
intel.

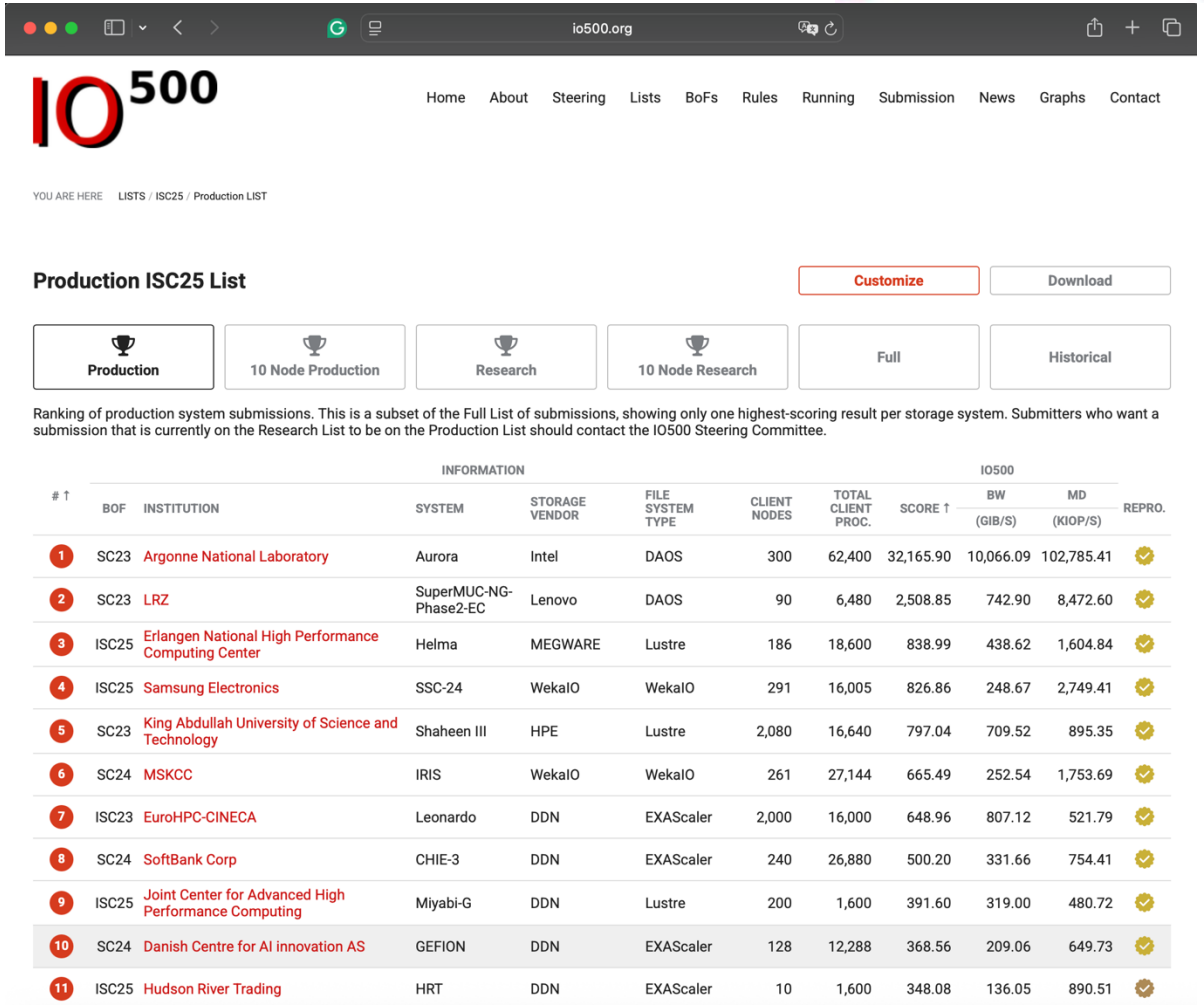
Distributed Asynchronous Object Storage (DAOS) on Aurora

Kaushik Velusamy,
Assistant Computer Scientist,
ALCF

DAOS on Aurora

- DAOS is a major file system in Aurora : 1024 DAOS Nodes, 230 PB, >25 TB/s
- Designed for massively **Distributed** Non Volatile Memory (NVM) and NVMe SSD
- Storage and retrieval of objects in a distributed, parallel, and **Asynchronous** manner.
- Open-source software-defined **Object Store**

- DAOS presents a unified storage model with a native Key-array Value storage interface – POSIX, MPIO, HDF5 etc
- Advanced data protection, self-healing, redundancy, versioning, distribution and fine-grained data control.



#	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 (646,736.2 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_
(2457598019688 4 bytes) in 39.2264s:
597492 MB/s
```

Lustre : 597,492 MB/s

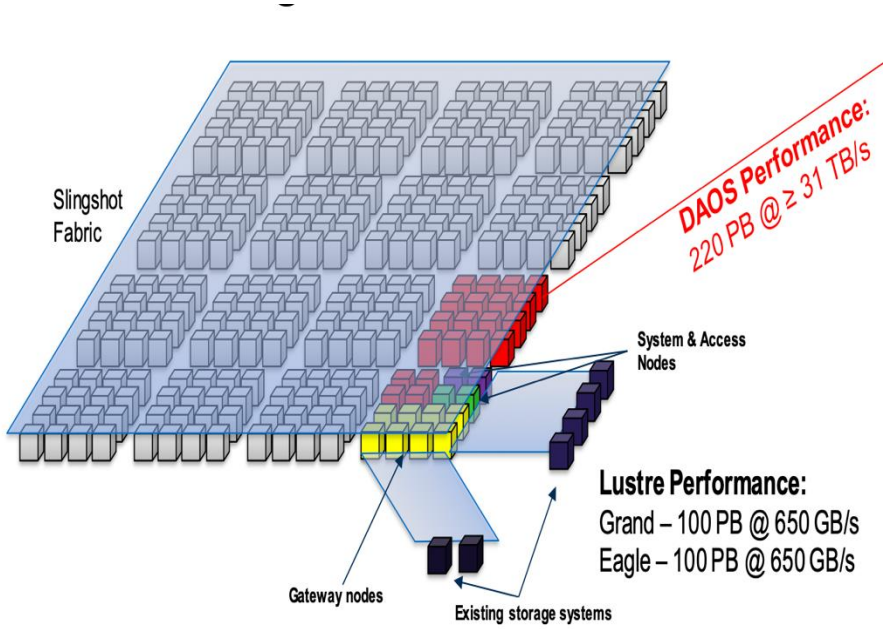
HACC achieved 5.2 TB/s with DAOS, which is close to the IOR peak for the daos_user 128 server cluster
IOR Max Write: 5,723,194.84 MB/s

```
Wrote 9 variables to to
/tmp/datascience/1_fSX_dS1_rd_fac_0
/daos_pos_test_kaus_1_
(2656000016839 2 bytes) in
pure_io_time 4.81287s: 5262890 MB/s
```

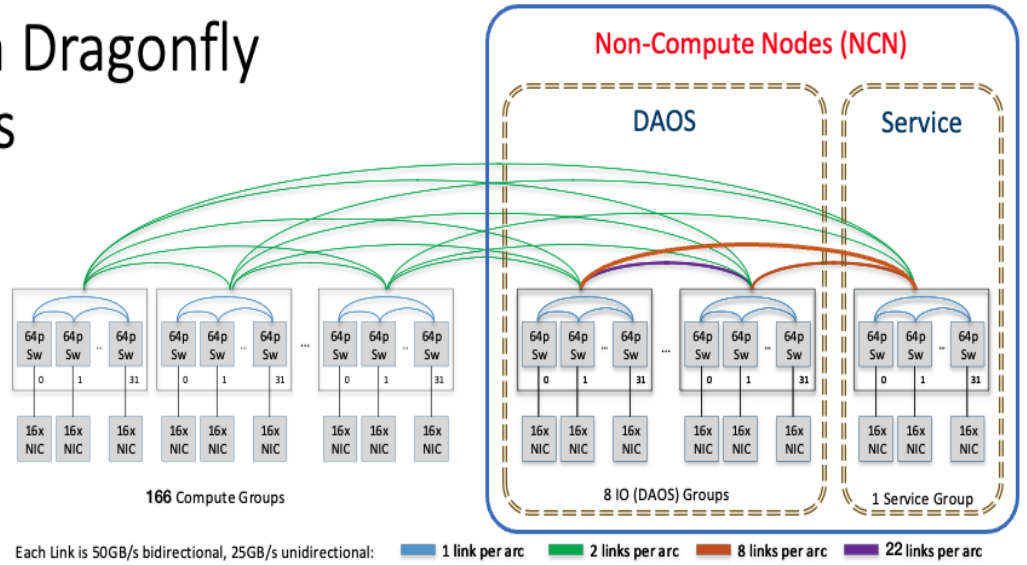
DAOS : 5,262,890 MB/s

TO BE EVENTUALLY INCREASED BY 8X FURTHER

Network Architecture – slingshot fabric - Dragonfly



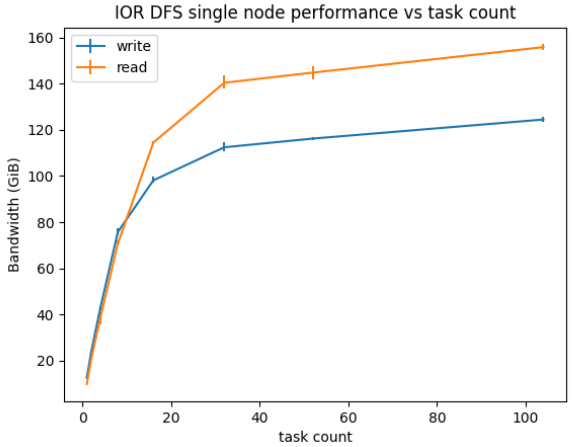
Aurora Dragonfly Groups

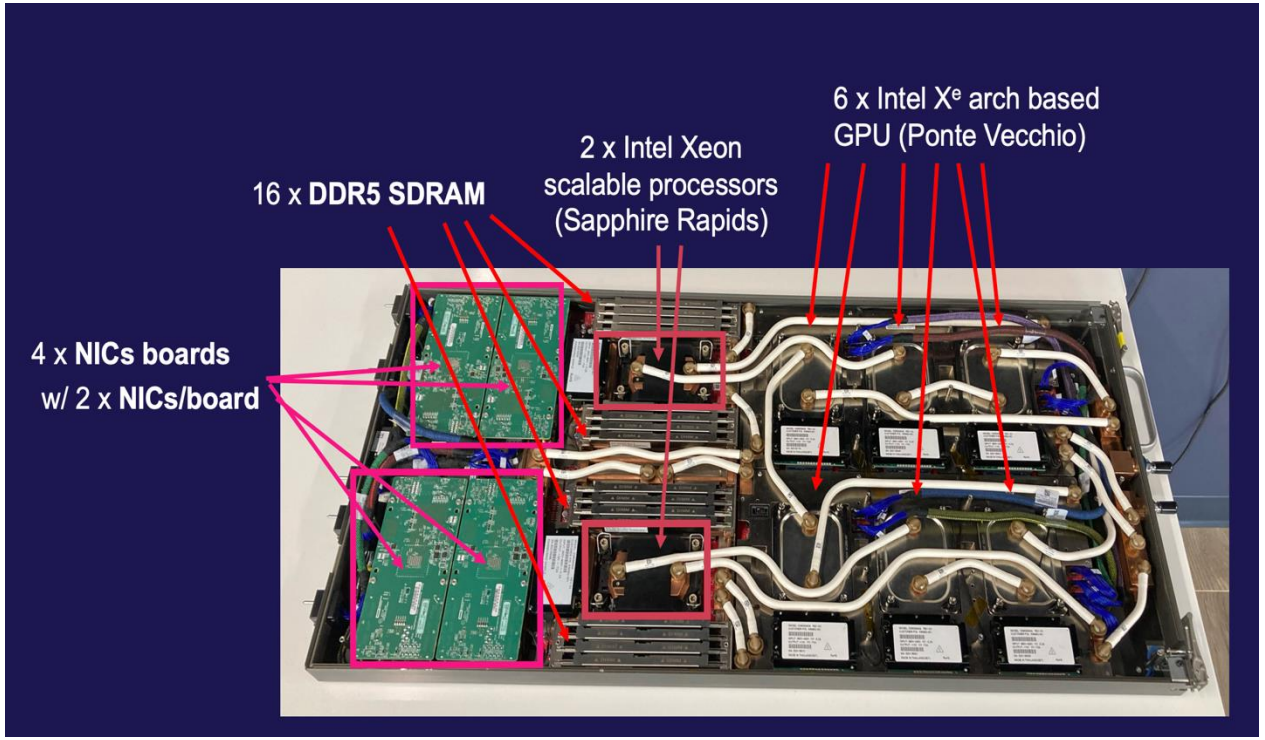


- 1-D Dragonfly Topology - 175 total groups (166 compute + 8 IO + 1 Service)
  - All the global links are optical, all the local links are electrical
  - 2 global links between any two compute groups
  - 22 links between any two IO groups, 8 links between the Service group and each IO group
- All NCN 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

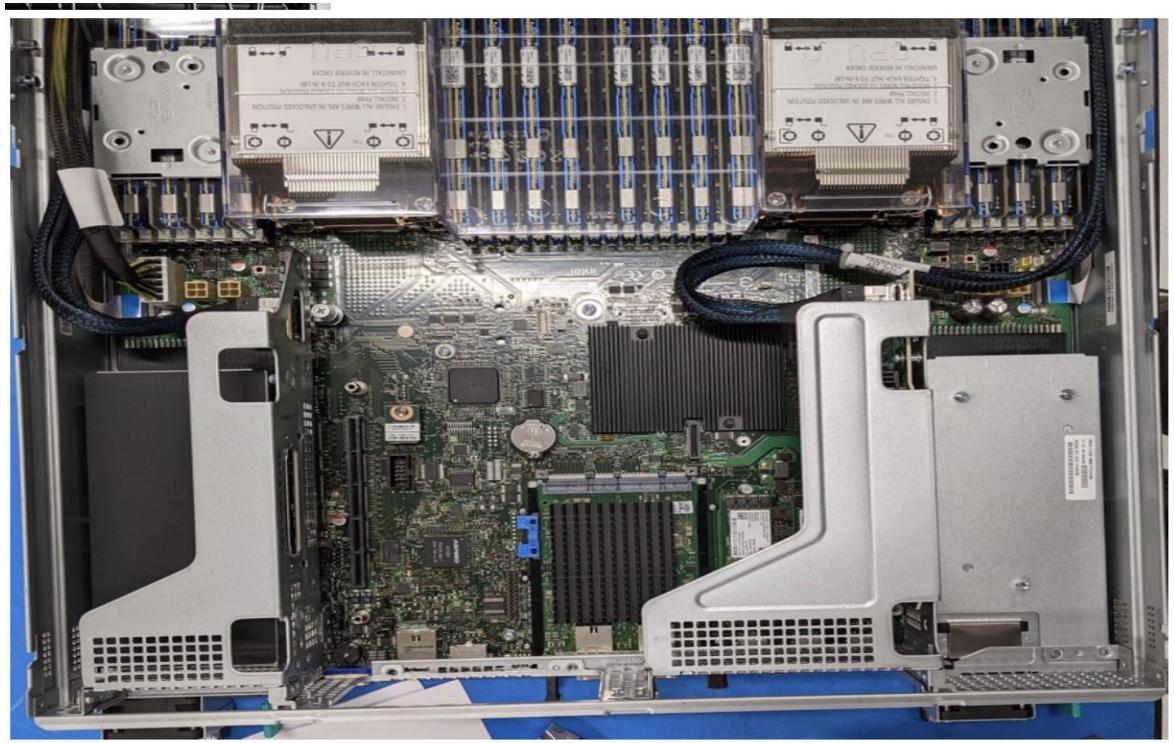
Intel Coyote Pass System - 1024 Total DAOS Servers

(2) Xeon 5320 CPU (Ice Lake) - Each node will run 2 DAOS engines
2048 DAOS engines

(16) 32GB DDR4 DIMMs

(16) 512GB Intel Optane Persistent Memory 200 and (16) 15.3TB Samsung PM1733 - **32 targets**

(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

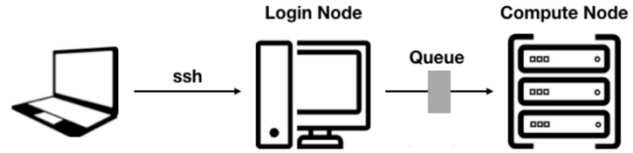
DAOS Pool Allocation

- 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



Job submission without DAOS

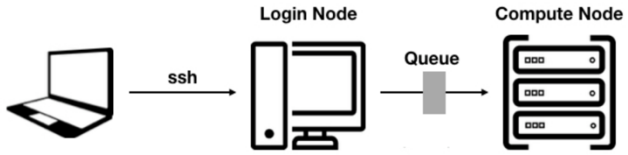
```
qsub   -l select=1
       -l walltime=01:00:00
       -A alcf_training
       -k doe
       -l filesystems=flare
       -q alcf_training
       -I
```

Job submission with DAOS

```
qsub   -l select=1
       -l walltime=01:00:00
       -A alcf_training
       -k doe
       -l filesystems=flare:daos_user_fs
       -q alcf_training
       -I
```

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

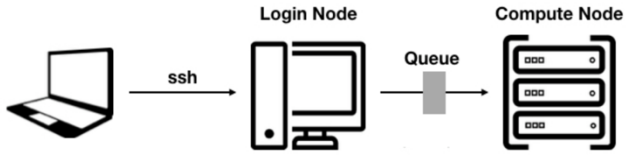
Step 1/5 : Module load daos




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

```
# Optional Validation

echo $DAOS_AGENT_DRPC_DIR
ps -ef | grep daos
clush --hostfile ${PBS_NODEFILE}  'ps -ef | grep agent | grep -v grep'  | dshbak -c
```

Step 2/5 : Verify your pool




```
kaushikvelusamy@aurora-uan-0010:/gecko/Aurora_deployment> daos pool query alcf_training
Pool 1d210bc7-309a-4d75-aea2-5913ff72eac9, ntarget=4064, disabled=0, leader=58, version=1, state=Ready
Pool health info:
- Rebuild idle, 0 objs, 0 recs
Pool space info:
- Target(VOS) count:4064
- Storage tier 0 (SCM):
  Total size: 15 TB
  Free: 14 TB, min:3.4 GB, max:3.5 GB, mean:3.5 GB
- Storage tier 1 (NVMe):
  Total size: 485 TB
  Free: 485 TB, min:119 GB, max:119 GB, mean:119 GB
```

ntarget= 4064 / 32 targets per node = 127 daos cluster size or daos server size

Physically allocated dedicated storage for your project.

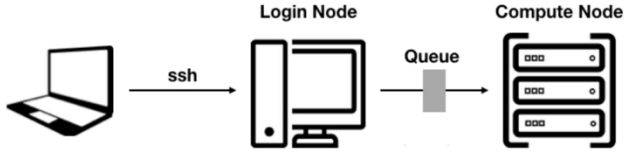
Step 3/5 : Create your container

- DAOS_POOL_NAME=alcf_training
- DAOS_CONT_NAME=INCITE_LLM_AGPT

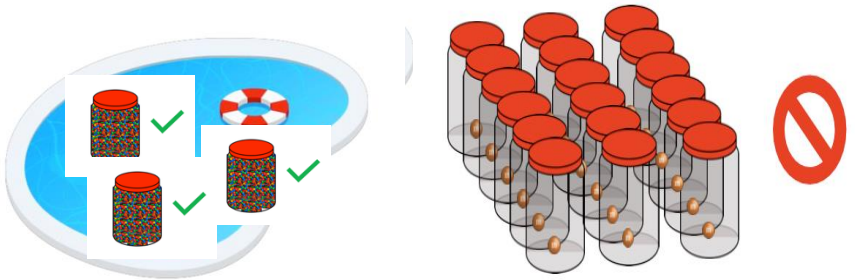
```
daos container create -type=POSIX ${DAOS_POOL_NAME}  ${DAOS_CONT_NAME}

# append optional
          --chunk-size=2097152 --file-oclass=EC_16P2G32  --dir_oclass=RP3G1
          --properties=rd_fac:2,ec_cell_sz:131072,cksum:crc32,srv_cksum:on
```

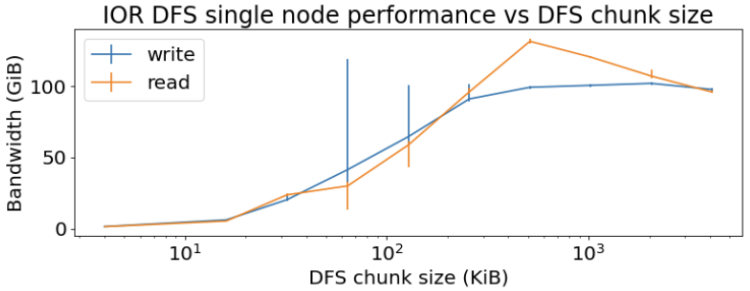
```
  Container UUID : 59747044-016b-41be-bb2b-22693333a380
  Container Label: INCITE_LLM_AGPT
  Container Type : POSIX
Successfully created container 59747044-016b-41be-bb2b-22693333a380
```

- daos pool list
- 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

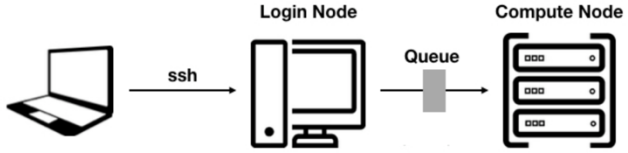


- A pool contains *thousands* of containers
- Basic unit of storage from user perspective





Step 4/5: Mouting your DAOS container



- launched using pdsh/clush on all **compute nodes** and mounted at: `ls /tmp/${DAOS_POOL}/${DAOS_CONT}`

- **`launch-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}  # To mount`**
- `clean-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}  # Optional – To unmount`

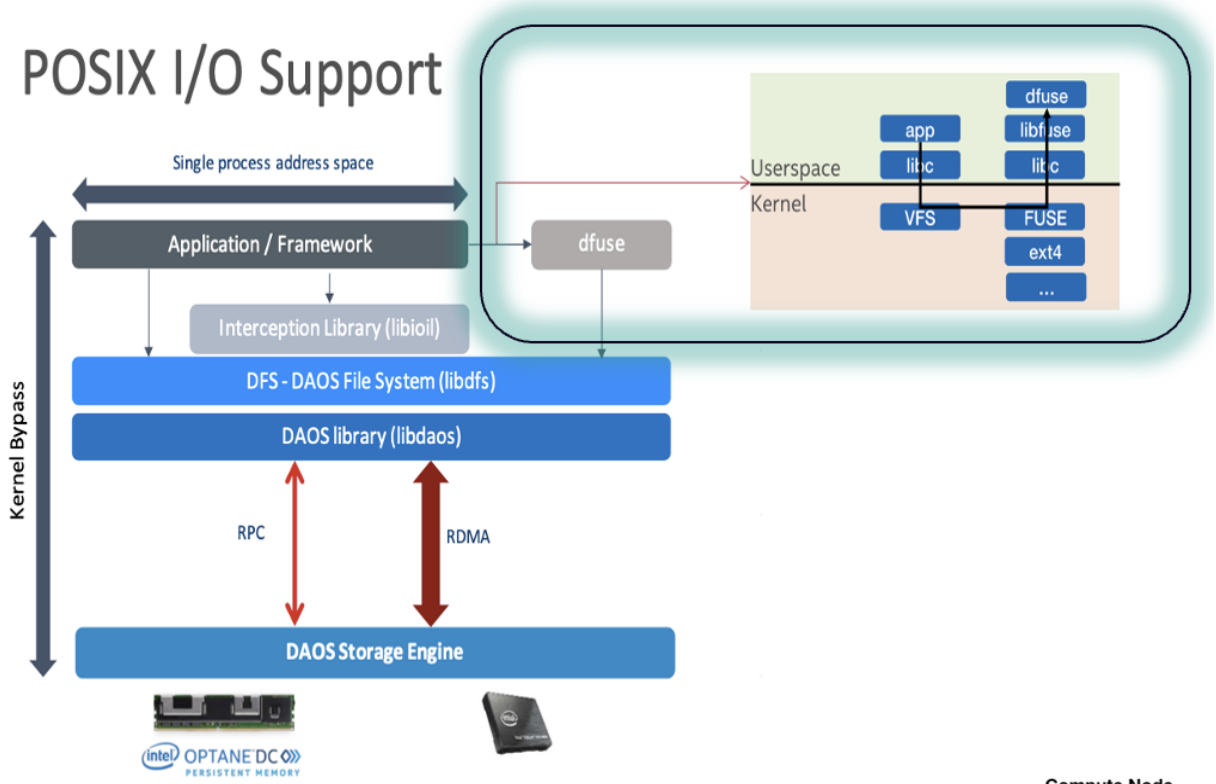
Validation

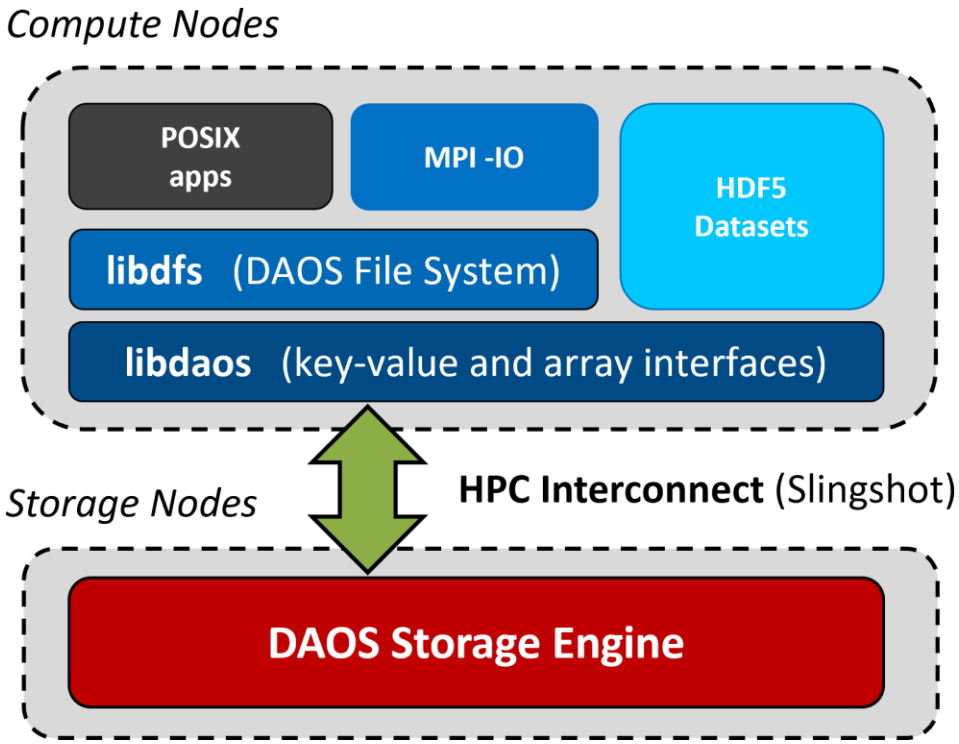
- `mount | grep dfuse`
- `ls /tmp/${DAOS_POOL}/${DAOS_CONT}/`

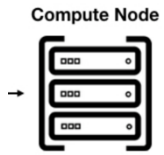
From login node

- `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} # Mandatory`

Step 5/5 : Interfaces & Interception library for POSIX mode and `-no-vni`

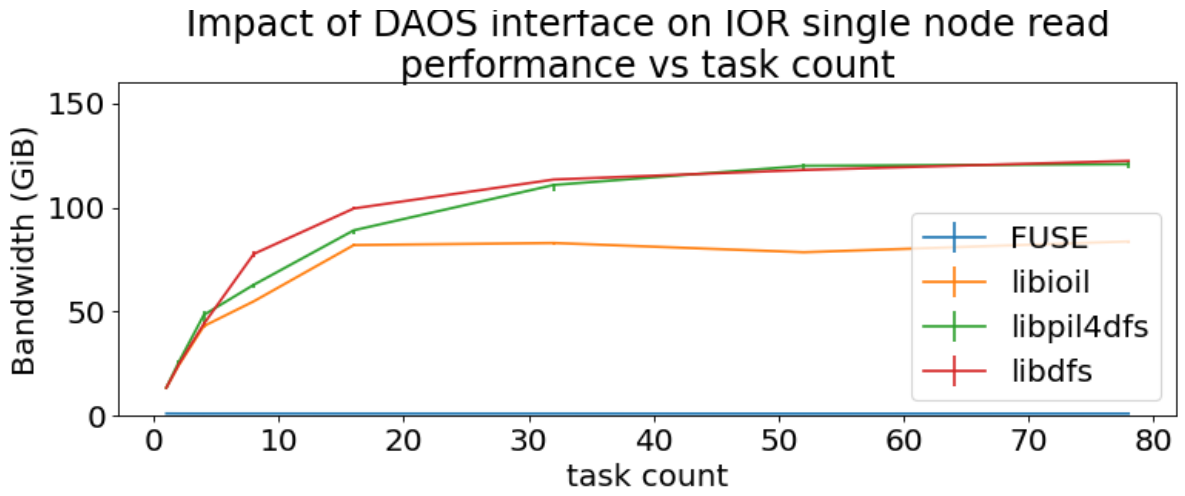


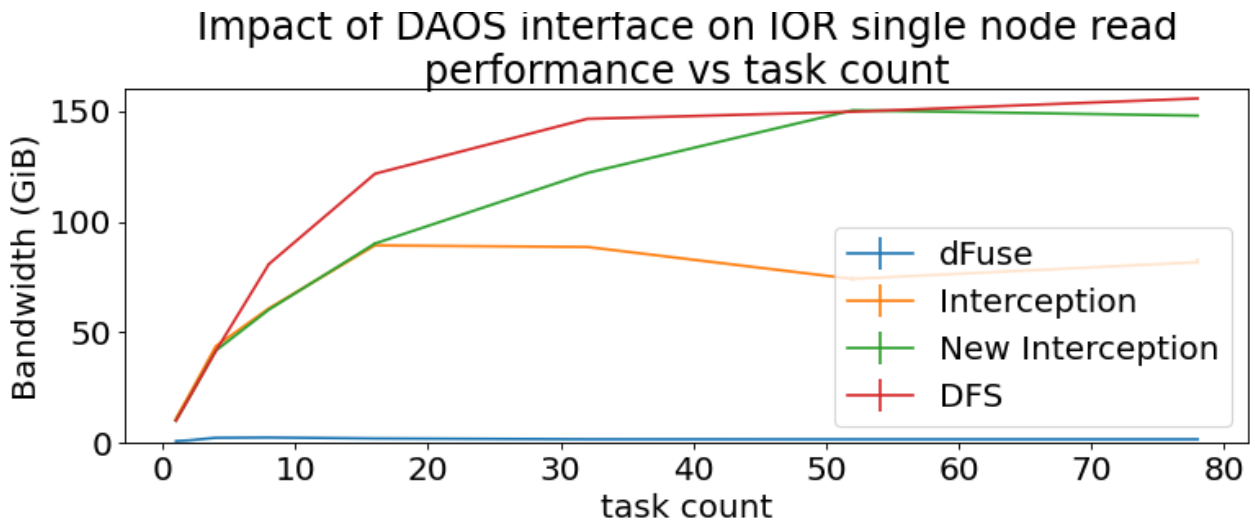




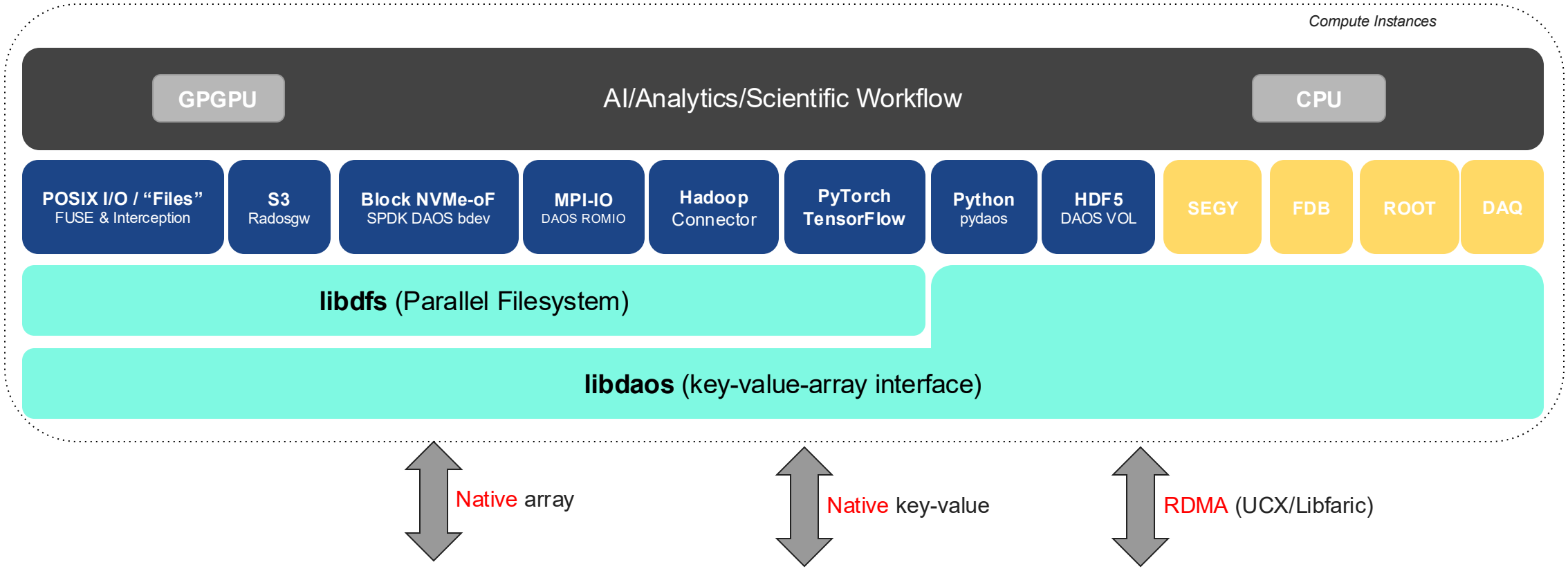
- `mpiexec  # no interception`
- **`mpiexec --env LD_PRELOAD=/usr/lib64/libpil4dfs.so -no-vni`** `# using interception`

`# preferred - both metadata and data are intercepted. This provides close to DFS mode performance.`


Impact of DAOS interface on IOR single node read performance vs task count
Bandwidth (GiB)
150
100
50
0
0
10
20
30
40
50
60
70
80
task count
FUSE
libioil
libpil4dfs
libdfs


Impact of DAOS interface on IOR single node read performance vs task count
Bandwidth (GiB)
150
100
50
0
0
10
20
30
40
50
60
70
80
task count
dFuse
Interception
New Interception
DFS

Software Ecosystem



Generic I/O Middleware/frameworks

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

Types of DAOS containers and interfaces

A. Posix Interface

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

MPIIO and HDF5 users can directly use this interface.

B. DFS interface

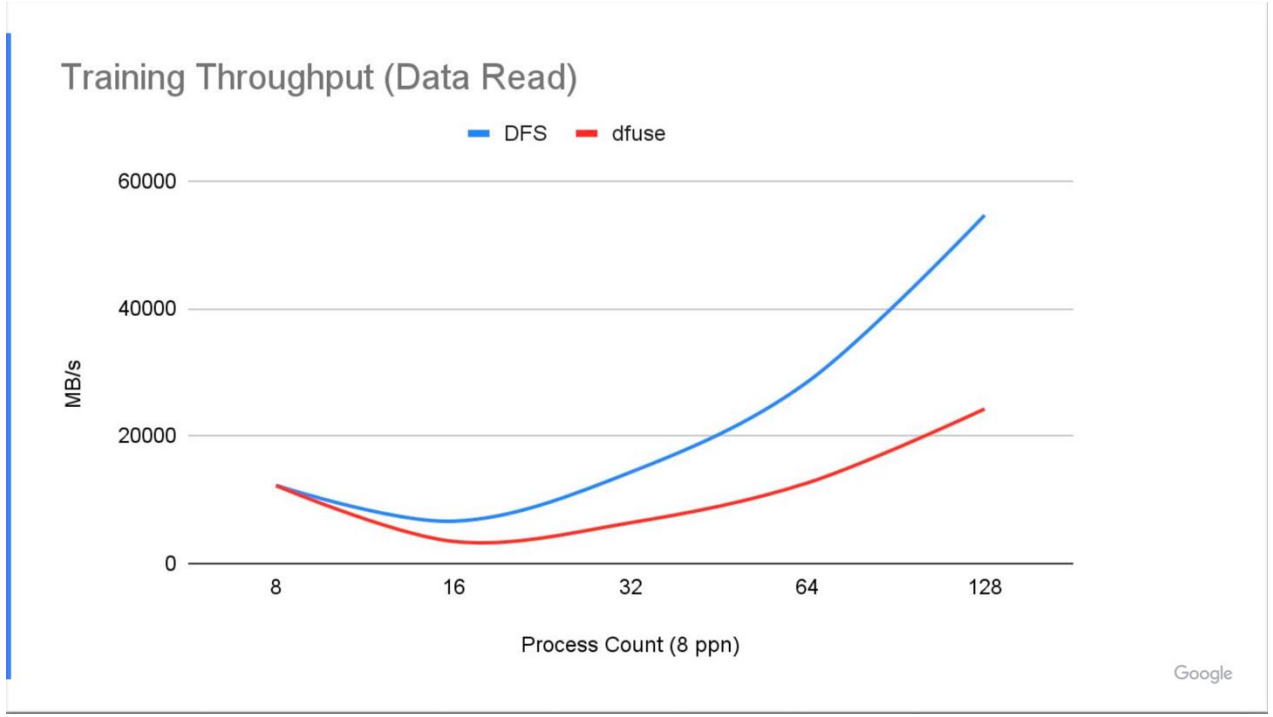
Do not pass the full dfuse path /tmp/poolname/containername,
directly start with / and the file or sub dir past /tmp/poolname/containername

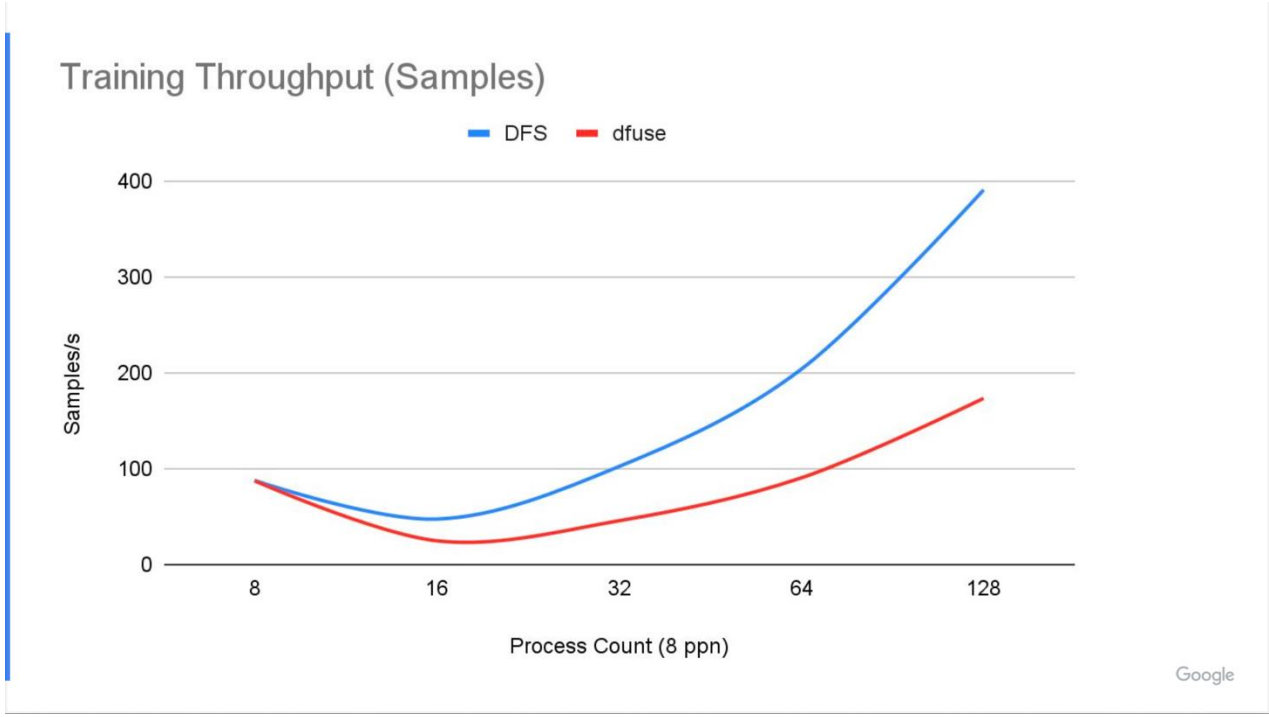
```
/myfile
```

Python PyDAOS interface A/B

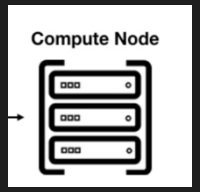
```
import pydaos
import torch as sys_torch
from pydaos.daos_torch import Dataset as DaosDataset
from pydaos.daos_torch import Checkpoint as DaosCheckpoint
from io import BytesIO
```

PyTorch DAOS Data Loader Module

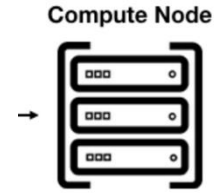




```
#PBS -q prod
#PBS -k doe
#PBS -l filesystems=flare:daos_user
#PBS -l daos=daos_user
# qsub -l select=512:ncpus=208 -l walltime=01:00:00 -A <ProjectName> -l filesystems=flare:daos_user -l daos=daos_user -q prod ./pbs_script.sh or - I

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
```



IOR hands on - ls /soft/daos/examples/daos_peak_test/



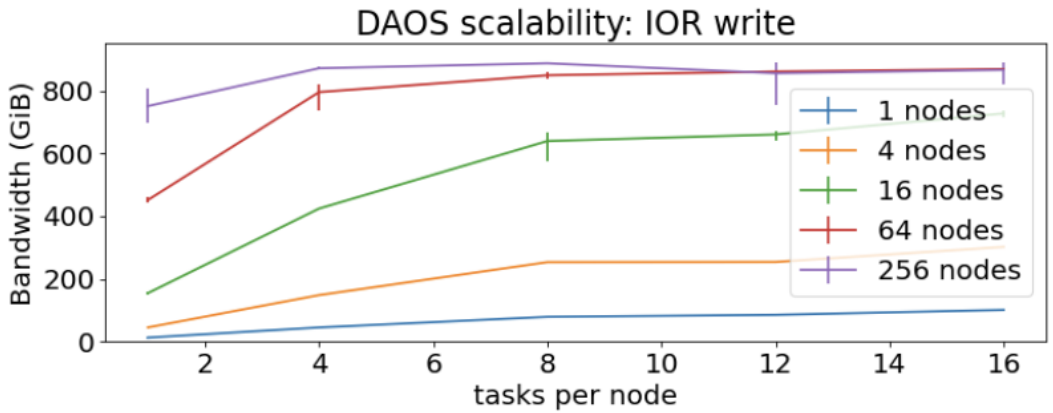

```
mpiexec --env LD_PRELOAD=/usr/lib64/libpil4dfs.so
        -np ${NRANKS} -ppn ${RANKS_PER_NODE} --cpu-bind ${CPU_BINDING}
        --no-vni -genvall
        ior -a posix
        -i 5 -b 1g -t 2m -D 100 -w -r -C -e -v
        -o /tmp/${DAOS_POOL}/${DAOS_CONT}/io_file.dat
```

```
mpiexec
        -np ${NRANKS} -ppn ${RANKS_PER_NODE} --cpu-bind ${CPU_BINDING}
        --no-vni -genvall
        ior -a DFS --dfs.pool=${DAOS_POOL} --dfs.cont=${DAOS_CONT}
        -i 5 -b 1g -t 2m -D 100 -w -r -C -e -v
        -o /io2.dat
```

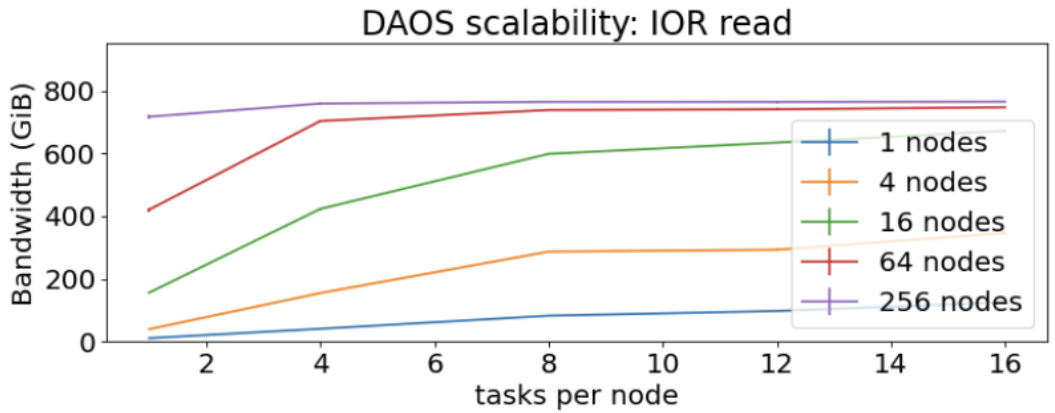
To get peak perf use at least 192 nodes - wall time 5 mins

```
#readme for IOR
# -F file-per-process - No -F is single shared file [This is an I/O Access parameter]
# -c collective I/O [I/O type parameter] with -a mpiio and add daos:/tmp/$DAOS_POOL/$DAOS_CONT/io.dat in -o
# No -c independent I/O [I/O type parameter] with -a posix and just /tmp/$DAOS_POOL/$DAOS_CONT/io.dat in -o
# -C reorderTasksConstant - changes task ordering to n+1 ordering for readback
# -e fsync - perform fsync upon POSIX write close
# -D N deadlineForStonewalling - seconds before stopping write or read phase
# -b N blockSize - contiguous bytes to write per task (e.g.: 8, 4k, 2m, 1g)
# -t N transferSize - size of transfer in bytes (e.g.: 8, 4k, 2m, 1g)
# -i N repetitions - number of repetitions of test
```

```
IOR_INSTALL=/soft/daos/examples/daos_peak_test/ior_mdtest_install_bin/
export LD_LIBRARY_PATH=$IOR_INSTALL/lib:$LD_LIBRARY_PATH
export PATH=$IOR_INSTALL/bin:$PATH
```

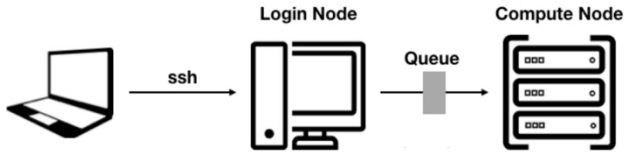


(a) Write



(b) Read

Moving data from lustre to DAOS



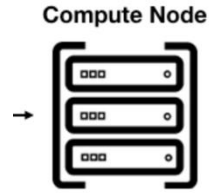

```
cp
/lus/flare/projects/CSC250STDM10_CNDA/kaushik/thundersvm/input_data/real-sim_M100000_K25000_S0.836
/tmp/${DAOS_POOL}/${DAOS_CONT}
```

```
daos filesystem copy
--src /lus/flare/projects/CSC250STDM10_CNDA/kaushik/thundersvm/input_data/real-sim_M100000_K25000_S0.836
--dst daos://${DAOS_POOL}/${DAOS_CONT}/path_starting_from_within_container/
# mounting not required here
```

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
```

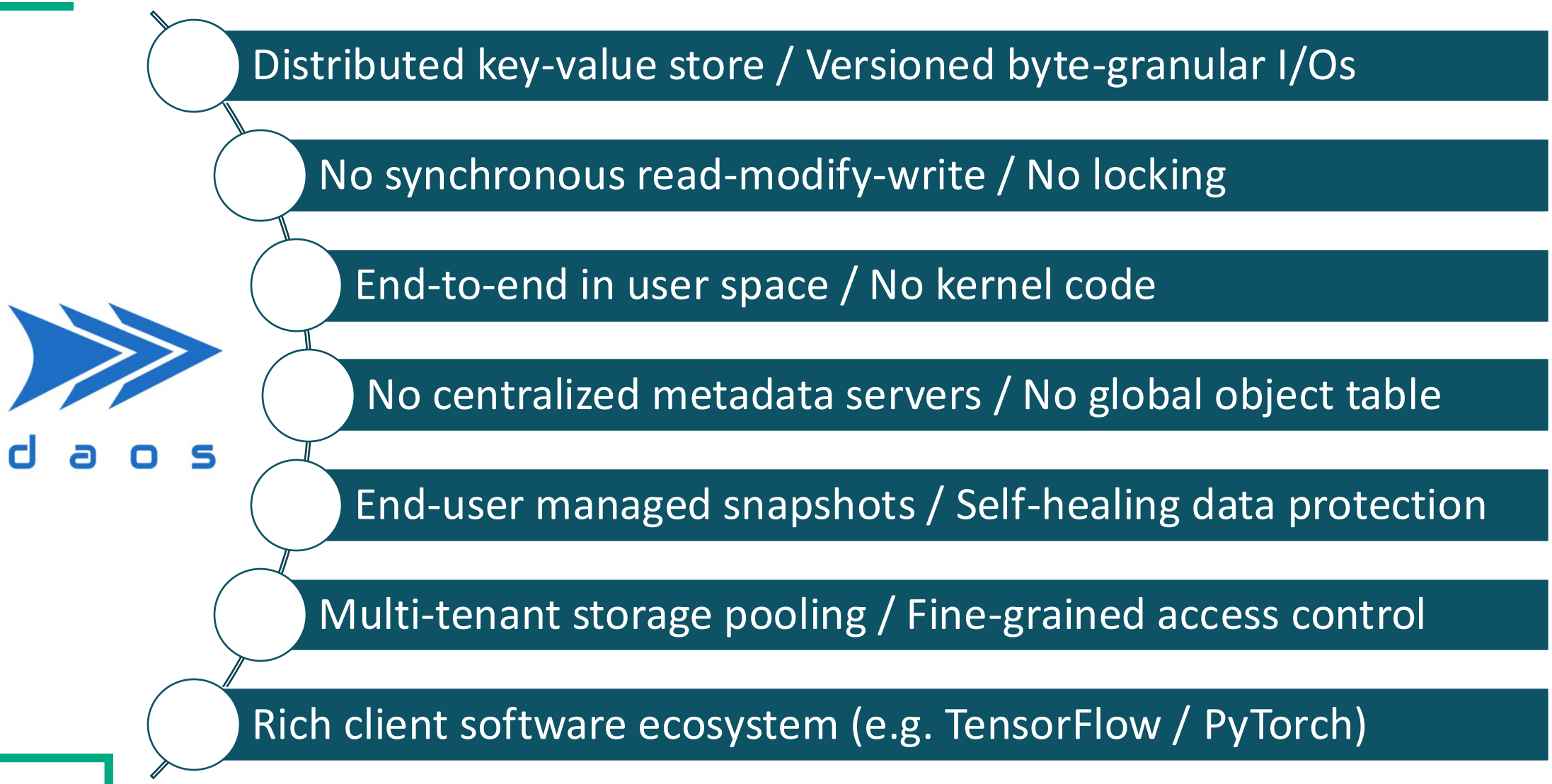
Parallel copy (splitting both large dirs/files) using DAOS-aware mpiFileUtils

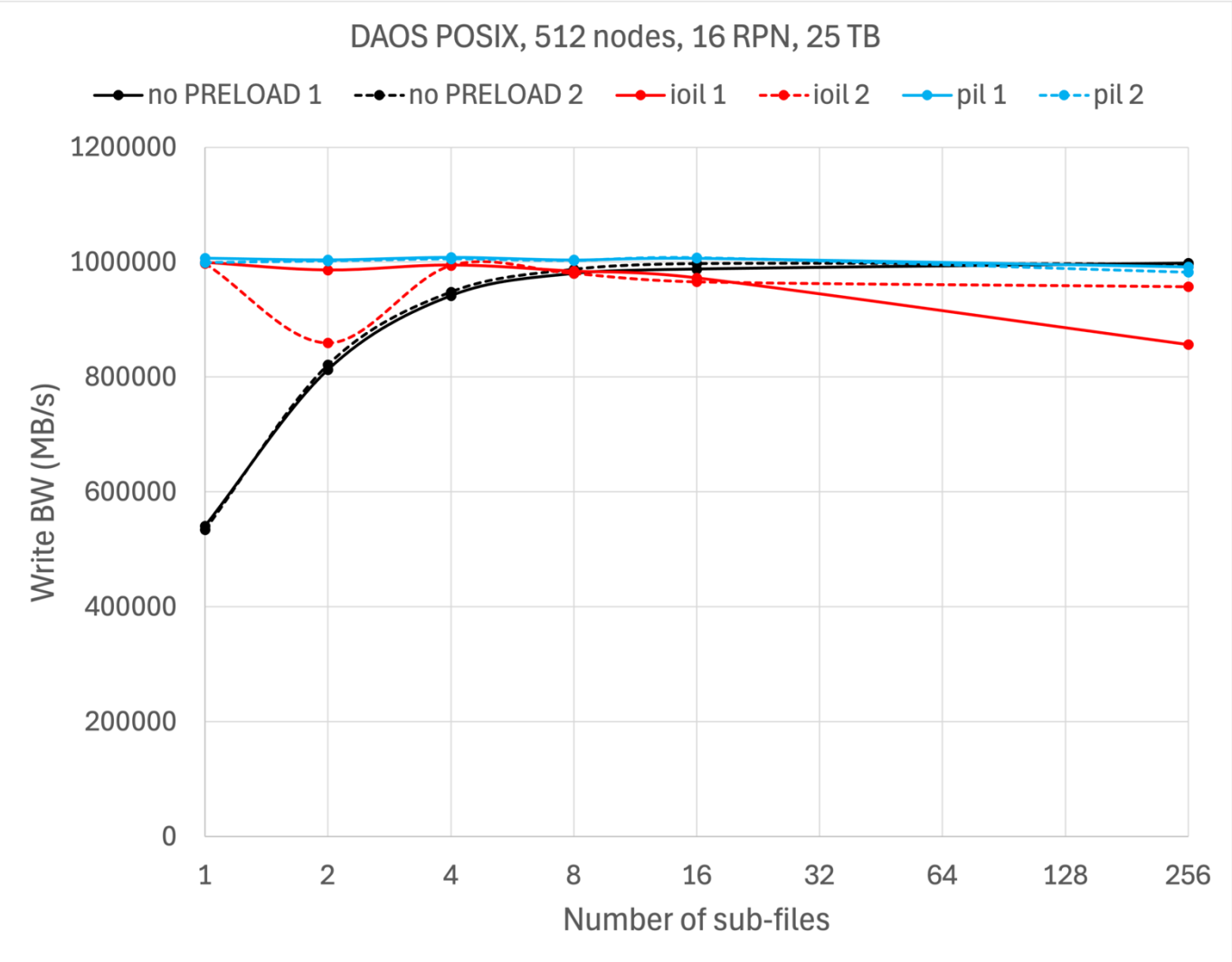



```
kaushikvelusamy@x4210c6s0b0n0:/tmp> mpiexec --env LD_PRELOAD=/usr/lib64/libpil4dfs.so  -np 16 -ppn 16 --cpu-bind
list:4:5:6:7:8:9:10:11:56:57:58:59:60:61:62:63 /soft/daos/mpifileutils/bin/dcp  source/ /tmp/datascience/1_fSX_dS1_rd_fac_0/
[2025-05-17T04:08:39] Walking /tmp/source
[2025-05-17T04:08:39] 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)
```

What makes DAOS different from Lustre and other file systems?

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




DAOS POSIX, 512 nodes, 16 RPN, 25 TB
no PRELOAD 1
no PRELOAD 2
ioil 1
ioil 2
pil 1
pil 2
1200000
1000000
800000
600000
400000
200000
0
Write BW (MB/s)
1
2
4
8
16
32
64
128
256
Number of sub-files

DAOS Foundation

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

- Argonne : Kevin Harms, Paul Coffman, Alex Kulyavtsev, Huihuo Zheng, Venkatram Vishwanath, Rick Wagner
- HPE DAOS team: Mohamad Chaarawi, Johann Lombardi, John Carrier, Samirkumar Raval
- 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.