Hewlett Packard
Enterprise

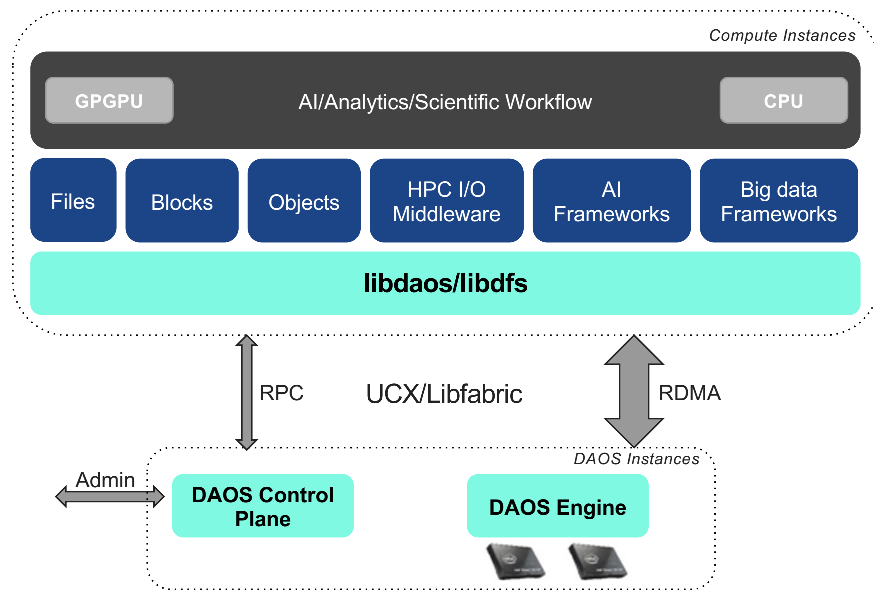
Distributed Asynchronous Object Storage (DAOS) on Aurora

Johann Lombardi, Senior Distinguished Technologist, HPE
Kaushik Velusamy, Assistant Computer Scientist, ALCF

ALCF Developer Session June 25, 2025
https://www.alcf.anl.gov/events/distributed-asynchronous-object-storage-daos-aurora

DAOS: Nextgen Open Storage Platform

- Open platform for innovation
- Files, blocks, objects and more
- Full end-to-end userspace
- Flexible built-in data protection
  - EC/replication with self-healing
  - No performance impact on reads
- Flexible network layer
- Strong distributed consistency
  - Version-based with MVCC
- Efficient single server
  - O(100)GB/s and O(1M) IOPS per server
- Highly scalable
  - TB/s and billions IOPS of aggregated performance
  - O(1M) client processes
- Time to first byte in O(10) µs



DAOS Use Cases

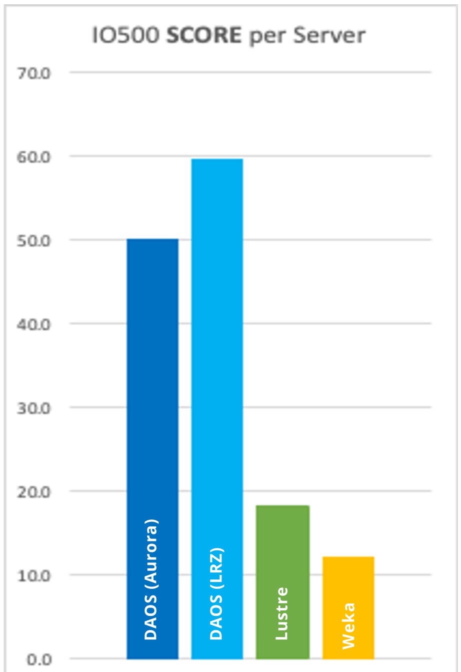
- Checkpoint/restart
  - High bandwidth and large capacity
  - Integration with HPC frameworks (HDF5, MPIIO, …)
- Out-of-core data / irregular accesses
  - Input/output data for simulation
  - High IOPS/bandwidth and low latency
  - Native POSIX and integration with HPC frameworks (HDF5, MPIIO, …)
- Analytics
  - High IOPS and low latency
  - Integration with Apache ecosystem (Spark)
- AI/ML
  - Optimizations for read-only workloads
  - High write bandwidth for checkpointing
  - Low latency **KVCache** for inference (VLLM or Dynamo)
  - Large capacity
  - Integration with AI framework (PyTorch)

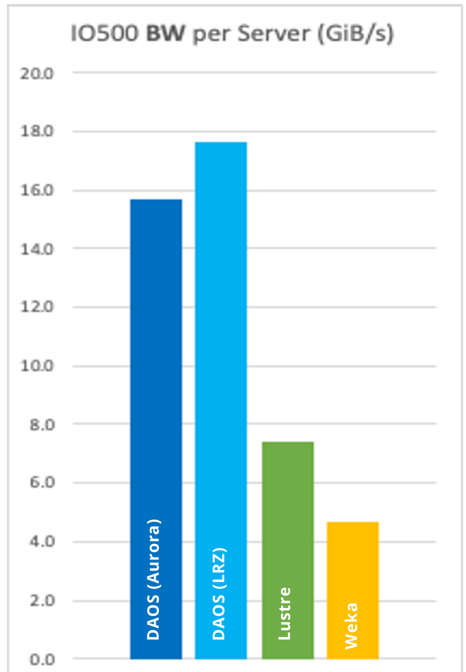
What makes DAOS different from Lustre and other file systems?

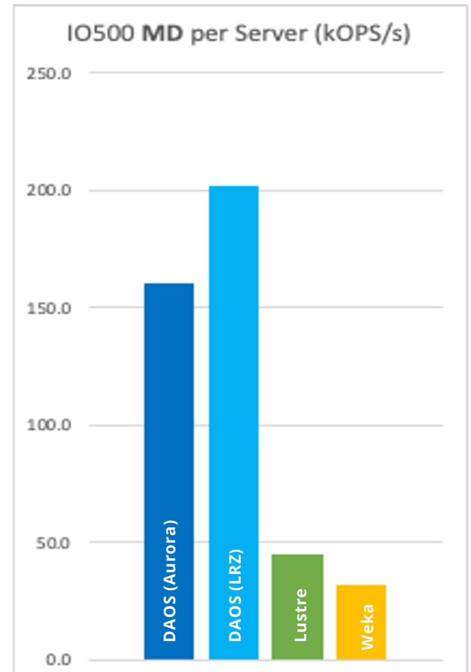
Enabling extreme scalability, performance and application integration for specific use cases

- Distributed key-value store / Versioned byte-granular I/Os
- No synchronous read-modify-write / No locking
- End-to-end in user space / No kernel code
- No centralized metadata servers / No global object table
- End-user managed snapshots / Self-healing data protection
- Multi-tenant storage pooling / Fine-grained access control
- Rich client software ecosystem (e.g. TensorFlow / PyTorch)

IO500 Per-server Performance Comparison (Production List)







Source: https://io500.org

DAOS Foundation

DAOS Community Release

Color coding schema:
- Committed (or released) release/features
- In-planning release/features
- Future possible release/features

Jul'24 → Q4'25 → Q2'26 → H2'26+

DAOS 2.6
(Intel Release)

OS Packages:
- Leap 15.5
- RHEL/Rocky/Alma 8.8/9.2

Networking:
- Change provider w/o reformat
- MD duplicate RPC detection

Features:
- Non-PMem support phase 1
- libpil4dfs
- Intel VMD hotplug
- Delayed rebuild

Tech preview:
- Distributed consistency checker (CR)

UX Improvements:
- Improved version interoperability

DAOS 2.8
(DAOS Foundation Release)

OS Packages:
- Leap 15.6
- RHEL/Rocky/Alma 8.10/9.4

Networking:
- DOCA-OFED support

Features:
- Optimized object placement
- Mount POSIX snapshots RO
- Client telemetry
- Incremental rebuild/reintegration
- Catastrophic recovery and distributed consistency checker
- Fault domains beyond servers

Tech preview:
- Non-PMem support phase 2
- Pytorch data loader
- Rolling upgrade preparation

UX Improvements:
- Reintegration of all pools
- daos pool listing

DAOS 3.0
(DAOS Foundation Release)

OS Packages:
- Leap 15.7 (x86_64)
- RHEL/Rocky/Alma 8.10/9.x (x86_64)
- RHEL/Rocky/Alma 9.x client (ARM64)
- Ubuntu 22.04 client (x86_64/ARM64)

Features:
- Non-PMem support phase 2
- SSD hotplug & LED without VMD

Tech preview:
- Rolling upgrade
- WORM containers phase 1
- Multi-provider support
- flock support
- SSD encryption support via SED

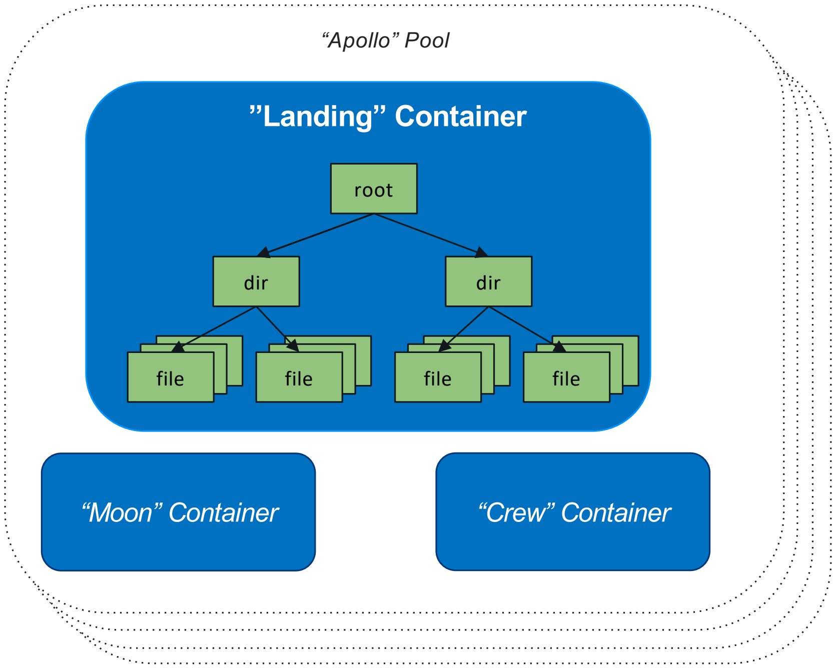
Future Releases
(DAOS Foundation Release)

OS Packages:
- Leap 15.7 (x86_64)
- RHEL/Rocky/Alma 8.10/9.x (x86_64)
- RHEL/Rocky/Alma 9.x (ARM64/x64_64)
- Ubuntu 24.04 client (x86_64/ARM64)

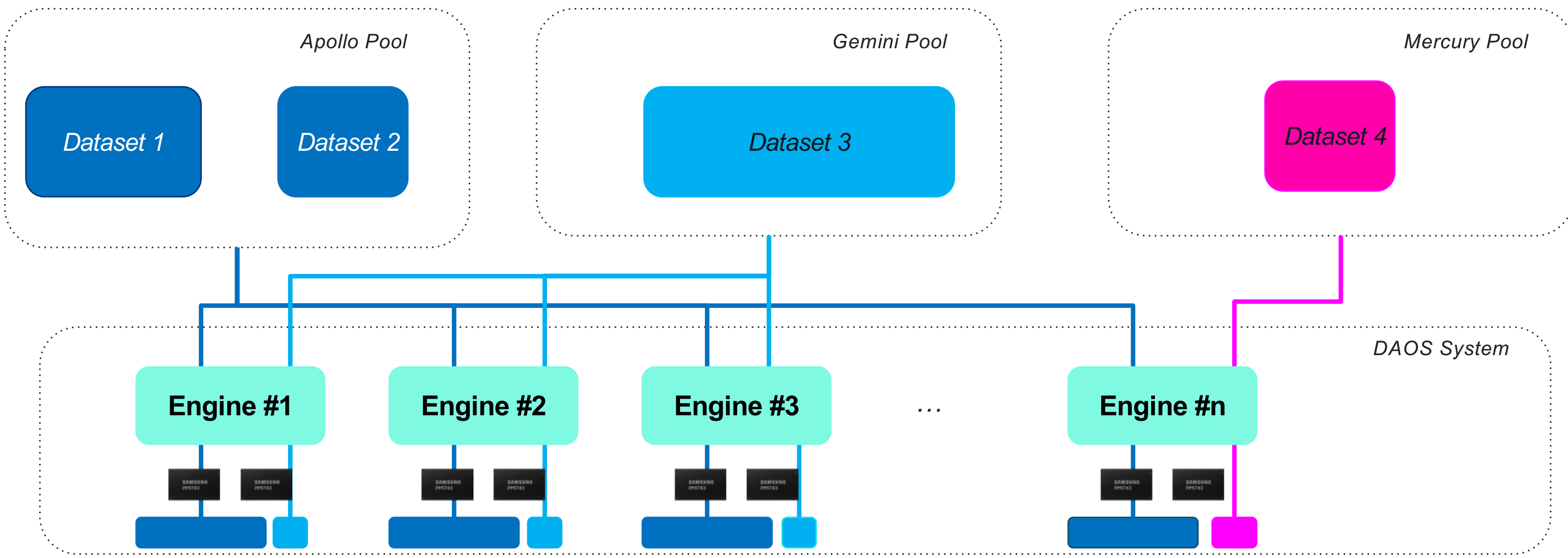
Features:
- Pool resizing
- Inline compression
- Inline encryption
- Inline deduplication
- Middleware consistency checker
- Progressive layout
- Pipeline API
- SQL support with predicate pushdown
- Distributed transactions
- Pool/container freeze
- CXL SSD support / QLC
- Tiered container phase 1
- Support for multiple DAOS systems
- multi-NIC support per engine/process
- hardlinks support in libdfs
- network multipath support
- Container parking/serialization

DAOS Concepts

- Pool
  - private partition/tenant allocated to a project
  - distributed across all the storage nodes
    - Maximum BW/IOPS regardless of size
  - O(100) pools in a system
- Container
  - Dataset/bucket inside a pool
  - O(100) containers in a pool
  - e.g. checkpoints from May campaign
- Object
  - Array or key-value store
  - O(1T) objects in a container
  - e.g. files and directories in a POSIX container



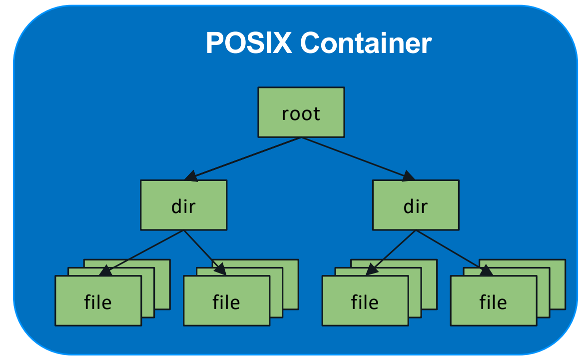
Storage Pooling And Multi-tenancy

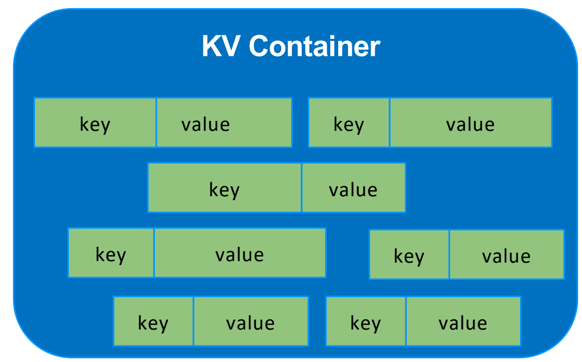


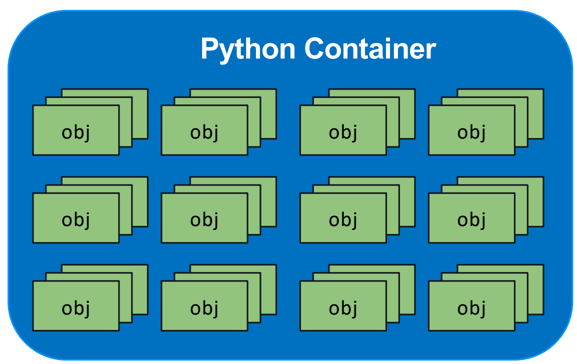
Pool 1	Apollo Pool	100PB	20TB/s	200M IOPS
Pool 2	Gemini Pool	10PB	2TB/s	20M IOPS
Pool 3	Mercury Pool	30TB	80GB/s	2M IOPS

DAOS Containers = Datasets/Buckets

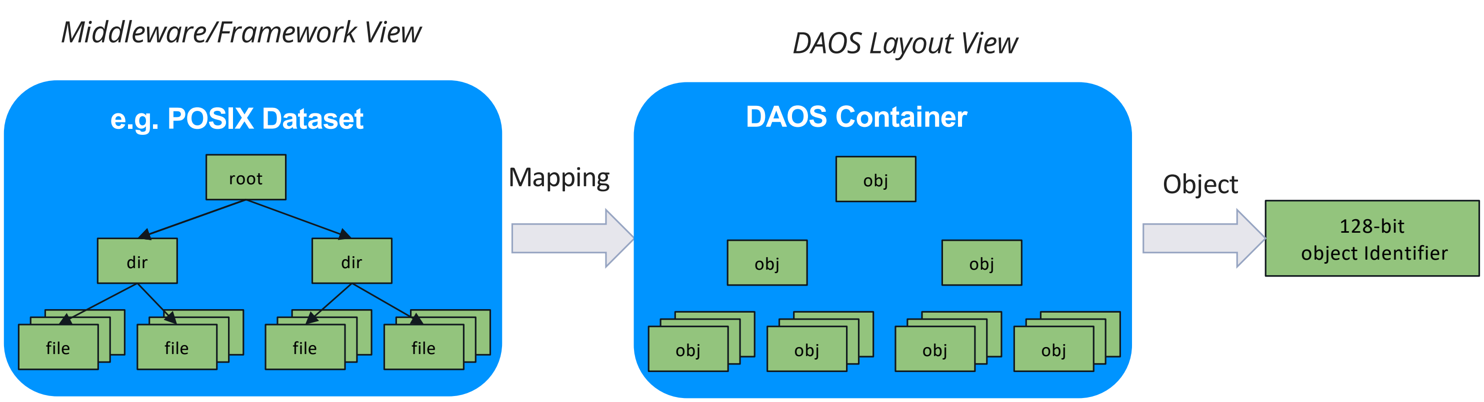
- New data model to unwind 30+y of file-based management
- Introduce notion of dataset = container
- Basic unit of storage
- Containers have a type (e.g. POSIX, pyDAOS, …)
- POSIX containers can include trillions of files/directories
- Advanced dataset query capabilities
- Unit of snapshots
- ACLs
- Best practices:
  - 100’s containers

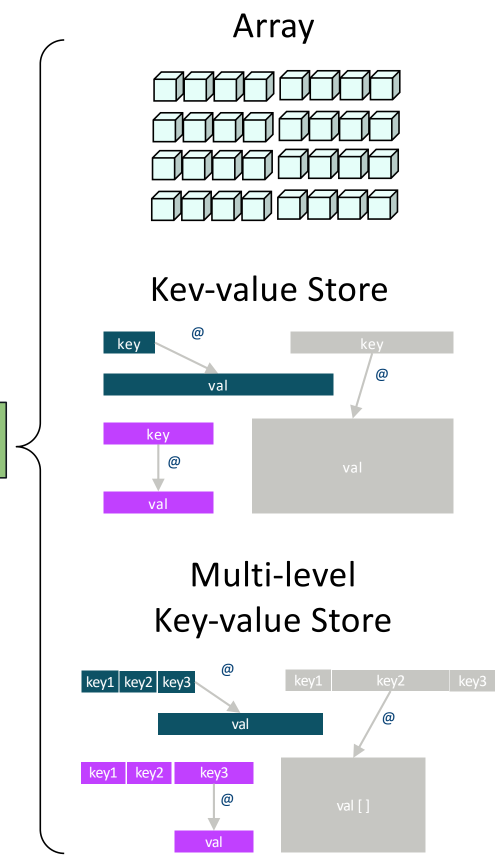






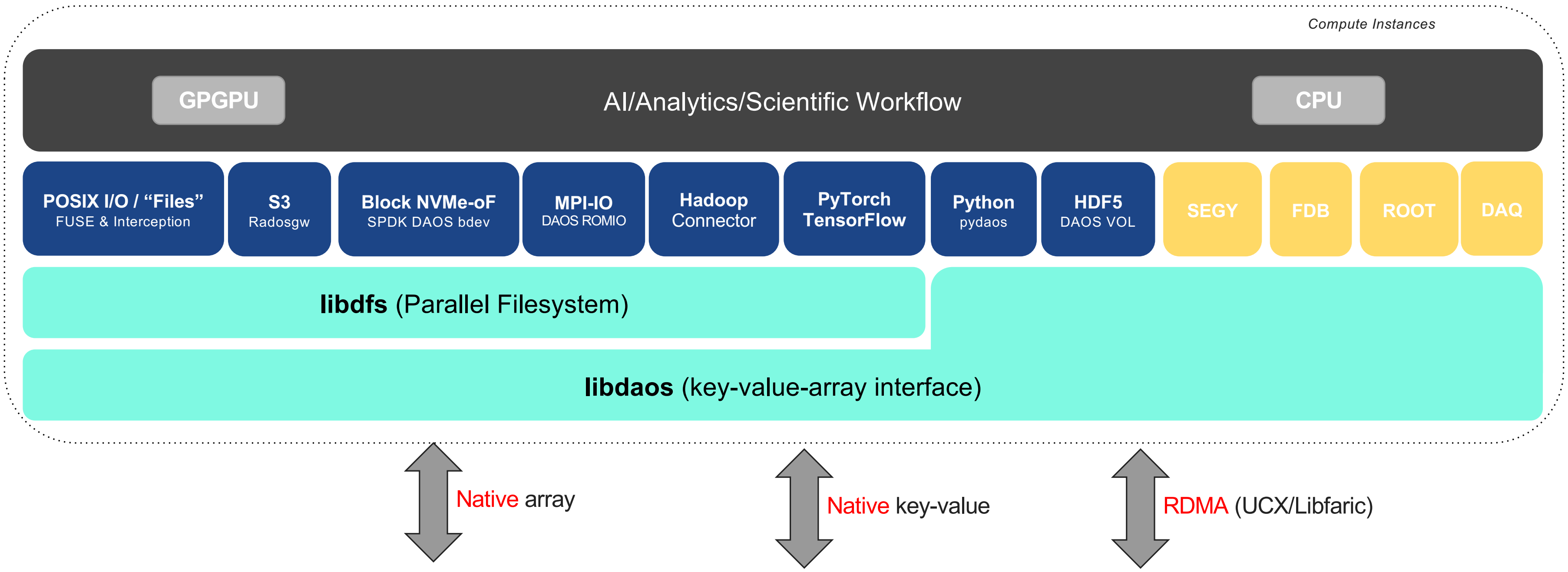
Object Interface





- No object create/destroy
- No size, permission/ACLs or attributes
- Sharded and erasure-coded/replicated
- Algorithmic object placement
- Very short Time To First Byte (TTFB)

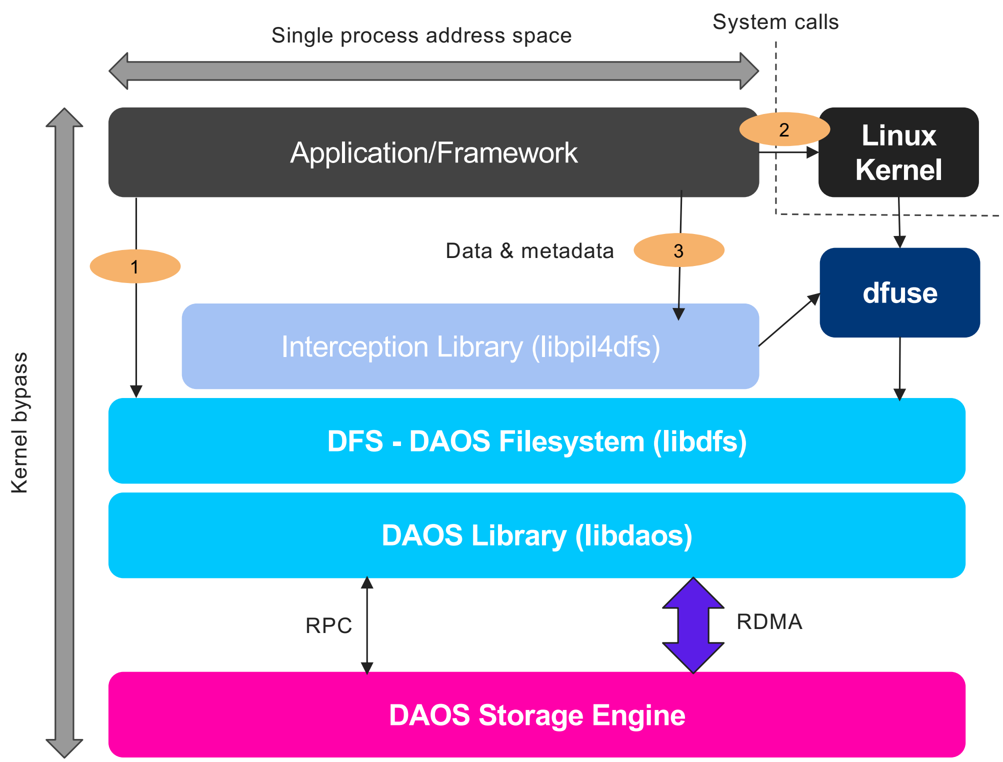
Software Ecosystem



Generic I/O Middleware/frameworks

Domain-specific data models under development in co-design with partners

POSIX Containers



1 Userspace DFS library with API like POSIX

– **Require** application changes
– Low latency & high concurrency
– No caching

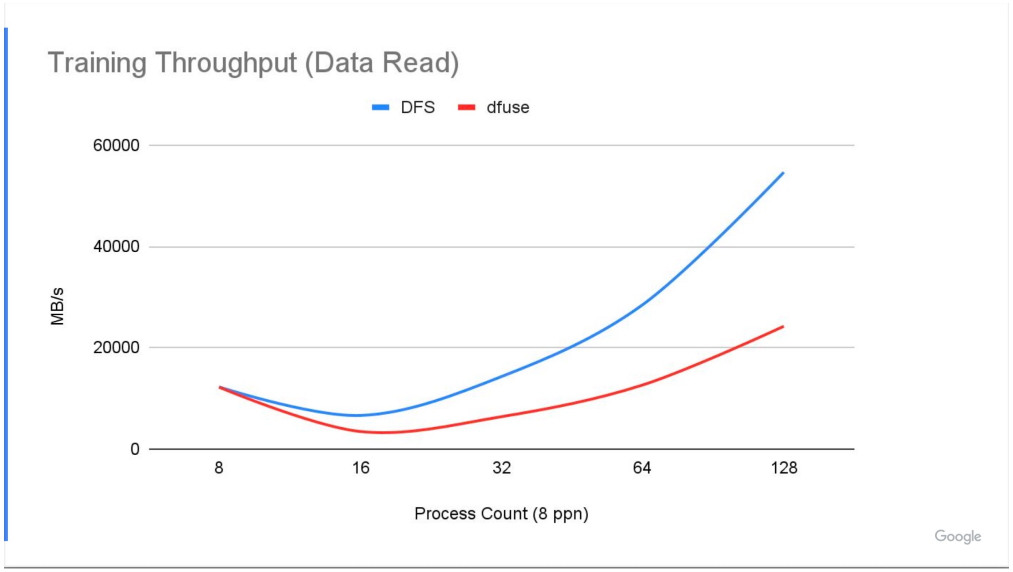
2 DFUSE daemon to support POSIX API

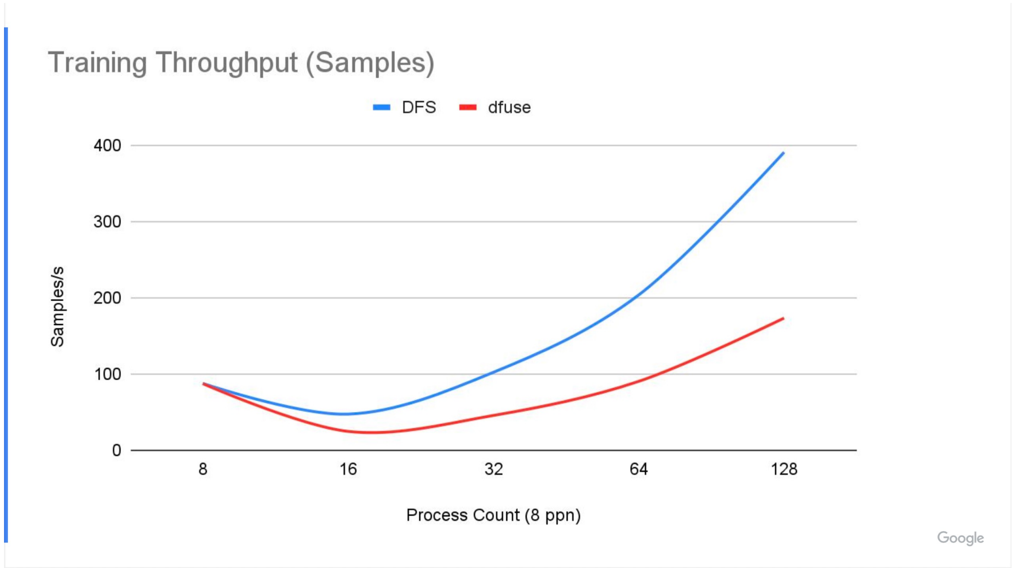
– **No** application changes
– VFS mount point & high latency
– Caching by Linux kernel

3 DFUSE + Interception library

– Aim at delivering same performance as #1
– **No** application changes
– Using LD_PRELOAD intercepting:
  – Data & metadata interception
  – Mmap & binary execution via fuse
– Compatibility to offload fd creation to fuse

PyTorch DAOS Data Loader Module





DAOS on Aurora

- DAOS is a major file system in Aurora : 1024 DAOS Nodes, 230 PB, >25 TB/s
- Open-source software-defined **object store**
- Designed for massively **distributed** Non Volatile **Memory** (NVM) and NVMe **SSD**
- DAOS presents a unified storage model with a native Key-array Value storage interface – POSIX, MPIO, HDF5 etc
- Storage and retrieval of objects in a distributed, parallel, and **asynchronous** manner.
- Advanced data protection, self-healing, redundancy, versioning, distribution and fine-grained data control.

io500.org

IO500

Home About Steering Lists BoFs Rules Running Submission News Graphs Contact

YOU ARE HERE LISTS / ISC25 / Production LIST

Production ISC25 List

Customize Download

Production | 10 Node Production | Research | 10 Node Research | Full | Historical

Ranking of production system submissions. This is a subset of the Full List of submissions, showing only one highest-scoring result per storage system. Submitters who want a submission that is currently on the Research List to be on the Production List should contact the IO500 Steering Committee.

	INFORMATION							IO500			
# ↑	BOF	INSTITUTION	SYSTEM	STORAGE VENDOR	FILE SYSTEM TYPE	CLIENT NODES	TOTAL CLIENT PROC.	SCORE ↑	BW (GIB/S)	MD (KIOP/S)	REPRO.
1	SC23	Argonne National Laboratory	Aurora	Intel	DAOS	300	62,400	32,165.90	10,066.09	102,785.41	✓
2	SC23	LRZ	SuperMUC-NG-Phase2-EC	Lenovo	DAOS	90	6,480	2,508.85	742.90	8,472.60	✓
3	ISC25	Erlangen National High Performance Computing Center	Helma	MEGWARE	Lustre	186	18,600	838.99	438.62	1,604.84	✓
4	ISC25	Samsung Electronics	SSC-24	WekaIO	WekaIO	291	16,005	826.86	248.67	2,749.41	✓
5	SC23	King Abdullah University of Science and Technology	Shaheen III	HPE	Lustre	2,080	16,640	797.04	709.52	895.35	✓
6	SC24	MSKCC	IRIS	WekaIO	WekaIO	261	27,144	665.49	252.54	1,753.69	✓
7	ISC23	EuroHPC-CINECA	Leonardo	DDN	EXAScaler	2,000	16,000	648.96	807.12	521.79	✓
8	SC24	SoftBank Corp	CHIE-3	DDN	EXAScaler	240	26,880	500.20	331.66	754.41	✓
9	ISC25	Joint Center for Advanced High Performance Computing	Miyabi-G	DDN	Lustre	200	1,600	391.60	319.00	480.72	✓
10	SC24	Danish Centre for AI innovation AS	GEFION	DDN	EXAScaler	128	12,288	368.56	209.06	649.73	✓
11	ISC25	Hudson River Trading	HRT	DDN	EXAScaler	10	1,600	348.08	136.05	890.51	✓

Filesystems on Aurora

DAOS

daos_user 128 server cluster
daos_perf 128 server cluster

Nodes	Percentage	Throughput
20	2%	1 TB/s
128	12.50%	5 TB/s
600	60%	10 TB/s
800	78%	20 TB/s
1024	100%	30 TB/s

230 PB

Lustre

- Flare is a **91 PB** Lustre Filesystem with 160 OSTs, 40 MDTs, and 48 Gateway nodes mounted at /lus/flare/projects/ with a peak theoretical performance of **650 GB/s.**

You should launch jobs only from this Flare space.

- Home is a **12 PB** Gecko Lustre Filesystem with 32 OSTs and 12 MDTs.

Recent results from Hardware Accelerated Cosmology Code (HACC)

```
NUM_OF_NODES=512  TOTAL_NUM_RANKS=6144  RANKS_PER_NODE=12
RecordSize = 38
NpTotal = 646736283406 (646736.283406 million particles)
```

HACC achieved Lustre peak theoretical max 600,000 MB/s

```
Wrote 9 variables to
./lustre_out_712762.aurora-pbs-
0001.hostmgmt.cm.aurora.alcf.anl.go
v_512/lus_pos_test_kaus_1_
(24575980196884 bytes) in 39.2264s:
597492 MB/s
```

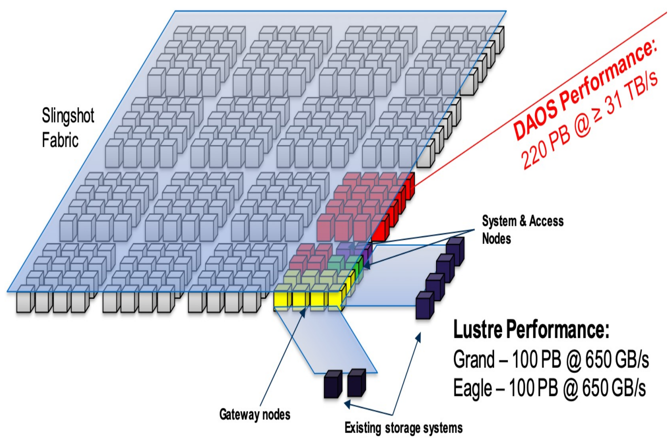
Lustre : 597,492 MB/s

HACC achieved 5.2 TB/s with DAOS, which is close to the IOR peak for the daos_user 128 server cluster
IOR Max Write: 5,723,194.84 MB/s

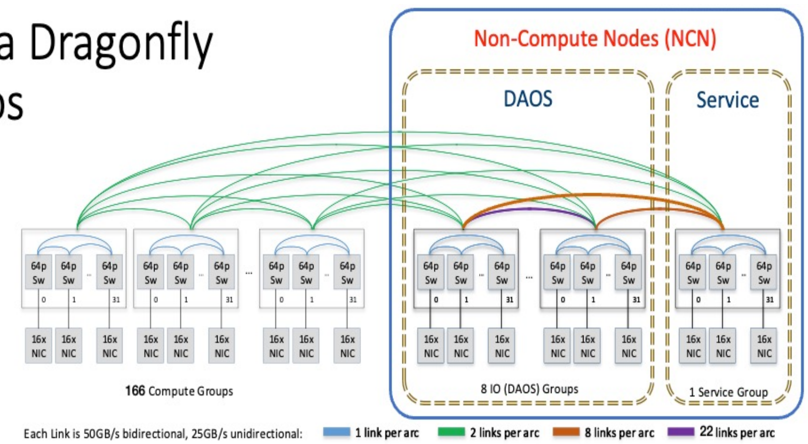
```
Wrote 9 variables to to
/tmp/datascience/1_fSX_dS1_rd_fac_0
/daos_pos_test_kaus_1_
(26560000168392 bytes) in
pure_io_time 4.81287s: 5262890 MB/s
```

DAOS : 5,262,890 MB/s

Network Architecture – slingshot fabric - Dragonfly



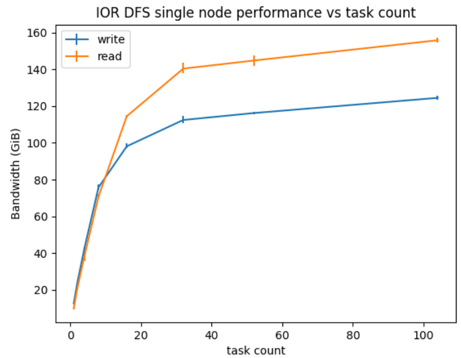
Aurora Dragonfly Groups

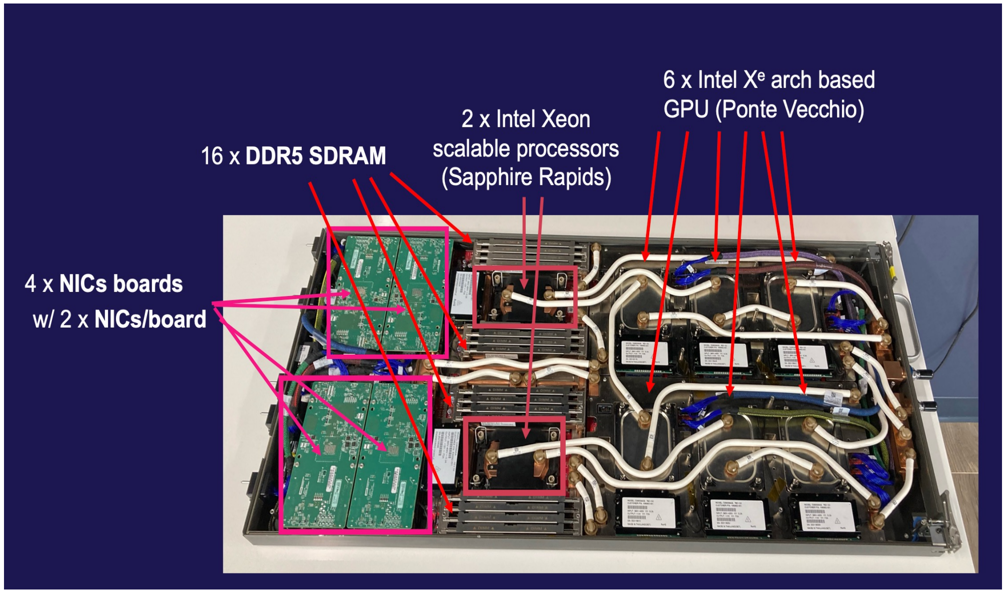


- 1-D Dragonfly Topology - 175 total groups (166 compute + 8 IO + 1 Service)
  - All the global links are optical, all the local links are electrical
  - 2 global links between any two compute groups
  - 22 links between any two IO groups, 8 links between the Service group and each IO group
- All NCN will be racked in HPE cabinets and under HPE System Management.
  - The DAOS cluster is 64 DAOS racks of 1024 DAOS nodes organized in 8 Dragonfly groups.
  - The Service cluster is a single Dragonfly group composed of 6 racks of I/O gateway nodes and 6 racks of HPE Cray front-end nodes (FENs).

A single Aurora compute node

8 NICs, 52 cores per socket, 2 sockets
25GB/s X 8 NICs = 200 GB/s

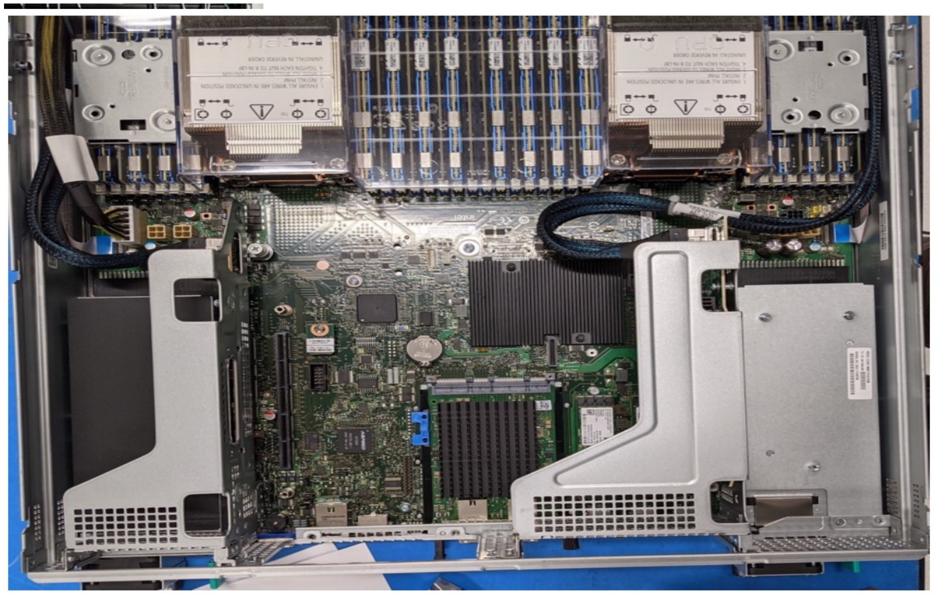




A single DAOS storage node

1024 Total DAOS Servers
Each node will run 2 DAOS engines
2048 DAOS engines, **32 targets**

Intel Coyote Pass System
(2) Xeon 5320 CPU (Ice Lake)
(16) 32GB DDR4 DIMMs
(16) 512GB Intel Optane Persistent Memory 200
(16) 15.3TB Samsung PM1733
(2) HPE Slingshot NICs
(25 ~ 20) GB/s X 2 NICs = 40GB/s



Recommended NIC binding for 12 PPN

```
CPU_BINDING1=list:4:9:14:19:20:25:56:61:66:71:74:79
```

NIC 0	NIC 1	NIC 2	NIC 3	NIC 4	NIC 5	NIC 6	NIC 7
0	1	2	3	52	53	54	55
4	5	6	7	56	57	58	59
8	9	10	11	60	61	62	63
12	13	14	15	64	65	66	67
16	17	18	19	68	69	70	71
20	21	22	23	72	73	74	75
24	25	26	27	76	77	78	79
28	29	30	31	80	81	82	83
32	33	34	35	84	85	86	87
36	37	38	39	88	89	90	91
40	41	42	43	92	93	94	95
44	45	46	47	96	97	98	99
48	49	50	51	100	101	102	103

DAOS Pool Allocation

- DAOS Overview
- The first step in using DAOS is to get DAOS POOL space allocated for your project. Users should submit a request as noted below to have a DAOS pool created for your project.

DAOS pool is a physically allocated dedicated storage space for your project.

Email support@alcf.anl.gov to request a DAOS pool with the following information:

- Project Name
- ALCF User Names
- Total Space requested (typically 100 TBs++)
- Justification
- Preferred pool name

Step 1/5 : Module load daos

```
$ module use /soft/modulefiles
$ module load daos
```

Step 2/5 : Verify your pool

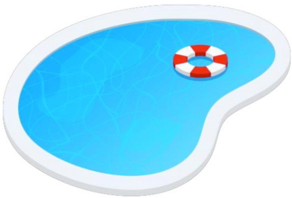
```
kaushikvelusamy@aurora-uan-0010:/gecko/Aurora_deployment> daos pool query datascience
Pool 0477964f-3d74-4053-ba42-ac0c0f9feb95, ntarget=640, disabled=144, leader=14,
version=282
Pool space info:
- Target(VOS) count:496
- Storage tier 0 (SCM):
  Total size: 2.3 TB
  Free: 1.2 TB, min:2.5 GB, max:2.5 GB, mean:2.5 GB
- Storage tier 1 (NVMe):
  Total size: 75 TB
  Free: 75 TB, min:151 GB, max:152 GB, mean:151 GB
Rebuild busy, 81 objs, 6798049280 recs
```

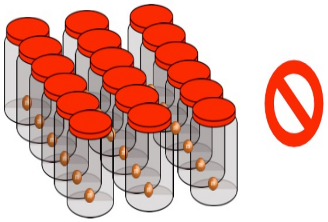
ntarget=640 / 32 targets per node = daos cluster size or daos server size

Physically allocated dedicated storage for your project.

Step 3/5 : Create your container

- A pool contains *thousands* of containers
- Basic unit of storage from user perspective





- DAOS_POOL_NAME=datascience
- DAOS_CONT_NAME=LLM-GPT-1T

- **daos container create –type=POSIX ${DAOS_POOL_NAME} ${DAOS_CONT_NAME} --properties=cksum:crc32,srv_cksum:on,rd_fac:2**

Or daos container create --type=POSIX ${DAOS_POOL_NAME} ${DAOS_CONT_NAME}
--chunk-size=2097152 --file-oclass=EC_16P2G32 --dir_oclass=RP3G1
--properties=rd_fac:2,ec_cell_sz:131072,cksum:crc32,srv_cksum:on

```
  Container UUID : 59747044-016b-41be-bb2b-22693333a380
  Container Label: LLM-GPT-1T
  Container Type : POSIX
Successfully created container 59747044-016b-41be-bb2b-22693333a380
```

- daos container query $DAOS_POOL_NAME $DAOS_CONT_NAME
- daos container get-prop $DAOS_POOL_NAME $DAOS_CONT_NAME
- daos container list $DAOS_POOL_NAME $DAOS_CONT_NAME
  - daos pool   autotest $DAOS_POOL_NAME
- daos container destroy $DAOS_POOL_NAME $DAOS_CONT_NAME
- daos container check --pool=$DAOS_POOL_NAME --cont=$DAOS_CONT_NAME

Step 4/5: Mouting your DAOS container

Interacting with DAOS (Login Vs Compute nodes)

From login node

- `mkdir -p /tmp/${USER}/${DAOS_POOL}/${DAOS_CONT}`
- **`start-dfuse.sh -m /tmp/${USER}/${DAOS_POOL}/${DAOS_CONT} --pool ${DAOS_POOL} --cont ${DAOS_CONT}`**
- `fusermount3 -u /tmp/${USER}/${DAOS_POOL}/${DAOS_CONT}`

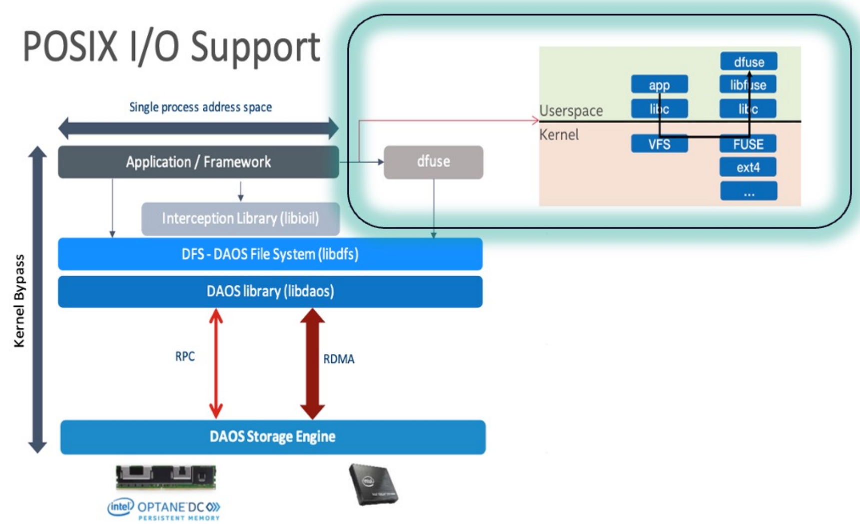
From compute node

- launched using pdsh/clush on all compute nodes mounted at: `/tmp/${DAOS_POOL}/${DAOS_CONT}`
- **`launch-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}`**
- `clean-dfuse.sh  ${DAOS_POOL}:${DAOS_CONT}`

Validation

- `mount | grep dfuse`
- **`ls /tmp/${DAOS_POOL}/${DAOS_CONT}/`**

Step 5/5 : Interception library for POSIX mode



- mpiexec # no interception
- mpiexec --env LD_PRELOAD=/usr/lib64/libioil.so # only data is intercepted
- **mpiexec --env LD_PRELOAD=/usr/lib64/libpil4dfs.so** *# preferred - both metadata and data is intercepted. This provides close to DFS mode performance.*

Job submission without DAOS

```
qsub -l select=1
      -l walltime=01:00:00
      -A <ProjectName>
      -k doe
      -l filesystems=flare
      -q debug
      ./pbs_script1.sh
```

Job submission with DAOS

```
qsub -l select=1
      -l walltime=01:00:00
      -A <ProjectName>
      -k doe
      -l filesystems=flare:daos_user
      -l daos=daos_user
      -q debug
      ./pbs_script1.sh
```

```
mpiexec
-env LD_PRELOAD=/usr/lib64/libpil4dfs.so
….. -no-vni …
```

```
#PBS -q prod
#PBS -k doe
#PBS -l filesystems=flare:daos_user
#PBS -l daos=daos_user
# qsub -l select=512:ncpus=208 -l walltime=01:00:00 -A <ProjectName> -l filesystems=flare:daos_user -l daos=daos_user -q prod ./pbs_script.sh or - I

module use /soft/modulefiles
module load daos
DAOS_POOL=datascience
DAOS_CONT=thundersvm_exp1
daos container create --type POSIX ${DAOS_POOL}  ${DAOS_CONT} --properties rd_fac:2
launch-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}           # To mount on a compute node
ls /tmp/${DAOS_POOL}/${DAOS_CONT}                   #optional for compute node

# cp /lus/flare/projects/CSC250STDM10_CNDA/kaushik/thundersvm/input_data/real-sim_M100000_K25000_S0.836 /tmp/${DAOS_POOL}/${DAOS_CONT} #one time
NNODES=`wc -l < $PBS_NODEFILE`
RANKS_PER_NODE=12          # Number of MPI ranks per node
NRANKS=$(( NNODES * RANKS_PER_NODE ))
echo "NUM_OF_NODES=${NNODES}  TOTAL_NUM_RANKS=${NRANKS}  RANKS_PER_NODE=${RANKS_PER_NODE}"
CPU_BINDING1=list:4:9:14:19:20:25:56:61:66:71:74:79

export THUN_WS_PROB_SIZE=1024
export ZE_FLAT_DEVICE_HIERARCHY=COMPOSITE
export AFFINITY_ORDERING=compact
export RANKS_PER_TILE=1
export PLATFORM_NUM_GPU=6
export PLATFORM_NUM_GPU_TILES=2

date
LD_PRELOAD=/usr/lib64/libpil4dfs.so mpiexec -np ${NRANKS} -ppn ${RANKS_PER_NODE} --cpu-bind ${CPU_BINDING1}  \
                                            --no-vni -genvall  thunder/svm_mpi/run/aurora/wrapper.sh thunder/svm_mpi/build_ws1024/bin/thundersvm-train \
                                            -s 0 -t 2 -g 1 -c 10 -o 1  /tmp/datascience/thunder_1/real-sim_M100000_K25000_S0.836
date

clean-dfuse.sh ${DAOS_POOL}:${DAOS_CONT} #to unmount on compute node
```

Types of DAOS containers and interfaces

A. Posix Interface

```
>ls /tmp/datascience/
train-data/ test-data/ val-data/
```

MPIIO and HDF5 users can directly use this interface.

B. DFS interface

Do not pass the full dfuse path /tmp/poolname/containername,
directly start with / and the file or sub dir past /tmp/poolname/containername

```
/myfile
```

C. Python PyDAOS interface

```
import pydaos
import torch as sys_torch
from pydaos.daos_torch import Dataset as DaosDataset
from pydaos.daos_torch import Checkpoint as DaosCheckpoint
from io import BytesIO
```

Bonus! Distributed Data mover tools

```
kaushikvelusamy@x4210c6s0b0n0:/soft/daos/mpifileutils/bin> ls
dbcast  dbz2  dchmod  dcmp  dcp  dcp1  ddup  dfilemaker1  dfind  dreln  drm  dstripe  dsync  dtar  dwalk

kaushikvelusamy@x4210c6s0b0n0:/tmp> mpiexec --env LD_PRELOAD=/usr/lib64/libpil4dfs.so  -np 16 -ppn 16 --cpu-bind list:4:5:6:7:8:9:10:11:56:57:58:59:60:61:62:63 /soft/daos/mpifileutils/bin/dcp  source/ /tmp/datascience/1_fSX_dS1_rd_fac_0/
[2025-05-17T04:08:39] Walking /tmp/source
[2025-05-17T04:08:39] Walked 11 items in 0.002 secs (5838.681 items/sec) ...
[2025-05-17T04:08:39] Walked 11 items in 0.002 seconds (4502.556 items/sec)
[2025-05-17T04:08:39] Copying to /tmp/datascience/1_fSX_dS1_rd_fac_0
[2025-05-17T04:08:39] Items: 11
[2025-05-17T04:08:39]   Directories: 1
[2025-05-17T04:08:39]   Files: 10
[2025-05-17T04:08:39]   Links: 0
[2025-05-17T04:08:39] Data: 200.000 GiB (20.000 GiB per file)
[2025-05-17T04:08:45] Copy data: 200.000 GiB (214748364800 bytes)
[2025-05-17T04:08:45] Copy rate: 38.088 GiB/s (214748364800 bytes in 5.251 seconds)
[2025-05-17T04:08:45] Fixing permissions.
[2025-05-17T04:08:45] Updated 11 items in 0.001 seconds (7689.630 items/sec)
[2025-05-17T04:08:45] Syncing directory updates to disk.
[2025-05-17T04:08:45] Sync completed in 0.002 seconds.
[2025-05-17T04:08:45] Started: May-17-2025,04:08:39
[2025-05-17T04:08:45] Completed: May-17-2025,04:08:45
[2025-05-17T04:08:45] Seconds: 5.299
[2025-05-17T04:08:45] Items: 11
[2025-05-17T04:08:45]   Directories: 1
[2025-05-17T04:08:45]   Files: 10
[2025-05-17T04:08:45]   Links: 0
[2025-05-17T04:08:45] Data: 200.000 GiB (214748364800 bytes)
[2025-05-17T04:08:45] Rate: 37.744 GiB/s (214748364800 bytes in 5.299 seconds)
```

Monitoring with DAOS interception library report

export D_IL_REPORT=1

mpiexec --env LD_PRELOAD=/usr/lib64/libpil4dfs.so
-np ${NRANKS}..

```
Options:
api                 : POSIX
apiVersion          :
test filename       : /tmp/datascience/1_fSX_dS1_rd_fac_0/ior_file_1.dat
access              : single-shared-file
type                : independent
segments            : 1
ordering in a file  : sequential
ordering inter file : constant task offset
task offset         : 1
nodes               : 2
tasks               : 24
clients per node    : 12
memoryBuffer        : CPU
dataAccess          : CPU
GPUDirect           : 0
repetitions         : 1
xfersize            : 1 MiB
blocksize           : 10 MiB
aggregate filesize  : 240 MiB
verbose             : 1
stonewallingTime    : 100
stoneWallingWearOut : 0

libpil4dfs intercepting summary for ops on DFS:
[read   ]  10
[write  ]  10

[open   ]  2
[stat   ]  2
[opendir]  0
[readdir]  0
[link   ]  0
[unlink ]  0
[rdlink ]  0
[seek   ]  20
[mkdir  ]  0
[rmdir  ]  0
[rename ]  0
[mmap   ]  0

[op_sum ]  44
libpil4dfs intercepting summary for ops on DFS:
[read   ]  10
[write  ]  10

[open   ]  2
```

Darshan module on aurora and LD_PRELOAD ORDER

```
module use /soft/perftools/darshan/darshan-3.4.7/share/craype-2.x/modulefiles
module load darshan
```

```
LD_PRELOAD=/soft/perftools/darshan/darshan-3.4.7/lib/libdarshan.so:
            /opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-
x86_64/oneapi-2025.0.5/hdf5-1.14.5-zrlo32i/lib/libhdf5.so:
            /opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-
x86_64/oneapi-2025.0.5/parallel-netcdf-1.12.3-cszcp66/lib/libpnetcdf.so:
            /usr/lib64/libpil4dfs.so
```

If your application is using gpu_tile_compact.sh then this whole LD_PRELOAD will go in your personal copy of the bash script via export.

```
mpiexec…
```

```
/lus/flare/logs/darshan/aurora/2025/5/21/mgarcia_fornax-GPU-3D_id4916309-158881_5-21-61181-
16288946849860429419_1.darshan
```

pydarshan

```
~~module use /soft/perftools/darshan/darshan-3.4.7/share/craype-2.x/modulefiles~~
~~module load darshan~~
~~module load python~~
~~. /soft/daos/pydarshan_plots_venv/bin/activate~~
~~pip show darshan~~
```

```
module load darshan
module load darshan-util
module load py-darshan
```

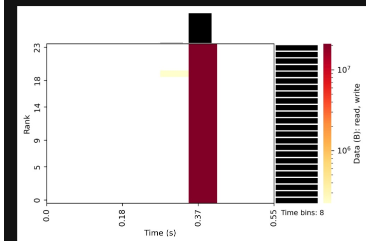
```
LD_PRELOAD=/soft/perftools/darshan/darshan-3.4.7/lib/libdarshan-util.so

# For html file
python -m darshan summary
 /lus/flare/logs/darshan/aurora/2025/5/21/mgarcia_fornax-GPU-3D_id4916309-158881_5-21-61181-
16288946849860429419_1.darshan

# For text form
/soft/perftools/darshan/darshan-3.4.7/bin/darshan-parser
/lus/flare/logs/darshan/aurora/2025/5/21/mgarcia_fornax-GPU-3D_id4916309-158881_5-21-61181-
16288946849860429419_1.darshan > out.txt.
```

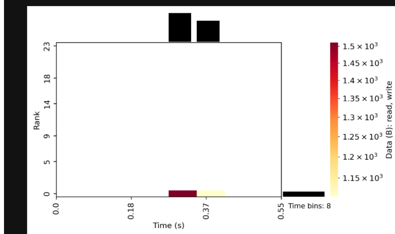
~~deactivate~~

Heat Map: HEATMAP POSIX



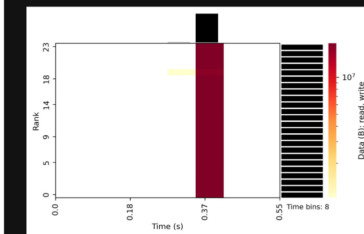
Heat map of I/O (in bytes) over time broken down by MPI rank. Bins are populated based on the number of bytes read/written in the given time interval. The top edge bar graph sums each time slice across ranks to show aggregate I/O volume over time, while the right edge bar graph sums each rank across time slices to show I/O distribution across ranks.

Heat Map: HEATMAP STDIO



Heat map of I/O (in bytes) over time broken down by MPI rank. Bins are populated based on the number of bytes read/written in the given time interval. The top edge bar graph sums each time slice across ranks to show aggregate I/O volume over time, while the right edge bar graph sums each rank across time slices to show I/O distribution across ranks.

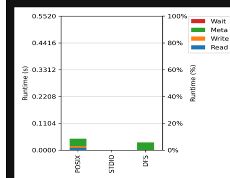
Heat Map: HEATMAP DFS



Heat map of I/O (in bytes) over time broken down by MPI rank. Bins are populated based on the number of bytes read/written in the given time interval. The top edge bar graph sums each time slice across ranks to show aggregate I/O volume over time, while the right edge bar graph sums each rank across time slices to show I/O distribution across ranks.

Cross-Module Comparisons

I/O Cost



Average (across all ranks) amount of run time that each process spent performing I/O, broken down by access type. See the right edge bar graph on heat maps in preceding section to indicate if I/O activity was balanced across processes. The 'Wait' category is only meaningful for PNETCDF asynchronous I/O operations.

Per-Module Statistics: POSIX

Overview

files accessed	1
bytes read	240.00 MiB
bytes written	240.00 MiB
I/O performance estimate	8567.13 MiB/s (average)

File Count Summary (estimated by POSIX I/O access offsets)

	number of files	avg. size	max size
total files	1	240.00 MiB	240.00 MiB
read-only files	0	0	0
write-only files	0	0	0
read/write files	1	240.00 MiB	240.00 MiB

Per-Module Statistics: STDIO

Overview

files accessed	1
bytes read	0 Bytes
bytes written	2.56 KiB
I/O performance estimate	39.07 MiB/s (average)

File Count Summary (estimated by STDIO I/O access offsets)

	number of files	avg. size	max size
total files	1	2.56 KiB	2.56 KiB
read-only files	0	0	0
write-only files	1	2.56 KiB	2.56 KiB
read/write files	0	0	0

Per-Module Statistics: DFS

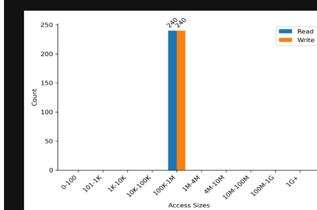
Overview

files accessed	1
bytes read	192.02 MiB
bytes written	240.00 MiB
I/O performance estimate	10492.77 MiB/s (average)

File Count Summary (estimated by DFS I/O access offsets)

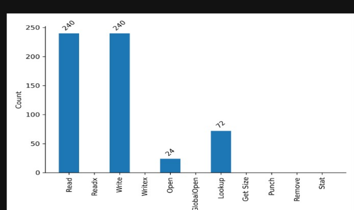
	number of files	avg. size	max size
total files	1	0	0
read-only files	0	0	0
write-only files	0	0	0
read/write files	1	0	0

Access Sizes



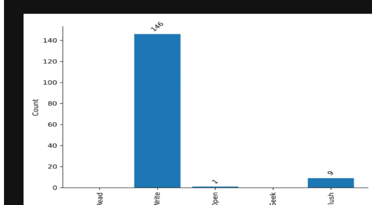
Histogram of read and write access sizes. The specific values of the most frequently occurring access sizes can be found in the *Common Access Sizes* table.

Operation Counts



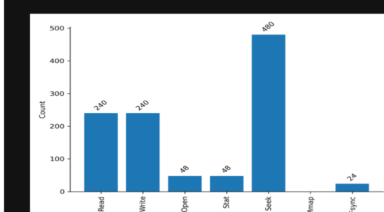
Histogram of I/O operation frequency.

Operation Counts



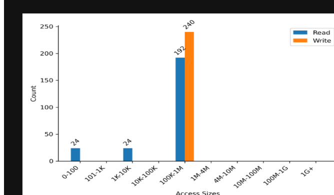
Histogram of I/O operation frequency.

Operation Counts



Histogram of I/O operation frequency.

Access Sizes



Histogram of read and write access sizes. The specific values of the most frequently occurring access sizes can be found in the *Common Access Sizes* table.

Reference

- https://docs.alcf.anl.gov/aurora/data-management/daos/daos-overview/
- https://docs.alcf.anl.gov/aurora/data-management/daos/daos-overview/#best-practices

Acknowledgements

- Gordon Mcpheeters, Kevin Harms, Paul Coffman, Huihuo Zheng, Venkatram Vishwanath
- HPE DAOS team: Mohamad Chaarawi, John Carrier, Sylvia Chan
- MCS team: Rob Latham, Shane Snyder

- This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.