DAOS

Distributed Asynchronous Object Store

Part I : Theory and getting started - Jan 25, 2024
Part II: Hands on - Feb 1, 2024

Kaushik Velusamy

Tutorial prepared for ALCF data science team.

Tutorial Agenda

1. Why DAOS
2. DAOS architecture
3. Compute and IO Network architecture
4. Terminologies: What is a target, a pool and a container?

5. Exercise 0: Daos sanity tests
6. Exercise 1: Creating and querying a container
7. Exercise 2: Interacting with daos from Login node vs compute node
8. Exercise 3: Your first DAOS job script

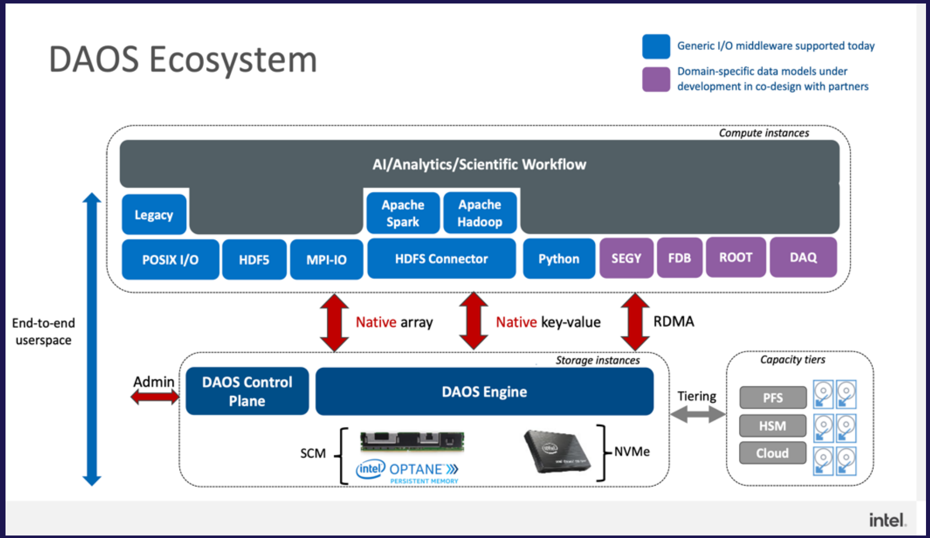
9. Exercise 4: Lustre Vs Daos Single node NIC
10. Exercise 5: Lustre Vs Daos Single node Request Size
11. Exercise 6: Lustre Vs Daos multi node Scaling
12. Exercise 7: Lustre Vs Daos metadata
13. Exercise 8: Lustre Vs Daos Single node VPIC-IO h5bench
14. Extras : Debugging a container and fine tuning
15. Give script and readme file

1. Why DAOS

- DAOS is a major file system in Aurora : 230 PB, >25 TB/s, 1024 DAOS Nodes
- 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.

Efficient for unstructured data.
Efficient for accessing small data.
High bandwidth, low latency, and IOPS

Currently available as of Jan 24, 2024
Aurora Lustre Gecko : 12PB with 96 OSTs
Aurora DAOS : 20 / 1024 DAOS nodes with 5PB of NVMe SDD & 160 TB SCM
What we share in data science pool: 75 TB NVMe SSD and 1.5 TB SCM
** Will be increased when we move to 1024 DAOS nodes.*



System	Capacity	Performance	System
DAOS	220 PB @ EC16+2 ▪ 250 PB NVMe ▪ 8 PB Optane PMEM	≥ 25 TB/s Read & Write	Aurora
Eagle	100 PB @ RAID6 ▪ 8480 HDD ▪ 40 Lustre MDT	> 650 GB/s Read & Write	Aurora and Polaris
Grand	100 PB @ RAID6 ▪ 8480 HDD ▪ 40 Lustre MDT	> 650 GB/s Read & Write	Aurora and Polaris
Local	3.2 TB/node • 1.8 PB agg	~ 3 GB/s Read & Write per node 1.7 TB/s aggregate	Polaris

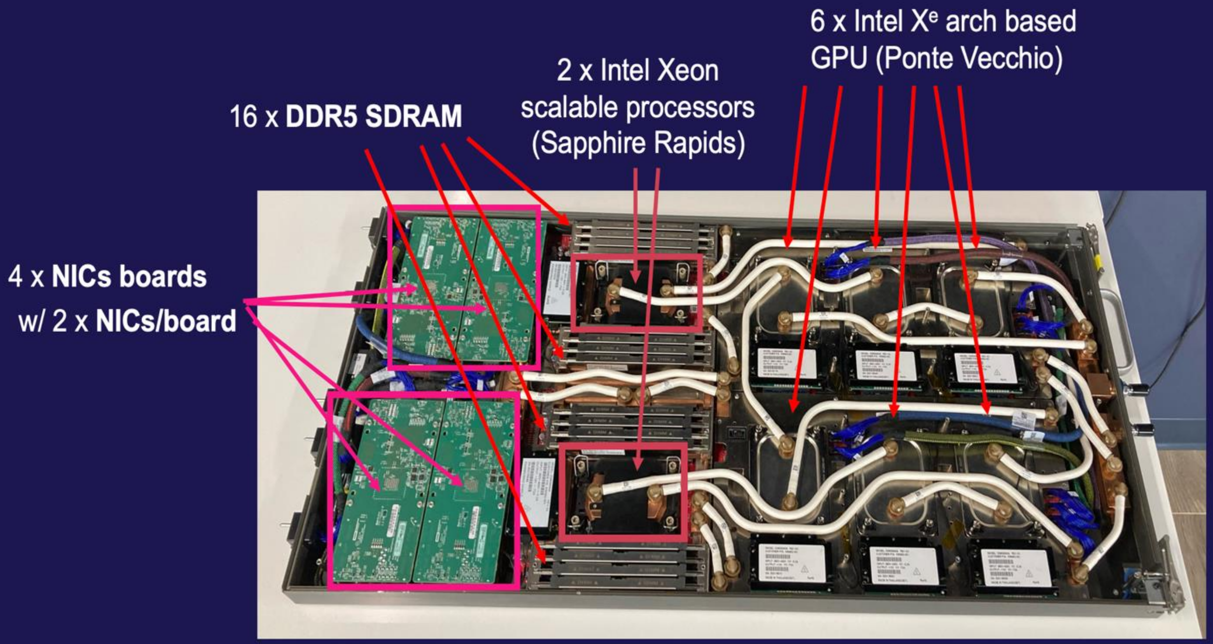
Aurora DAOS : #1 in the IO500 Bandwidth with 260 / 1024 DAOS nodes
Easy: 8 TiB/s (write) and 7.9 TiB/s (read)
Hard: 4.8TiB/s (write) and 4.3 TiB/s (read)

A single compute node

8 NICs
52 cores per socket
2 sockets

25GB/s X 8 NICs = 200 GB/s

More info on binding at https://docs.alcf.anl.gov/aurora/running-jobs-aurora/

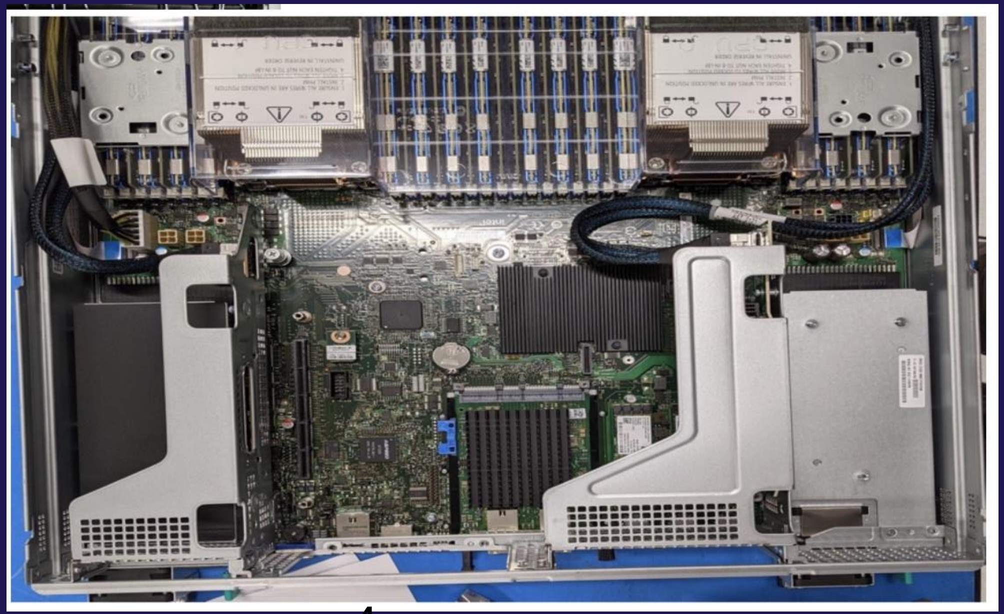


A single storage node

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

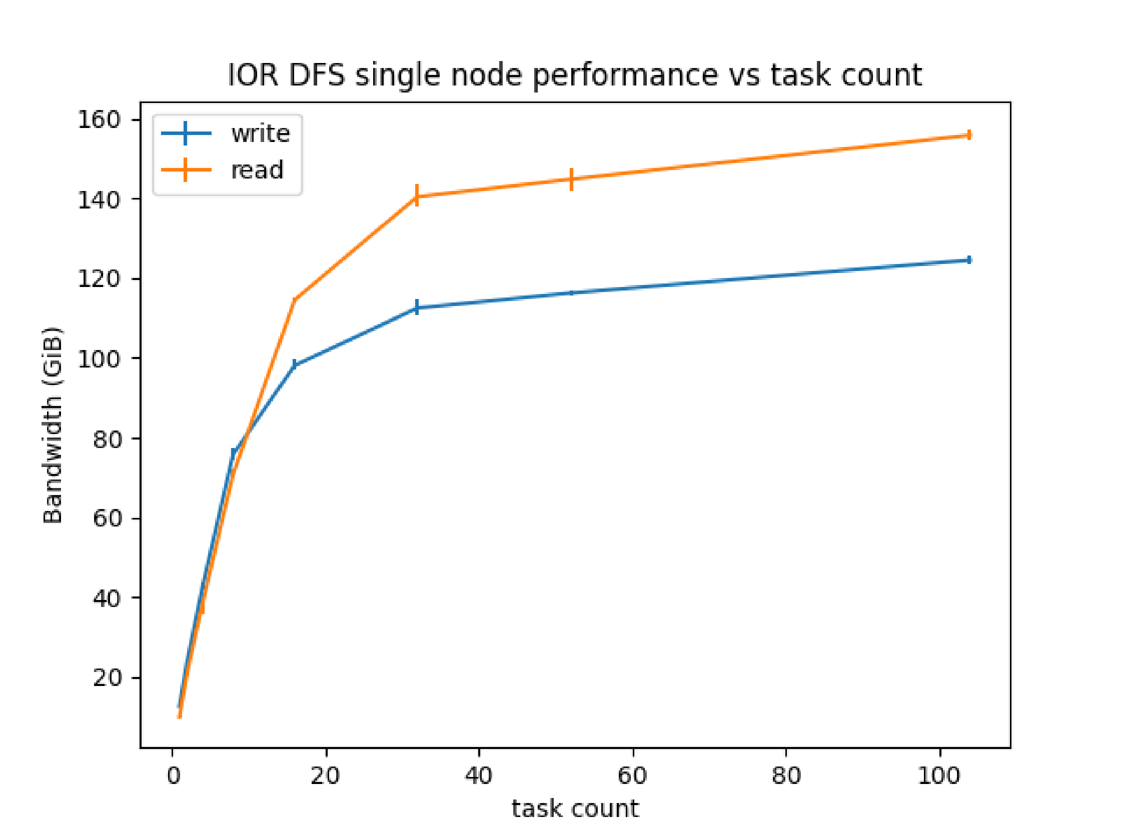
Intel Coyote Pass System
(2) Xeon 5320 CPU (Ice Lake)
(16) 32GB DDR4 DIMMs
(16) 512GB Intel Optane Persistent Memory 200
(16) 15.3TB Samsung PM1733
(2) HPE Slingshot NICs

(25 ~ 20) GB/s X 2 NICs = 40GB/s



4

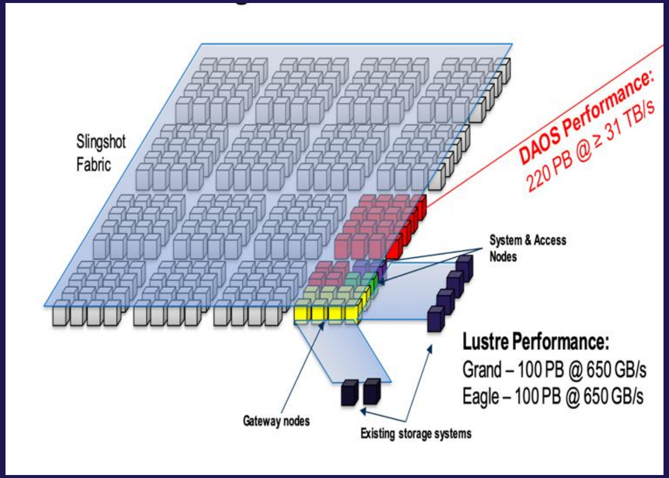
Argonne NATIONAL LABORATORY

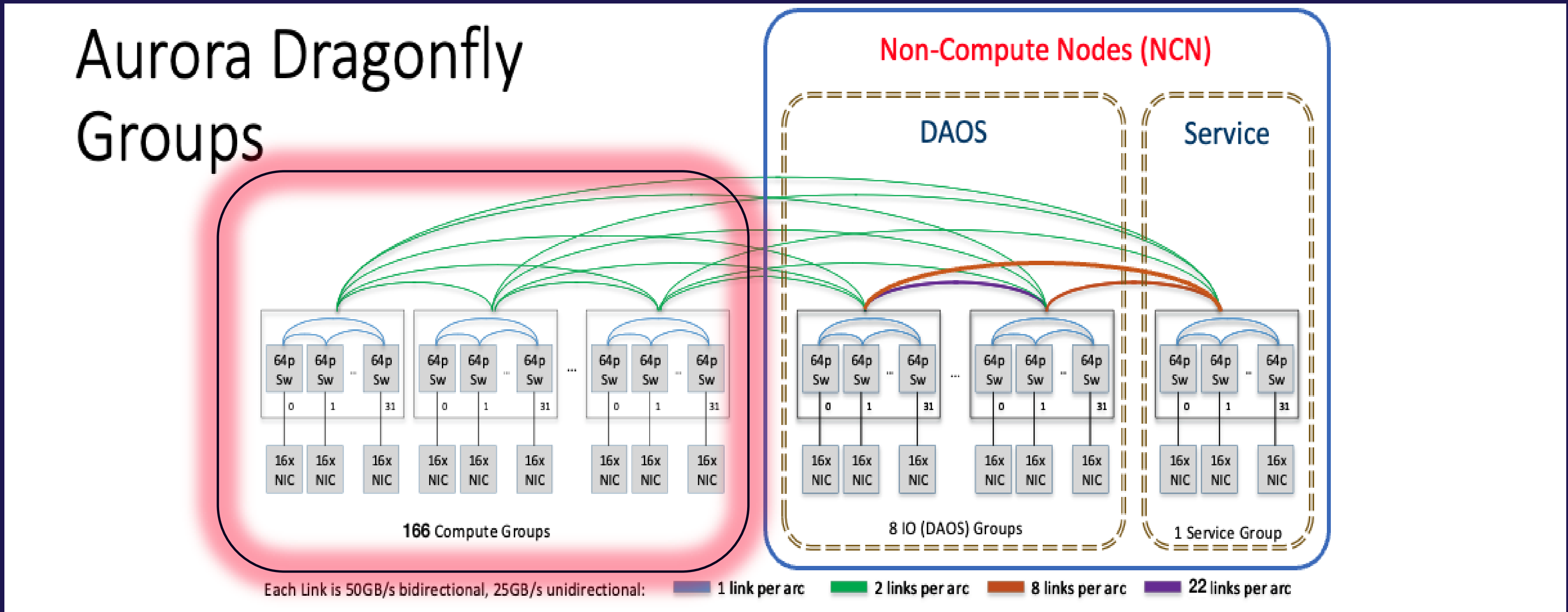


Credits : Rob Lathem – Jan 19

A lot of changes happening within Aurora DAOS and bug fixing under process.
We will be using posix (not DFS) in this tutorial.

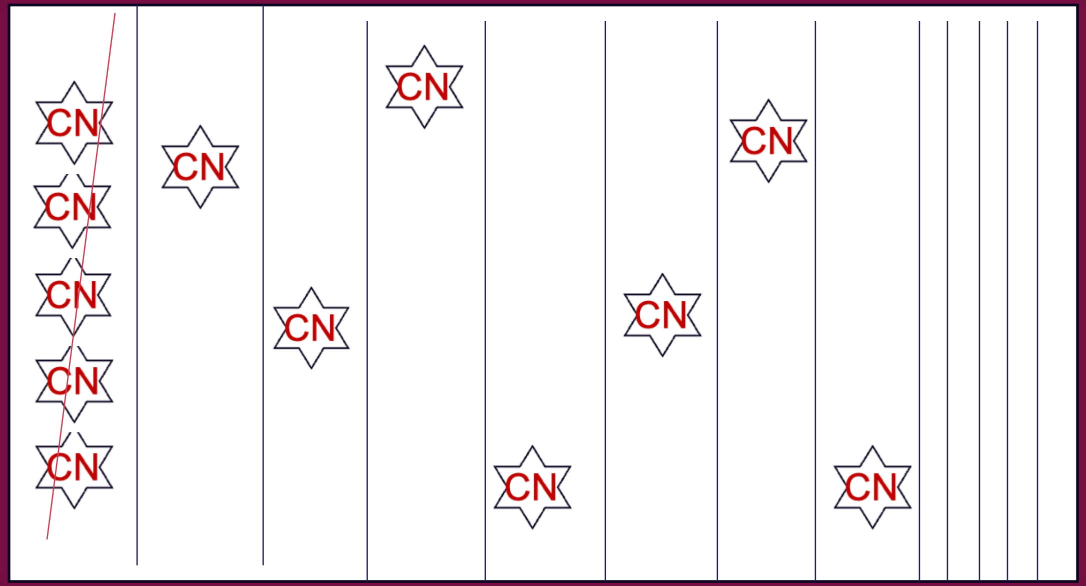
Network Architecture – slingshot fabric - Dragonfly

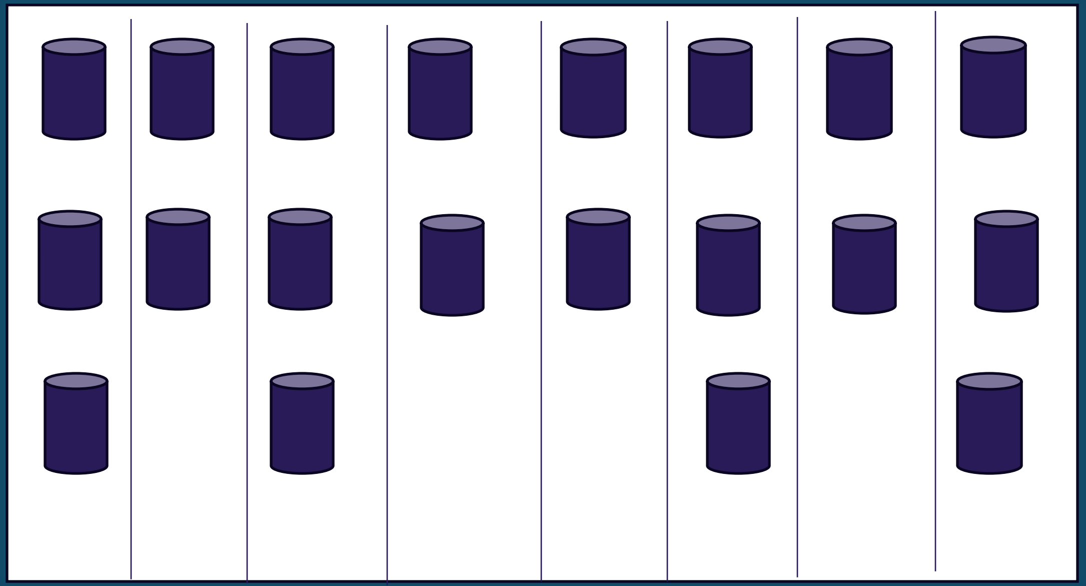




- 1-D Dragonfly Topology - 175 total groups (166 compute + 8 IO + 1 Service)
  - All the global links are optical, all the local links are electrical
  - 2 global links between any two compute groups
  - 22 links between any two IO groups, 8 links between the Service group and each IO group
- All NCN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

Example Scenario on placement of nodes for effective B/W utilization





166 compute dragonfly groups

64 CN per rack *166 groups – 10,624 CNs

8 daos dragon fly groups


20/1024 daos nodes on 64 racks

More info at https://wiki.jlse.anl.gov/display/inteldga/Allocating+Nodes+in+Specific+Racks+or+Cabinets Credits: Nathan nichols

5. Exercise 0 : Daos sanity tests

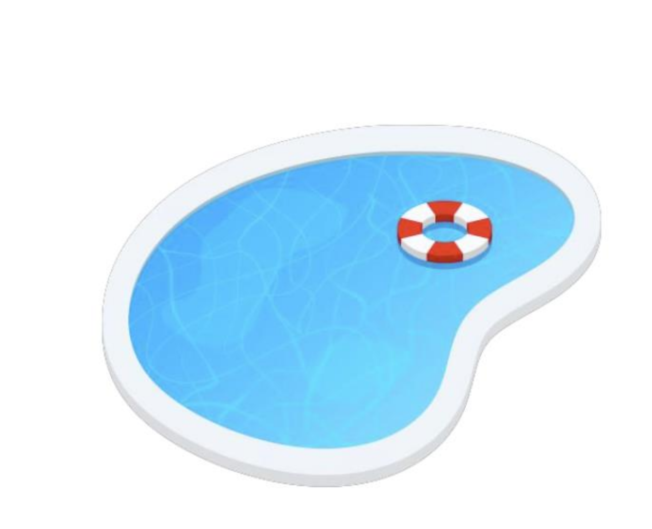
```
module use  /soft/modulefiles
module load daos/base
module load mpich/51.2/icc-all-pmix-gpu
module list

daos version
env | grep DRPC

daos pool query datascience
daos cont list  datascience
```

What is a DAOS 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
```

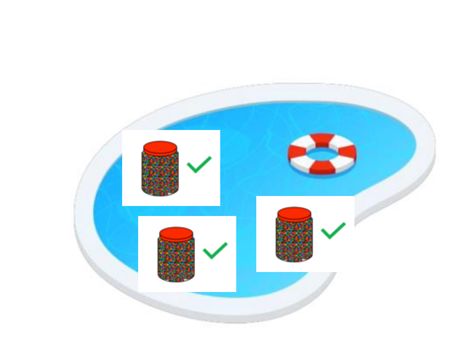
```
kaushikvelusamy@aurora-uan-0010:/gecko/Aurora_deployment> daos cont list datascience
UUID                                 Label
----                                 -----
d85f4fac-d486-4e76-a46d-a60135b2dc64 kaushik_resnet_dataset_cont_sx
2fe037ee-715d-4186-aa19-250fa1fc2988 posix-s1
c9ffb011-28d1-4356-9afd-feaf8269d2bb posix-s4
```

Physically allocated dedicated storage for your project.



What is a DAOS container?
Exercise 1: Creating and querying a container

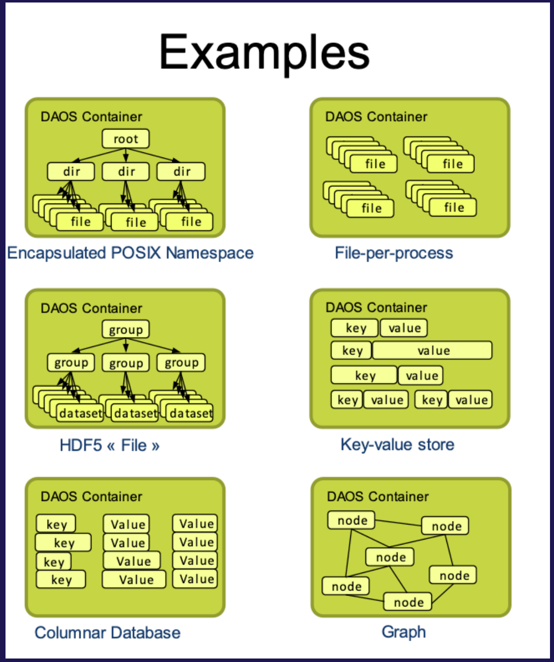
- A pool contains *thousands* of containers
- Basic unit of storage from user perspective
- Containers can be of type [POSIX, HDF, PYTHON, SPARK]*
- Currently: POSIX, MPIIO, DAOS VOL all uses containers of type POSIX and launched through DFUSE

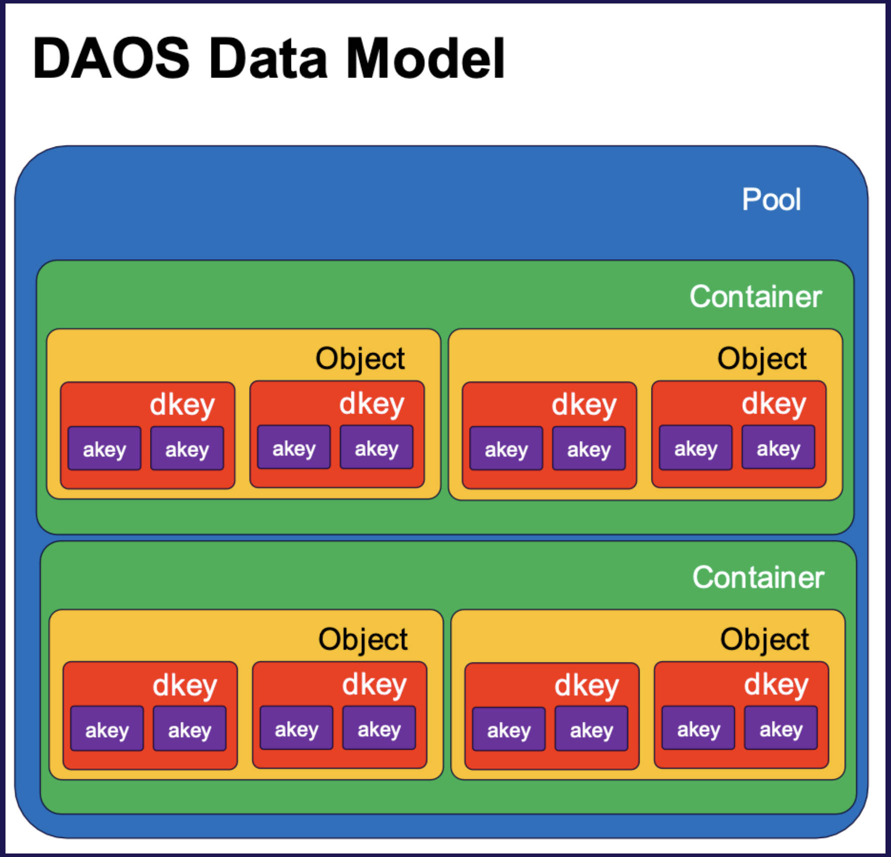


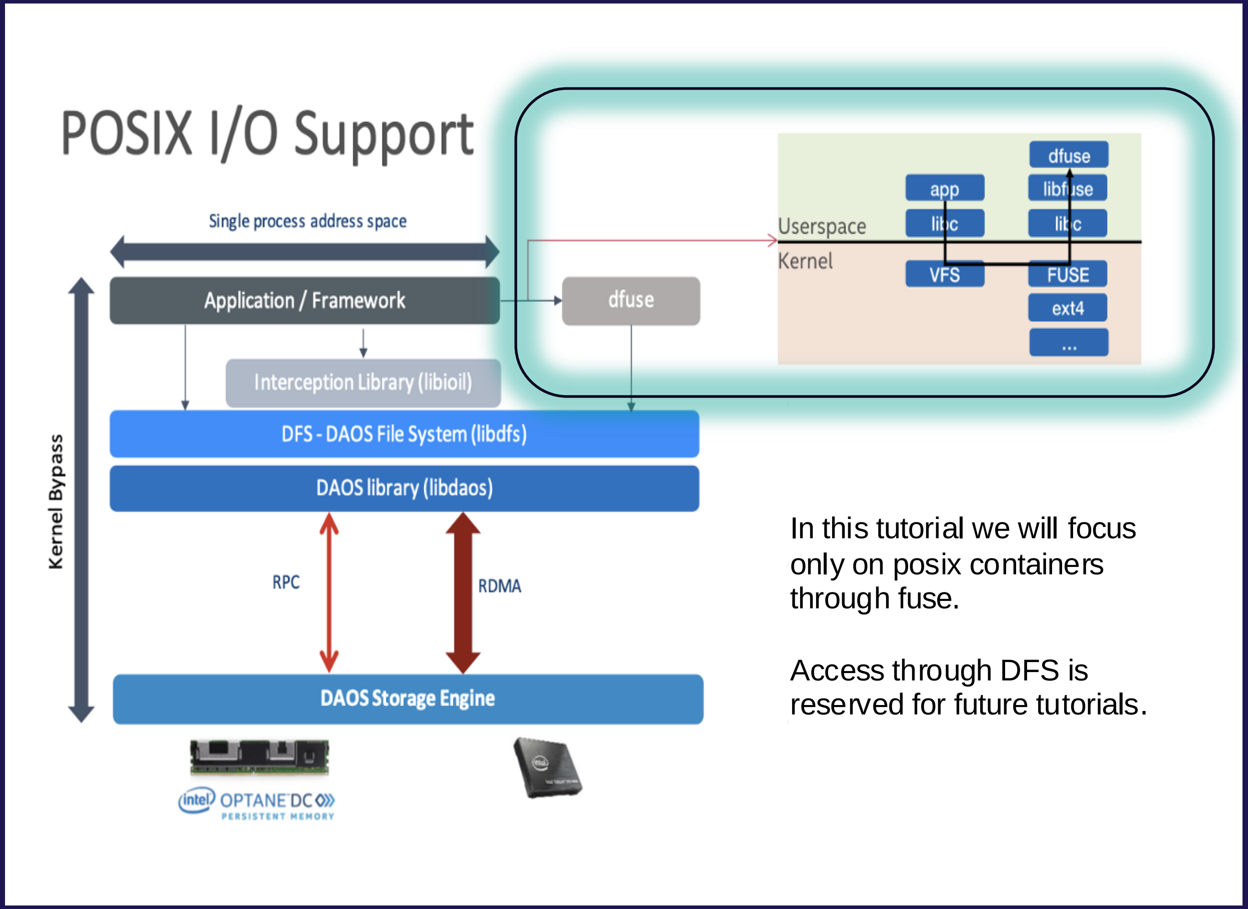
```
DAOS_POOL_NAME = datascience
DAOS_CONT_NAME = username_contype_projname_label
```

```
daos container create --type POSIX --dir-oclass=S1 --file-oclass=SX  $DAOS_POOL_NAME  $DAOS_CONT_NAME
```

```
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
```

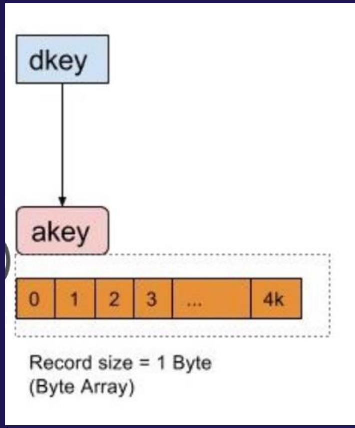


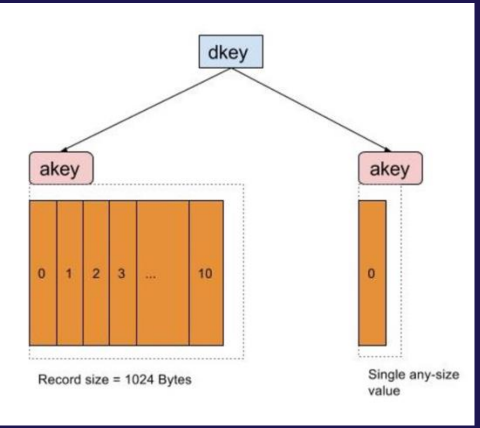




In this tutorial we will focus only on posix containers through fuse.

Access through DFS is reserved for future tutorials.





Additional, improved performance can be had by using a special preloaded library. Adding **LD_PRELOAD=$DAOS_PRELOAD** into the mpiexec command will enable kernel bypass of most POSIX I/O calls, but still use metadata via the FUSE mount point.

kaushikvelusamy@aurora-uan-0010:/gecko/Aurora_deployment> echo $DAOS_PRELOAD
/usr/lib64/libioil.so

Exercise 2: Interacting with DAOS (Login Vs Compute nodes)

From login node

```
mkdir –p /tmp/${DAOS_POOL}/${DAOS_CONT}
start-dfuse.sh -m /tmp/${DAOS_POOL}/${DAOS_CONT} --pool ${DAOS_POOL} --cont ${DAOS_CONT}
clean-dfuse.sh  ${DAOS_POOL_NAME}:${DAOS_CONT_NAME}
fusermount3 -u /tmp/${DAOS_POOL}/${DAOS_CONT}
```

From compute node

```
# launched using pdsh on all compute nodes mounted at: /tmp/<pool>/<container>
launch-dfuse.sh ${DAOS_POOL_NAME}:${DAOS_CONT_NAME}
clean-dfuse.sh  ${DAOS_POOL_NAME}:${DAOS_CONT_NAME}
```

Validation

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

The DAOS MPI-IO driver is implemented within the I/O library in MPICH (ROMIO)

To use this driver append “daos:/path/to/myfile” or

export ROMIO_FSTYPE_FORCE="daos:"

Python Container

To create a python container in a pool labeled tank:

```
$ daos cont create tank --label neo --type PYTHON
  Container UUID : 3ee904b3-8868-46ed-96c7-ef608093732c
  Container Label: neo
  Container Type : PYTHON

Successfully created container 3ee904b3-8868-46ed-96c7-ef608093732c
```

One can then connect to the container by passing the pool and container labels to the DCont constructor:

```
>>> import pydaos
>>> dcont = pydaos.DCont("tank", "neo")
>>> print(dcont)
tank/neo
```

Note

PyDAOS has its own container layout and will thus refuse to access a container that is not of type "PYTHON"

POSIX Container

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

Lustre job script

```
#!/bin/bash -x
# qsub -l select=1 -l walltime=00:20:00 -A Aurora_deployment -k doe -l filesystems=gecko -q workq ./ex1.sh  or - I


module load mpich/51.2/icc-all-pmix-gpu
module list

cd ${PBS_O_WORKDIR}

# Run your app
mpiexec -np 16 -ppn 16 -genvall --no-vni ior  -a mpio  -i 5 -t 16M -b 640M  -w  -r -o ./io_1.dat
```

DAOS job script

```
#!/bin/bash -x

# qsub -l select=1 -l walltime=00:20:00 -A Aurora_deployment -k doe -l filesystems=gecko -q alcf_daos_cn -l daos=default ./ex1.sh
or - I

module use /soft/modulefiles                                  # To load DAOS module
module load daos/base                                         # To load DAOS module


export DAOS_POOL_NAME=datascience
export DAOS_CONT_NAME=kaus_posix_llm-agpt_sx
launch-dfuse.sh ${DAOS_POOL_NAME}:${DAOS_CONT_NAME}    # To mount your DAOS FS at /tmp/

cd ${PBS_O_WORKDIR}
mpiexec -np 16 -ppn 16 -genvall --no-vni ior  -a mpio  -i 5 -t 16M -b 640M  -w  -r -o daos:/io_1.dat
```

Exercise 3: Your first DAOS job script

```
#!/bin/bash -x

# qsub -l select=1 -l walltime=00:20:00 -A Aurora_deployment -k doe -l filesystems=gecko -q alcf_daos_cn -l daos=default ./ex1.sh  or - I

module use /soft/modulefiles                            # To load DAOS module
module load daos/base                                   # To load DAOS module
module load mpich/51.2/icc-all-pmix-gpu                 # To load mpi for DAOS MPIO ADIO
module list                                             # To confirm is DAOS module is loaded
env|grep DRPC                                           # To confirm if DAOS service is running
ps -ef|grep daos|grep -v grep

export DAOS_POOL_NAME=datascience
export DAOS_CONT_NAME=kaus_posix_llm-agpt_sx


clean-dfuse.sh ${DAOS_POOL_NAME}:${DAOS_CONT_NAME}  # Good practice to clean before mount
launch-dfuse.sh ${DAOS_POOL_NAME}:${DAOS_CONT_NAME} # To mount your DAOS FS at /tmp/
mount | grep dfuse                                      # To confirm if mount is successful

export ROMIO_FSTYPE_FORCE="daos:"                       # To avoid giving daos:/filepath everytime when using mpio

# Run your app
mpiexec -np 16 -ppn 16 -genvall --no-vni ior  -a mpio -i 5 -t 16M -b 640M  -w  -r -o /io_1.dat


clean-dfuse.sh ${DAOS_POOL _NAME}:${DAOS_CONT _NAME}        # To unmount
```

With Darshan

```
kaushikvelusamy@aurora-uan-0011:> /gecko/Aurora_deployment/kaushik/soft/install-darshan/bin/darshan-config  --log-path
/gecko/Aurora_deployment/kaushik/soft/install-darshan/main-log-dir/
```

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib64/:/gecko/Aurora_deployment/kaushik/soft/install-darshan/lib/
```

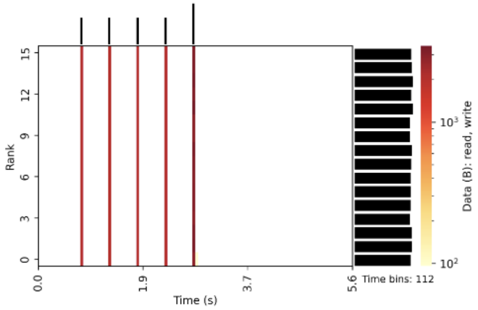
```
LD_PRELOAD=$DAOS_PRELOAD:/gecko/CSC250STDM10_CNDA/kaushik/gitrepos/install-darshan/lib/libdarshan.so YOUR_APP APP_CONFIG daos:/filename
```

```
ls /gecko/Aurora_deployment/kaushik/soft/install-darshan/main-log-dir/2024/02/22/app_with_daos.darshan
```

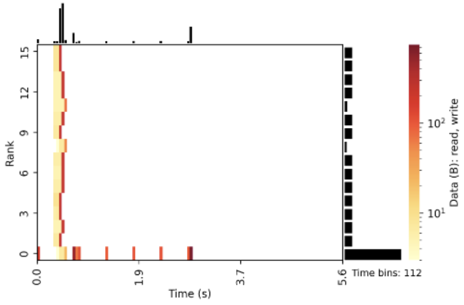
DAOS – Darshan – Analyser – laptop – (only MPIO-ADIO)

```
(pyenv) Kaushik:spack kvelusamy$ spack install darshan darshan-util
(pyenv) Kaushik:spack kvelusamy$ python3 -m darshan summary  app_with_daos.darshan
```

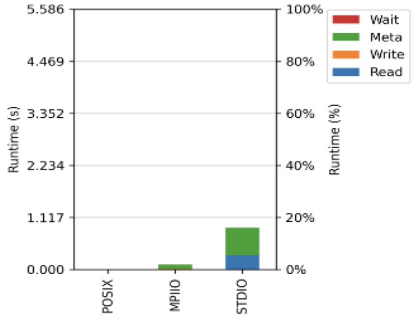
Heat Map: HEATMAP MPIIO



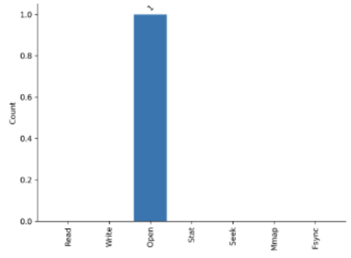
Heat Map: HEATMAP STDIO



I/O Cost



Operation Counts

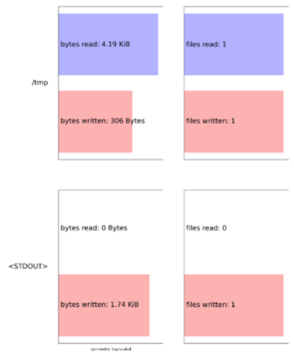


Histogram of I/O operation frequency.

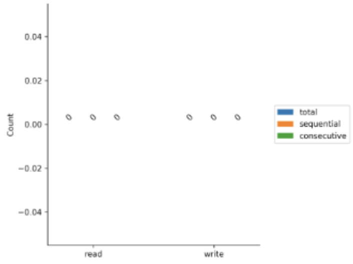
Common Access Sizes

Access Size	Count
256	640
272	39
544	6
328	3

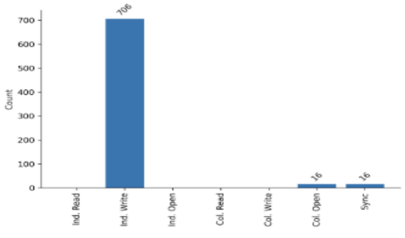
Data Access by Category



Access Pattern

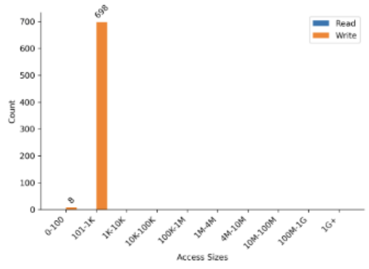


Operation Counts

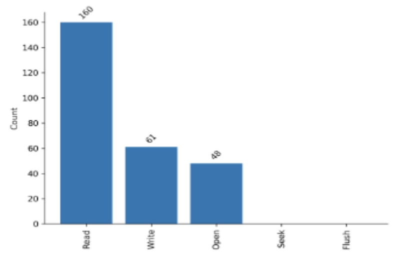


Histogram of I/O operation frequency.

Access Sizes



Operation Counts



Histogram of I/O operation frequency.

Exercise 4: Lustre Vs Daos Single node NIC

```
mpiexec -np 8-ppn 8 -d 2 --cpu-bind list:0,1,2,3,52,53,54,55 -genvall
        /gecko/Aurora_deployment/kaushik/soft/install-darshan/bin/ior
        -a mpio
    -b 50G -t 1M -w -r -i 1 -v
        -o /path_to_an_output_file_on_lustre.dat
```

Exercise 4: Lustre Vs Daos Single node NIC

launch-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}

mpiexec -np 8 -ppn 8 -d 2 --cpu-bind list:0,1,2,3,52,53,54,55 -genvall
/gecko/Aurora_deployment/kaushik/soft/install-darshan/bin/ior
-a mpio
-b 50G -t 1M -w -r -i 1 -v
-o **daos:/**output_file.dat

clean-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}

Exercise 4: Lustre Vs Daos Single node NIC

launch-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}

mpiexec -np 8-ppn 8 -d 2 --cpu-bind list:0,1,2,3,52,53,54,55 -genvall
/gecko/Aurora_deployment/kaushik/soft/install-darshan/bin/ior
-a DFS
--dfs.pool=$DAOS_POOL
--dfs.cont=$DAOS_CONT
--dfs.dir_oclass=S1
--dfs.oclass=SX
--dfs.chunk_size=$((128*1024))
-b 50G -t 1M -w -r -i 1 -v
-o **/io.dat**

clean-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}

Exercise 5: Lustre Vs Daos Single node Request Size

```
launch-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}

#for i in 4 32 64 128 256 512 1024 2048 4096 8192 16384 32768;
#do

mpiexec -np 8 -ppn 8 -d 2 --cpu-bind list:0,1,2,3,52,53,54,55 -genvall
        /gecko/Aurora_deployment/kaushik/soft/install-darshan/bin/ior
                -a mpio
                -b ${bsz[$i]}G -t ${tfz[$i]}M -w -r -i 1 -v
                -o daos:/output_file.dat

#done

clean-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}
```

Exercise 6: Lustre Vs Daos multi node Scaling

```
launch-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}


#for nnodes in 1 2 4 8 16 32 64 128 256;
#do


mpiexec -np $((8*nnodes)) -ppn 8 -d 2 --cpu-bind list:0,1,2,3,52,53,54,55 -genvall
          /gecko/Aurora_deployment/kaushik/soft/install-darshan/bin/ior
                   -a mpio
                   -b 1G -t $1M -w -r -i 1 -v
                   -o daos:/output_file.dat


#done


clean-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}
```

Exercise 8: Lustre Vs Daos Single node VPIC-IO h5bench

launch-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}

export HDF5_HOME=/gecko/Aurora_deployment/kaushik/soft/install-hdf5/library/install/**hdf5**

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/gecko/Aurora_deployment/kaushik/soft/install-hdf5/library/install/**hdf5/lib**:/usr/lib64/

export PATH=$PATH\:/gecko/Aurora_deployment/kaushik/soft/install-hdf5/library/install/**hdf5/bin**

export PATH=$PATH\:/gecko/Aurora_deployment/kaushik/soft/install-**h5bench**-daos-prefix/bin

LD_PRELOAD=$DAOS_PRELOAD

h5bench -d path_to_h5bench_config.json

clean-dfuse.sh ${DAOS_POOL}:${DAOS_CONT}

```json
{
   "mpi"              : {
      "command"      : "mpiexec",
      "ranks"        : "16",
      "configuration" : "-np <NTOTRANKS> -ppn 16 -d 1  --cpu-bind \"list:0,1,2,3,4,5,6,7,52,53,54,55,56,57,58,59\"  --no-vni -genvall"
                    },
   "vol"             : { },
   "file-system"      : { },
   "directory"        : "/tmp/CSC250STDM10/<CONTAINER_NAME>/storage",
   "benchmarks": [
             {
               "benchmark"     : "write",
               "file"         : "test.h5",
               "configuration": {
                        "MEM_PATTERN"                    : "CONTIG",
                        "FILE_PATTERN"                   : "CONTIG",
                        "TIMESTEPS"                      : "5",
                        "DELAYED_CLOSE_TIMESTEPS"          : "2",
                        "COLLECTIVE_DATA"                : "YES",
                        "COLLECTIVE_METADATA"             : "YES",
                        "EMULATED_COMPUTE_TIME_PER_TIMESTEP": "1 s",
                        "NUM_DIMS"                       : "1",
                        "DIM_1"                          : "4194304",
                        "DIM_2"                          : "1",
                        "DIM_3"                          : "1",
                        "CSV_FILE"                       : "write_large_coll.csv",
                        "MODE"                           : "SYNC"
                        }
             },
{
               "benchmark"     : "read",
               "file"         : "test.h5",
               "configuration": {
                        "MEM_PATTERN"                    : "CONTIG",
                        "FILE_PATTERN"                   : "CONTIG",
                        "READ_OPTION"                    : "FULL",
                        "COLLECTIVE_DATA"                : "YES",
                        "COLLECTIVE_METADATA"             : "YES",
                        "TIMESTEPS"                      : "5",
                        "DELAYED_CLOSE_TIMESTEPS"          : "2",
                        "EMULATED_COMPUTE_TIME_PER_TIMESTEP": "1 s",
                        "NUM_DIMS"                       : "1",
                        "DIM_1"                          : "4194304",
                        "DIM_2"                          : "1",
                        "DIM_3"                          : "1",
                        "CSV_FILE"                       :"read_large_coll.csv",
                        "MODE"                           : "SYNC"
                        }
             }
         ]
}
```

Extras : Debugging a container and fine tuning

- Accessing DAOS VOL
- DFS
- Object classes
- MPIO tuning
- MPIO aggregator
- Erasure closure
- Redundancy factors.
- Transferring from DFS to POSIX containers
- Using the DFS Container with Dfuse
- Directly with c and python
- Set-attr and get-attr
- Metadata tests

Acknowledgements

- Gordon Mcpheeters, Vitali Morozov, Huihuo Zheng, Paul Coffman, Mohamad Charawi, Kevin Harms
- Intel DAOS team, HPE cray team & MCS team

- 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.

MPI-IO Driver for DAOS

For fine tuning,

```
export ROMIO_PRINT_HINTS=1
echo "cb_nodes 128" >> ${PBS_O_WORKDIR}/romio_hints
mpiexec  --env ROMIO_HINTS = romio_hints_file daos:/YOUR_PROGRAM

mpiexec  --env MPICH_MPIIO_HINTS = path_to_your_file*:cb_config_list=#*:2#
                                                        :romio_cb_read=enable
                                                        :romio_cb_write=enable
                                                        :cb_nodes=32
                                        daos:/YOUR PROGRAM
```

Starting a job

- Before rerunning the same scripts again – check for overwriting on the same file- have a clean environment