Storage options and strategy at the US ATLAS Tier 1 Facility

Qiulan Huang <qhuang@bnl.gov>, Vincent Garonne <vincent.garonne@bnl.gov>
Shigeki Misawa <misawa@bnl.gov>, Tejas Rao <raot@bnl.gov>
Robert Hancock <hancock@bnl.gov>, Doug Benjamin <dbenjamin@bnl.gov>

Scientific Data and Computing Center (SDCC)

Brookhaven™ National Laboratory

U.S. DEPARTMENT OF ENERGY

@BrookhavenLab

ATLAS Pre-Scrubbing Review - June 27, 2022

Motivation

Steady increase in ATLAS needs for DISK** storage capacity

- Projected reductions in DISK cost/TB not keeping up with needs requiring ATLAS to reduce disk storage requirements
- No budget to continue with dCache and two DISK copies (timeline: FY23)

**** TAPE is no concerned**

Storage "Ecosystem" have changed over the years

- Changes in access software (e.g., dCache, XRootD, EOS, ~~DPM~~)
- New storage software stacks (e.g., Ceph, Lustre, MinIO, …)
- New data protection schemes (e.g., distributed RAID, erasure coding)
- Hardware capabilities have changed
  - Network bandwidth
  - Server capability
  - HDD bandwidth/capacity
  - SSD capacity/performance
  - Overall hardware reliability
- ATLAS Storage Environment has changed
  - Migration to new transfer protocols (WebDAV/XRootD), storage tokens, …

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

Brookhaven National Laboratory

US ATLAS Tier 1 Priorities

Ensure that storage services meet current and future requirements

- Proven performance and capacity scalability (to meet projected ATLAS requirements)
- Access protocol support
- Service availability and reliability
- Cost efficient system architecture and implementation
- Sustainable operational costs
- Long term viability of the storage system

Any changes to US ATLAS Tier 1 storage services need to be transparent to ATLAS operations

Critical Storage System Characteristics

- Mature software, with well worn code paths when run in the ATLAS operational environment
- Tolerant of component failures
  - In worst case, graceful degradation of service
    - Avoid service loss if possible
  - Automatic self healing
  - Minimize time spent in a degraded configuration
- Immunity to hot spots within a node and across nodes
- Multiple levels of data protection within the system
  - E.g. node level or disk level, depending on the requirements
- Efficient utilization of hardware resources
- Simple and transparent system. No unnecessary complexity
  - Operational simplicity
  - Simple software and hardware configuration and upgrades

Storage Components

1. Access Layer Frontend
Client access protocol support (e.g., WebDAV, XRootD, …)

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

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

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

Current vs HL-LHC Scale

	Current Tier 1 dCache	HL-LHC Tier 1 dCache
Access Layer Frontend	**dCache: doors (~20)** Peak write traffic: ~36GB/s Peak read traffic: ~28GB/s Peak deletion traffic: ~300Hz	**dCache: doors (~200)** Peak write traffic: ~360GB/s Peak read traffic: ~280GB/s Peak deletion traffic: ~3KHz
Unified Storage System Layer	**dCache: Pool Nodes(~60), services, DB, ..** Disk capacity: 54PB 170M name space objects	**dCache: Pools (~600), services, DB, ..** Disk capacity: 540PB 1.7B name space objects
Backend Storage Layer	**SAS JBOD storage**	**SAS JBOD storage**

Straight 10x scaling raises concerns over scalability of current system

Brookhaven National Laboratory

Storage Components: Concerns

Operational concerns

- “Stalled” doors
- Problematic NFS support

- Pool nodes - single point of failure
- Poor load balancing over pool nodes
- File replication only mechanism for higher level data protection

- Poor load balancing over pools (in one pool per RAID6 LUN configuration)
- Disk rebuild times continue to grow, increasing likelihood of data loss

Current Tier 1 dCache

dCache: doors (~20)
Peak write traffic: ~36GB/s
Peak read traffic: ~28GB/s
Peak deletion traffic: ~300Hz

dCache: Pools (~60), services, DB, ..
Disk capacity: 54PB
170M name space objects

SAS JBOD storage
Linux MD RAID
7 x RAID6 (12+2) LUN
One LUN per dCache Pool

With projected Run 4 Configuration (X10)
⇒ Concerns over scalability

Storage Components: Evaluation

	Evaluated components
1. Access Layer Frontend *Client access protocol support (e.g., WebDAV, XRootD, …)*	**dCache | XRootD**
2. Unified Storage System Layer *Organizes the storage blocks provided by the backend into a coherent and unified storage space for storing user data*	**dCache | XRootD + Lustre**
3. Backend Storage Layer *Creates the storage "blocks" (space) used by the storage system to store user data*	**OS level:** ● Linux Software RAID (MDRAID) ● OpenZFS **Software defined:** ● Ceph ● Lustre

Storage Components: Evaluation

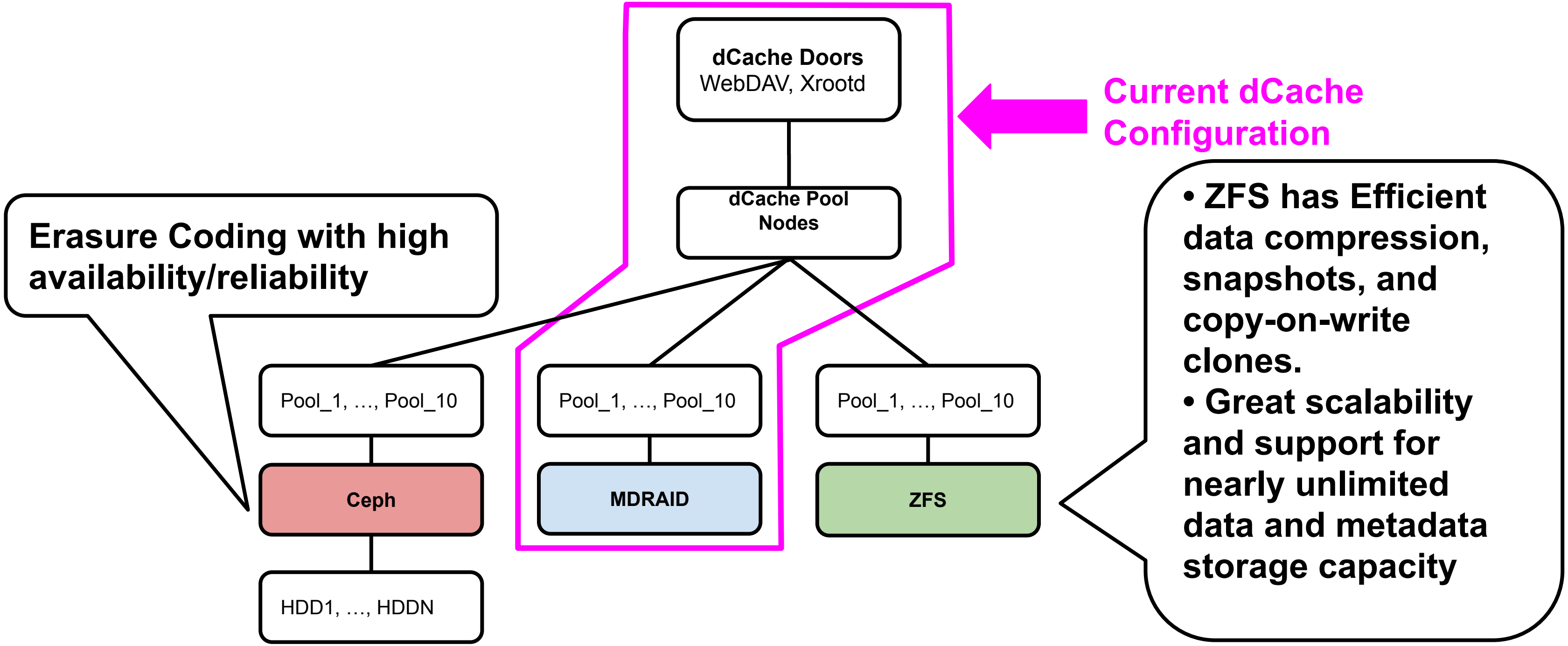
Evaluated components

1. Access Layer Frontend *Client access protocol support (e.g., WebDAV, XRootD, …)*	**dCache | X**
2. Unified Storage Syste *Organizes the storage block* *a coherent and uni*	**dCache** **XRootD**
rage Layer *Cre blocks” (space) used by the storage* *syst store user data*	**OS level** ● Linux Software RAID (MDRAID) ● OpenZFS **Software defined:** ● Ceph ● Lustre

Multiple combinations of unified Storage Layer and backend Storage are possible.

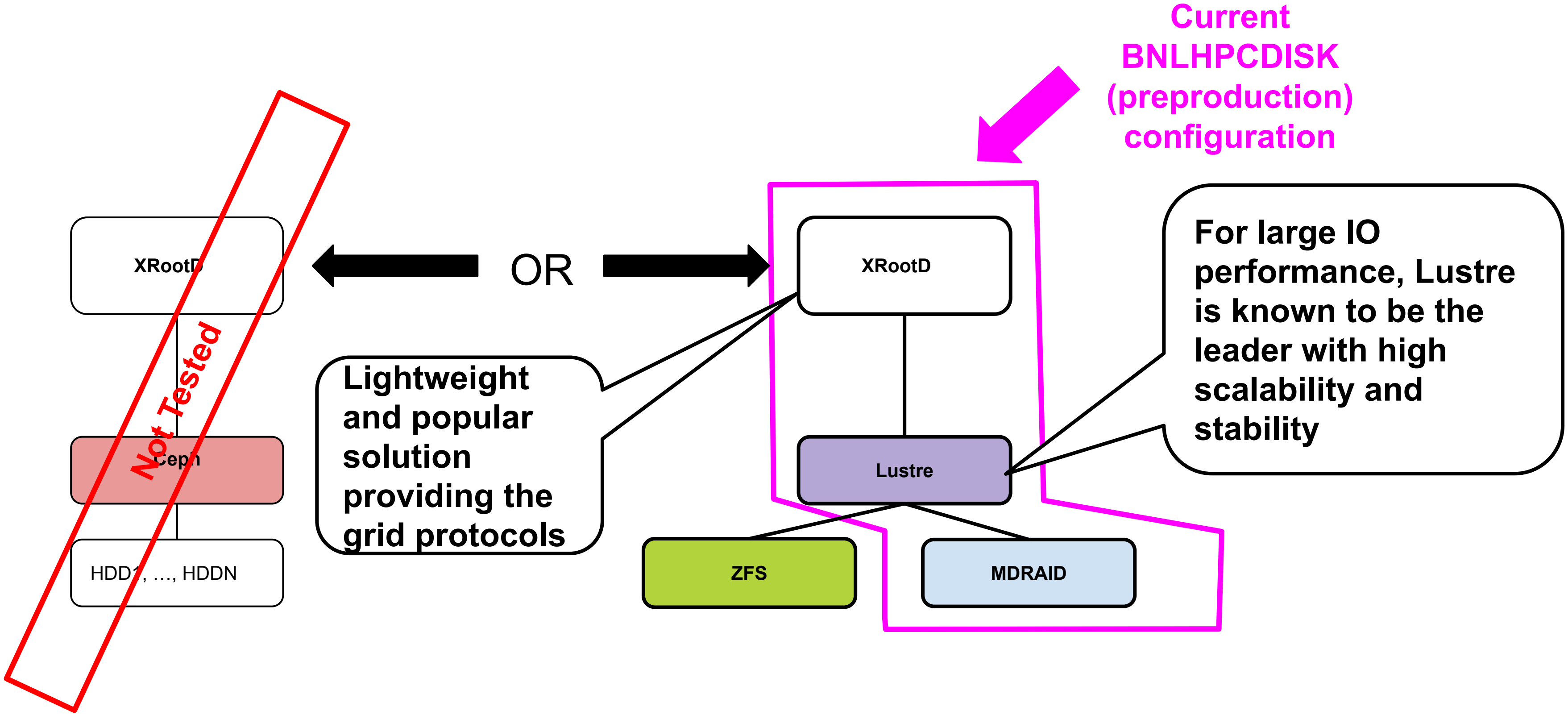
Brookhaven National Laboratory

Possible dCache Deployment



Brookhaven National Laboratory

Possible XRootD Deployment



Brookhaven National Laboratory

Backend Storage Layer : OS Level

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

Multi-vdev/Striped RAID-N/dRAID

- Better disk level load balancing relative to individual RAID-N/RAIDzN LUNs
- Larger data loss in case of vdev or RAID-N failure for multi-vdev/Striped RAID-N
- dRAID failure modes different from multi-vdev Zpool and striped RAID-N
    - Faster data rebuild compared to RAIDzN and RAID-N

OpenZFS vs MDRAID

OpenZFS advantages over MDRAID

- Better error detection/correction via block checksum(no bitrot / RAID 5 write hole issues)
- Variable stripe size with RAIDZn (but not with dRAID)
- ARC cache provides automatic high speed tier for hot workloads in RAM
- No spurious disk expulsion from RAID due to transient effects.
- Faster failed disk rebuild time with dRAID compared to RAID-N/RAIDzN
    - But relatively new, released in OpenZFS 2.1 (mid-2021)

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

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

Brookhaven National Laboratory

Reasons for switch to ZFS

- Better data integrity
  - Per block checksums verified every read
  - Auto healing corrupted data
- Separate filesystems in same Zpool can be tuned to data access patterns (metadata / large files)
- Automatic load balancing across LUNs
- Built in hot file cache (ARC) in memory
- (future) dRAID can significantly lower rebuild times to reduce risk of concurrent disk failures
- Reduced manual intervention
  - No manual intervention on reboot required
  - No false positive failed disks.
  - Less complicated / manual steps for disk replacement

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

ZFS FIO Bandwidth (GBytes / sec)

1MB ZFS recordsize (RS) used in previous tests are valid for large average file sizes. However, a large recordsize reduces performance for small request sizes, e.g. database. Effect of reducing recordsize from 1M to 64K on a 10x10 ZFS configuration are shown below:

	Seq Read (bs=1M)	**Seq Write (bs=1M)**	**64k rand read**	**64k rand write**	**8k rand read**	**8k rand write**
RS=1M	9.610	3.837	0.686	0.226	0.247	0.028
RS=64k	3.209	1.483	1.380	1.199	1.085	0.151

Backend Storage Layer: Ceph

Erasure Coding - Primary motivation for looking at Ceph

- Higher availability/reliability compared to single copy dCache with OS level backend disk (mdraid/zfs)
- Potentially lower $/TB compared to file replication in dCache

Three possible dCache pool configurations:

- Pool on XFS file system on Ceph “Rados Block Device” (RDB)
- Pool on CephFS file system
- Pool on Ceph S3/Swift object storage using librdb
    - With “pool.backend = ceph” dCache configuration

Brookhaven National Laboratory

ceph

Hardware requirements for a Ceph cluster

- 4GB/OSD memory is recommended for Bluestore backends. 8GB/OSD is advised for large datasets
    - This equates to 5120 OSD devices for 20PB usable storage assuming 12TB drives, so about 20TB of RAM is required. Approximate cost is $544K for 20TB of RAM @ $26.5/GB.

- 128GB RAM is recommended for Ceph monitors and manager nodes. Three dedicated monitor/manager nodes are recommended.

- DB/WAL storage block devices should be on dedicated SSD drives and capacity should be at least 4% of block device.
    - This equates to 1040TB of storage capacity on dedicated SSD drives for 20PB usable storage.

        Assuming Dell 3.84TB SAS mixed use SSD is $4310, this will be $1.1M.
        The $1.1M cost is based on 20PB of usable storage with (8+3) erasure coding and no replication for DB/WAL SSD devices.

Ceph — Cost

- The hardware requirements for Ceph are very high which leads to higher net cost compared to Lustre
  - Higher memory/CPU requirements on the Ceph OSD nodes and 4% of raw capacity on SSDs
  - For XFS/ZFS, RAID6 is a viable option even with higher capacity drives.
- For usable capacity of 20PB disk storage
  - Ceph would need $544K for RAM and $1.1M for DB/WAL storage, for a total of about $1.6M .
  - XFS/ZFS would need $68K for RAM (128GB X 20 nodes @ $26.5/GB) and $4K for metadata storage.

- Just for metadata storage and RAM, Ceph is 20X more.

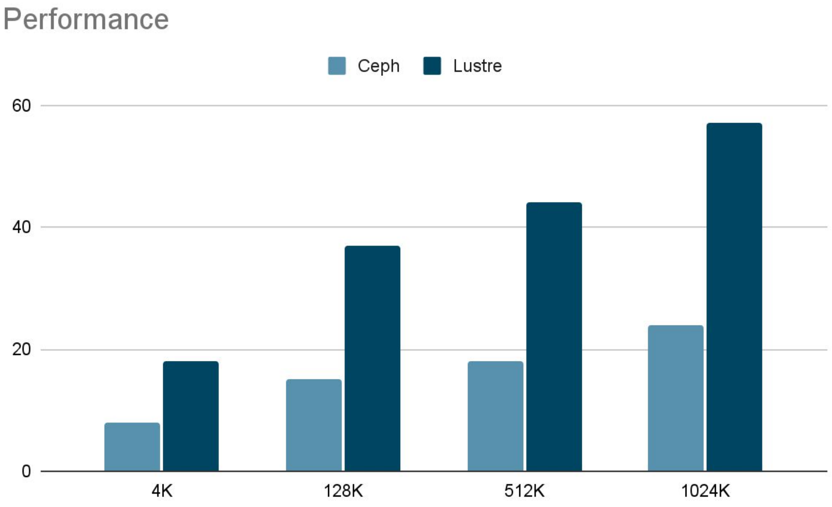
Ceph — Performance

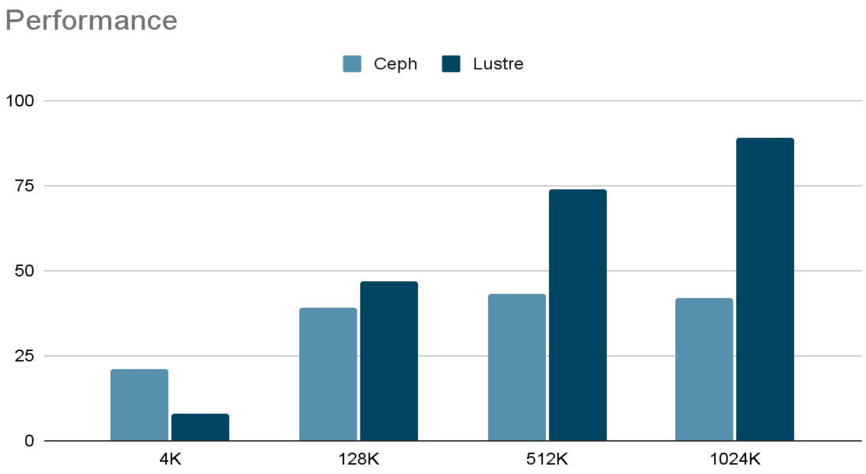
- Performance for the Ceph kernel block device and CephFS as a backend filesystem was only 40% of hardware capabilities for large sequential IO.
  - For 60 drives, streaming performance was around 2780MB/s for reads and 1100MB/s for writes.
  - Significantly higher fraction of HW performance is achieved with Lustre on same HW
- Poor performance with Ceph when data is on flash media
  - At best 25% of hardware capabilities for sequential workloads.

Brookhaven National Laboratory

Ceph — Performance

- Dedicated network links are required for replication and recovery traffic and average performance during major recovery/scrub periods is poor.
- Recovery and scrubbing can potentially impact service availability for large periods of time while WLCG requires 99% uptime by MOU/SLAs.





Sequential Read/write performance of Ceph and Lustre on same hardware

Ceph — Other factors

- **Ease of management.**

  Ceph is not trivial to setup. One has to be very careful with how crush maps and cache tiering is configured to get it to work correctly otherwise performance would be impacted and data would not be distributed evenly. Troubleshooting is also not straightforward as most ceph services run in a containerized environment.

- **Enterprise support**

  Ceph licensing model for enterprise support is capacity based and is prohibitively expensive. List pricing is $160K/PB/year for Redhat Ceph enterprise support. Lustre, on the other hand, has a site-wide license option based on the number of incidents and is not capacity-based.

Ceph — Conclusion

- **Cost**
  Hardware requirements for DB/WAL storage and RAM is very high compared to other storage solutions.

- **Enterprise support**
  Enterprise support for Ceph is prohibitively expensive.

- **Performance**
  Streaming sequential performance is lower at 40% of hardware capabilities.

- **Ease of management.**
  Troubleshooting and management of Ceph services could be complex as services run in a containerized environment.

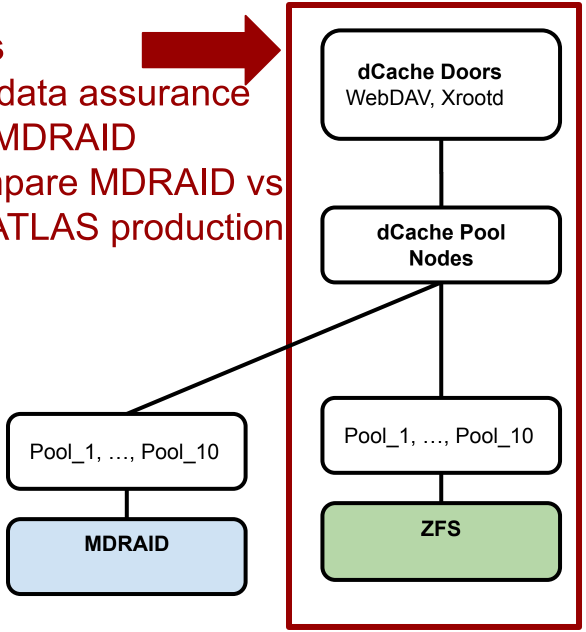
Ceph was not adopted as a long term solution for ATLAS

Brookhaven National Laboratory

Evaluated Storage Software Stacks

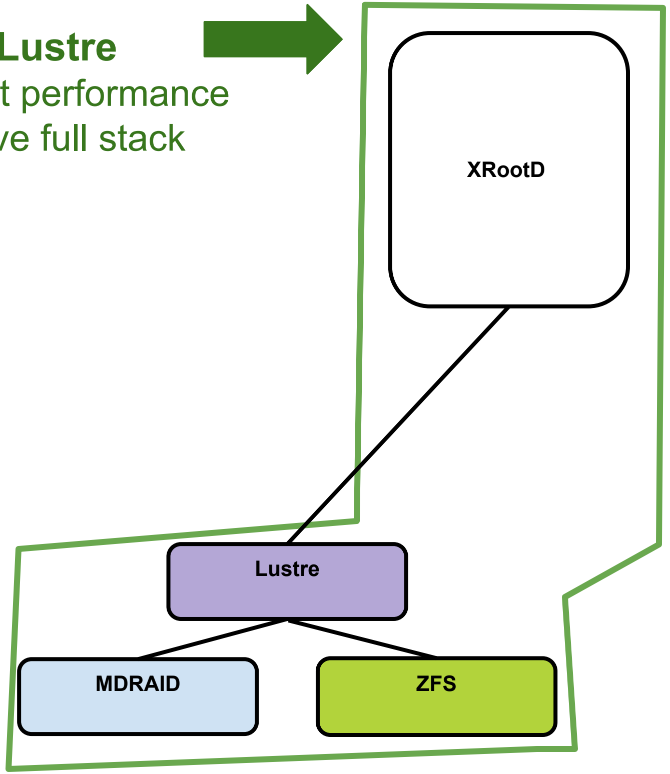
dCache + zfs disks

- Provides additional data assurance relative to dCache + MDRAID
- Can be used to compare MDRAID vs ZFS performance in ATLAS production environment



XrootD + Lustre

- Ability test performance of alternative full stack



Testbed: Hardware

10 Servers with identical HW specifications

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

Allows “apples to apples” comparison between Lustre and dCache

Server HW specifications

- 384GB RAM, 36 cores (18 cores/CPU)
- Network - 2 x 25 Gbps = 50Gbps
- One JBOD per server

Lustre Disk Organization

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

dCache Disk Organization

- Single ZFS zpool
- 5 vdevs per zpool
- Each vdev configured as 20 disk RAIDz2

Testbed: XRootD+Lustre Deployment

XRootD+Lustre

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

Monitoring:

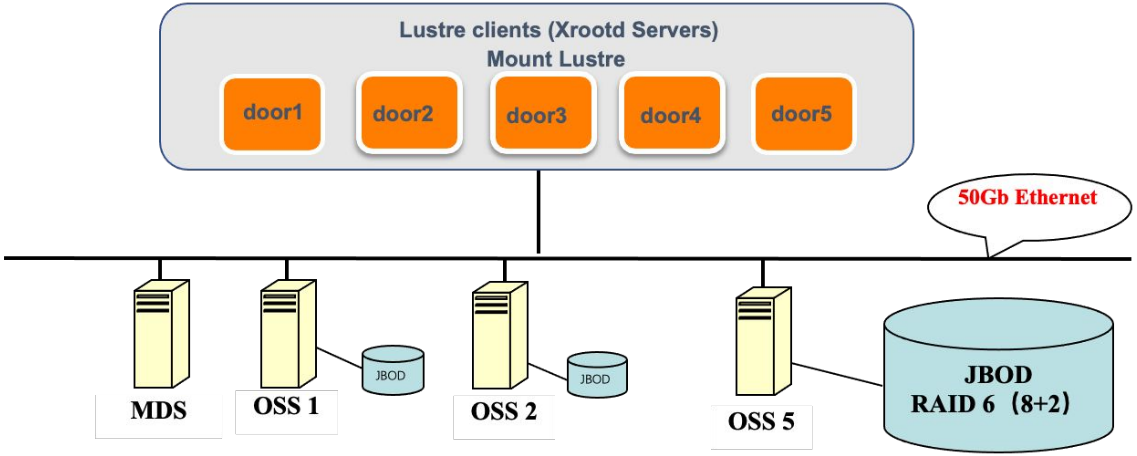
Grafana

Ganglia

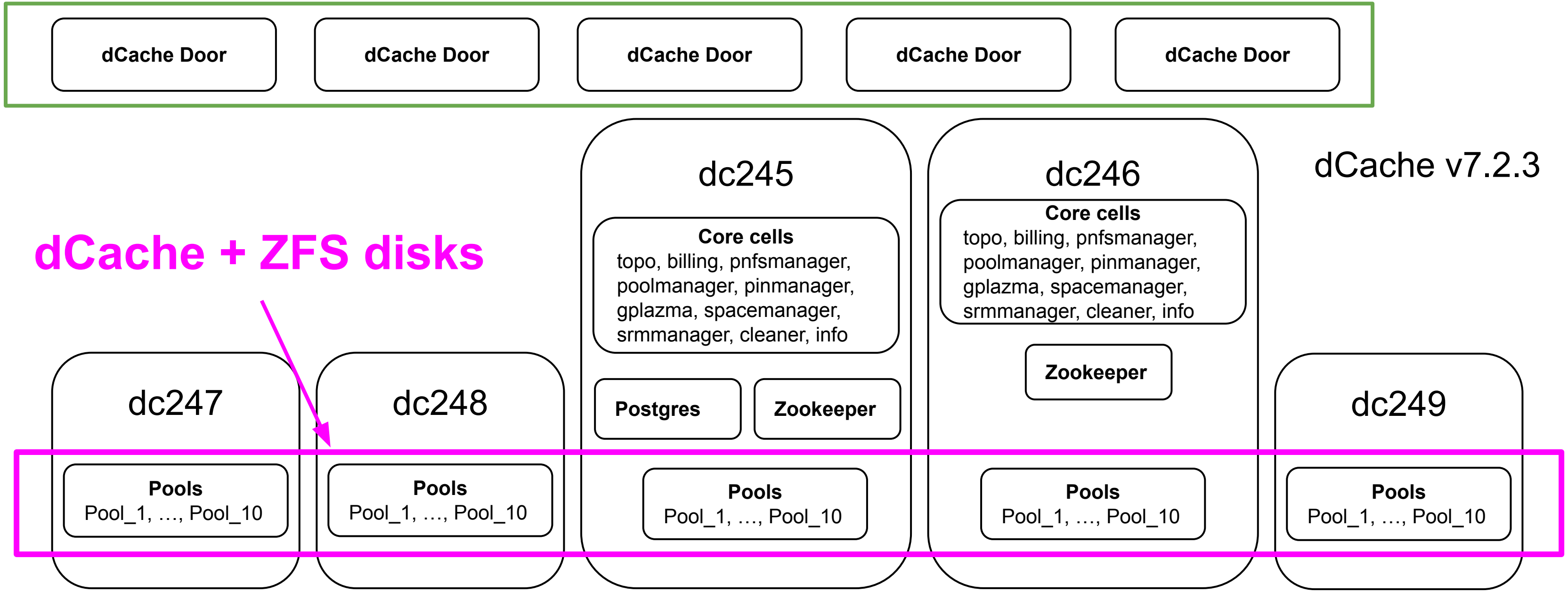
Robinhood

Barreleye

Brookhaven National Laboratory



Testbed: dCache Deployment



Monitoring:

dCache/XRootD Access Layer

dCache or XRootD required to allow storage system to work within the ATLAS distributed storage environment

- ATLAS RSE protocol support requirements defined at: https://twiki.cern.ch/twiki/bin/view/AtlasComputing/StorageSetUp#Protocols

Protocol	XRootd + Lustre [1]		dCache + zfs disks	
	TPC[2]	Direct read/write	TPC	Direct read/write
davs / https	✓	✓	✓	✓
xroot	***Disabled [3]***	✓	✓	✓

Not required for ATLAS RSEs

[1] XrootD configuration with Lustre backend refined with XrootD core team

[2] TPC - Third Party Copy

[3] Problems with xrootd protocol support in XRootD with Lustre backend

Brookhaven National Laboratory

TPC - Write Stress Tests

Source
- Production ATLAS dCache (Datadisk & Scratchdisk)

TPC write →

Testbed
- dCache + ZFS disks
- XRootD + Lustre

FTS

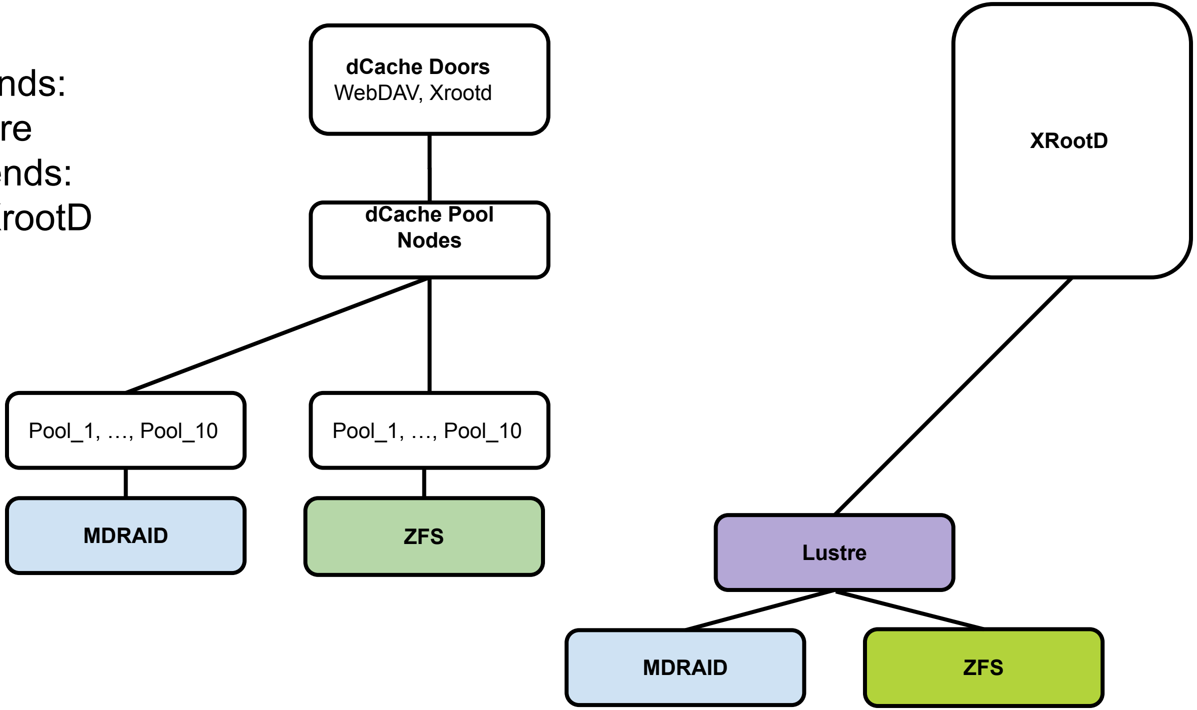
Goal: Saturate the different storage configurations and sustain the peak rates with production data

TPC Copy Configuration
- Scripted bulk FTS transfers
- Files: 500K, Max active limit (FTS): 1200

How to compare XRootD vs. dCache ?

Different backends:
- ZFS vs Lustre

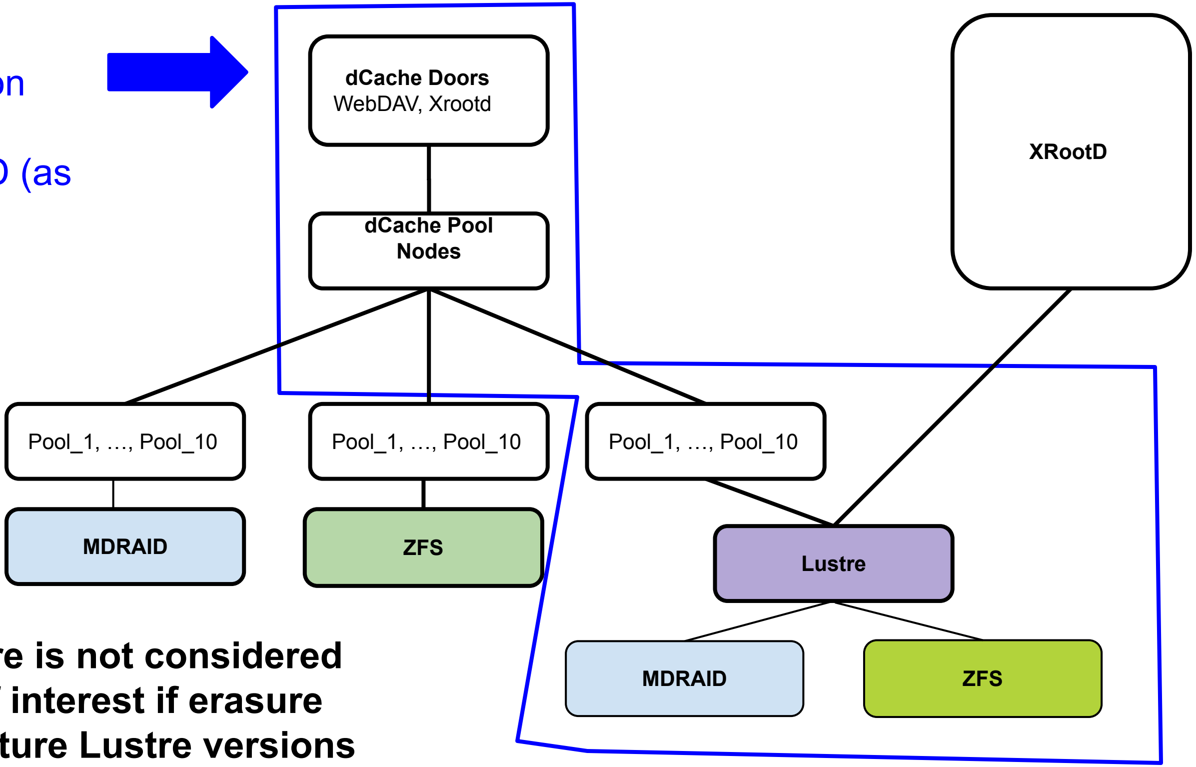
Different front ends:
- dCache vs XrootD



dCache + Lustre !

dCache + Lustre
Allows direct comparison between:

- dCache and XRootD (as backend is identical to XRootD+Lustre)
- ZFS and Lustre (as frontend is identical to dCache + ZFS)



Caveat: dCache + Lustre is not considered as an option. May be of interest if erasure code is supported in future Lustre versions

Brookhaven National Laboratory

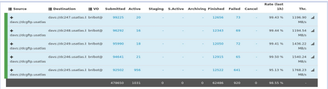
TPC - Write : Stress Tests

Preliminary numbers

Davs TPC	XRootd + Lustre	dCache + Lustre
Aggregate traffic of doors	26GB/s (around 16-17GB for backend Lustre, Extra traffic is caused by checksum) The effective traffic is around 3.4GB/s per door	~21GB/s 4.2GB/s per pool node
CPU Usage	~10% per door	~15% per door, ~ 50-60% per pool node
Checksum	Compute checksum needs reading back, which causes extra traffic	Checksum computed on the fly
Success rate	>98.5%	>98%
Comments	1) It needs more doors to saturate backend Lustre performance because of the extra traffic caused by checksum 2) Some door behave slowly with high CPU usage that degrades cluster performance	1)Higher CPU usage of dCache nodes, probably related to Java

Xrootd vs dCache Checksum

- dCache calculates checksum as the file is received or written to disk, i.e., “on the fly”
- XRootD calculates checksums after the file has been written to disk
  - OS/Lustre client caches aren’t large enough to buffer file content
  - File read from backend storage increasing load on network and backend OSSes
- Observed errors during stress tests, most of which are checksum related issues:

Error Description	Lustre+xrootd	dCache	Comments
Recoverable error: [110] DESTINATION CHECKSUM timeout of 1800s	✗	✗	From FTS side 
Recoverable error: [5] DESTINATION CHECKSUM HTTP 500 : Unexpected server error: 500	✓	✓	**Fixed** by increasing max value for check **max>=512(According to tuning tests)**
Some xrootd doors will degrade with high CPU usage	✗	✓	**Degradation of xrootd door performance. Suspect it’s related to the checksum**

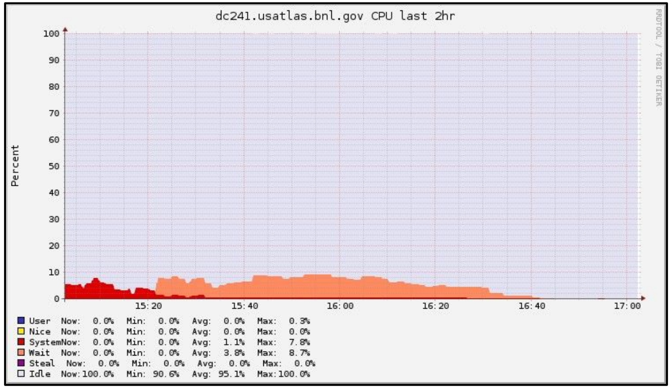
✗ : Exist
✓: Fixed

Brookhaven National Laboratory

Bulk deletion: 500k files

Preliminary numbers

Davs DEL	XRootd + Lustre	dCache + Lustre
Rate	141.79Hz	201Hz
CPU Usage	<2% per door	<2% per door
Comments	dCache with Lustre pool get the best performance, but some contention observed	



Brookhaven National Laboratory

Decision matrix

	Current dCache (MDRAID (14x7)) (2x 1.21)	Unreplicated dCache (MDRAID (14x7)) (1x 1.21)	dCache + Ceph	dCache + disk(zfs (14x7)) (1x 1.27x)	XrootD + Lustre (1x 1.29x)
$/Usable Byte (Arbitrary unit)	1.0 ✗	0.5 ✓	✗	0.52 ✓	0.53 ✓
Ops effort	2 FTEs	>=	+	>=	>=
A&R	✓	?	✓ + ?	✓	? + ?
Replication	X 2	X 1	X 1.25 (12+3)	X 1	X 1
High Availability	✓	✓	✓	✓	? + ?
...	...	...	...	...	...

Brookhaven National Laboratory

Summary

We have gain expertise and operational experience on alternate storage options

- ~~Ceph~~
- OpenZFS (timescale: Q4 2022)
- dCache, XRootD + Lustre

All alternate configurations provide the ATLAS needed functionalities

- XRootD + Lustre — XRootD maturity ?

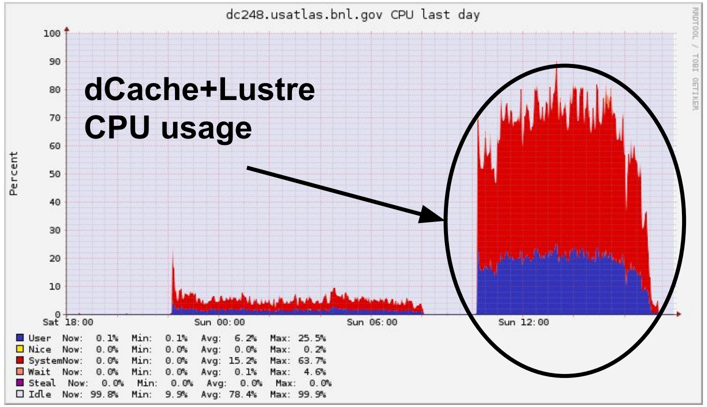
Methods and tools (testbed, monitoring, etc) in place for further evaluation and decide on the optimal option for future storage (timescale: Q1 2023)

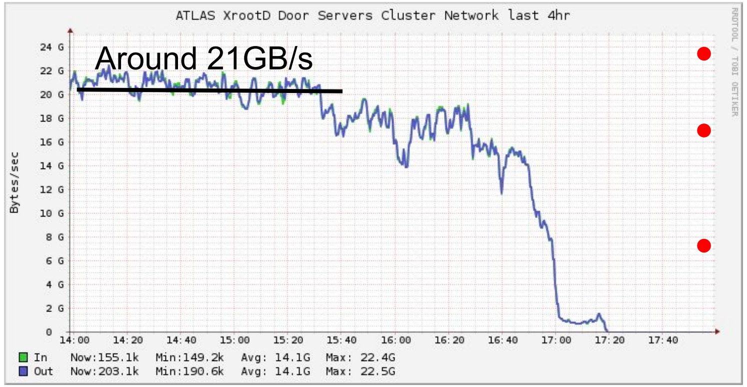
- Performance: XrootD Lustre vs. dCache for TPC read, etc.

<u>Any changes to US ATLAS Tier 1 storage services need to be transparent to ATLAS operations (change management)</u>

Backup

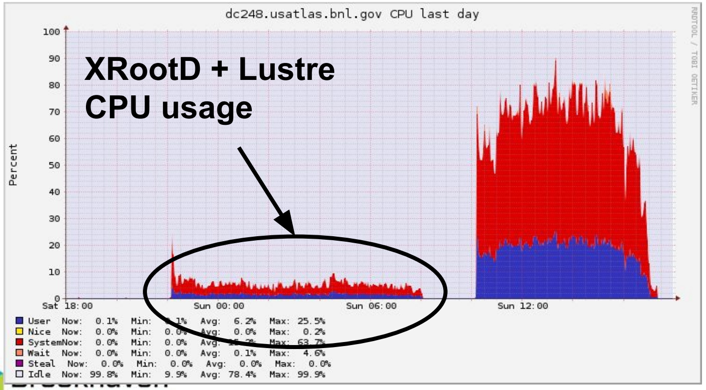
TPC-Write: Dcache+Lustre VS XRootD+Lustre

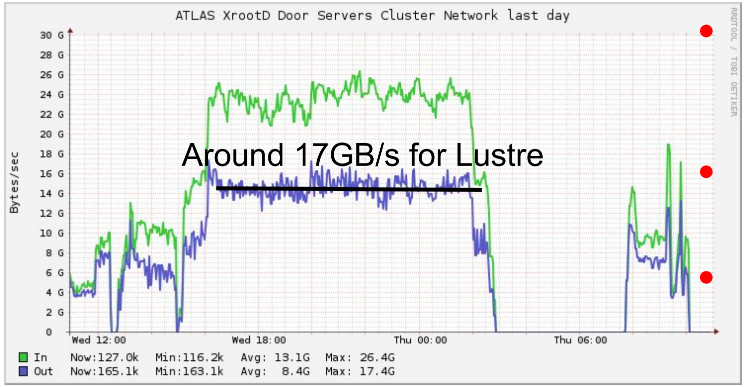




- 5 doors IO traffic: ~21 GB/s
- **4.2GB/s per pool node** from dCache
- **No extra traffic from checksum**

Backend is dCache with Lustre pools





- 5 doors IO traffic: 26GB/s, around 16-17GB for Lustre IO
- **The effective traffic is around 3.4GB/s per door**
- 9-10GB/s traffic is caused by checksum,

Backend is Lustre

Lustre vs dCache: Direct read/write

```
10 X time -p gfal-copy file_randombytes_2G <scheme>://<host>/<path>/file<i>
10 X time -p gfal-copy -f <scheme>://<host>/<path>/file<i> /dev/null
10 X time -p gfal-rm <scheme>://<host>/<path>/file<i>
```

root://	**Write**	**Read**	**Delete**
dCache+localdisk	54.57	22.18	1.75
XRootd+Lustre	54.12	21.55	2.16

➔ **For xrootd protocol: two configurations behave similarly**

Lustre IO performance testing

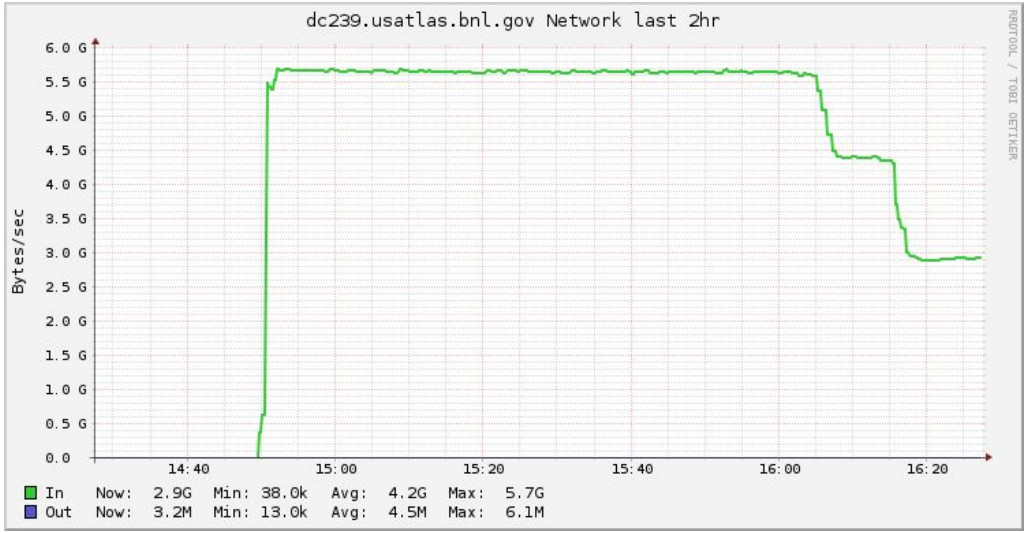
- IOZONE testing
  - write/rewrite, read/reread and stride read
  - **Peak write IO throughput: ~5.7GB/s per OSS(see next slide)**
    - **Network ring buffer for the nics increase from 1024 to 4096**
  - File size
    - Large size:50GB
    - Average Atlas file size: 0.9GB
- Test results before optimizing network parameters

	Nodes#	Process#	Filesize	Block size	Write	Read	Offset	Stride Read
large file size	5	270	50G	1M	24.5GB/s	6.9GB/s	2M	1.6GB/s
Avg Atlas file size	5	270	0.9G	1M	22.1GB/s	10GB/s	2M	7.6GB/s

Brookhaven National Laboratory

Peak write throughput per OSS

Optimization: network parameter (ring buffer for the nics to 4096 from 1024)

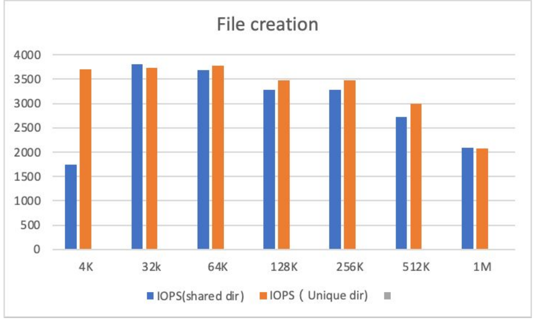


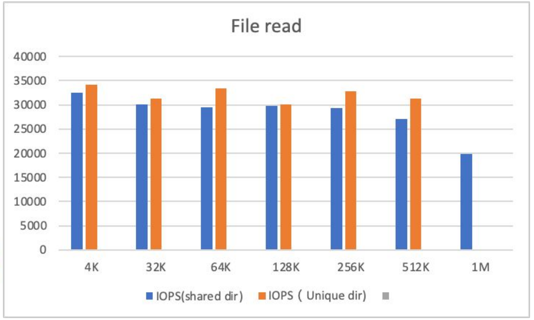
i=0	Client#	Process#	File size	Record size	initial writers	rewriters
	5	180	50G	1M	28579415.15 kB/sec	28588751.47 kB/sec
	5	180	900M	1M	26102364.69 KB/sec	28207374.36 KB/sec

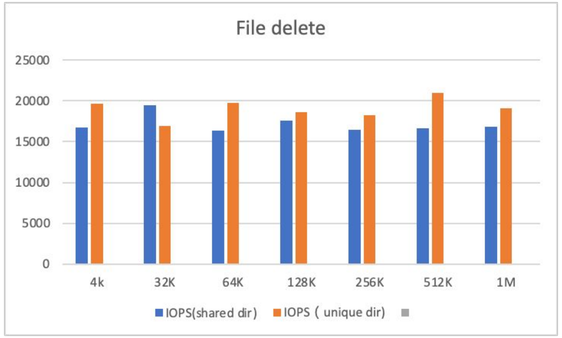
Brookhaven National Laboratory

Mdtest- Metadata performance of Lustre

- Install and configure mdtest/MPI env
- 5 clients, 180 processes
- Metadata performance is good. **Higher performance possible if SSDs are used instead of HDDs**
- More details:
  - https://docs.google.com/document/d/1fNdgHLXNS4D0pQYZSkyFI086QLTtyV9P/edit#







- ➢ File creation:~3000 IOPS
- ➢ File stat: ~60000 IOPS under shared dir
- ➢ File read: ~30000 IOPS
- ➢ File delete: ~16000 IOPS

Higher IOPS in unique dir than shared dirs

Observation: dCache not support MPI

MPI_ABORT was invoked on rank 2 in communicator MPI_COMM_WORLD with errorcode -1.

NOTE: invoking MPI_ABORT causes Open MPI to kill all MPI processes.

You may or may not see output from other processes, depending on exactly when Open MPI kills them.

Lustre vs dCache POSIX Performance

- Lustre - Standard in kernel Lustre client access
- dCache (w/ local pools) - NFS mount

- Simple dd of 1GB file
    - **dd if=/dev/urandom of=sample.txt bs=1G count=60**

	dd 2GB file
dCache	121 MB/s
Lustre	170 MB/s

- First observation: dCache does not support all nfs methods for iozone.

```
$ cd /pnfs/experimental.bnl.gov/data/atlas/
$ iozone -Rac -i 0 -i 1
…
close: Operation not permitted  or fsync: Operation not permitted
…
exiting iozone
```

Protocol	Posix
dCache	Not fully support
Lustre	✓

Brookhaven National Laboratory

Lustre (Release 2.14)

- POSIX-like file system
- Scale out/up features
  - Striping over LUNS
  - Horizontal OSS scaling (striping over nodes) capacity and performance
  - DNE Distributed Name Space horizontal namespace scaling
  - Storage Tiering for performance (SSD/HDD)
- Higher level reliability
  - File Replication (Release 2.13)
  - Erasure code - Still in development. Targeting release 2.15
- HA features
  - HA OSS configurations possible
  - HA MDS configuration