregate traffic	~2.3GB/s	~1.15GB/s	~1.2GB/s
CPU Usage	<3% per door	<3% per door	<3% per door
Comments	1)XRootd+Lustre gets best read performance, about 50% higher than dCache+ZFS and dCache+Lustre pools. 2) dCache with ZFS and Lustre pools perform about the same.		





Brookhaven National Laboratory

The results was reported to dCache team. They mentioned a potential issue, e.g., the remote transfer manager and RemoteHttpDataTransferProtocol

Backend Storage evaluation: OS Level

LINUX MDRAID		OpenZFS
RAID-6 LUN	→	Single RAIDz2 vdev Zpool
No equivalent	→	Single RAIDz3 vdev Zpool
Striped RAID-N LUN	→	Multi-vdev Zpool
No equivalent	→	dRAID “distributed” RAID

MDRAID advantages over OpenZFS

- Supported by Redhat
- Faster rebuild on very full LUNs (compared to ZFS RAIDzN)
- No performance penalty for > 85% capacity usage
- Less capacity overhead for similar configuration(**cf. Slide 15**)

OpenZFS advantages over MDRAID

- Better data integrity（block checksum, auto healing corrupt data）
- Better IO performance in sequential read/write**(cf.slide 16,17)**
- Separate filesystems in same Zpool can be tuned to data access patterns Automatic load balancing across LUNs
- Built in hot file cache (ARC) in memory
- (future) dRAID can significantly lower rebuild times to reduce risk of disk failures
- Reduced manual intervention

➔ **OpenZFS has been chosen to work as backend storage for the new hardware of ATLAS dCache**

Brookhaven National Laboratory

Summary

What we learned

- Gained expertise with alternate storage options
  - All alternate configurations provide the ATLAS needed functionalities
  - XrootD Lustre vs. dCache Lustre vs. dCache ZFS
- Evaluated the performance of dCache and XRootD with alternate options
  - XRootD + Lustre can show better I/O performance than dCache+ZFS for third party copy

What we choose

- We have chosen the **dCache ZFS** configuration for medium term
- ZFS gives reliability with low operation cost
- XRootd+Lustre performs better (for TPC) but missing important operation experience
  - WLCG T1 sites needs 99% of availability
- Latest dCache or forthcoming might give improvement (thanks to dCache developers and their good support)

Next step

- Further validation for various production workflows is required
- Convergence toward a tiering storage strategy at a data center for different workflows
  - E.g., Fast I/O disk for analysis with dCache as data management / tiering layer
- Lustre is still a possible candidate for long term (not Run 3) as we are gaining operation experience with NSLS, sPHENIX and ATLAS
- **To be continued…**

Brookhaven National Laboratory

Thank you!

Backup

Test Hardware for Storage

10 Servers with identical HW specifications

- 5 Servers configured as Lustre OSS servers
- 5 Servers configured as dCache pool servers

Server HW specifications

- 384GB RAM, 36 cores (18 cores/CPU)
- Network - 2 x 25 Gbps = 50Gbps
- One JBOD per server
  a. 102 x 14TB drives
  b. ~1 PB available

Lustre Disk Organization

- 10 x (8+2) RAID 6 LUNs
- One LUN one OST

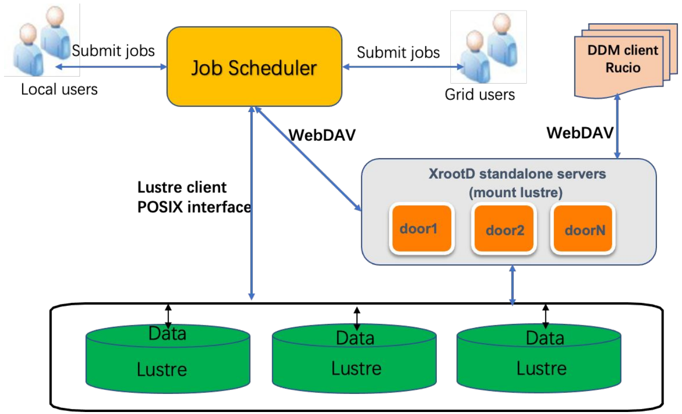
dCache Disk Organization

- Single ZFS zpool (14X7)
- 7 vdevs per zpool
- Each vdev configured as 14 disk RAIDz2

Testbed: XRootD+Lustre Deployment

XRootD+Lustre

- Lustre MDS - Lustre v2.12.8
  - One VM - 1TB **HDD**, 16 cores, 64GB RAM
- Single Lustre file system constructed from 5 OSS servers
- 5 standalone XRootD servers
  - Lustre filesystem accessed via standard Lustre kernel client module



Monitoring:

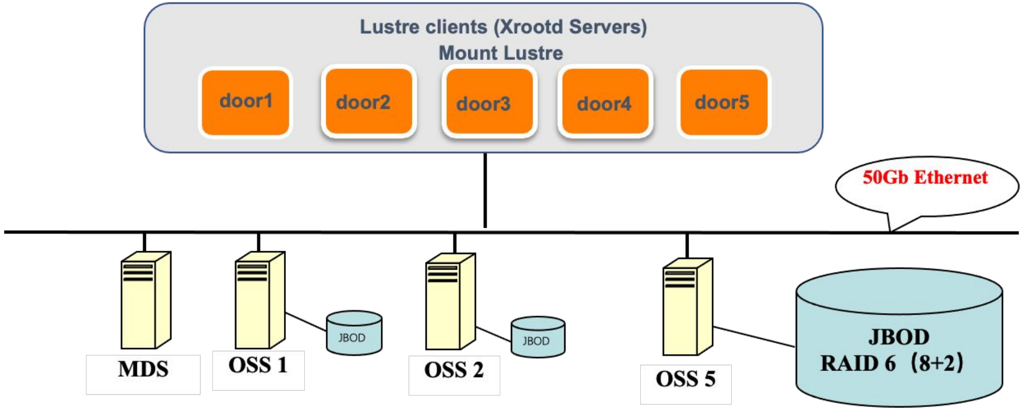
Grafana

Ganglia

Robinhood

Barreleye

Brookhaven National Laboratory

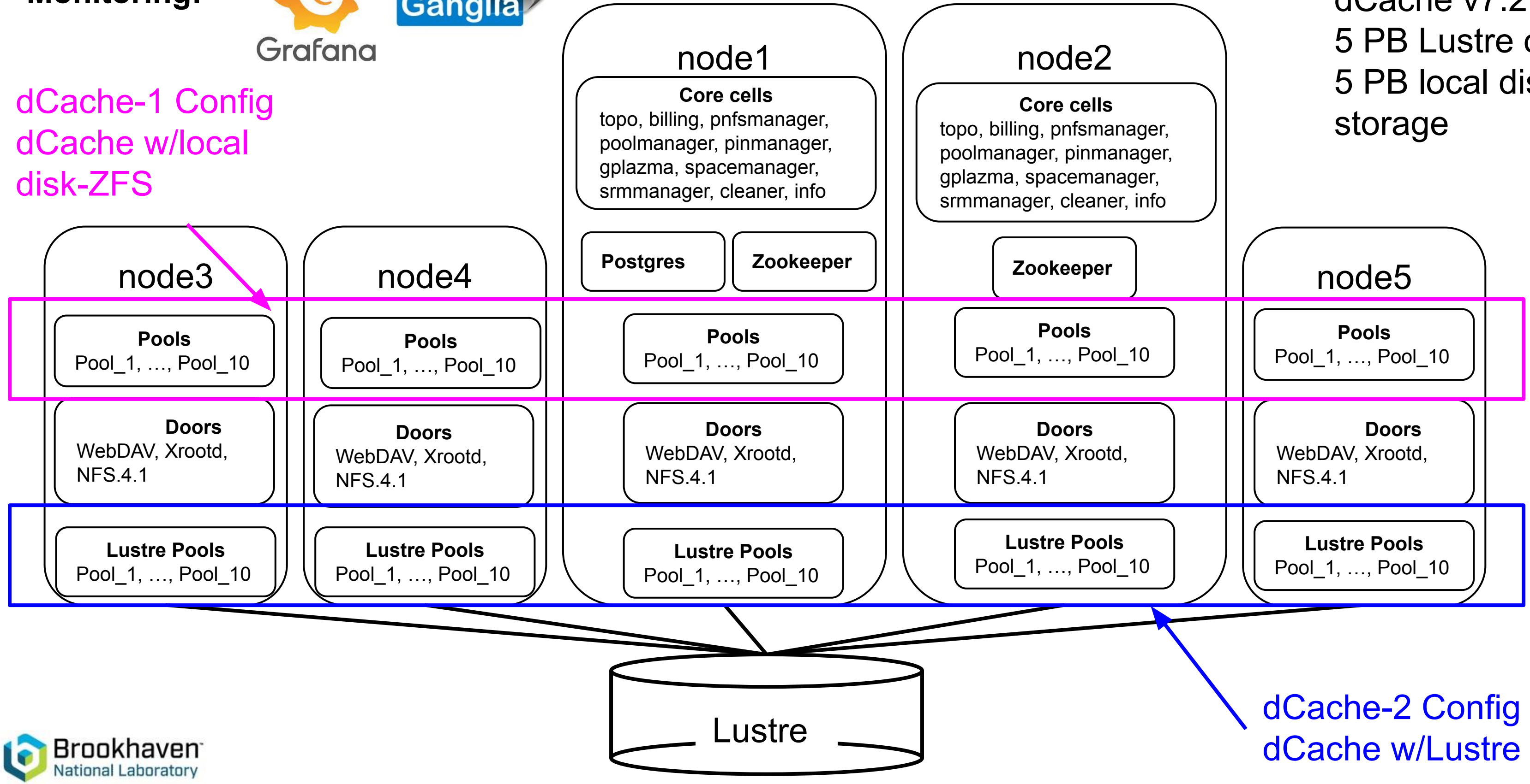


Testbed: dCache Deployment

Monitoring: Grafana, Ganglia

dCache-1 Config
dCache w/local
disk-ZFS

dCache v7.2.3
5 PB Lustre or
5 PB local disk
storage



dCache-2 Config
dCache w/Lustre

Brookhaven National Laboratory

Capacity Comparison (TiB)

Configurations for 100+ disk JBOD Chassis

Test Name	ZFS 20x5	ZFS 10x10	ZFS 14x7	MD RAID 20x5	MD RAID 10x10	MD RAID 14x7
Full Capacity (TiB)	1132	970	1024	1150	1020	1071
Overhead Factor	1.148	1.339	1.269	1.133	1.286	1.214

FIO Bandwidth comparison (GBytes / sec)

	ZFS/MD RAID Configuration (disks/LUN) x (# LUNs)					
Test Name	**ZFS 20x5**	**ZFS 10x10**	**ZFS 14x7**	**MD RAID 20x5**	**MD RAID 10x10**	**MD RAID 14x7**
Seq Read	10.339	9.610	9.119	5.230	8.031	6.862
Seq Write	3.969	3.837	3.874	2.719	4.480	3.789
64k Rand Write	0.233	0.226	0.228	0.175	0.393	0.239
64k Rand Read	0.528	0.686	0.772	1.609	3.181	2.740
8k Rand Write	0.029	0.028	0.028	0.026	0.057	0.041
8k Rand Read	0.300	0.247	0.208	0.540	0.539	0.544

FIO IOPS Comparison

	ZFS/MD RAID Configuration (disks/LUN) x (# LUNs)					
Test Name	**ZFS 20x5**	**ZFS 10x10**	**ZFS 14x7**	**MD RAID 20x5**	**MD RAID 10x10**	**MD RAID 14x7**
Seq Read	10586	9840.9	9337.5	5353.7	8224.1	7026.3
Seq Write	4064.1	3929.2	3966.7	2784.6	4587.9	3879.6
64k Rand Write	3819.4	3697.8	3738.7	2861.1	6436.9	3921.6
64k Rand Read	8648.1	11242	12651	26363	52115	44899
8k Rand Write	3838.7	3689.1	3735.1	3350.5	7497.7	5312.8
8k Rand Read	39383	32326	27198	70744	70685	71343

Brookhaven National Laboratory

Some additional discussion of differences for random reads might be found in the Arstechnica article ZFS versus RAID: Eight Ironwolf disks, two filesystems, one winner | Ars Technica