DAOS: A New Storage Paradigm for Exascale

HLRS Workshop, April 2017

Johann Lombardi, High Performance Data Division, Intel

Legal Information

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

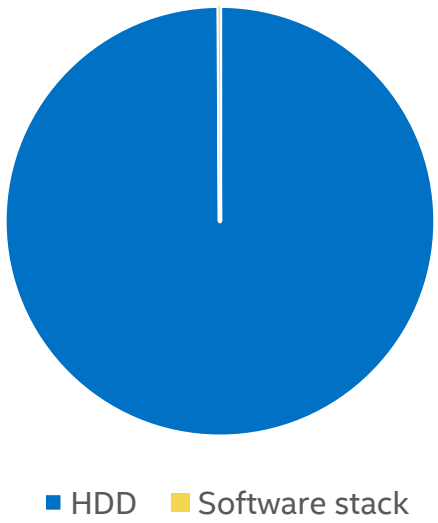
Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation.

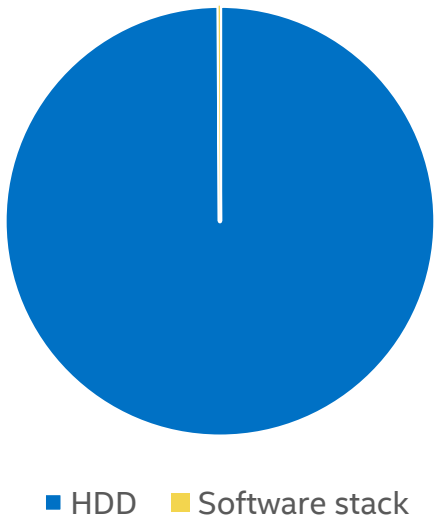
- Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at http://www.intel.com/content/www/us/en/software/intel-solutions-for-lustre-software.html.
- Intel may make changes` to specifications and product descriptions at any time, without notice.
- Any code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.
- Intel, Intel Inside, the Intel logo and Xeon Phi are trademarks of Intel Corporation in the United States and other countries.
- Material in this presentation is intended as product positioning and not approved end user messaging.

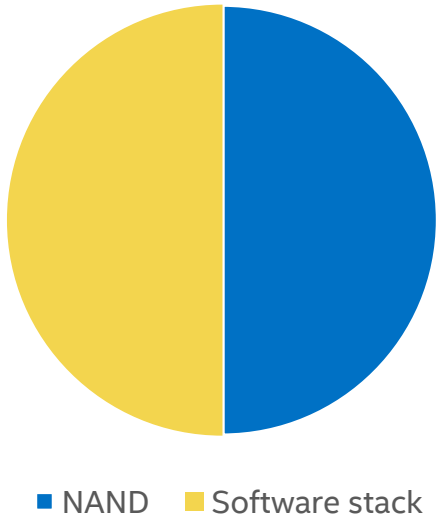
*Other names and brands may be claimed as the property of others.

Challenge: I/O Latency

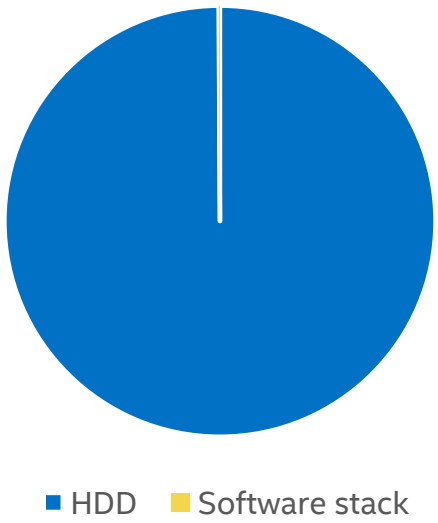


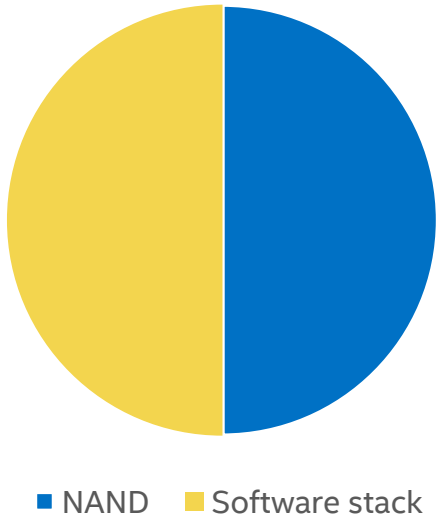
Challenge: I/O Latency

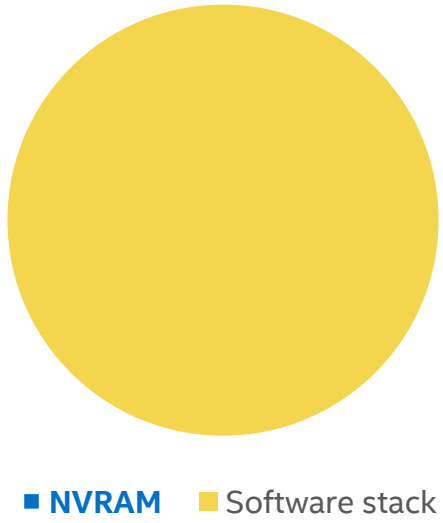




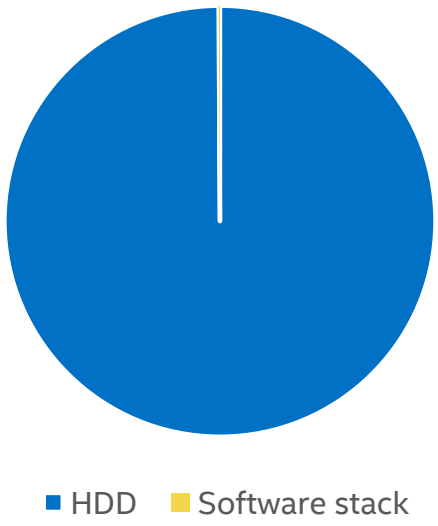
Challenge: I/O Latency

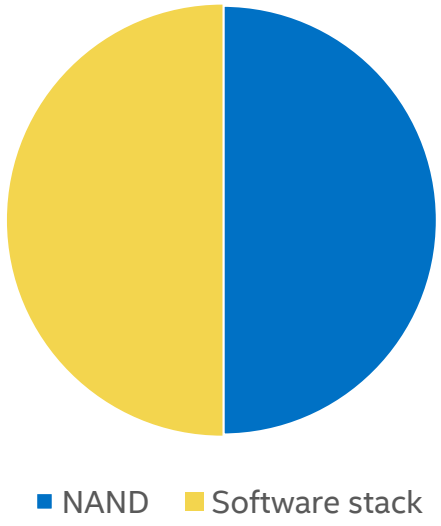


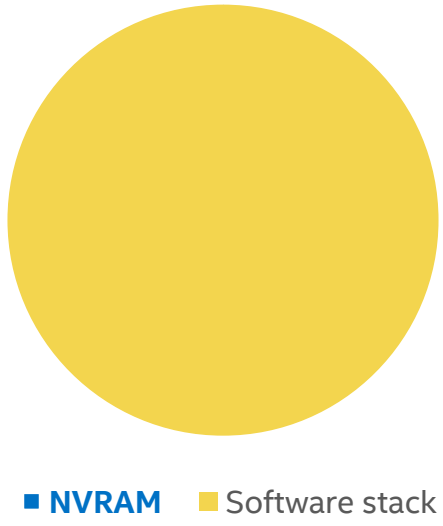




Challenge: I/O Latency

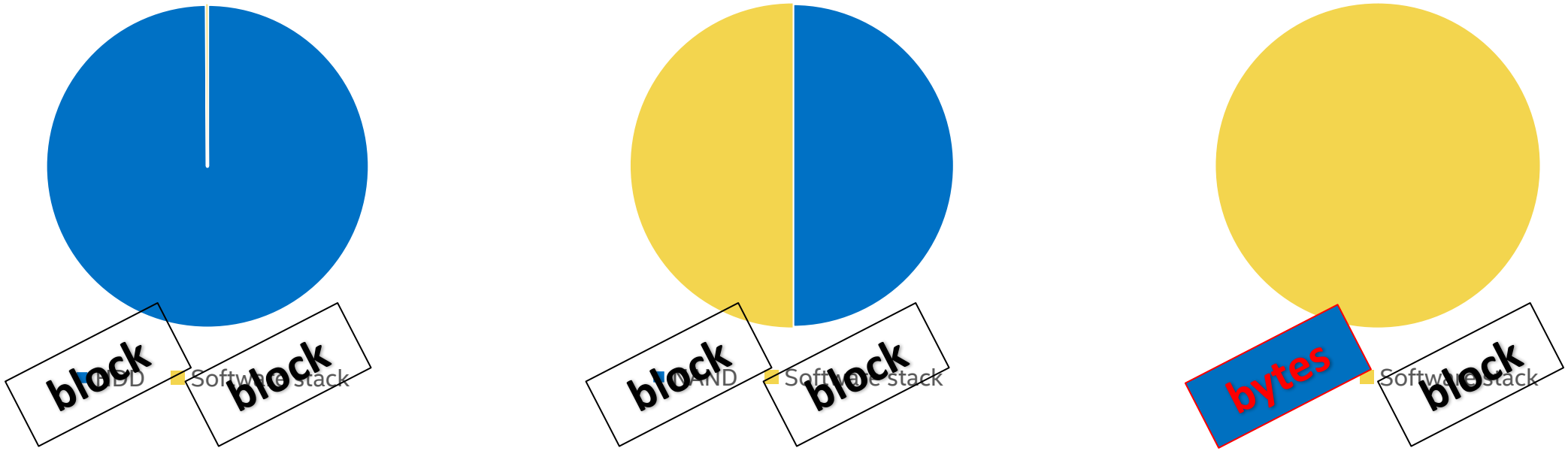






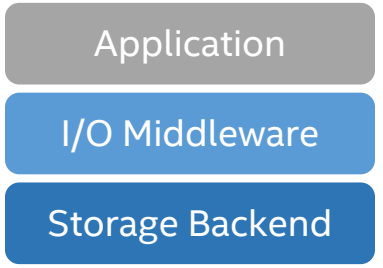
Traditional storage stack entirely **masks** low latency of **NVRAM**!

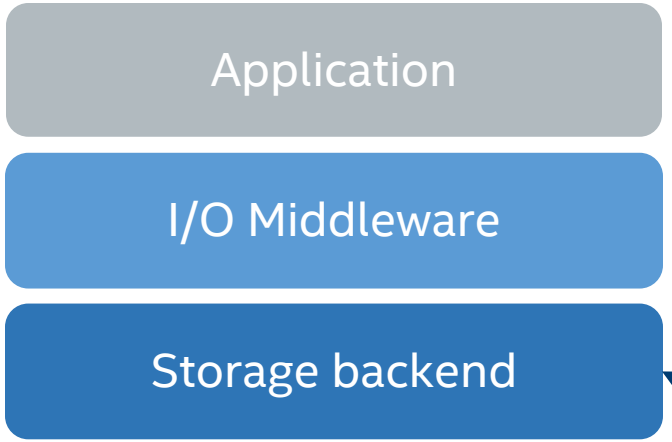
Challenge: Access Granularity



Traditional storage stack entirely **masks**
low latency & capabilities of **NVRAM**!

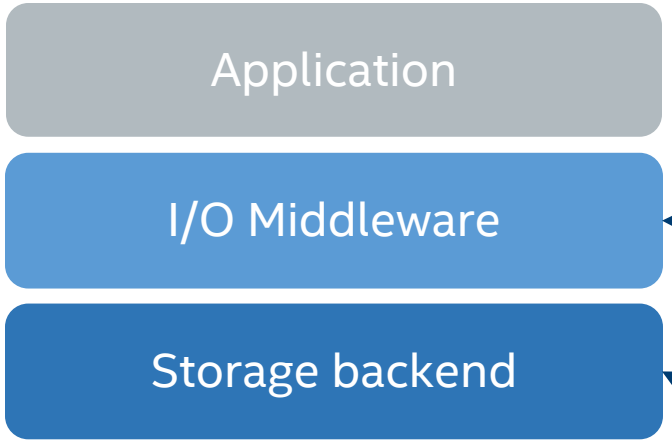
ESSIO Storage Stack





New storage API (*DAOS*) provides extended capabilities and low-latency I/O to middleware

ESSIO Storage Stack

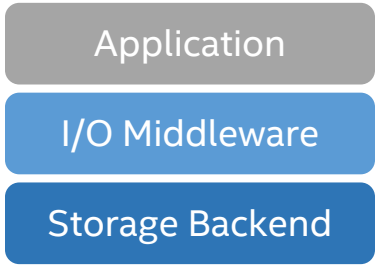


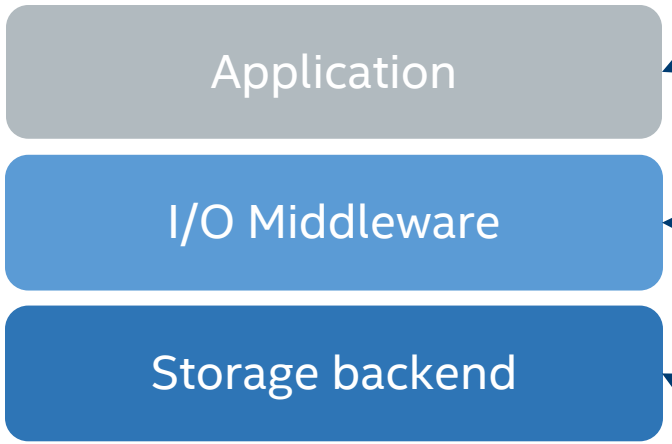
Port I/O middleware (*HDF5, ADIOS, …*) to new storage backend and **augment** API to take advantage of new capabilities

New storage API (*DAOS*) provides extended capabilities and low-latency I/O to middleware

Application
I/O Middleware
Storage Backend

ESSIO Storage Stack





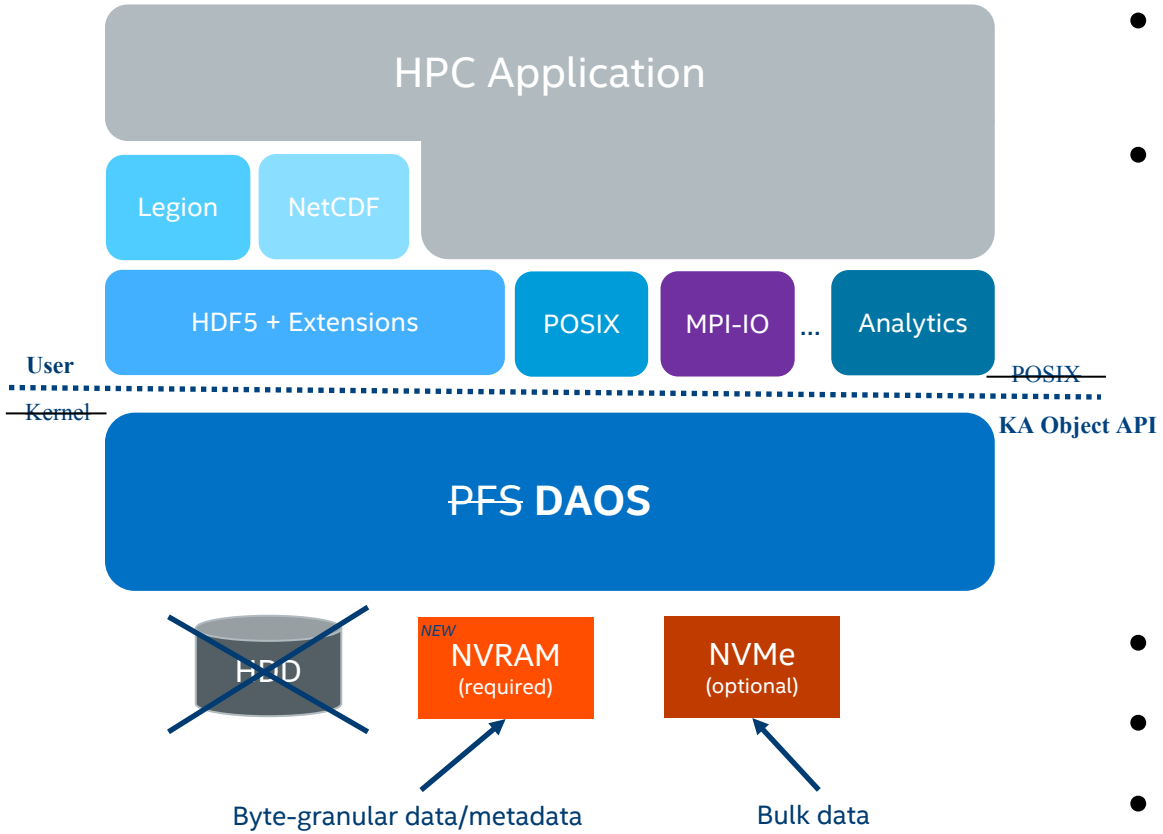
Evaluate applications (*HACC, ACME, CLAMR)* and new programming model (*Legion*) over enhanced I/O middleware

Port I/O middleware (*HDF5, ADIOS, …*) to new storage backend and **augment** API to take advantage of new capabilities

New storage API (*DAOS*) provides extended capabilities and low-latency I/O to middleware

Distributed Asynchronous Object Storage

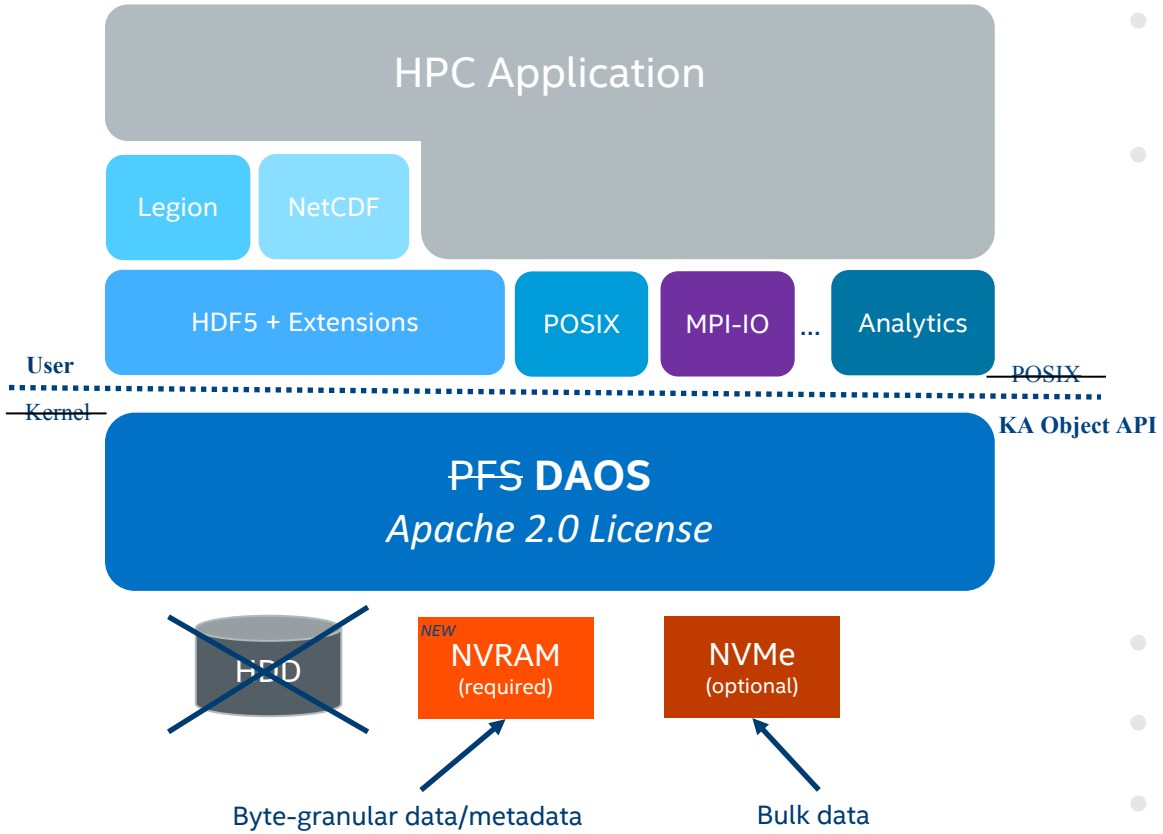
Application
I/O Middleware
Storage Backend



- Storage stack designed for NVRAM & NVMe storage technologies with integrated fabric
- Advanced backend storage API providing:
  - New storage model based on key-array objects
  - Ultra-low latency & non-blocking I/O
  - Distributed transactions & snapshots
  - Extreme scalability & resilience
  - Native support for producer/consumer pipeline
  - Ad-hoc concurrency control mechanism
  - Query & indexing
- Resource/workflow manager integration
- Software defined storage platform
- Integrated multi-tier support

Distributed Asynchronous Object Storage

Application
I/O Middleware
Storage Backend

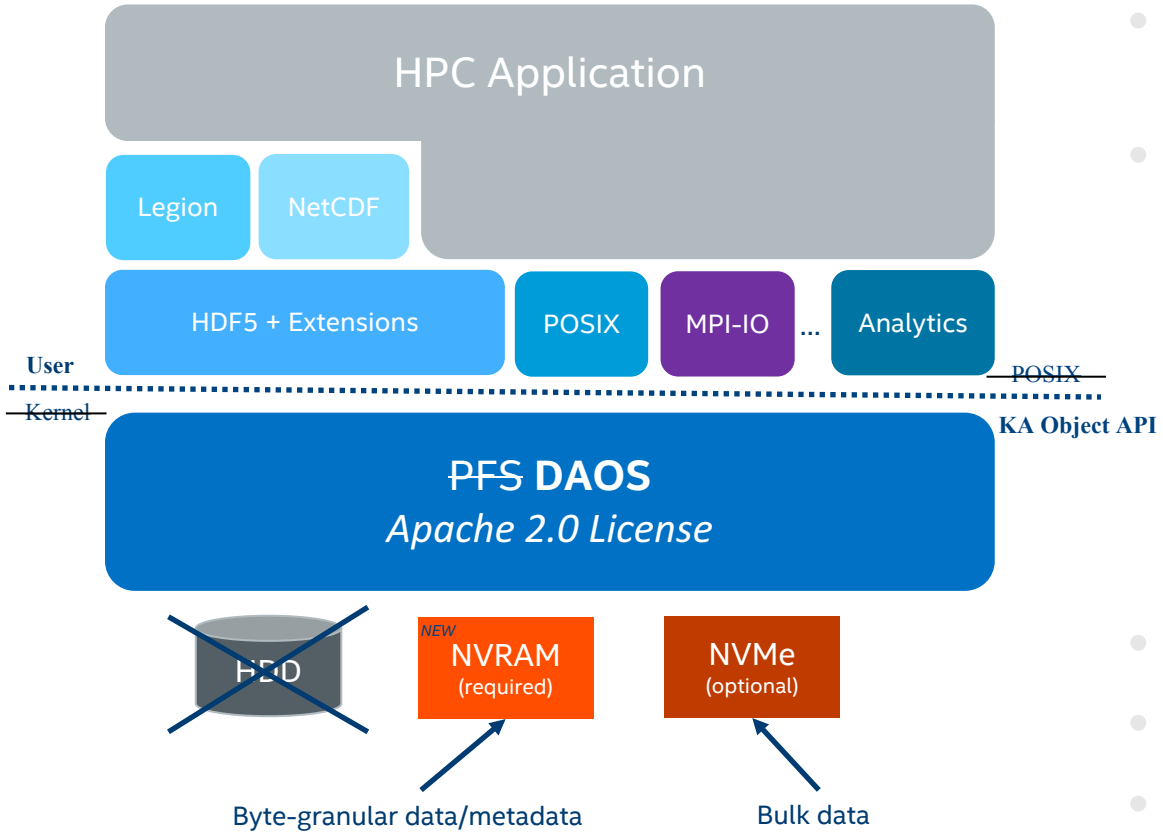


- Storage stack designed for NVRAM & NVMe storage technologies with integrated fabric
- Advanced backend storage API providing:
  - New storage model based on key-array objects
  - Ultra-low latency & non-blocking I/O
  - Distributed transactions & snapshots
  - Extreme scalability & resilience
  - Native support for producer/consumer pipeline
  - Ad-hoc concurrency control mechanism
  - Query & indexing
- Resource/workflow manager integration
- Software defined storage platform
- Integrated multi-tier support

Open source
APACHE 2.0 License

Distributed Asynchronous Object Storage

Application
I/O Middleware
Storage Backend

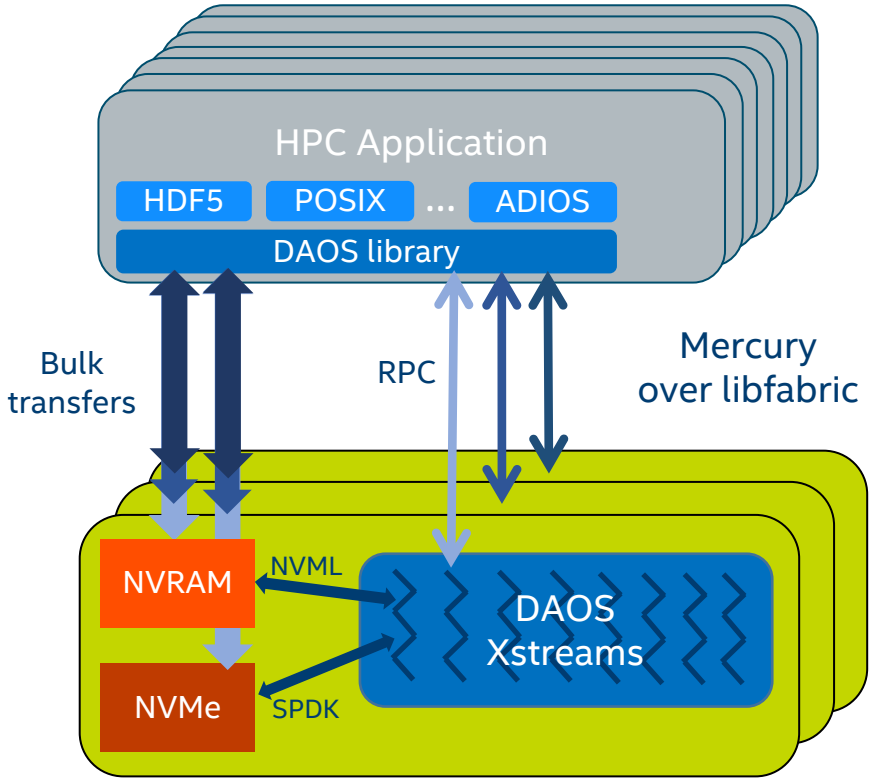


- Storage stack designed for NVRAM & NVMe storage technologies with integrated fabric
- Advanced backend storage API providing:
  - New storage model based on k-v array objects
  - Ultra-low latency & non-blocking I/O
  - Distributed transactions & snapshots
  - Extreme scalability & resilience
  - Native support for producer/consumer pipeline
  - Ad-hoc concurrency control mechanism
  - Query & indexing
- Resource/workflow manager integration
- Software-defined storage platform
- Integrated multi-tier support

Community Project
https://github.com/daos-stack

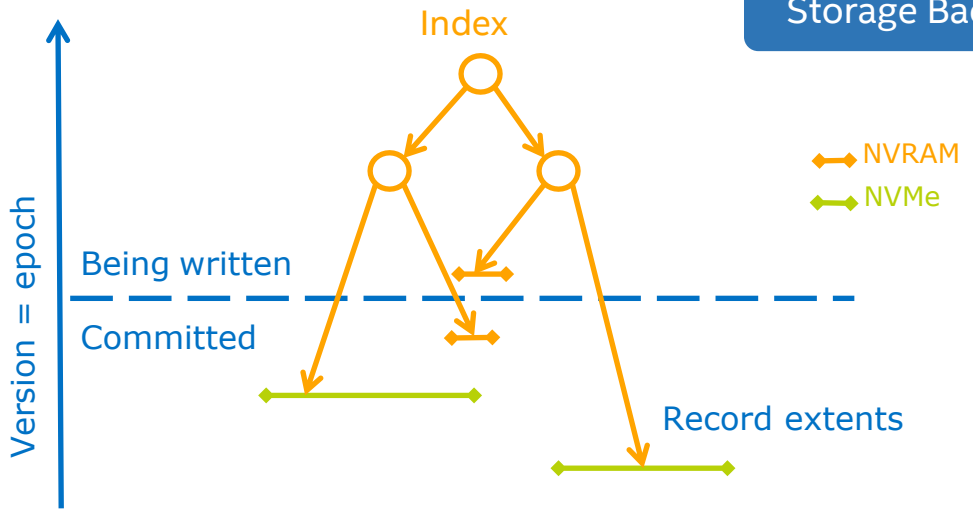
Lightweight Storage Stack

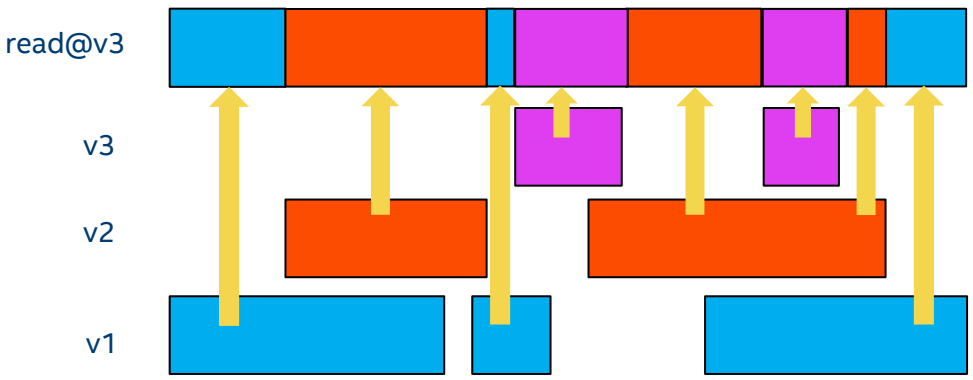
- Mercury userspace function shipping
  - MPI equivalent communications latency
  - Built over libfabric
- Applications link directly with DAOS lib
  - Direct call, no context switch
  - No locking, caching or data copy
- Userspace DAOS server
  - Mmap non-volatile memory (NVML)
  - NVMe access through SPDK/BlobFS
  - User-level thread with Argobots



Ultra-fine grained I/O

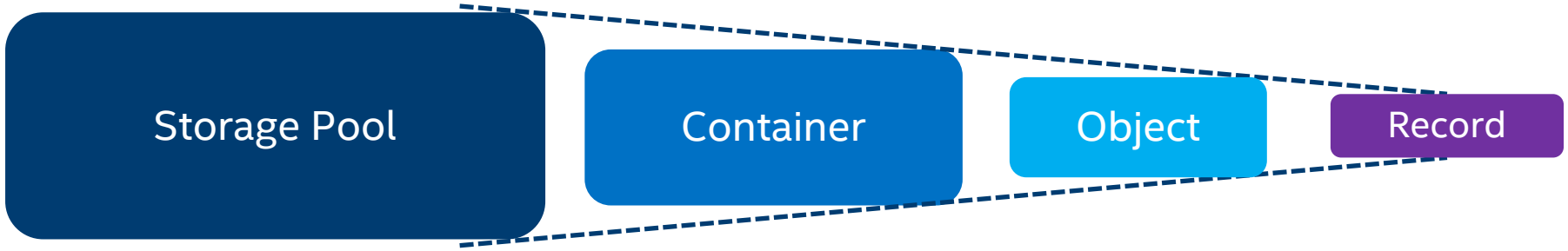
- Mix of storage technologies
  - NVRAM
    - DAOS metadata & application metadata
    - Byte-granular application data
  - NVMe
    - Cheaper storage for bulk data
    - Multi-KB
- I/Os are logged & inserted into persistent index
  - All I/O operations tagged/indexed by version
    - Non-destructive write & consistent read
  - No alignment constraints
    - no read-modify-write





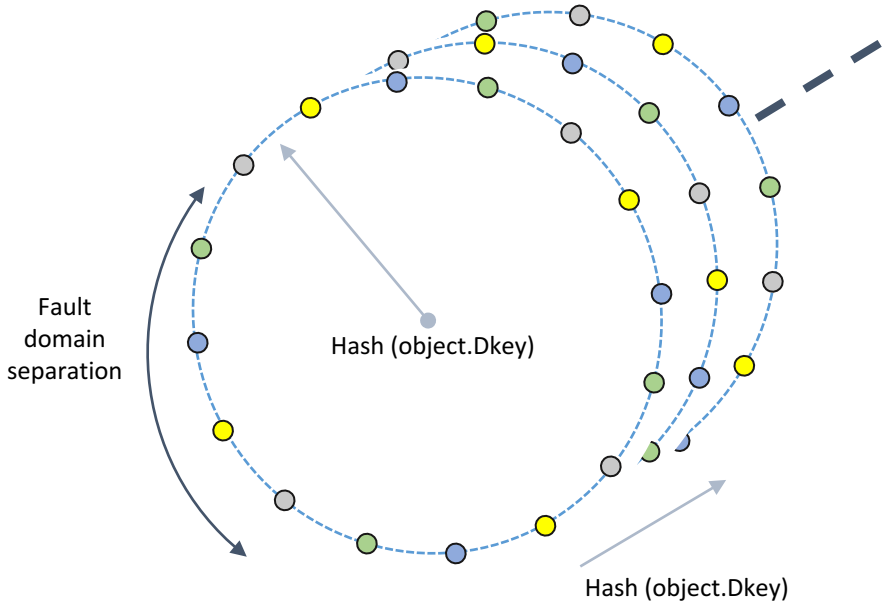
DAOS Storage Model

- Storage Pool
  - Reservation of distributed storage within a tier
  - Integration with resource manager
- Container
  - Aggregate related datasets into manageable entity
  - Distributed across entire storage pool
  - Unit of snapshot/transaction
- Object
  - Collection of related arrays/values with own distribution/resilience schema
  - Key-array store with flexible multi-level key
    - fine-grain control over colocation of related data
- Record
  - Arbitrary binary blob from single byte to several Mbytes



Scalability & Resilience

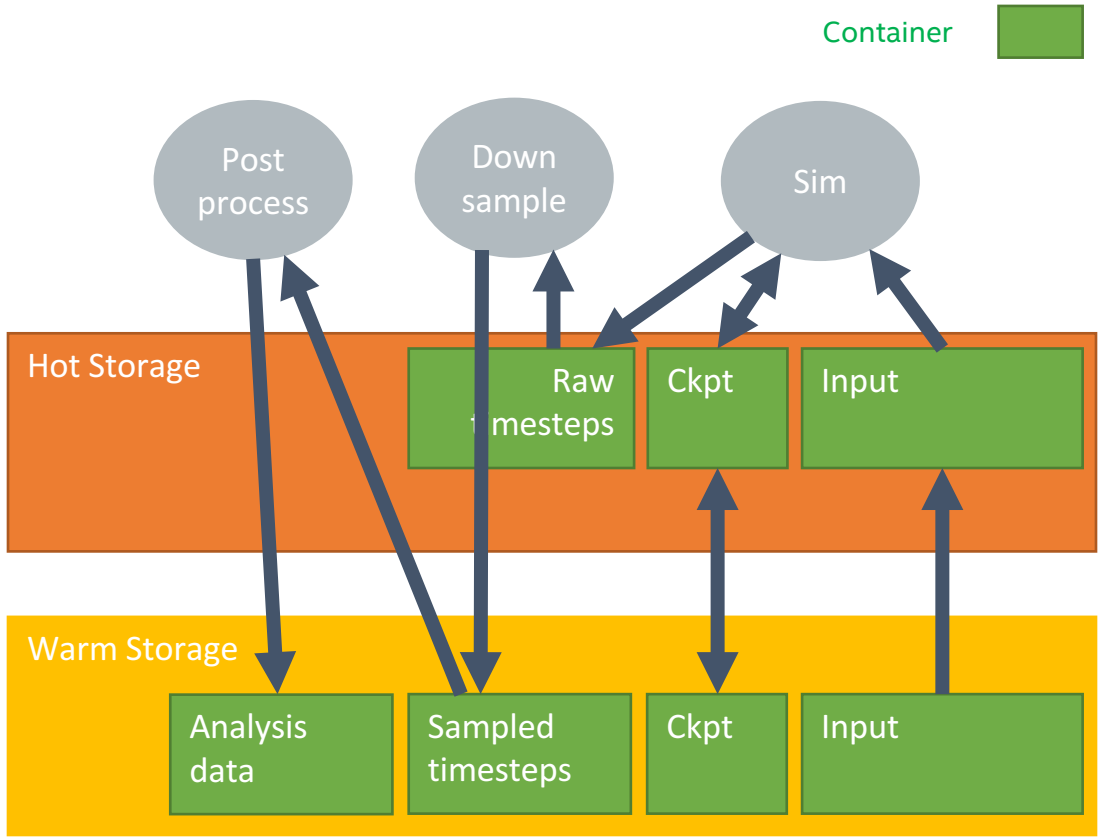
- Scalable communications
- Scalable I/O
- Shared-nothing distribution & redundancy schema



- Storage system failure
  - Failed target(s) evicted
  - Automated online rebuild & rebalancing
- End-to-end integrity
  - Checksums can be provided by I/O middleware
  - Stored and verified on read & scrubbing
- Application failure
  - Scalable distributed transaction exported to I/O middleware
  - Automated recovery to roll back uncommitted changes

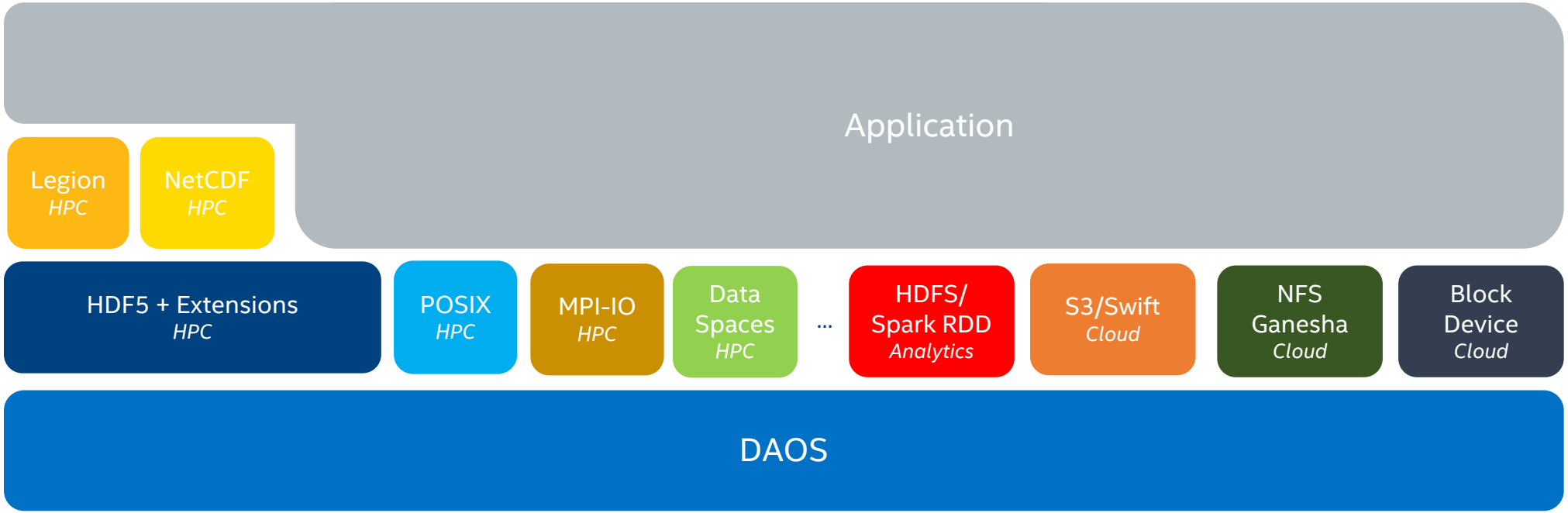
New Workflow Methodologies

- Simplify application & workflow development
  - Transparently span multiple tiers
  - Support producer/consumer pipeline
  - Simplified data management
    - Aggregated related dataset into manageable entitiers, i.e. DAOS container
    - Snapshots
- Integration with resource manager
  - Storage management & job monitoring
  - Data-aware scheduling



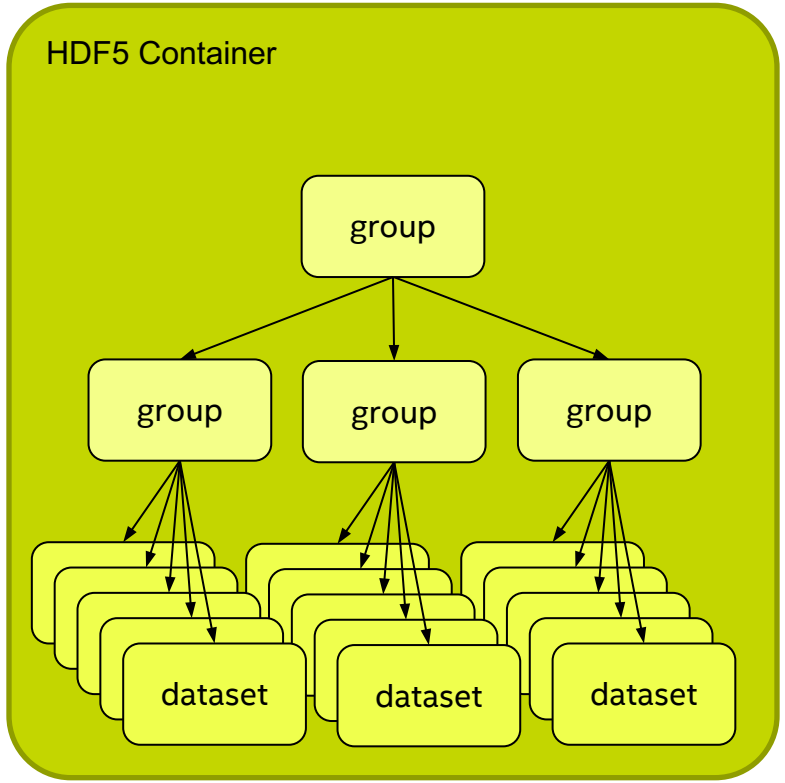
DAOS Ecosystem

Application
I/O Middleware
Storage Backend

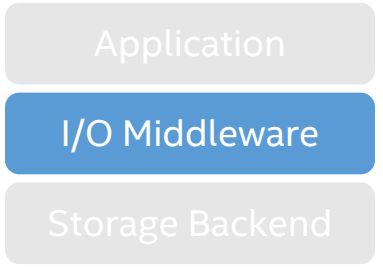


HDF5

- HDF5 file = DAOS container
- DAOS VOL Plugin with smooth migration path
  - Minimal application changes
  - Application no longer needs to explicitly control transactions
    - HDF5 library manages DAOS transactions internally
      - Consistency from H5Fopen() to H5Fflush()/H5Fclose()
    - User-controlled transactions available as extension
- Ported two high-level HDF5-based I/O libraries
  - *NetCDF4*
  - Parallel I/O (*PIO*)



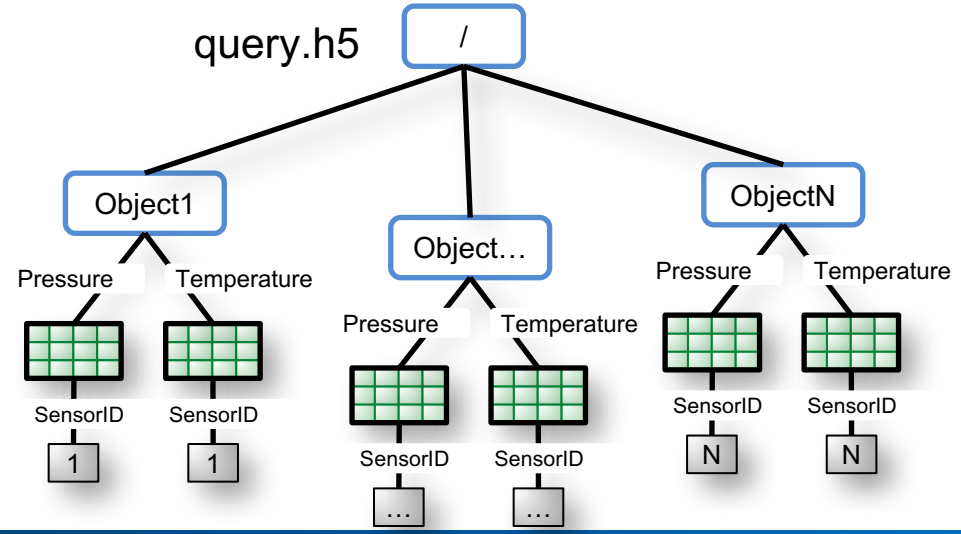
HDF5 Extensions

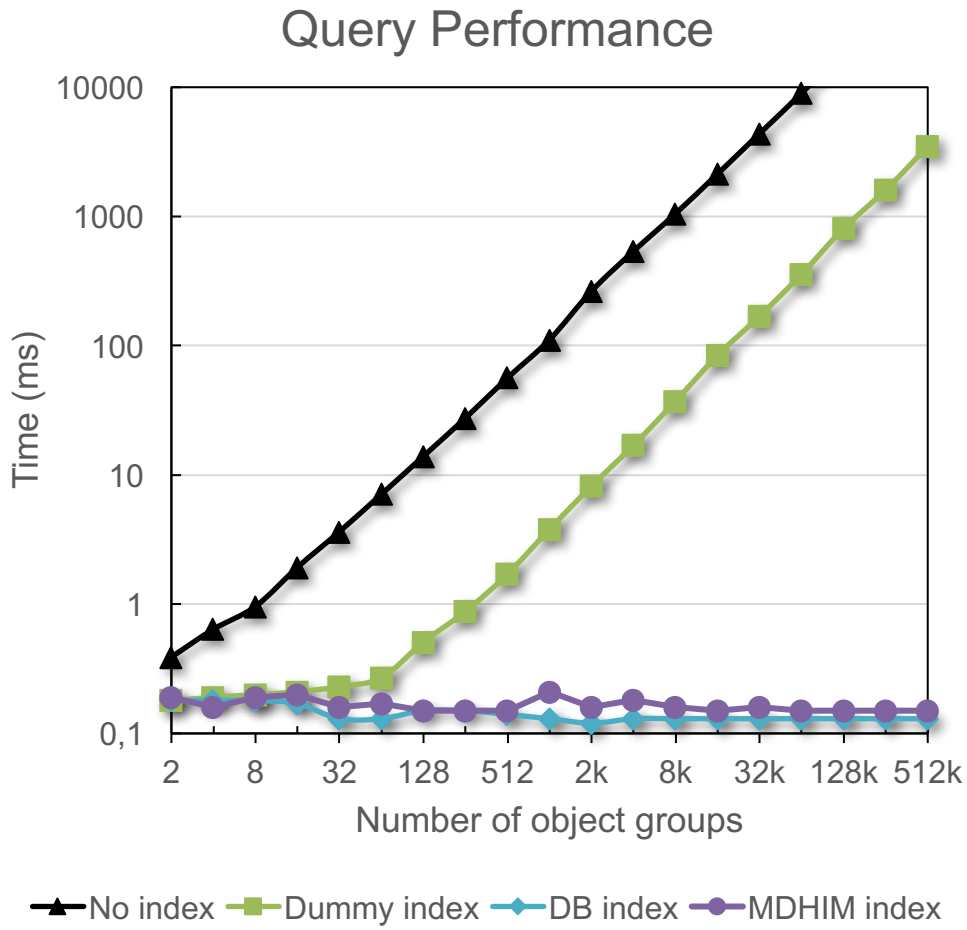


- Asynchronous I/O
  - Focus on ease-of-use
    - Operations still synchronous by default
    - Dependencies tracked by library
  - Extend API with HDF5 “context”
    - Structure for managing and monitoring multiple operations
    - Application can test/wait on individual operations if desired
- End-to-end data integrity
  - Applications can provide checksum buffer to HDF5
    - Receive checksum on write, return value on read
    - Optional
  - HDF5 passes down to DAOS
    - Stored & verified on read/scrubbing
- User-controlled transactions
  - Finer grained control of atomic updates to an HDF5 file
  - Export transaction capabilities to applications through HDF5 API
  - Can pipeline transactions
  - Loosely-coupled execution
- Snapshots
  - Create/open/destroy persistent snapshot of an HDF5 file

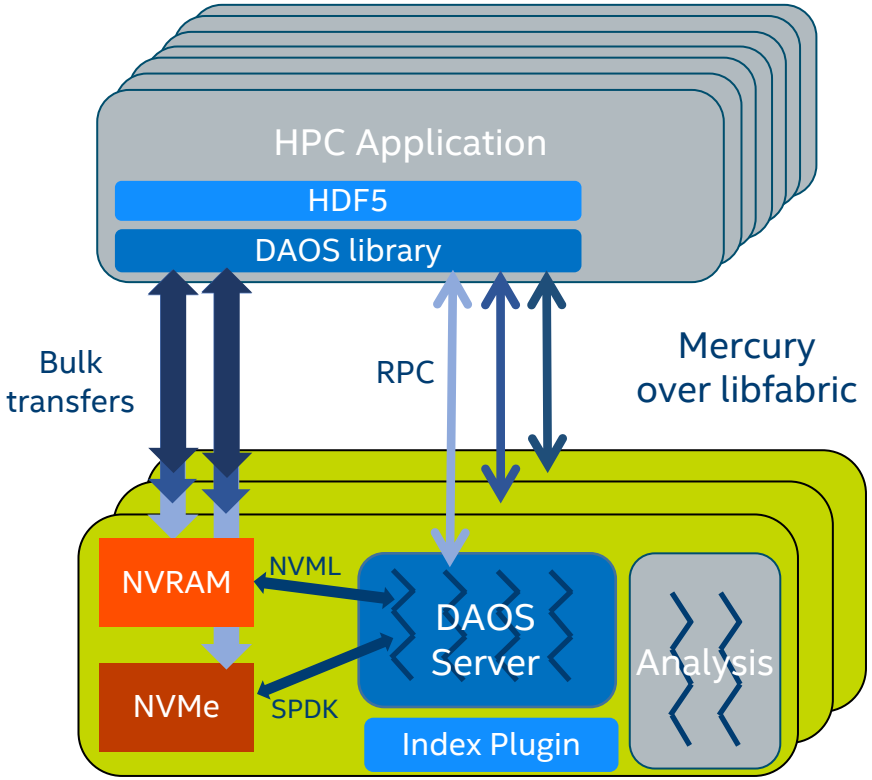
Query & Indexing

- Query API (H5Q)
  - View created in memory
  - Support cross-container queries
- Index API (H5X) accelerates generation of view
  - Data Index (FastBit)
  - Metadata Index (Berkeley DB / MDHIM)
  - Plugin framework





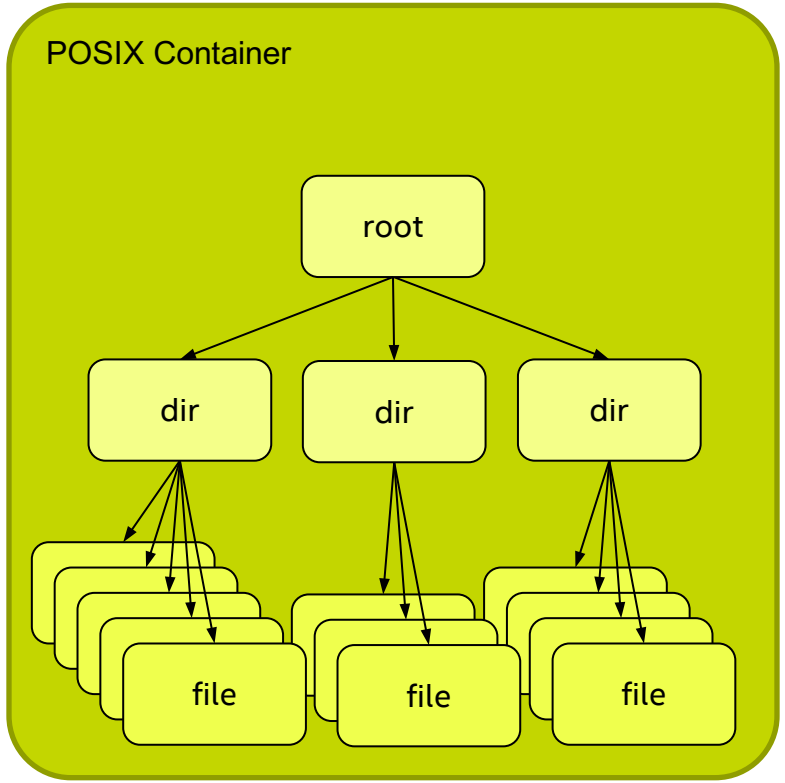
Data Analysis



- Analysis shipping
  - Send user-defined Python script to storage service
  - Execute MapReduce-like operations where data is located
  - Send results back to caller

POSIX Namespace Encapsulation

- POSIX Namespace = DAOS container
  - Full OS bypass for both data & metadata
- Application-private namespace
  - Highly scalable metadata operation
  - Relaxed POSIX compliance



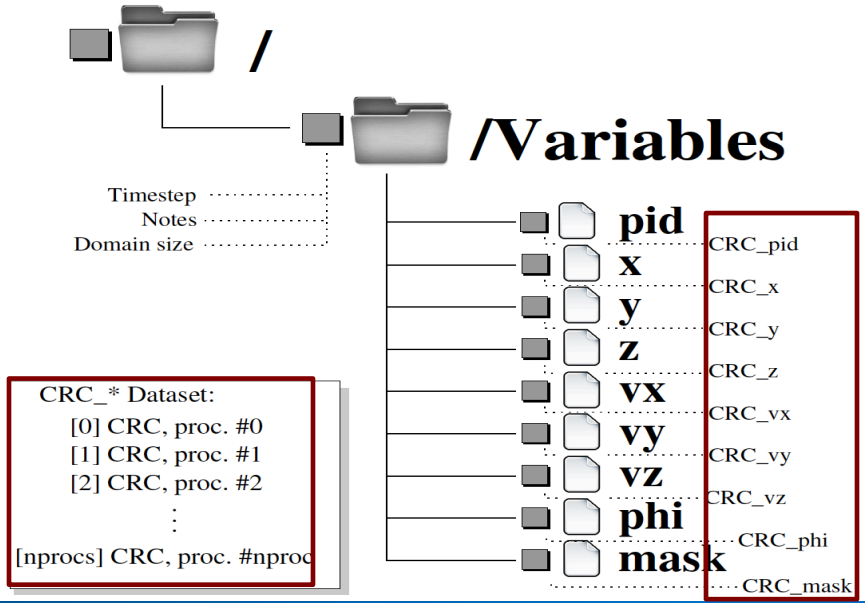
Legion on ESSIO Stack

- *Legion* attaches to HDF5 files using low-level Realm runtime
  - Realm maps memory objects to HDF5 file/objects
- New DAOS capabilities featured in Legion
  - Multiple independent container handles
  - Flexibility with epoch updates from independent container handles
  - Ability to read from un-committed epochs within a container handle
- Using *Legion* with DAOS
  - Individual tasks can be scheduled independently
  - Tasks can complete at various time
  - Legion can schedule different phases of I/O based on data dependency
    - May have reading and writing tasks run concurrently on the same logical region

HACC

Hardware/Hybrid **A**ccelerated **C**osmology **C**ode

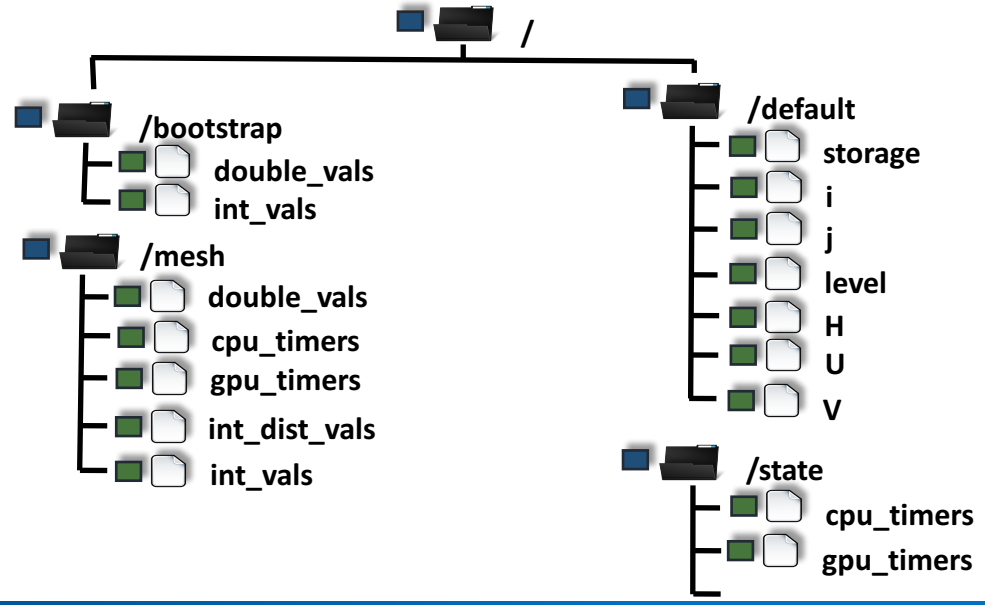
- Updated to store application metadata in HDF5 container
- Ported to rely on end-to-end data integrity feature from the ESSIO stack
- Improved fault tolerance by storing transactional checkpoints



CLAMR

Ce**l**l-Based **A**daptive **M**esh **R**efinement

- Updated to use HDF5 instead of POSIX I/O
- Added restart capabilities using HDF5
- Demonstrated CLAMR restart capabilities over the ESSIO stack



Questions?

Contact: johann.lombardi@intel.com

Resources:

- Git repository
  - https://github.com/daos-stack
- High Level Design (HLD) document
  - https://wiki.hpdd.intel.com/download/attachments/36966823/MS54_CORAL_NRE_DAOSM_HLD_v2.3_final.pdf?version=1&modificationDate=1460746910000&api=v2