

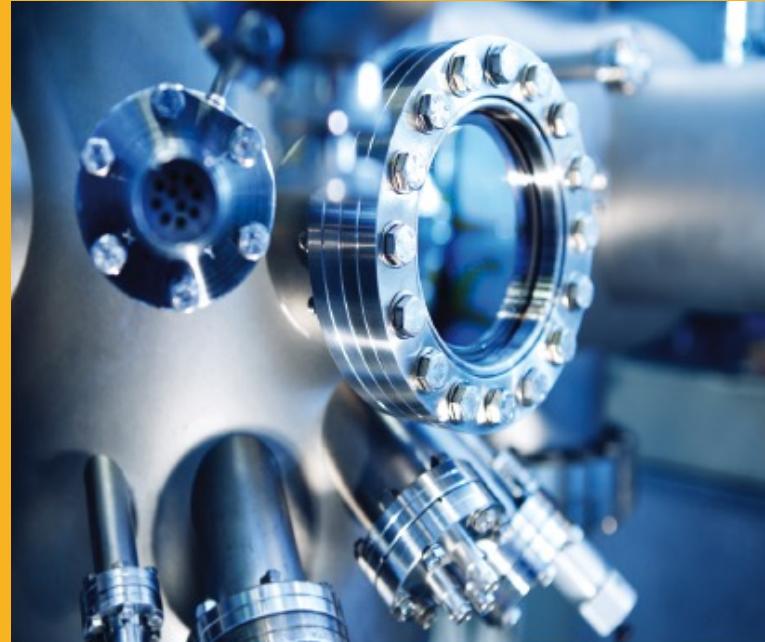


Parallel HDF5 and compression filters with synchrotron scattering data

Zdenek Matej, Andreas Mattsson
MAX IV Laboratory, Lund University

Ingredients

- parallel HDF5 with MPI
- image like synchrotron data
- compression

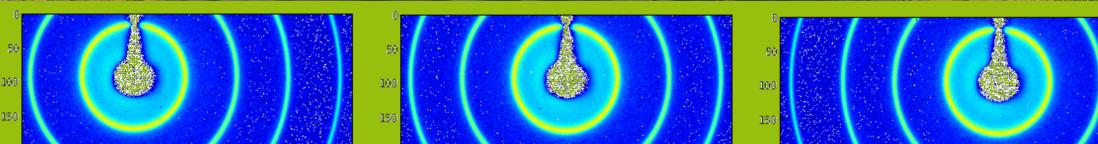


Parallel HDF5 with MPI

- message passing interface (MPI) is a **standard for parallel computing** architecture; invented in 1991-1992; popular at high-performance-computing (HPC) clusters
- Sophie Servan (DESY) et al., **Technical-Workshop-Survey (2020)**: 9/10 facilities have SLURM HPC cluster
<https://github.com/ExPaNDS-eu/ExPaNDS/blob/master/WP4/20201009-Technical-Workshop-Survey.pdf>
- “reference” HDF5 library implementation
 - a complex sw and have numerous technical limitations one should be aware of
 - MPI enables distributed multi-task application can effectively read/write data to a single HDF5 file and even dataset
 - well defined parallel HDF5 library and compatible software (h5py/mpi4py) are available off-the-shelf in HPC sw distributions (e.g. EasyBuild)
 - parallel HDF5 is an “old” feature, i.e. 1.8
 - **sw**: savu (Diamond), PtyPy - both are mpi4py & h5py
- alternatives with serial HDF5
 - virtual datasets distributed over multiple files
 - direct chunk write/read
 - disadvantage: 1.10 required, some sw cannot read such data

Image like synchrotron data

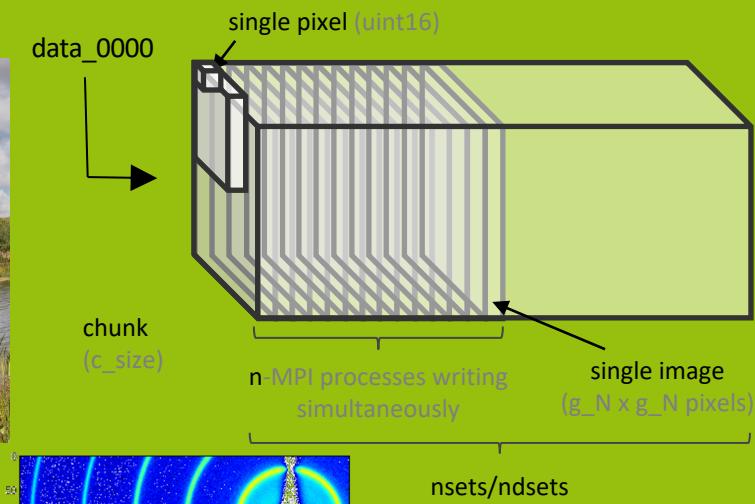
- general synchrotron data have complex structure/format
- “detector” data (99% volume of all data): usually 2D or 3D datasets
- here focus on performance
 - example JungFRAU detector (PSI)
 - 4M pixels (2 bytes depth) x 2kHz
 - **16 GB/s uncompressed, bslz compression factor ~ 4x, finally 4 GB/s compressed**



4

2021-07-08

European HDF Users Group Summer 2021



pHDF5, image like data, no compression

tools

- tools: h5perf_parallel, ior (-a HDF5), hdf5_pwrite3dc, h5py-snippets/gist
- [hdf5_pwrite3dc](#) is deeply inspired by code from *Timothy Brown* on [Stackoverflow](#)

```
1  from mpi4py import MPI
2  import h5py
3  import numpy as np
4  import sys
5
6  def _create_dataset_nofill(group, name, shape, dtype, chunks=None):
7      spaceid = h5py.h5S.create_simple(shape)
8      plist = h5py.h5P.create(h5py.h5P.DATASET_CREATE)
9      plist.set_fill_time(h5py.h5D.FILL_TIME_NEVER)
10     if chunks not in [None, ()] and isinstance(chunks, tuple):
11         plist.set_chunk(chunks)
12     typeid = h5py.h5T.py_create(dtype)
13     datasetid = h5py.h5D.create(group.file.id, group.name+'/'+name, typeid, spaceid, plist)
14     dset = h5py.Dataset(datasetid)
15     return dset
16
17 n_frames = 6000
18 ch_n = g_n = 4096
19 comm = MPI.COMM_WORLD # common communicator
20 n_tasks = comm.size # number of tasks
21 rank = comm.rank # process ID
22
23 # prepare local data
24 data = np.ones((g_n,g_n),dtype=np.int32)
25
26 # set collective transport property
27 dxpl = h5py.h5P.create(h5py.h5P.DATASET_XFER)
28 dxpl.set_dxpl_mpio(h5py.h5fd.MPIO_COLLECTIVE)
29
30 f = h5py.File(sys.argv[1], 'w', driver='mpio', comm=MPI.COMM_WORLD)
31
32 dset = _create_dataset_nofill(f['/'],'test',(n_frames,g_n,g_n), dtype=np.int32, chunks=(1, ch_n, ch_n))
33
34 for i in np.arange(rank,n_frames,n_tasks):
35     data *= i
36     dset[i] = data
37
38 f.close()
```

workaround to set
FILL_TIME_NEVER in h5py
[h5py issue #1282](#)

code

pHDF5, image like data, no compression

typical figures

	compression	write [GB/s]	read [GB/s]	ntasks	comment
ior	no	5.7	4.1	8	1 x FDR
pwrite	no	5.8	4.8	8	1 x FDR
pwrite	no	10.9	8.2	8	2 x FDR
h5py-parallel	no	4.5	-	16	1 x FDR
h5py-serial (dask)	no	-	5.4	16	1 x FDR

- Nice ! It works. Write/read rates limited mainly by bandwidth to storage.

pHDF5, image like data, compression filter

typical figures

	compression	write [GB/s]	read [GB/s]	ntasks (omp)	comment
pwrite	no	5.8	4.8	8	1 x FDR
pread-eiger	bslz4	-	23.8 ✓	20(2)	compression rate: 8.1
pwrite-bslz4	bslz4	4.9 (?)	-	24(2)	compression rate: 8.1

- Table: raw i.e. uncopressed data rates
- $4.9 / 8.1 \sim 0.6$ GB/s - storage is likely not a bottleneck
- bslz4 compression is very effective, in particular at least 2x faster
- uncompressed write faster than compressed, **read OK**
- with multiple files, VDS, in-memory bslz4 compression and direct chunk write better performance can be achived
- parallel HDF5 is great but writing data with compression-filters is not so shiny (?)