hdf_pwrite3dc
- writing large series of image-like data into HDF5 dataset

Simple test of MAX IV software and storage capabilities to write series of image-like data into a single file and (single) HDF5 dataset(s) based on MPI IO and parallel HDF5.

Motivation: Many detectors and data processing pipelines at light sources are producing series of 2D image data that are usually saved as 3D HDF5 datasets. HDF5 format is mostly used in order to keep organization of data simple and compact.

Under some circumstances there may be several limitations, mainly on the dataset sizes, and so as ena exception from writing to a single dataset pwrite3dc is adding an option to split the 3D dataset into several datasets with labels: `data_0000`, `data_0001` etc. saved in the single file. Hence user can specify beside the total number "`nsets`" of sets (images) to be saved also the number "`ndsets`" of datasets in which the data will be divided.

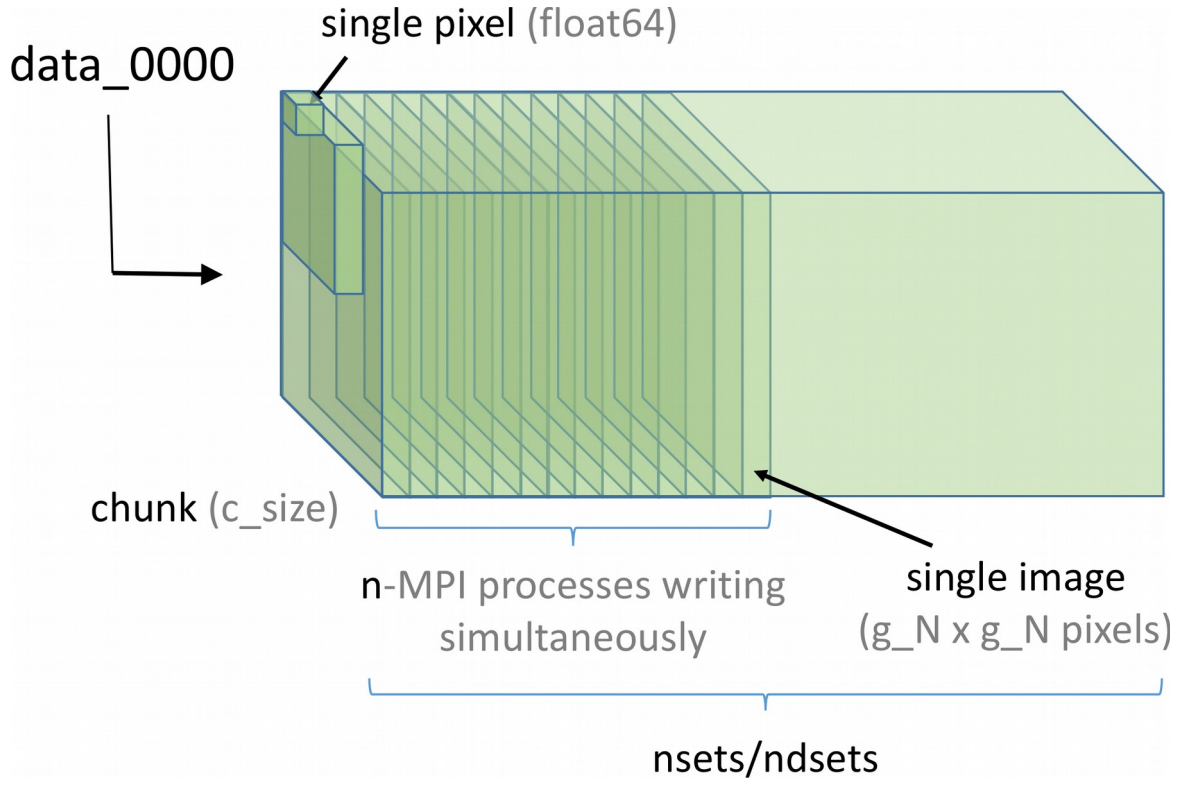

Figure 1: `pwrite3dc` writes in total `nsets` of image like data (`g_N x g_N` 64 bits pixels where `g_N` = 8000) into `ndsets`. Writes are done collectively in chunks (of e.g. 1000 x 1000 pixels) with `n`-MPI processes. There is a loop over `ndsets` and a loop over `nsets/ndsets/n` writes. Writes over chunks and `n`-MPI processes are done by HDF5.

Figure 1 shows a typical data block written. As no padding is supported `MPI rank`, `nsets` and `ndsets` must confirm several constraints:

- `nsets must be divisible by ndsets`
- `nsets must be divisible by (n*ndsets) where n is the MPI rank`

- ratio of nsets to (n*ndsets) can be limited (E.g. in case of g_N=8000 and 8 bytes per pixel it should be at maximum equal to 4. This will ensure that there is not more than 8k*8k*8*4 ~ 2 GB per MPI process)
-

Runnig program:

```
hdf_pwrite3dc filepath nsets ndsets
```

Examples of working configurations:

- n = 1 (MPI rank), nsets = 400, we need ndsets = 100

```
mpirun -n 1 ./hdf_pwrite3dc /gpfs/tetsFile.h5 400 100
```

- n = 5, nsets = 400, ndsets = 20 (20*5 x 4 = 400)

```
mpirun -n 5 ./hdf_pwrite3dc /gpfs/tetsFile.h5 400 20
```

- n = 40, lets use ndset = 3 to get number 120 close to 100 and then we can have at maximum nsets = 480

```
mpirun -n 40 ./hdf_pwrite3dc /gpfs/tetsFile.h5 480 3
```

- n = 200, we have many MPI processes we can afford to write to ndset = 1 as many as nsets = 800 but we can write less e.g. nsets = 400

```
mpirun -n 200 ./hdf_pwrite3dc /gpfs/tetsFile.h5 400 1
```

Program output:

Each MPI process runs over a series of checkpoints. After checkpoint no=8 it starts h5dcreate, after checkpoint no=9 it invokes h5dwrite, before checkpoint no=10 h5dclose is called and so the critical work should be done. The parts with h5dcreate and h5dwrite are limited by barriers and 2 different time periods are measured (“creation” and “write”). And so 2 values are returned at the end.

```
bash-4.2$ module purge
bash-4.2$ module load GCC/5.4.0-2.26 OpenMPI/1.10.3 HDF5/1.10.0-patch1
bash-4.2$ rm -f hdf_pwrite3dc
bash-4.2$ make hdf_pwrite3dc
h5pfc -O3  kinds.f90 hdf_pwrite3dc.f90 -o hdf_pwrite3dc

mpirun --map-by node -n 80 --oversubscribe --bind-to hwthread ./hdf_pwrite3dc
/gpfs/gpfs8m/zdenek-sw-test/testFile.h5 240 1
. . .
 Process:          76 , chckpoint:          10
 Process:          78 , chckpoint:          10
 Process:          25 , chckpoint:          10
 Process:          29 , chckpoint:          10
 Process:          62 , chckpoint:          10
 Process:          60 , chckpoint:          10
 elapsed time(create):    152.360001
 elapsed time( write):    64.5220032
saved 122880.0 (MB), rate(create): 806.510 (MB/s), rate(write): 1904.467 (MB/s)
```

Figure 2 shows a typical run of pwrite3dc. Two stages are distinguishable. During the first one the hdf5 file is created and you see purely write throughput, during the second stage there are reads for the data written to the file. Each MPI process is saving its MPI-id into the 2D dataset (image) saving.

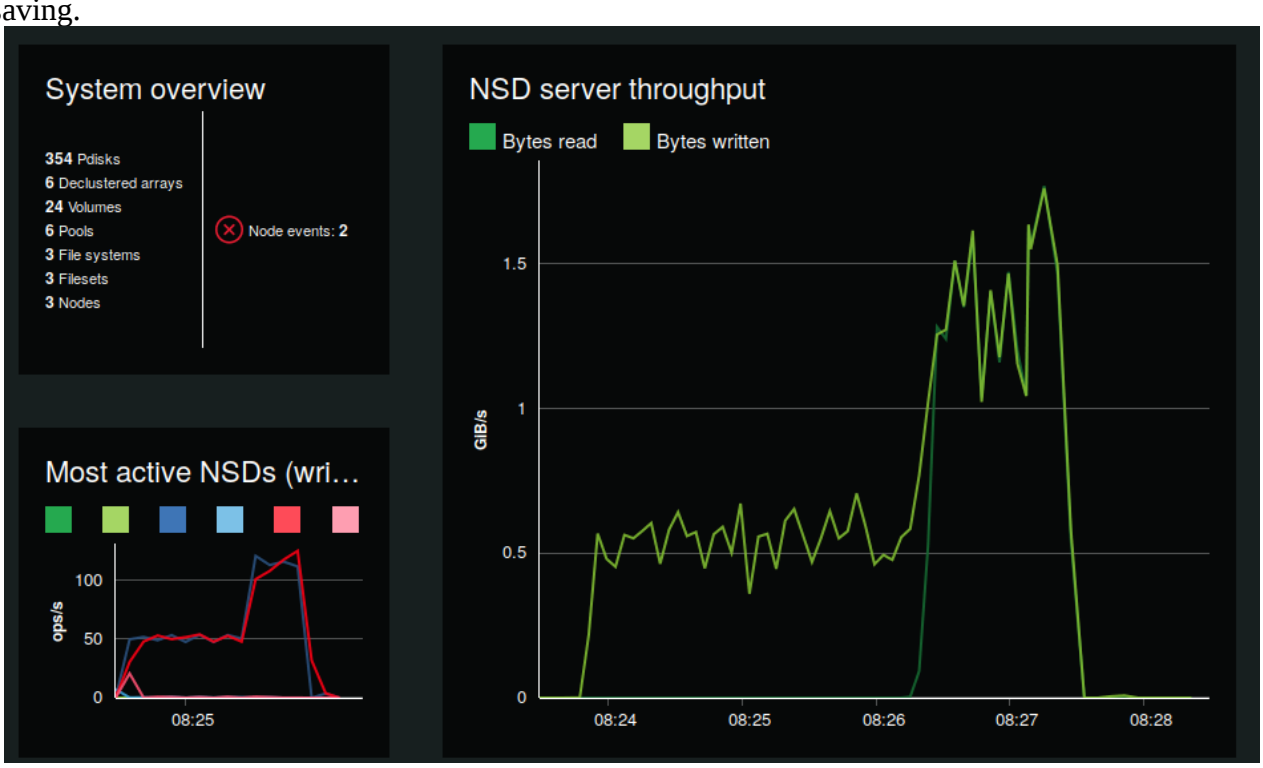

Figure 2: Record from IBM monitor of a pwrit3dc single run process. Two stages (h5dcreate and h5dwrite) are visible. As huge MPI oversubscribe was used at the node the first file creation stage is exceptionaly slow (~ 600 MB/s).

Acknowledgements:

pwrite3cd is an addaption of simple hdf5 example “hdf_pwrite” by Timothy Brown from StackOverflow:http://stackoverflow.com/questions/29075591/creating-hdf5-file-and-datasets-with-openmpi. The code was extended to save large series of image like data (3D datasets, from thiis “3d” in some continuous “c” sense splitting them in multiple datasets only when needed from memmory limitations.

Similar more standard benchmarks: IOR, https://github.com/LLNL/ior

- building (Thanks for hits to to Jan-Frode IBM)

```
module load GCC/5.4.0-2.26 OpenMPI/1.10.3 HDF5/1.10.0-patch1 netCDF/4.4.1
make ./bootstrap
LIBS=-lgpfs ./configure --with-mpiio --with-hdf5
make
```

running:

```
mpirun -np 1 ./ior -a HDF5  -n -b 8M -s 5000 -i 3 -w -o /gpfs/iorTest.h5
mpirun -np 1 ./ior -a MPIIO    -b 8M -s 5000 -i 3 -w -o /gpfs/iorTest.raw
```

results (average and standard deviation):

```
IOR-3.0.1: MPI Coordinated Test of Parallel I/O

ior WARNING: unable to determine HDF5 version for 'no fill' usage.
Began: Fri Mar  3 17:21:04 2017
Command line used: ./../ior -a HDF5 -n -b 8M -s 5000 -i 3 -w -o
/gpfs/gpfs8m/iorTest.h5
Machine: Linux cn0

Test 0 started: Fri Mar  3 17:21:04 2017
Summary:
      api                = HDF5-1.8.17 (Parallel)
      test filename      = /gpfs/gpfs8m/iorTest.h5
      access             = single-shared-file
      ordering in a file = sequential offsets
      ordering inter file= no tasks offsets
      clients            = 1 (1 per node)
      repetitions        = 3
      xfersize           = 262144 bytes
      blocksize          = 8 MiB
      aggregate filesize = 39.06 GiB

access    bw(MiB/s)  block(KiB) xfer(KiB)  open(s)    wr/rd(s)   close(s)
total(s)   iter
------    ---------  ---------- ---------  --------   --------   --------
--------   ----
write     3287.06    8192       256.00     0.001800   11.94      0.225166
12.17      0
remove    -          -          -          -          -          -
0.000263   0
write     3191.24    8192       256.00     0.000721   12.19      0.339779
12.53      1
remove    -          -          -          -          -          -
0.000366   1
write     3269.78    8192       256.00     0.000752   12.03      0.199015
12.23      2
remove    -          -          -          -          -          -
0.000306   2

Max Write: 3287.06 MiB/sec (3446.74 MB/sec)

Summary of all tests:
Operation   Max(MiB)   Min(MiB)  Mean(MiB)     StdDev    Mean(s) Test# #Tasks
write        3287.06    3191.24    3249.36      41.70   12.31216 0 1
. . . HDF5

Finished: Fri Mar  3 17:21:41 2017

------------------------------------------------------------------------------

Operation   Max(MiB)   Min(MiB)  Mean(MiB)     StdDev    Mean(s)
write        4195.58    4152.04    4169.88      18.62    9.59280
. . . MPIIO
```