COMPARISON OF ARRAY MANAGEMENT LIBRARY PERFORMANCE - A NEUROSCIENCE USE CASE

BERKELEY LAB
Lawrence Berkeley National Laboratory

THE OHIO STATE UNIVERSITY

Donghe Kang, Oliver Rübel, Suren Byna, Spyros Blanas

Storage Software Stack

Application
Array management library
MPI-IO
POSIX-IO
File system

➢ Problem
1. What are the optimal parameters in each layer?

2. Which array management library is most efficient?

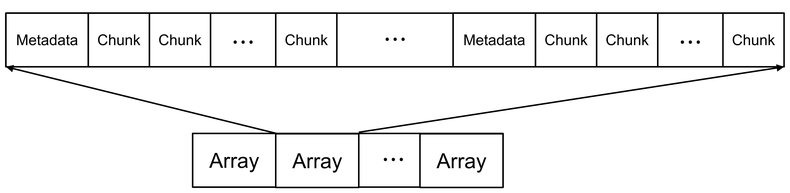
Key contributions

➢ A micro-benchmark summarizing array access patterns in neuroscience applications

➢ Systematic comparison of two array management libraries, HDF5 and Zarr, in the NERSC Cori supercomputer

Array Management Libraries

➢ HDF5 stores arrays in one file
- Partition an array into chunks
- Serialize chunks in row-major order
- A metadata block describes a number of chunks stored sequencially



➢ Zarr stores arrays as folders, and chunks as separate files in the folders

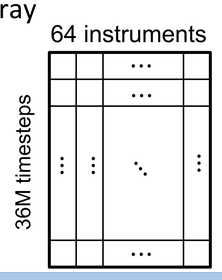
➢ Hypothesis

Accessing metadata blocks should slow HDF5 down, especially when applications read or update the array in column-major

Accessing a large number of small chunks should be the overhead of Zarr

Benchmark

➢ Array



➢ Read/write an array

Multiple processes concurrently read or write separate arrays.

Evaluate sequential I/O performance

Use case: applications flush the observations to a file

➢ Read rows

Read multiple rows from an array in random order.

Evaluate random I/O performance

Use cases: applications get the observations when the brain is stimulated

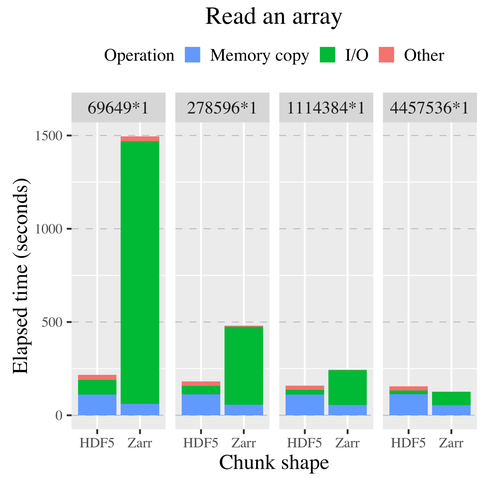
➢ Read a column

Read all cells in one column.

Evaluate fixed-stride I/O performance

Use case: applications gets the observations of a region in the brain

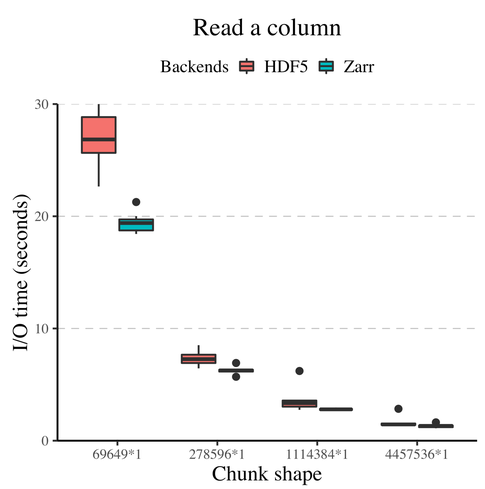
Row-major read



➢ Zarr spends significant cost to open a large amount of files

➢ Memory copy time is constant and higher than I/O time, because the array is column-major serialized in disk, given the chunk shape, but row-major serialized in the output buffer

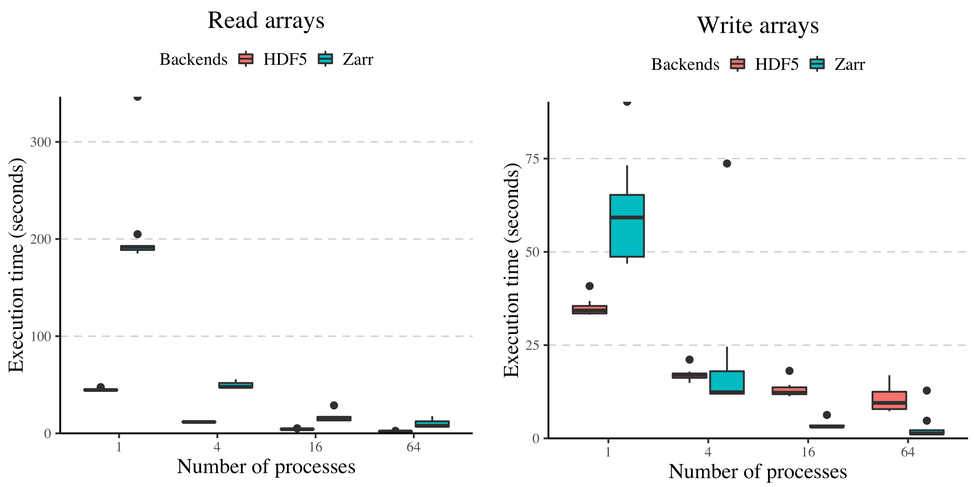
Column-major read



➢ I/O time decreases as we increase the chunk size, due to less I/O operations

➢ HDF5 I/O time is higher than Zarr, because it reads metadata blocks to locate chunks in the file, while Zarr only reads chunk data

Parallel I/O



Conclusion

➢ Reading or writing a file per chunk slows Zarr down. Zarr has competitive performance when the chunk size is large

➢ Accessing metadata blocks is a bottleneck for HDF5

➢ The cost of memory copy is significant when the array is serialized in column-major but accessed in row-major