DAOS Next Generation Storage

Approved for public release

Kevin Harms (ALCF)
Mohamad Chaawari (Intel)
Johann Lombardi (Intel)
John Mellor-Crummey (Rice)
Yumeng Liu (Rice)

Agenda

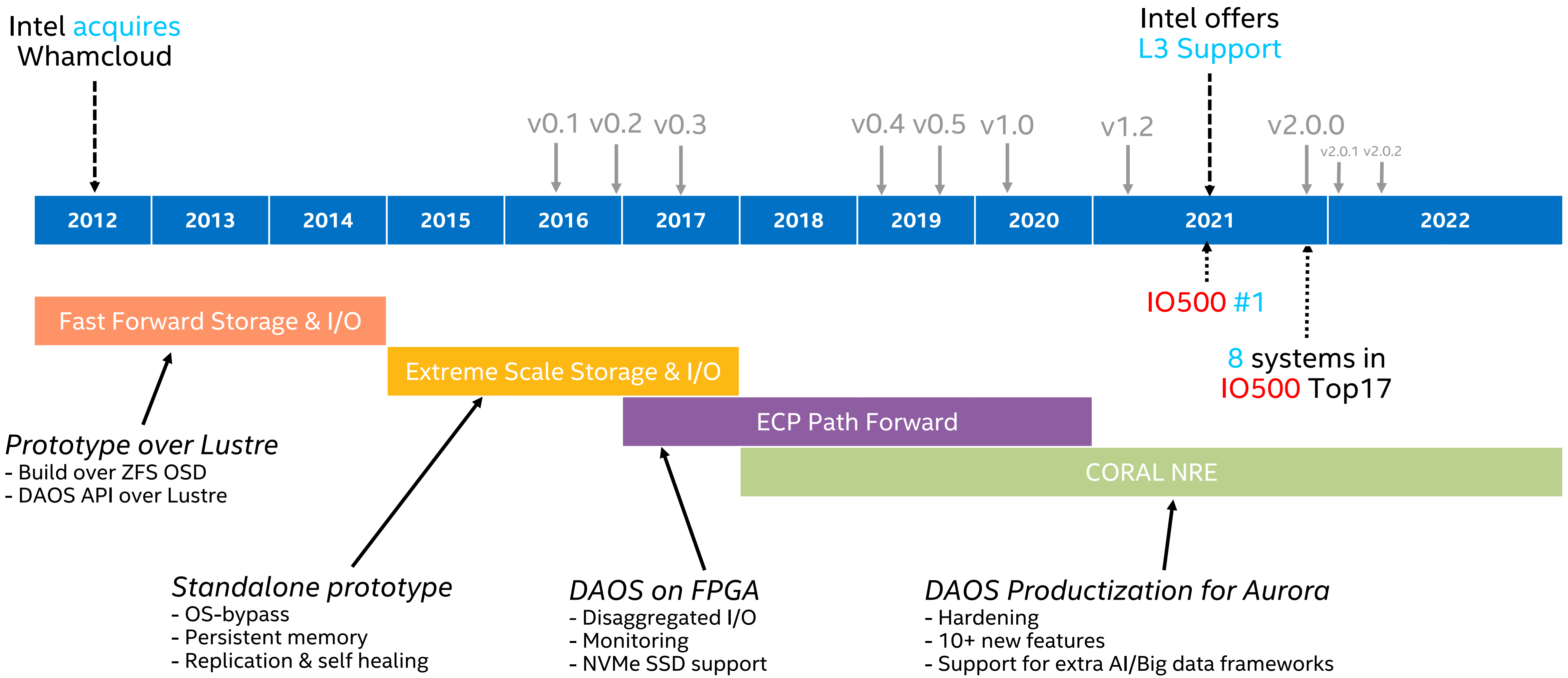
- Introductions
- DAOS Overview - Johann
- DFS Overview - Mohamad
- Accelerating I/O in HPCToolkit using DAOS - Yumeng
- Discussion - Kevin

DAOS Next Generation Storage

2022 ECP Community BOF Days

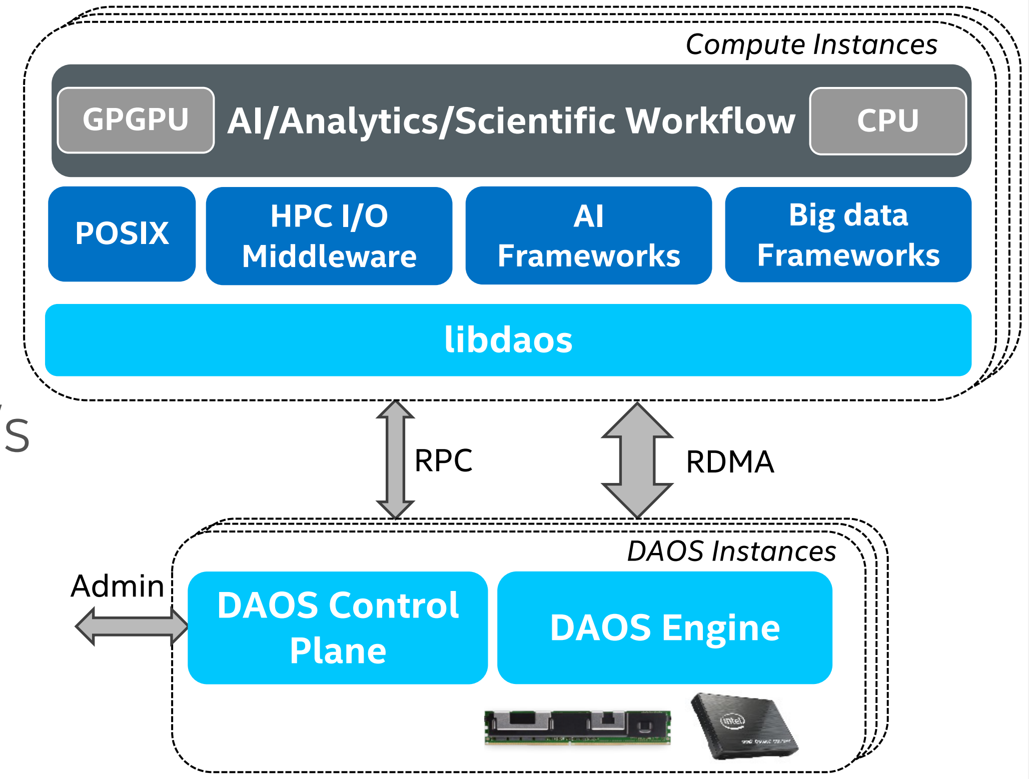
Notices and Disclaimers

DAOS History



DAOS: Nextgen Open Storage Platform

- Fully Distributed multi-tenant global namespace
- Platform for innovation
  - Modular API and layering
  - Can leverage latest HW & SW technology
- Built for high performance
  - 10’s µs latency, billions of IOPS, TB/s to PB/s
- Full userspace model
  - Run on-prem or in the cloud
- Growing open-source community

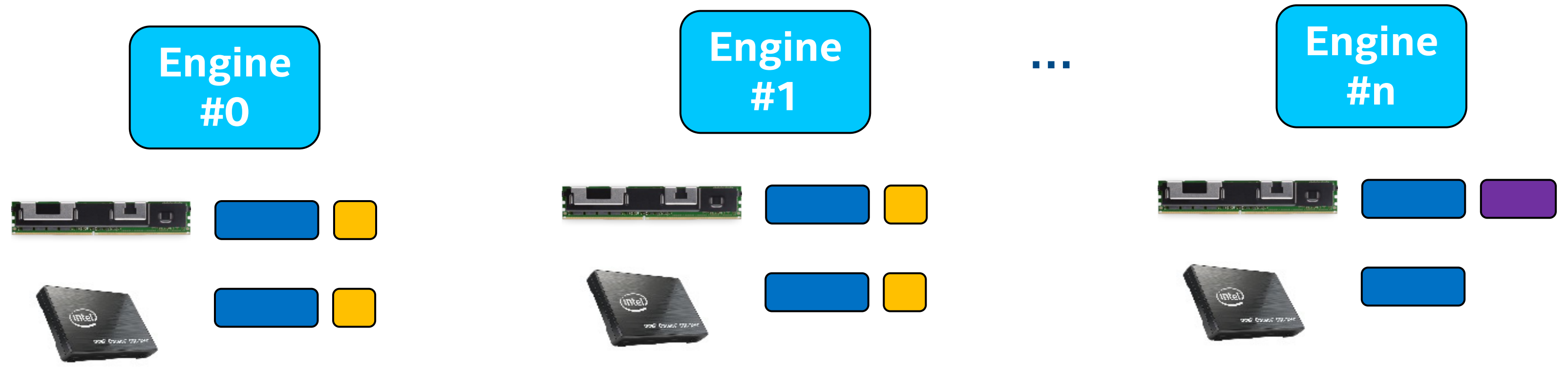


Storage Pooling

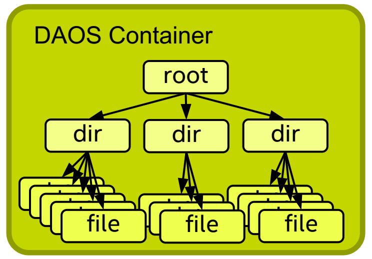
Pool 1 🟦	Project Apollo	100PB usable	20TB/s	200M IOPS
Pool 2 🟨	Project Gemini	10PB usable	2TB/s	20M IOPS
Pool 3 🟪	Project Mercury	30TB usable	80GB/s	2M IOPS

DAOS System

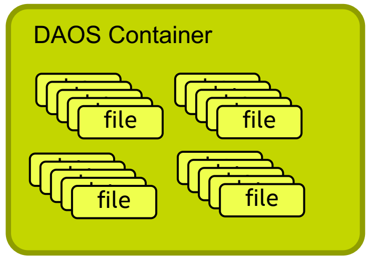
DAOS Nodes (DN)



Dataset Management

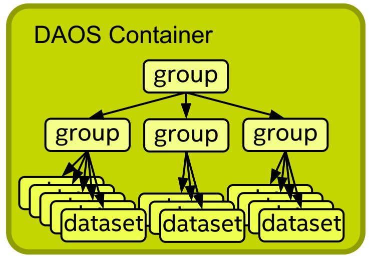


POSIX Namespace

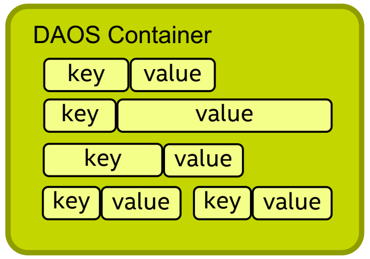


File-per-process

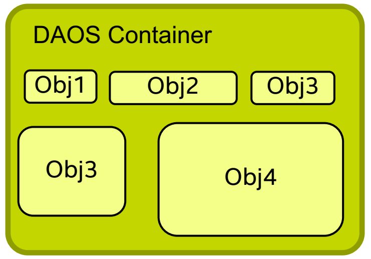
- New data model to unwind 30+y of file-based management
- Introduce notion of dataset: DAOS container
  - Basic unit of storage
  - Containers have a type
  - POSIX containers can have billions of files/directories
  - Many parameters
  - ACLs
- Advanced container query capabilities



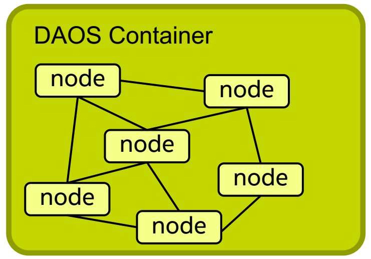
HDF5 « File »



Key-value store



S3 Bucket

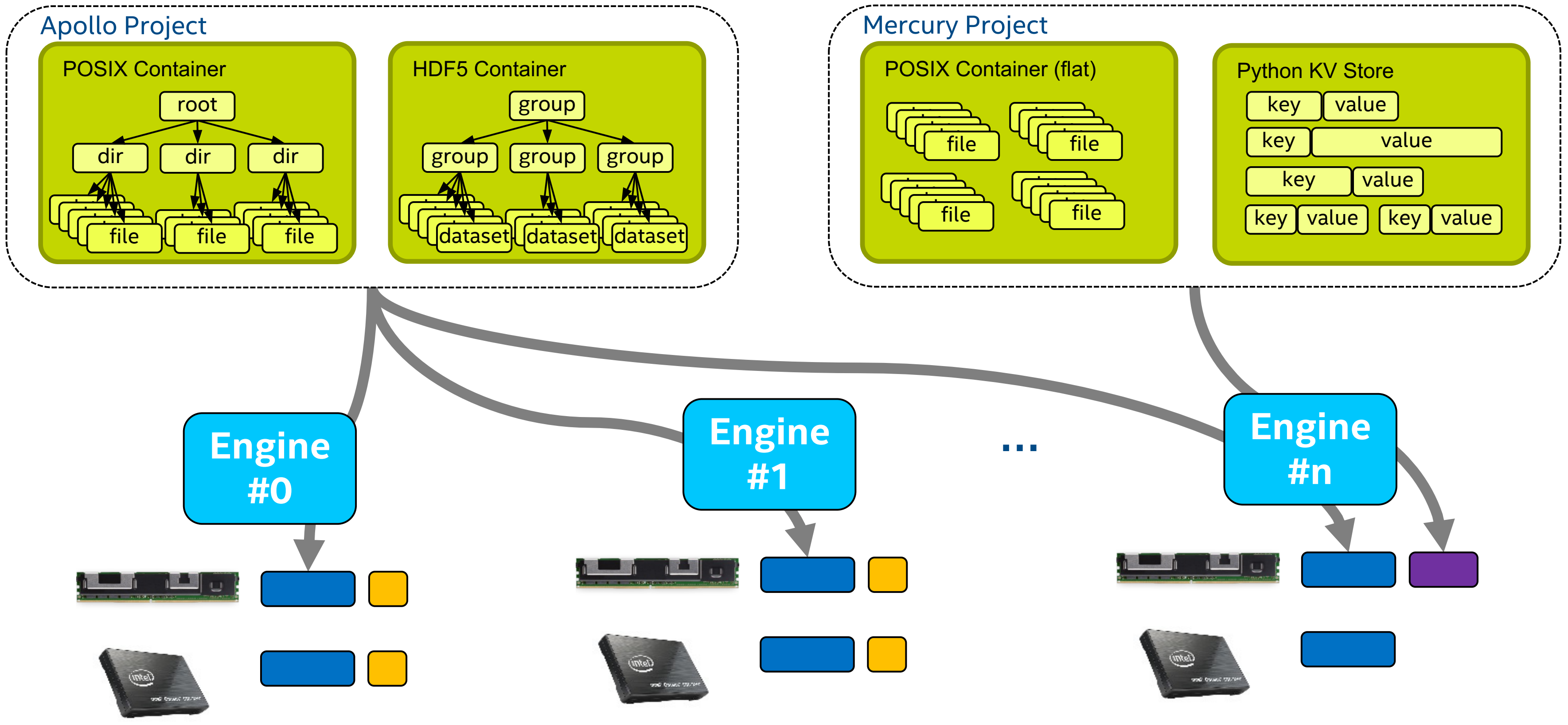


Graph

Storage Management

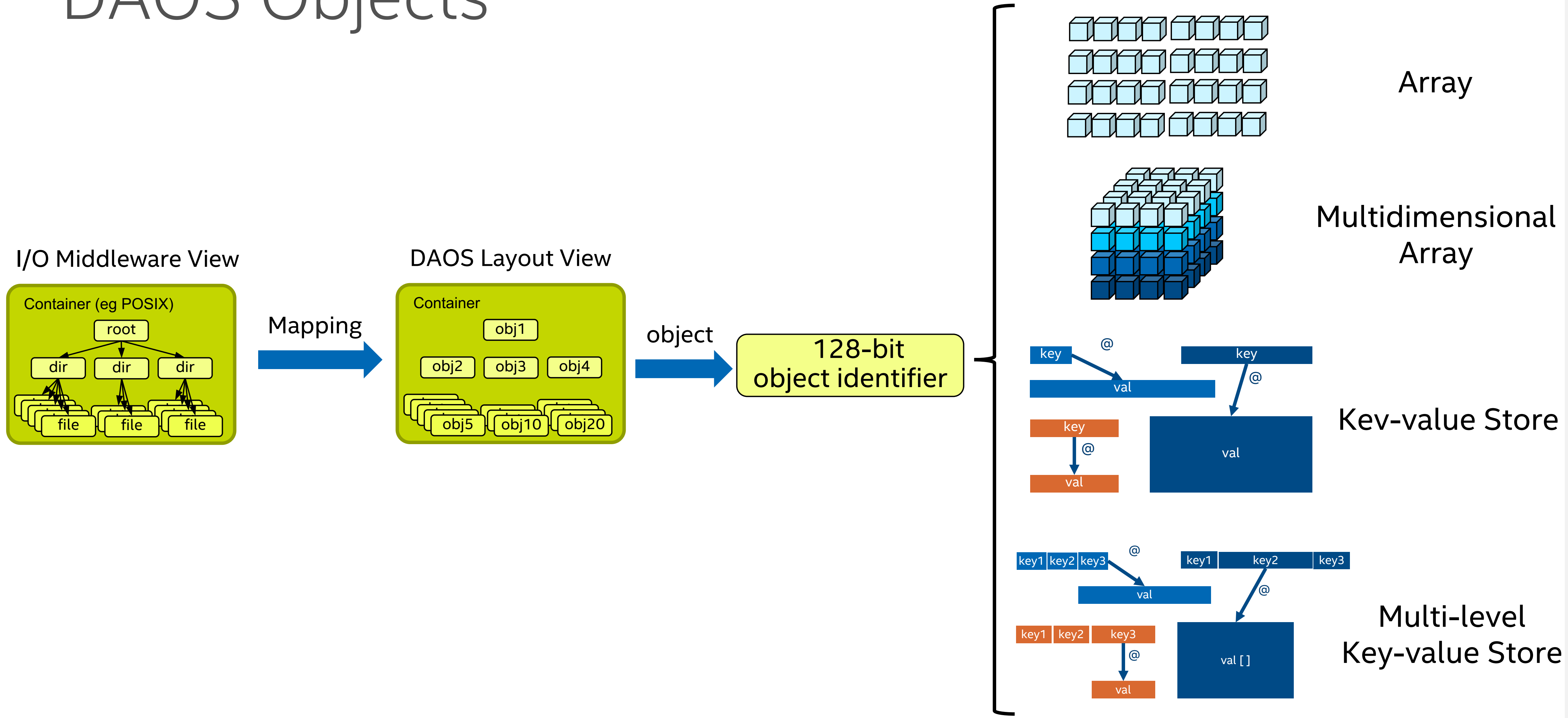
Applications
Compute Nodes (CN)

DAOS System
DAOS Nodes (DN)



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

DAOS Objects

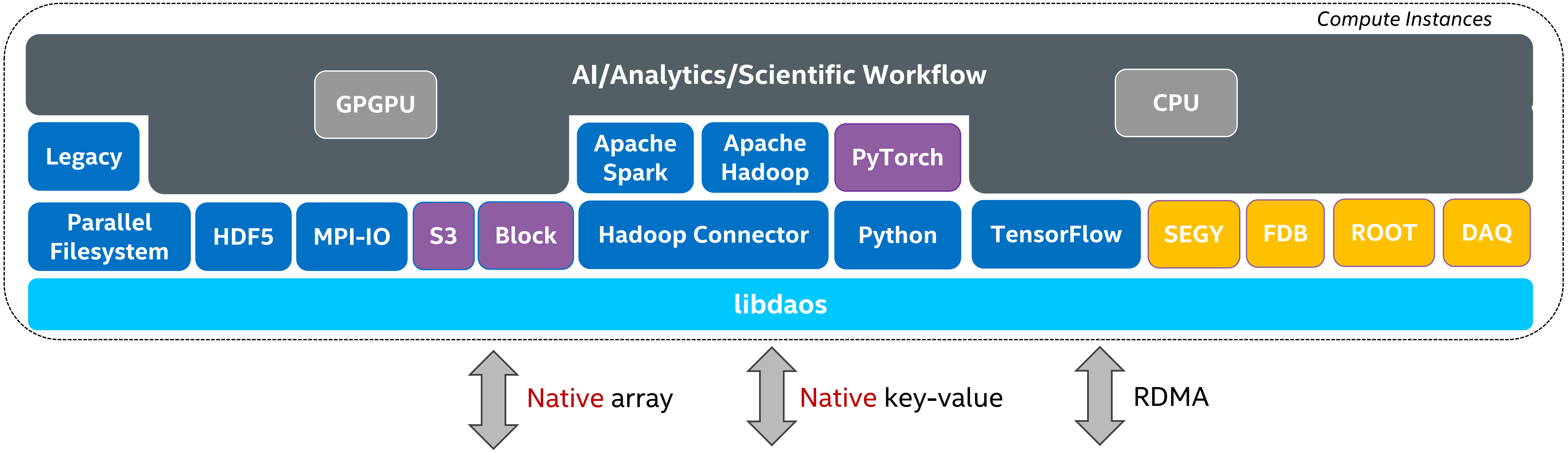


Software Ecosystem

🟦 Generic I/O middleware supported today

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

🟪 Enablement in progress



DAOS on Aurora

- 1024x DAOS nodes, each with:
  - 2x Xeon 5320 CPUs
  - 512GB DRAM
  - 8TB Optane Persistent Memory 200
  - 244TB NVMe SSDs
  - 2x HPE Slingshot NIC
- Usable capacity
  - between 220PB and 249PB depending on redundancy level chosen

DAOS Community Roadmap

1H'21 → **DAOS 1.2** *April 30, 2021*

Leap 15 support
End-to-end data integrity
Per-container ACL
Replication & self-healing
Quiescent server reintegration
HDF5 DAOS VOL Connector
POSIX I/O improvements
POSIX data mover
Apache Spark support
Erasure Code (Tech-Preview)

2H'21 → **DAOS 2.0** *December 23, 2021*

CentOS 8 support
Erasure Code
Telemetry & per-job statistics
Quiescent server drain
Libfabric stabilization
Improved usability

1H'22 → **DAOS 2.2**

OS support:
- Rocky 8.5 (CentOS 8 unsupported)

Networking:
- Libfabric TCP Provider support
- MLX UCX (Tech-Preview)

UX improvements:
- interoperability 2.0 and 2.2
- Intel VMD support
- Pool creation improvements
- DFS: mtime support

2H'22 → **DAOS 2.4**

OS support:
- Leap 15.4
- Rocky 8.6 (CentOS 7 unsupported)

Networking
- MLX UCX support
- OmniPath Express (Tech-Preview)
- Multiple engines per socket

DAOS Storage features:
- Online drain/reintegration/add
- Checksum scrubbing

UX improvements:
- SSD Hotplug & LED management
- Dfuse multi-user & optimizations
- Lustre Integration

1H'23 → **DAOS 2.6**

OS support:
- Ubuntu client

Networking:
- OmniPath Express
- Multihomed network support

DAOS Storage features:
- Catastrophic recovery
- Distributed transactions
- Container parking/serialization
- DSA optimizations
- Fault domains

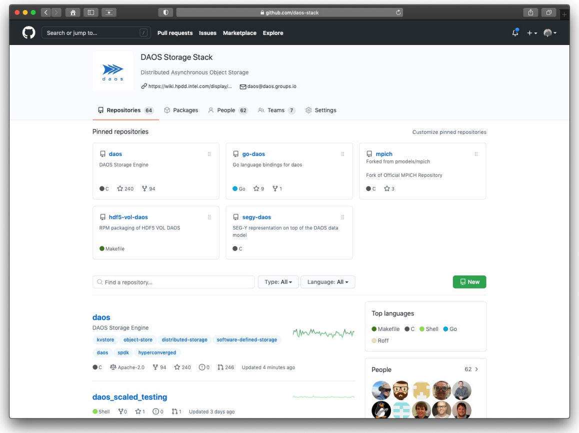
2H'23 → **DAOS 2.8**

Network topology awareness
DAOS Pool resizing
Tiering support
Inline compression (QAT)
Middleware Catastrophic Recovery

NOTE: All information provided in this roadmap is subject to change without notice.

Resources

- Open-source Community
  - Github: https://github.com/daos-stack/daos
  - Online doc: http://daos.io
  - Mailing list & slack: https://daos.groups.io
  - YouTube channel: http://video.daos.io
- 5th DAOS User Group (DUG'21)
  - Recordings available at http://dug.daos.io
- Upcoming BoF at ISC'22
- Intel landing page
  - https://www.intel.com/content/www/us/en/high-performance-computing/daos.html


github.com/daos-stack

Search or jump to... Pull requests Issues Marketplace Explore

DAOS Storage Stack
Distributed Asynchronous Object Storage
https://wiki.hpdd.intel.com/display/... daos@daos.groups.io

Repositories 64 Packages People 62 Teams 7 Settings

Pinned repositories — Customize pinned repositories

Repository	Description	Language	Stars	Forks
daos	DAOS Storage Engine	C	240	94
go-daos	Go language bindings for daos	Go	9	1
mpich	Forked from pmodels/mpich. Fork of Official MPICH Repository	C	3	
hdf5-vol-daos	RPM packaging of HDF5 VOL DAOS	Makefile		
segy-daos	SEG-Y representation on top of the DAOS data model	C		

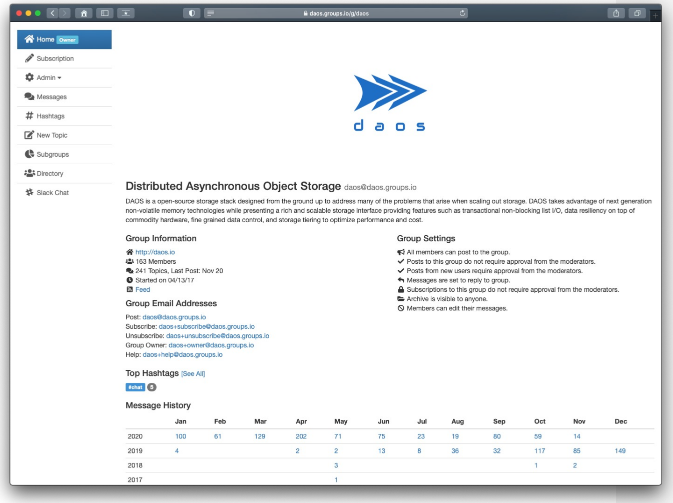
Find a repository... Type: All Language: All New

daos
DAOS Storage Engine
kvstore object-store distributed-storage software-defined-storage daos spdk hyperconverged
C Apache-2.0 94 240 0 246 Updated 4 minutes ago

daos_scaled_testing
Shell 0 1 0 1 Updated 3 days ago

Top languages
Makefile C Shell Go Roff

People 62


daos.groups.io/g/daos

Distributed Asynchronous Object Storage daos@daos.groups.io

DAOS is a open-source storage stack designed from the ground up to address many of the problems that arise when scaling out storage. DAOS takes advantage of next generation non-volatile memory technologies while presenting a rich and scalable storage interface providing features such as transactional non-blocking list I/O, data resiliency on top of commodity hardware, fine grained data control, and storage tiering to optimize performance and cost.

Group Information
- http://daos.io
- 163 Members
- 241 Topics, Last Post: Nov 20
- Started on 04/13/17
- Feed

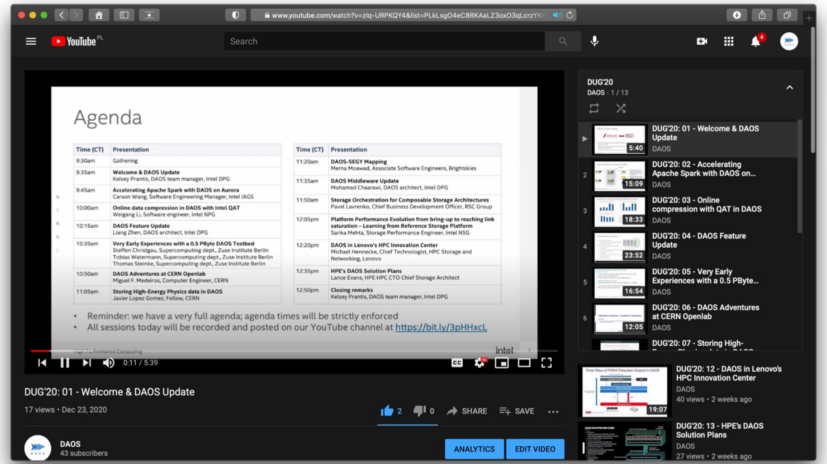
Group Email Addresses
- Post: daos@daos.groups.io
- Subscribe: daos+subscribe@daos.groups.io
- Unsubscribe: daos+unsubscribe@daos.groups.io
- Group Owner: daos+owner@daos.groups.io
- Help: daos+help@daos.groups.io

Group Settings
- All members can post to the group.
- Posts to this group do not require approval from the moderators.
- Posts from new users require approval from the moderators.
- Messages are set to reply to group.
- Subscriptions to this group do not require approval from the moderators.
- Archive is visible to anyone.
- Members can edit their messages.

Top Hashtags [See All]
#chat 5

Message History

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2020	100	61	129	202	71	75	23	19	80	59	14	
2019	4			2	2	13	8	36	32	117	85	149
2018					3					1	2	
2017					1							


www.youtube.com/watch?v=zlq-URPKQY4&list=PLkLsgO4eC8RKAaLZ3oxO3qLcrzYK

YouTube PL Search

Agenda

Time (CT)	Presentation
9:30am	Gathering
9:35am	**Welcome & DAOS Update** Kelsey Prantis, DAOS team manager, Intel DPG
9:45am	**Accelerating Apache Spark with DAOS on Aurora** Carson Wang, Software Engineering Manager, Intel IAGS
10:00am	**Online data compression in DAOS with Intel QAT** Weigang Li, Software engineer, Intel NPG
10:15am	**DAOS Feature Update** Liang Zhen, DAOS architect, Intel DPG
10:35am	**Very Early Experiences with a 0.5 PByte DAOS Testbed** Steffen Christgau, Supercomputing dept., Zuse Institute Berlin; Tobias Watermann, Supercomputing dept., Zuse Institute Berlin; Thomas Steinke, Supercomputing dept., Zuse Institute Berlin
10:50am	**DAOS Adventures at CERN Openlab** Miguel F. Medeiros, Computer Engineer, CERN
11:05am	**Storing High-Energy Physics data in DAOS** Javier Lopez Gomez, Fellow, CERN

Time (CT)	Presentation
11:20am	**DAOS-SEGY Mapping** Merna Moawad, Associate Software Engineers, Brightskies
11:35am	**DAOS Middleware Update** Mohamad Chaarawi, DAOS architect, Intel DPG
11:50am	**Storage Orchestration for Composable Storage Architectures** Pavel Lavrenko, Chief Business Development Officer, RSC Group
12:05pm	**Platform Performance Evolution from bring-up to reaching link saturation – Learning from Reference Storage Platform** Sarika Mehta, Storage Performance Engineer, Intel NSG
12:20pm	**DAOS in Lenovo's HPC Innovation Center** Michael Hennecke, Chief Technologist, HPC Storage and Networking, Lenovo
12:35pm	**HPE's DAOS Solution Plans** Lance Evans, HPE HPC CTO Chief Storage Architect
12:50pm	**Closing remarks** Kelsey Prantis, DAOS team manager, Intel DPG

- Reminder: we have a very full agenda; agenda times will be strictly enforced
- All sessions today will be recorded and posted on our YouTube channel at https://bit.ly/3pHHxcL

High Performance Computing intel

0:11 / 5:39

DUG'20: 01 - Welcome & DAOS Update
17 views • Dec 23, 2020
2 0 SHARE SAVE
DAOS 43 subscribers ANALYTICS EDIT VIDEO

DUG'20
DAOS - 1 / 13

1. DUG'20: 01 - Welcome & DAOS Update — DAOS — 5:40
2. DUG'20: 02 - Accelerating Apache Spark with DAOS on... — DAOS — 15:09
3. DUG'20: 03 - Online compression with QAT in DAOS — DAOS — 18:33
4. DUG'20: 04 - DAOS Feature Update — DAOS — 23:52
5. DUG'20: 05 - Very Early Experiences with a 0.5 PByte... — DAOS — 16:54
6. DUG'20: 06 - DAOS Adventures at CERN Openlab — DAOS — 12:05
7. DUG'20: 07 - Storing High-

DUG'20: 12 - DAOS in Lenovo's HPC Innovation Center — DAOS — 40 views • 2 weeks ago — 19:07

DUG'20: 13 - HPE's DAOS Solution Plans — DAOS — 27 views • 2 weeks ago intel®

DAOS NEXT GENERATION STORAGE

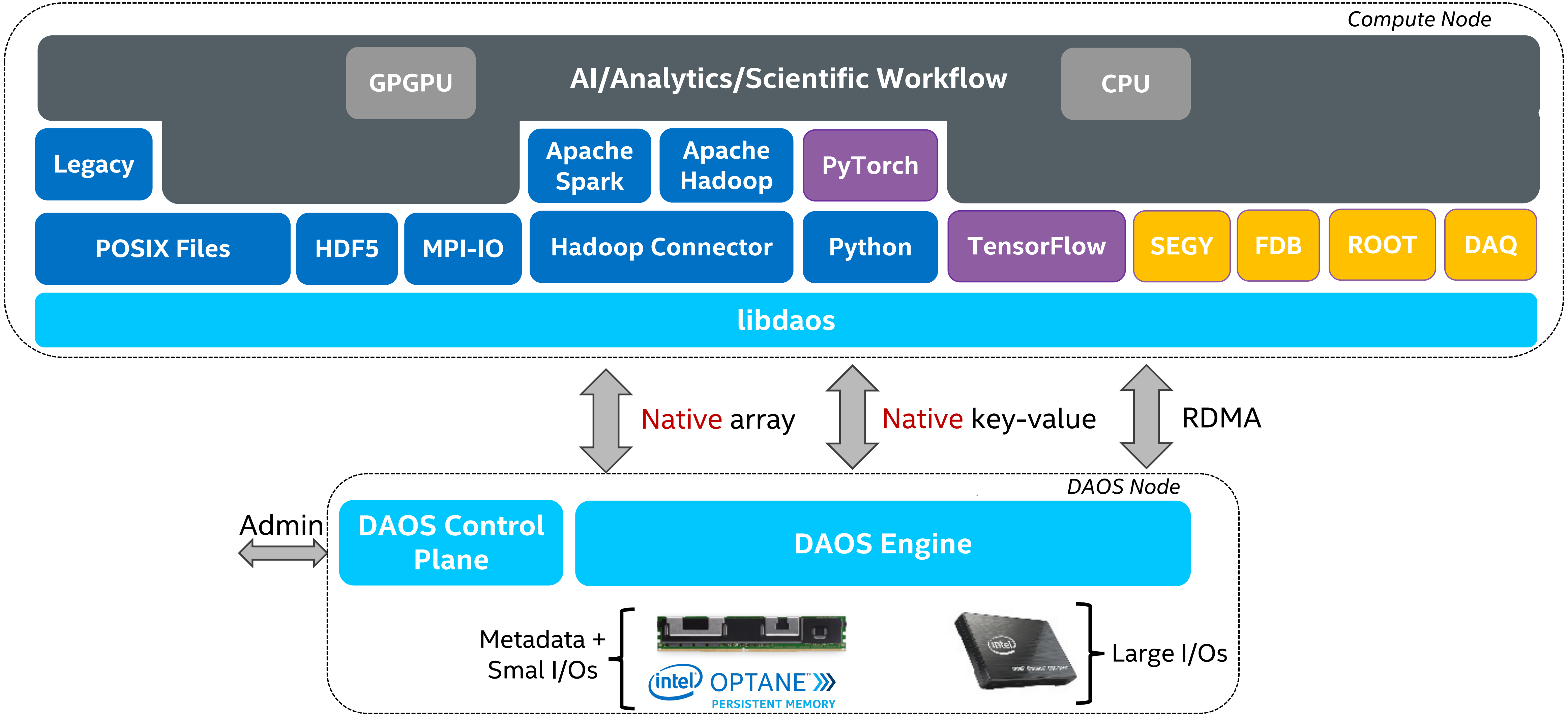
Johann Lombardi, Mohamad Chaarawi (Intel)
Kevin Harms (Argonne National Laboratory)

May 11, 2022

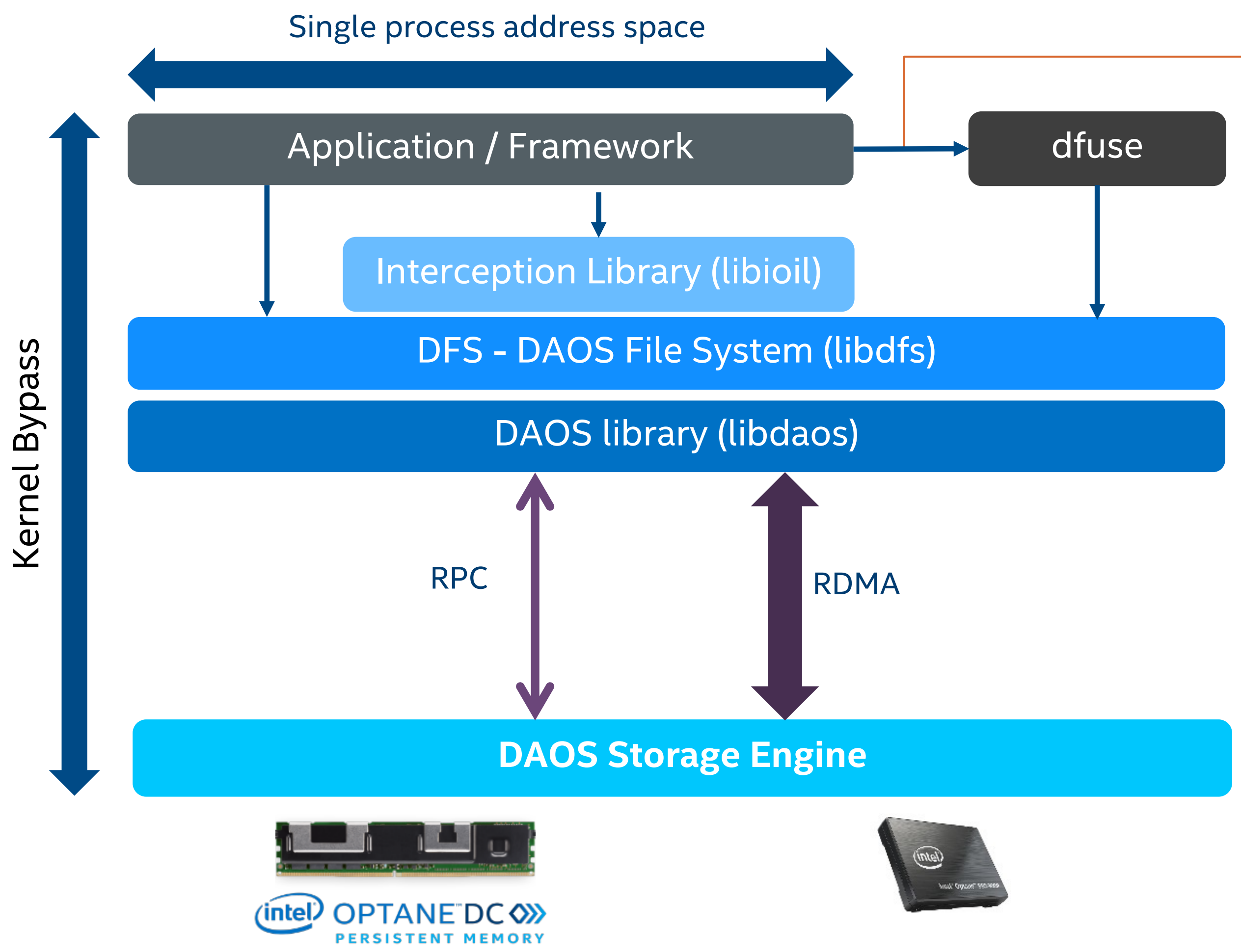
Notices and Disclaimers

DAOS Ecosystem

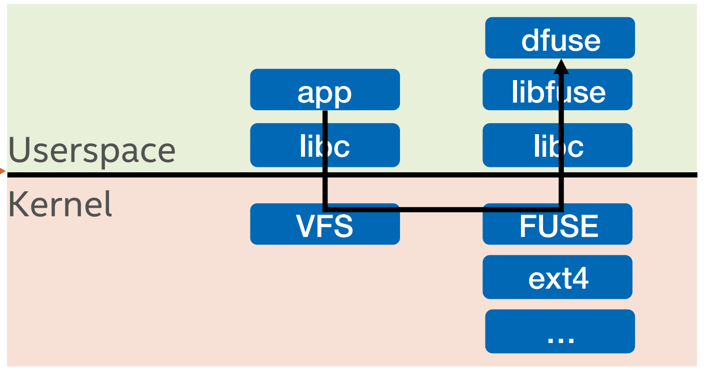
🟦	Generic I/O middleware supported today
🟨	Domain-specific data models under development in co-design with partners
🟪	Enablement in progress



POSIX I/O Support



Single process address space
Kernel Bypass
Application / Framework
dfuse
Interception Library (libioil)
DFS - DAOS File System (libdfs)
DAOS library (libdaos)
RPC
RDMA
DAOS Storage Engine
intel OPTANE DC PERSISTENT MEMORY



- 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 raw data IO only.

How to use DFS?

- You should have access to a pool (identified by a string label).
- Create a POSIX container with the daos tool:
  - `daos cont create mypool --label=mycont --type=POSIX`
  - Alternatively, you can programmatically create a container to use directly in your application (if you are using DFS and changing your app).
- Open the DFS mount:
  - `dfs_connect (mypool, mycont, O_RDWR, .. &dfs);`
  - `dfs_disconnect (dfs);`

DFS API

POSIX	DFS
mkdir(), rmdir()	dfs_mkdir(), dfs_rmdir()
open(), close(), access()	dfs_open(), dfs_release(),dfs_lookup()
pwritev(), preadv()	dfs_read/write()
{set,get,list,remove}xattr()	dfs_{set,get,list,remove}xattr
stat(), fstat()	dfs_stat(),ostat()
readdir() …	dfs_readdir() …

- Mostly 1-1 mapping from POSIX API to DFS API.
- Instead of File & Directory descriptors, use DFS objects.
- All calls need the DFS mount which is usually done once initialization.

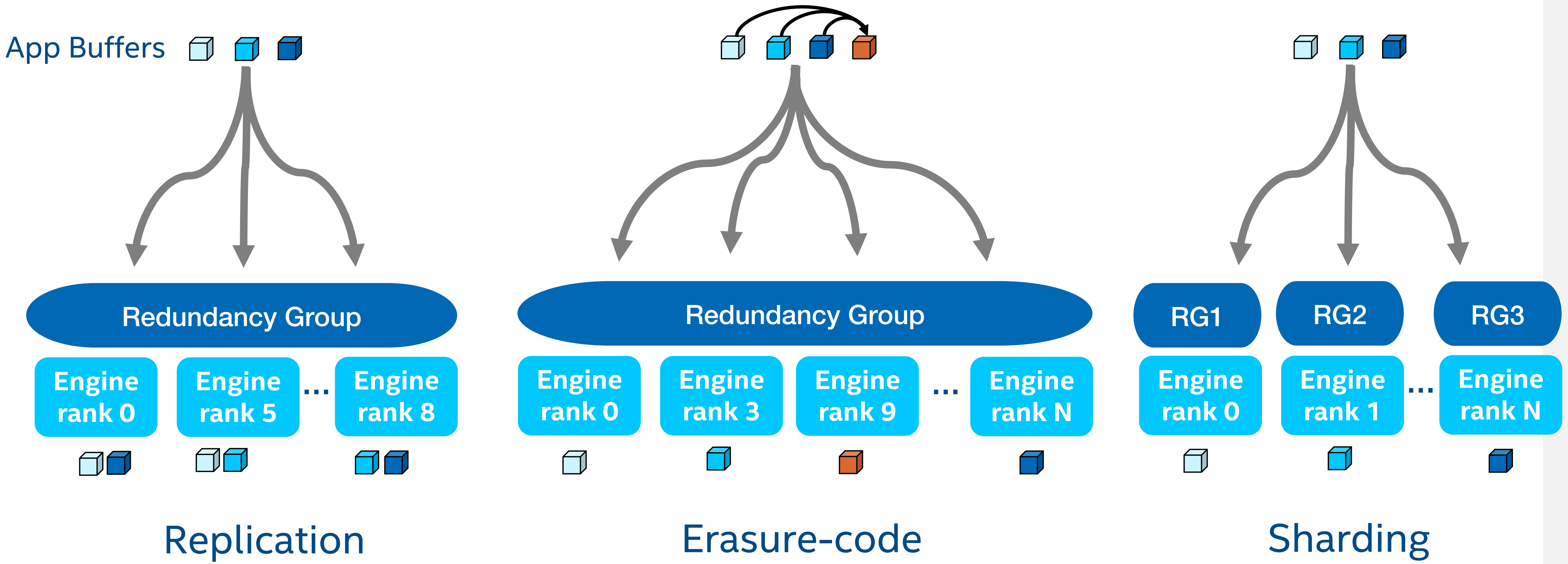
DFUSE

- To mount an existing POSIX container with dfuse, run the following command:
  - `dfuse --pool mypool --container mycont -m /mnt/dfuse`
  - No one can access your container / mountpoint unless access is provided on the pool and container (through ACLs).
  - Multi-user concurrent support is planned for DAOS 2.4
- Now you can access files / directories as any namespace in the container, and applications can run without any modifications (the easy path).
- Interception Library:
  - This library works in conjunction with dfuse and allow to interception of POSIX I/O calls and issue the I/O operations directly from the application context through libdaos without any application changes.
  - This provides kernel-bypass for I/O. To use this set the LD_PRELOAD to point to the shared library in the DOAS install dir
    - LD_PRELOAD=/path/to/daos/install/lib/libioil.so

To Intercept or Not Intercept

- The dfuse interception library bypasses the fuse kernel and sends IO to DFS directly
  - No caching (no writeback / readahead)
  - Works well for bulk IO.
  - Does not work well for very tiny IO (order of bytes) where dfuse without interception takes advantage of the kernel write back and read ahead caching to reduce IO over the wire.

DAOS Data Model: Distribution & Fault Tolerance



DAOS Object Class

- Each DAOS object is identified by an object ID which encode the object class.
- The object class controls the redundancy and sharding of the object and allows the client to do algorithmic placement when accessing the object.
  - Redundancy:
    - None: S
    - N-way Replication: RP_n
    - Erasure Code: EC_dPp (d = data, p = parity)
  - Sharding:
    - Single target/group:
      - S1, RP_nG1, EC_dPpG1
    - Widely / max sharding:
      - SX, RP_nGX, EC_dPpGX

Container Level Redundancy Setting

- On Container create, one can set the redundancy factor property (`--properties=rf:n`):
  - 0 (default): no redundancy
    - One can create objects with higher redundancy after
  - 1: tolerate 1 failure domain
    - At least RP2 or ECxP1
  - Up to 4
- DAOS automatically selects an object class for files and directories using the redundancy factor.
  - Widely striped, EC oclass for files (SX, EC_xP1GX, EC_xP2GX, etc.)
  - Single stripe, RP oclass for directories (S1, RP_2GX, RP_3GX, etc.)

Setting the Object Class

- Users can select the object class if the default is not appropriate.
  - Default works well if files are large, directories are small (not many entries)
- DFS API allows users to pass an object class when creating an object
  - Passing 0 would use the default
- DAOS tool allows also to create a **new** file with a specific oclass.
  - `daos fs set-attr --path=/mnt/dfuse/testfile --oclass=RP_2G1`
- DAOS tool allows changing the object class on a directory to use that oclass for all entries created in that directory (does not change the oclass of the directory itself)
  - `mkdir /mnt/dfuse/dir1`
  - `daos fs set-attr --path=/mnt/dfuse/dir1 --oclass=RP_2G1`

Striping Considerations

- Widely striped objects
  - Best IO BW (read/write) for single shared file, Highest IOPS for entry creation in single shared directory
  - Slow file stat (file size), ls (directory listing)
    - Must query all targets
    - Caching with dfuse helps in this case – limited to single node or read-only
- Small stripe objects
  - Limited size and bandwidth to number of targets
  - Good for FPP or DPP
  - Fast stat, ls (not querying all targets on storage system)

Redundancy Considerations

- Erasure Code performance characterization for IO:
  - Best for large IO access patterns
  - Full stripe write: 33% performance hit
  - Partial stripe write: 66% performance hit
  - Read should be the same.
- Replication:
  - Best for metadata objects (directories)
  - Small files (<= 16k)
  - Write IOPS: n x slower
  - Read IOPS: equal or better – more shards to serve concurrent requests

Challenges / Future Work

- Selecting correct object class still not straightforward:
  - Auto object class selection works fine but is not ideal in some situations.
- Extend (POSIX) container create options to allow users to provide hints as to what their expected dataset looks like:
  - small/medium/large directories
  - small/medium/large files
  - Based on those hints, the auto object class selection can be tuned to select better object class for performance considerations in such scenarios
- Extend DAOS FS tool to accept hints too for setting object class on parent directories or new files.
- *Likely DAOS 2.4 features*

Accelerating I/O in HPCToolkit using DAOS

Yumeng Liu and John Mellor-Crummey

Rice University

ECP Community BoF Days

11 May 2022

HPCToolkit Funding Acknowledgments

- **Exascale Computing Project 17-SC-20-SC**
- **Lawrence Livermore National Laboratory Subcontract No. B645220**
- **Argonne National Laboratory Subcontract 9F-60073**
- **Advanced Micro Devices**
- **Intel Corporation**
- **TotalEnergies EP Research & Technology USA, LLC.**

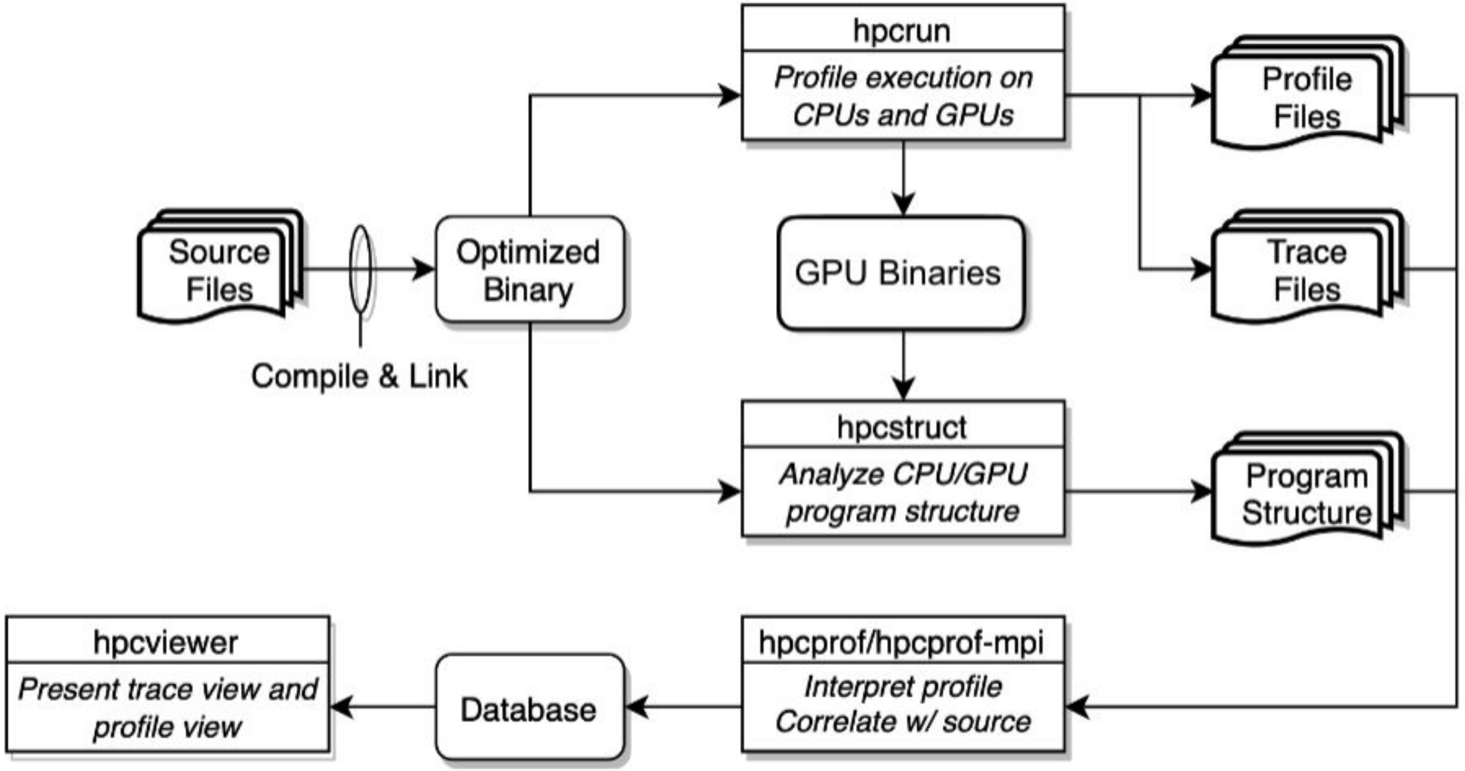
Outline

- HPCToolkit Project Overview
- Why DAOS in HPCToolkit?
- An I/O abstraction layer for HPCToolkit
- Integration experience
    - DAOS usage
    - Choice of parameters
    - Challenges
    - Differences from POSIX
- Early results

HPCToolkit Project Overview

- **Goals**
  - Develop effective tools to measure, attribute, analyze, and understand performance of applications, libraries, tools, and system software on extreme-scale parallel systems
  - Develop new strategies and mechanisms to measure and analyze performance and resource utilization of GPU-accelerated compute nodes
  - Influence the evolution of hardware and software ecosystems to enable better tools
- **Team**
  - Lead Institution: Rice University (HPCToolkit performance tools)
    - PI: Prof. John Mellor-Crummey
    - Research staff: Dr. Laksono Adhianto, Dr. Mark Krentel, Dr. Xiaozhu Meng, Dr. Scott Warren
    - Grad students: Keren Zhou, Jonathon Anderson, Yumeng Liu, Aaron Cherian, Dejan Grubisic
  - Subcontractor: University of Wisconsin – Madison (Dyninst binary analysis toolkit)
    - Lead: Prof. Barton Miller

HPCToolkit Project Overview



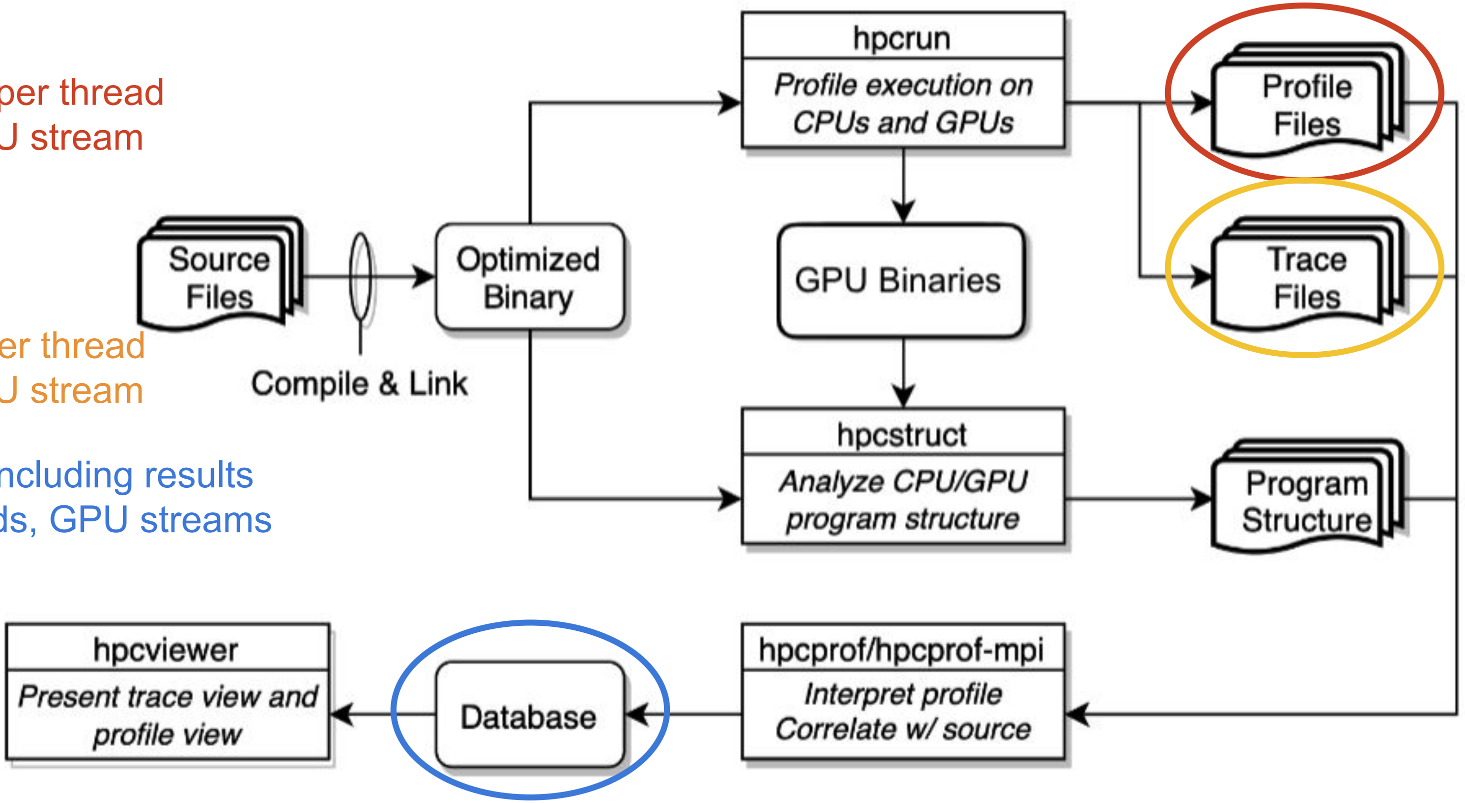
EXASCALE COMPUTING PROJECT

HPCToolkit Project Overview

One profile per thread and per GPU stream

One trace per thread and per GPU stream

Three files including results for all threads, GPU streams and traces



Why DAOS in HPCToolkit? - I

Concern: Opening millions of small measurement files in parallel

- Without DAOS: metadata server can be a bottleneck
  - Solution:
    - merge profiles and traces for one rank or even one node in one file during writing
    - separate the profiles and traces on the fly in post-processing

- With DAOS: no default metadata besides 128bits
  - We can keep our original code: one profile and one trace per thread and per GPU stream

Why DAOS in HPCToolkit? - II

Concern: **Parallel assembly of a file with irregular structure**

- All processes and threads know their offsets for writing

- Without DAOS:
  - Overhead for maintaining page cache consistency between multiple processes on different nodes

- With DAOS:
  - No cache, no extra overhead

EXASCALE COMPUTING PROJECT

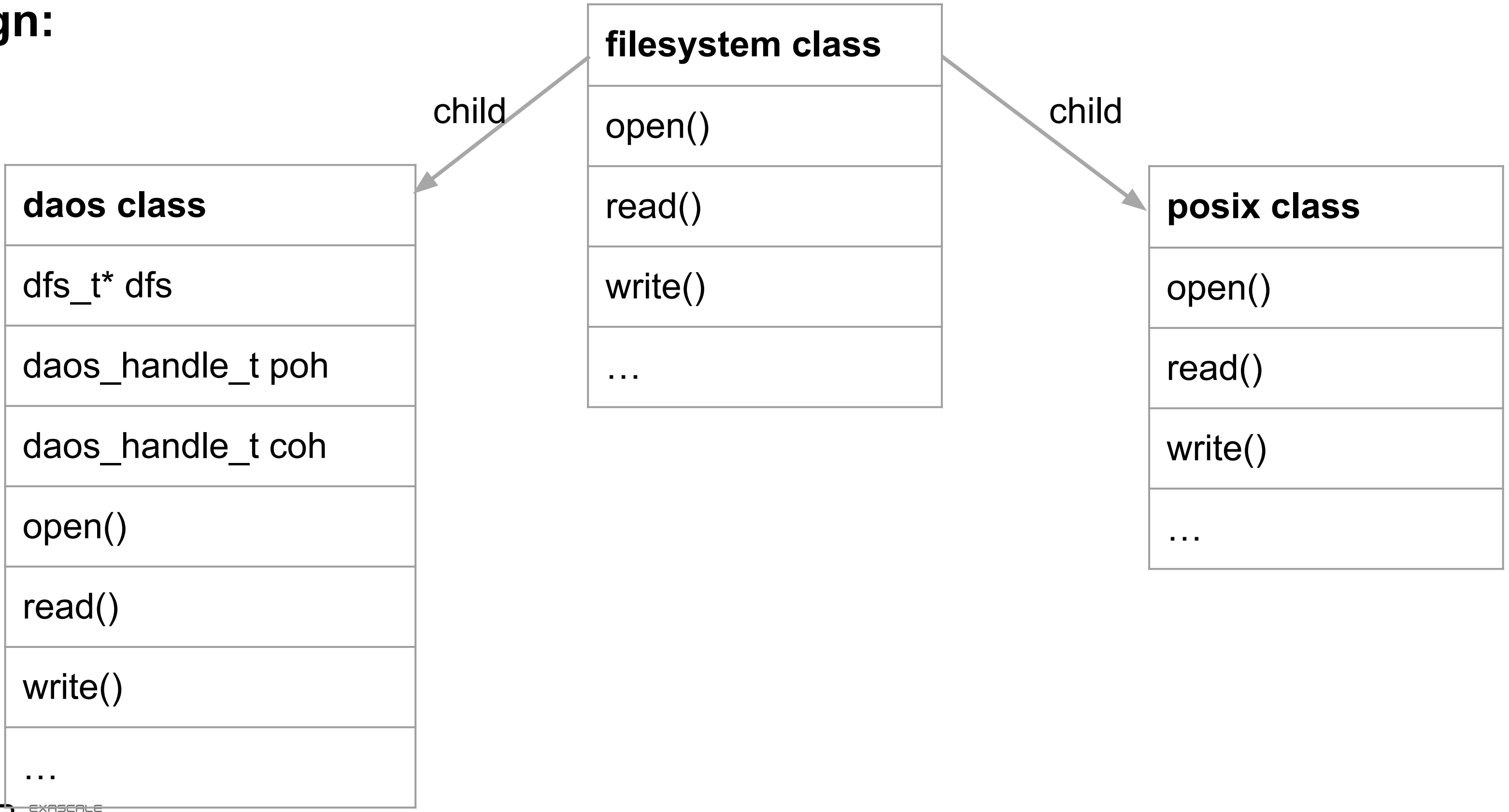
An I/O abstraction layer for HPCToolkit

Goals:

- Freedom to choose any I/O option (DAOS, POSIX, Lustre …) for any file
  - Example:
    - `hpcprof -o daos://<POOL>/<CONT>/database-dir measurement-dir`
    - We use DAOS for database and use POSIX for measurement files at the same time
- Easy integration of other I/O options in the future
  - Example: HPE Rabbit Near Node Storage (Livermore’s El Capitan), Lustre ...
- Details of I/O are invisible to other parts of the software
  - Example: they just call `io->write(...)` without concern for underlying I/O implementations

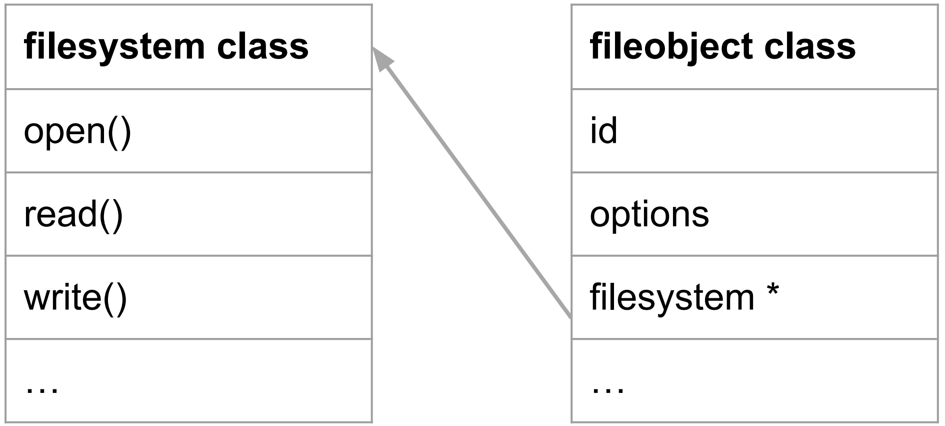
An I/O abstraction layer for HPCToolkit

Design:



An I/O abstraction layer for HPCToolkit

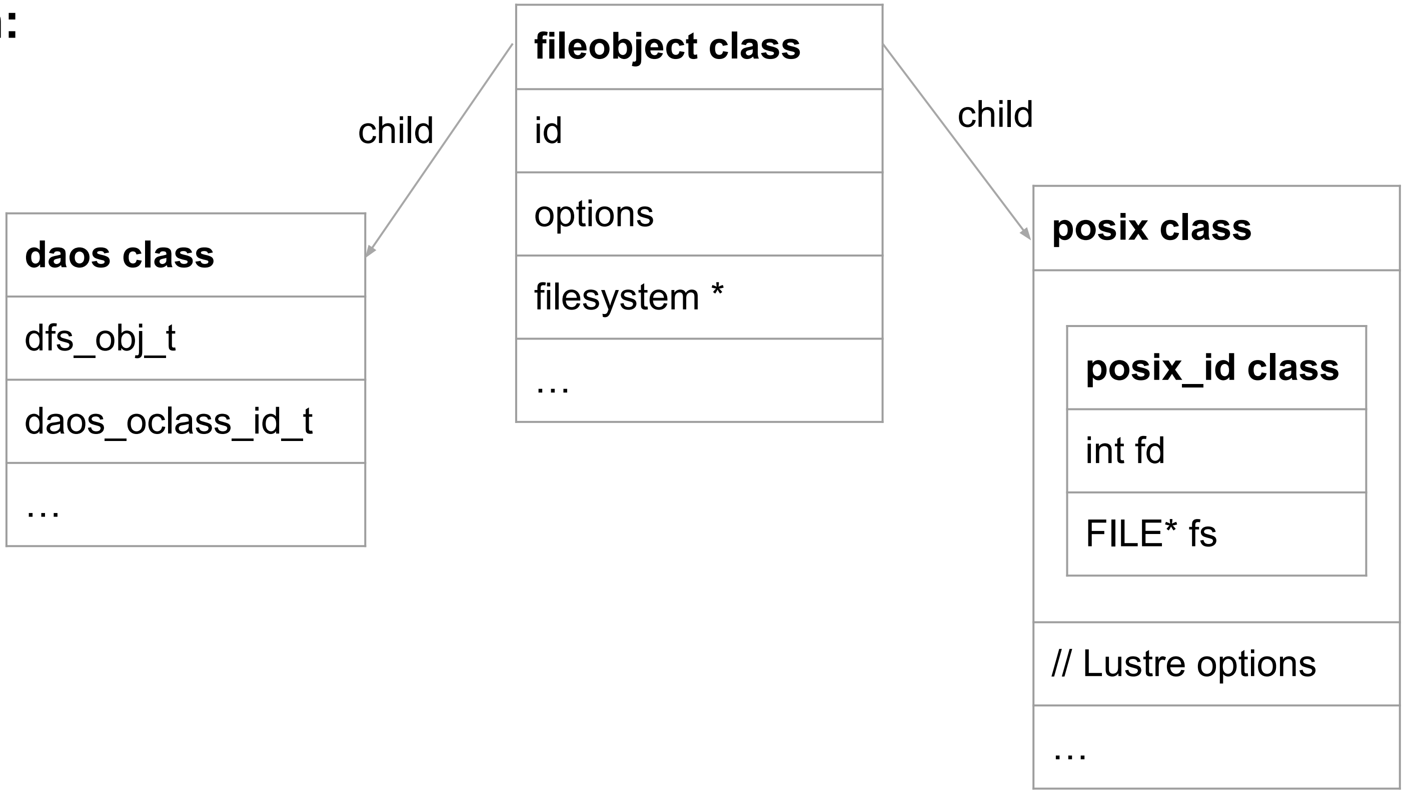
Design:



Pointer to the filesystem containing this object

An I/O abstraction layer for HPCToolkit

Design:



An I/O abstraction layer for HPCToolkit

Simplified example of calling open():

```
hpctio_obj_t * hpctio_obj_open(const char *path, int flags, mode_t md, int
wmode, int sizetype, hpctio_sys_t * sys)
{
   hpctio_obj_t * obj = (hpctio_obj_t *) malloc(sizeof(hpctio_obj_t));

   obj->sys_ptr = sys;
   obj->opt_ptr = sys->create_obj_options(wmode, sizetype);
   obj->id = sys->open(path, flags, md, opt, sys->params);

   return obj;
}
```

An I/O abstraction layer for HPCToolkit

Revisit goals:

✓ Freedom to choose any I/O option (DAOS, POSIX, Lustre …) for any file
  - By defining different filesystem objects

✓ Easy integration of other I/O options in the future
  - Just define another children pair of filesystem class and fileobject class

✓ Details of I/O are invisible to other parts of the software
  - After defining filesystem, pass it in when opening the file object, no details exposed

Integration experience - DAOS usage

	bypass OS kernel for read/write	bypass OS kernel for metadata operations	Use more than one pool and one container at a time
DFuse			
DFuse + IL	YES		
DFS	YES	YES	YES

We expect better performance with DFS API directly

Integration experience - choice of parameters

- Object Class
  - Profile files and Trace files: **OC_S1**
    - no striping: several MB, relatively small
    - no redundancy: not concerned about file loss, ok to lose a few profiles out of many
  - Analysis files: **OC_EC_2P1GX (Aurora: OC_EC_16P2GX)**
    - maximum striping:
      - MB - GB, relatively large
      - written in parallel by multiple processes and threads
    - erasure coding: don’t want to lose analysis results

- Chunk size
  - default 1MB

Integration experience - challenges

No matching DAOS function for every POSIX function

- `fread, fwrite, fseek` …
  - `fread`: We reworked our code to avoid it by using `read_at`
  - `fwrite`: DAOS only has `write_at`, manage our own buffer and cursor
  - `fseek`: We reworked our code to avoid it by using `read_at` and `write_at`
- `write`
  - Multiple processes append to the same log file (debug info for developers)
  - With `write_at`, we need to share a buffer or a cursor between processes
  - We reworked so that each process creates its own log file
- C++ filesystem abstractions (`directory_iterator`, `exists`, `remove_all` …)
  - We could use them if we chose DFuse + IL
  - With DFS API, we needed to reimplement these abstractions using our I/O abstraction layer

EXASCALE COMPUTING PROJECT

Integration experience - differences from POSIX

- Extra initialization and finalization steps
  - Each process initializes its own DAOS pool and container handles
  - We set up the output directory and begin recording trace data BEFORE MPI initialization
- Need to manage file system accesses carefully
  - Before: use POSIX anywhere without any direct coordination
  - Now: pass around I/O abstractions to access the correct file system
    - e.g., `hpctio_sys_mkdir(const char *path, mode_t md,` **`hpctio_sys_t* sys`**`)`
- Caching semantics
  - Objects created were only visible through command line after cache refreshed
  - Using `--disable-caching` can help, but may hurt performance

DAOS Testbed: Presque on JLSE

8 Clients

- Intel Xeon Gold 6140 (Skylake) CPU @ 2.30Ghz with 18 cores
- 192 GB DDR4
- Mellanox ConnectX-5

4 Servers

- Dual Intel Xeon Gold 6248 (Cascade Lake) CPU @ 2.5 GHz with 22 cores
- 384GB DDR4-2666 DRAM
- 1.5TB Intel Optane
- 12TB Intel P4510 NVMe SSDs
- 2 Mellanox ConnectX-5

Collecting data for DAOS experiments with `hpcprof`

Measurement files from AMG2006 on Theta

- 1 CPU metric
- 8 nodes, 512 ranks, 2048 threads
- profile files: ~948MB
    - 512 files: ~1.5MB each
    - 1544 files: ~120KB each
- trace files: ~22GB
    - 2056 files: ~11MB each

Early Results

- Run hpcprof on 1 node of Presque with 1 MPI rank, 36 threads
  - POSIX: 4m56s
  - DAOS: 1m50s

- Using DAOS I/O is 2.69x faster than POSIX I/O
- With POSIX, data may still be in the disk cache, may not be in stable storage
- We expect to have larger benefits from using DAOS on Aurora

Discussion

EXASCALE COMPUTING PROJECT

Acknowledgements

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.

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.