Exascale Features in HDF5

Suren Byna

ECP ExaIO project

- Quincey Koziol, Houjun Tang, and Tony Li (Lawrence Berkeley National Laboratory)
- Scot Breitenfeld, John Mainzer, Dana Robinson, Jordan Henderson, Neil Fortner, Jerome Soumagne, Richard Warren, Neelam Bagha, and Elena Pourmal (The HDF Group)
- Venkat Vishwanath and Huihuo Zheng (Argonne National Laboratory)
- Michela Becchi and John Ravi (North Caroline State University)

March 30th, 2021

Summary of ECP HDF5 features

- HDF5 Virtual Object Layer (VOL)
  - A layer just below the HDF5 API to allow new storage methods
- Asynchronous I/O overlaps I/O operations with compute operations and hides I/O latency
- Caching and prefetching to use node-local memory and storage to reduce the performance gap between memory and long-term storage
- Subfiling allows writing data to multiple HDF5 files instead of a single large file
- Efficient I/O between GPU and storage
  - GPU direct storage (GDS) Virtual file driver, and extension to asynchronous I/O to move data between GPUs and storage

Thanks and contact info

- **Contacts**
  - Suren Byna (LBNL) SByna@lbl.gov
  - Scot Breitenfeld (The HDF Group) brtnfld@hdfgroup.org
  - Quincey Koziol (LBNL - NERSC) koziol@lbl.gov
  - Elena Pourmal epourmal@hdfgroup.org

HDF5 Community BOF

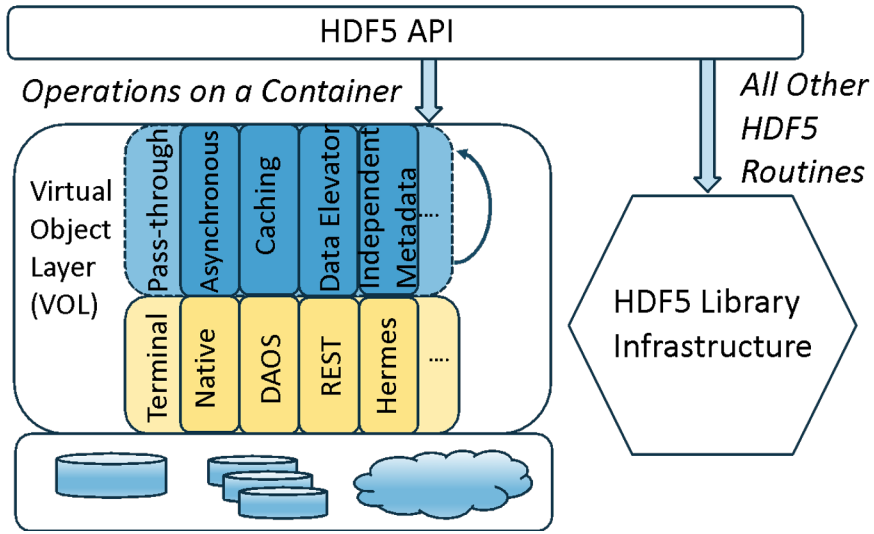
March 30th (Today)
@ 3pm ET

HDF5 User Support:
HDF Helpdesk: help@hdfgroup.org
HDF Forum: https://forum.hdfgroup.org/

Details about ECP HDF5 features and results

Virtual Object Layer to open the HDF5 API to other storage models

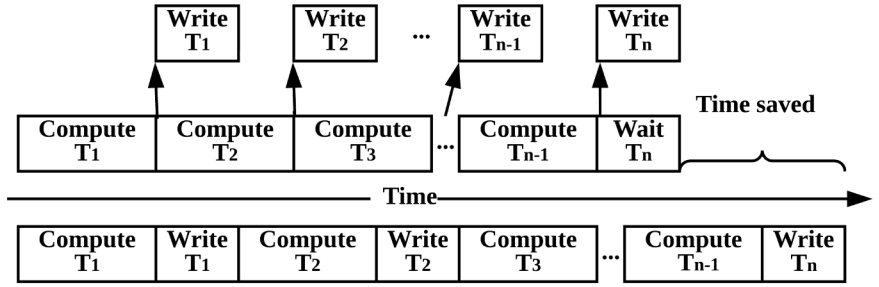
- Virtual Object Layer (VOL) provides an application with the HDF5 data model and API, but allows different underlying storage mechanisms
- Enables developers to use HDF5 on novel current and future storage systems easily
- Usage: VOL connectors to be used are specified with environment variables
- Several VOL connectors are already available, with more in development
  - Async I/O, Caching (using deeper memory and storage hierarchy), Data Elevator, provenance, etc.
  - DAOS VOL for using DAOS system
  - ADIOS VOL for writing and reading BP format files, log-structured VOL from the DataLib team

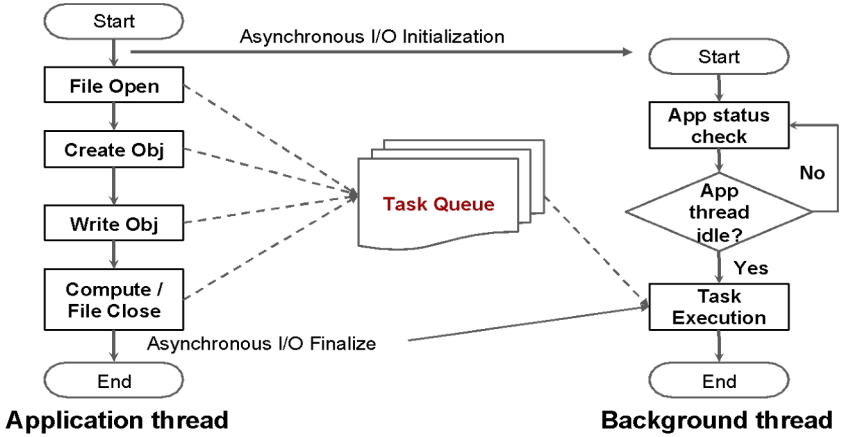


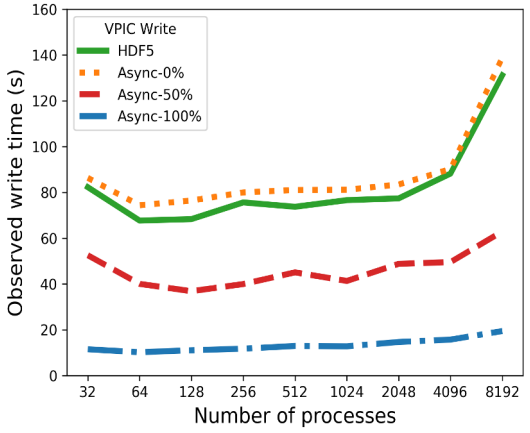
Released in HDF5 1.12.x (Current public release)

Hiding I/O latency with asynchronous I/O

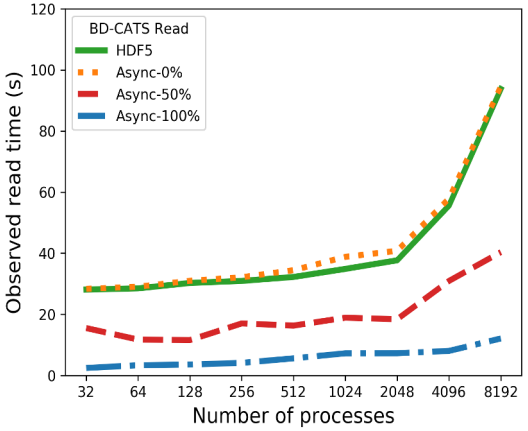
- Overlaps I/O operations with compute operations and hides I/O latency
- Pass-through VOL connector w/background threads, using Argobots
- Implicit: transparent by setting environment variable
- Explicit: For applications that want more control of async operations







VPIC-IO on Cori



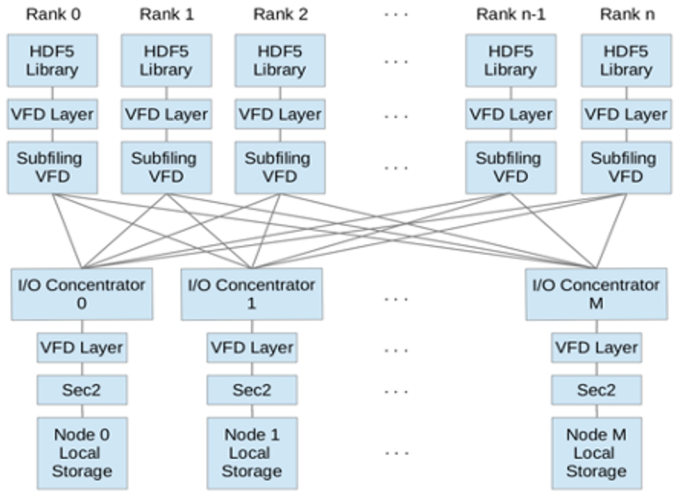
BD-CATS-IO on Cori

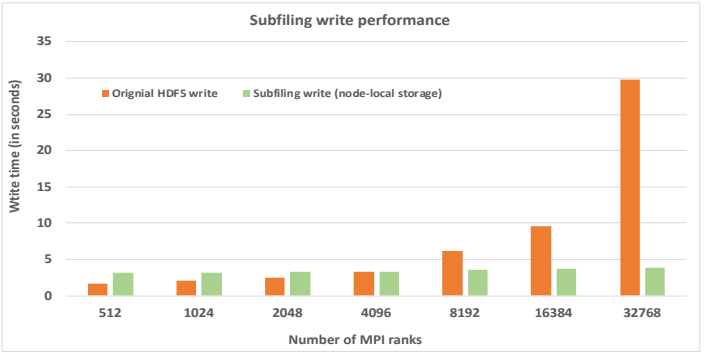
https://github.com/hpc-io/vol-async

To be released with HDF5 1.13.0

Subfiling to write data to multiple HDF5 files

- Writing to single shared file is slow, due to file system lock contention
- Subfiling allows applications to write data to multiple small files (N procs writing to M files, N >> M)
  - Plus a metadata file stitching the small files together
- Sub-filing Virtual File Driver (VFD) and VOL connector with I/O concentrators writing files to storage
- Current status
  - Initial prototype using “vector I/O” VFD and a subfiling VOL
  - Designing “selection I/O” VFD for further optimizations
  - Testing and tuning performance with the current prototype
  - Exploring caching options (cache VOL as well as UnifyFS) for storing intermediate data





Preliminary results on Summit are promising

I/O directly from GPUs

- GPUs are becoming workhorses of HPC computing
- File I/O to move data between GPUs and storage devices becomes critical
- HDF5 team efforts (with contingency funding):
  - Virtual File Driver (VFD) for NVIDIA's GPU Direct Storage (GDS)
    - Performance benefits with larger data sizes
    - Integrated in HDF5 (https://github.com/HDFGroup/hdf5/tree/cu_dev)
  - Asynchronous data movement between GPUs and CPUs, and between CPUs and storage
    - Designed a set of benchmarks and ongoing optimizations
    - Initial results show significant benefit when overlapping write time and transfers between CPU and GPU
    - Designing integration with HDF5 using async and cache VOL connectors
    - More testing on GPUs from more vendors
    - Considering RAJA, Kokkos, HIP, Sycl, One API, etc.