DAOS Usage and Application

Kevin Harms

Argonne National Laboratory – Leadership Computing

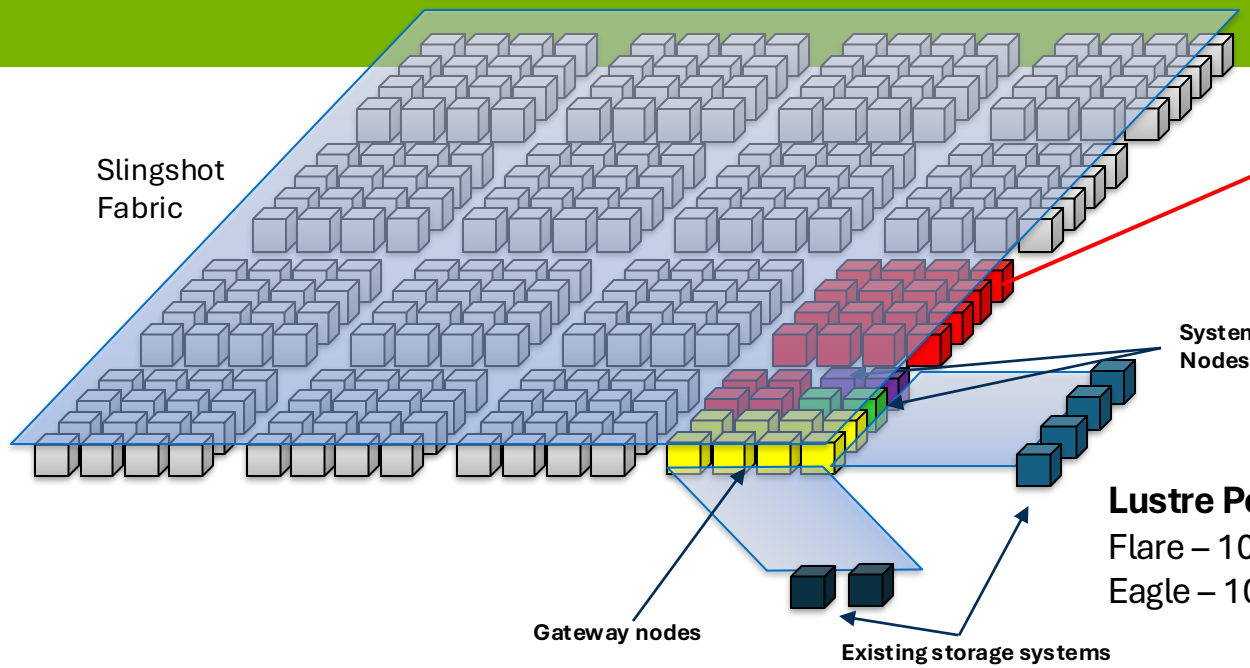
What is DAOS

- Distributed Asynchronous Object Storage (DAOS) is an open-source software-defined high-performance scalable key-value-array store providing support for multiple data models.
- DAOS provides functionality similar to that of Parallel File Systems (PFS) such as Lustre or Spectrum Scale (GPFS) but is not built on files but objects.
- Provides several APIs for storing and retrieving data, most importantly, POSIX.

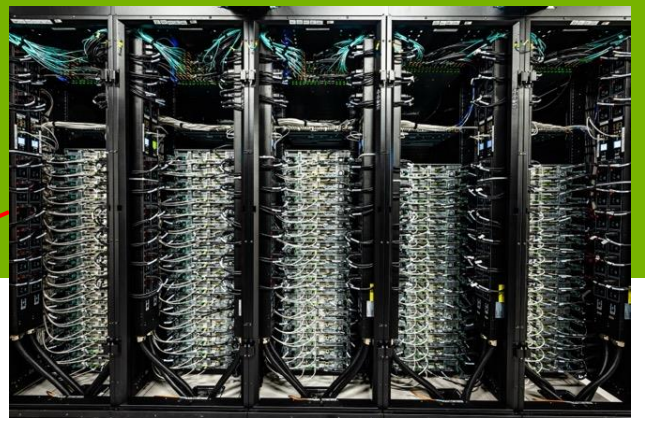
Why DAOS

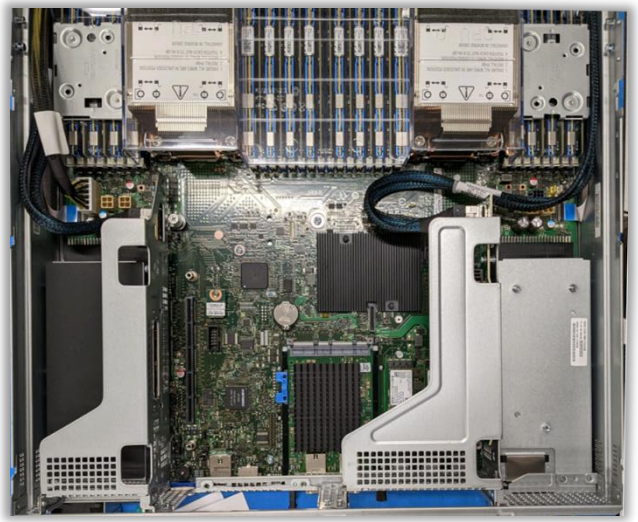
- Extreme scalability
  - No synchronous read-modify-write
  - No locking
  - No client tracking or client recovery
  - Multi-version concurrency control
  - Not all POSIX semantics are preserved
    - https://docs.daos.io/latest/user/filesystem/?h=posix#posix-compliance
- Maximum performance
  - Specifically designed to take advantage of NVMe and Storage Class Memory (SCM)

Aurora Storage Architecture



Lustre Performance:
Flare – 100 PB @ 650 GB/s
Eagle – 100 PB @ 650 GB/s





The Aurora open-source storage strategy strongly favors cooperation:

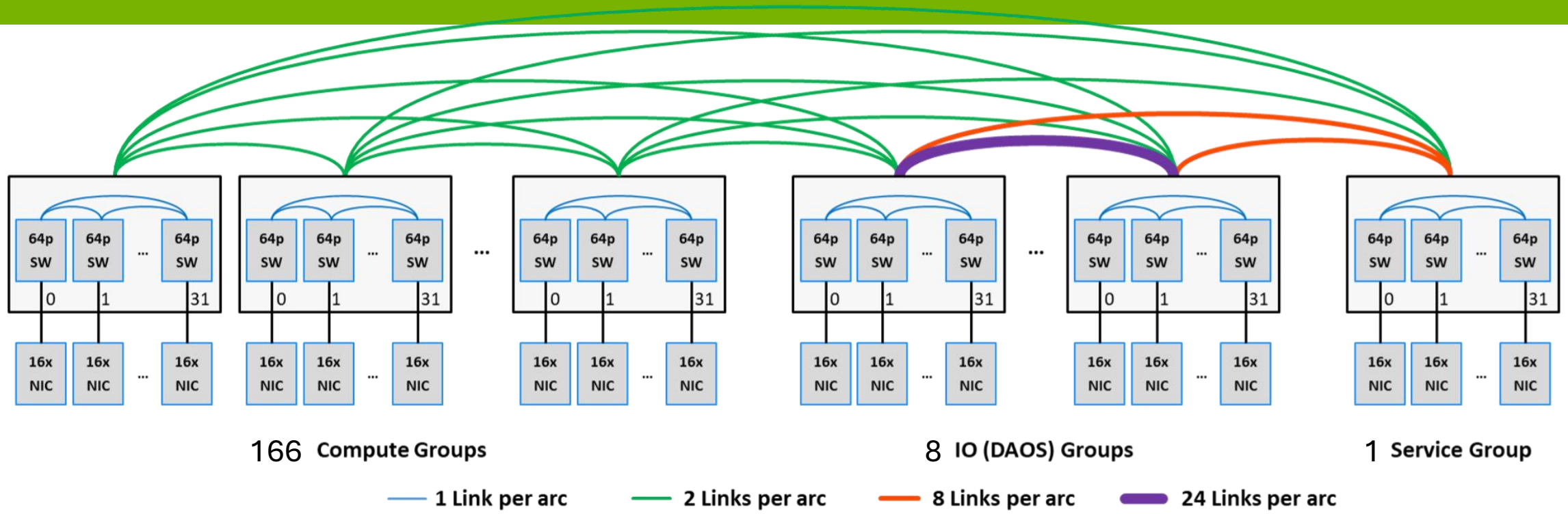
- DAOS: object storage system for in-fabric high-performance platform storage (the first of its kind on a DOE leadership system!)
- Lustre: parallel file systems for facility-wide access and data sharing

Namespace integration will make it easier for users to manage data.

•1024 DAOS server nodes, each with:

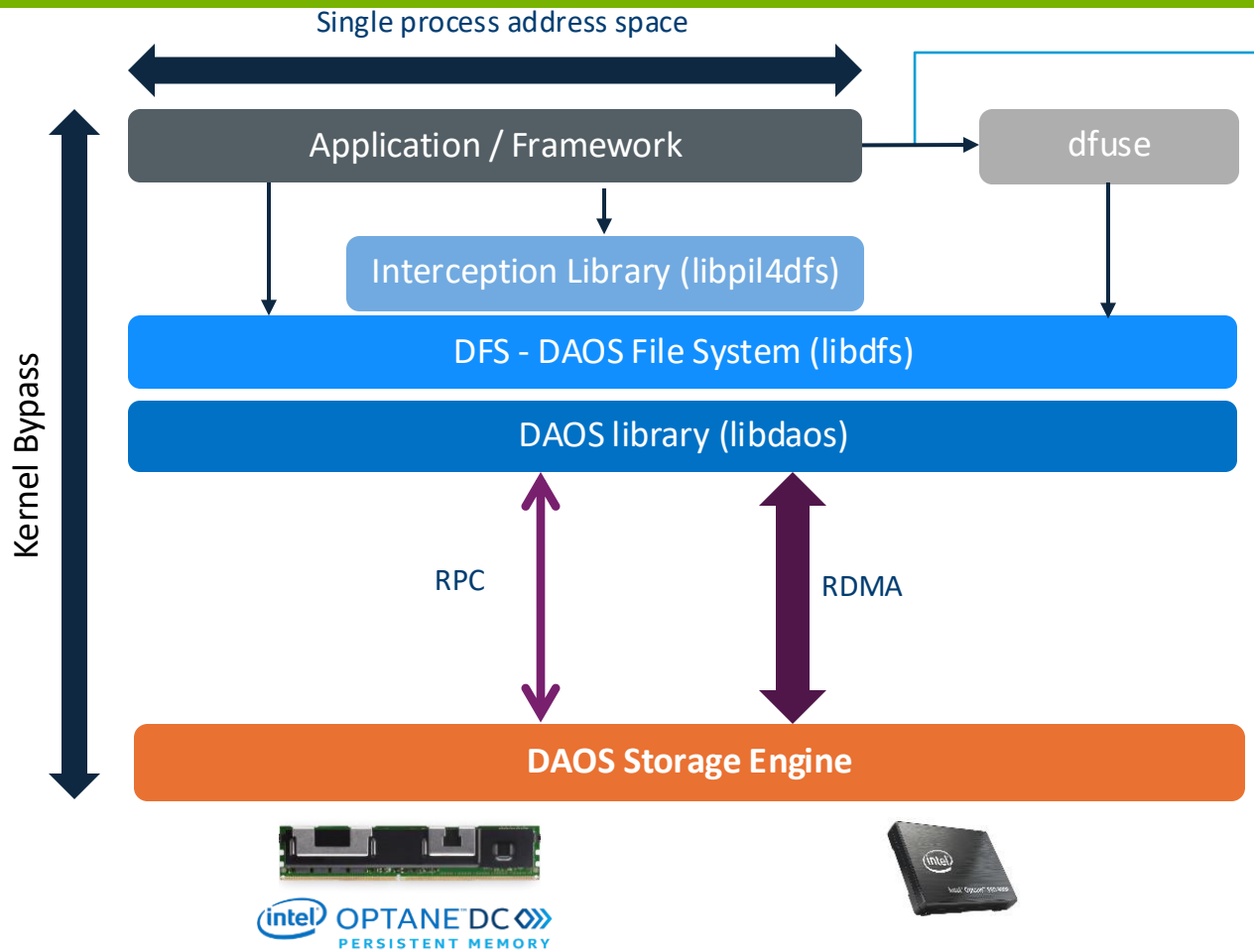
- 16 x 512GB persistent memory
- 16 x 15.3TB NVMe drives
- 2 x HPE Slingshot NICs
- Dual CPU with 512 GB RAM

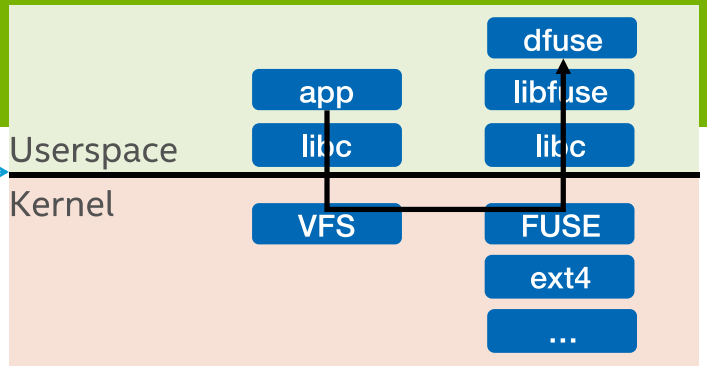
Aurora Network Architecture



- Increased DAOS inter-group bandwidth
  - Support rebuilding and inter-server communication
  - Prevent DAOS server traffic interfering with application communication
- Increased bandwidth to service group
  - Support off-cluster access and data-movement to other storage systems

I/O Architecture

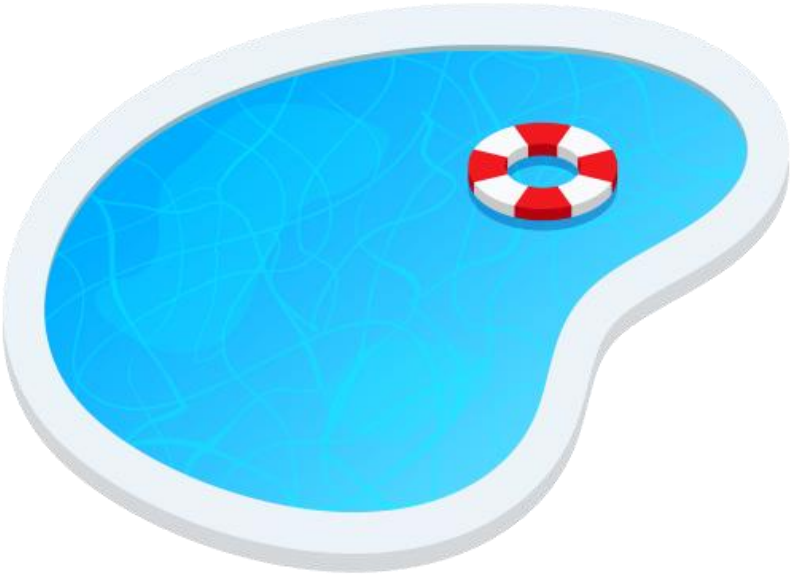




- User space DFS library with an API like POSIX.
  - Requires application changes (new API)
  - Kernel Bypass, no client cache
- DFUSE plugin to support POSIX API
  - No application changes
  - Fuse Kernel Supports data (wb and ra) & metadata caching (stat, open, etc.)
- DFUSE + IL
  - No application changes, runtime LD_PRELOAD
  - Kernel Bypass for IO and metadata

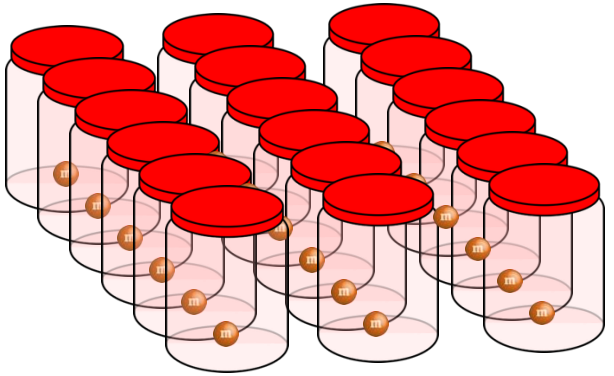
DAOS Pools

- Pools
  - A system may contain *hundreds*
  - Physically allocated storage
    - Decided at pool creation time
  - Equal storage allocated per storage target
  - Contains Access Control Lists (ACLs)
  - Contains default parameters for containers

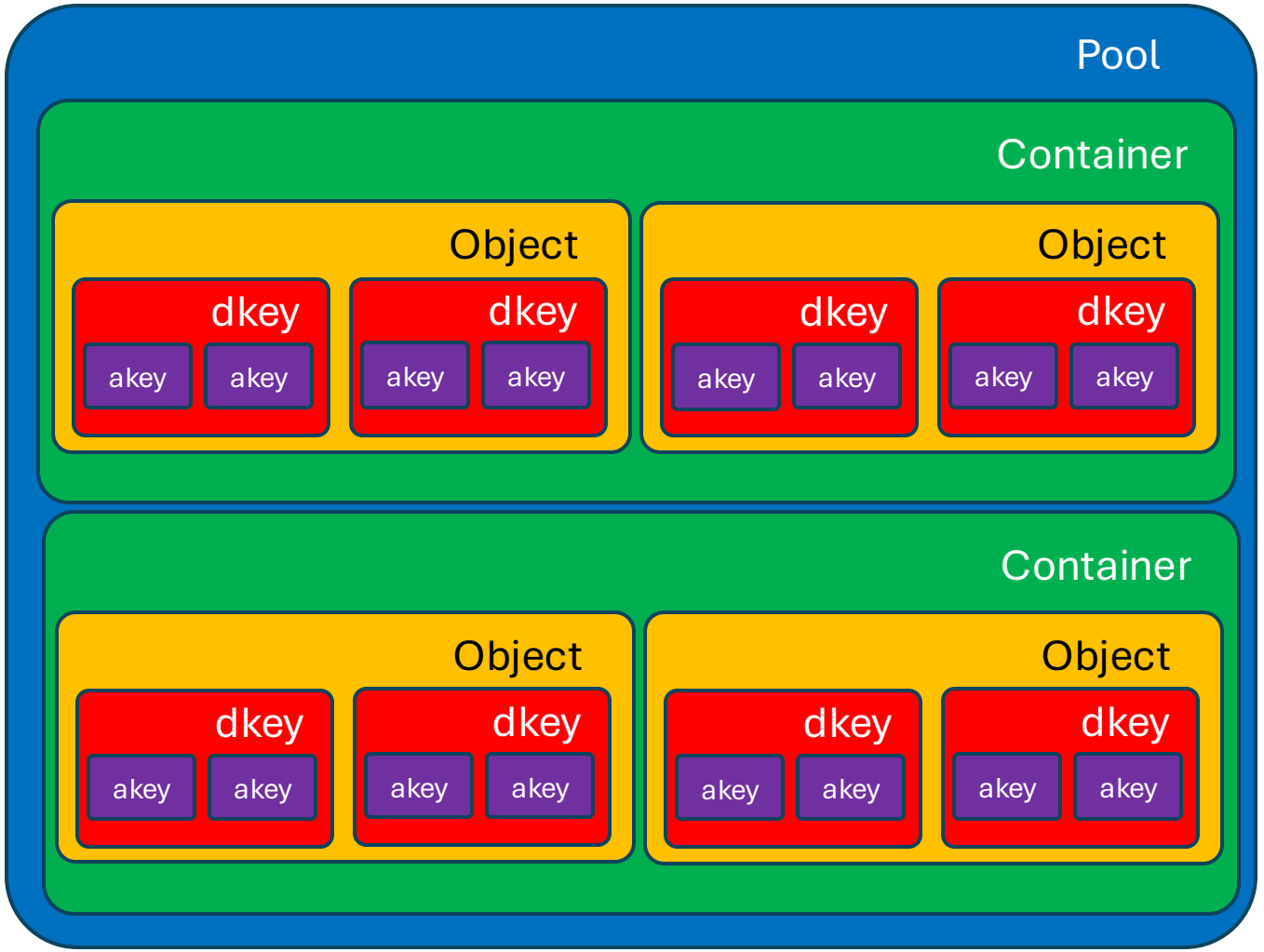


DAOS Containers

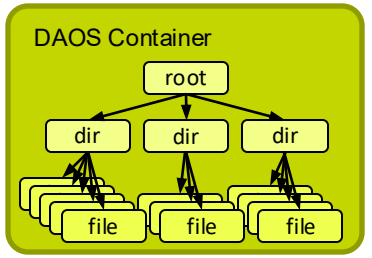
- Containers
  - A pool may contain *thousands* of containers
  - Basic unit of storage from user perspective
  - Containers have a type (POSIX, HDF5, pyDAOS, SEGY, …)
  - POSIX containers can have many *millions* of files/directory/data
  - Configuration for object class/redundancy, checksums, cell size, etc.
    - Many options
    - Determines distribution across pool
  - ACLs
    - Determine access rights, not POSIX permissions



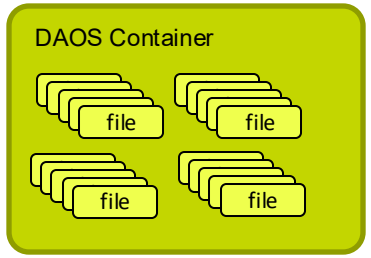
DAOS Data Model



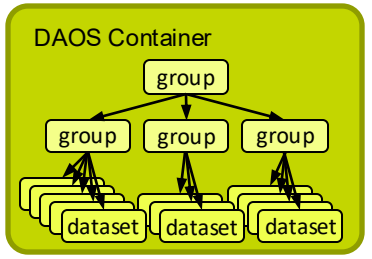
Examples



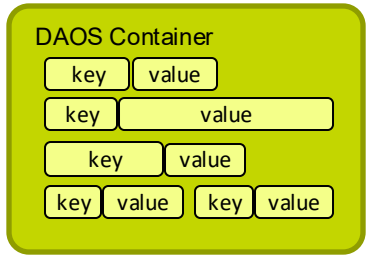
Encapsulated POSIX Namespace



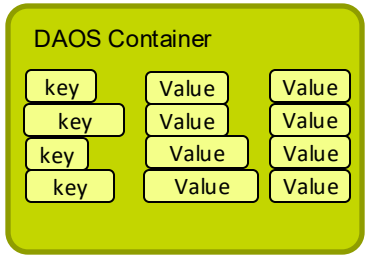
File-per-process



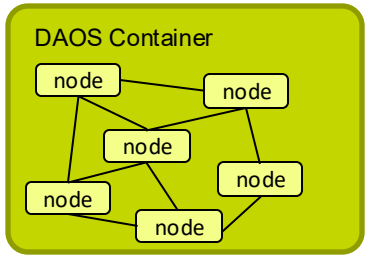
HDF5 « File »



Key-value store



Columnar Database



Graph

Aurora Pools and Containers

- ALCF will assign pools to projects
  - Large allocations will receive pools with ~80% of available targets
    - Number of targets proportional to performance
  - Pools are a physical allocation (guaranteed allocation of storage)
  - Users of the project will be given full rights to the pool
  - Users create their own containers with their desired settings
    - The initial pool will have the suggested defaults from ALCF
- POSIX containers must be mounted by the user
  - dfuse started by the user
- Lustre/DAOS integration should allow easy POSIX container access
  - Access DAOS POSIX containers via existing Lustre mount points

Containers

- User created
- Ability to select
  - Data protection
    - Checksums
    - Redundancy factor
  - EC Cell size
  - Features
    - compression
    - encryption
  - Security
    - ACLs

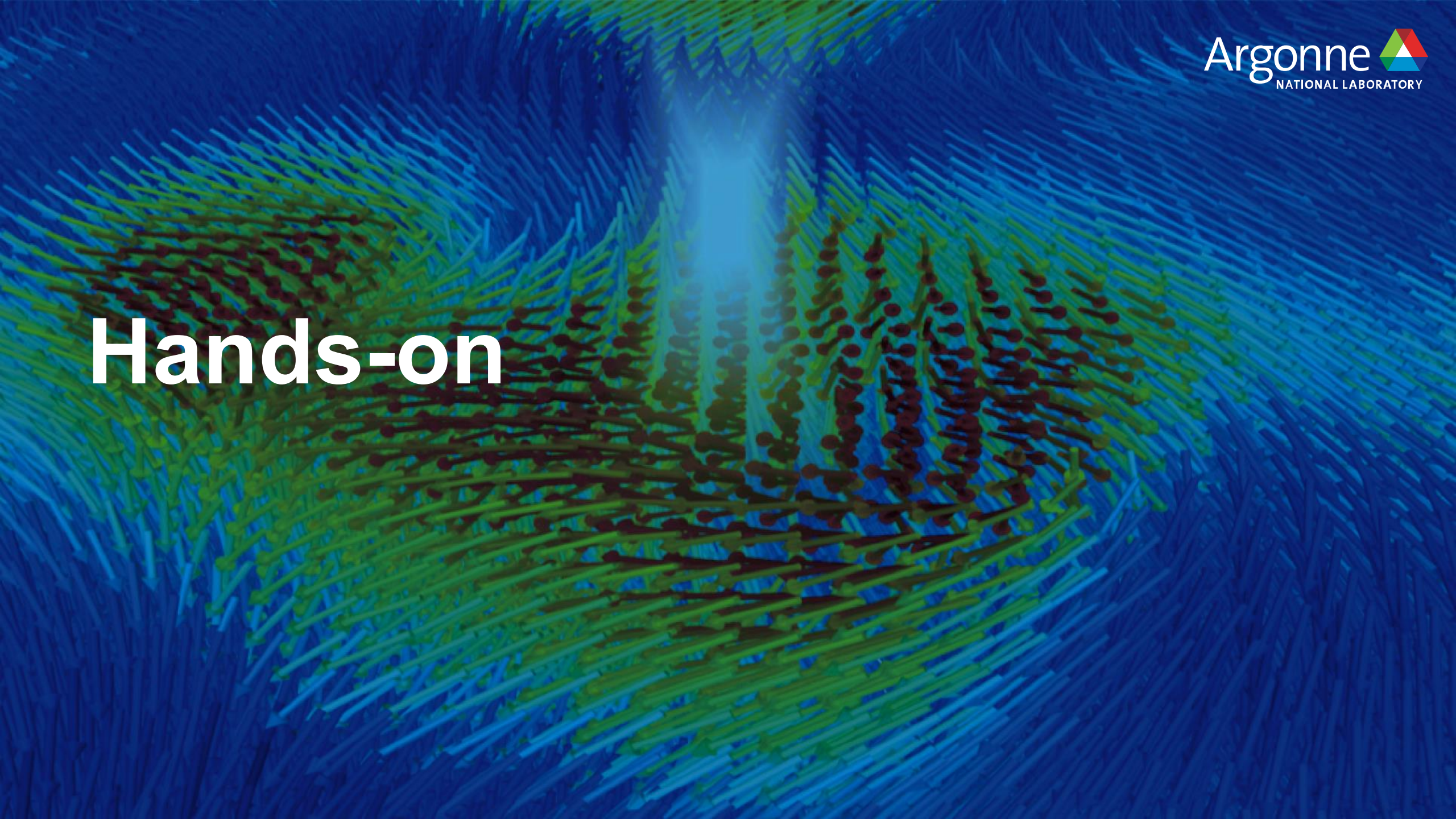
APIs for POSIX Containers

- POSIX
  - Provides POSIX API but not full POSIX semantics
  - Most POSIX semantics supported
- MPI-IO
  - Native DAOS backend
- DFS (DAOS File System)
  - POSIX-like API
  - Provides more control over file/object configuration

Further Reading

- https://daos.io
- https://docs.daos.io/v2.6/
  - https://docs.daos.io/v2.6/user/workflow/
- https://docs.alcf.anl.gov/aurora/data-management/daos/daos-overview/

- https://github.com/daos-stack/daos
  - README.md contain information about structure and design



Hands-on

DAOS @ ALCF

- daos_user
  - ~128 servers
  - Primary storage system where projects that request storage are allocated
  - Peak performance around ~4 TB/s with EC16+2
- daos_perf
  - ~256 servers
  - Experimental storage system – used for testing with certain users
    - Some risk to using
  - Pools only allocated here if you're working with ALCF

Setup

- ATPESC2025
  - DAOS pool created for all attendees
  - Within “daos_user” instance
  - All attendees have access, only put data within that you are willing to share with everyone
  - Someone could also delete your data
- Container
  - Create ***your own*** container, name with your user name so it is clear
    - Using “daos-test” just invites issues with confusion for other attendees

Container Creation

- From the login node…
- Load modules
  - module use /soft/modulefiles
  - module load daos/base
- Check the pool is accessible
  - daos pool query ATPESC2025
- Create a container
  - daos cont create --type=POSIX --dir-oclass=RP_3G1 --file-oclass=EC_16P2GX --chunk-size=2097152 --properties=rd_fac:2,cksum:crc32,cksum_size:131072,srv_cksum:on ATPESC2025 ${USER}_cont
  - daos cont query ATPESC2025 ${USER}_cont

```
harms@aurora-uan-0009:~> daos cont query ATPESC2025 ${USER}_cont
  Container UUID             : 03081c8f-a34d-4888-9f05-e0c6bc526f20
  Container Label            : harms_cont
  Container Type             : POSIX
  Pool UUID                  : 23afd63a-9571-43d1-9cf4-03b55eaa7141
  Container redundancy factor : 2
  Number of open handles     : 1
  Latest open time           : 0x20353e8d42cc0000 (2025-08-06 20:11:12.463208448 +0000 UTC)
  Latest close/modify time   : 0x20353e8d59800000 (2025-08-06 20:11:12.4870144 +0000 UTC)
  Number of snapshots        : 0
  Object Class               : UNKNOWN
  Dir Object Class           : RP_3G1
  File Object Class          : EC_16P2GX
  Chunk Size                 : 2.0 MiB
```

Mount DAOS (Login)

- Mount DAOS container
  - mkdir $HOME/${USER}_cont
  - dfuse --disable-caching $HOME/${USER}_cont ATPESC2025 ${USER}_cont
  - mount | grep $USER
  - ls $HOME/${USER}_cont
  - touch $HOME/${USER}_cont/this_is_daos
- Unmount DAOS container (optional)
  - fusermount3 –u $HOME/${USER}_cont
  - ls $HOME/${USER}_cont

NOTE:
You only did this on one UAN

Setup Container

- Clone the example job scripts and code to your $HOME
  - git clone https://github.com/radix-io/hands-on.git
- Copy examples into your container
  - cp -R $HOME/hands-on/daos $HOME/${USER}_cont/.
  - ls $HOME/${USER}_cont
  - mount | grep $USER
- Modify the job script(s) if you wish to point to your own application

Job Script

- qsub must include the --filesystems=daos_user_fs
  - This tells the prologue to start the appropriate daos agent process
- Load the DAOS module
- launch-dfuse.sh
  - This will launch ”dfuse” on all the jobs compute nodes
  - dfuse is the userspace component of Linux FUSE which connects to DAOS servers
  - This will mount your specific container at a known location
- Running mpiexec must use the ‘--no-vni’ option
  - This enables the DAOS client to talk the DAOS servers
- Clean-up of the mount point is handled by the epilogue of the job script
- vi -R $HOME/hands-on/daos/job-posix.sh


Argonne
NATIONAL LABORATORY
Show Job Script

Test Cases

- Simple write/read tests coded different ways which are all compatible and all use a POSIX container
- Assumes target data file is in the root of the container
- Note about format
  - Binary file
  - repeated
    - small header
    - data “frame”
- Can change the size of the data written by modifying src/array.h
  - adjusting the XDIM and YDIM -> I/O request size and file size increased
  - ITER -> increase the file size
  - cd $HOME/${USER}_cont/daos/src && make clean && make

Run the test

- Submit the job
- qsub -q ATPESC -v DAOS_POOL=ATPESC2025,DAOS_CONT=${USER}_cont $HOME/hands-on/daos/job-posix.sh
- qstat -u $USER
- tail -f job-posix.sh.[eo]*
- ls -l $HOME/${USER}_cont

```
write
total time; 0.029693 second
data size: 3 KiB
total size: 141 KiB
BW: 4.640393 MiB/s
IOps 2.694244 Kops
read
total time; 0.003773 second
data size: 3 KiB
total size: 141 KiB
BW: 36.516270 MiB/s
IOps 21.201597 Kops
```

ARGONNE ATPESC2025 EXTREME-SCALE COMPUTING

ARGONNE TRAINING PROGRAM ON EXTREME-SCALE COMPUTING

Produced by Argonne National Laboratory, a U.S. Department of Energy Laboratory managed by UChicagoArgonne, LLC under contract DE-AC02-06CH11357.

Special thanks to the National Energy Research Scientific Computing Center (NERSC) and Oak Ridge Leadership Computing Facility (OLCF) for the use of their resources during the training event.