Parallel file writing in hdf5 format with open_mpi usage

Anna Belova, december 2021

OnDatRa

- We are developing an application that allows you to read data from a very large file, and then parallelize the processing of this data into several worker threads, for parallel processing of data in an online data filter. Our concept assumes the usage of the open_mpi distribution kit, and the final data after processing should be written in the hdf5 format. We previously named the prototype of this multithreaded framework “OnDatRa” (online data filter and reconstruction, in russian it is the muskrat).

Problems

- With parallelization, data processing is quite fast, but if you do not write data to a file, it will take a lot of resources upon completion of the work
- The hdf5 format generally assumes a one-time write to the file
- When creating hdf5 file, a single file space is created in the file and a single data set in it, which in general contradicts multithread writing (if it is assumed that each thread writes its own data set)

Solution in C\C++: hdf5_openmpi

- Preinstalling: sudo apt-get install build-essential gcc libhdf5-openmpi-dev

  Configuring and building:

- sudo cmake .
- mpic++ main.cpp -o <application> -lhdf5_openmpi

  Important c-functions other than the standard mpi and hdf5 libraries:

- plist_id = H5Pcreate(H5P_FILE_ACCESS);
- H5Pset_fapl_mpio(plist_id, comm, info);
- plist_id = H5Pcreate(H5P_DATASET_XFER);
- H5Pset_dxpl_mpio(plist_id, H5FD_MPIO_COLLECTIVE);

Writing dataset sing chunking

- **filespace = H5Dget_space(dset_id);**
- **status = H5Sselect_hyperslab(filespace, H5S_SELECT_SET, offset, stride, count, block);**

  The start array specifies the offset of the starting element of the specified hyperslab.

  The stride array chooses array locations from the dataspace with each value in the stride array determining how many elements to move in each dimension. Setting a value in the stride array to 1 moves to each element in that dimension of the dataspace; setting a value of 2 in allocation in the stride array moves to every other element in that dimension of the dataspace. In other words, the stride determines the number of elements to move from the start location in each dimension. Stride values of 0 are not allowed. If the stride parameter is NULL, a contiguous hyperslab is selected (as if each value in the stride array were set to 1).

  The count array determines how many blocks to select from the dataspace, in each dimension.

  The block array determines the size of the element block selected from the dataspace. If the block parameter is set to NULL, the block size defaults to a single element in each dimension (as if each value in the block array were set to 1).

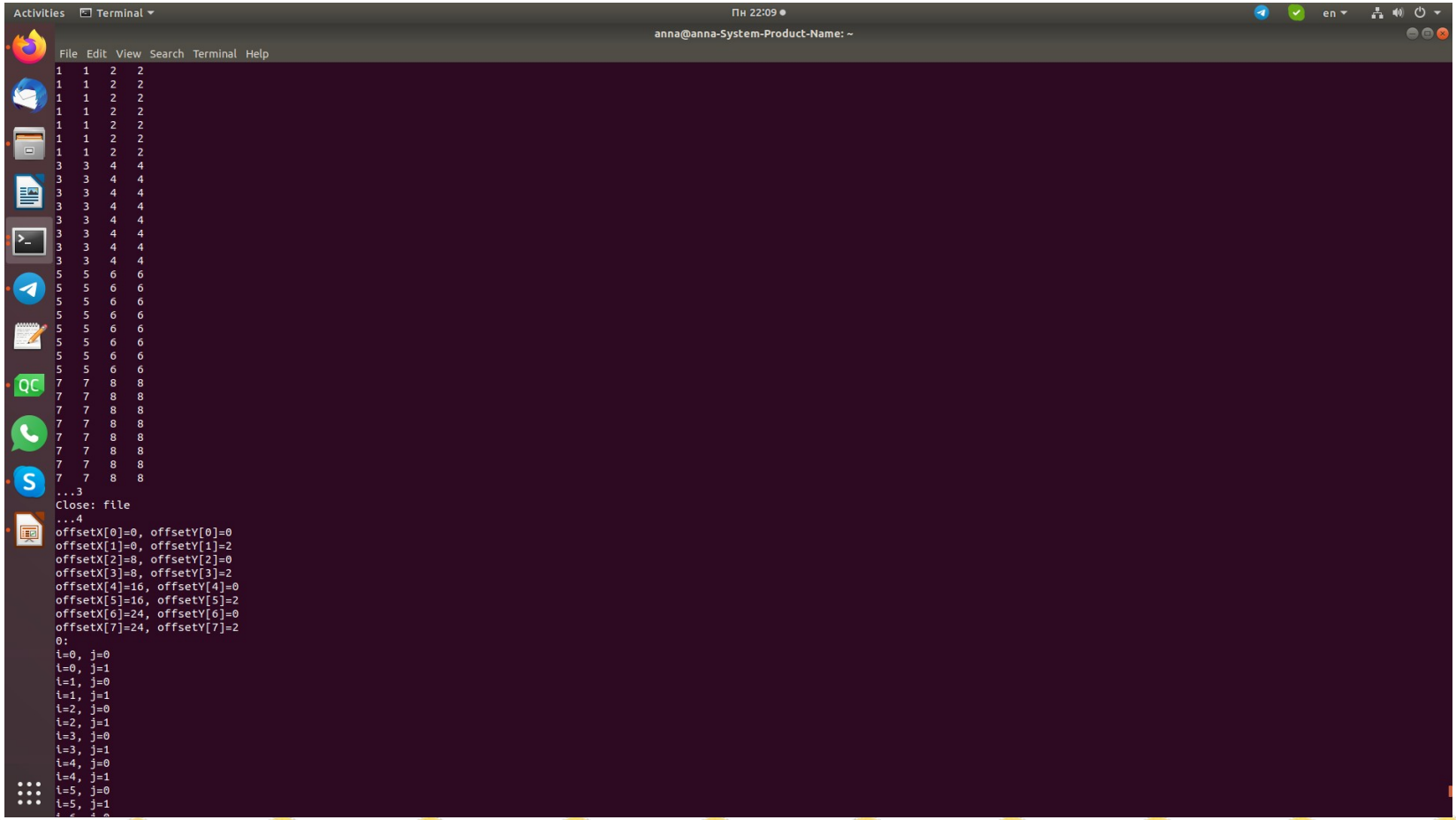
  For example, consider a 2-dimensional dataspace with hyperslab selection settings as follows: the start offset is specified as [1,1], stride is [4,4], count is [3,7], and block is [2,2]. In C, these settings will specify a hyperslab consisting of 21 2x2 blocks of array elements starting with location (1,1) with the selected blocks at locations (1,1), (5,1), (9,1), (1,5), (5,5), etc.; in Fortran, they will specify a hyperslab consisting of 21 2x2 blocks of array elements starting with location (2,2) with the selected blocks at locations (2,2), (6,2), (10,2), (2,6), (6,6), etc.

  Regions selected with this function call default to C order iteration when I/O is performed.

  Parameters:

- hid_t space_id IN: Identifier of dataspace selection to modify
- H5S_seloper_t op IN: Operation to perform on current selection.
- const hsize_t *start IN: Offset of start of hyperslab
- const hsize_t *count IN: Number of blocks included in hyperslab.
- const hsize_t *stride IN: Hyperslab stride.
- const hsize_t *block IN: Size of block in hyperslab.

Example


Activities
Terminal
ПН 22:09
en
anna@anna-System-Product-Name: ~
File Edit View Search Terminal Help
1 1 2 2
1 1 2 2
1 1 2 2
1 1 2 2
1 1 2 2
1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4
3 3 4 4
3 3 4 4
3 3 4 4
3 3 4 4
3 3 4 4
3 3 4 4
5 5 6 6
5 5 6 6
5 5 6 6
5 5 6 6
5 5 6 6
5 5 6 6
5 5 6 6
7 7 8 8
7 7 8 8
7 7 8 8
7 7 8 8
7 7 8 8
7 7 8 8
7 7 8 8
...3
Close: file
...4
offsetX[0]=0, offsetY[0]=0
offsetX[1]=0, offsetY[1]=2
offsetX[2]=8, offsetY[2]=0
offsetX[3]=8, offsetY[3]=2
offsetX[4]=16, offsetY[4]=0
offsetX[5]=16, offsetY[5]=2
offsetX[6]=24, offsetY[6]=0
offsetX[7]=24, offsetY[7]=2
0:
i=0, j=0
i=0, j=1
i=1, j=0
i=1, j=1
i=2, j=0
i=2, j=1
i=3, j=0
i=3, j=1
i=4, j=0
i=4, j=1
i=5, j=0
i=5, j=1

Unresolved problems

- Amount of threads must be equal

  nx * ny/(ch_nx * ch_ny),

  while ch_nx % nx and ch_ny % ny must be zerro for valid work of this application. What if otherwise?
- Work to be tested on big data
- Reading and correct writing of the hdf5 header

Thank you for attention!