Efficient Parallel I/O with HDF5 and Proactive Data Containers (PDC)

Suren Byna

Staff Scientist

Scientific Data Management Group

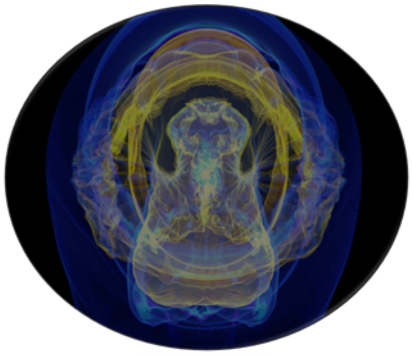
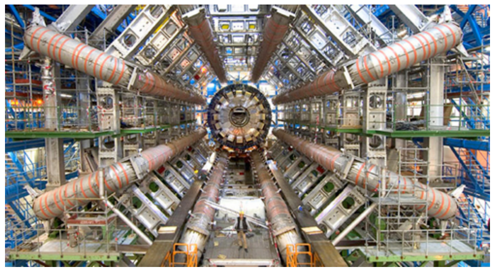
Lawrence Berkeley National Laboratory

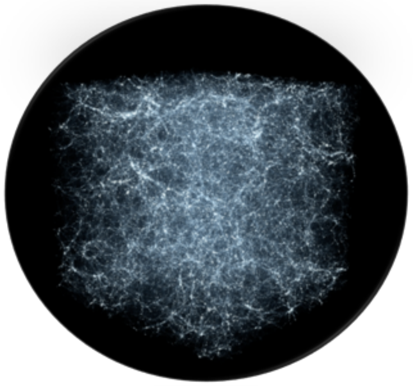
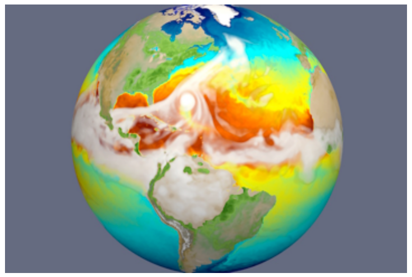
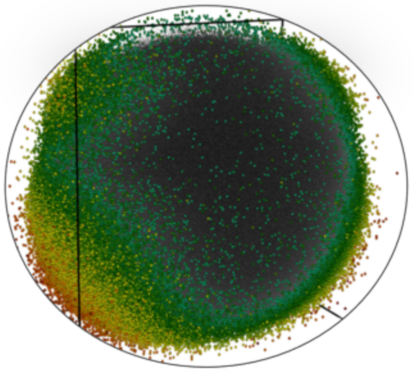
Teams contributed to these slides

- ExaHDF5: Suren Byna, Quincey Koziol, Houjun Tang, Bin Dong, Junmin Gu, Jialin Liu, Alex Sim, Tonglin Li, Teng Wang, Venkat Vishwanath, Rick Zamora, Paul Coffman, Todd Munson, Scot Breitenfeld, John Mainzer, Frank Willmore, Dana Robinson, Jerome Soumagne, Richard Warren, Neelam Bagha, Elena Pourmal,
- EOD-HDF5: Quincey Koziol, Suren Byna, Houjun Tang, Tonglin Li, John Mainzer, Scot Breitenfeld, Gerd Heber, Elena Pourmal, Wei Zhang, Yong Chen
- Proactive Data Containers: Suren Byna, Quincey Koziol, Houjun Tang, Teng Wang, Bin Dong, Jialin Liu, Jerome Soumagne, Kimmy Mu, Richard Warren, Venkat Vishwanath, François Tessier

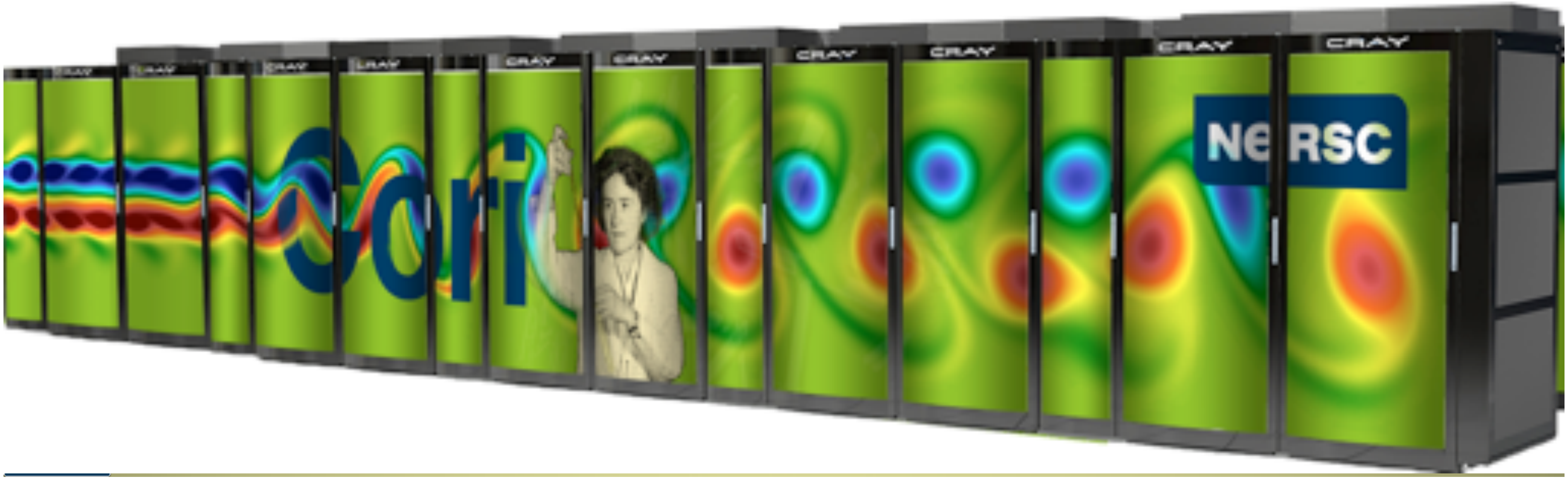
Scientific Data - Where is it coming from?

- Simulations
- Experiments
- Observations

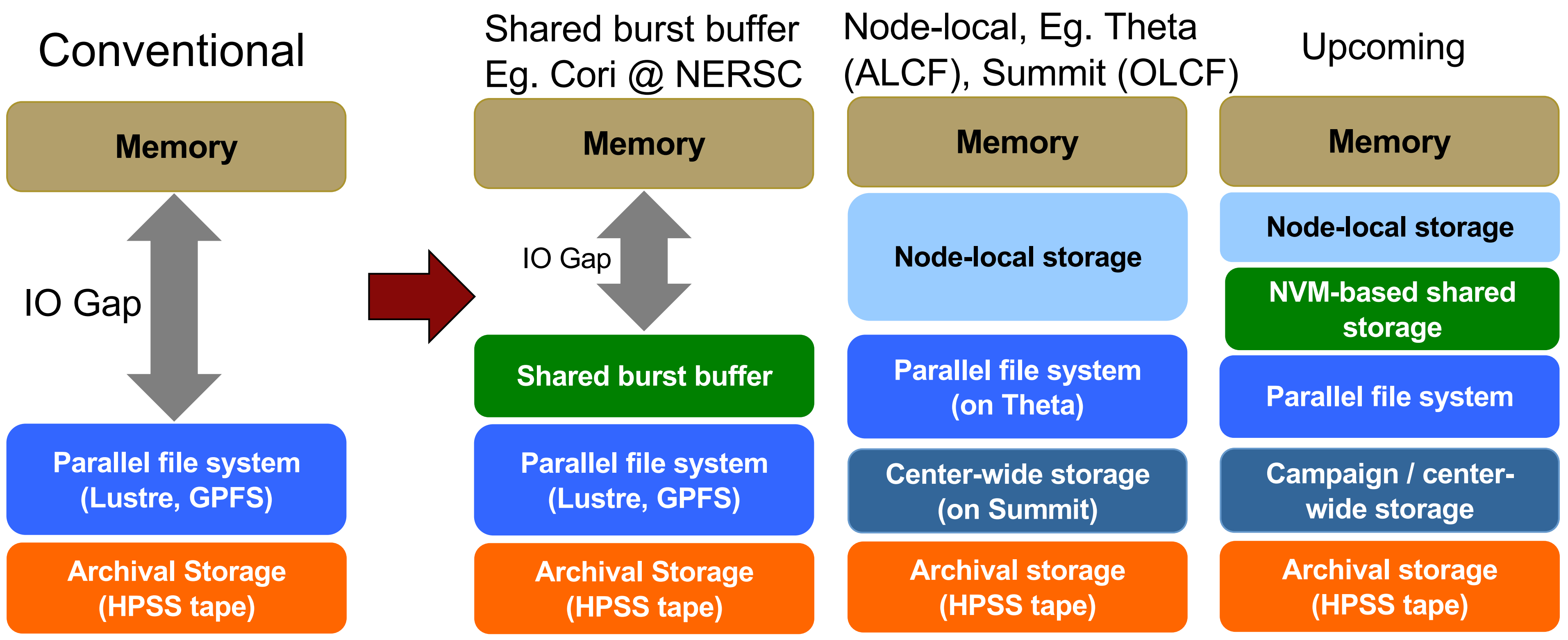



Supercomputing systems

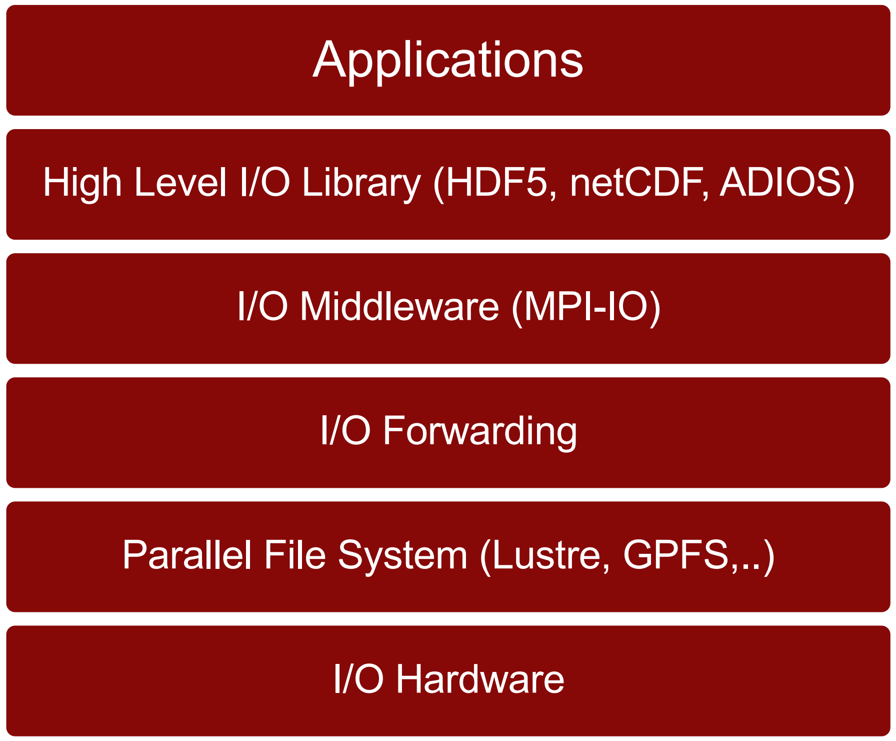


Trends – Storage system transformation


Conventional
Memory
IO Gap
Parallel file system (Lustre, GPFS)
Archival Storage (HPSS tape)
Shared burst buffer Eg. Cori @ NERSC
Memory
IO Gap
Shared burst buffer
Parallel file system (Lustre, GPFS)
Archival Storage (HPSS tape)
Node-local, Eg. Theta (ALCF), Summit (OLCF)
Memory
Node-local storage
Parallel file system (on Theta)
Center-wide storage (on Summit)
Archival storage (HPSS tape)
Upcoming
Memory
Node-local storage
NVM-based shared storage
Parallel file system
Campaign / center-wide storage
Archival storage (HPSS tape)

- IO performance gap in HPC storage is a significant bottleneck because of slow disk-based storage
- SSD and new memory technologies are trying to fill the gap, but increase the depth of storage hierarchy

Parallel I/O software stack

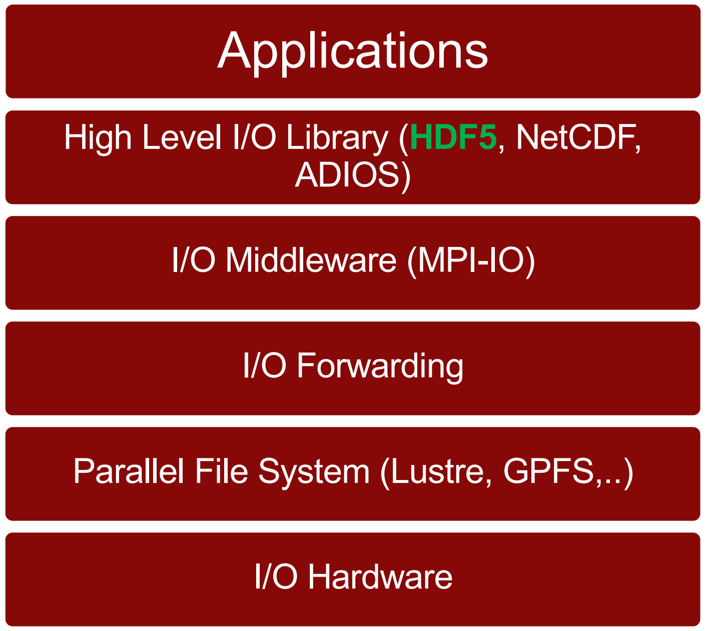

Applications
High Level I/O Library (HDF5, netCDF, ADIOS)
I/O Middleware (MPI-IO)
I/O Forwarding
Parallel File System (Lustre, GPFS,..)
I/O Hardware

- I/O Libraries
  - HDF5, ADIOS, PnetCDF, NetCDF-4, …
- Middleware – POSIX-IO, MPI-IO
- I/O Forwarding
- File systems: Lustre, GPFS / Spectrum Scale, DataWarp, …
- I/O Hardware (disk-based, SSD-based, …)

Focus of this presentation

- Storing and retrieving data – Parallel I/O and HDF5
  - Brief intro to HDF5
  - New features in HDF5 (funded by ECP and ASCR)

- Autonomous data management system
  - Proactive Data Containers (PDC) system
  - Metadata management service
  - Data management service

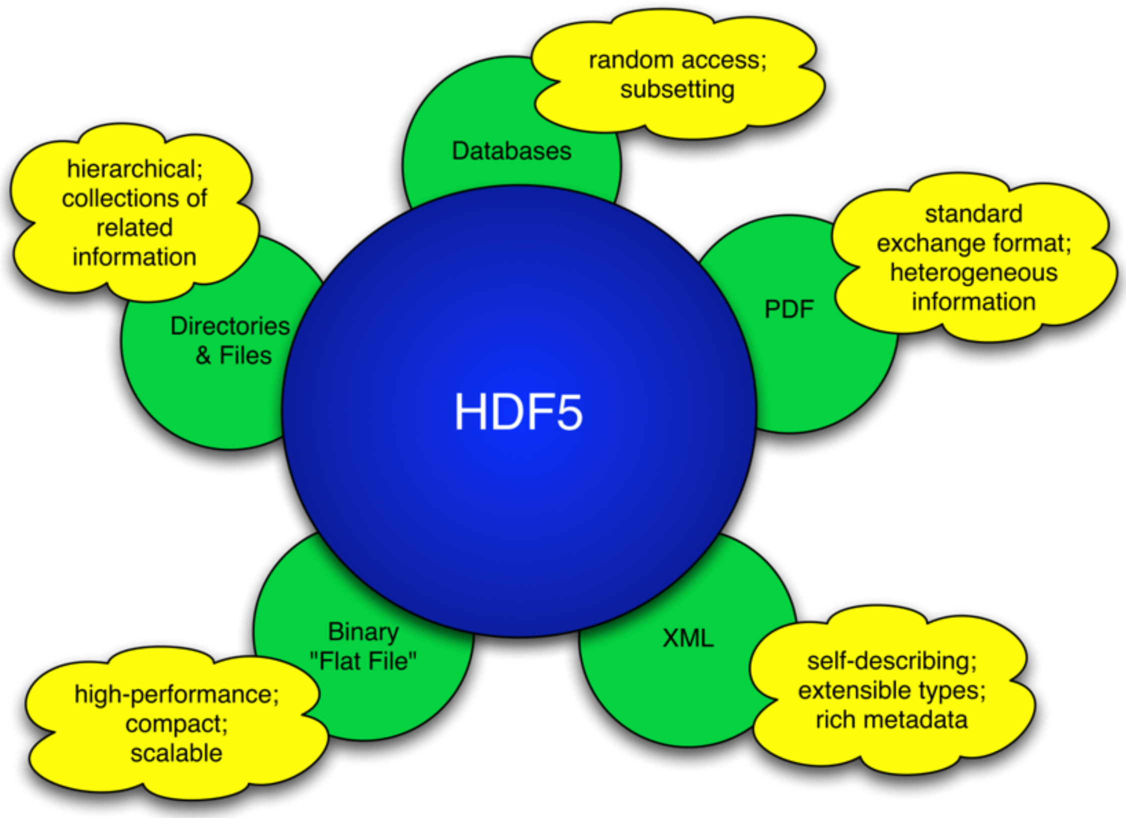

Applications
High Level I/O Library (HDF5, NetCDF, ADIOS)
I/O Middleware (MPI-IO)
I/O Forwarding
Parallel File System (Lustre, GPFS,..)
I/O Hardware

WHAT IS HDF5?

What is HDF5?

- HDF5 ➔ Hierarchical Data Format, v5
- Open **file format**
  - Designed for high volume and complex data
- Open source **software**
  - Works with data in the format
- An extensible **data model**
  - Structures for data organization and specification

HDF5 is like …

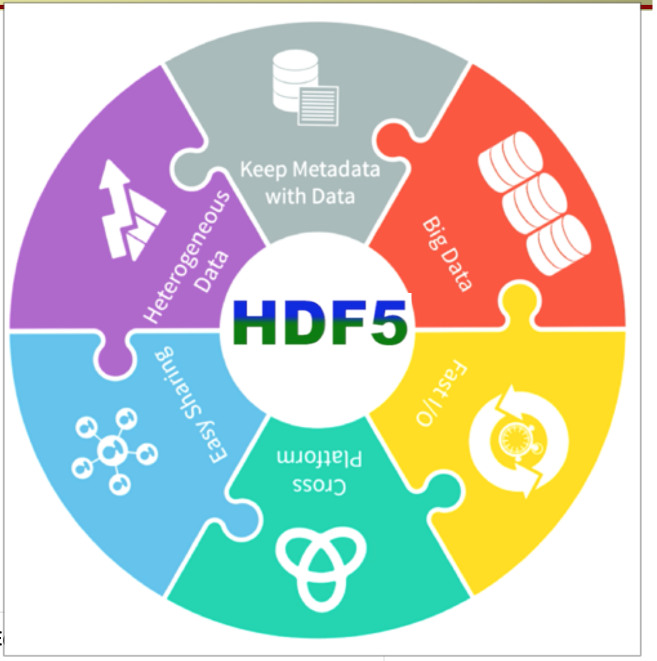

random access; subsetting
Databases
hierarchical; collections of related information
Directories & Files
HDF5
PDF
standard exchange format; heterogeneous information
Binary "Flat File"
high-performance; compact; scalable
XML
self-describing; extensible types; rich metadata

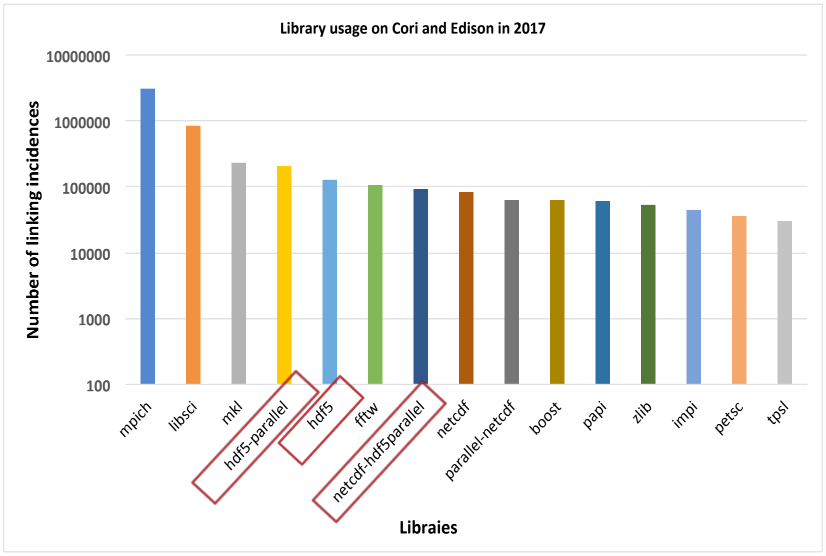
HDF5 Overview

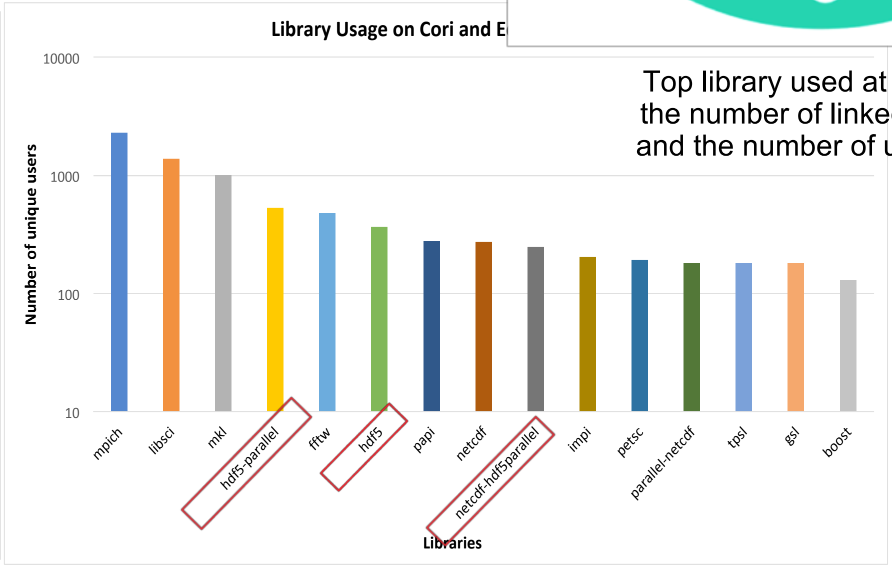
- HDF5 is designed to organize, store, discover, access, analyze, share, and preserve diverse, complex data in continuously evolving heterogeneous computing and storage environments.
- First released in 1998, maintained by **The HDF Group**

"De-facto standard for scientific computing" and integrated into every major scientific analytics + visualization tool

- Heavily used on DOE supercomputing systems

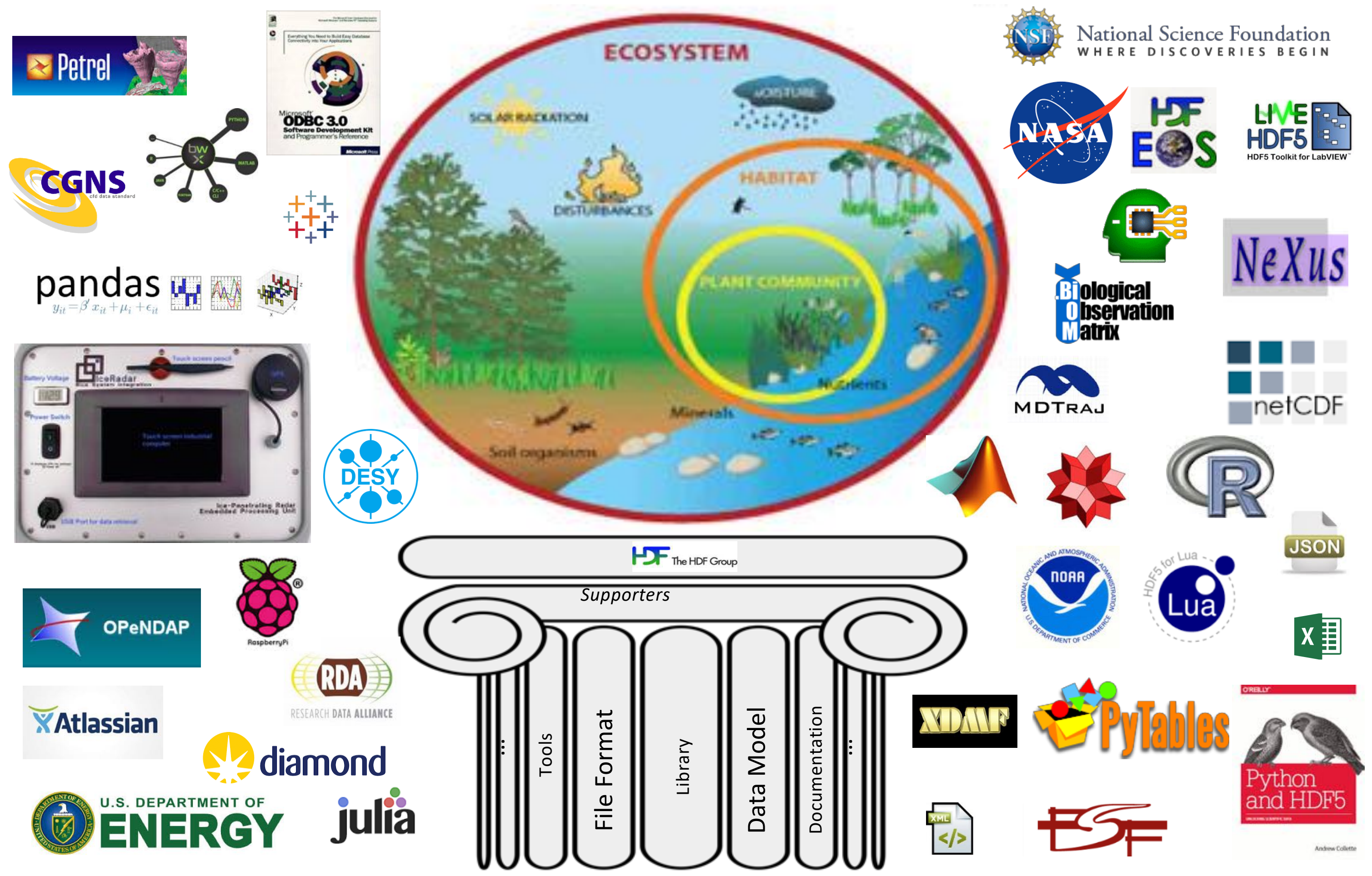






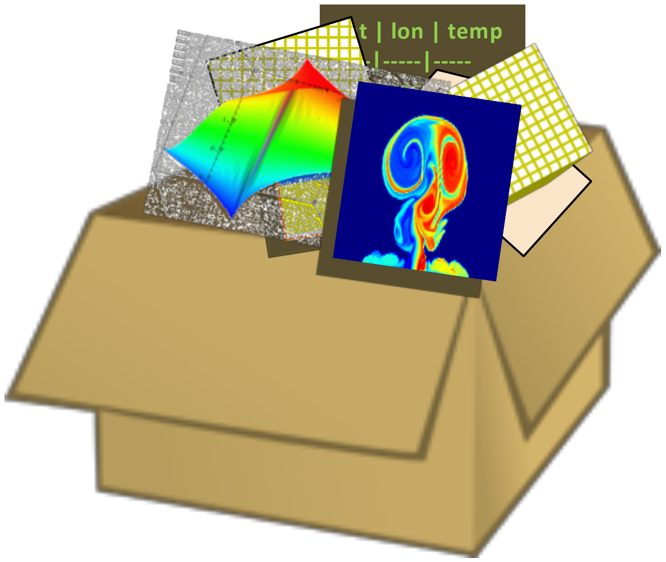
Top library used at NERSC by the number of linked instances and the number of unique users

HDF5 Ecosystem



HDF5 File

An HDF5 file is a **container** that holds data objects.



HDF5 Data Model

Dataset

Link

Group

Datatype

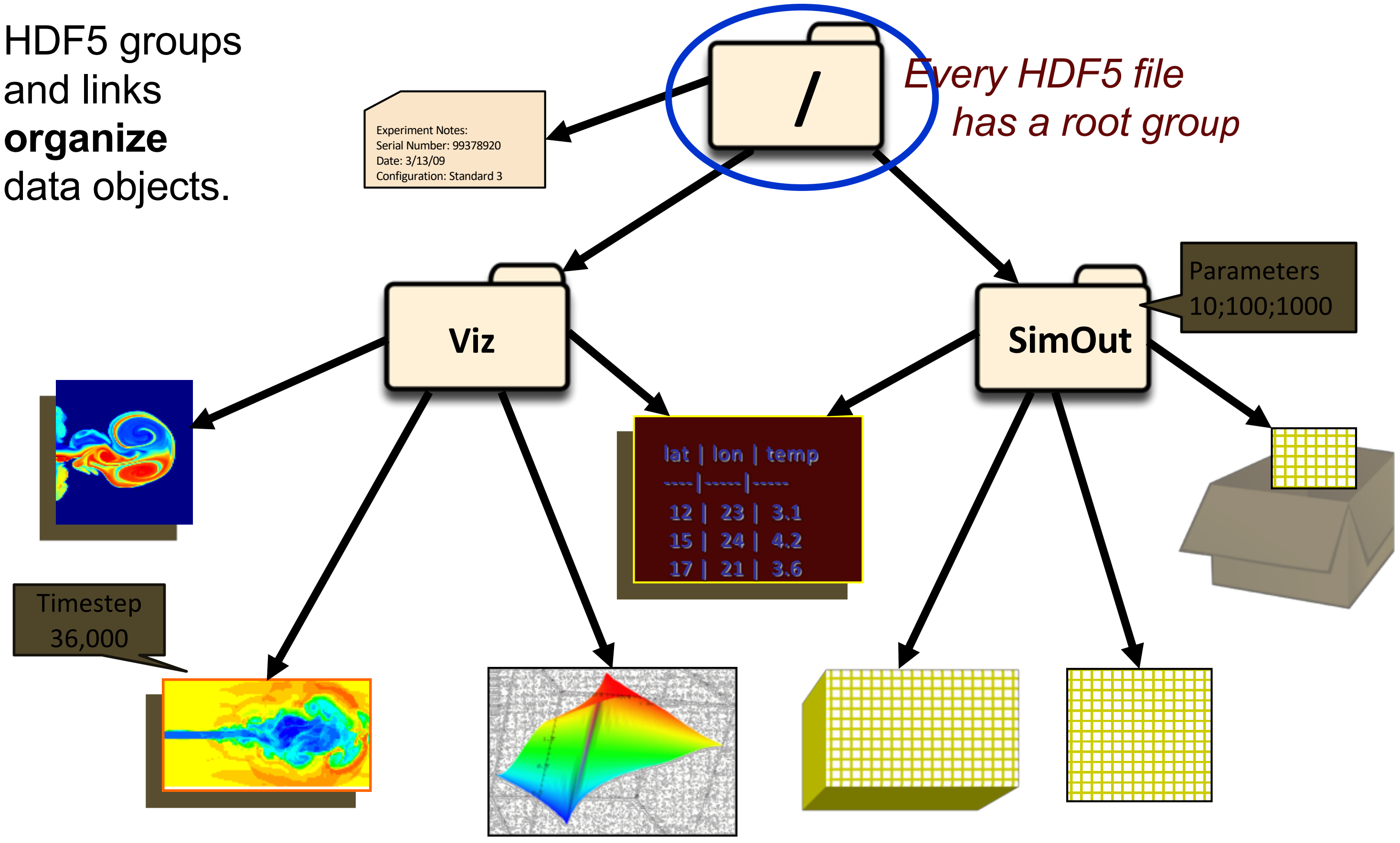
Attribute

Dataspace

File

HDF5 Groups and Links

HDF5 groups and links **organize** data objects.



HDF5 Home Page

HDF5 home page: http://www.hdfgroup.org/solutions/hdf5/

- Latest release: HDF5 1.10.5 (1.12 coming soon)

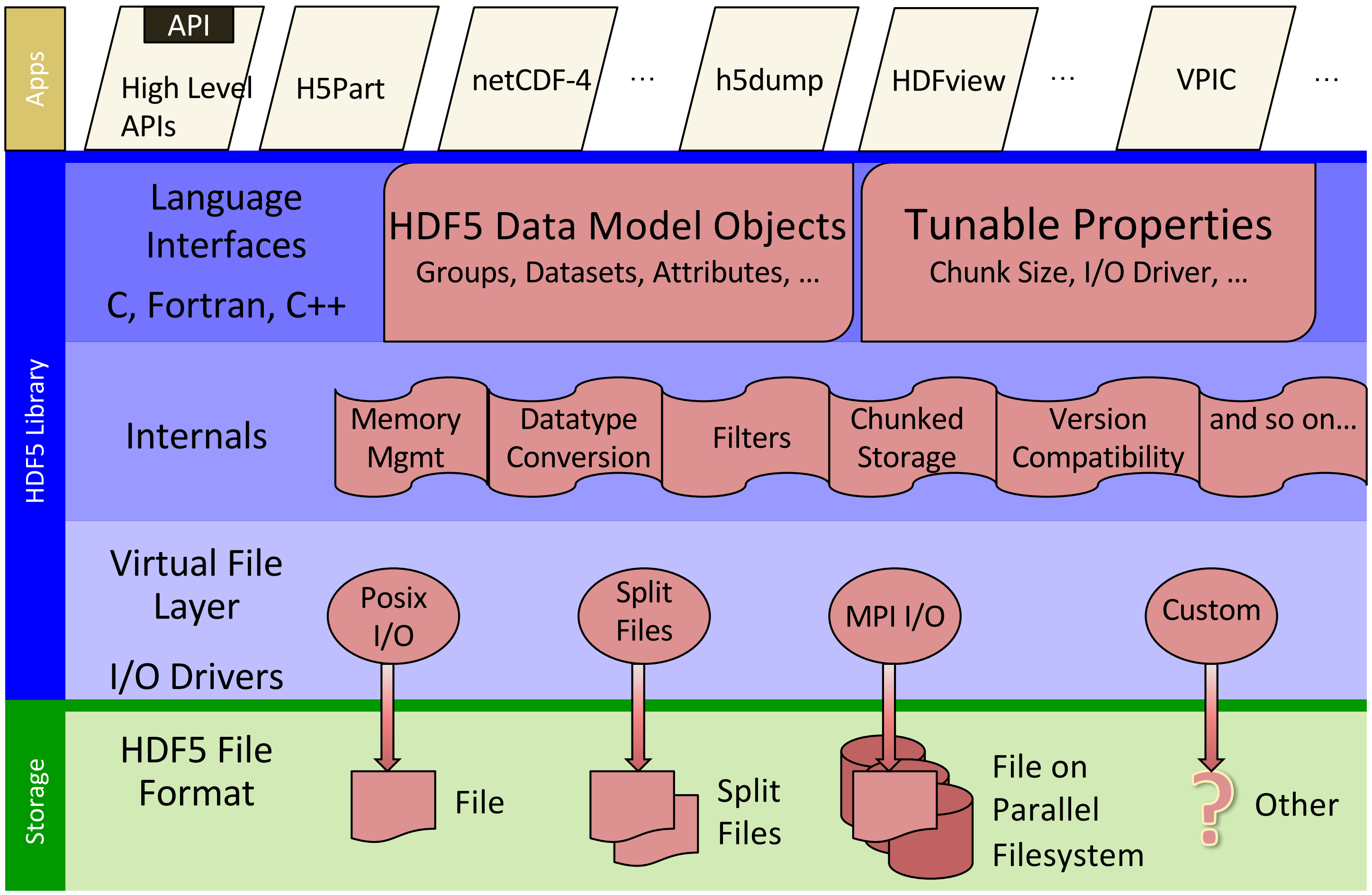
HDF5 source code:

- Written in C, and includes optional C++, Fortran, and Java APIs
  - Along with "High Level" APIs
- Contains command-line utilities (h5dump, h5repack, h5diff, ..) and compile scripts

HDF5 pre-built binaries:

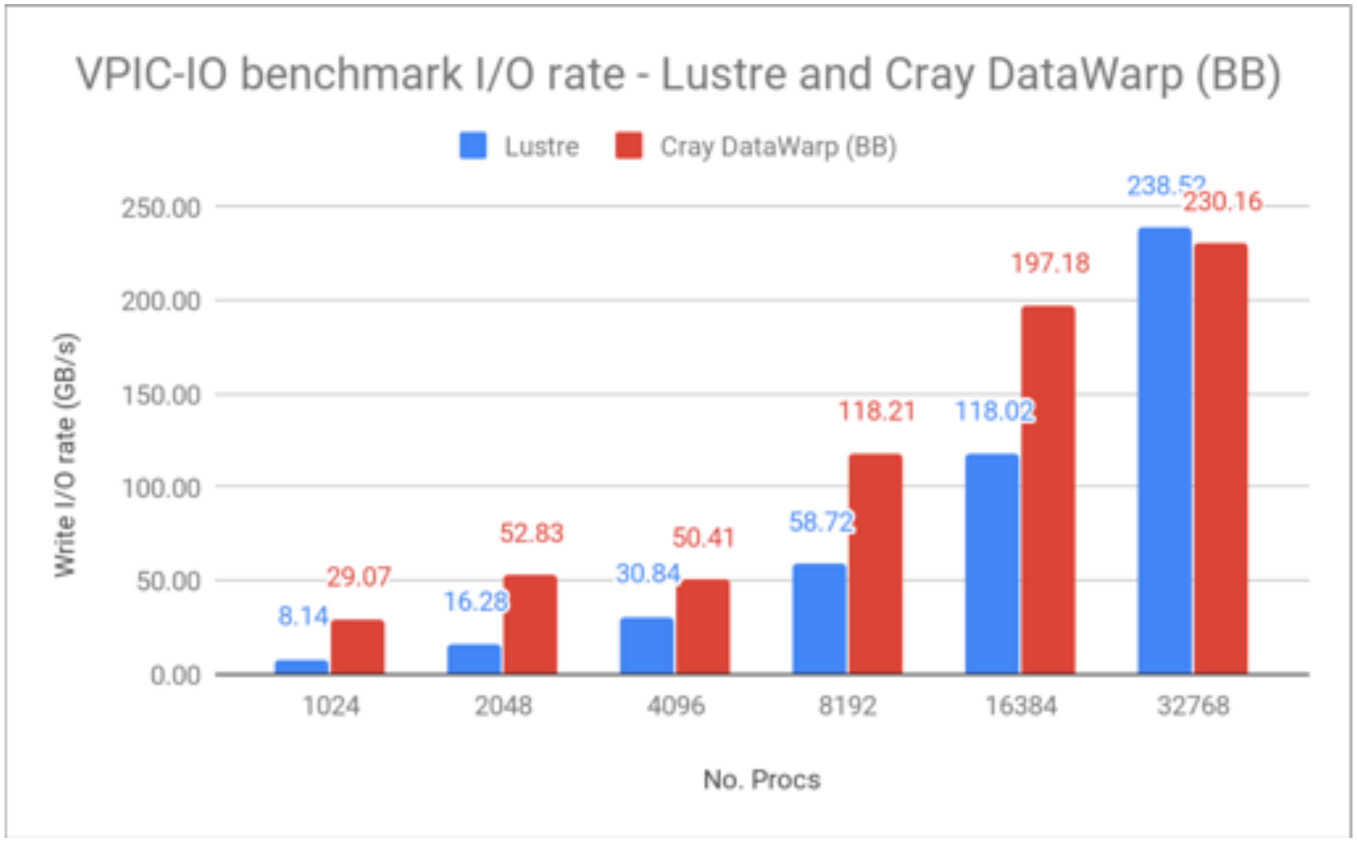
- When possible, include C, C++, Fortran, Java and High Level libraries.
  - Check ./lib/libhdf5.settings file.
- Built with and require the SZIP and ZLIB external libraries

HDF5 Software Layers & Storage



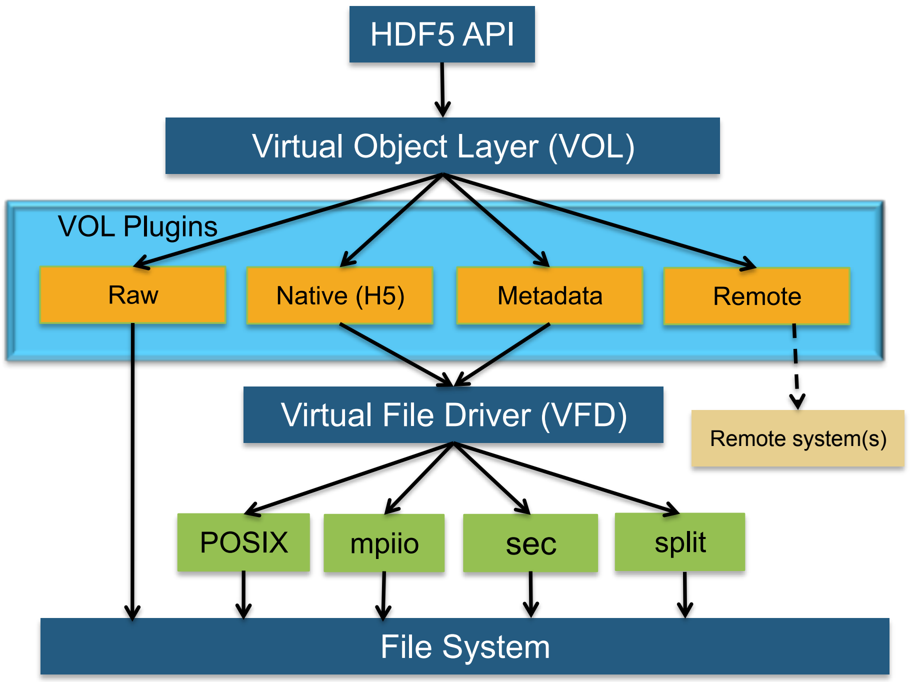
HDF5 performance on supercomputers

- A plasma physics simulation's I/O kernel, using VPIC code
  - I/O kernel with MPI processes, where each process writes 8 variables of 8 M particles



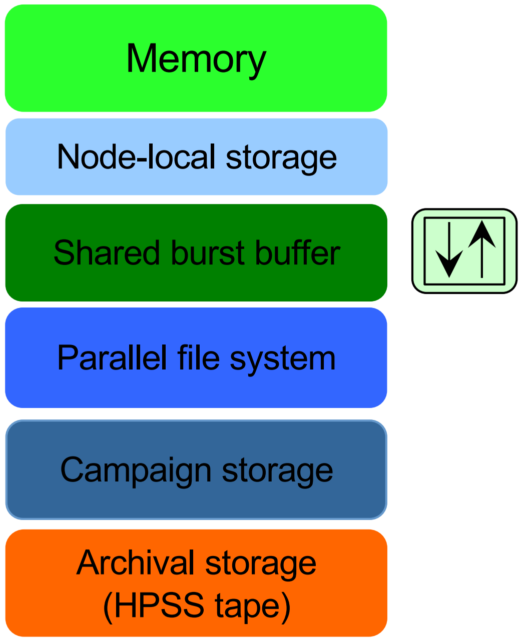
ExaHDF5 features – Virtual Object Layer (VOL)

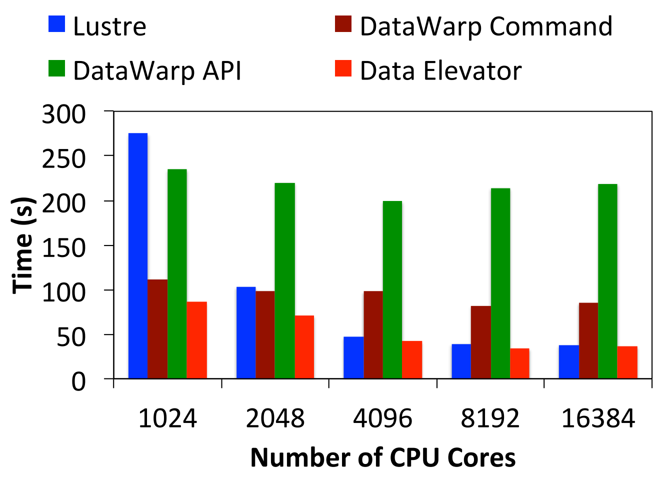
- Provides an application with the HDF5 data model and API, but allows different underlying storage mechanisms
- Enables developers to easily use HDF5 on novel current and future storage systems
  - Plugins for using fast storage layers transparently and for accessing DAOS are available
  - VOL plugins for reading netCDF and ADIOS data are in development
- VOL connectors can be stacked and set with a configuration variable
- Integrated into the HDF5 https://bitbucket.hdfgroup.org/projects/HDFFV/repos/hdf5/
  - Will be released in 1.12.x

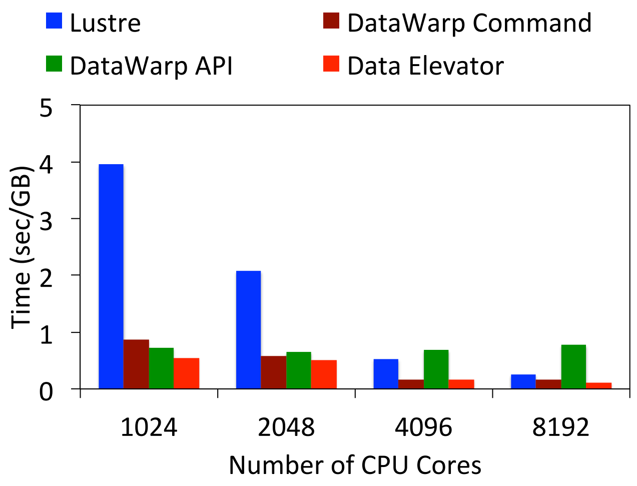


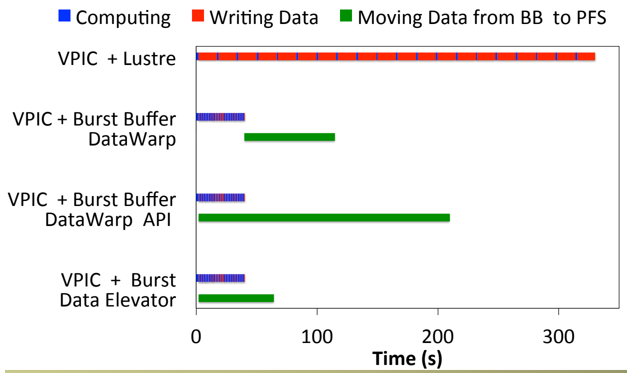
ExaHDF5 features – Data Elevator

- Data Elevator caching VOL
  - Transparent data movement in storage hierarchy
  - In situ data analysis capability using burst buffers
- Tested with a PIC code and Chombo-IO benchmark
- Applications evaluating Data Elevator
  - E3SM-MMF and Sandia ATDM project is evaluating performance
  - Other candidates: EQSim, AMReX co-design center
- Installed on NERSC's Cori system (*module load data-elevator*)







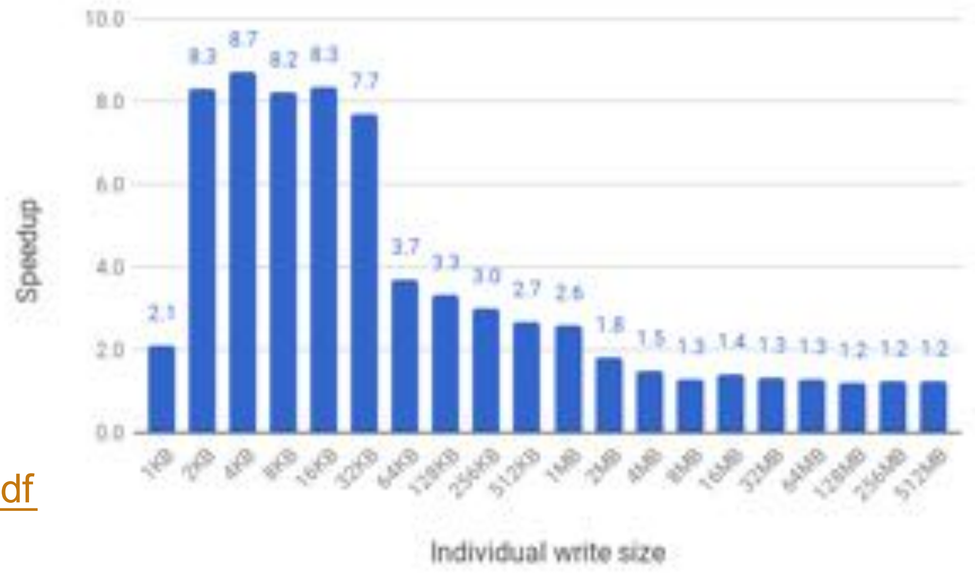


https://bitbucket.org/sbyna/dataelevator/

ExaHDF5 features – Full SWMR

- SWMR enables a single writing process to update an HDF5 file, while multiple reading processes access the file in a concurrent, lock-free manner
- Previously limited to the narrow use-case of appending new elements to HDF5 datasets
- Full SWMR extends existing SWMR to support all metadata operations, such as object creation and deletion, attribute updates

In ECP, ExaFEL project requires this feature. In general, Full SWMR is useful for managing experimental and observational data

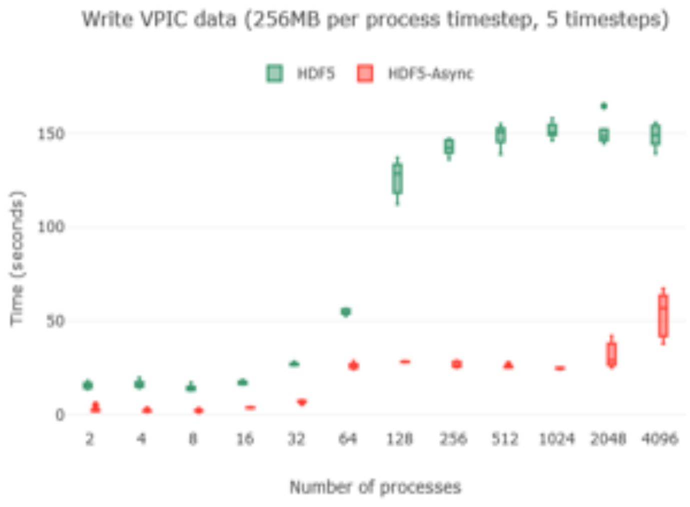


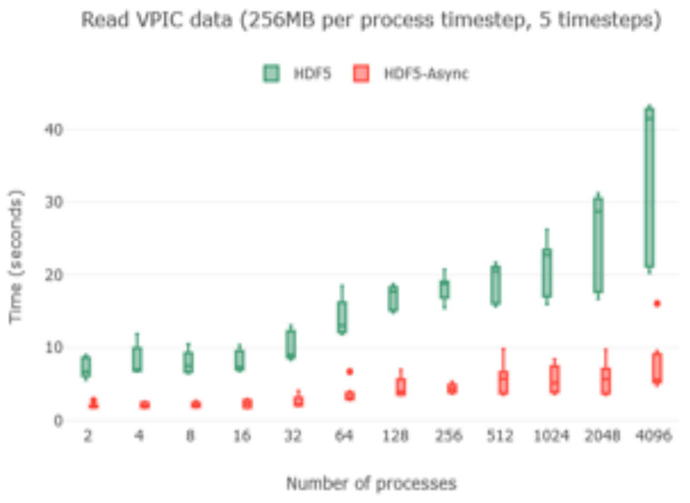
Full SWMR branch of HDF5: https://bitbucket.hdfgroup.org/projects/HDFFV/repos/hdf5/browse?at=refs%2Fheads%2Ffull_swmr

ExaHDF5 features – Async I/O

- Asynchronous I/O allows an application to overlap I/O with other operations
- Implemented the asynchronous I/O feature as a VOL (Virtual Object Layer) connector for HDF5
- Asynchronous task execution, by running those tasks in separate background threads
  - Using Argobots for task execution
  - The thread execution interface has been abstracted to allow to replace Argobots

Async I/O branch of HDF5: https://bitbucket.org/berkeleylab/exahdf5/src/master/vol_plugins/async/



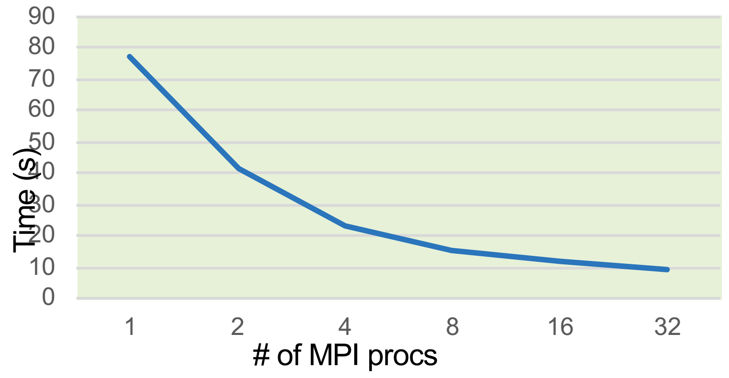


ExaHDF5 features – Parallel querying

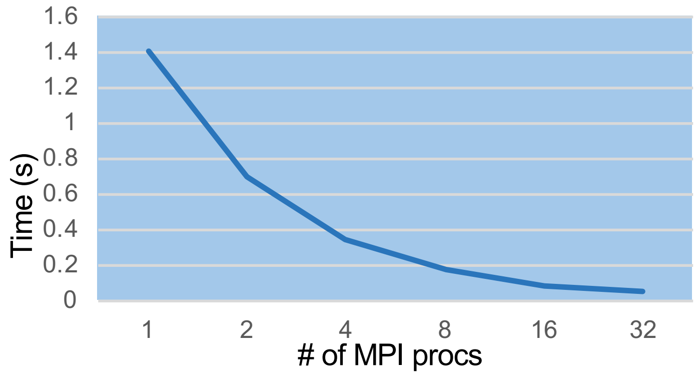
- HDF5 *index* objects and API routines allow the creation of indexes on the contents of HDF5 containers, to improve query performance
- HDF5 *query* objects and API routines enable the construction of query requests for execution on HDF5 containers
  - H5Qcreate
  - H5Qcombine
  - H5Qapply
  - H5Qclose

- HDF5 Bitbucket repo containing the “topic-parallel-indexing” source code: https://bitbucket.hdfgroup.org/projects/HDFFV/repos/hdf5

Build Index (seconds)



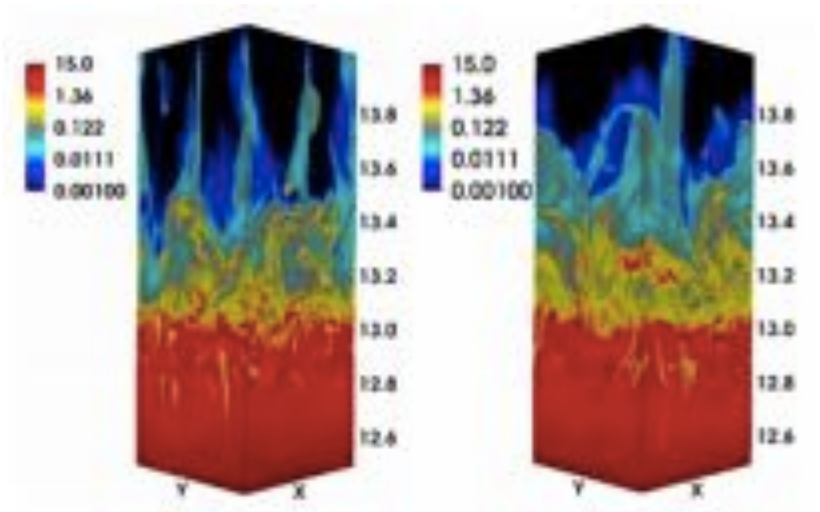
Evaluate Query (seconds)



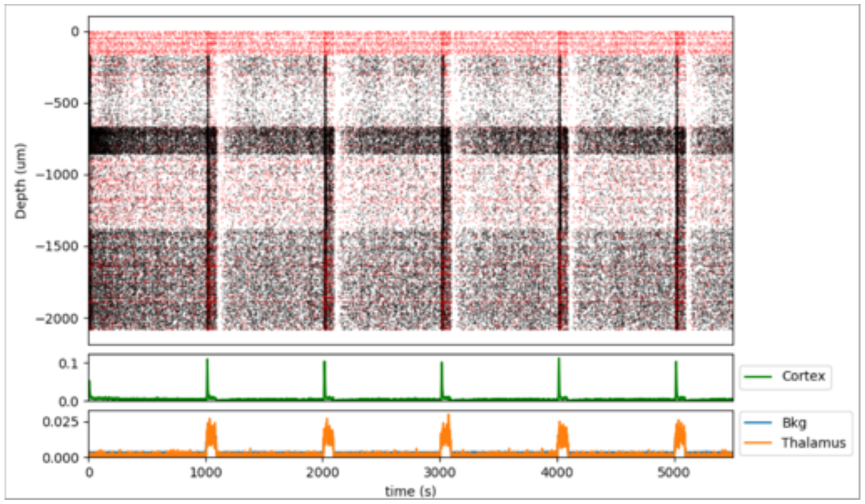
- Parallel scaling of index generation and query resolution is observed even for small-scale experiments:

HDF5 performance tuning – Athena

- Athena astrophysics code experienced poor performance



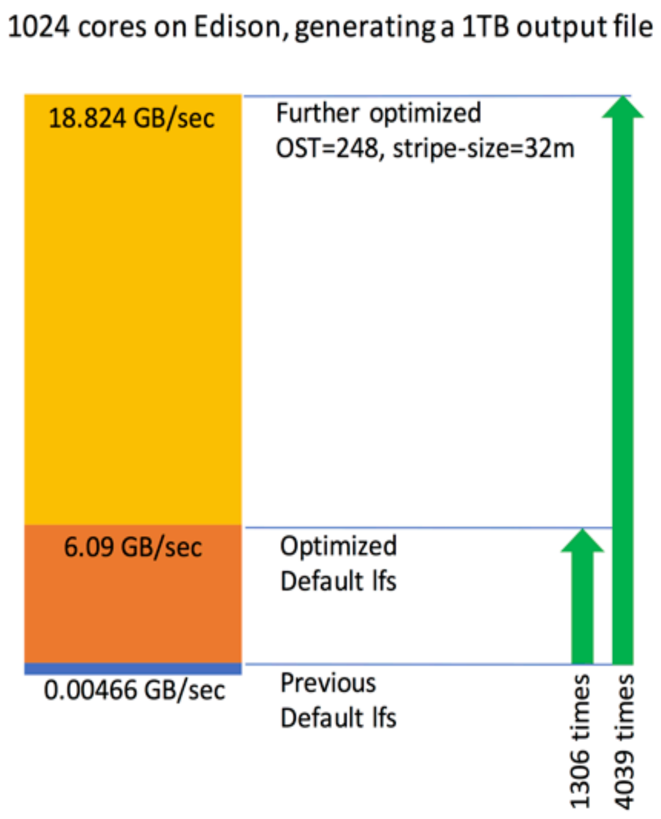
Athena astrophysics code
40% of execution time in I/O, using HDF5 profiling tools identified a large number of concurrent writes; with collective I/O, reduced I/O portion to less than 1% of the execution time.



Neurological Disorder I/O Pipeline
Identified that h5py interface was prefilling HDF5 dataset buffers unnecessarily and avoiding that improved performance by 20X (from 40 min to 2 min)

HDF5 performance tuning – Accelerator physics

- Accelerator physics simulation code



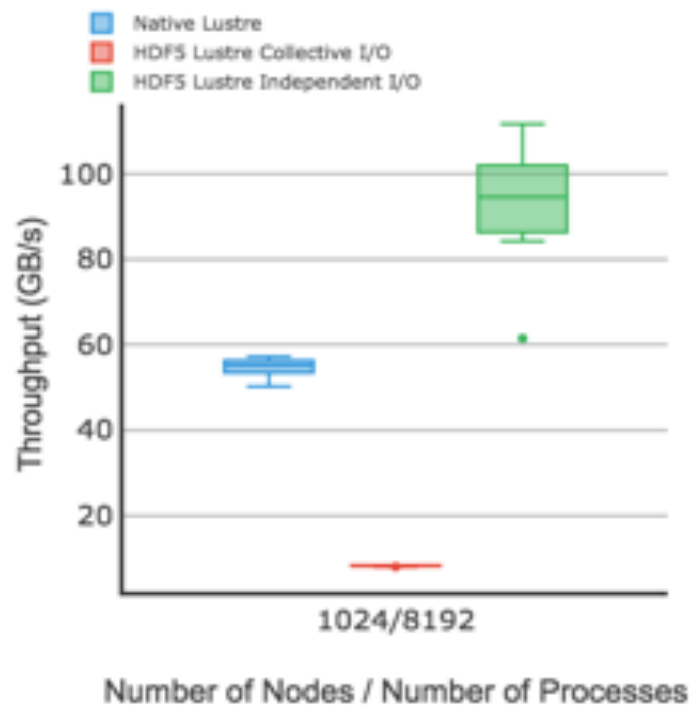
h5py bug fix
+
Lustre tuning

Lustre tuning

Default

WarpX-IO

HDF5 performance tuning – AMReX I/O



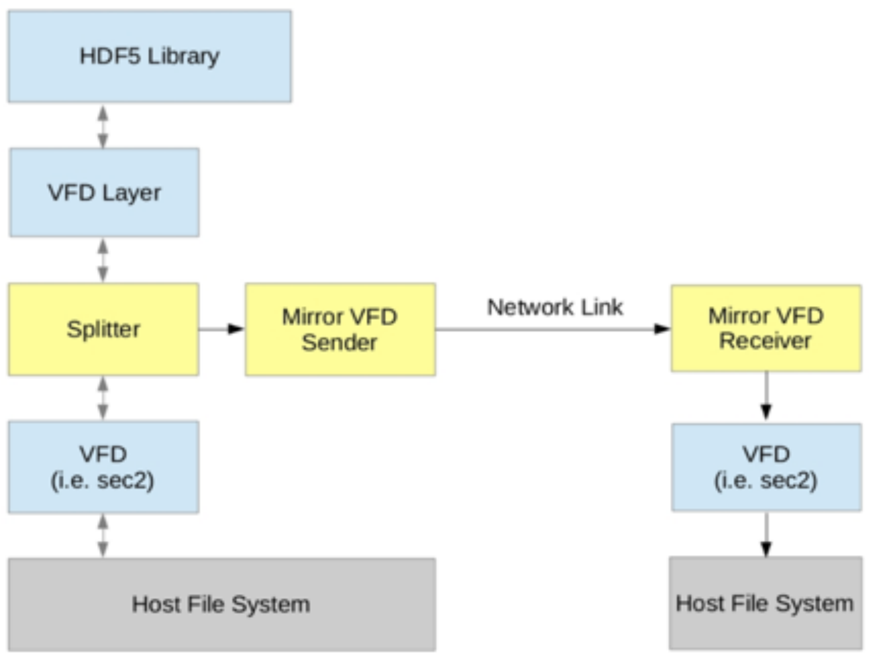
AMReX I/O Benchmark initial tuning

Experimental and Observational Data (EOD) management requirements

- Requirements of LCLS data
  - Handle multiple producers and multiple consumers of data
  - Support remote streaming synchronization
  - Support for variable length modes of data
- Requirements of NCEM data
  - Support for sparse data management
  - Support for remote streaming synchronization
  - Multiple producers and multiple consumers of data -- has interest
- Requirements of ALS data
  - Extend data models to support changes in data and data schemas
  - Metadata and provenance management for multiple HDF5 containers

Remote streaming synchronization

- Remote streaming synchronization
  - Use case: LCLS-2 at SLAC will need to write HDF5 files locally and simultaneously duplicate the files on Cori at NERSC
  - Evaluated various design options
    - Virtual Object Layer (VOL), Virtual File Driver (VOL), HDF server, rsync file system, and snapshot + transmit
  - Splitter and Mirror Virtual File Driver (Mirror VFD) development in progress



Multiple writers and multiple readers (MWMR)

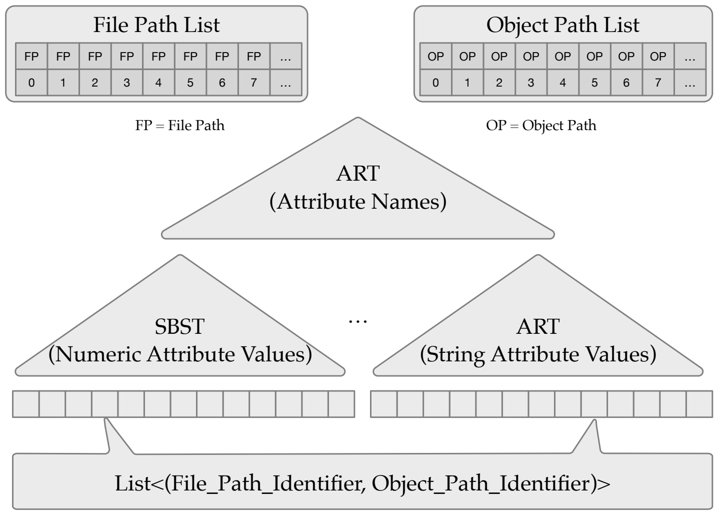
- Multiple producers and multiple consumers
    - Use case: Multiple producers of LCLS-2 data at SLAC need to write to the same HDF5 file, while the same file is being analyzed by multiple readers
    - Full Single writer and multiple readers (SWMR) implementation is in progress
    - Exploring consistency and coherence models of MWMR in existing tools, such as TileDB
        - Works only on a single node

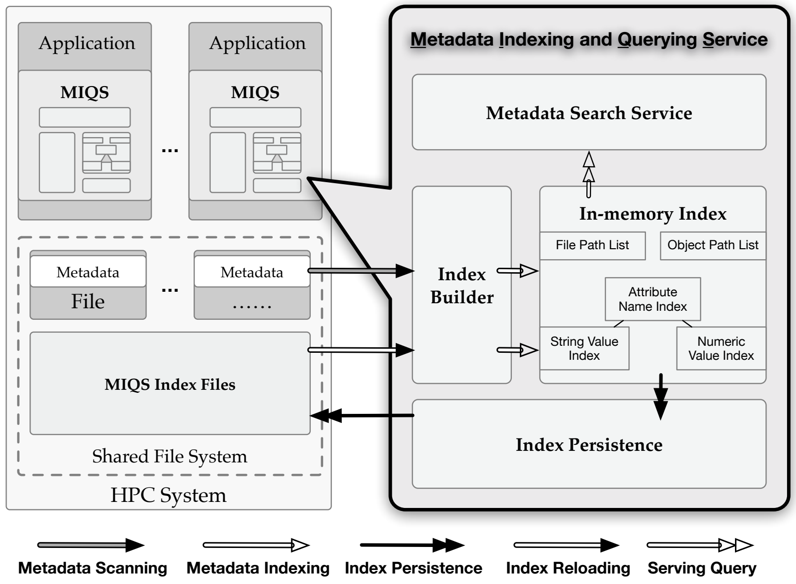
Variable length arrays and sparse data management

- Evaluating performance of variable-length arrays and sparse datasets in current HDF5 implementation
- Ongoing activity
  - Identify performance bottlenecks from an LCLS use case
  - Devise optimization strategies for writing and reading variable length arrays
  - Implement in HDF5 and evaluate

Metadata indexing and querying for HDF5 files

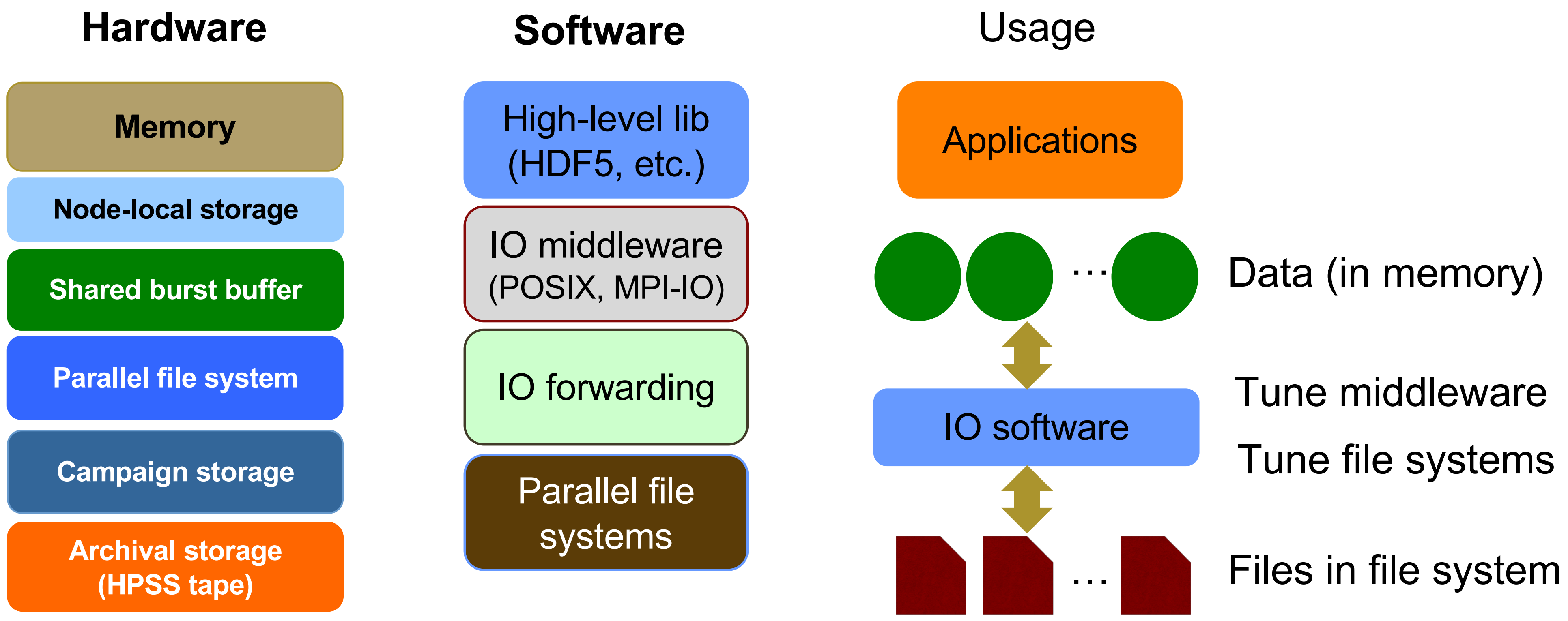
- Developed indexing structures for searching metadata
  - Existing metadata search rely on relational databases (PostgreSQL) and NoSQL databases (MongoDB), etc.
  - Developed indexes that can be built in for self-describing formats, i.e., HDF5.
    - A hybrid index with Adaptive radix tree (ART) for strings and SBST for numeric attribute values


File Path List
FP = File Path
Object Path List
OP = Object Path
ART
(Attribute Names)
SBST
(Numeric Attribute Values)
ART
(String Attribute Values)
List<(File_Path_Identifier, Object_Path_Identifier)>



Autonomous data management using object storage – Proactive Data Containers (PDC)

Storage Systems and I/O: Current status



- **Challenges**
  - **Multi-level hierarchy complicates data movement, especially if user has to be involved**
  - **POSIX-IO semantics hinder scalability and performance of file systems and IO software**

HPC data management requirements

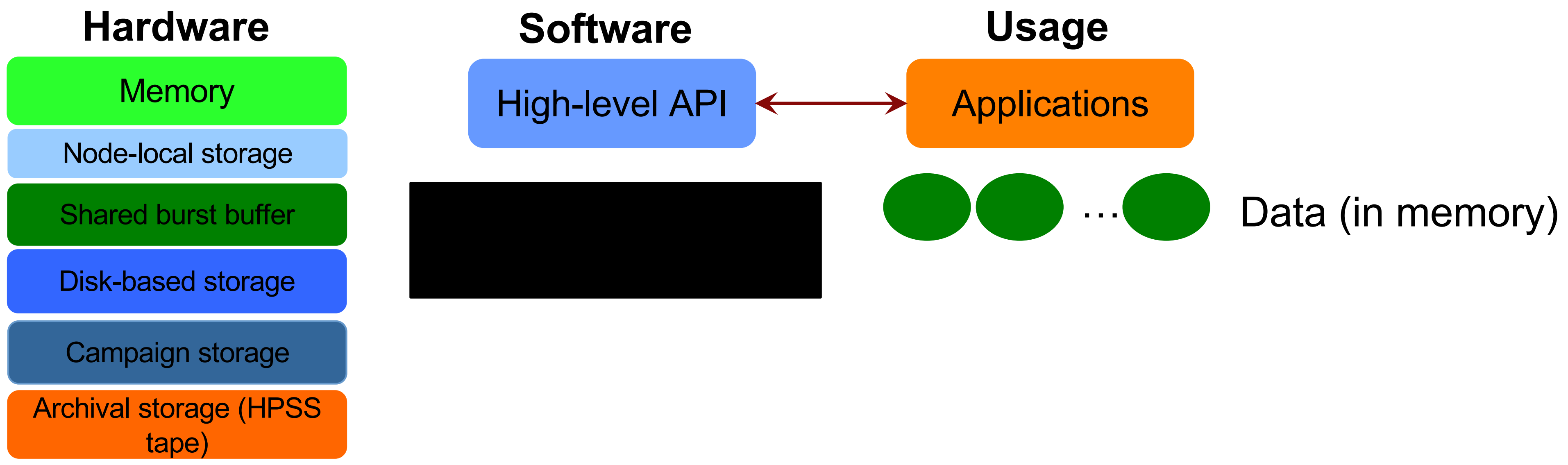
Use case	Domain	Sim/EOD/analysis	Data size	I/O Requirements
FLASH	High-energy density physics	Simulation	~1PB	Data transformations, scalable I/O interfaces, correlation among simulation and
CMB / Planck	Cosmology	Simulation, EOD/Analysis	10PB	Automatic data movement optimizations
DECam & LSST				es, data transformations
ACME	Climate	Simulation	~10PB	Async I/O, derived variables,
TECA	Climate	Analysis	10PB	Data organization and efficient data movement
HipMer	Genomics	EOD/Analysis	~100TB	Scalable I/O interfaces, efficient and automatic data movement

Easy interfaces and superior performance

Autonomous data management

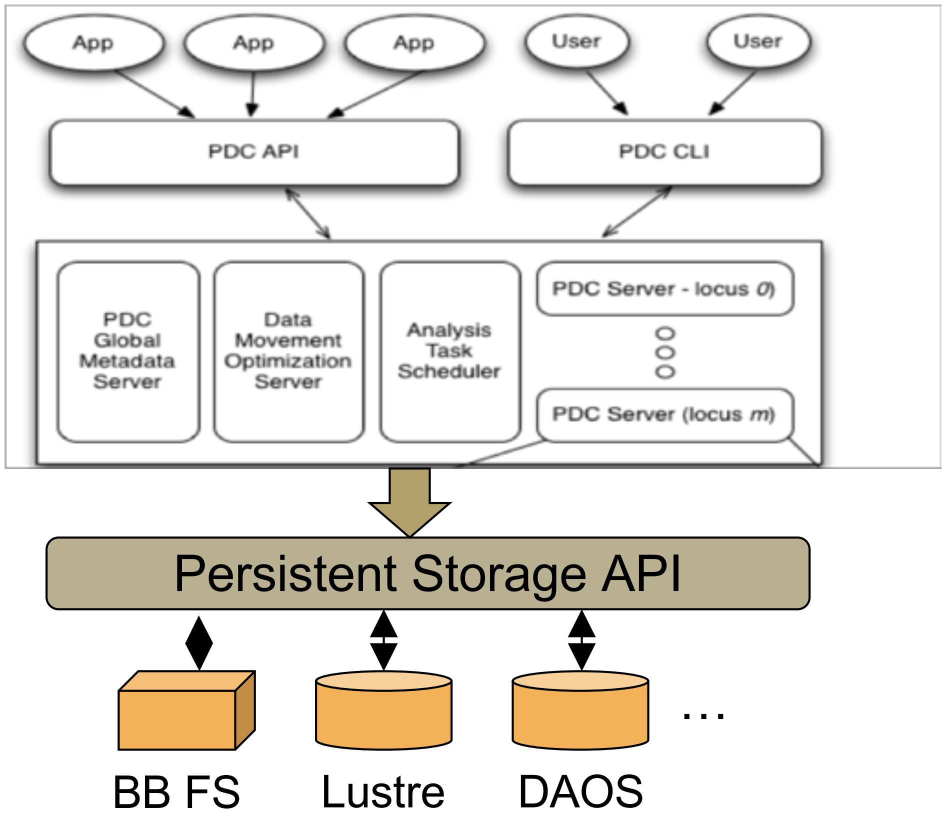
Information capture and management

Next Gen Storage – Proactive Data Containers (PDC)



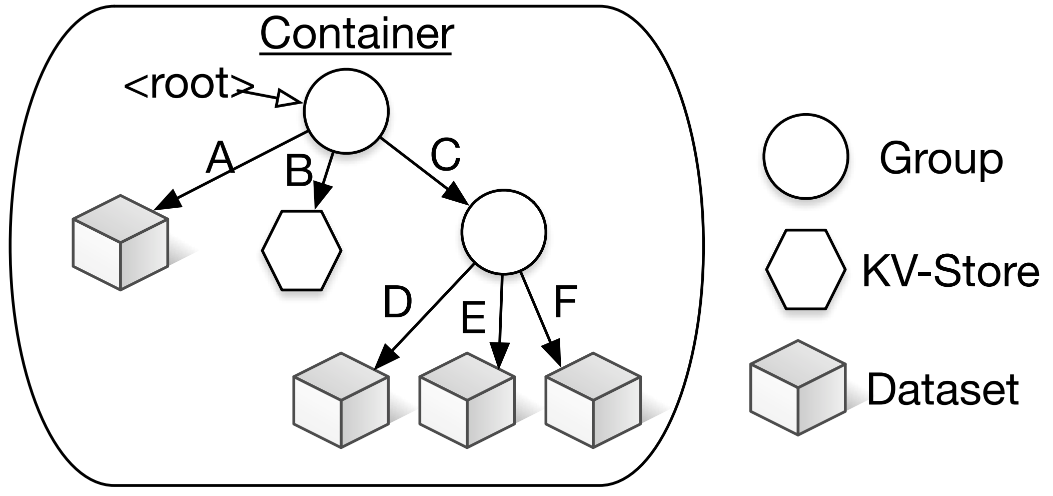
PDC System – High-level Architecture

- **Object-centric data access interface**
  - Simple put, get interface
  - Array-based variable access
- **Transparent data management**
  - Data placement in storage hierarchy
  - Automatic data movement
- **Information capture and management**
  - Rich metadata
  - Connection of results and raw data with relationships



Object-centric PDC Interface

- Object-level interface
  - Create – containers and objects
  - Add attributes
  - Put object
  - Get object
  - Delete object
- Array-specific interface
  - Create regions
  - Map regions in PDC objects
  - Lock
  - Release

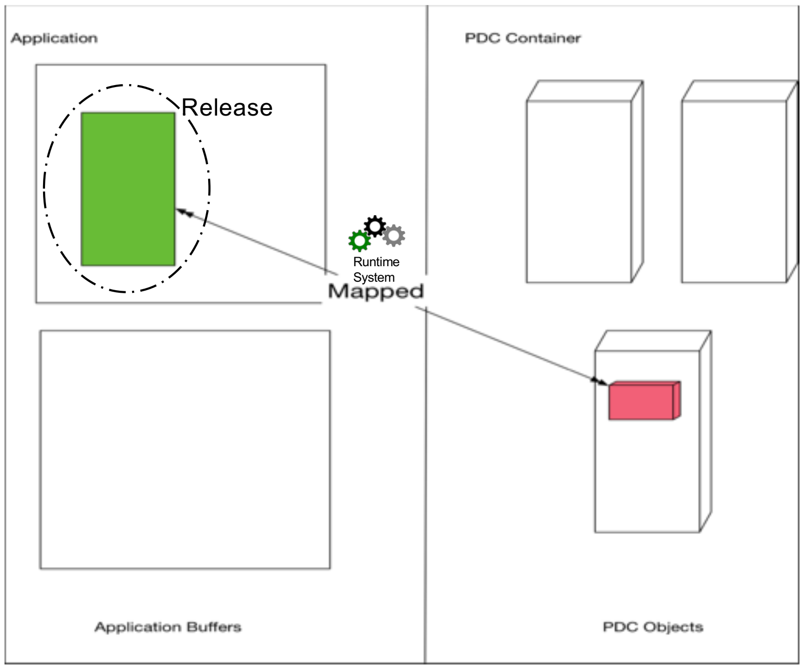


Proactive Data Container

J. Mu, J. Soumagne, et al., "A Transparent Server-managed Object Storage System for HPC", IEEE Cluster 2018

Object-centric PDC Interface

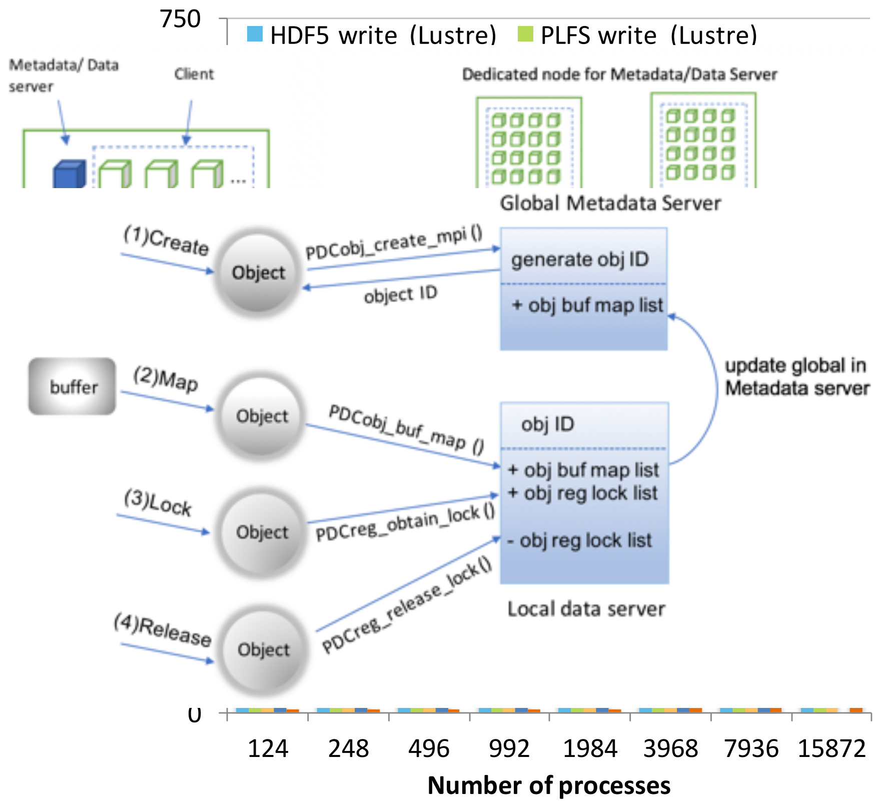
- Object-level interface
  - Create – containers and objects
  - Add attributes
  - Put object
  - Get object
  - Delete object
- Array-specific interface
  - Create regions
  - Map regions in PDC objects
  - Lock
  - Release



J. Mu, J. Soumagne, et al., “A Transparent Server-managed Object Storage System for HPC”, IEEE Cluster 2018

Transparent data movement in storage hierarchy

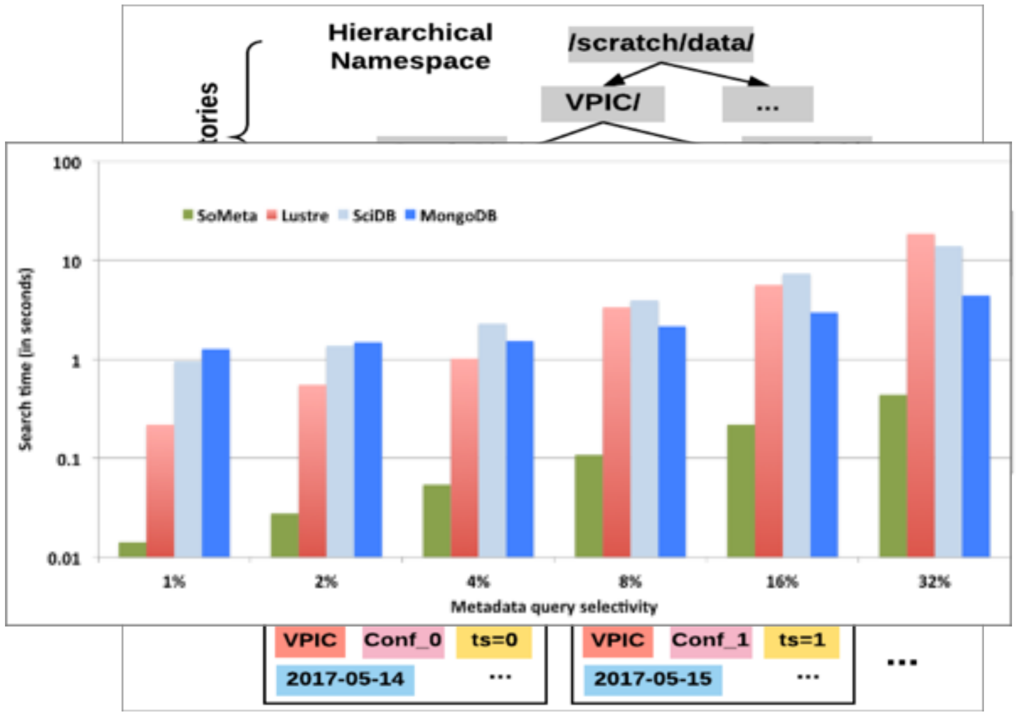
- Usage of compute resources for I/O
  - Shared mode – Compute nodes are shared between applications and I/O services
  - Dedicated mode – I/O services on separate nodes
- Transparent data movement by PDC servers
  - Apps map data buffers to objects and PDC servers place and manage data
  - Apps query for data objects using attributes
- Superior I/O performance



H. Tang, S. Byna, et al., "Toward Scalable and Asynchronous Object-centric Data Management for HPC", IEEE/ACM CCGrid 2018

Metadata management

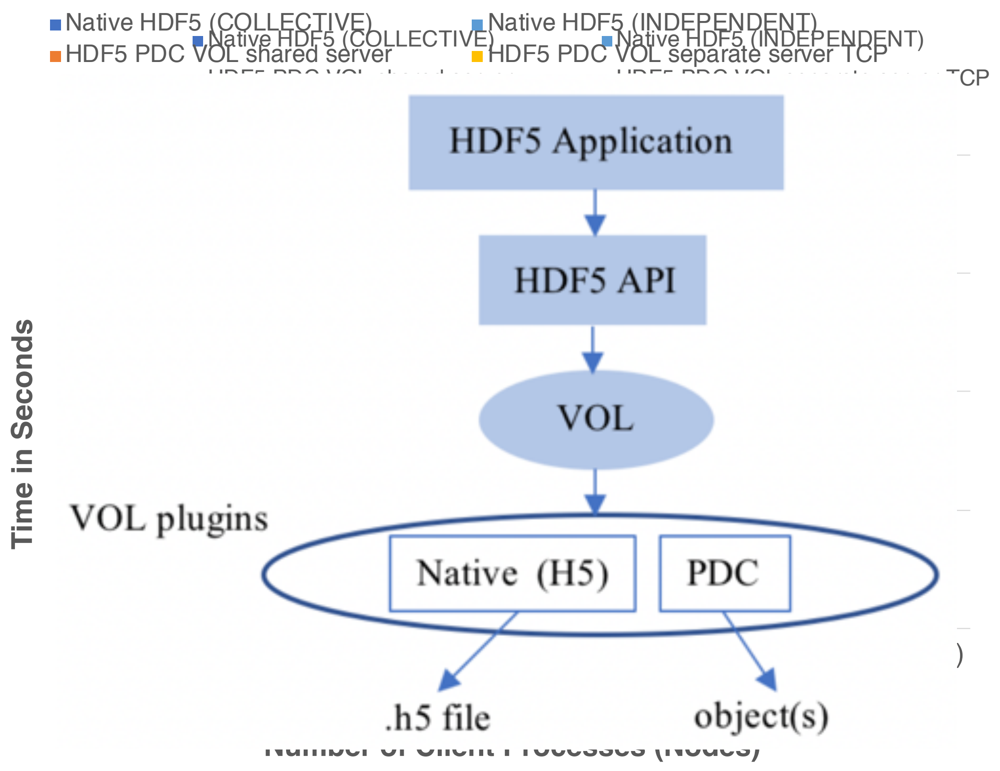
- Flat name space
- Rich metadata
  - Pre-defined tags that includes provenance
  - User-defined tags for capturing relationships between data objects
- Distributed in memory metadata management
  - Distributed hash table and bloom filters used for faster access



H. Tang, S. Byna, et al., “SoMeta: Scalable Object-centric Metadata Management for High Performance Computing”, to be presented at IEEE Cluster 2017

HDF5 and PDC bridge

- Developed a HDF5 Virtual Object Layer (VOL) to make PDC available to all HDF5 applications
- Minimal code change for HDF5 applications and working towards no code change requirement
  - 2X to 7X speed up with dedicated mode of PDC


■ Native HDF5 (COLLECTIVE)
■ Native HDF5 (INDEPENDENT)
■ HDF5 PDC VOL shared server
■ HDF5 PDC VOL separate server TCP
Time in Seconds
HDF5 Application
HDF5 API
VOL
VOL plugins
Native (H5)
PDC
.h5 file
object(s)

Collaborators: THG

Conclusions

- HDF5 file format is highly used and the ECP ExaHDF5 project is optimizing the library and tools
- Features
  - Virtual Object Layer (VOL)
  - Data Elevator and async I/O VOL connectors
  - Full SWMR and parallel querying
  - In progress: subfiling, topology-aware I/O, etc.
  - EOD management – remote sync, MWMR, metadata querying, etc.
- Towards automated object-centric data management, developing Proactive Data Containers (PDC) runtime system
  - Scalable namespace management, automated and async data movement, rich in-memory metadata management services
  - Bridge between HDF5 and PDC via a VOL connector

Thank you!

- Contact:
  - Suren Byna (sdm.lbl.gov/~sbyna/) [SByna@lbl.gov]
- Contributions to this presentation
  - ExaHDF5 project team (sdm.lbl.gov/exahdf5)
  - Proactive Data Containers (PDC) team (sdm.lbl.gov/pdc)
  - SDM group: sdm.lbl.gov