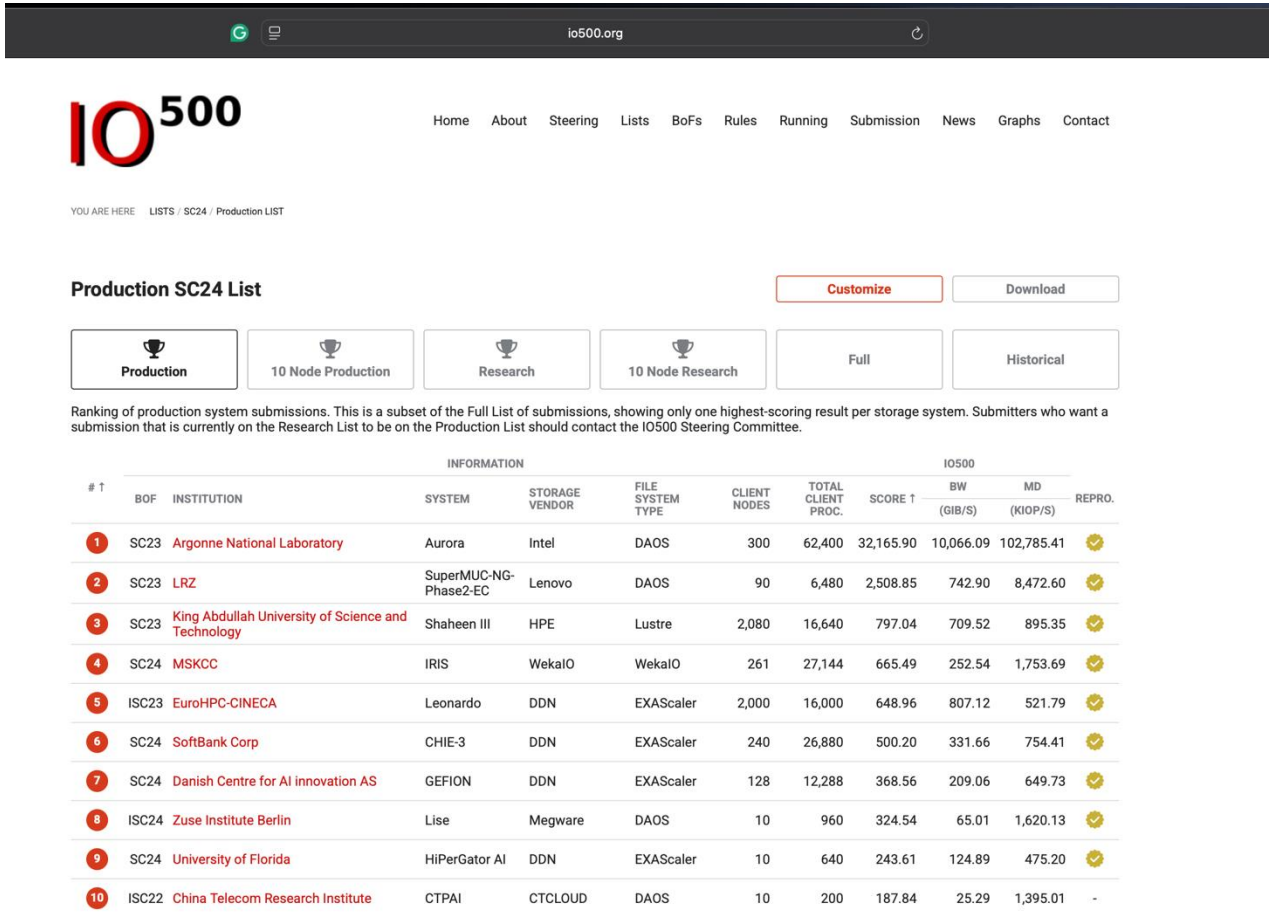
Overview of DAOS

Distributed Asynchronous Object Store

- Kaushik Velusamy

Why DAOS

- 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.



#	BOF	INSTITUTION	SYSTEM	STORAGE VENDOR	FILE SYSTEM TYPE	CLIENT NODES	TOTAL CLIENT PROC.	IO500 SCORE	IO500 BW (GIB/S)	IO500 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	SC23	King Abdullah University of Science and Technology	Shaheen III	HPE	Lustre	2,080	16,640	797.04	709.52	895.35	
4	SC24	MSKCC	IRIS	WekaIO	WekaIO	261	27,144	665.49	252.54	1,753.69	
5	ISC23	EuroHPC-CINECA	Leonardo	DDN	EXAScaler	2,000	16,000	648.96	807.12	521.79	
6	SC24	SoftBank Corp	CHIE-3	DDN	EXAScaler	240	26,880	500.20	331.66	754.41	
7	SC24	Danish Centre for AI innovation AS	GEFION	DDN	EXAScaler	128	12,288	368.56	209.06	649.73	
8	ISC24	Zuse Institute Berlin	Lise	Megware	DAOS	10	960	324.54	65.01	1,620.13	
9	SC24	University of Florida	HiPerGator AI	DDN	EXAScaler	10	640	243.61	124.89	475.20	
10	ISC22	China Telecom Research Institute	CTPAI	CTCLOUD	DAOS	10	200	187.84	25.29	1,395.01	-

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

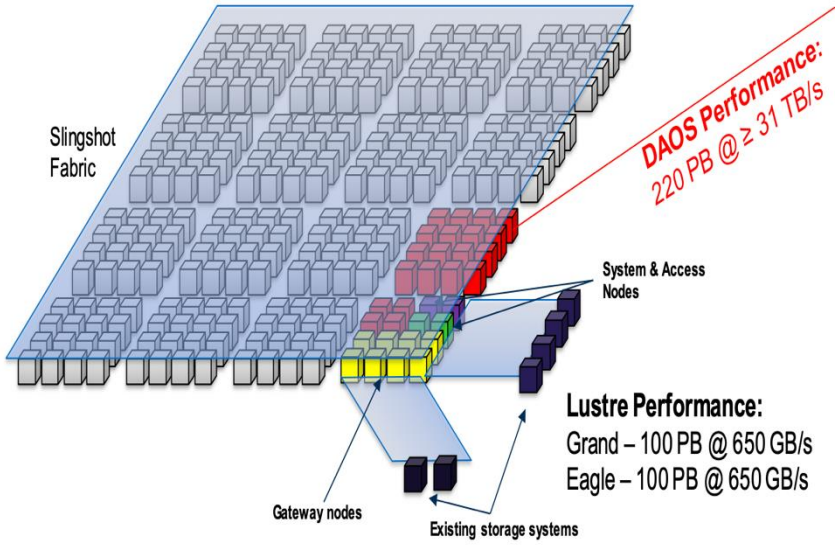
Lustre

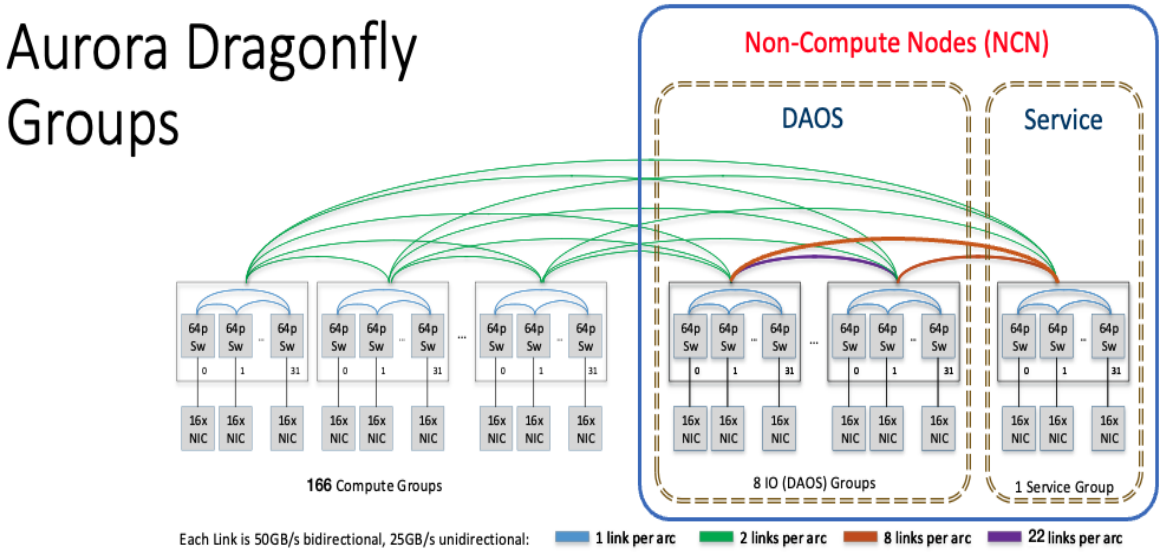
- Flare is a **91 PB** Lustre Filesystem with 160 OSTs, 40 MDTs, and 48 Gateway nodes mounted at /lus/flare/projects/ with a peak theoretical performance of **650 GB/s.**

  You should launch jobs only from this Flare space.

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

Network Architecture – slingshot fabric - Dragonfly

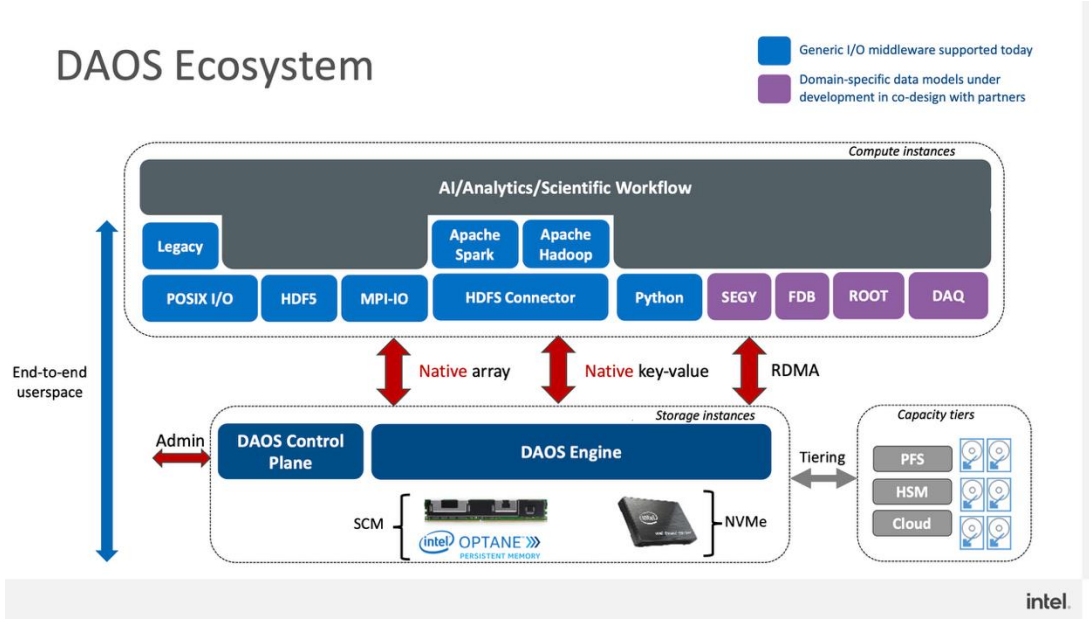


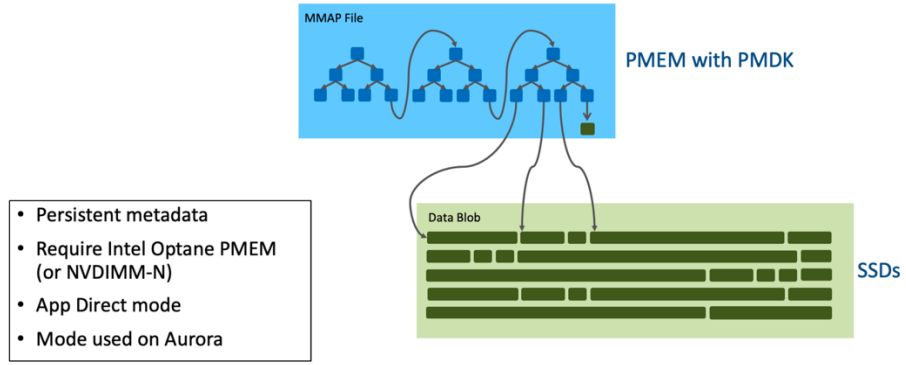


- 1-D Dragonfly Topology - 175 total groups (166 compute + 8 IO + 1 Service)
  - All the global links are optical, all the local links are electrical
  - 2 global links between any two compute groups
  - 22 links between any two IO groups, 8 links between the Service group and each IO group
- All NCN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

DAOS Ecosystem and Design Fundamentals

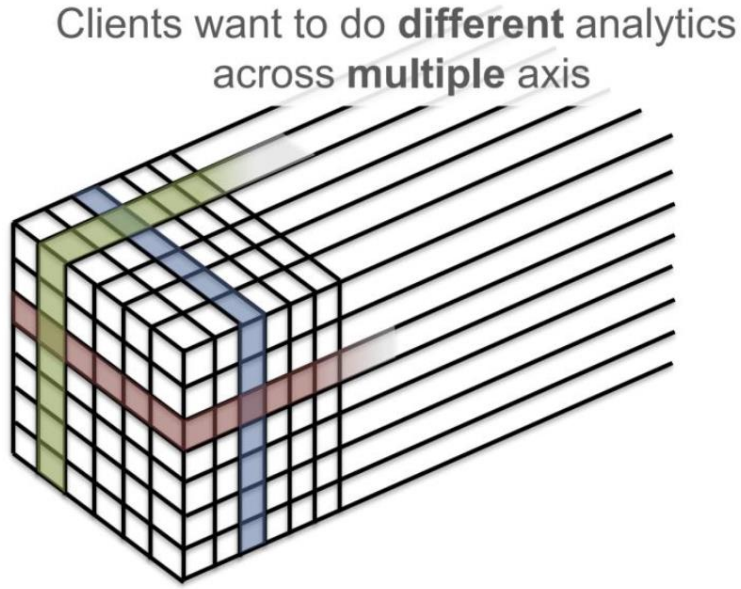
- Efficient for unstructured data
- Efficient for accessing small data.
- High bandwidth, low latency, and IOPS
- No read-modify-write on I/O path (use versioning)
- No locking
- No client tracking or client recovery
- No centralized (meta)data server
- No global object table

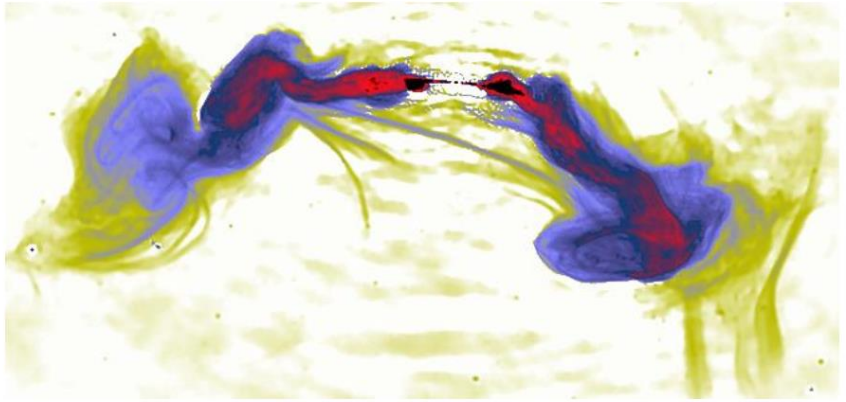




Object Stores

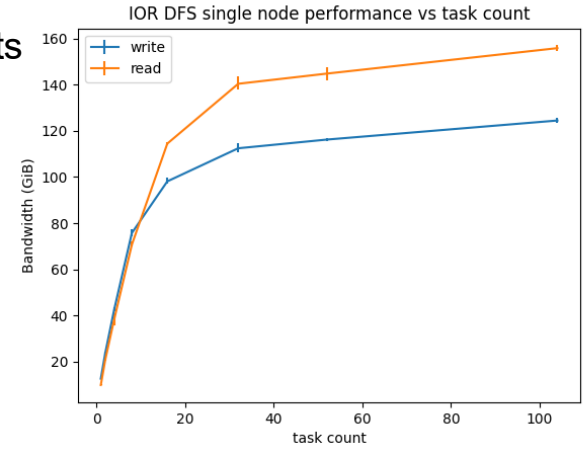
- Object Stores can unlock previously expensive I/O patterns
- Supports different creation, querying, analysis, and use patterns
- Data retrieval leverages metadata - Build structure on the fly
- Weather/climate - Simulation (data generation) only one part Consumption workloads different layout/pattern from production.
- Radio astronomy- Data collected and stored by antenna (frequency and location) and capture time. Reconstruction of images done in time order.
- Evaluation of transients or other phenomenon undertaken across frequency and location.

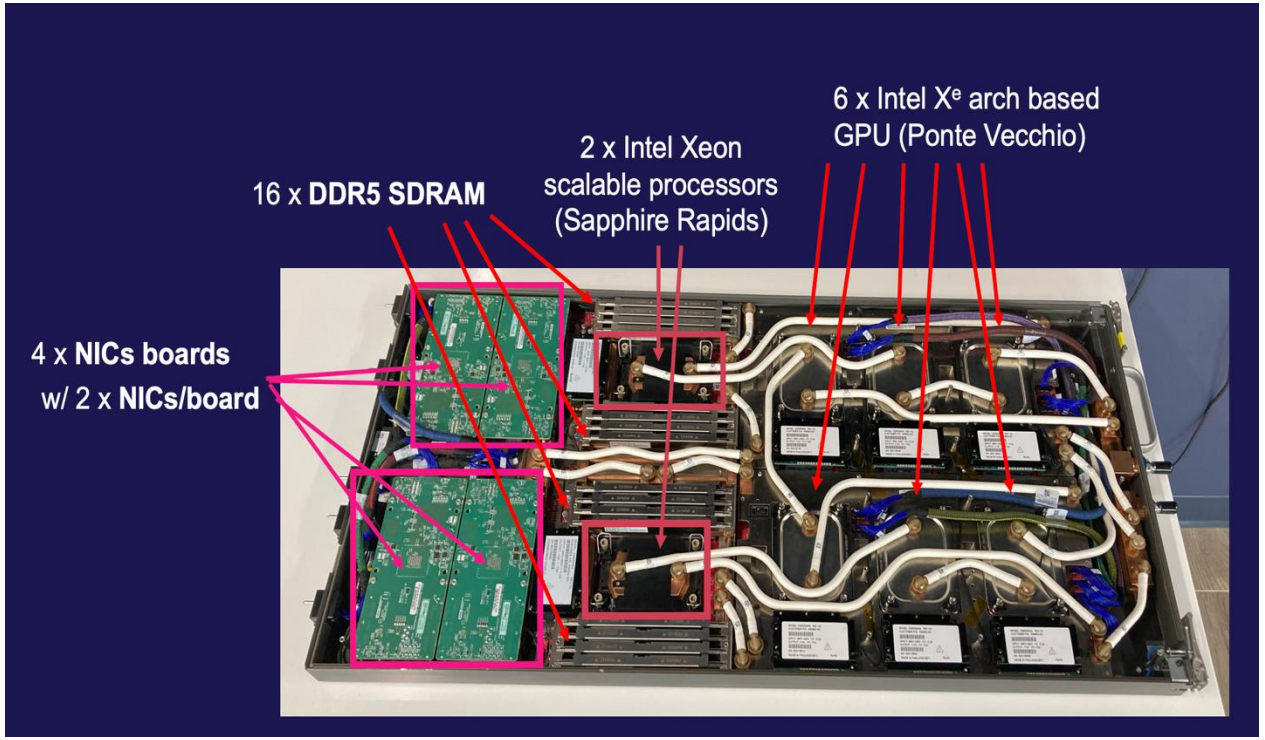




A single Aurora compute node

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



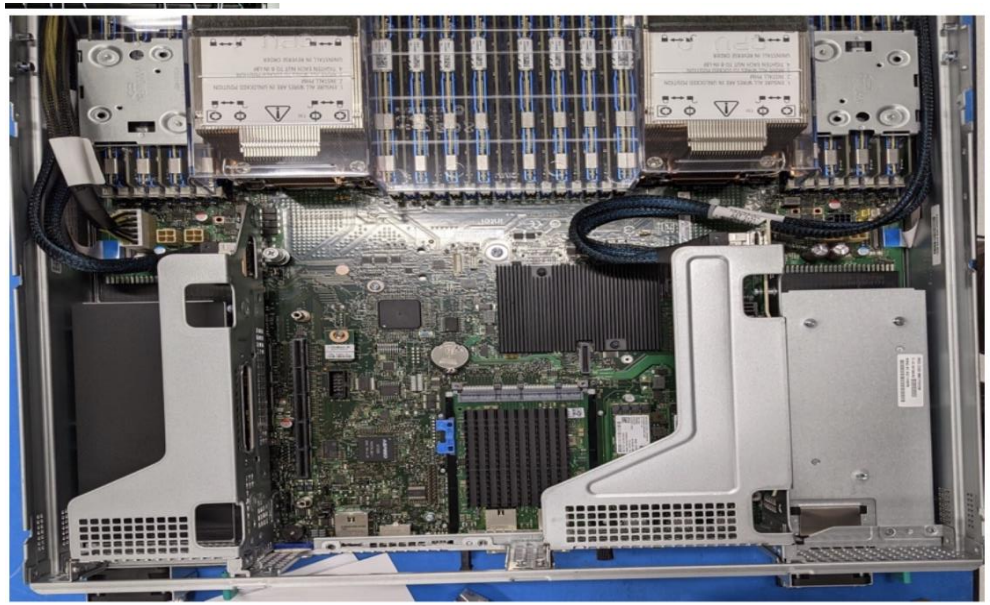


A single DAOS storage node

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
```

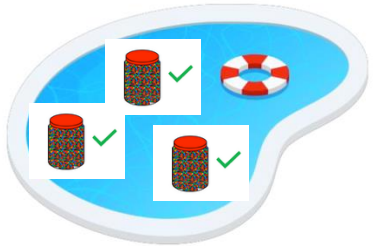
Step 2/5 : Verify your pool

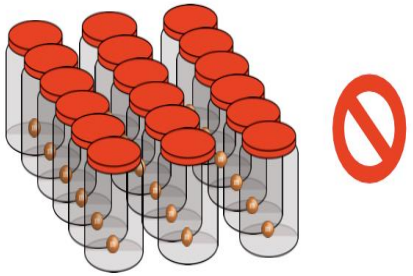
Physically allocated dedicated storage for your project.

```
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
```

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

  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}`**
- `clean-dfuse.sh  ${DAOS_POOL_NAME}:${DAOS_CONT_NAME}`
- `fusermount3 -u /tmp/${USER}/${DAOS_POOL}/${DAOS_CONT}`

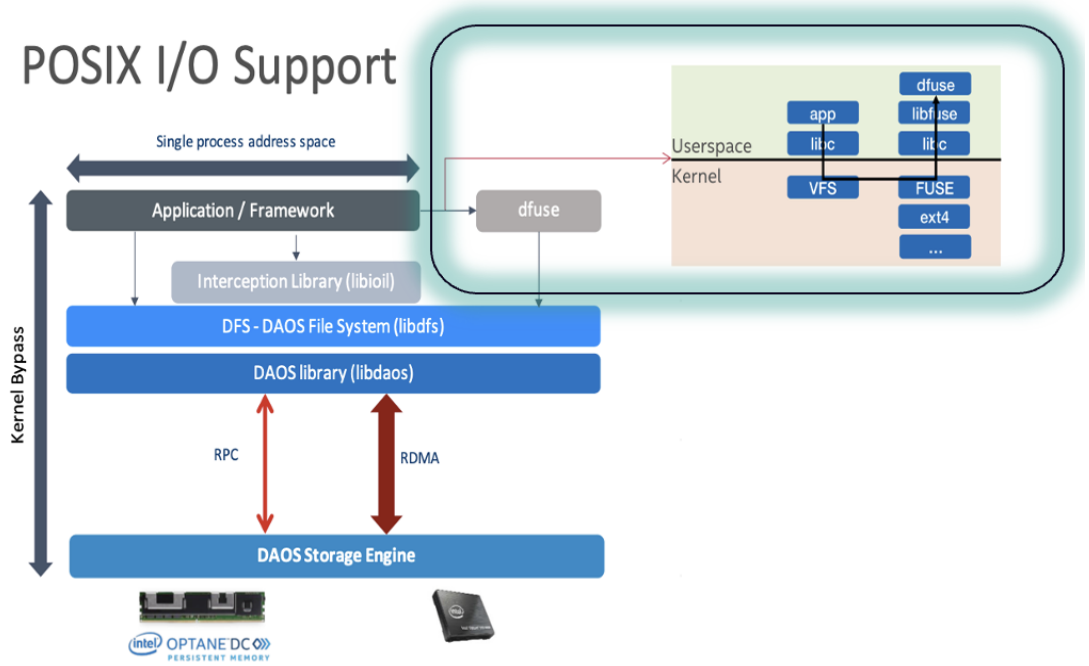
From compute node

- 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.
```

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
```

Job Submission

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
```

 # replace `./pbs_script1.sh` with `-I` for an interactive job

```
#!/bin/bash -x
#PBS -l select=512
#PBS -l walltime=01:00:00
#PBS -A <ProjectName>
#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
```

Recent results from Hardware Accelerated Cosmology Code (HACC)

HACC achieved Lustre peak theoretical max 600GB/s

```
NUM_OF_NODES=512  TOTAL_NUM_RANKS=6144  RANKS_PER_NODE=12 RecordSize = 38 NpTotal = 646736283406 (646736.283406 million particles)
24575978769428 bytes uncompressed data (24,575,978.769428 MB)
Wrote 9 variables to ./lustre_out_712762.aurora-pbs-0001.hostmgmt.cm.aurora.alcf.anl.gov_512/lus_pos_test_kaus_1_ (24575980196884 bytes) in 39.2264s: 597492 MB/s
```

HACC achieved 5.2 TB/s with DAOS, which is close to IOR peak for **daos_user 128** server cluster Max Write: 5723194.84 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
```

Lustre : 597 492 MB/s

DAOS : 5 262 890 MB/s

Reference

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

alcf.anl.gov/events/alcf-incite-gpu-hackathon

Tuesday May 13	Topics	Speakers
10:00 AM - 10:45 AM	DAOS on Aurora: Speeds and Feeds and Use Cases	Johann Lombardi, HPE
10:45 AM -11:30 AM	DAOS with Dfuse and ILs and MPI-IO	Johann Lombardi, Mohamad Chaarawi, HPE
11:30 AM - 12:15 PM	DAOS Tuning and Best Practices	Mohamad Chaarawi, HPE
12:15 PM - 1:00PM	Lunch Break	
1:00PM -1:20 PM	Advanced Middleware	Johann Lombardi, Mohamad Chaarawi, HPE
1:20 PM -1:50 PM	Monitoring with Darshan	Kaushik Velusamy, ANL
1:50 PM -2:20 PM	Q&A	
2:20 PM	Adjourn	

Acknowledgements

- Gordon Mcpheeters, Kevin Harms, Paul Coffman, Huihuo Zheng, Venkatram Vishwanath
- HPE DAOS team: Mohamad Chaarawi, Johann Lombardi, 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.