Efficient Scientific Data Management on Supercomputers

Suren Byna

Staff Scientist

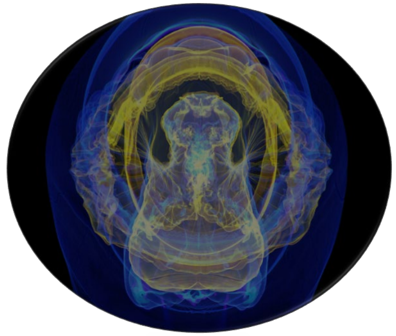
Scientific Data Management Group

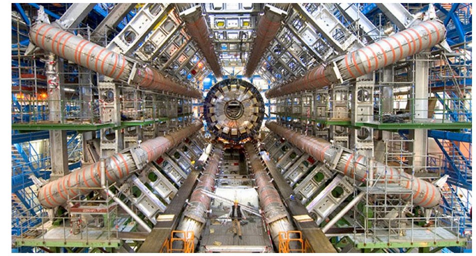
Data Science and Technology Department

Lawrence Berkeley National Laboratory

Scientific Data - Where is it coming from?

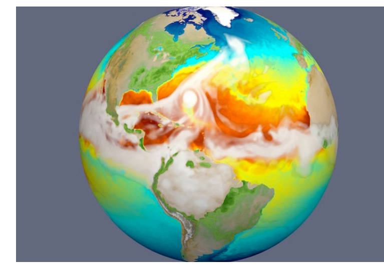
- Simulations



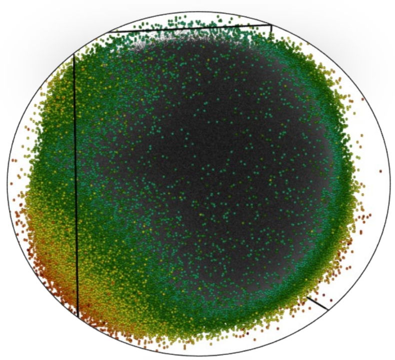


- Experiments

ALS
ADVANCED LIGHT SOURCE

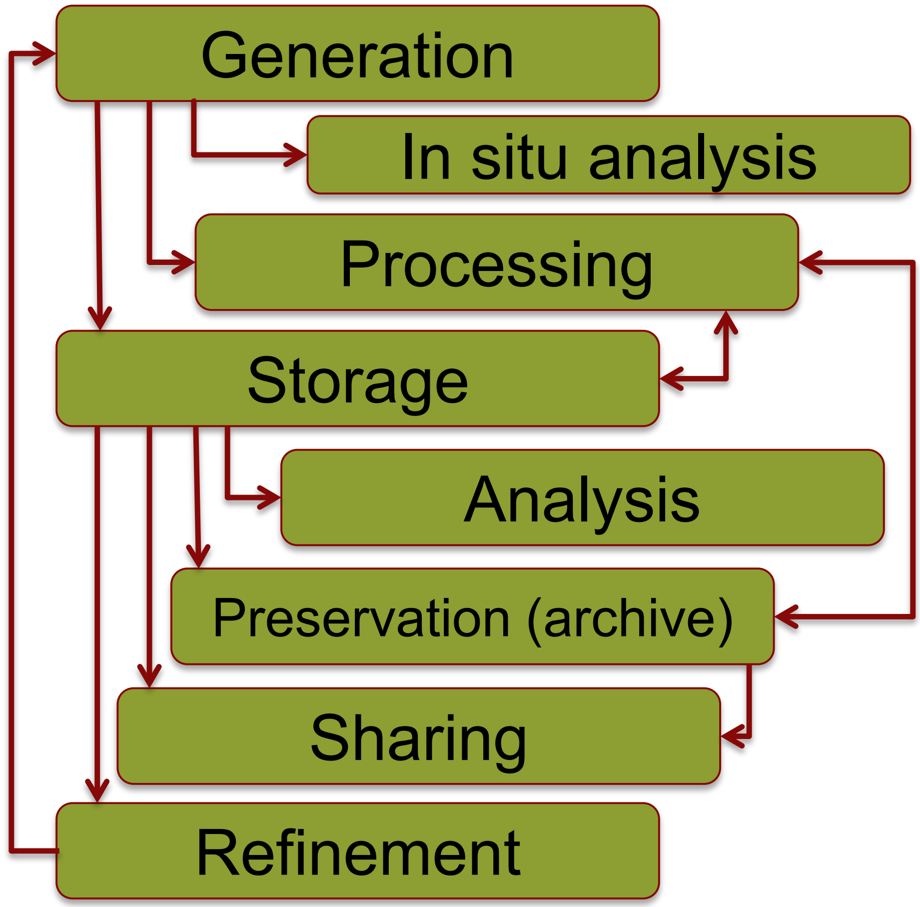


- Observations



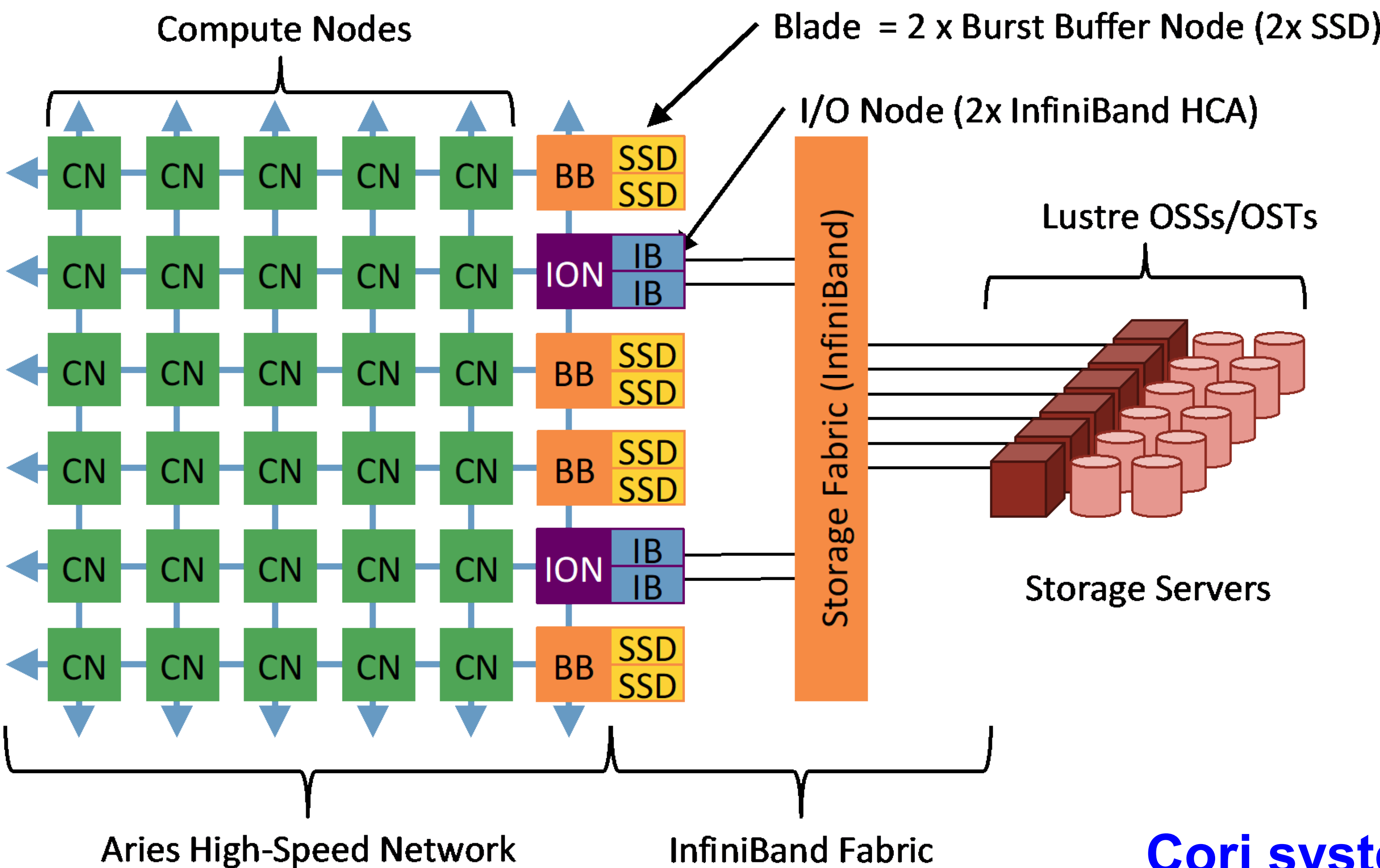
SLAC
NATIONAL ACCELERATOR LABORATORY

Life of scientific data



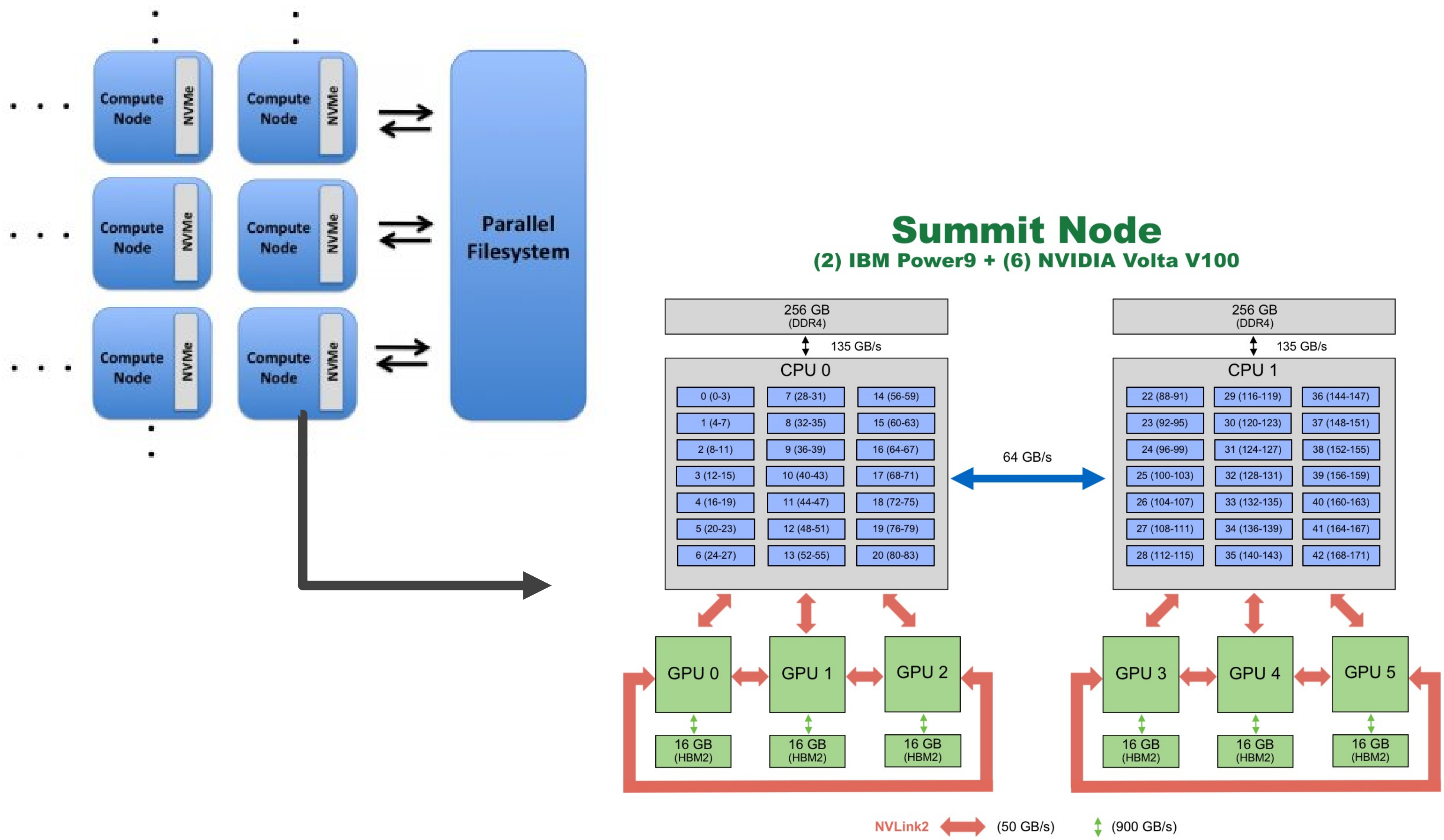
Supercomputing systems

Supercomputer architecture - Cori



Cori system

Supercomputer architecture - Summit



Scientific Data Management in supercomputers

- Data representation
  - Metadata, data structures, data models
- Data storage
  - Storing and retrieving data and metadata to file systems fast
- Data access
  - Improving performance of data access that scientists desire
- Facilitating analysis
  - Strategies for supporting finding the meaning in the data
- Data transfers
  - Transfer data within a supercomputing system and between different systems

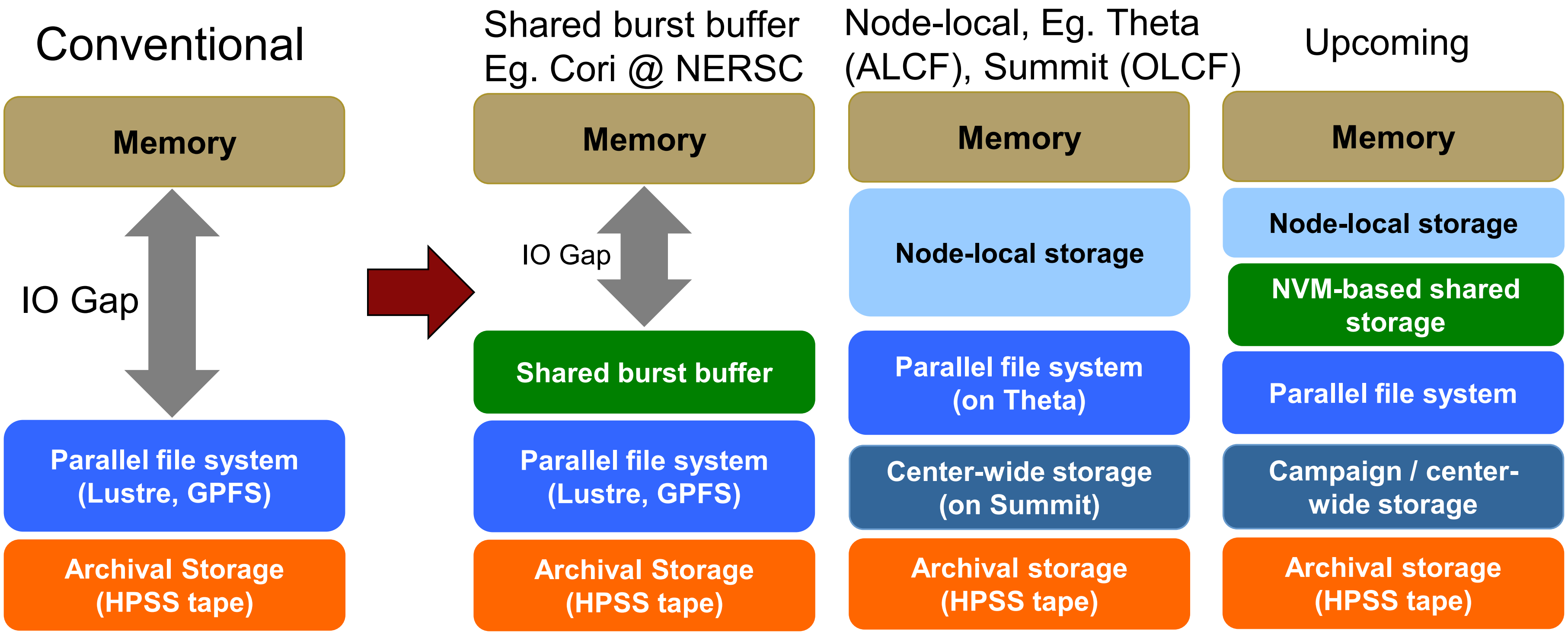
Scientific Data Management in supercomputers

- Data representation
  - Metadata, data structures, data models
- Data storage
  - Storing and retrieving data and metadata to file systems fast
- Data access
  - Improving performance of data access that scientists desire
- Facilitating analysis
  - Strategies for supporting finding the meaning in the data
- Data transfers
  - Transfer data within a supercomputing system and between different systems

Focus of this presentation

- Storing and retrieving data – Parallel I/O and HDF5
  - Software stack
  - Modes of parallel I/O
  - Intro to HDF5 and some tuning I/O of exascale applications
- Autonomous data management system
  - Proactive Data Containers (PDC) system
  - Metadata management service
  - Data management service

Trends – Storage system transformation



- 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

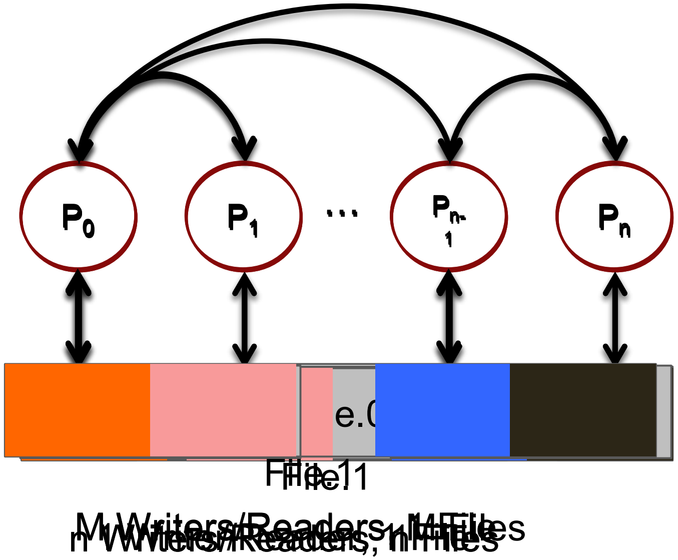
Parallel I/O software stack

- 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 (The HDF Group) [LBL, ANL]**
  - ADIOS (ORNL)
  - PnetCDF (Northwestern, ANL)
  - NetCDF-4 (UCAR)
- Middleware – POSIX-IO, MPI-IO (ANL)
- I/O Forwarding
- File systems: Lustre (Intel), GPFS (IBM), DataWarp (Cray), …
- I/O Hardware (disk-based, SSD-based, …)

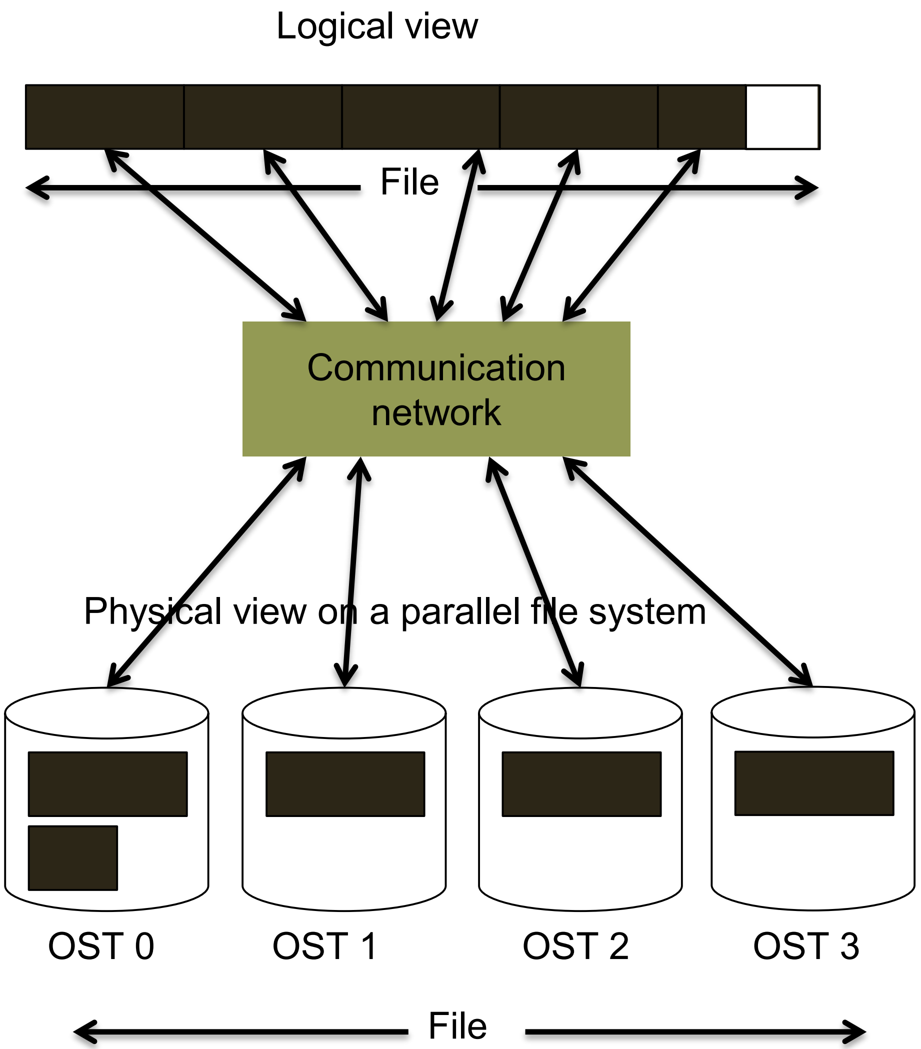
Parallel I/O – Application view

- **Types of parallel I/O**
  - 1 writer/reader, 1 file
  - N writers/readers, N files (File-per-process)
  - N writers/readers, 1 file
  - M writers/readers, 1 file
    - Aggregators
    - Two-phase I/O
  - M aggregators, M files (file-per-aggregator)
    - Variations of this mode



Parallel I/O – System view

- Parallel file systems
  - Lustre and Spectrum Scale (GPFS)
- Typical building blocks of parallel file systems
  - Storage hardware – HDD or SSD RAID
  - Storage servers (in Lustre, Object Storage Servers [OSS], and object storage targets [OST]
  - Metadata servers
  - Client-side processes and interfaces
- Management
  - Stripe files for parallelism
  - Tolerate failures



Applications

High Level I/O Library (HDF5, NetCDF, ADIOS)

I/O Middleware (MPI-IO)

I/O Forwarding

Parallel File System (Lustre, GPFS,..)

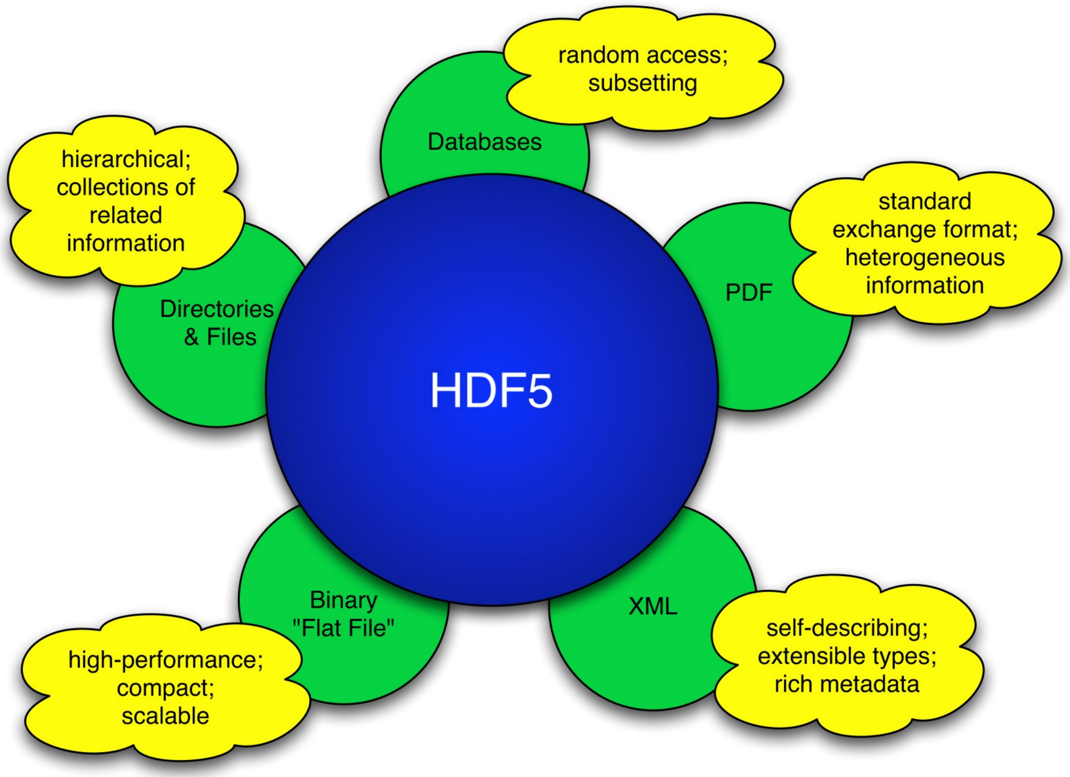
I/O Hardware

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 …


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

HDF5 is designed …

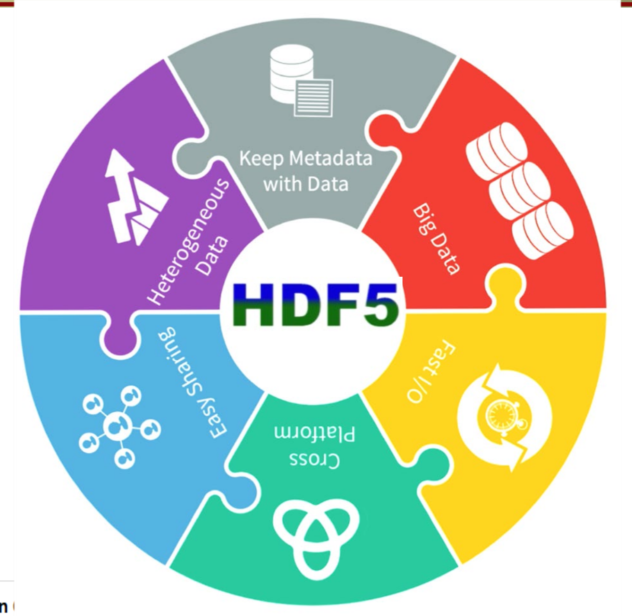
- for high volume and / or complex data
- for every size and type of system – from cell phones to supercomputers
- for flexible, efficient storage and I/O
- to enable applications to evolve in their use of HDF5 and to accommodate new models
- to support long-term data preservation

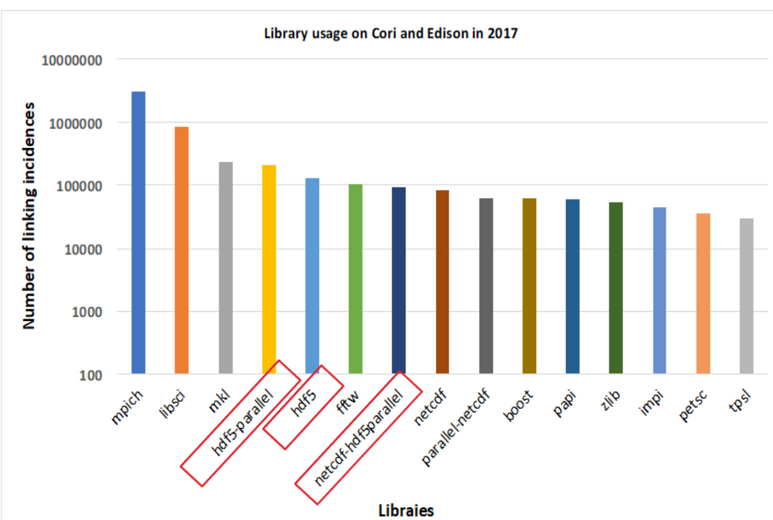
HDF5 Overview

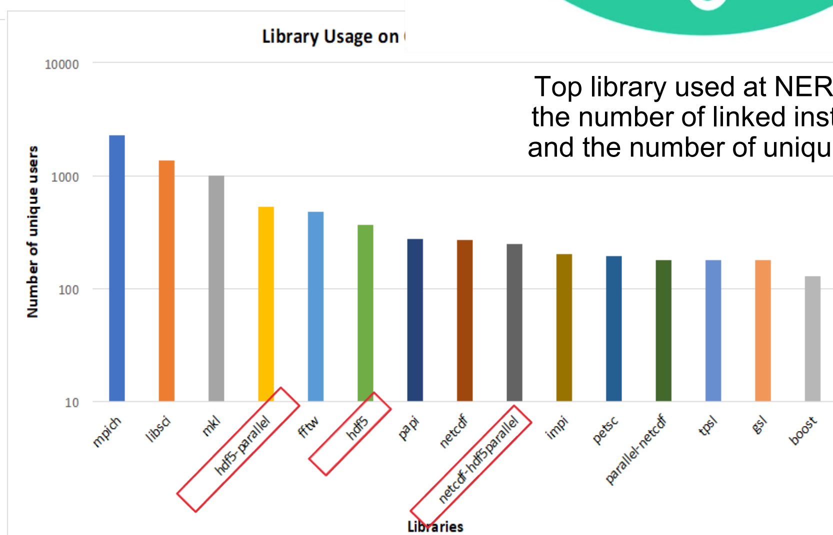
- HDF5 is designed to organize, store, discover, access, analyze, share, and preserve diverse, complex data in continuously evolving heterogeneous computing and storage environments.
- First released in 1998, maintained by **The HDF Group**

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

- Heavily used on DOE supercomputing systems

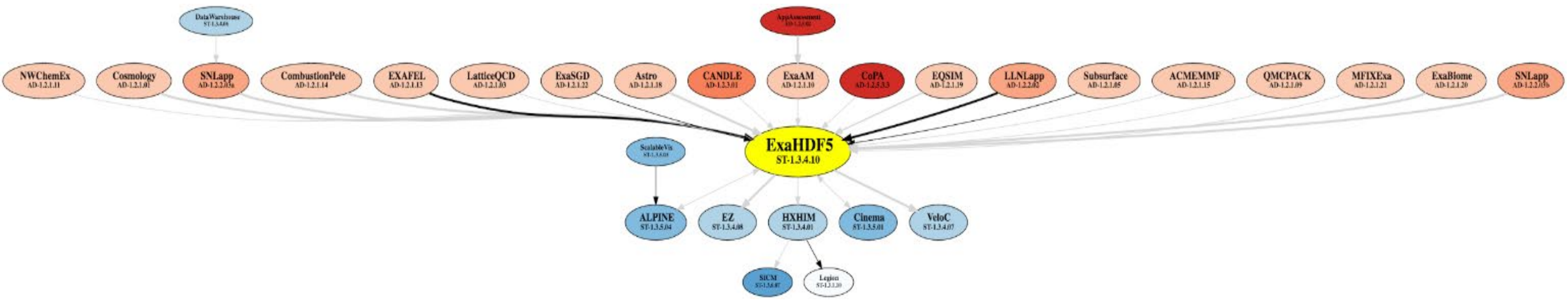




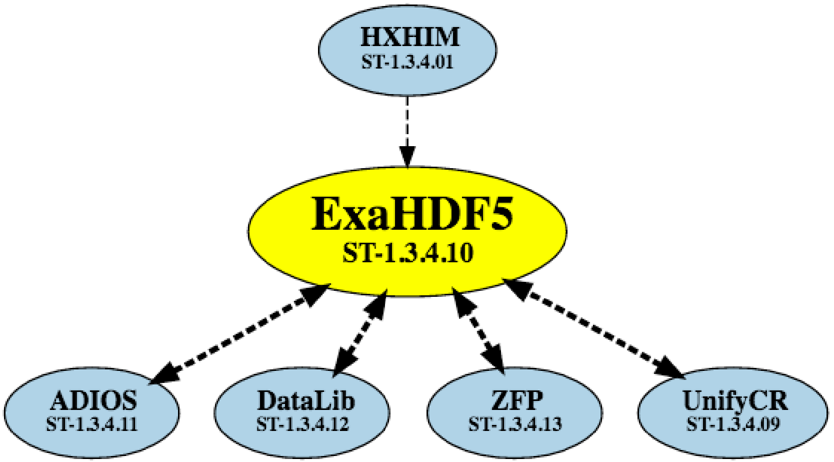


Top library used at NERSC by the number of linked instances and the number of unique users

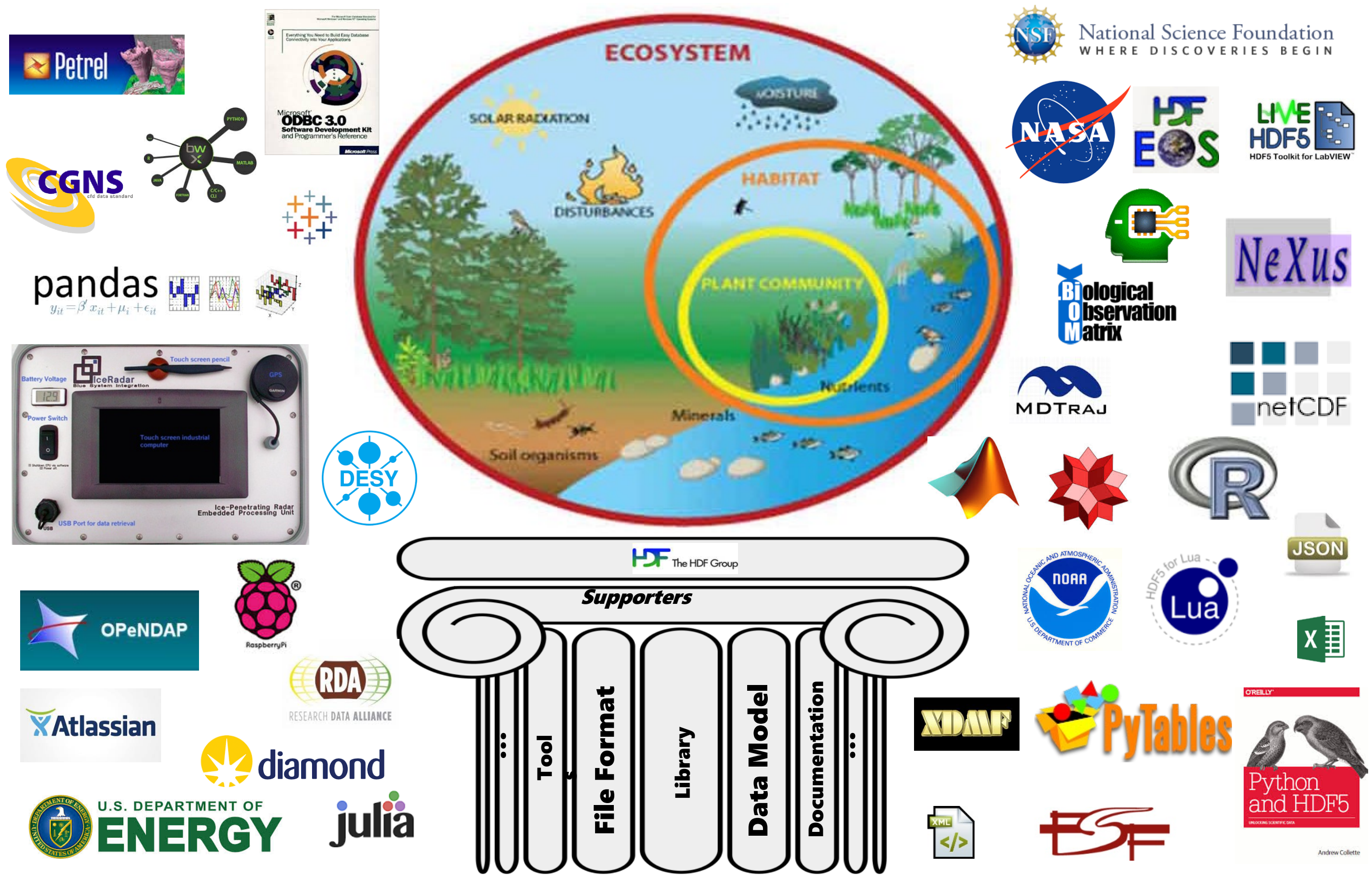
HDF5 in Exascale Computing Project



19 out of the 26 (22 ECP + 4 NNSA) apps currently use or planning to use HDF5



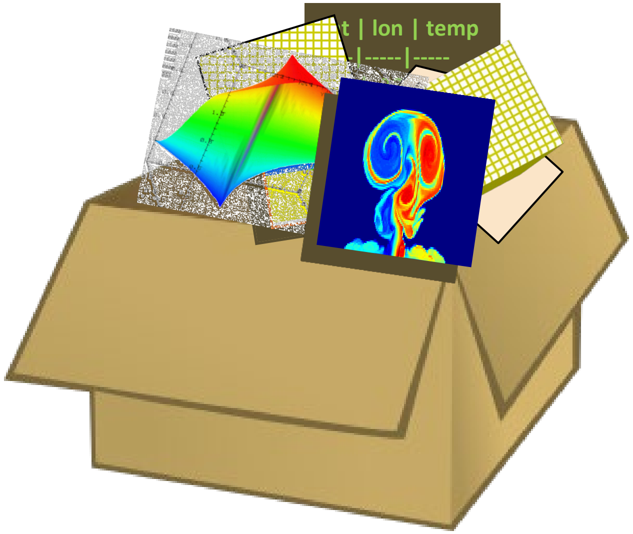
HDF5 Ecosystem



HDF5 DATA MODEL

HDF5 File

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



HDF5 Data Model

Dataset

Group

Attribute

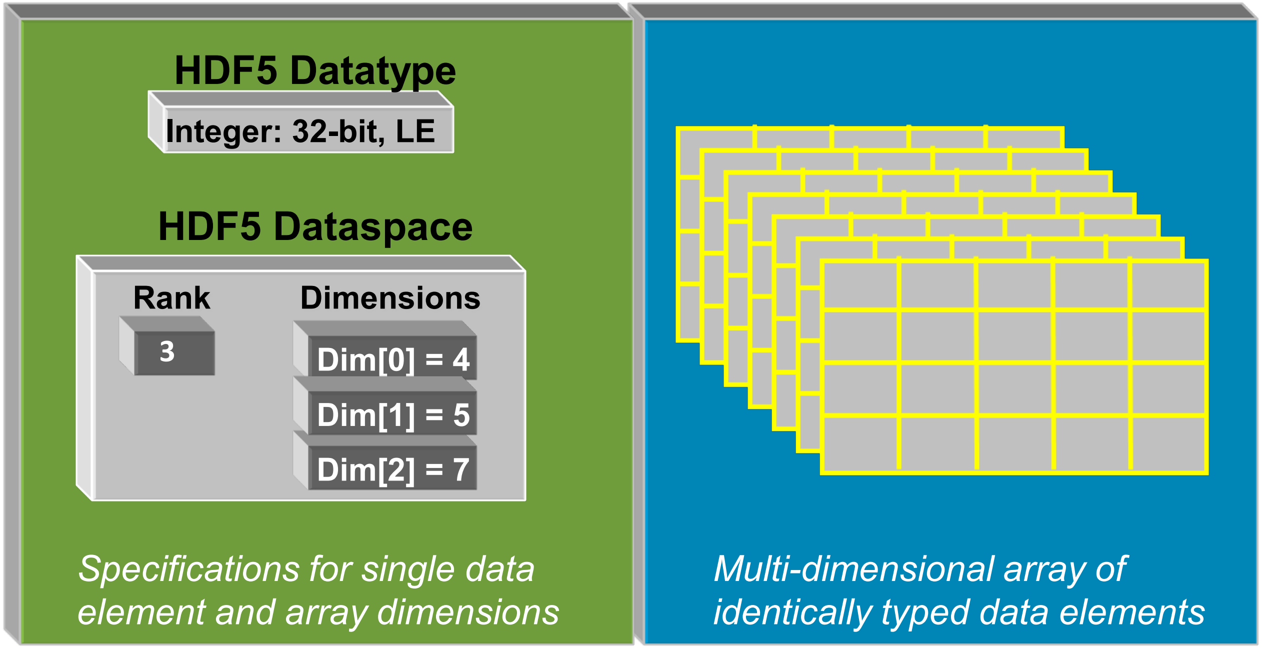
Link

Datatype

Dataspace

File

HDF5 Dataset



- HDF5 datasets **organize and contain** data elements.
- HDF5 datatype describes individual data elements.
- HDF5 dataspace describes the logical layout of the data elements.

HDF5 Dataspace

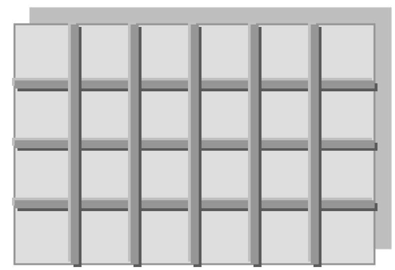
- Describe individual data elements in an HDF5 dataset
- Wide range of datatypes supported
  - Integer
  - Float
  - Enum
  - Array
  - User-defined (e.g., 13-bit integer)
  - Variable-length types (e.g., strings, vectors)
  - Compound (similar to C structs)
  - More …

HDF5 Dataspace

Two roles:

Dataspace contains spatial information

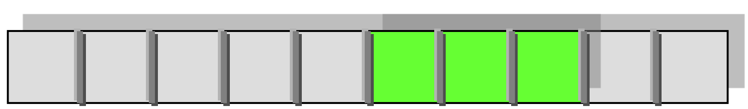
- Rank and dimensions
- Permanent part of dataset definition



Rank = 2

Dimensions = 4x6

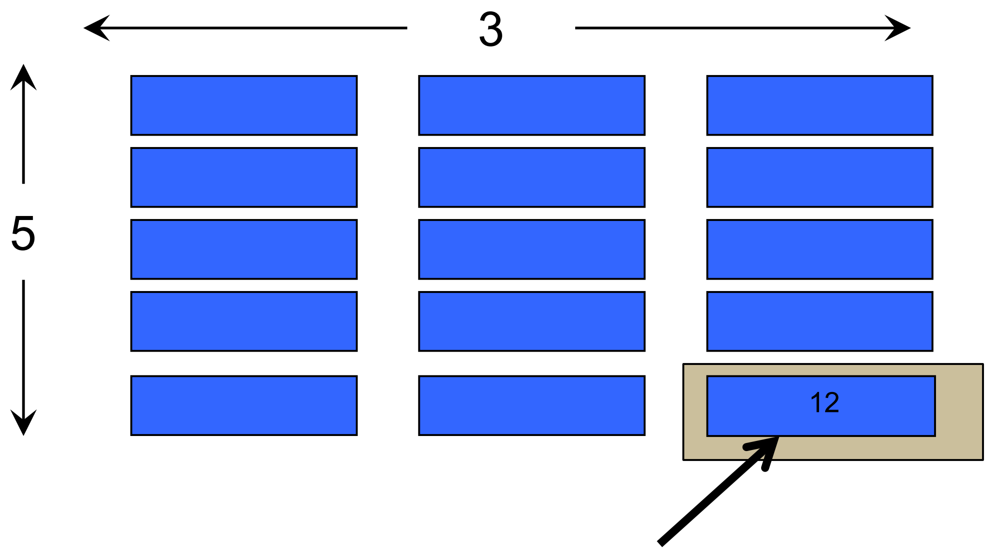
Partial I/0: Dataspace describes application's data buffer and data elements participating in I/O



Rank = 1

Dimension = 10

HDF5 Dataset with a 2D array

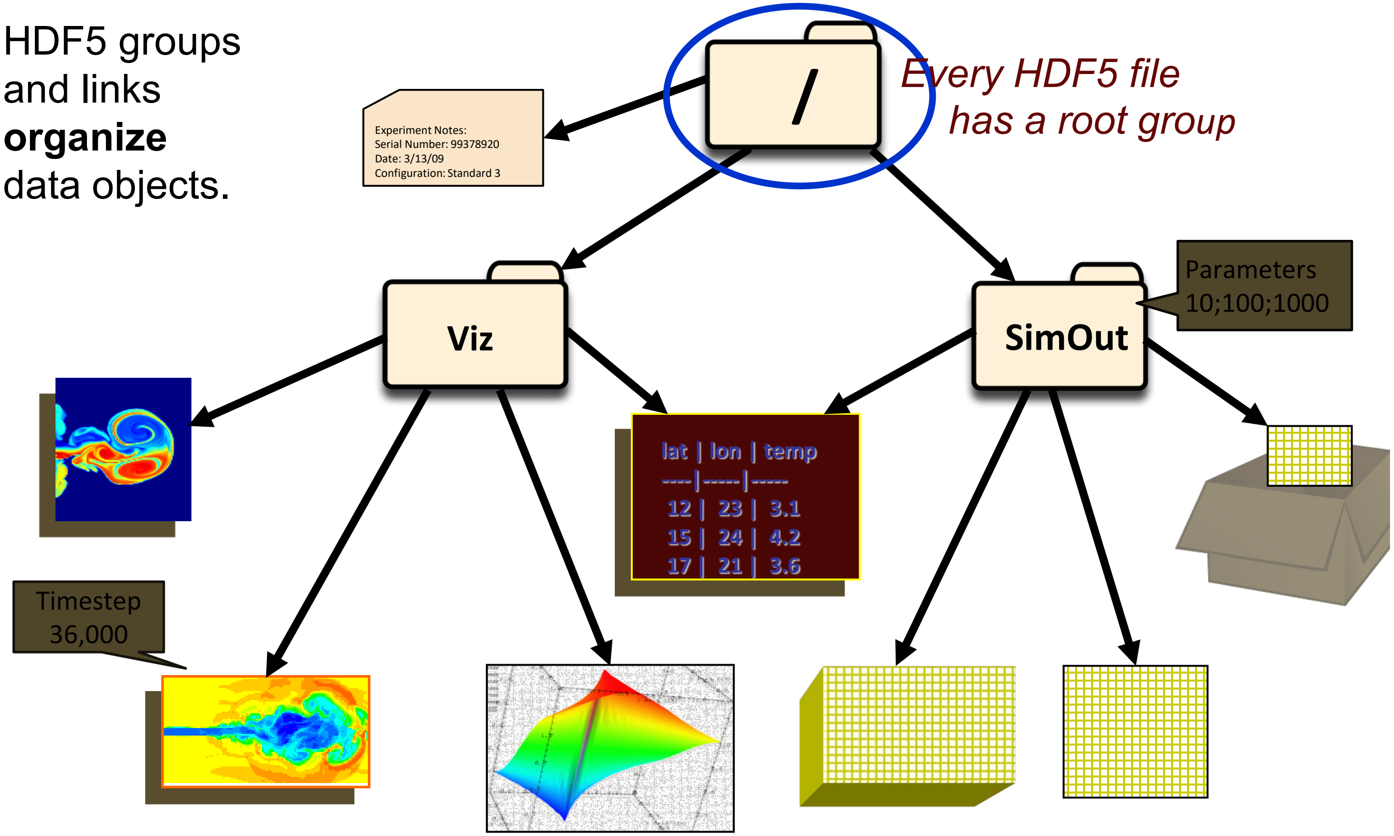


Datatype: 32-bit Integer

Dataspace: Rank = 2
Dimensions = 5 x 3

HDF5 Groups and Links

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



HDF5 Attributes

- Typically contain user metadata
- Have a <u>name</u> and a <u>value</u>
- Attributes “decorate” HDF5 objects
- Value is described by a datatype and a dataspace
- Analogous to a dataset, but do not support partial I/O operations
  - Nor can they be compressed or extended

HDF5 Home Page

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

- Latest release: HDF5 1.10.5 (1.12 coming soon)

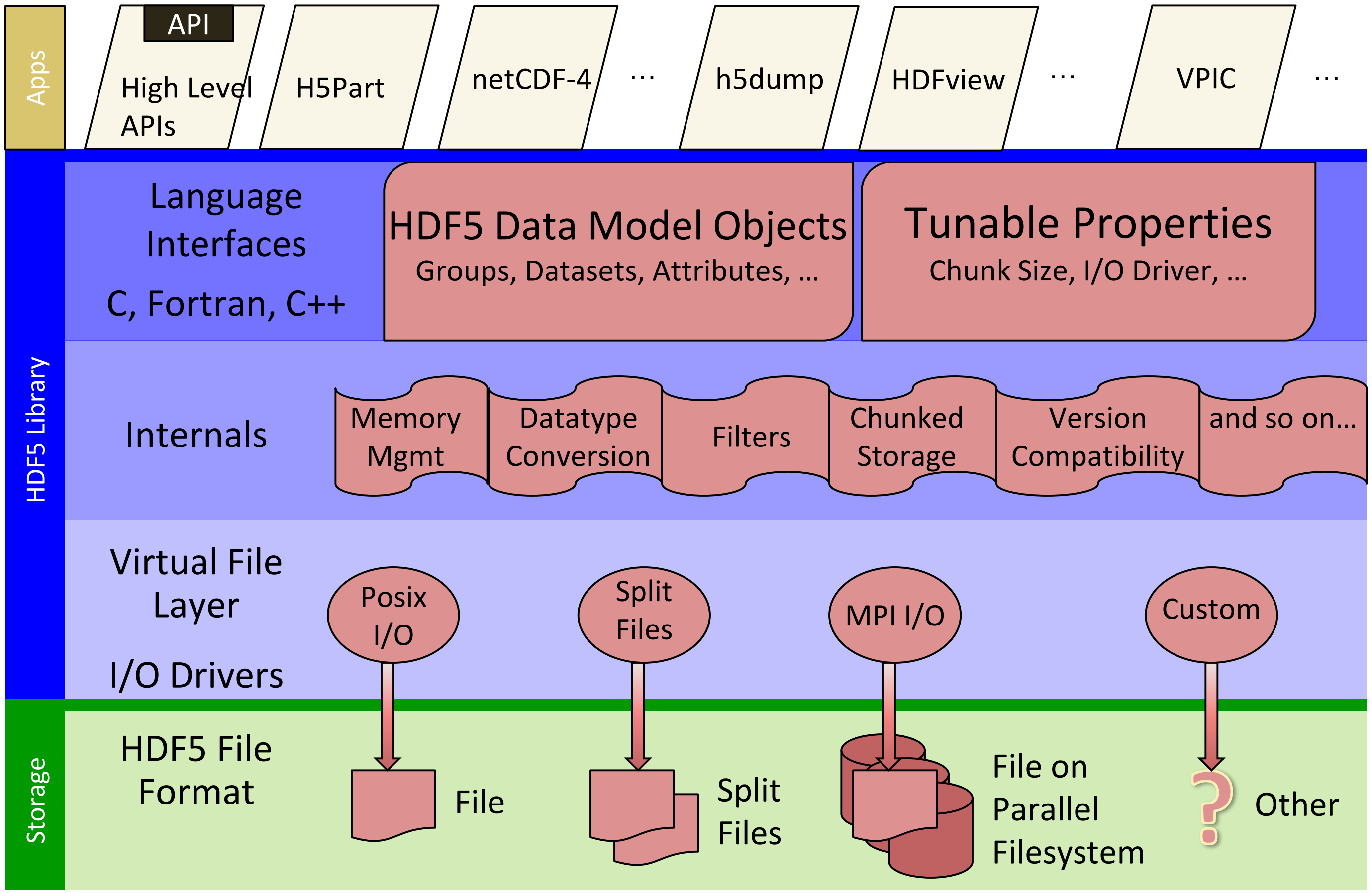
HDF5 source code:

- 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



The General HDF5 API

- C, Fortran, Java, C++, and .NET bindings
  - Also: IDL, MATLAB, Python (H5Py, PyTables), Perl, ADA, Ruby, …
- C routines begin with prefix: H5***?***

  ? is a character corresponding to the type of object the function acts on

Example Functions:

H5D : **D**ataset interface *e.g.,* **H5Dread**

H5F : **F**ile interface *e.g.,* **H5Fopen**

H5S : data**S**pace interface *e.g.,* **H5Sclose**

The HDF5 API

- For flexibility, the API is extensive
  - ✓ 300+ functions

Victorinox
Swiss Army
Cybertool
34

- This can be daunting… but there is hope
  - ✓ A few functions can do a lot
  - ✓ Start simple
  - ✓ Build up knowledge as more features are needed

General Programming Paradigm

- Object is opened or created
- Object is accessed, possibly many times
- Object is closed

- Properties of object are <u>optionally</u> defined
  - ✓ Creation properties (e.g., use chunking storage)
  - ✓ Access properties

Basic Functions

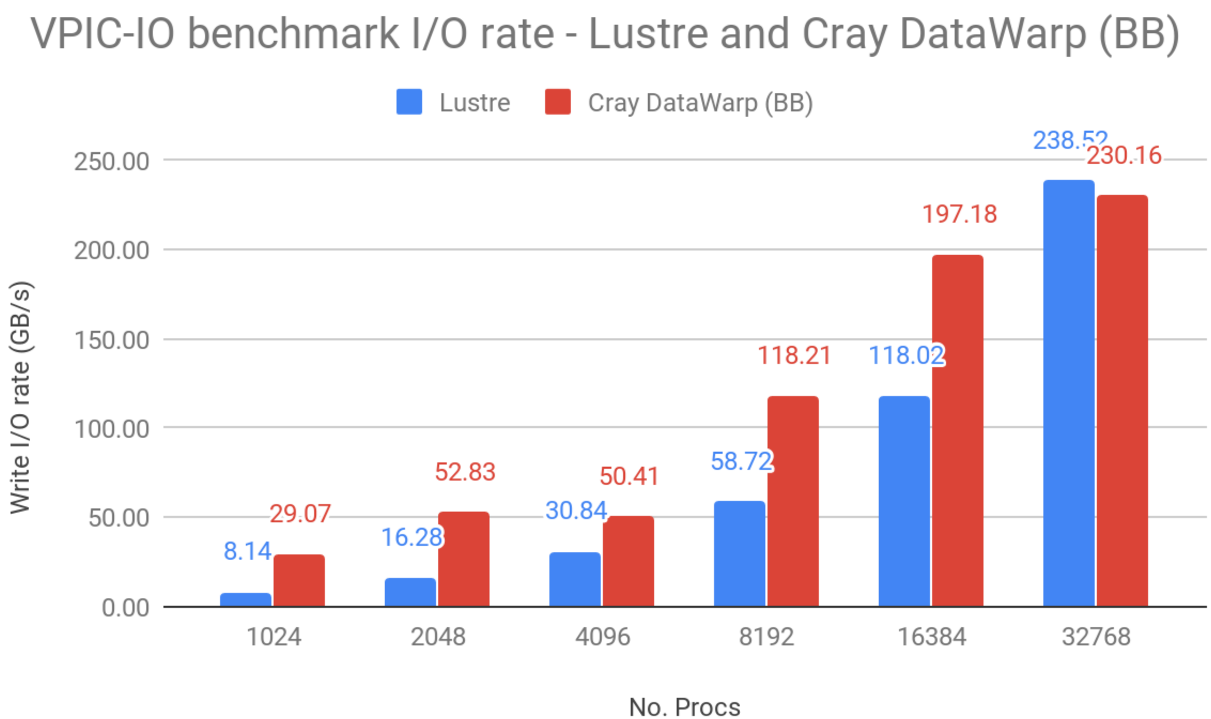
H5**F**create (H5**F**open)	*create (open) File*
H5**S**create_simple/H5**S**create	*create dataSpace*
H5**D**create (H5**D**open)	*create (open) Dataset*
H5**D**read, H5**D**write	*access Dataset*
H5**D**close	*close Dataset*
H5**S**close	*close dataSpace*
H5**F**close	*close File*

Other Common Functions

DataSpaces:	H5Sselect_hyperslab (Partial I/O) H5Sselect_elements (Partial I/O) H5Dget_space
DataTypes:	H5Tcreate, H5Tcommit, H5Tclose H5Tequal, H5Tget_native_type
Groups:	H5Gcreate, H5Gopen, H5Gclose
Attributes:	H5Acreate, H5Aopen_name, H5Aclose H5Aread, H5Awrite
Property lists:	H5Pcreate, H5Pclose H5Pset_chunk, H5Pset_deflate

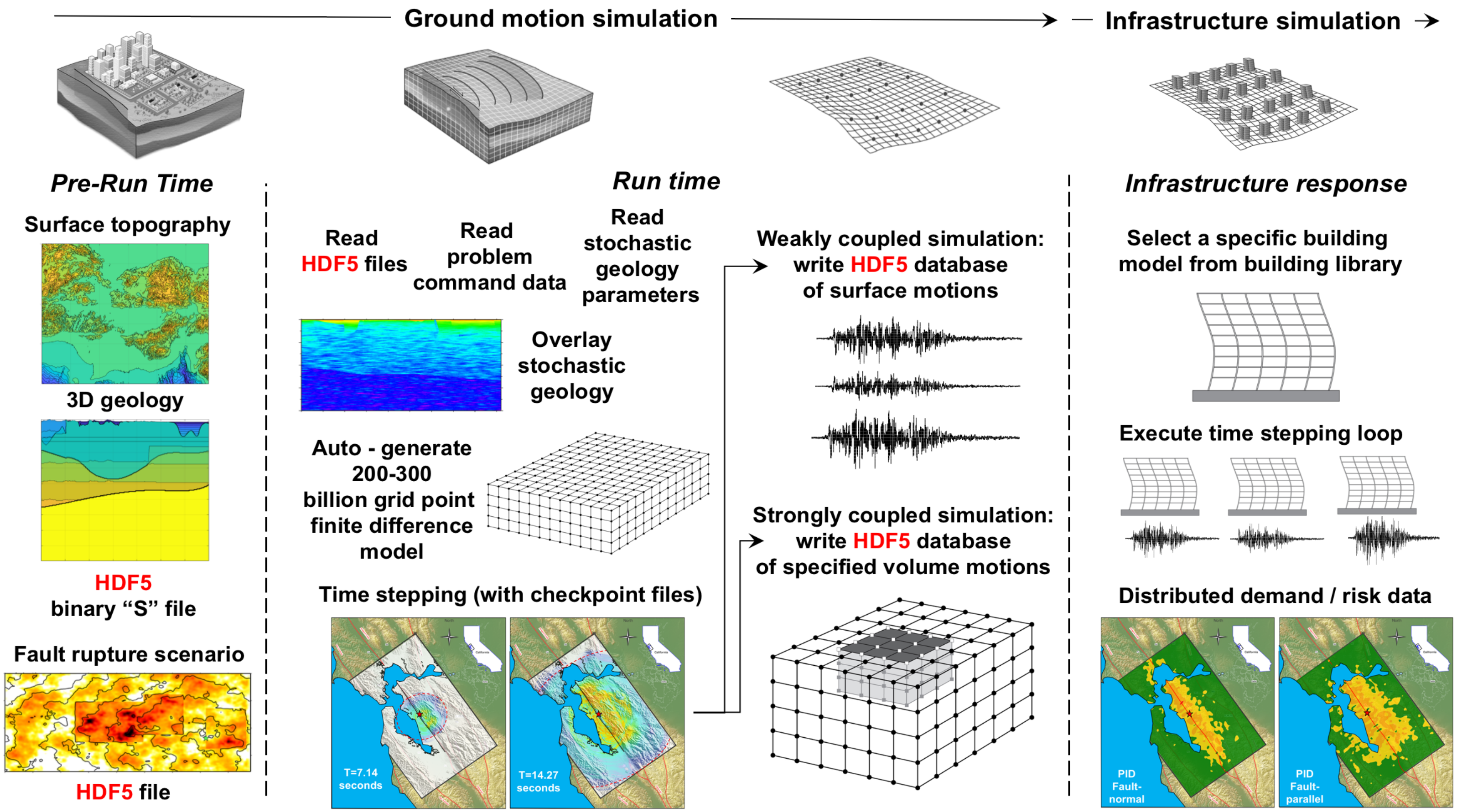
HDF5 performance on supercomputers

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

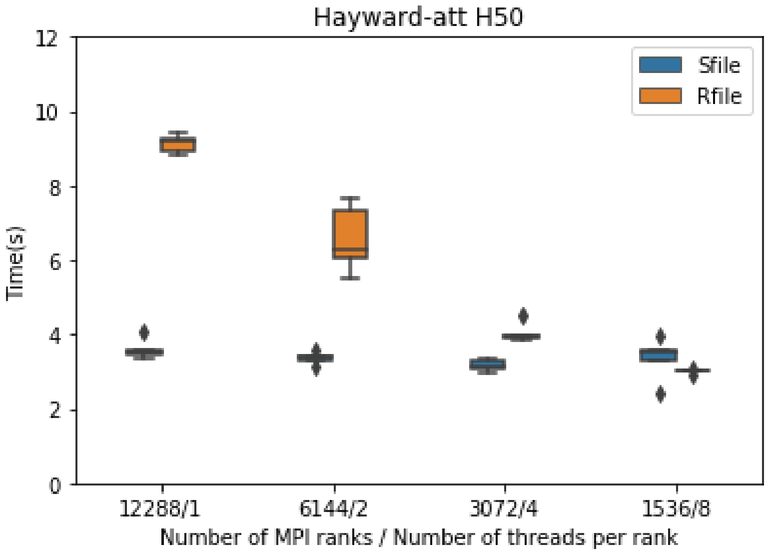


Applications: EQSIM

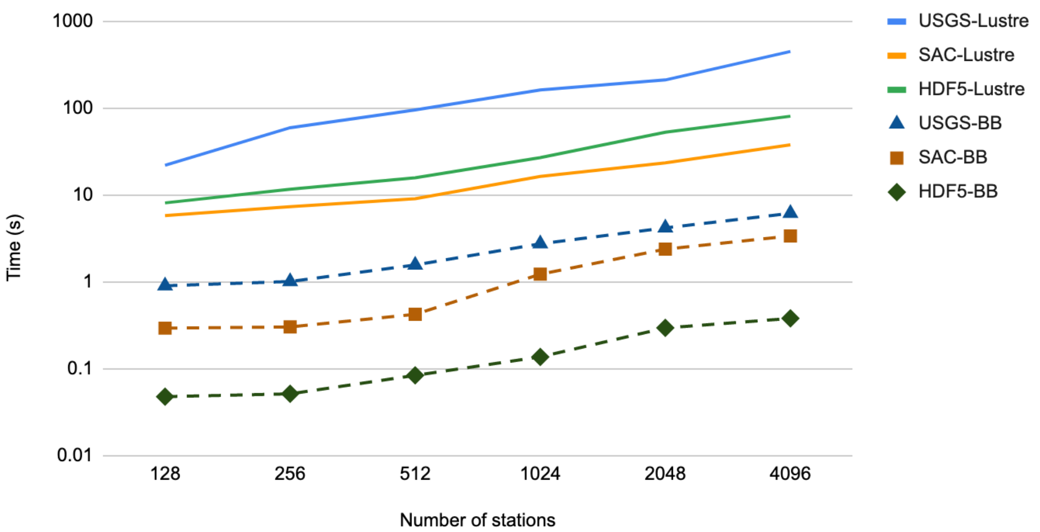
EQSIM is a high performance, multidisciplinary simulation for regional-scale earthquake hazard and risk assessments.



Applications: EQSIM



Read material properties from Sfile (HDF5) and Rfile (native), with varying number of MPI ranks.

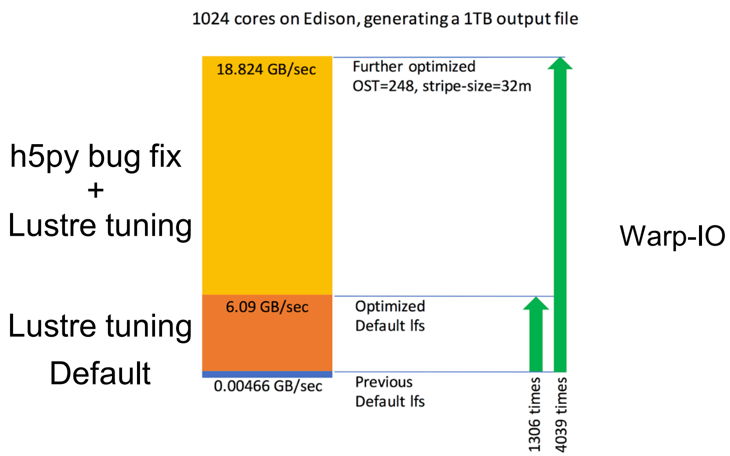


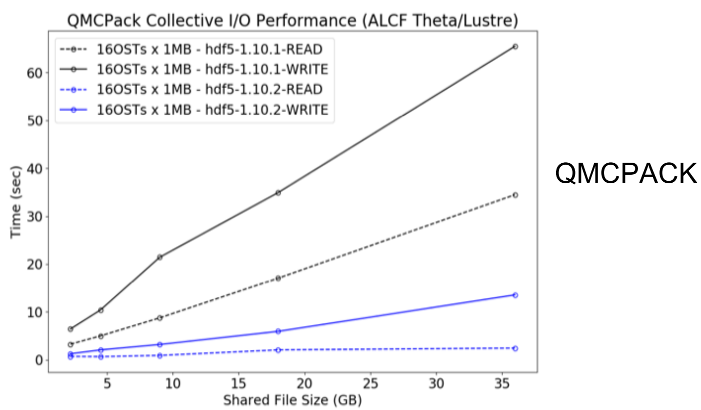
Write time-series data with different number of record stations to Lustre and burst buffer, on Cori with 64 nodes.

Applications: Warp and QMCPACK

- WarpX is an advanced electromagnetic Particle-In-Cell code
- Applied file system and MPI-IO level optimizations to achieve good HDF5 I/O performance (uses h5py)

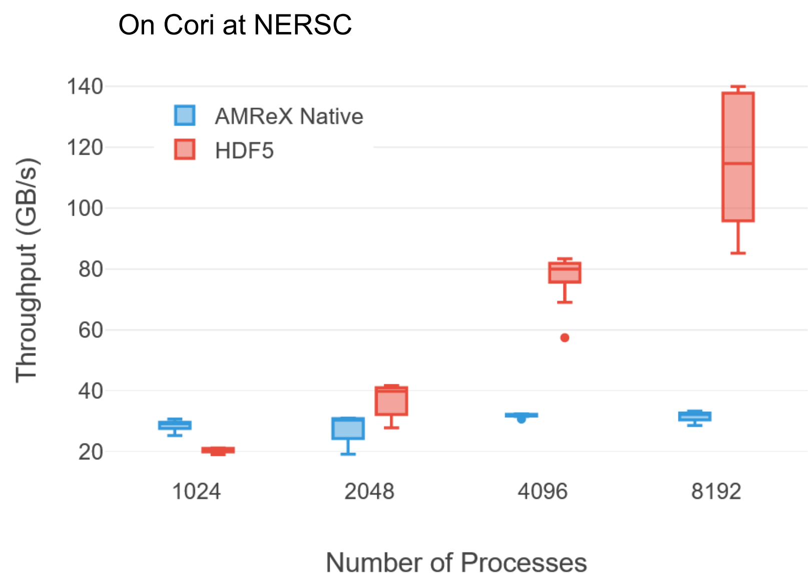
- QMCPACK, is a modern high-performance open-source Quantum Monte Carlo (QMC) simulation code
- HDF5 optimizations in file close and fixing a bug improved I/O performance

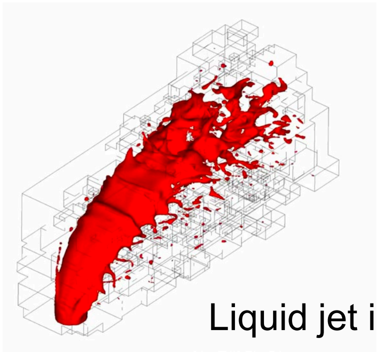




Applications: AMReX-based applications

- AMReX - SW framework for building massively parallel block- structured adaptive mesh refinement (AMR) applications
  - Combustion, accelerator physics, carbon capture, cosmology apps from ECP use this framework
- HDF5: Integrated HDF5-based I/O functions for reading and writing plot files and particle data

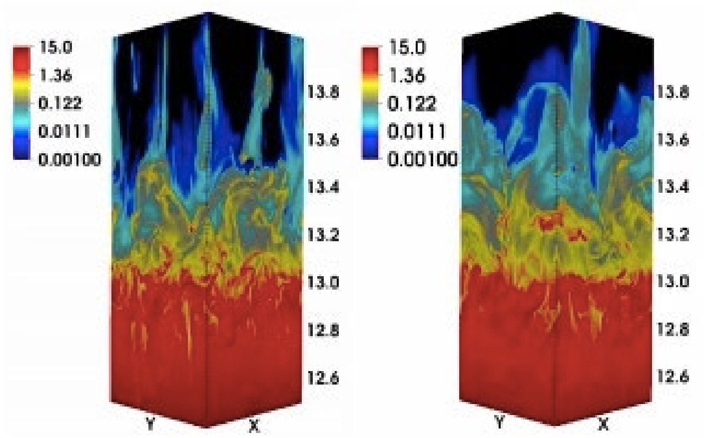




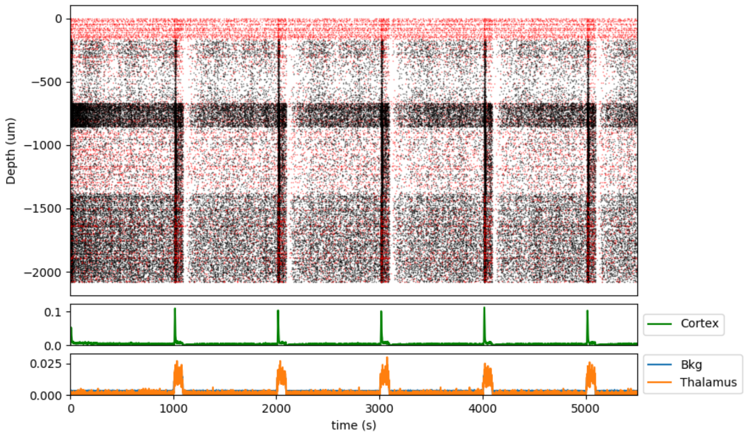
Liquid jet in supersonic flow

Facilities: Astrophysics and Neuroscience codes

- Supporting any I/O issue related tickets at facilities
- The following are astrophysics and neurological disorder pipelines that experienced high I/O overhead
- Used performance introspection interfaces of HDF5 to identify bottlenecks



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)

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

Storage Systems and I/O: Current status

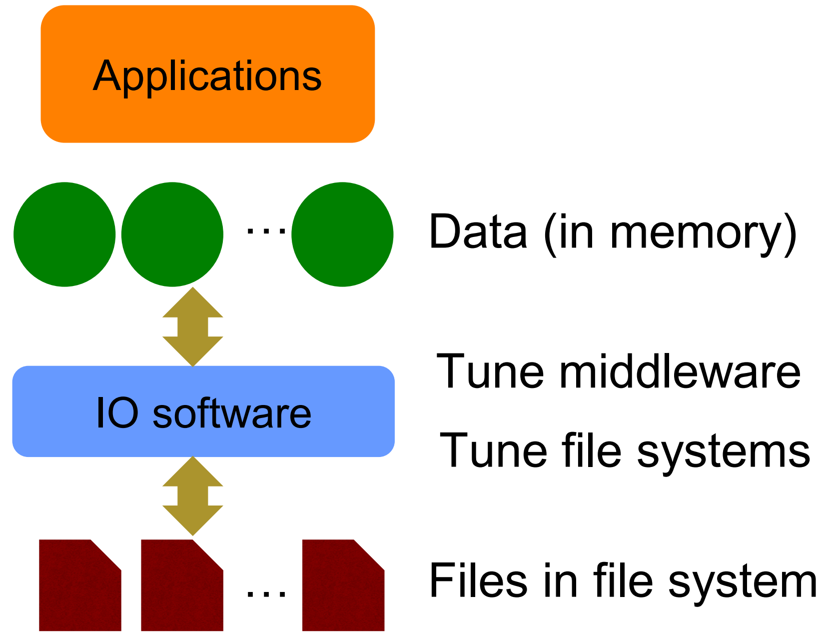
Hardware

- Memory
- Node-local storage
- Shared burst buffer
- Parallel file system
- Campaign storage
- Archival storage (HPSS tape)

Software

- High-level lib (HDF5, etc.)
- IO middleware (POSIX, MPI-IO)
- IO forwarding
- Parallel file systems

Usage



- ## 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)

Hardware

- Memory
- Node-local storage
- Shared burst buffer
- Disk-based storage
- Campaign storage
- Archival storage (HPSS tape)

Software

High-level API ↔ Applications

[redacted]

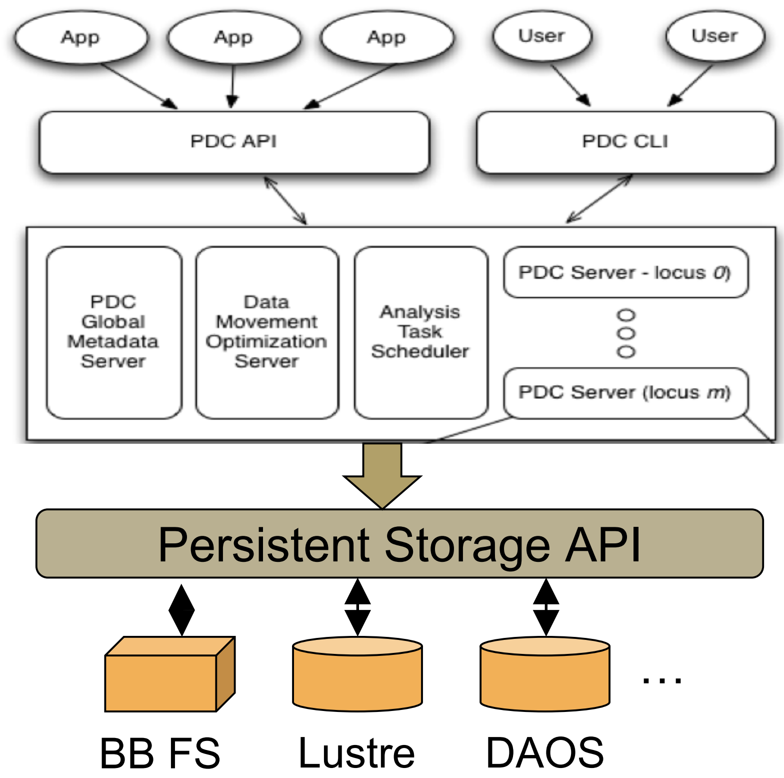
Usage

Applications

Data (in memory)

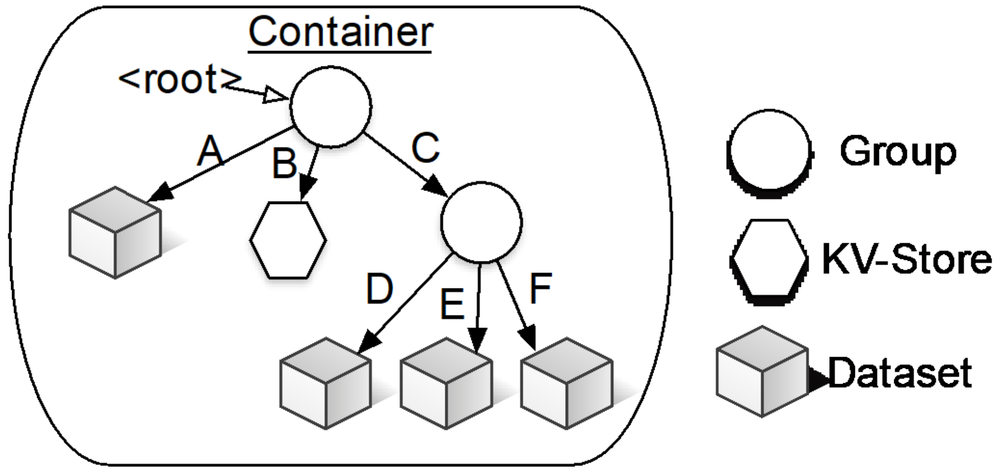
PDC System – High-level Architecture

- **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

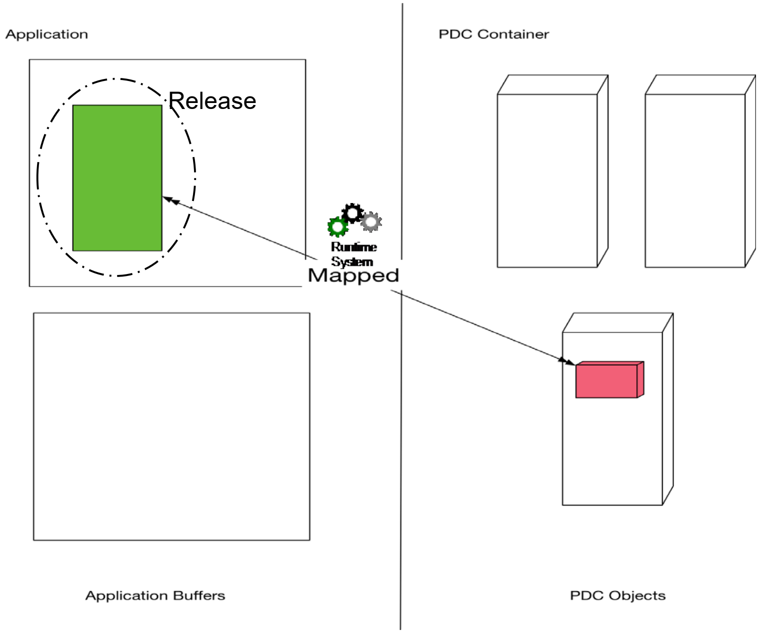


Proactive Data Container

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

Object-centric PDC Interface

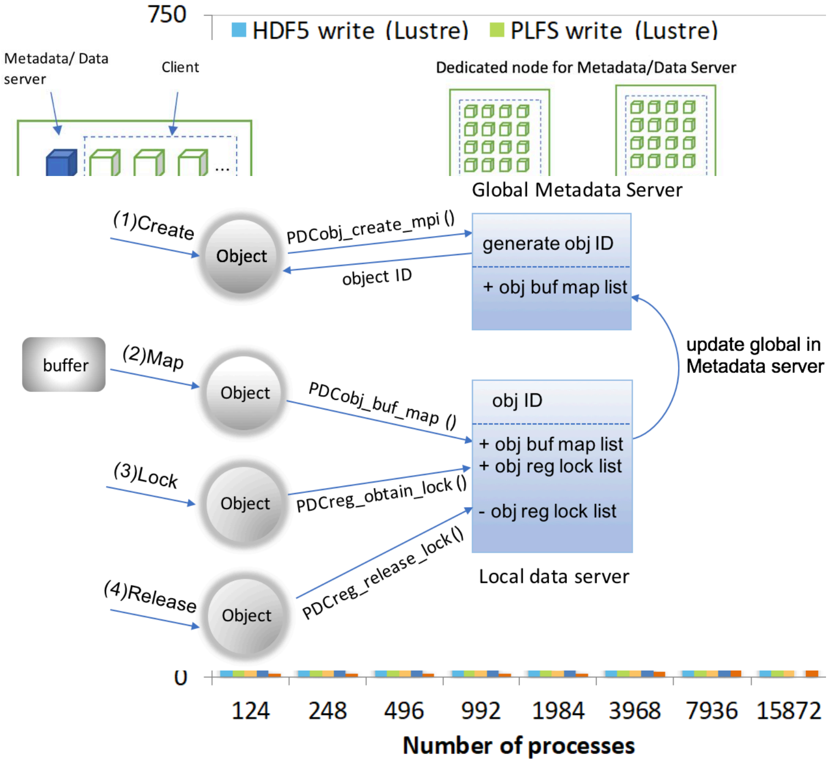
- Object-level interface
  - Create – containers and objects
  - Add attributes
  - Put object
  - Get object
  - Delete object
- Array-specific interface
  - Create regions
  - Map regions in PDC objects
  - Lock
  - Release



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

Transparent data movement in storage hierarchy

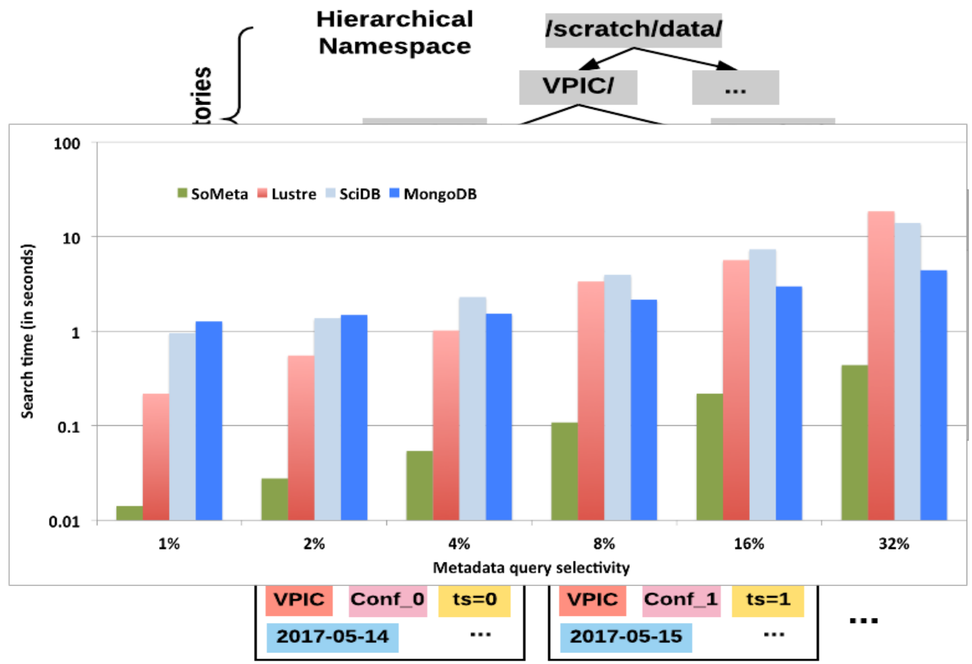
- Usage of compute resources for I/O
  - Shared mode – Compute nodes are shared between applications and I/O services
  - Dedicated mode – I/O services on separate nodes
- Transparent data movement by PDC servers
  - Apps map data buffers to objects and PDC servers place and manage data
  - Apps query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

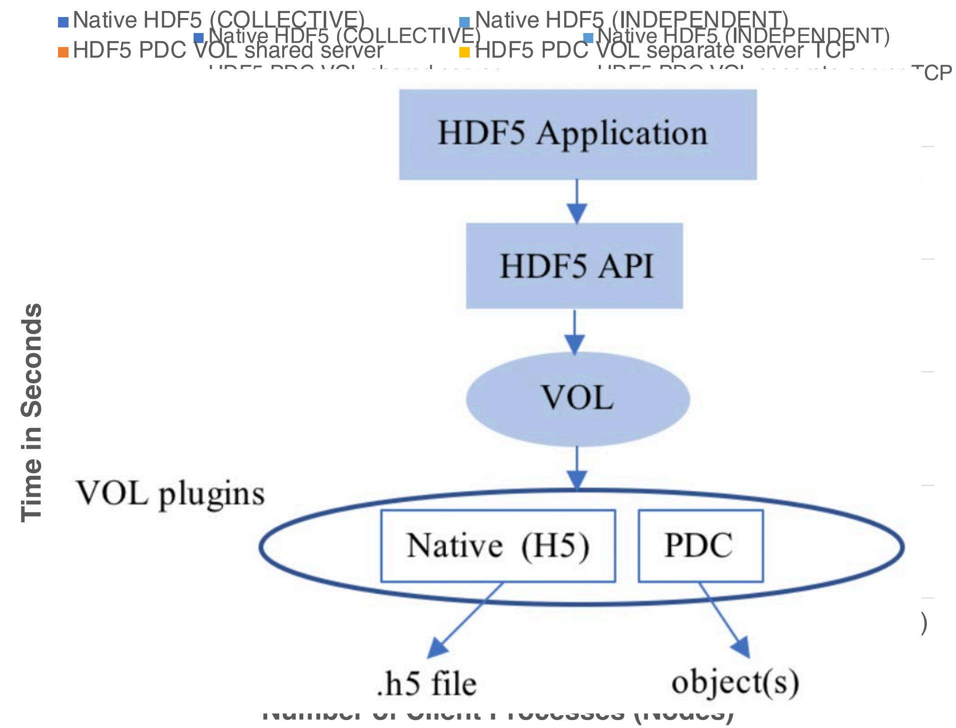
- Flat name space
- Rich metadata
  - Pre-defined tags that includes provenance
  - User-defined tags for capturing relationships between data objects
- Distributed in memory metadata management
  - Distributed hash table and bloom filters used for faster access



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



Collaborators: THG

Conclusions

Easy interfaces and superior performance

- Simpler object interface
  - Applications produce data objects and declare to keep them persistent
  - Applications request for desired data

Autonomous data management

- Asynchronous and autonomous data movement
- Bring interesting data to apps

Information capture and management

- Manage rich metadata and enhance search capabilities
- Perform analysis and transformations in the data path

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