BERKELEY LAB
LAWRENCE BERKELEY NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

ExaHDF5 - ECP ST Project

Delivering Efficient Parallel I/O on Exascale Systems

Presenters

Quincey Koziol and Suren Byna

Project Collaborators

Lawrence Berkeley Lab

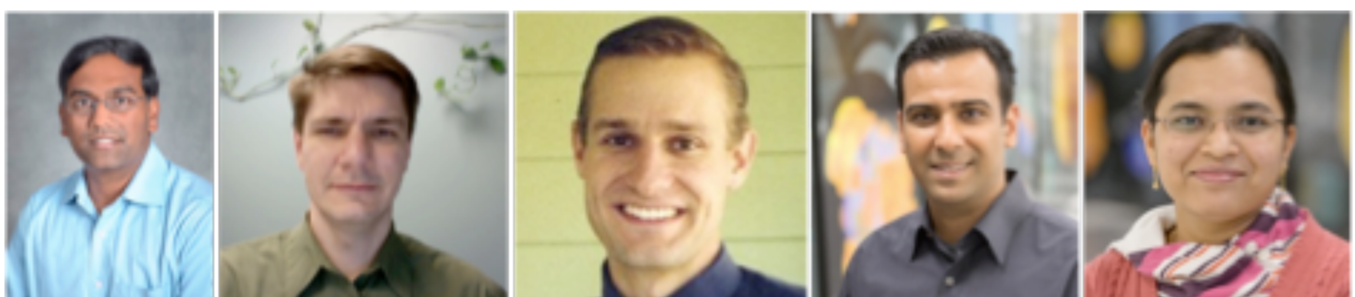
The HDF Group

Argonne National Lab

ExaHDF5 Team

PI Name	Affiliation and Project Role
Suren Byna	LBNL, Project Lead
Quincey Koziol	LBNL, Software development lead
Scot Breitenfeld	THG, SW Integration and release lead
Venkat Vishwanath	ANL, Integration and collaboration lead
Preeti Malakar	ANL, Data movement optimization

Staff: Houjun Tang, Bin Dong, Junmin Gu, Jialin Liu, Alex Sim, Paul Coffman, Todd Munson, Jerome Soumagne, Dana Robinson, and John Mainzer

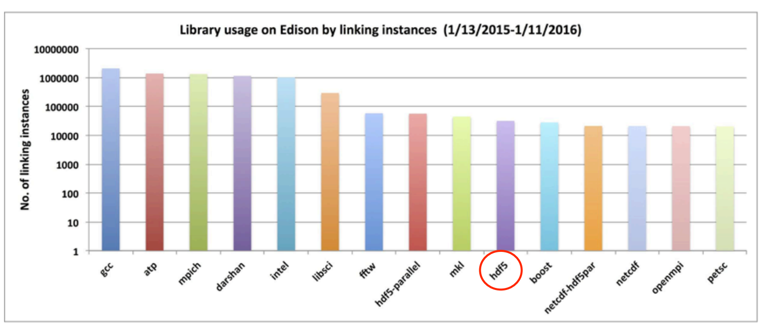


Maturity and usage of HDF5

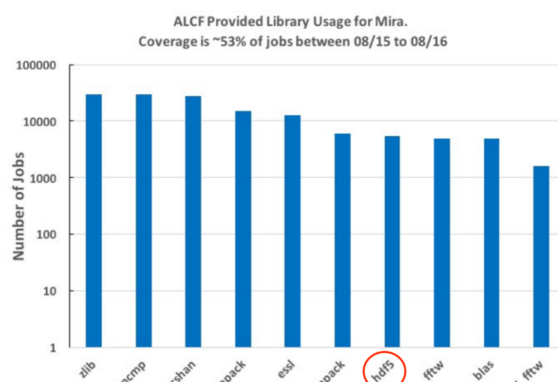
- Satisfies major requirements of contemporary scientific data management:
  - Open Source, Portable, self-describing, longevity / preservation, support for domain-specific data models, provenance
- NASA satellite data (Terra, Aqua, Aura, etc.)
  - **Highest Technology Readiness Level (TRL 9)**
  - “Flight proven” through successful mission operations

2002 R&D 100 Award Winner

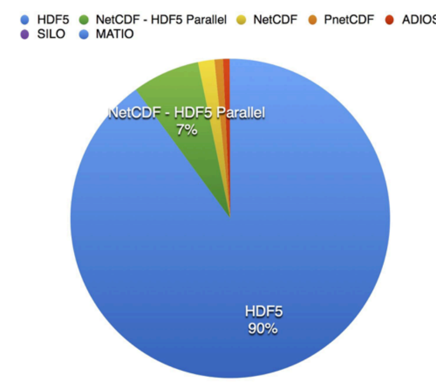
- **Top I/O library at NERSC and LCFs**



Number of linking instances on Edison (NERSC)

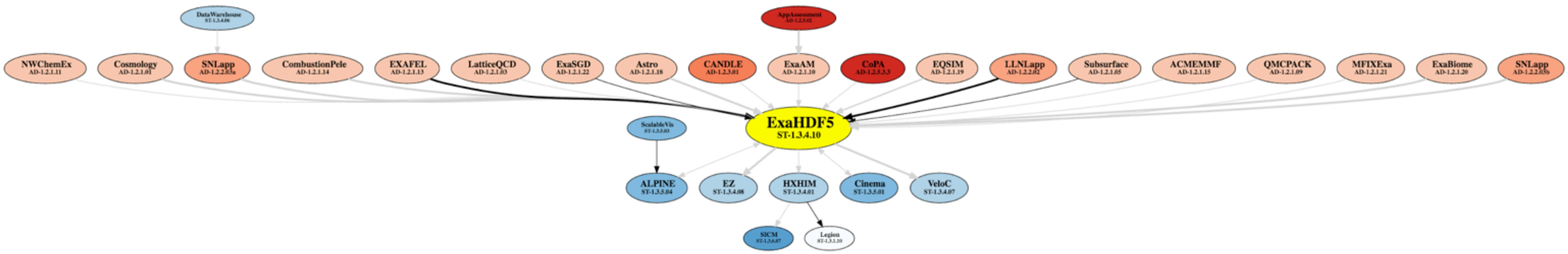


Number of linking instances on Mira (ALCF)

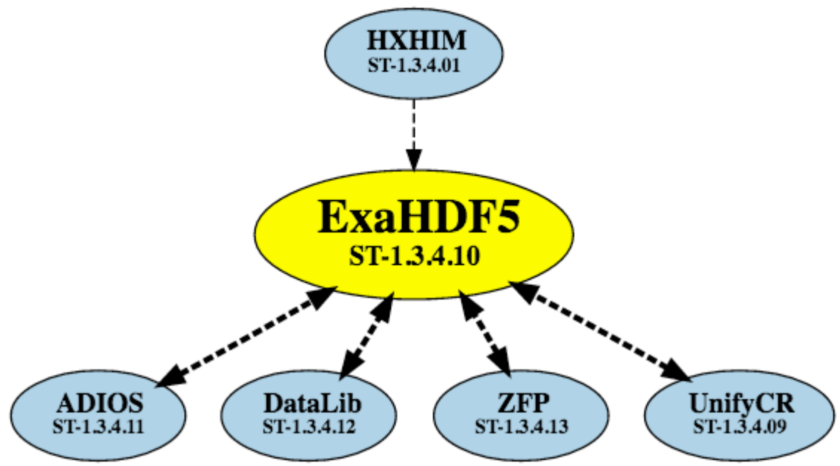


I/O library usage on Titan (OLCF)

HDF5 in ECP Apps



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



ExaHDF5 Mission

- Work with ECP applications to meet their needs
- Productize HDF5 features
- Support, maintain, package, and release HDF5
- Research toward future architectures and incoming requests from ECP teams

Outline

- HDF5 features to be developed in ECP ExaHDF5
- Timeline
- EOD-HDF5 - Features specific for EOD
- Looking further ahead

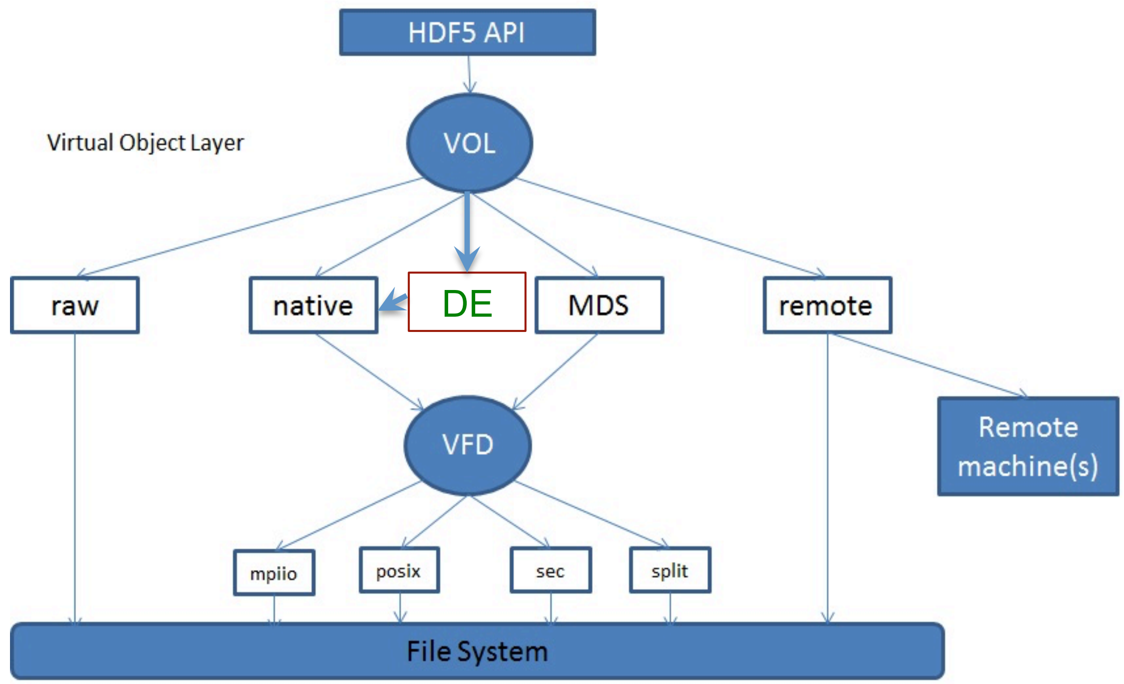
ExaHDF5 – Features

- Virtual Object Layer (VOL)
  - Abstraction layer within HDF5, similar to PMPI layer
  - Allows interception of HDF5 calls at runtime, to access data in alternate ways
- Caching and prefetching
  - Data Elevator for moving data efficiently among storage layers
- Topology-aware I/O
  - Select data movement optimizations based on topology
  - Topology-aware I/O API and HDF5 VOL based on Open Fabrics
- Support Advanced Workflows
  - Full Single Writer – Multiple Reader (SWMR)
  - Design Parallel SWMR

HDF5 Virtual Object Layer

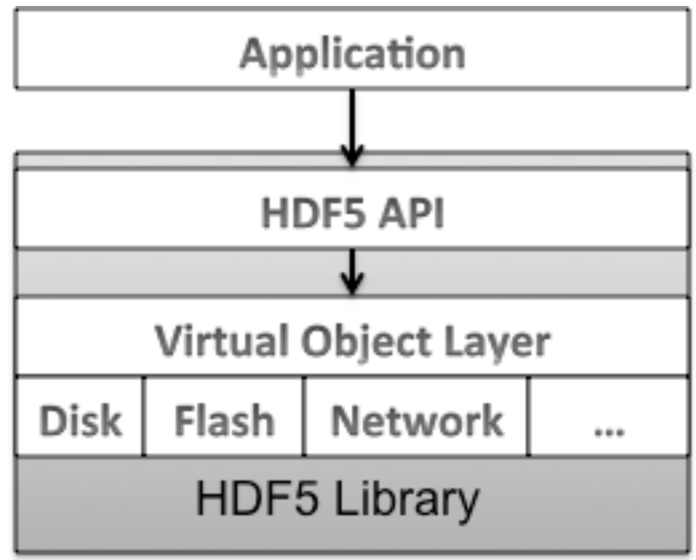
- An abstraction layer for plugins to access data on the file system

- Allows interception of HDF5 calls at runtime



Virtual Object Layer

- **Objectives**: Abstract HDF5 object storage;
  Enables developers to easily use HDF5 on novel current and future storage systems
- **Accomplishments:**
  - Implemented object-oriented framework allowing user-defined plugins to efficiently store and access HDF5 objects in arbitrary storage methods and formats
  - Developed plugins for both classic HDF5 file format and new split metadata/raw data files; which removed scalability limitations for HDF5 metadata operations
  - Collaborated with developers at LANL, CSCS and other organizations to develop plugins for distributed shared memory and PLFS storage methods, without modifying HDF5 application

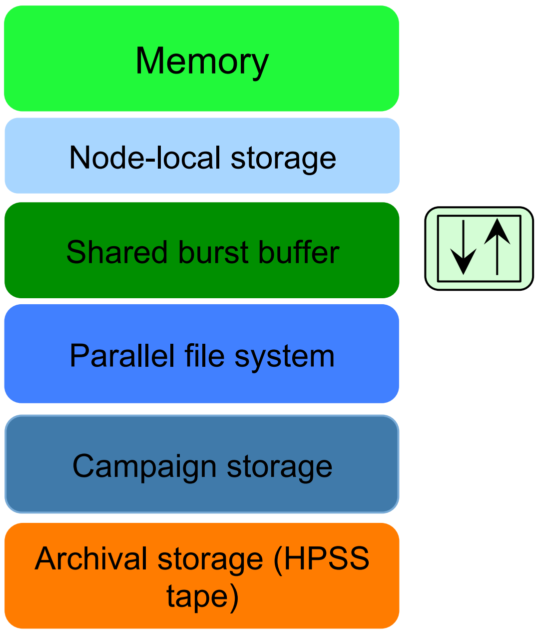


- **Impact:**
  - Allows all HDF5 applications to migrate to future storage systems and mechanisms with no source code modifications
  - **Enables DOE Exascale Storage FastForward project** to create plugin to access prototype exascale storage system with minimal effort and HDF5 applications to run without modifications in that environment

ExaHDF5 – Features

- Virtual Object Layer (VOL)
  - Abstraction layer within HDF5, similar to PMPI layer
  - Allows interception of HDF5 calls at runtime, to access data in alternate ways
- Caching and prefetching
  - Data Elevator for moving data efficiently among storage layers
- Topology-aware I/O
  - Select data movement optimizations based on topology
  - Topology-aware I/O API and HDF5 VOL based on Open Fabrics
- Support Advanced Workflows
  - Full Single Writer – Multiple Reader (SWMR)
  - Design Parallel SWMR

Data Elevator for moving data



- **Contributions**
  - Low-contention data movement library for hierarchical storage systems
  - Offload of data movement task to a few compute nodes or cores
  - Data Elevator on NERSC's Cori system
    - With a couple of science applications, demonstrated that Data Elevator is **4X** faster than Cray DataWarp ***stage_out*** and **4X** faster than writing data to parallel file system
- **Benefits of using Data Elevator**
  - **Transparent data movement**: Applications using **HDF5** specify destination of data file and the Data Elevator transparently moves data from a source to the destination
  - **Efficiency**: Data Elevator reduces contention on BB
  - **In transit analysis**: While data is in a faster storage layer, analysis can be done in the data path

Data Elevator functionality

Start Data Elevator along with an application

DE intercepts HDF5 calls at run time using VOL

...

H5Fopen (..) → Sets up staging area, and give file handle to app
Records the destination in a Berkeley DB table

...

H5Dwrite (..) → Writes the data to staging area
Updates the status

...

H5Fclose (..) → Staging area ready for use
(analysis, transfer, etc.)
Application can continue with computation

...

MPI_Finalize () → Check staging area status
Move any remaining data

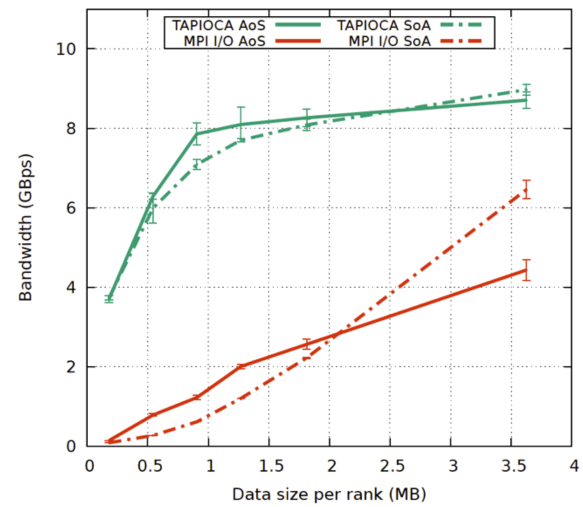
Available on Cori → *module load data-elevator*

ExaHDF5 – Features

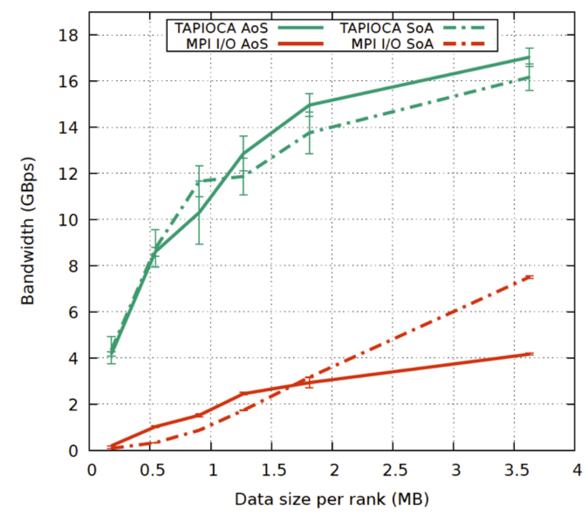
- Virtual Object Layer (VOL)
  - Abstraction layer within HDF5, similar to PMPI layer
  - Allows interception of HDF5 calls at runtime, to access data in alternate ways
- Caching and prefetching
  - Data Elevator for moving data efficiently among storage layers
- Topology-aware I/O
  - Select data movement optimizations based on topology
  - Topology-aware I/O API and HDF5 VOL based on Open Fabrics
- Support Advanced Workflows
  - Full Single Writer – Multiple Reader (SWMR)
  - Design Parallel SWMR

Topology-aware I/O optimizations

- Data aggregation algorithm based on the two-phase I/O scheme
  – Aggregators placement considering topology and data access pattern
- Optimizations:
  – Double-buffering
  – RMA operation - using non-blocking MPI one-sided communication



1K nodes (16 ranks per node)



2K nodes (16 ranks per node)

F. Tessier, V. Vishwanath, E. Jeannot - **TAPIOCA: An I/O Library for Optimized Topology-Aware Data Aggregation on Large-Scale Supercomputers** - IEEE Cluster 2017

ExaHDF5 – Features

- Virtual Object Layer (VOL)
  - Abstraction layer within HDF5, similar to PMPI layer
  - Allows interception of HDF5 calls at runtime, to access data in alternate ways
- Caching and prefetching
  - Data Elevator for moving data efficiently among storage layers
- Topology-aware I/O
  - Select data movement optimizations based on topology
  - Topology-aware I/O API and HDF5 VOL based on Open Fabrics
- Support Advanced Workflows
  - Full Single Writer – Multiple Reader (SWMR)
  - Design Parallel SWMR

Full *SWMR*

- Single-Writer / Multiple-Reader (SWMR) allows
  - Concurrent access to HDF5 file by a single writing process and many readers
  - High-performance, lock-free updates
  - Changes to HDF5 files can be streamed to remote locations, enabling super-facility solutions
  - Moves HDF5 containers closer to “file system in a file”
  - Serial only, currently
    - ECP project includes funding for parallel SWMR design though…

ExaHDF5 – Production Features

- Asynchronous I/O
  - Support for asynchronous I/O operations in HDF5 (serial only)
- Independent metadata updates for parallel HDF5
  - Metadata updates currently require collective operations
  - Break the collective dependencies in updating metadata
- Querying HDF5 Files - Data and Metadata
  - Basic implementation of querying data is available
  - Integrating indexing and querying into HDF5
  - Adding metadata querying feature
- Interoperability with other file formats
  - Capability to read netCDF/PnetCDF and ADIOS files, using VOL

Asynchronous I/O

- Asynchronous I/O for HDF5 allows
  - Application to queue operations on an HDF5 file, then check back later for completion
  - Uses "event set" object that holds many operations, instead of tokens on single operations
    - For ease of use and to preserve dependencies
      - H5Fopen → H5Gcreate → H5Dcreate → H5Dwrite
  - Applications can overlap compute, communication, and I/O
    - The "trifecta" of high-performance computing: use the *entire* system simultaneously

ExaHDF5 – More Features

- Asynchronous I/O
  - Support for asynchronous I/O operations in HDF5 (serial only)
- Independent metadata updates for parallel HDF5
  - Metadata updates currently require collective operations
  - Break the collective dependencies in updating metadata
- Querying HDF5 Files - Data and Metadata
  - Basic implementation of querying data is available
  - Integrating indexing and querying into HDF5
  - Adding metadata querying feature
- Interoperability with other file formats
  - Capability to read netCDF/PnetCDF and ADIOS files, using VOL

Independent Metadata Updates

- Independent Metadata Updates (IMU) allow any MPI process to modify the structure of an HDF5 file
- IMU addresses the “all collective metadata” limit on parallel HDF5 files
  - Currently, any operation that modifies metadata in an HDF5 file must be done collectively
- Moves even closer to “file system in a file” for HDF5 containers

ExaHDF5 – More Features

- Asynchronous I/O
  - Support for asynchronous I/O operations in HDF5 (serial only)
- Independent metadata updates for parallel HDF5
  - Metadata updates currently require collective operations
  - Break the collective dependencies in updating metadata
- Querying HDF5 Files - Data and Metadata
  - Basic implementation of querying data is available
  - Integrating indexing and querying into HDF5
  - Adding metadata querying feature
- Interoperability with other file formats
  - Capability to read netCDF/PnetCDF and ADIOS files, using VOL

Querying HDF5 Data and Metadata

- Application queries into HDF5 containers:
  - Link / attribute name
  - Dataspace dimensionality / size
  - Datatype choice
  - Dataset / attribute element value / range
- "Programmatic", not "text-based"
  - e.g. "H5Qdefine(qid, H5Q_LESSTHAN, type_id, &52);"
- Pluggable interface for third-party index modules
  - Optional, but used to accelerate queries when available / appropriate
- Queries return "views"
  - Temporary groups in the HDF5 file that contain datasets with the actual query results

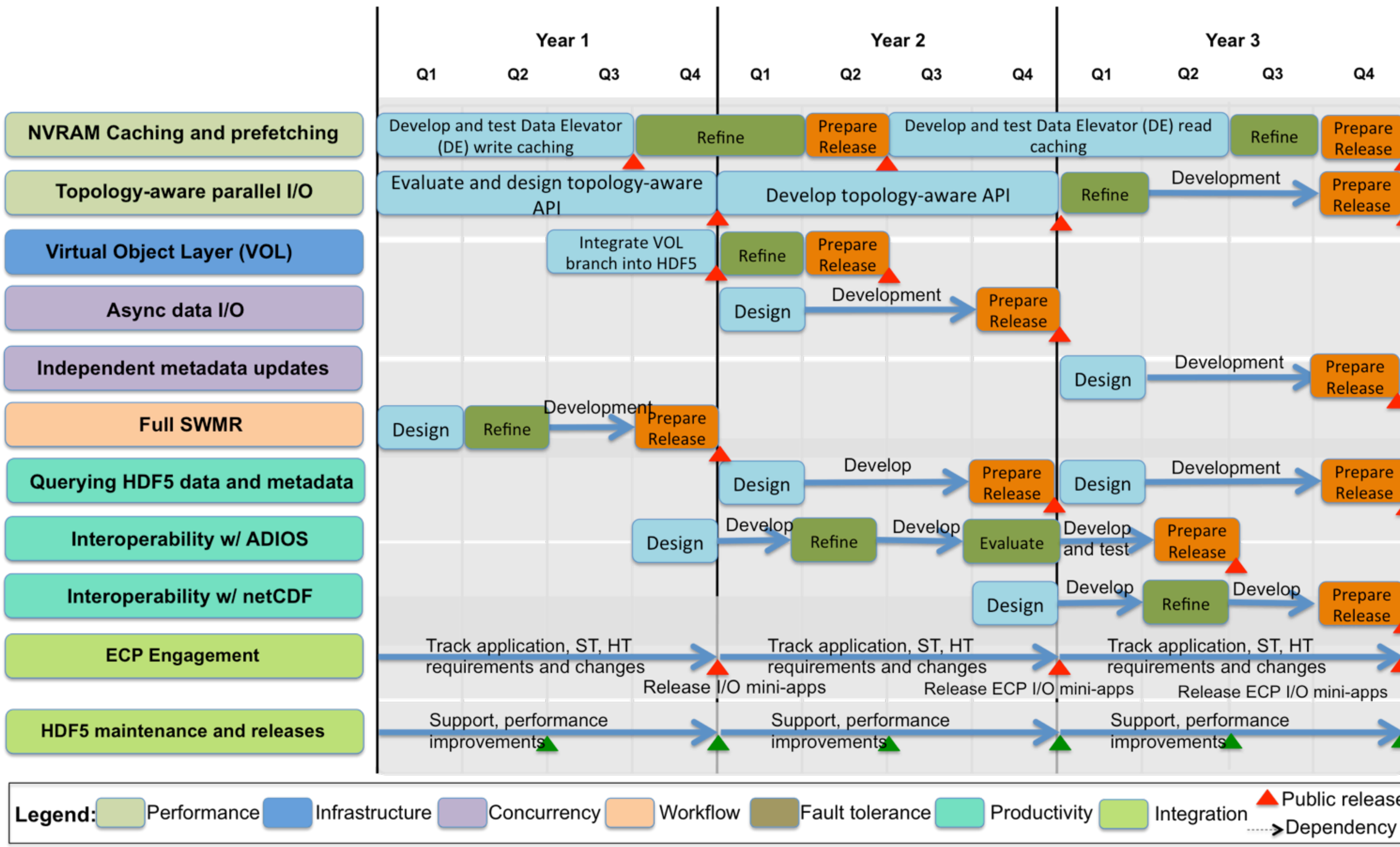
ExaHDF5 – More Features

- Asynchronous I/O
  - Support for asynchronous I/O operations in HDF5 (serial only)
- Independent metadata updates for parallel HDF5
  - Metadata updates currently require collective operations
  - Break the collective dependencies in updating metadata
- Querying HDF5 Files - Data and Metadata
  - Basic implementation of querying data is available
  - Integrating indexing and querying into HDF5
  - Adding metadata querying feature
- Interoperability with other file formats
  - Capability to read netCDF/PnetCDF and ADIOS files, using VOL

Interoperability w/ Other File Formats

- Virtual Object Layer (VOL) allows intercepting HDF5 API and accessing data in alternate ways, including other file formats
- ExaHDF5 feature enables expanding the HDF5 API to access other file formats
  - netCDF/PnetCDF, ADIOS, etc.
- Intercept HDF5 Read API calls using VOL
  - Redirect the calls to read data from other formats

ExaHDF5 – Development timeline



Contact:

Quincey Koziol (koziol@lbl.gov)

Suren Byna (sbyna@lbl.gov)

Thanks!