h5bench: HDF5 I/O Kernel Suite for Exercising HPC I/O Patterns

Tonglin Li, Suren Byna, Quincey Koziol, Houjun Tang, Jean Luca Bez, and Qiao Kang

Lawrence Berkeley National Laboratory
North Carolina State University

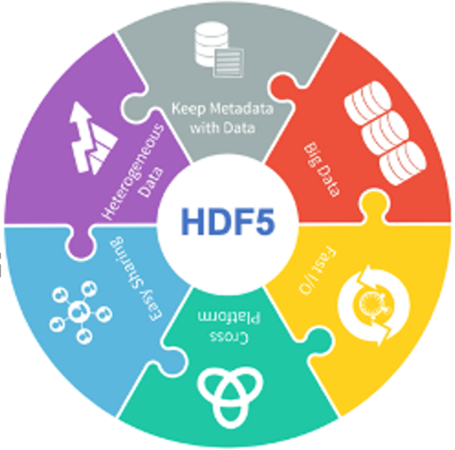
May 3rd, 2021
Cray User Group (CUG) meeting 2021

Agenda

- Brief intro to HDF5
- Existing HDF5 benchmarks
- Issues and requirements
- h5bench
- Evaluation on Cori
  - Write benchmark
  - Read benchmark

HDF5 - I/O API, self-describing file format

- Self-describing file format, API, and tools designed to store, access, analyze, share, and preserve diverse, complex data in continuously evolving heterogeneous computing and storage environments
    - Maintained and developed by The HDF Group in collaboration with the ECP ExaIO team
    - https://github.com/HDFGroup/hdf5
- Heavily used library on DOE supercomputing systems
- Many ECP AD & ST teams have critical dependency on HDF5



Existing HDF5 performance benchmarks

- h5perf
- Application specific benchmarks
  - Flash-IO benchmark
  - HACC-IO
  - …
- I/O patterns
  - IOR has a HDF5 output
  - MinIO
  - PIOK (Parallel I/O kernels)
    - VPIC-IO, BD-CATS-IO, VORPAL-IO, GCRM-IO
  - …

Issues with existing benchmarks

- Coverage of access patterns is sparse
- New HDF5 optimizations are not implemented
  - Asynchronous I/O
  - Caching
  - I/O between GPU and storage
- Other tuning parameters are not exercised
  - MPI-IO aggregation strategies (collective buffering settings)
  - File system-specific tuning parameters (alignment, striping, etc.)

Requirements

- Coverage of representative app I/O patterns (read / write, data / metadata, locality)
- Scalable
- Exercise new HDF5 features
- Tuning parametersI/O software layers
  - File system layer

BERKELEY LAB
Lawrence Berkeley National Laboratory

ECP EXASCALE COMPUTING PROJECT

h5bench - A suite of HDF5 benchmarks

- Captures various I/O patterns
  - Locality in memory and in files
    - Contiguous, strided, compound data types (structures)
  - Array dimensionality - 1D, 2D, and 3D
- I/O modes
  - Synchronous
  - Asynchronous - Implicit and explicit
    - to overlap I/O time between successive compute phases
- Processor type - CPUs and GPUs
- MPI-IO modes
  - Collective buffering on or off
- File system configuration
  - Alignment and striping

Other configurable options in h5bench

- Several configurable parameters
  - Scale -- number of MPI processes
  - Data size per MPI rank
  - Dimensionality
  - For read benchmark - read all data, partial data, random data
  - Emulated computation phase time
  - Memory limit for double-buffering in asynchronous I/O
  - MPI-IO collective buffering
- Metadata stress testing benchmarks

Code repository and usage instructions

- Available on GitHub for public access
  - https://github.com/hpc-io/h5bench
  - README.md has instructions to install, configure, and use
  - Contact: sbyna@lbl.gov / koziol@lbl.gov
- GPU benchmarks are in a separate branch
  - PR and code review in progress
- Open source to add new benchmarks or to fix current code
  - Communicate with us about any new benchmarks to add
  - Fork, add new HDF5 benchmarks / modify / bug fixes, and submit a PR

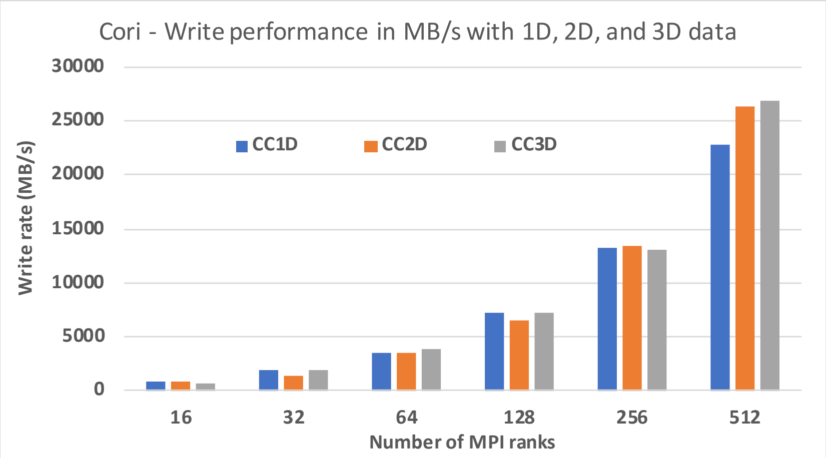
Early evaluation on Cori

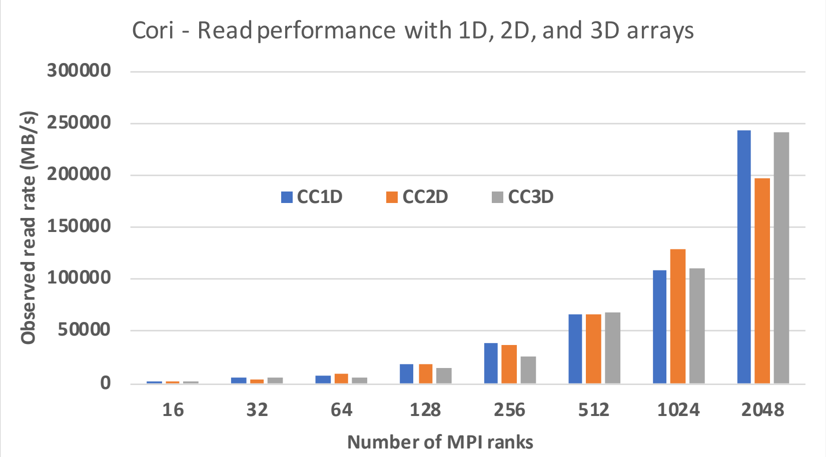
- Cori - Cray XC40 system at NERSC
- Configuration used in early evaluation
  - Haswell partition with 32 cores per node
  - Lustre parallel file system
- I/O Patterns
  - Write benchmarks at various scales
    - Dimensionality - 1D, 2D, and 3D
    - Locality - memory and file offsets
    - Synchronous and asynchronous I/O
  - Read benchmarks
    - Reading the entire data - synchronous and asynchronous
    - Reading a subset of data

Write benchmarks - 1D, 2D, 3D

- Contiguous in memory and in file
- 8 million particles per MPI rank (weak-scaling)
- 8 variables per particle
- Organized as 1D, 2D, and 3D

Initial results show similar performance for contiguous writes (in memory and in file) of 1D, 2D, and 3D arrays





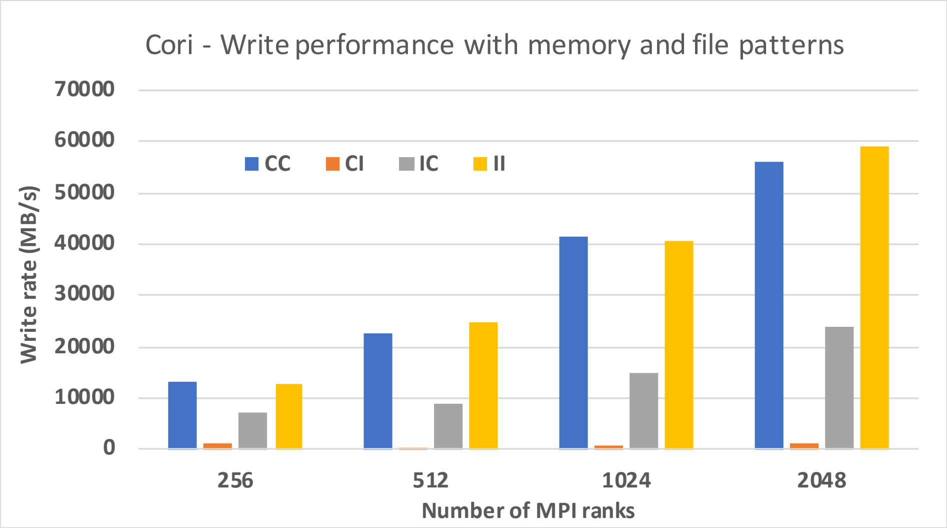
BERKELEY LAB
Lawrence Berkeley National Laboratory

Write benchmarks - Memory and file patterns

- Memory and file contiguity
- Contiguous - Memory buffers map to HDF5 file datasets directly
- Noncontiguous
  - Memory buffer is a C struct
  - HDF5 dataset is a compound datatype (C struct-like)

Constructing a compound datatype achieves poor performance

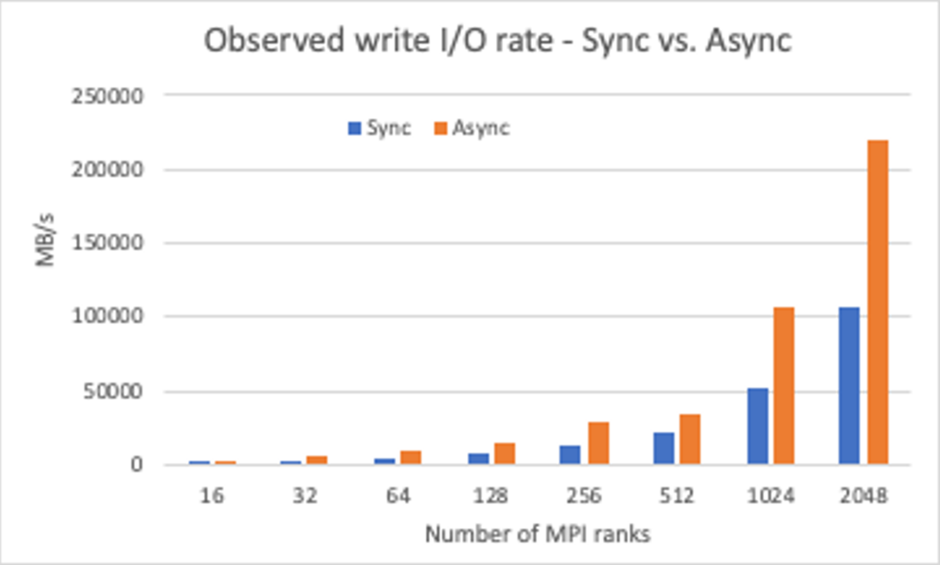
Direct mapping of memory and file data structures achieves good performance



Write benchmarks - Sync vs. Async I/O

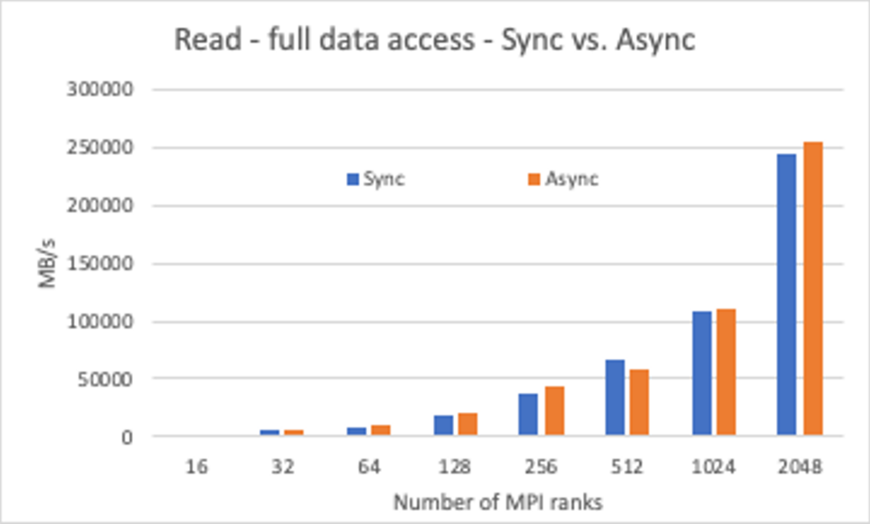
- Writing 5 time steps of data
- Computation phases between I/O phases
- Workloads similar to simulations that periodically checkpoint or write their memory state

I/O time is effectively overlapped by computation phases



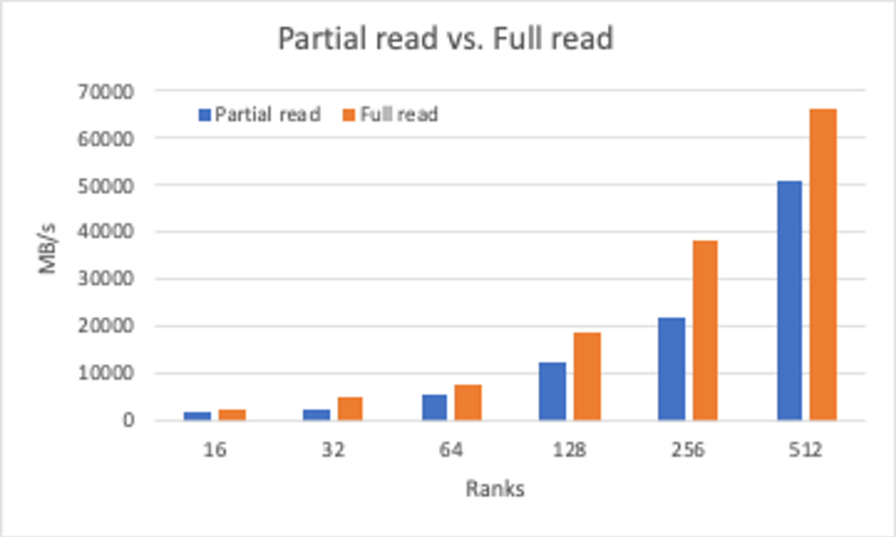
Read benchmarks - Full data access

- Reading all HDF5 datasets that write benchmark stored
- 8 variables, 8 million particles per rank
- Single time step
- Synchronous and asynchronous modes
  - Read right after writing
  - Caching effects have not been isolated
  - Further testing is in progress



Read benchmarks - Partial

- Read partial amount of data
  - First 10% particle data in each variable
- Reading full data achieves better I/O rate
  - Contiguous and large accesses
  - Further testing is in progress



Conclusions

- h5bench
  - Provides a wide variety of I/O patterns
  - Allows weak-scaling and strong scaling
  - Exercises new HDF5 features
  - Configurable options for tuning parameters
- Evaluation to understand I/O behavior is in progress (Full paper)
- Future work
  - Add more HDF5 features -- caching on node-local storage, metadata buffering, new file system features (Lustre progressive file layouts, etc.)
  - Add more I/O kernels or RW patterns from ECP and EOD applications -- variable length, streaming, ML / AI workloads

Thanks to ECP, and to the ExaIO HDF5 team (The HDF Group, ANL, LBL)