Advancing HDF5's Parallel I/O for Exascale with DAOS

October 15, 2020

S. Breitenfeld, N. Fortner, J. Henderson, R. Lu, E. Pourmal, D. Robinson, J. Soumagne

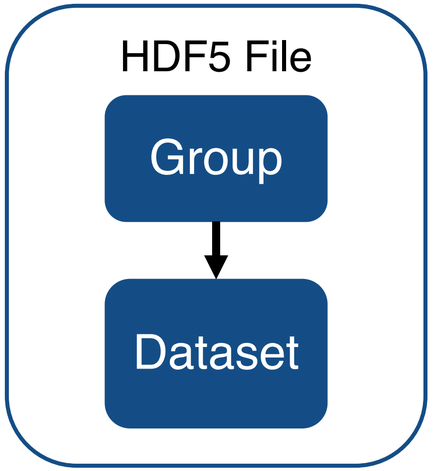
The HDF Group

The HDF Group

Outline

- **Current state of HDF5 parallel I/O**
  - Why is it not sufficient for Exascale?
- **Introduction to Intel Distributed Asynchronous Object Storage (DAOS)**
- **HDF5 DAOS VOL connector and file format**
- **New features and usage**
- **Early performance results**
- **Current status**

Current state of HDF5 parallel I/O

- 


```
H5Pset_fapl_mpio()
H5Fcreate()
H5Gcreate()
H5Dcreate()
H5Dwrite()
H5Dread()
```

Collective metadata operations

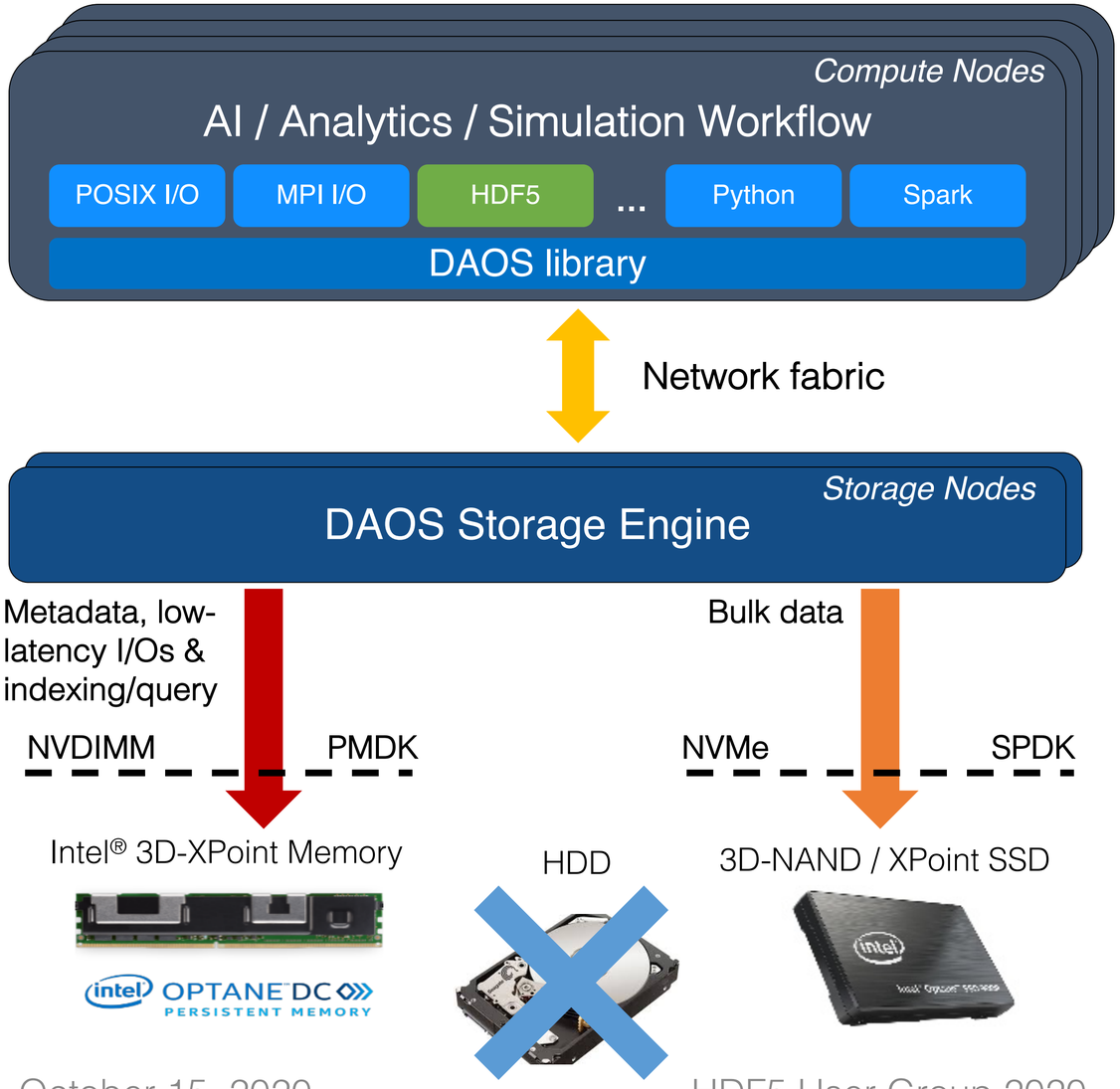
Collective or Independent raw data operations

- **POSIX was designed for HDDs (serial address space, offset/lseek)**
- **Current HDF5 native format and POSIX I/O impose global serialization**

Mitigations: subfiling / file per process I/O
Added complexity, always limited by POSIX

Intel ® DAOS

Credit: Mohamad Chaarawi (Intel Corporation)



- DAOS library directly **linked** with the applications
- No need for **dedicated** cores
- **Low** memory/CPU **footprint**
- End-to-end OS **bypass**
- Non-blocking, lockless, snapshot support, …

- **Low-latency** & **high-message-rate** communications
- Native support for **RDMA & scalable** collective operations
- Support for OPA, Infiniband, OPA, Slingshot, RoCE, …

- **Fine-grained I/O** with media selection strategy
- Only application data on SSD to **maximize throughput**
- Small I/Os **aggregated** in pmem & migrated to SSD in large chunks
- Full user space model with **no system calls** on I/O path
- **Built-in storage** management infrastructure (control plane)
- NFSv4-like **ACL**

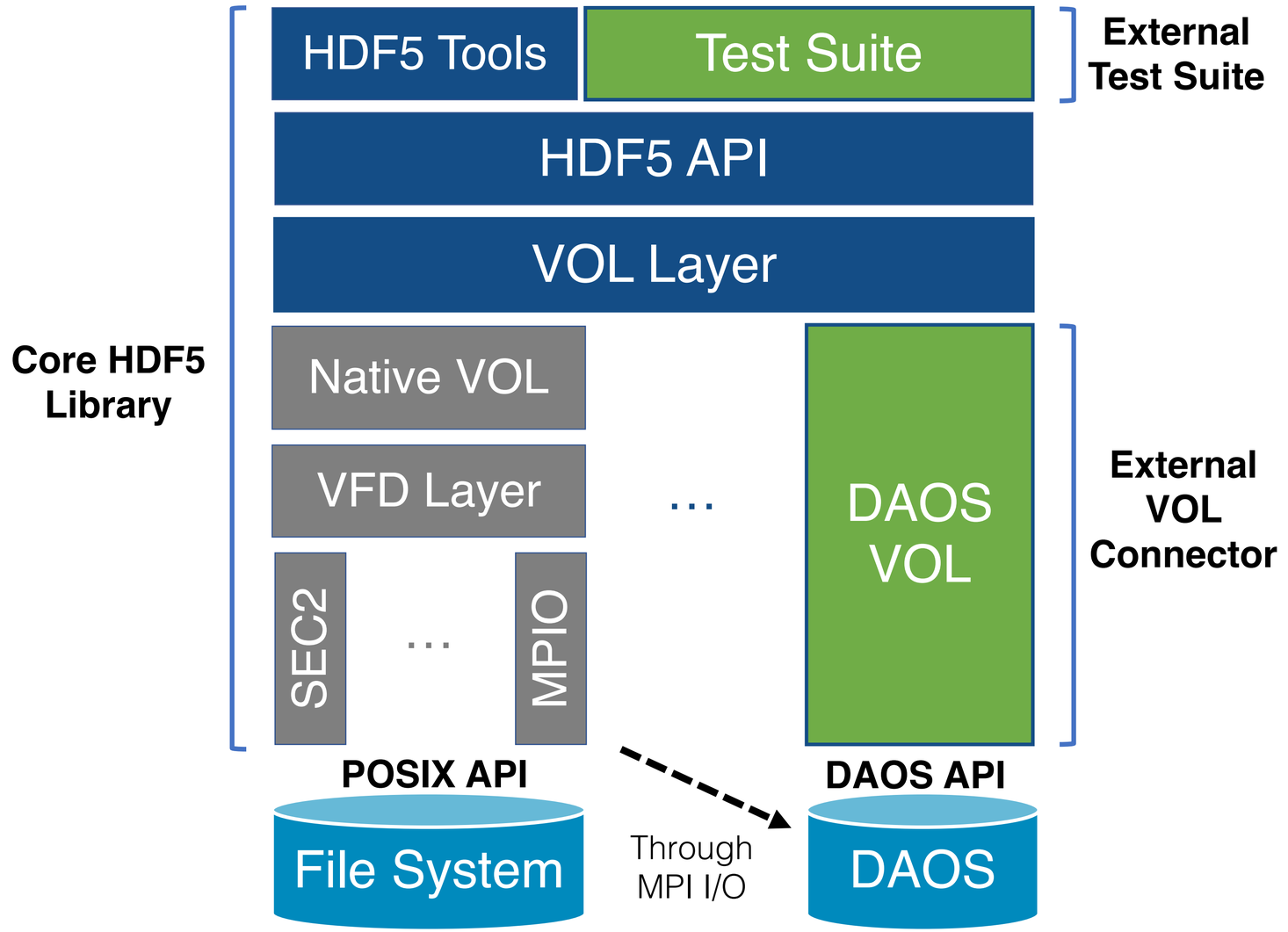
Delivers high-IOPs, high-bandwidth and low-latency storage with advanced features in a single tier

HDF5 VOL Architecture

- New Component
- Enhanced Component
- Native Component

- **Three main components:**
  - HDF5 Library
  - DAOS VOL Connector
  - (External) HDF5 Test Suite

- **Tools support:**
  - h5dump, h5ls, h5diff, h5repack, h5copy, etc



HDF5 DAOS VOL Connector

- **Allows the creation and use of HDF5 files in DAOS**
  - Minimal or no code changes for application developer (if only looking for compatibility)
  - Two ways to tell which connector to use
    - HDF5 file access property list (**recommended for new files or when manipulating multiple VOLs**)

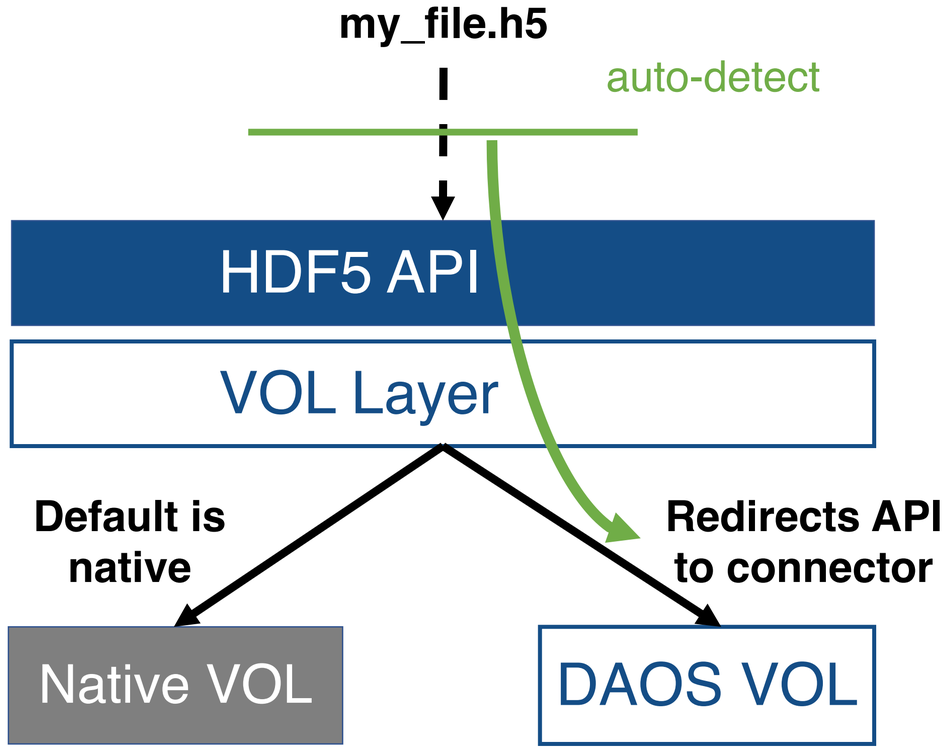
      ```
      H5Pset_fapl_daos()
      ```

    - Environment variable

      ```
      HDF5_VOL_CONNECTOR=daos
      ```

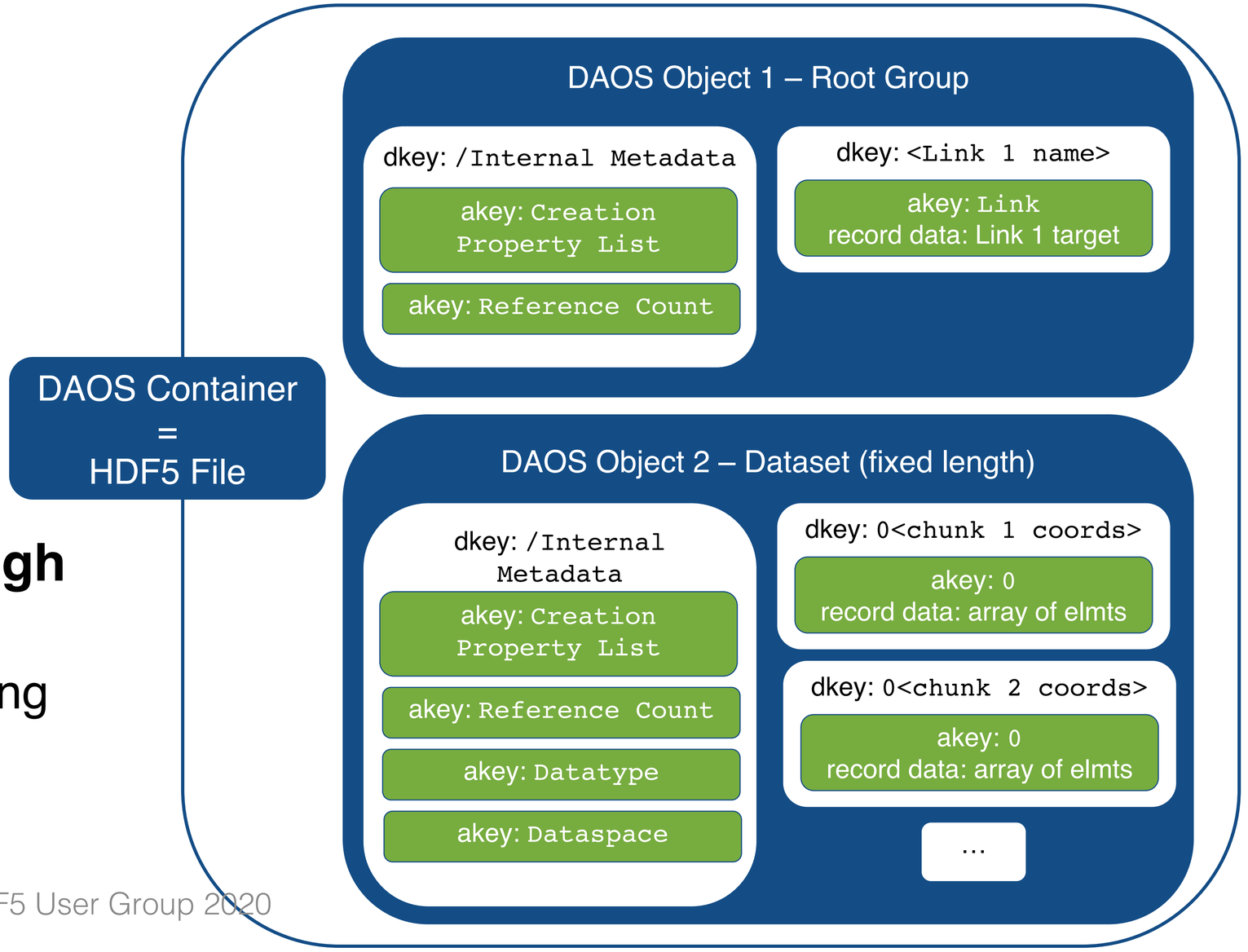
      ```
      HDF5_PLUGIN_PATH=/path/to/connector/folder
      ```

  - **Unified Namespace** component facilitates opening of DAOS files with the DAOS connector (embedded DAOS metadata, DAOS pool UUID, etc)



HDF5 DAOS VOL “file” format

- HDF5 File (Container)
  - Root Group
  - Dataset



- **Independent I/O through DAOS (K/V objects)**
  - Parallel I/O and chunking now become first-class citizens

Data placement and Replication

- **Multiple options**
  - Chunking enabled by default for contiguous datasets, controlled with:

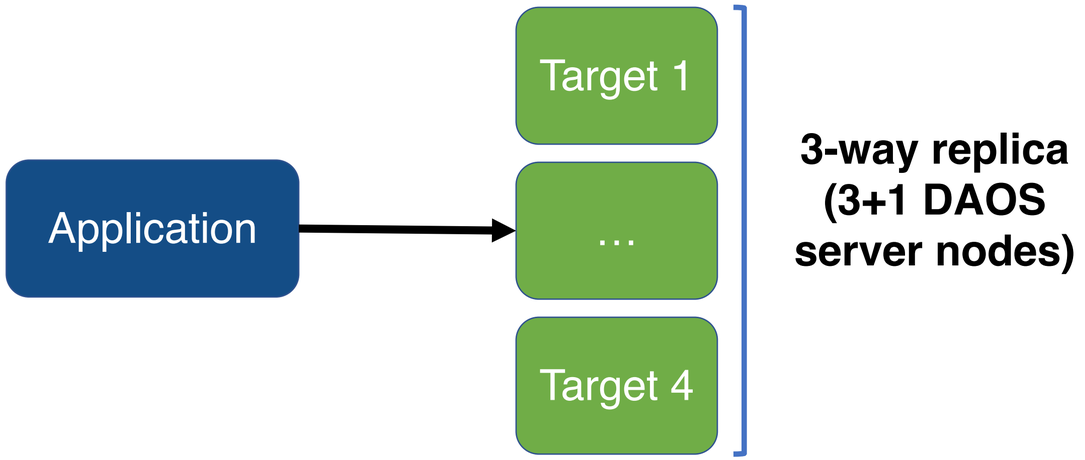
    `H5Pset_chunk()`
  - Set DAOS object class per DAOS object to control number of targets used for storing object (= **stripe count**):

    `H5daos_set_object_class()`

    default uses all targets available
  - Set property to control numbers of **replicas** (for recovery), also controlled through:

    `H5daos_set_object_class()`

    default is no replica



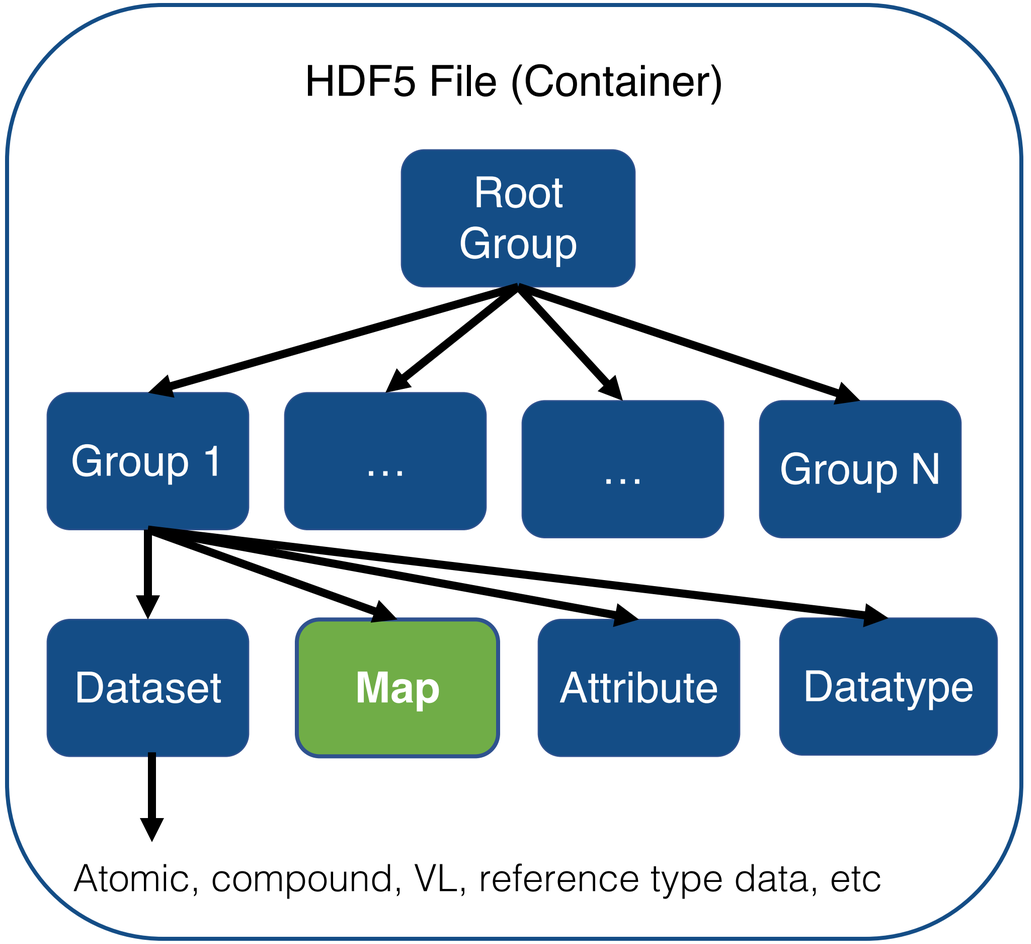
target ≠ storage node:
multiple storage targets per node

HDF5 DAOS VOL Connector

- **All HDF5 features are currently supported except features specific to the native file format**
- **Additional features implemented**
  - Map objects (enabled by K/V objects)
  - File deletion
  - Independent metadata
    - HDF5 objects can be created independently
    - Currently enabled with:

```
H5daos_set_all_ind_metadata_ops()
```

    - **Will be default behavior in the future**
  - Asynchronous I/O



Asynchronous I/O

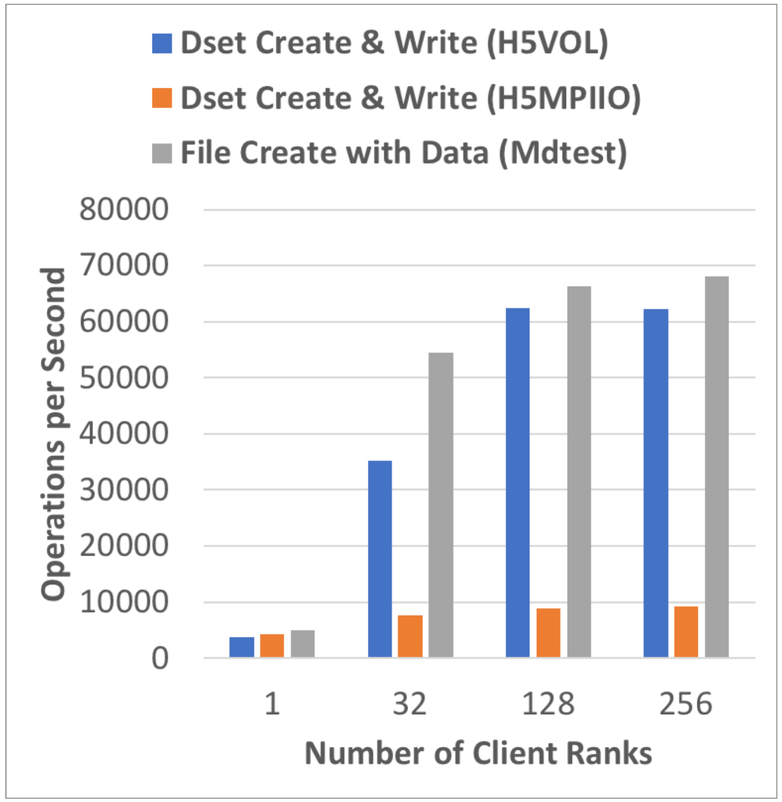
- **Enables asynchrony using Event Set API (see async I/O presentation)**
  - Implemented at DAOS connector level
  - Uses DAOS task engine (not necessarily need additional progress thread)
  - HDF5 API returns before operation completes, places operation in an “event set”

- **Asynchrony must be explicitly controlled by application**
  - Similar to existing async APIs, such as MPI non-blocking
  - Place async tasks in an Event Set (H5ES)
  - Use async versions of all routines that may block
  - Applications are expected to rework/optimize their code to avoid memcpy and do correct error handling

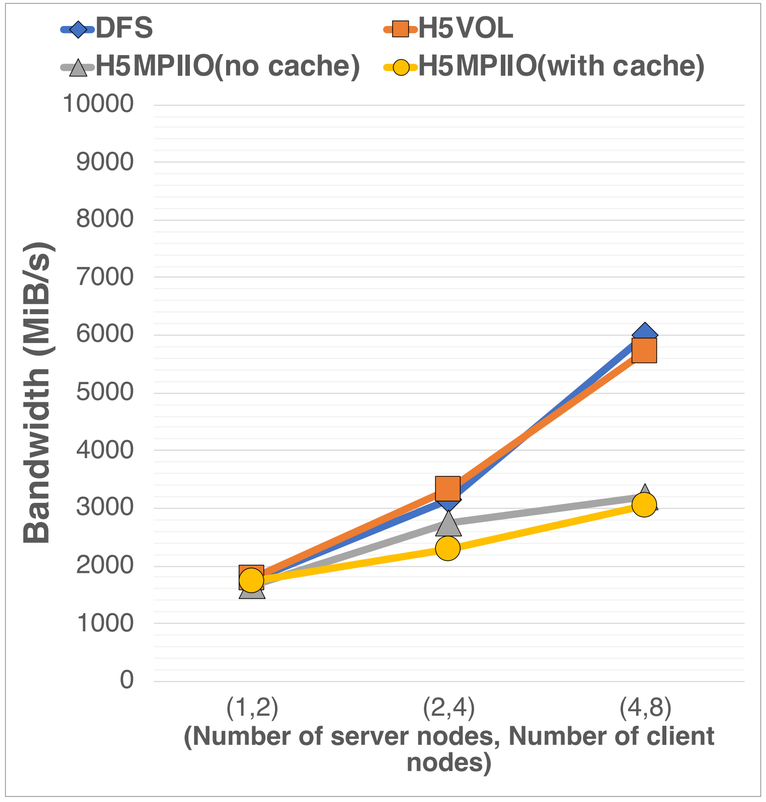
Early performance results

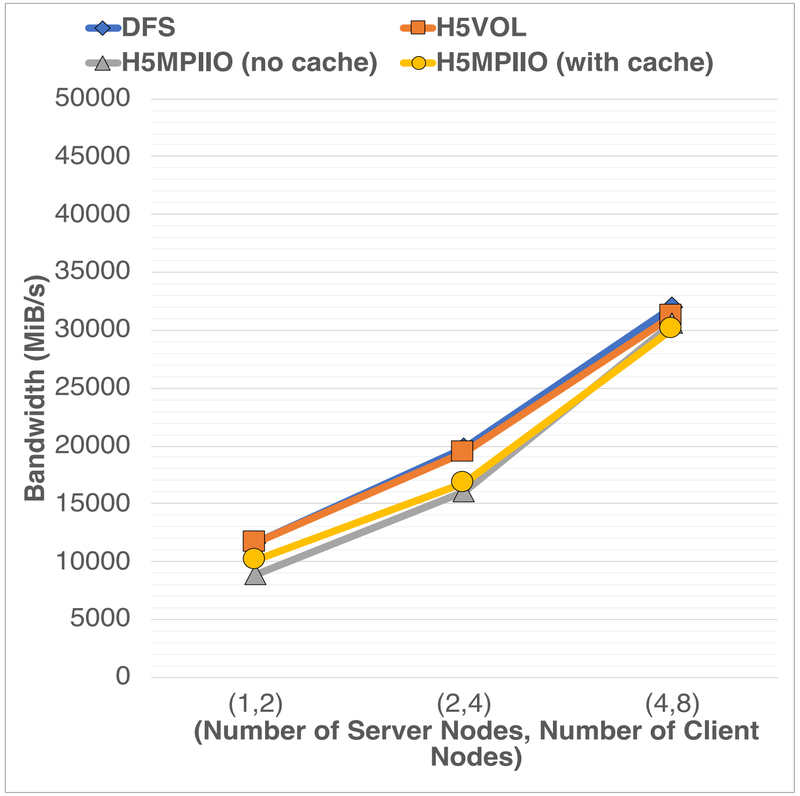
Paying more attention to performance of smaller I/O

Dataset creation comparison with mdtest (DAOS backend)



Dataset write comparison with IOR (16 KB, 1MB respectively, transfer size = chunk size)





Current Status

- **Application porting work**
  - QMCPACK: open-source Quantum Monte Carlo (QMC) simulation code for electronic structure calculations of molecular, quasi-2D and solid-state systems.
  - E3-SM: Earth system model development and simulation project
    - Scorpio: A high-level Parallel I/O Library for structured grid applications
    - NetCDF: software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.
  - VPIC: Vector Particle-In-Cell (VPIC) Project
- **Tuning work**
- **Async I/O work**
- **Release scheduled for end of November**

Repositories

- **HDF5 base repository:**
  - https://github.com/HDFGroup/hdf5
  - Will require at minimum HDF5 1.12.1
- **DAOS VOL Connector repository:**
  - https://github.com/HDFGroup/vol-daos
- **External HDF5 test suite repository:**
  - https://github.com/HDFGroup/vol-tests
- **Also available through Spack repository (initial support)**
  - https://github.com/HDFGroup/spack_daos

Acknowledgments / Questions

- **Intel**
- **Argonne National Laboratory**