The HDF Dataverse

#SC17 Birds of a Feather

Part 1: HDF5 STATE OF THE UNION

State of the Union

Dave Pearah, CEO, The HDF Group
Elena Pourmal, Client Management Director and Interim Engineering Director, The HDF Group
John Mainzer, Principal Architect, The HDF Group

ECP Exascale, Big Data Initiatives

Quincey Koziol, Principal Data Architect, National Energy Research Scientific Computing Center (NERSC)

Part 2: USER LIGHTING TALKS

Andreas Dilger, Intel
Sean Ziegeler, Engility
Brian van Straalen, Lawrence Berkeley National Laboratory

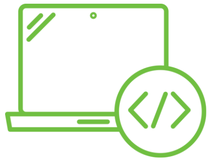
The HDF Group

Who is the HDF Group?

The HDF Group

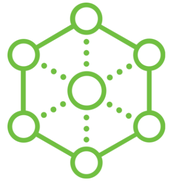
"De-facto standard for scientific computing" and integrated into every major scientific analytics + visualization tool

HDF Group has developed open source solutions for Big Data challenges for over 31 years



Small not-for-profit company (~ 40 employees) with focus on High Performance Computing and Scientific Data

Headquarters in Champaign, IL



Our flagship platform – HDF5 – is the heart of our open source ecosystem.

Thousands use + build on HDF5 every day (983 projects on Github)

What does the HDF Group do?

The HDF Group

Products

- HDF5 Community Edition (Open Source)
- HDF5 Enterprise Apps: Spark Connector, ODBC Connector, S3 Connector, Compression Library, Binaries, etc.
- HDF Cloud (Beta)

Consulting

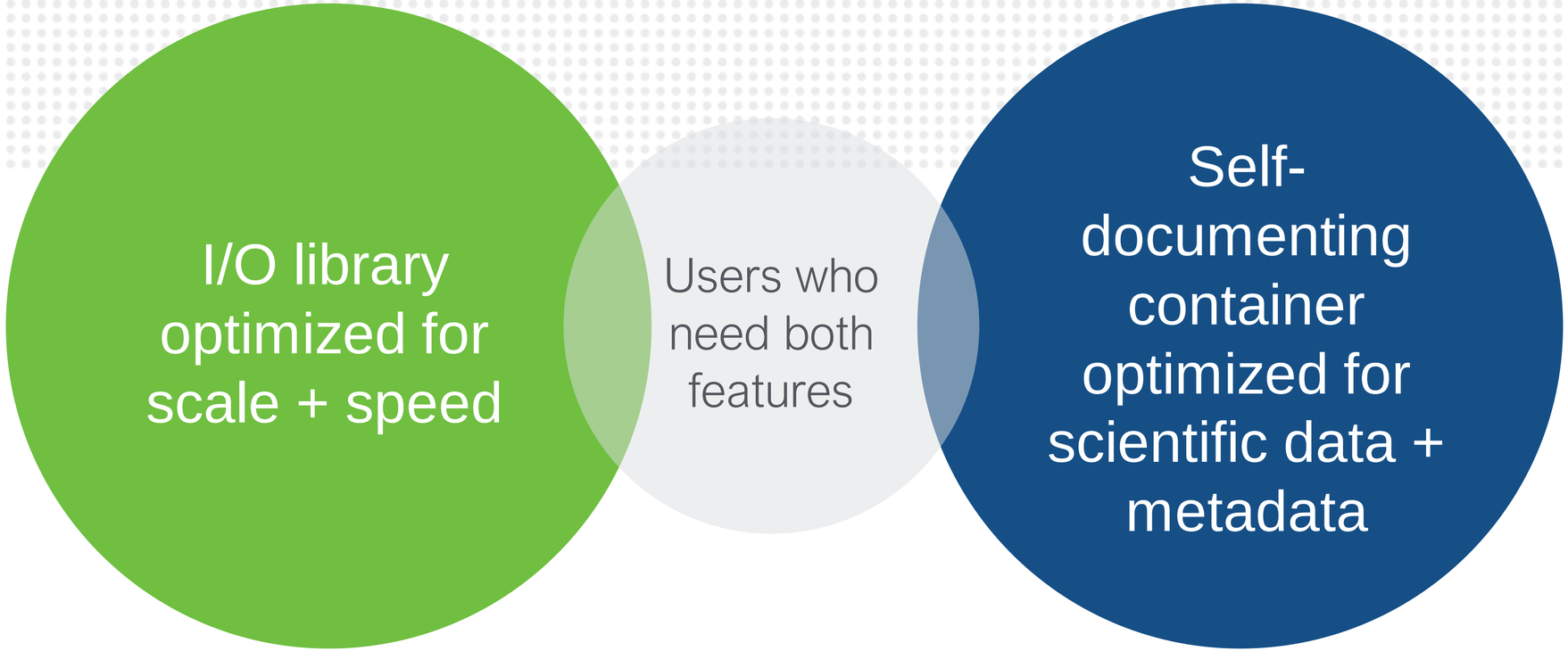
- HDF: new functionality + performance tuning
- General HPC software engineering with scientific expertise
- Metadata science and standards services

Support

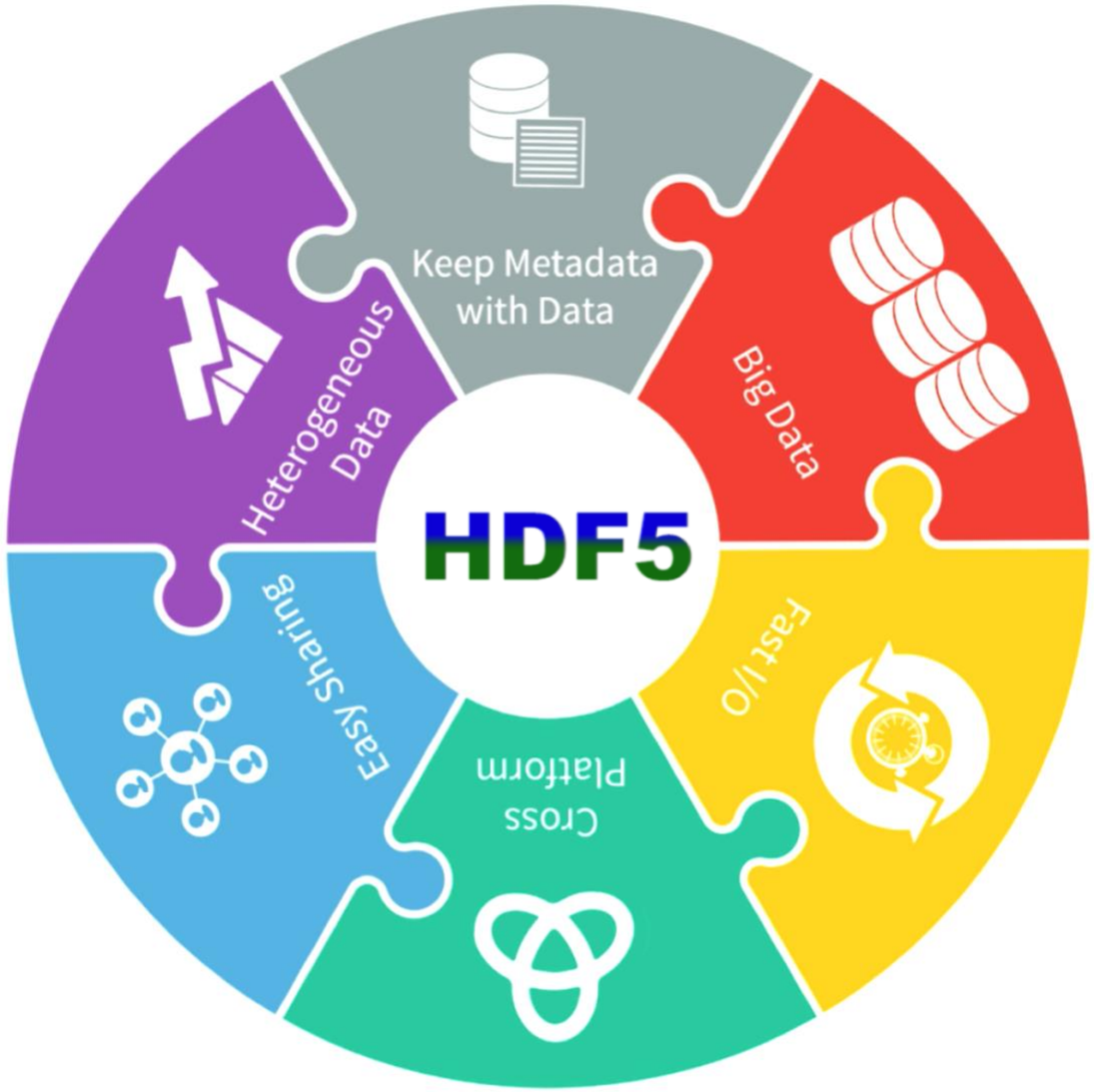
- HDF Support Packages (Basic + Pro + Premier)
- Python support for h5py + PyTables + pandas
- Training

A few of our users

Why Use HDF5?


I/O library optimized for scale + speed
Users who need both features
Self-documenting container optimized for scientific data + metadata

Why is this concept so different + useful?



- **Native support for multidimensional data**
- **Data and metadata in one place => streamlines data lifecycle & pipelines**
- Portable, no vendor lock-in
- Maintains logical view while adapting to storage context
- In-memory, over-the-wire, on-disk, parallel FS, object store
- Pluggable filter pipeline for compression, checksum, encryption, etc.
- High-performance I/O
- Large ecosystem

HDF Strategy

Exploration

HDF5 "Database"

Analytics (Python)

Deep Learning

Existing HDF5 Users

HDF5 Enterprise Apps

- Compression
- Binaries
- Spark Connect
- S3 Connect
- ODBC Connect
- Etc…

HDF Cloud

- Web service + Object Storage
- 1 HDF5 file ➔ multiple objects
- Data caching
- REST API (h5py)
- SDK compatibility

Support

- Basic
- Pro
- Premier

HDF5 Community Edition

[Open Source + Free + Registration Required]

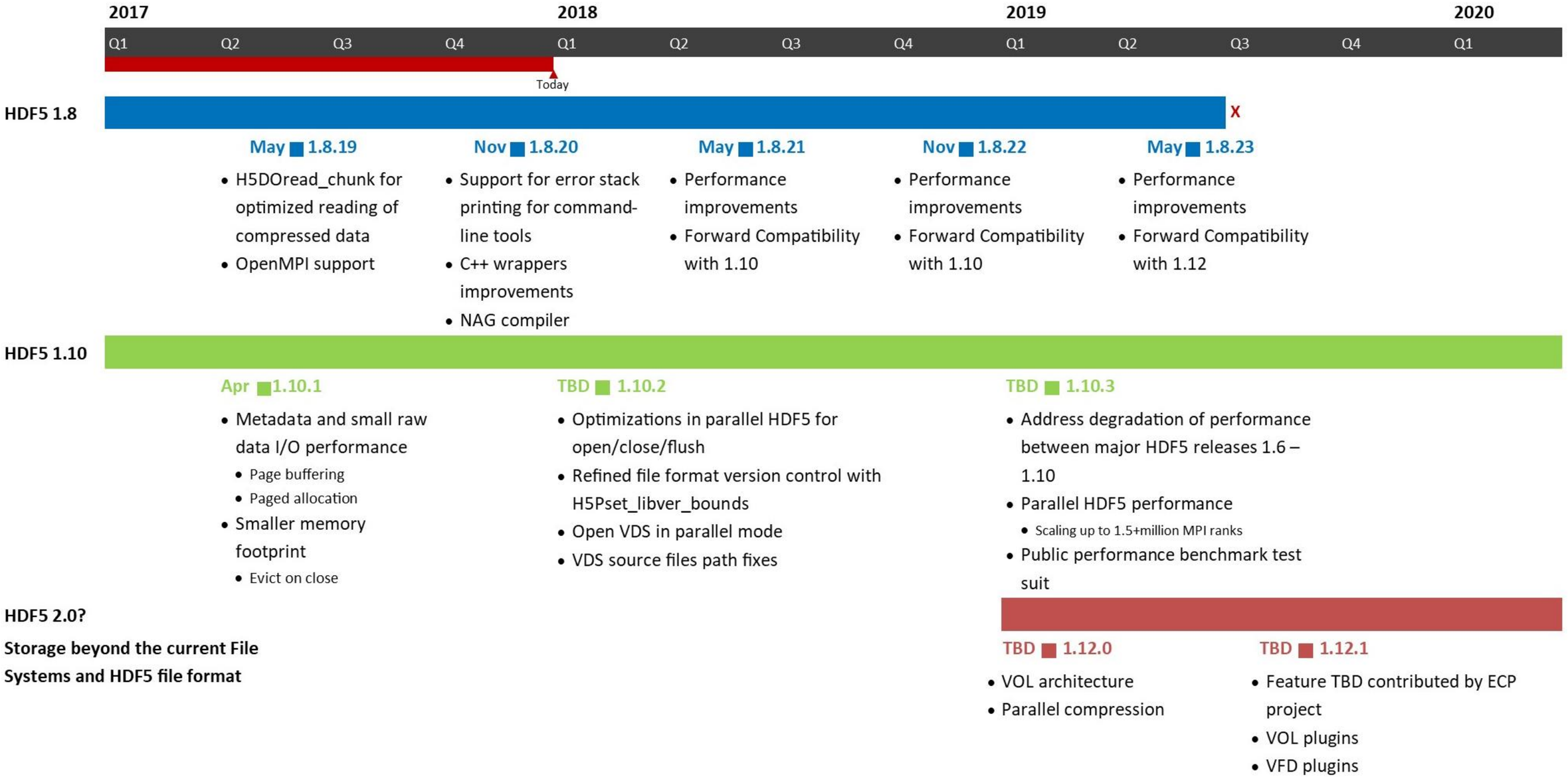
Vendor Partner Program (1 Year Update)

- Launched last year at SC16

- Key Partner Services
  - Optimized and customized HDF solutions for target platforms and technologies: HDF5 binaries, HDF Cloud, Object Storage connectors, etc.
  - Support platform vendors and their end users: premium support backed by HDF Group
  - Field sales assistance: private-label team that can build custom and semi-custom apps to support your own sales initiatives
  - Co-marketing: showcase platform benchmarks (e.g. STAC M3 in Fintech)

HDF5 Roadmap

The HDF Group



Community Involvement

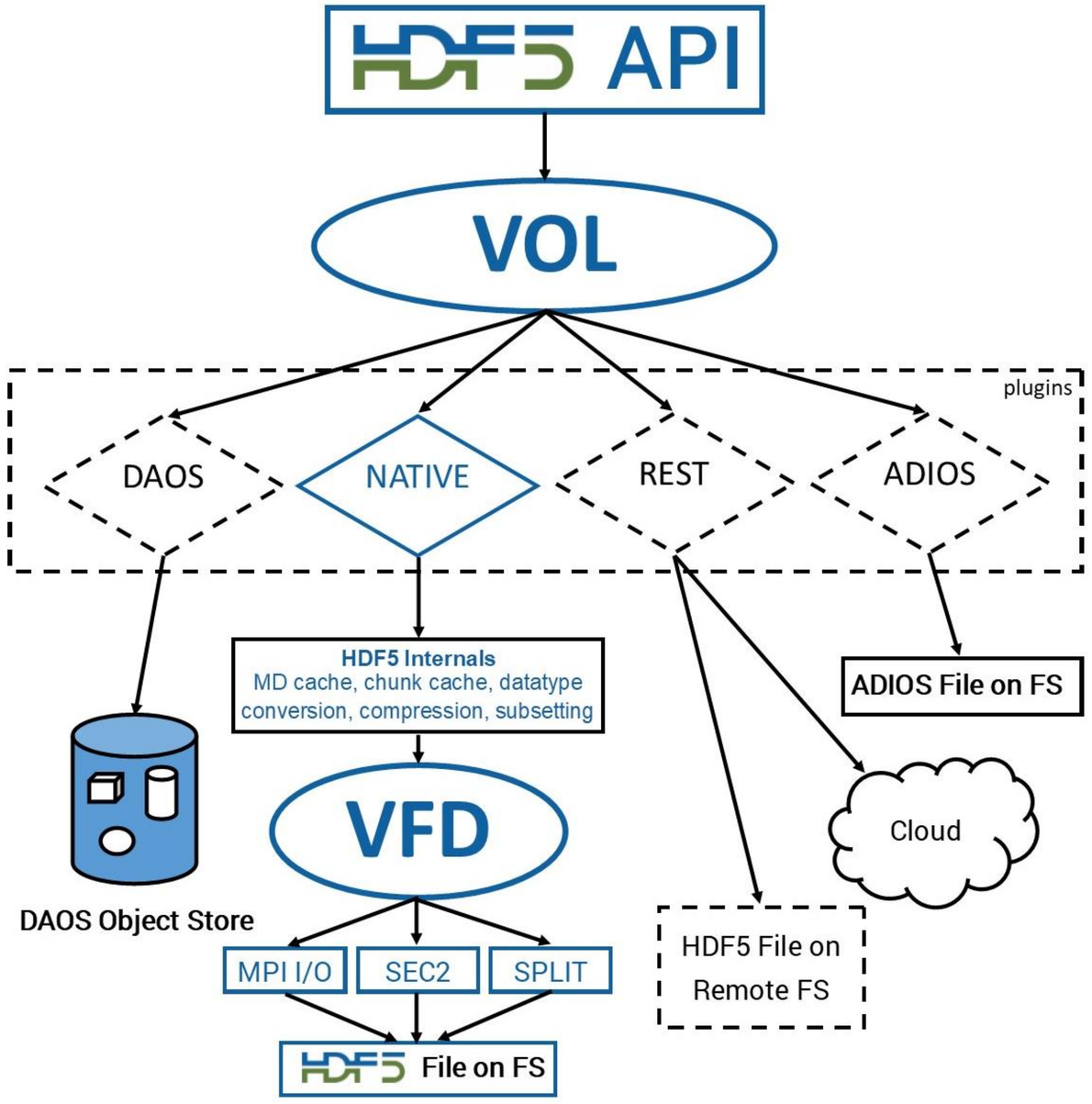
- **The HDF Group is very grateful for your:**
  - Participation in releases and features testing
  - Issues reporting, improvement suggestions
  - Expertise sharing
- **We accept contributions**
  - Send your patch (code, test code, documentation changes) to help@hdfgroup.org
- **We are working on lowering the barrier for contributions**
  - Public JIRA
  - Simplified contribution process
  - Developers documentation
  - Coding standards

Future HDF5 Enhancements

- **Storage versatility using VFD and VOL mechanisms**
- **Scalability**
  - Target 1.5M + MPI ranks
  - Use page buffering and metadata paged allocation
- **Data Model Extensions – possible directions**
  - Sparse storage
  - Key-Value
  - Column-Oriented

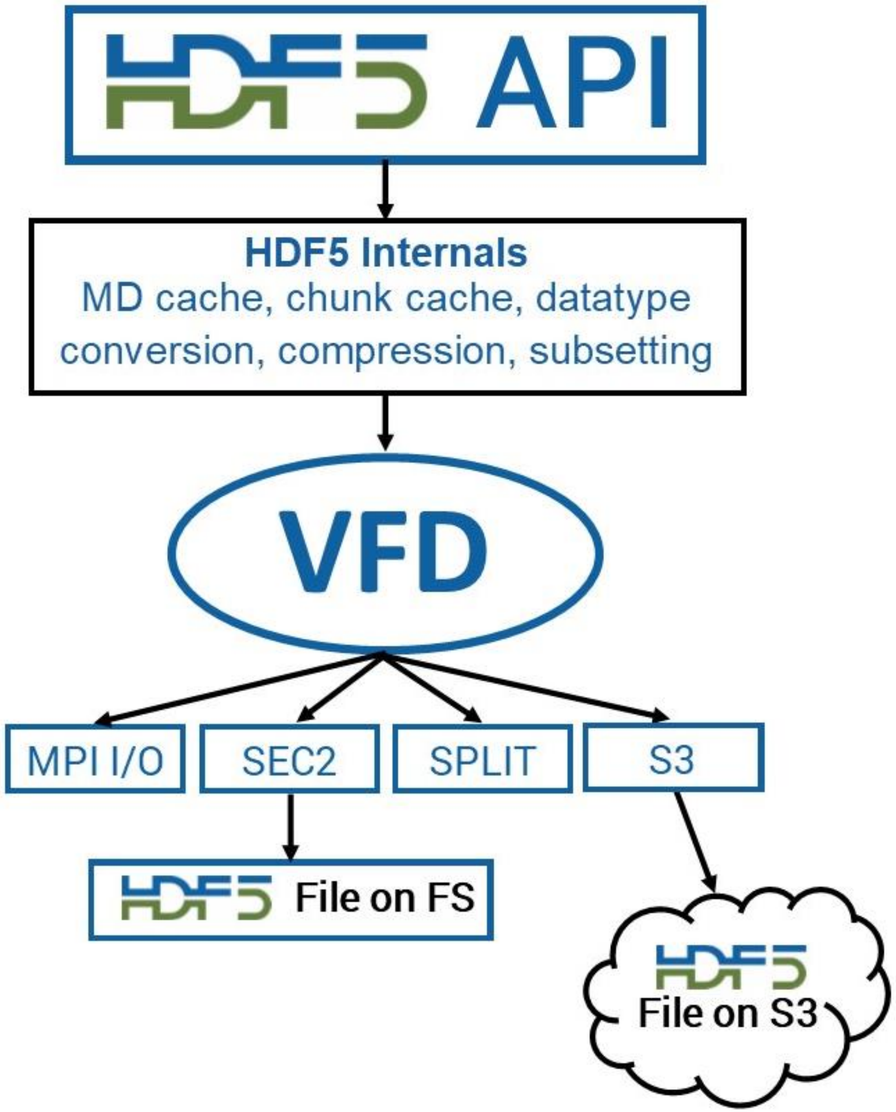
HDF5 Virtual Object Layer (VOL)

- **Goal**
  - Provide an application with the HDF5 data model and API, but allow different underlying storage mechanisms
- **New layer below HDF5 API layer**
  - Intercepts all API calls that potentially could touch the data on disk and route them to a Virtual Object Driver
- **Potential Object Drivers (or plugins):**
  - Native HDF5 driver (writes to HDF5 file)
  - Raw driver (maps groups to file system directories and datasets to files in directories)
  - Drivers to Object Store (DAOS)
  - Remote driver (the file exists on a remote machine)
  - Drivers to other file formats (netCDF, HDF4, ADIOS, FITS, etc.)
- **VOL properties**
  - Stackable
  - Dynamically loaded



HDF5 S3 VFD

- **Goal**
  - Serve HDF5 files from Object Store
- **Approach**
  - Use existing HDF5 library and new VFD drivers to access the HDF5 file (work in progress)
  - S3 VFD uses range gets to read the desired data from the HDF5 file
  - R/O case
    - Optimization is performed to avoid small metadata accesses
    - Ingestion tool is required to create object with metadata information
  - R/W case takes advantage of paged allocation feature introduced in HDF5 1.10.1
    - VFD tracks allocations for the pages containing metadata and raw data for an object.



HDF5 Scalability

- **Goal**
  - Scale to 1.5+ million MPI ranks
- **Problem on Lustre**
  - System contention when performing small I/O (HDF5 metadata and small raw data)
    - HDF5 metadata (~ few KB) is scattered all over the file
    - A single collective write just before close results in excessive lock contention
    - The MPI_File_set_size() call that is made shortly after the above collective metadata write typically cannot execute until it obtains locks on all OSTs. This requires the call to wait until the above contention is resolved – hence the delays we have been seeing on our truncate calls at file close.
- **Proposed Solution**
  - Implement paged aggregation and page buffering for parallel HDF5

ExaHDF5 Mission

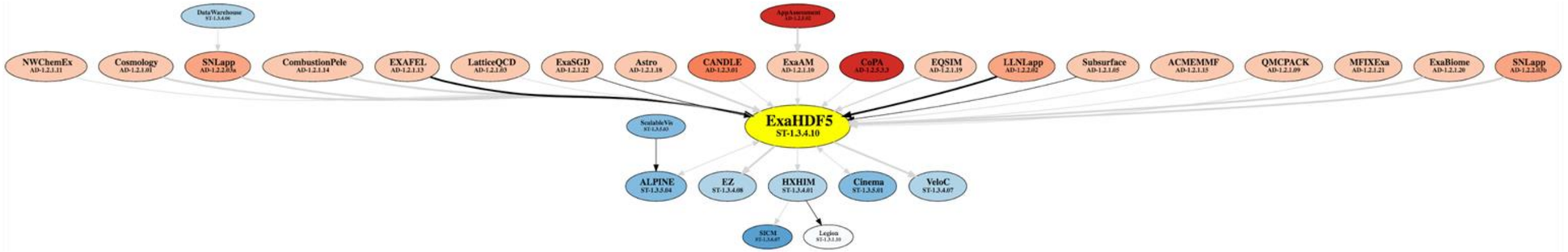
- Work with ECP applications to meet their needs
- Productize HDF5 features
- Support, maintain, package, and release HDF5
- Research toward future architectures and incoming requests from ECP teams

ExaHDF5 Team

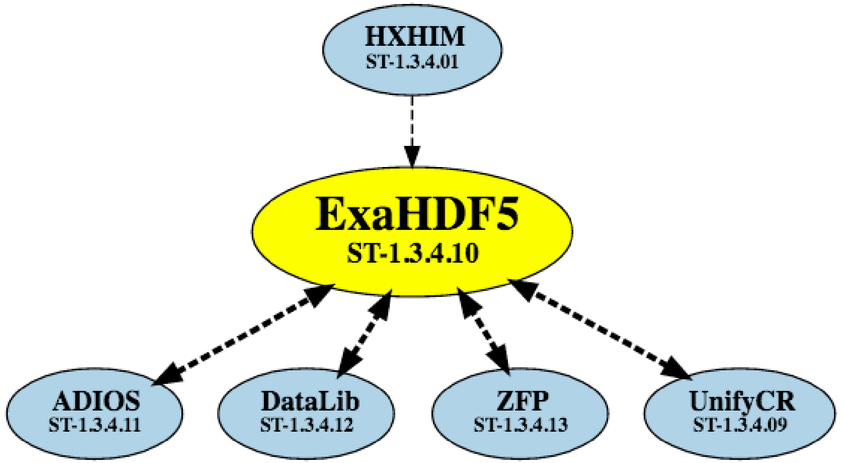
PI Name	Affiliation and Project Role
Suren Byna	LBNL, Project Lead
Quincey Koziol	LBNL, Software development lead
Scot Breitenfeld	THG, SW Integration and release lead
Venkat Vishwanath	ANL, Integration and collaboration lead
Preeti Malakar	ANL, Data movement optimization

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

HDF5 in ECP Apps



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



Outline

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

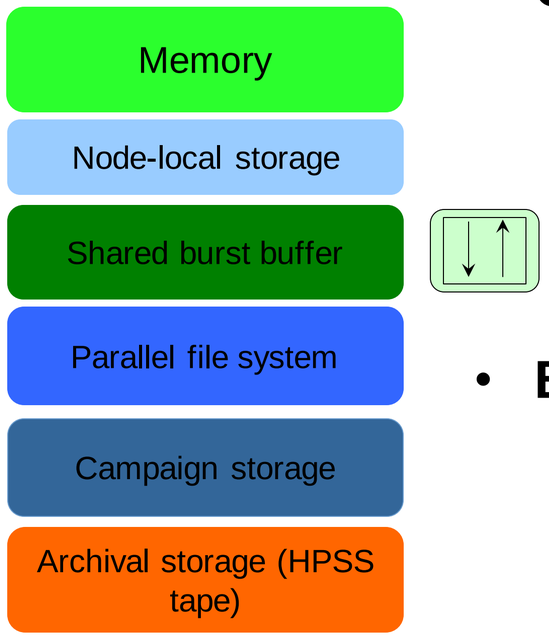
ExaHDF5 – Features

- 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

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



- **Contributions**
  - Low-contention data movement library for hierarchical storage systems
  - Offload data movement task to a few compute nodes or cores
  - DE intercepts HDF5 calls at run time using VOL
- **Benefits of using Data Elevator**
  - **Performance**: Have demonstrated that Data Elevator is 4x faster than Cray DataWarp stage_out and 4x faster than writing data to parallel file system, with several science applications on NERSC's Cori system
  - **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

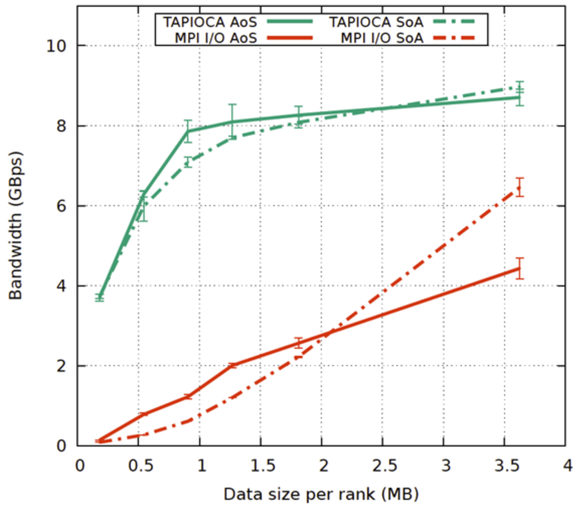
Available on Cori → *module load data-elevator*

ExaHDF5 – Features

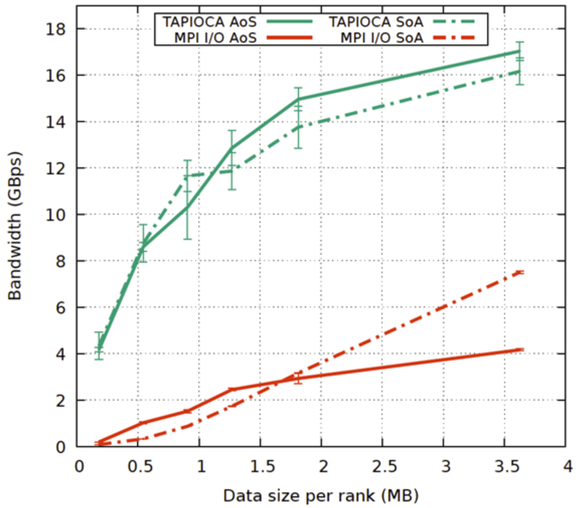
- Virtual Object Layer (VOL)
  - Abstraction layer within HDF5, similar to PMPI layer
  - Allows interception of HDF5 calls at runtime, to access data in alternate ways
- Caching and prefetching
  - Data Elevator for moving data efficiently among storage layers
- Topology-aware I/O
  - Select data movement 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 then 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 ☹
  - Fix the collective dependency 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

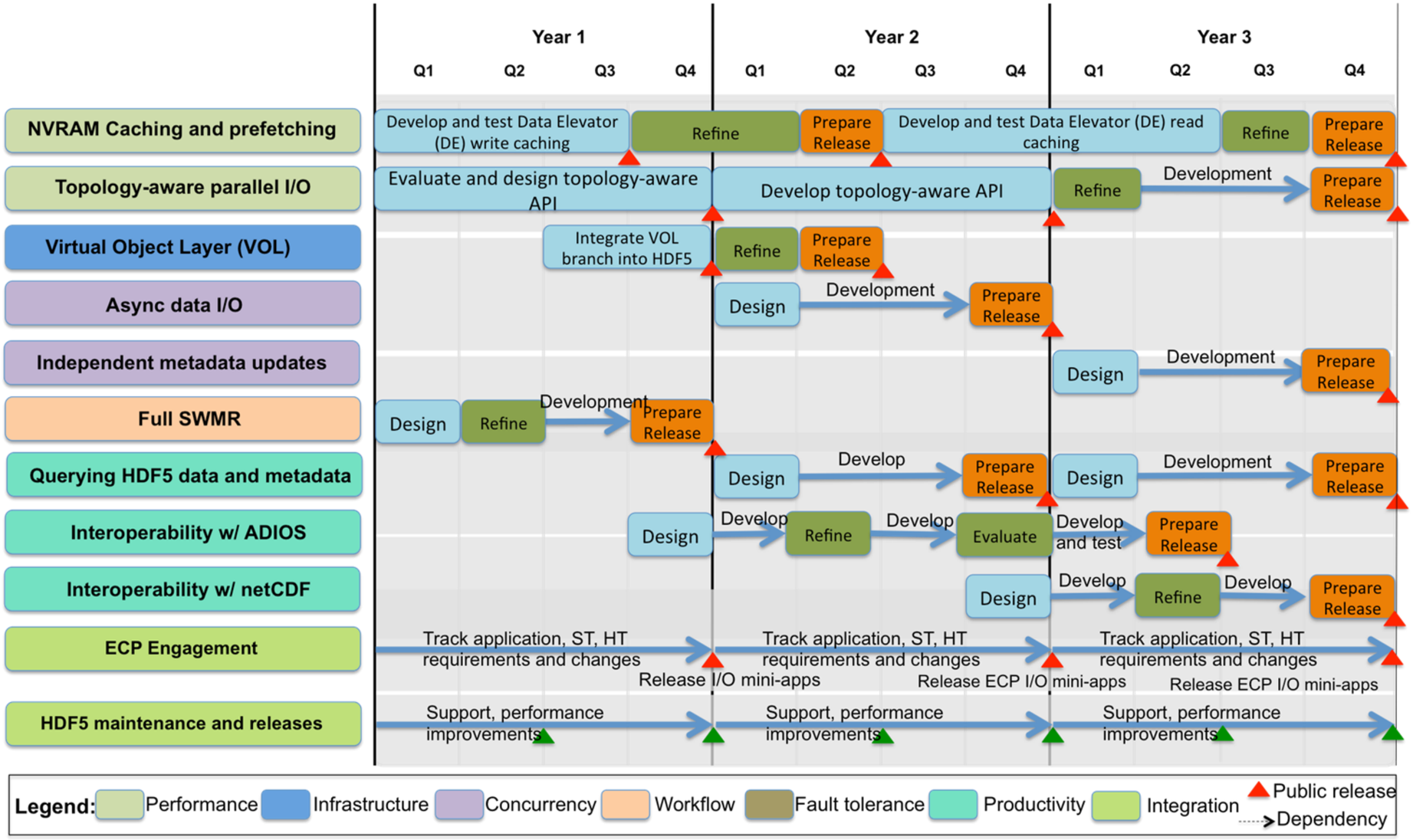
ExaHDF5 – More Features

- 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



Experimental & Observational Data (EOD) Management Requirements

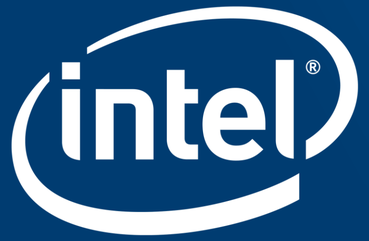
- Experimental and observational science (EOS) facilities have data management requirements beyond existing HDF5 features
- Targeted science drivers
  - LCLS / LCLS-II, LSST, ALS, NIF
- Requirements
  - Multiple producers and multiple consumers of data
  - Remote streaming synchronization
  - Handling changes in data, data types, and data schema
  - Search metadata and provenance directly in HDF5 files
  - Support for different forms of data - Streaming, sparse, KV, etc.
  - Optimal data placement

EOD-HDF5 - Proposed features

- Multi-modal access and distributed data in workflows
  - Multiple Writers, Multiple Readers (MWMR)
  - Distribution of local changes to remote locations
- Data model extensions
  - Storing new forms of data (KV, index-like data structures, streaming data)
  - Addressing science data schema variation
  - Managing collections of containers
- Metadata and provenance management
  - Capturing and storing rich metadata contents and provenance
  - Searching metadata and provenance
  - Optimal data placement based on data analysis patterns

Future HDF5 Features

- Performance improvements for both small and large-scale I/O
  - Resource usage profiling in HDF5 to identify bottlenecks
  - Automatic bottleneck avoidance techniques
- Sub-filing and other topology-aware I/O
  - TAPIOCA, etc.
- Fill Parallel I/O Gaps
  - Parallel Async & SWMR
- Enhance storage model
  - Column-oriented & Sparse data storage
- Track Future Technology Changes
  - Object file system VOL plugins (DAOS, Ceph, MarFS, …)



HDF BOF

Andreas Dilger

High Performance Data Division

SC'17, Denver

Composite/Progressive File Layout Lustre 2.10

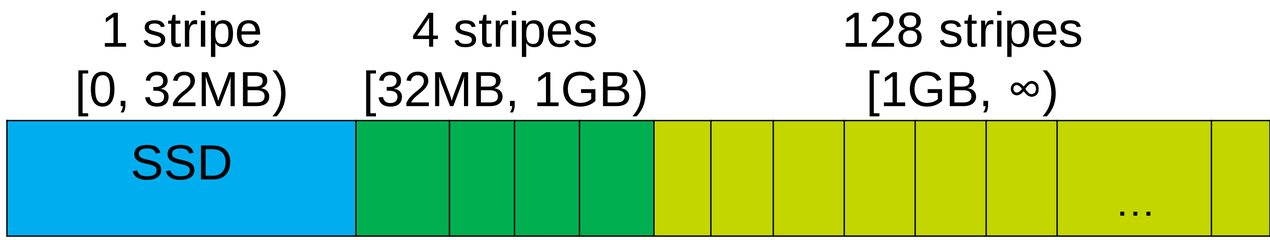
Composite File Layout allows different layout based on file offset

- Provides flexible layout infrastructure for upcoming features
  - Data-on-MDT (DoM), File Level Redundancy (FLR), HDF library integration, etc.
- Layout components can be disjoint (e.g. PFL) or overlapping (e.g. FLR)

Progressive File Layout (PFL) simplifies usage for users and admins

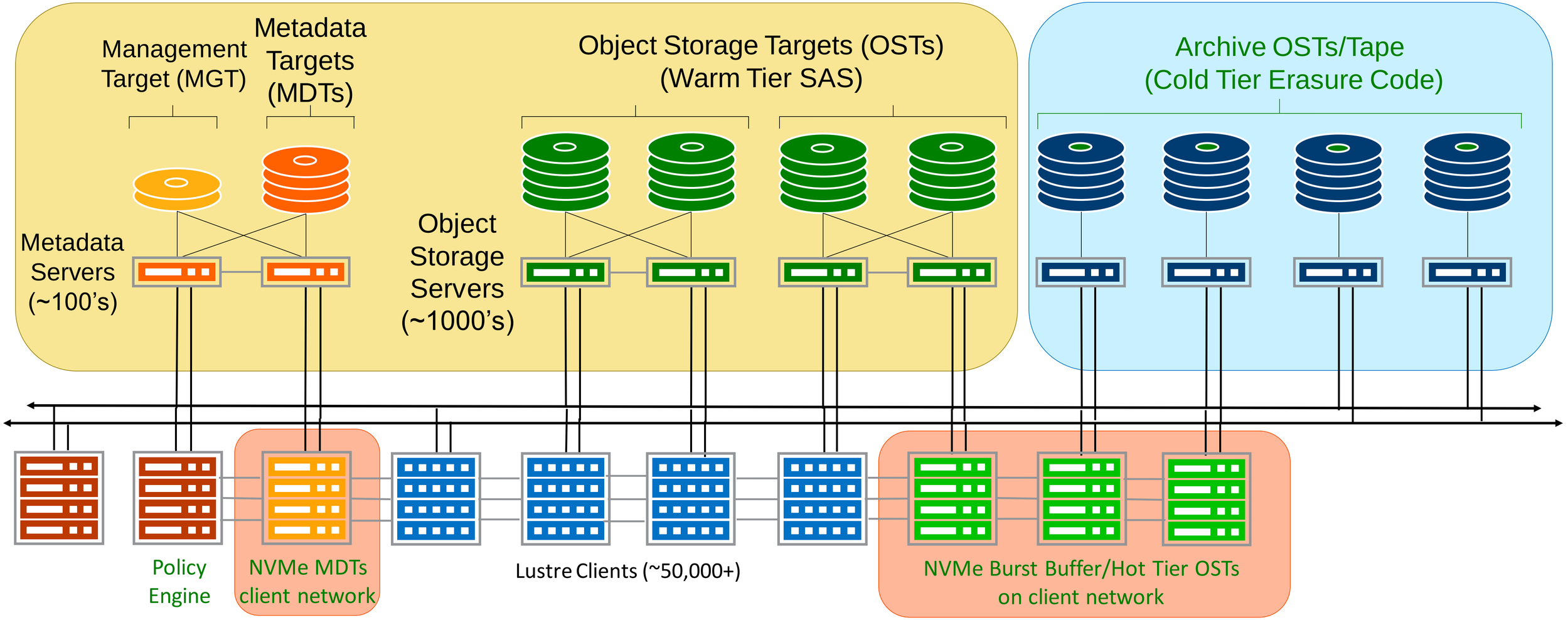
- Optimize performance for diverse users/applications
- One PFL layout could be used for all files
- Low stat overhead for small files
- High IO bandwidth for large files

Example progressive file layout with 3 components



Tiered Storage and File Level Redundancy

Data locality, with *direct access* from clients to all storage tiers as needed



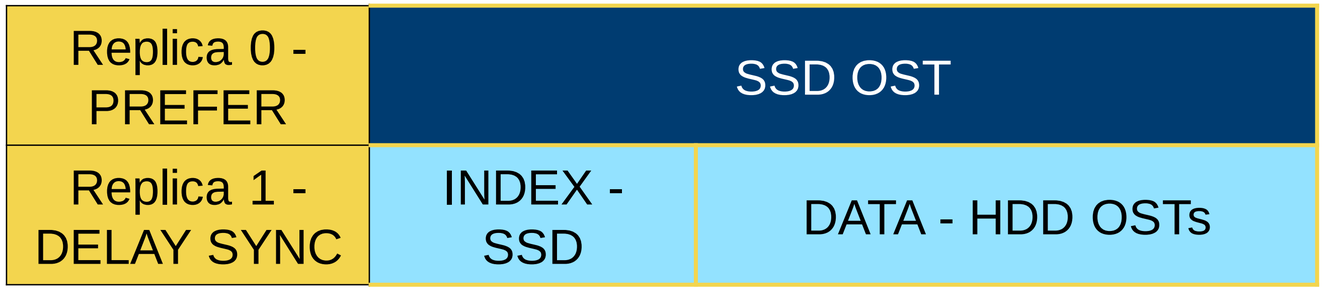
File Level Redundancy Lustre 2.11+

Significant value and functionality added for HPC and other environments

- Optionally **set on a per-file/dir basis** - flexibility to tune as application/user needs
- **Higher availability** for server/network failure – finally **better than HA failover**
- **Robust against data loss/corruption** – mirror (and later erasure code) data across OSTs
- Migrate NVM<->SSD<->HDD<->Archive, but **allows direct access if needed**

Configure redundancy on a per-file or directory basis, for example:

- Pre-stage files to SSD for read/write
- Mirror only 1 of 24 hourly checkpoints
- 12+3 erasure code large striped files
- Write to SSD for IOPS, mirror to HDD



FLR Phased Implementation Approach

Can implement Phases 2/3/4 in any order

Phase 0: Composite Layouts from PFL project — Lustre 2.10

- Plus OST pool inheritance, Project/Pool Quotas

Phase 1: Delayed read-only mirroring – depends on Phase 0 — Lustre 2.11

- Manually replicate and migrate files across multiple tiers

Phase 2: Integration with policy engine/copytool - with/after Phase 1

- Automated migration between tiers based on admin policy/space

Phase 3: Immediate write replication – depends on Phase 1 — Lustre 2.12?

- Client writes file data to multiple OSTs in parallel

Phase 4: Erasure coding for striped files - with/after Phase — Lustre2.12?

- Avoid 2x or 3x overhead of mirroring files

Legal Notices and Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life-saving, life-sustaining, critical control or safety systems, or in nuclear facility applications.

Intel products may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

This document may contain information on products in the design phase of development. The information herein is subject to change without notice. Do not finalize a design with this information. Intel may make changes to dates, specifications, product descriptions, and plans referenced in this document at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Intel Corporation or its subsidiaries in the United States and other countries may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Performance estimates or simulated results based on internal Intel analysis or architecture simulation or modeling are provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

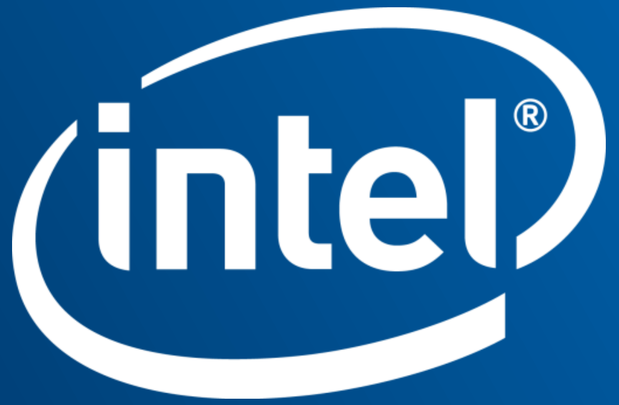
Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804.

Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

Intel processors of the same SKU may vary in frequency or power as a result of natural variability in the production process.

Intel, the Intel logo, 3D-Xpoint, Optane, Xeon Phi, and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

* Other names and brands may be claimed as the property of others.


intel

Experiences with HDF5: I/O Mini-apps, Compression & Physics-based Simulations

User Productivity Enhancement, Technology Transfer, and Training (PETTT)

Presented by

Sean Ziegeler (Engility PETTT)

November 15, 2017

DOD HPC MODERNIZATION PROGRAM

PETTT

Topics

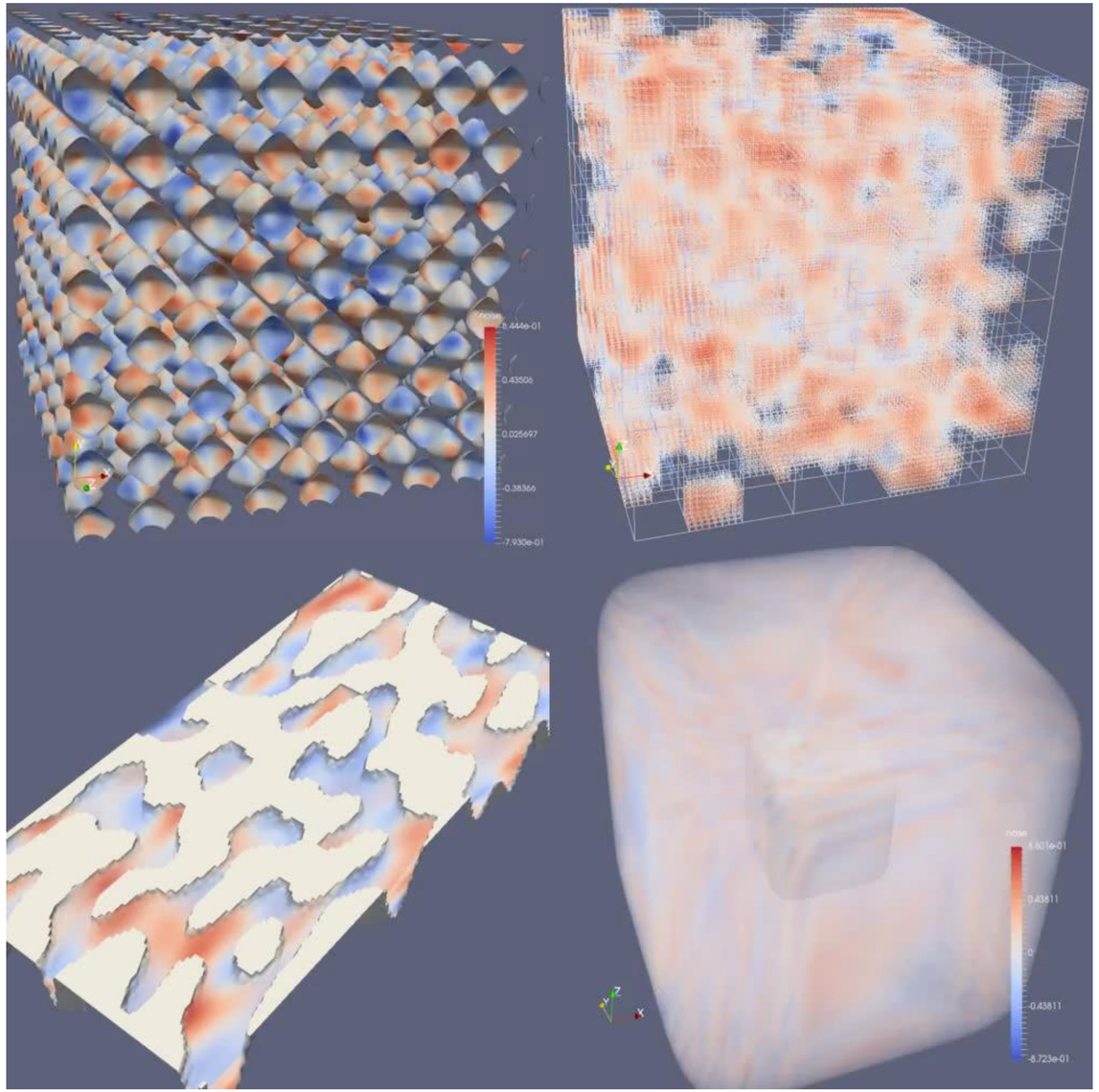
- **MiniIO: I/O Mini-apps**
- **Compression Study**
  - Results
- **HDF5 Autotuner Results**
- **How/Why HDF5**
- **What do we need from HDF5?**

MiniIO: I/O Mini-apps

- **MPI, no threads (yet, but can be simulated with fewer ranks per node)**
- **Four apps for now, common physics-based HPC simulation data structures**
  - All are interesting use cases for compression, but here we focus on one
- **MPI-IO, ADIOS, & HDF5 output options**
- **Built explicitly around physics-based code options**
  - Not extracted code kernels or emulators, as per, e.g., *skel* or *MACSio* or general I/O benchmarks like *IOR*
  - Why not? (1) Skel only for ADIOS; MACSio had not been published yet
  - Why not? (2) MiniIO provides numerous physics-based simulation options (grid settings, load balancing, variable settings …) that can be directly mapped to performance
  - IOR or MACSio could work in the long run, but we would need to ensure all MiniIO features mapped properly to them

MiniIO: I/O Mini-apps

”Cartiso”



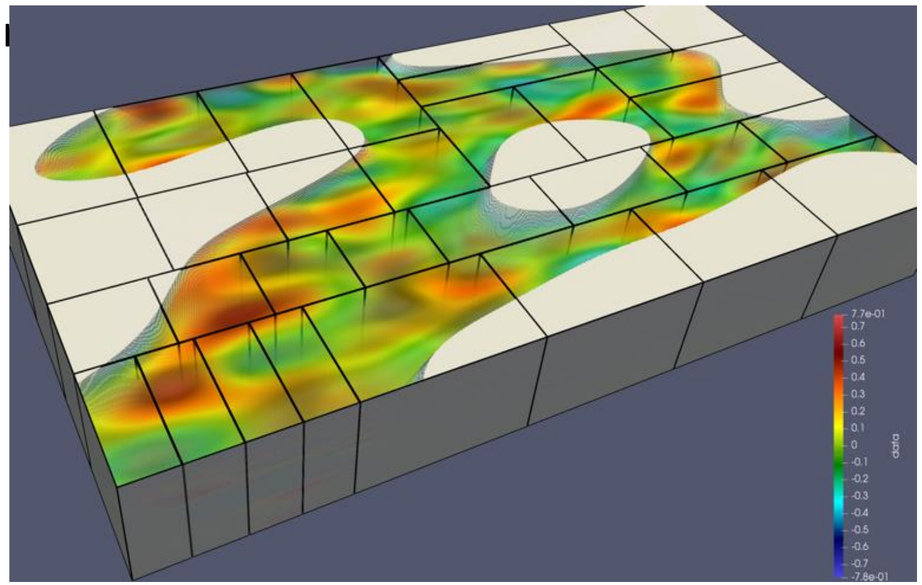
”AMR”

”Struct”

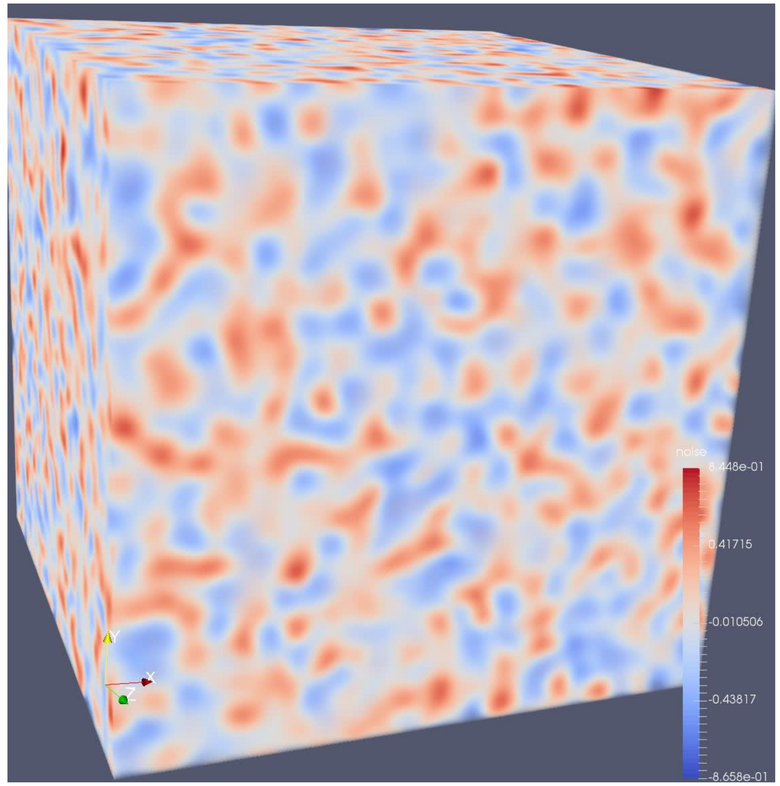
”Unstruct”

Struct Mini-app

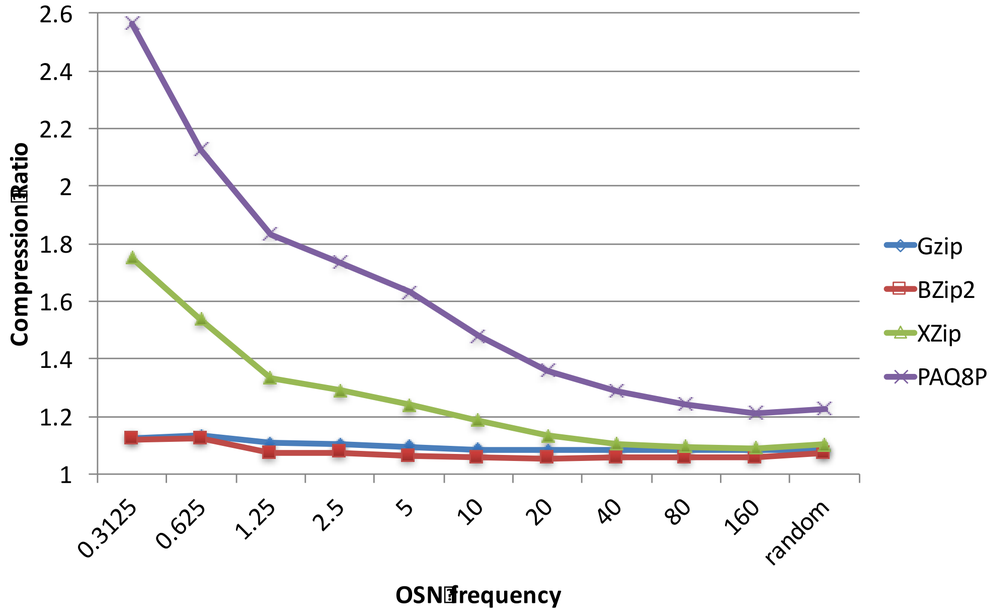
- **Struct: structured grids with masks/blanking**
  - Masks for missing or invalid data (e.g. land in an ocean model)
    - 2D simplectic noise to generate synthetic mask maps
    - Can choose % of blanked data points
    - Noise frequency governs sizes of blanked areas (continents vs islands)
  - 4D simplectic noise to fill time-variant variables
  - Option for load balancing non-masked points evenly (as desired) across r
    - But creates load imbalance for I/O because blanked data is still written
    - Compression theoretically rebalances the I/O (blanked constants compress well)



4D Simplectic Noise & Compression



Rendering of a volume
of simplectic noise

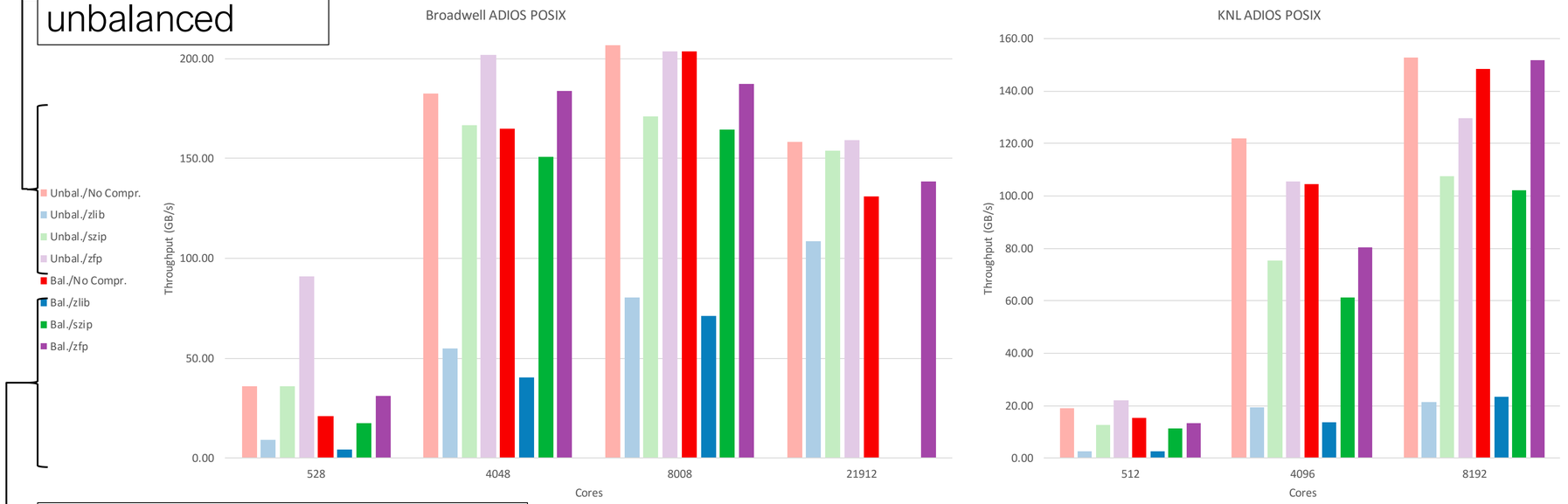


Compression results of simplectic noise
at various frequencies

Results

ADIOS POSIX: one file per rank

Computationally unbalanced

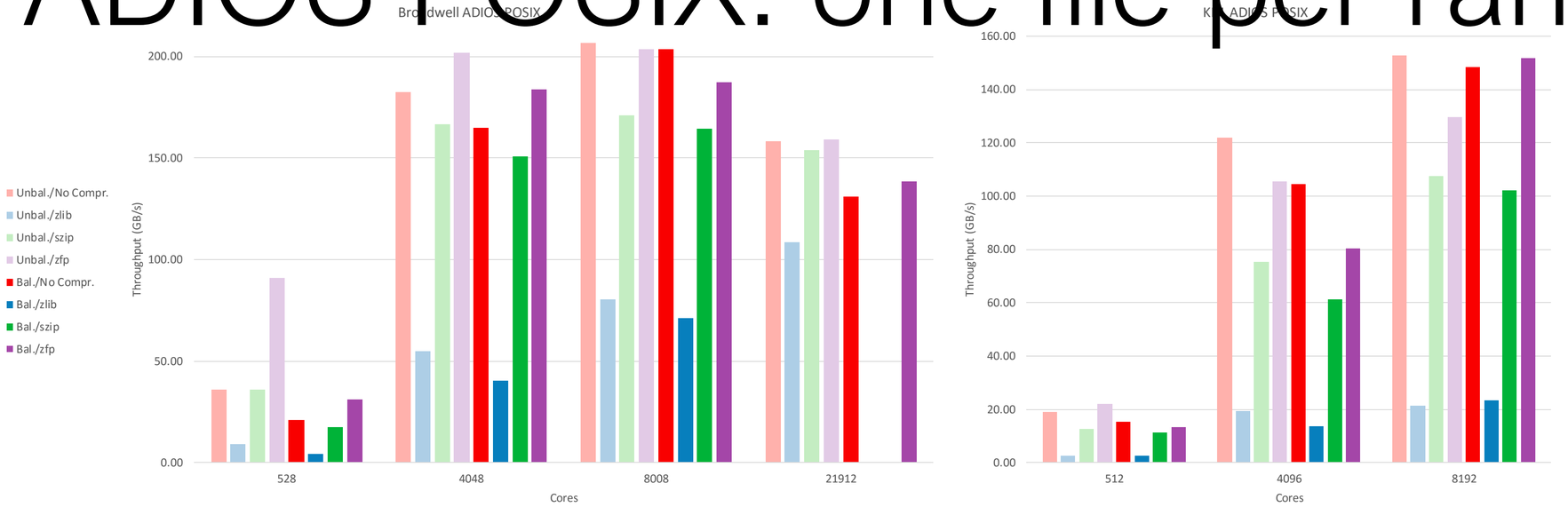


Balanced (I/O unbalanced!)

Red: No compression
Blue: zlib deflate compression (think gzip)
Green: szip compression
Purple: zfp (error bounded lossy, 0.0001), ~9:1 on average

Results

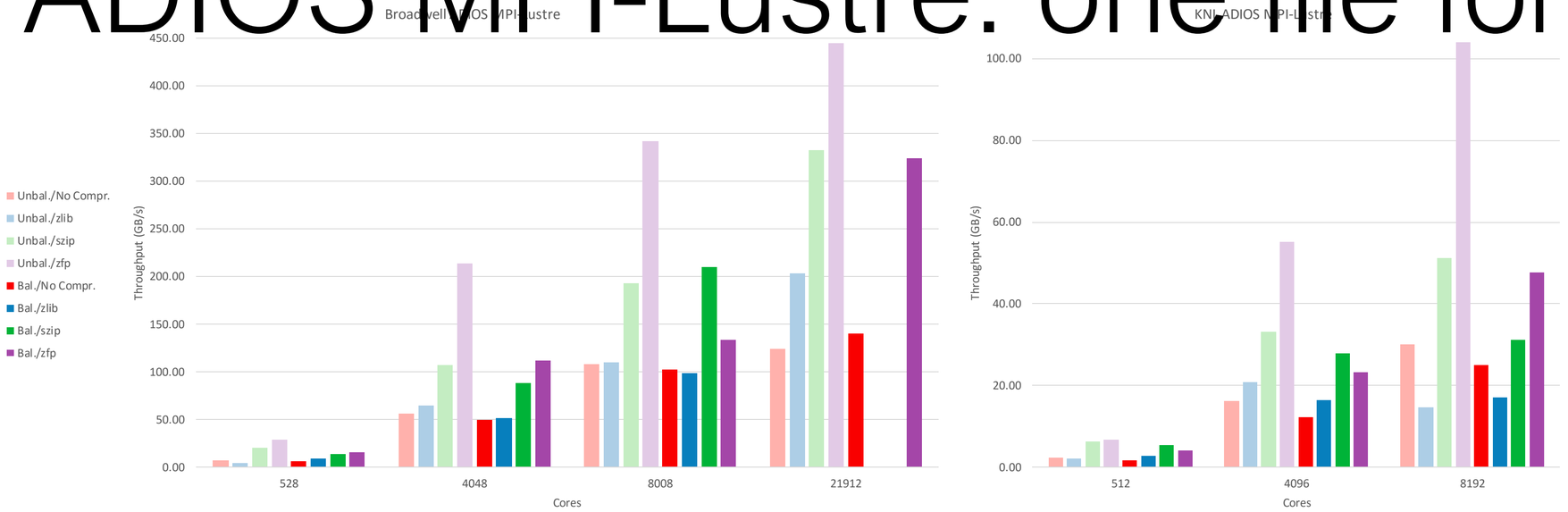
ADIOS POSIX: one file per rank



- **Initial scalability with core count**
- **Computational balancing hurts performance a little**
  - But compression sometimes helps
- **Zfp is the fastest compression**
- **KNL is slower**
- **ADIOS POSIX is the fastest without compression**

Results

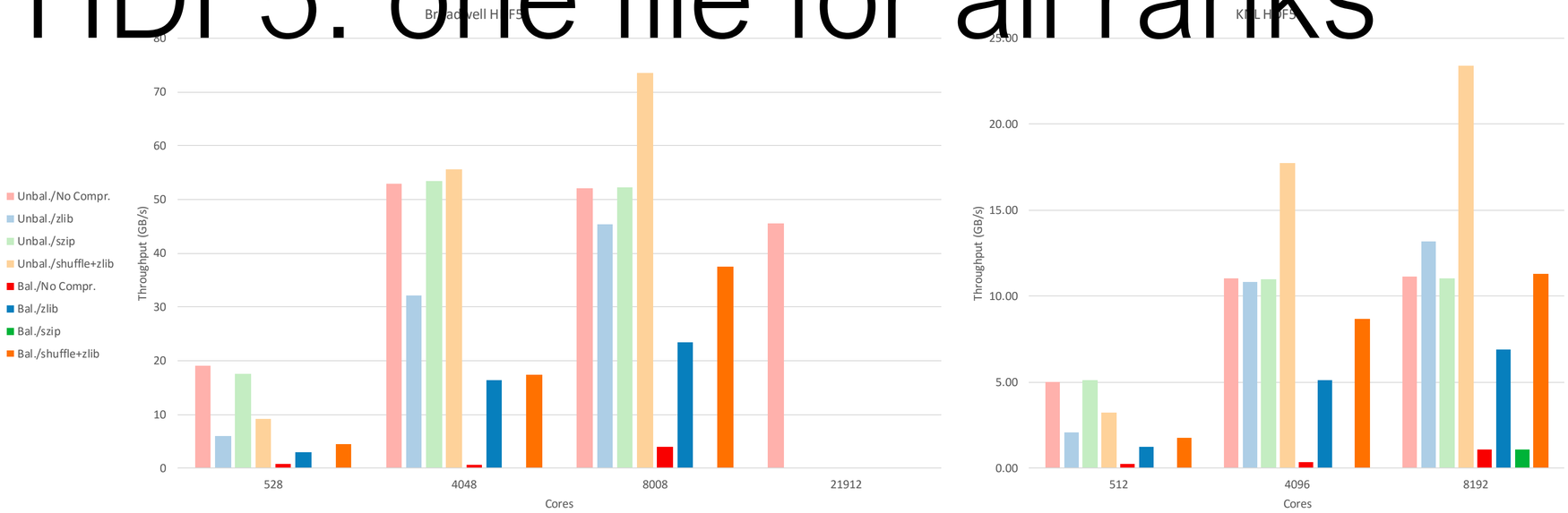
ADIOS MPI-Lustre: one file for all rank



- **Good scalability with core count, especially with compression**
- **Computational balancing hurts performance a little**
  - But compression mostly helps
- **Zfp is by far the fastest compression**
- **KNL is much slower, especially the compression**
- **MPI-Lustre is the fastest with compression**

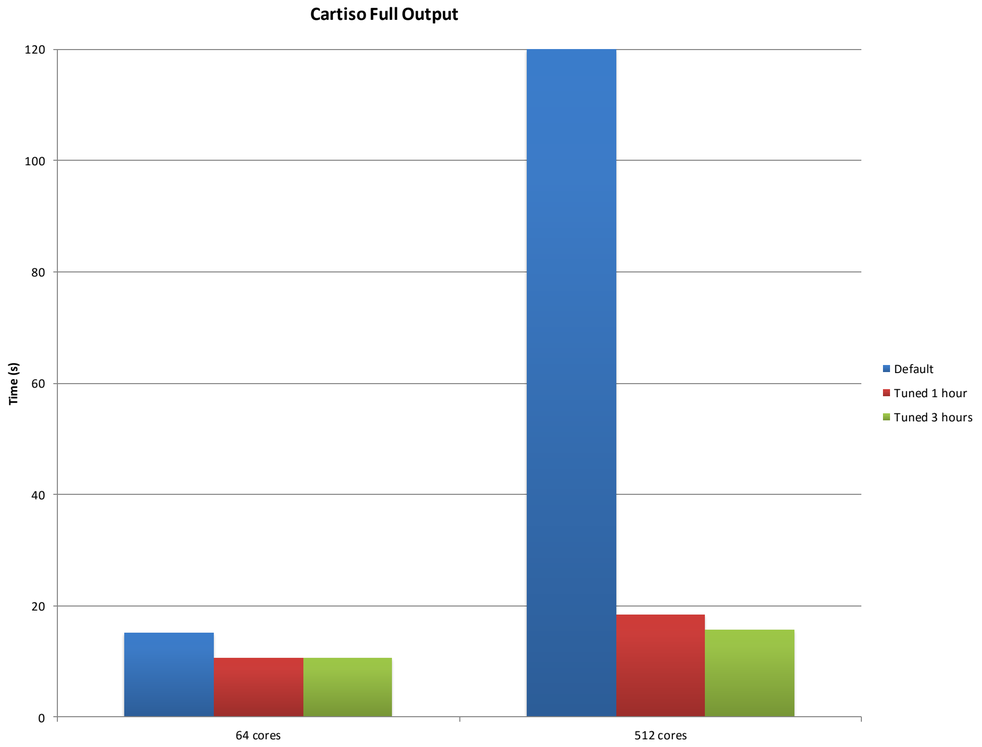
Results

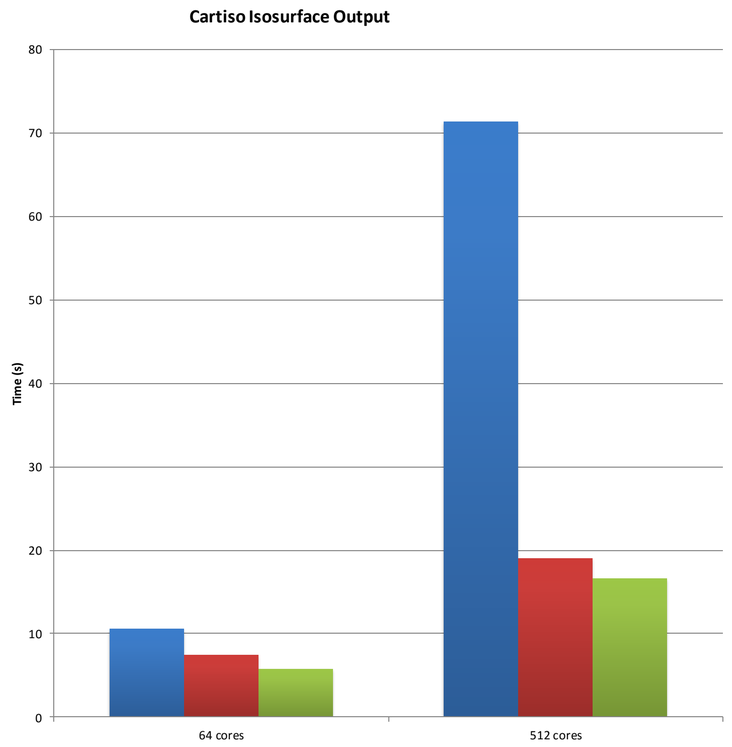
HDF5: one file for all ranks



- **Starts slower, but scalability with core count, especially with compression**
- **Computational balancing hurts performance a lot**
  - But compression helps somewhat
- **Shuffle+zlib is the fastest compression (zfp not available at the time)**
- **KNL is much slower, especially the compression**
- **HDF5 can scale with compression**

HDF5 Autotuner Results





How/Why do we use HDF5?

- **The most standardized format**
  - Can find some way to use HDF5 in Ensight, VisIt, ParaView …
  - The primary underlying format for the CGNS standard
  - Also via netCDF-4 (built atop HDF5)
- **Yet more flexible than other formats**
  - Hierarchies, non-structured data, analytics with binary data
  - Weirdness is usually at odds with standardization
- **Already installed on most of our systems**
- **Compression and other filters**
- **Emerging features like Virtual Data Sets (VDS)**

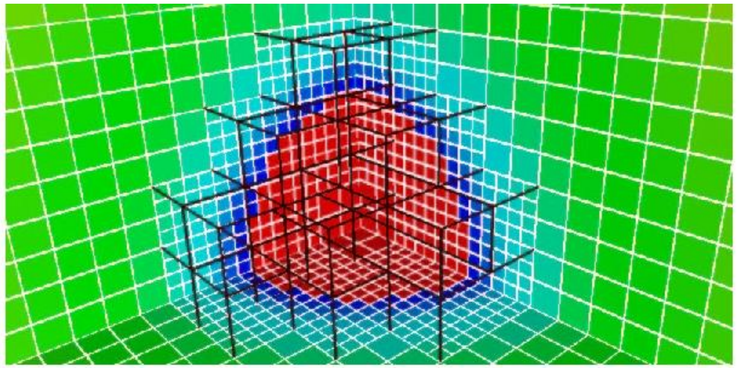
What do we need from HDF5?

- **Continued improvements to parallel performance**
  - Ensure that transitions to netCDF
- **Smarter handling of parallel file systems**
  - Very difficult for users to set stripe counts, stripe sizes, collective buffering
  - Autotuner is helpful but difficult to use
  - Potentially integrate into the library
- **Zfp with HDF5**
  - Transition to netCDF
- **VDS for parallel I/O**
  - Very promising approach to transcend single-file limitations
  - Transition to netCDF

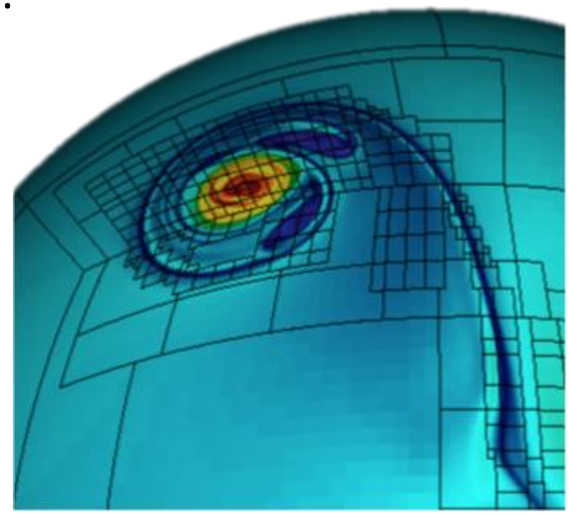
This material is based upon work supported by,
or in part by, the Department of Defense
High Performance Computing Modernization Program (HPCMP)
under User Productivity, Technology Transfer and Training (PETTT)
contract number GS04T09DBC0017.

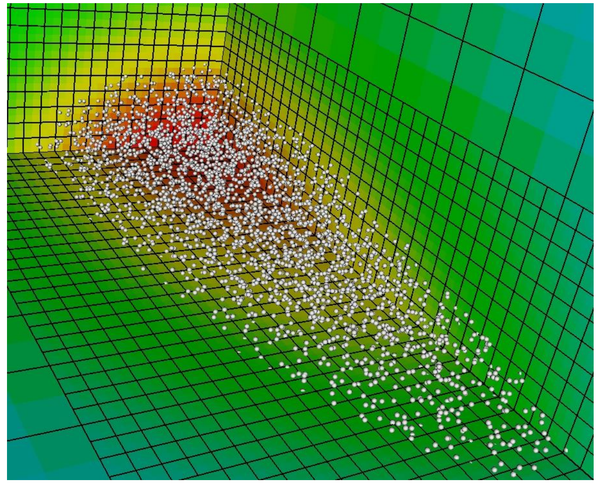
Block-Structured Adaptive Mesh Refinement (AMR)

- Refined regions are organized into rectangular patches.



- Refinement in time as well as in space for time-dependent problems.
- Local refinement can be applied to any structured-grid data, such as bin-sorted particles.





Chombo and HDF5

- HDF5 is our primary IO Middleware on all platforms
  - Portable plot and checkpoint files
  - Reader for Visit
  - Parallel IO with hyperslabs (using our own global variable ordering)
- New roles
  - Code coupling between Chombo (structured AMR) and GEOS (unstructured FEM)
  - Asynchronous workflow through NVRAM technologies

Wish List

- warning: 'tmpnam' is deprecated
  - Need temporary file creation mechanism
  - Workflow, using HDF5 in debugger
- Code coupling without requiring disk
  - RAM to RAM transfer in MPI jobs (ie pipe)
  - Possibly with disk backing
- DataMover (asynchronous data migration, workflow)
- FArrayBox to “Object Store”
  - Tag FArrayBox with [time,level,index] key, receive future
- FASTBit
- Aggregated IO handled for me
- Set Lustre file system parameters for me.