Hands-on
Parallel HDF5 Tutorial

October 13, 2020

The HDF Group

Scot Breitenfeld and Elena Pourmal, The HDF Group
Quincey Koziol, NERSC

Tutorial Prerequisites

- Tutorial assumes basic knowledge of
  - HDF5
  - MPI and MPI I/O
    - Concept of independent and collective I/O
  - C
- Access to HPC resource is provided by The HDF Group

Tutorial Goals

- Give a quick introduction to parallel HDF5 capabilities
- Help to avoid common mistakes when using parallel HDF5 library
  - For more information, check The HDF Group support portal and recent Webinars
    https://portal.hdfgroup.org/display/HDF5/HDF5
    https://portal.hdfgroup.org/display/HDF5/Introduction+to+Parallel+HDF5
    https://www.hdfgroup.org/category/webinar/
- Provide an opportunity to explore state-of-the-art HPC cluster

Tutorial Outcome

- You should have a better understanding of
  - How to use parallel HDF5 library
  - When to use collective or independent raw data I/O
  - How to avoid HDF5 “metadata storm”

Tutorial programs and how to run them

- Get Tutorial examples

    git clone https://github.com/HDFGroup/Tutorial.git

- README.txt has info about each example program

- To compile examples

```
spack load --first hdf5
make
```

- To run examples

```
srun -n 4 <exec_name>
```

Let’s start!

Parallel HDF5 library capabilities

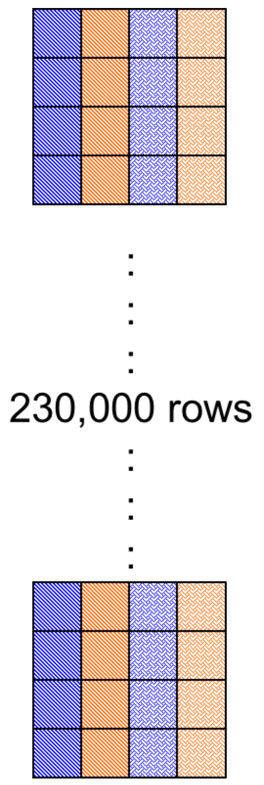
- Raw data I/O
  - All MPI ranks in communicator can write and read to / from the same or different datasets
- HDF5 metadata operations
  - All ranks in MPI communicator can access all objects in the HDF5 file
- Parallel HDF5 library limitations
  - Can’t create or write variable-length data
  - Doesn’t support SWMR mode
  - Doesn’t support independent modifications of HDF5 file structure (HDF5 metadata)

It is easy to start using parallel HDF5: `h5_ex0.c` and `h5par_ex0.c`

- Steps to convert HDF5 application to use the parallel HDF5 library
  - Convert an application to use MPI
  - Update `H5Fcreate/open` to use HDF5 *MPI I/O* virtual file driver (VFD)
  - Use `h5pcc` compilation script to build application
    - (found in the `bin` directory of parallel HDF5 library installation)
  - Run as you would run parallel applications on your HPC system
- Try it:
  - Compare the content of h5_ex0.c and h5par_ex0.c with `diff` command
  - Run both examples
  - Use `h5dump` utility to examine `SDS.h5` and `SDSpar.h5`
- Points to remember
  - There is no difference in HDF5 files created by the sequential or parallel HDF5 library
  - Use HDF5 MPI I/O VFD to write / read HDF5 file in parallel

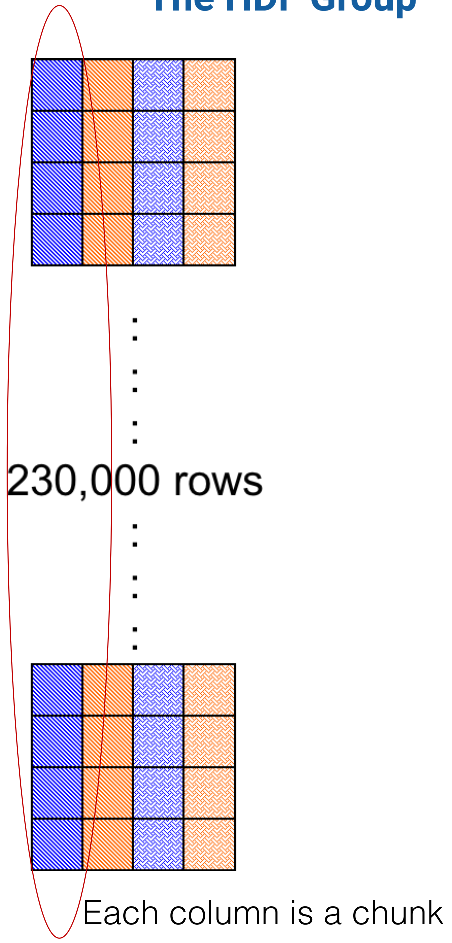
Writing HDF5 dataset by columns: `h5_ex1.c, h5par_ex1a.c and h5par_ex1b.c`

- How to convert HDF5 application to use parallel HDF5 library
  - Assign data to each MPI rank (vs. looping through parts of data to write / read) using HDF5 hyperslab selection
- Try it:
  - Compare the content of `h5_ex1.c` and `h5par_ex1a.c`
    - Notice that each hyperslab represents noncontiguous selection in the file
  - Run both examples
  - Compare the content of `h5_ex1a.c` and `h5par_ex1b.c`
  - Run `h5par_ex1b.c` and compare the results
- Points to remember
  - Default properties may not be optimal when doing raw data I/O
  - Consider using **`H5Pset_dxpl_mpio`** and **`H5FD_MPIO_COLLECTIVE`** to set collective I/O for `H5Dwrite/read`



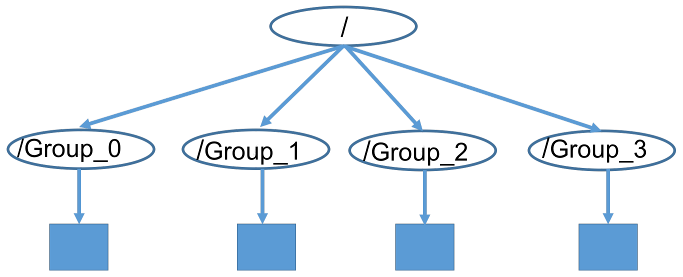
Writing HDF5 dataset by columns using chunking and compression `h5par_ex1c.c` and `h5par_ex1d.c`

- Try it:
  - Compare the content of `h5_ex1a.c` and `h5par_ex1c.c`
  - Run `h5par_ex1c.c`
  - Compare the content of `h5_ex1c.c` and `h5par_ex1d.c`
    - See how to set compression for parallel applications
  - Run `h5par_ex1d` and check the sizes of `h5par_ex1c.h5` and `h5par_ex1d.h5`
- Points to remember
  - I/O is performed on the *whole* chunk
  - Use chunks of at least 10-100MB (limit for chunk size is 4GB)
  - Think about chunk sizes if data must be read back
  - Compression requires *collective* raw data I/O



Creating HDF5 file structure: `h5par_ex2a.c` and `h5par_ex2b.c`

The HDF Group



- Try it:
  - Let's try to create the file structure shown here
  - Compare the content of `h5_ex2a.c` and `h5par_ex2b.c`
  - Run `h5par_ex2a` and `h5par_ex2b`
  - Run `h5dump -H h5par_ex2a.h5` and `h5dump -H h5par_ex2b.h5`
- Points to remember
  - Any operations that change HDF5 file structure or HDF5 metadata requires collective operations
  - See HDF5 documentation, General Topics in HDF5, Parallel HDF5 https://portal.hdfgroup.org/display/HDF5/Collective+Calling+Requirements+in+Parallel+HDF5+Applications

Reading HDF5 file structure: `h5par_ex2c.c`

- Try it:
  - See `h5par_ex2c.c` for how to read HDF5 metadata independently
  - Run `h5par_ex2c`
- Points to remember
  - `H5Fopen` is ***always a collective*** *call*
  - `H5G/D/Aopen` calls ***can be independent*** **if the file structure is not modified**

Using independent H5Dopen to write data from each MPI rank: `h5par_ex2d.c` `h5par_ex2e.c`

- Try it:
  - Review `h5par_ex2d.c` and `h5par_ex2e.c` for how to create datasets and to write them independently from each MPI rank
  - Run `h5par_ex2d` and `h5par_ex2e`
  - Use `h5dump h5par_ex2d.h5` to check the content
- Points to remember
  - Use **`H5Pset_alloc_time`** dataset creation property and **`H5D_ALLOC_TIME_EARLY`** to allocate space in the file
  - Shown approach cannot be applied to *extensible* and *compressed* datasets

Avoid MD storm writes and reads: `h5par_comparison`

- Try it:
  - Review `h5par_comparison.c`; it creates a dataset and writes it with hyperslabs using all ranks
  - Notice that multiple ranks access the ***same*** HDF5 metadata when creating a file and a dataset, and again when opening the file and the dataset. (Performance may suffer!)
  - Run `h5par_comparison`, `h5par-comparison-collmd` and `h5par-comparison-collio`
- Points to remember
  - When all ranks access the same objects in HDF5 file avoid "metadata write/read storm" by using **`H5Pset_all_coll_metadata_ops`** and **`H5Pset_coll_metadata_write`**
  - See https://portal.hdfgroup.org/display/HDF5/General+Access+Properties
  - Other hints:
    - Use HDF5 1.10.7 and later to use optimizations when reading the entire dataset by all MPI ranks
    - Use compact storage for a small dataset (order of KBs) and when reading/writing the dataset by all MPI ranks along with optimization above to avoid metadata storm

THANK YOU!

Questions & Comments?