HDF5 ↔ Zarr

2022 European HDF5 Users Group Meeting

Aleksandar Jelenak

Outline

The HDF Group

- About Zarr
- Zarr API ➔ HDF5 data
- HDF5 API ➔ Zarr data

Zarr Overview

The HDF Group

- Fairly recent *N*-dimensional array storage schema (~2016).
- Supports the *Holy Grail* of the array storage features: hierarchies (groups), chunking, chunk compression.
- Reference implementation as a Python package.
- Multiprocess and multithread read/write operations in Python implementation.
- Backend storage can be any system with a key-value interface: file system, cloud object store, key-value database, in-memory associative array data structure, ZIP archive, etc.
- No single file format yet. (Not an objective.)

Zarr Schema

The HDF Group

- Every Zarr object has a unique ASCII key.
- The value of every Zarr key is a byte sequence.
- Zarr schema metadata are easily understandable JSON objects.
- NumPy *dtype* string format describes Zarr array datatypes.
- Only one chunk compressor allowed, but multiple chunk filters. On write, filters are executed first, in the order of definition, then the compressor. The order is reversed on read.
- Chunk keys contain chunk's *logical array offset*: The "first" chunk element's array indices divided by the chunk shape.
- Zarr attributes are stored as key-value pairs in a separate JSON object.

Interpretation of Zarr Keys

The HDF Group

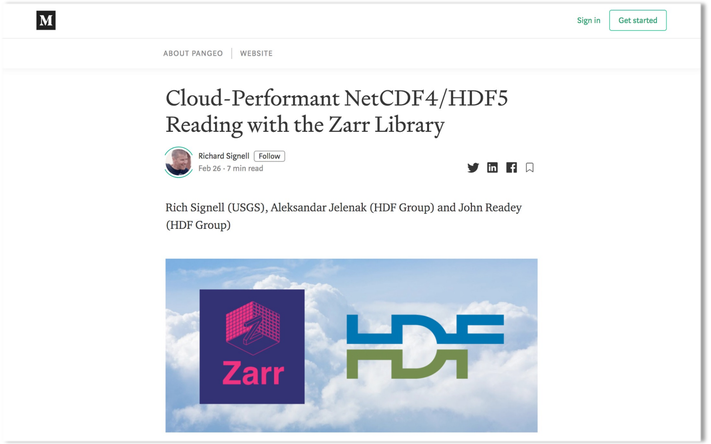
Key	Interpretation
foo/bar/.zgroup	“foo” and “foo/bar” are groups
foo/bar/.zarray	“foo” is a group; “foo/bar” is an array
foo/.zattrs	JSON object with the “foo” Zarr object’s attributes
foo/bar/0.0.0	First “foo/bar” array’s chunk (it’s 3D)
foo/bar/2.1.4	“foo/bar” chunk with first element [20, 20, 120] if chunk shape (10, 20, 30)
foo/barr/2/1/4	Same as above and allowed by the Zarr spec. (Never seen it used.) Allows to establish (sub-)chunk nesting.

Zarr API → HDF5 Data

Proof of Concept

The HDF Group

- In early 2020, the U.S. Geological Survey provided a small grant to explore how HDF5 file data could be read using the zarr Python package.
- The result was a new type of Zarr metadata JSON object: *.zchunkstore.*
- Its content provides the mapping of Zarr chunk keys of HDF5 dataset chunks to their HDF5 file location (offset and size).
- The performance of this approach equaled the native Zarr store for the same data and storage settings.



https://tinyurl.com/bdfd3r8a

kerchunk

The HDF Group

- The *.zchunkstore* approach can be applied to many other data formats.
- Because of its usefulness, it is now a separate Python package called *kerchunk*. Part of the *fsspec* Python project.
  - Docs: https://fsspec.github.io/kerchunk/
  - Repo: https://github.com/fsspec/kerchunk
- *.zchunkstore* concept became *ReferenceFileSystem*.
- Developed and maintained by the Zarr community.
- This package also supports reading from several other data formats (GRIB2, TIFF, CSV, Parquet).

HDF5 API → Zarr Data

Connecting HDF5 API and Zarr Data

The HDF Group

- Current Zarr represents a subset of HDF5 features.
- Zarr storage schema is conceptually equivalent to Highly Scalable Data Service (HSDS) schema.
- HDF5 API access to Zarr data is based on HSDS.
- Only Zarr data in AWS S3.
- Proof of concept.
- Also applied to the netCDF-3 and TIFF file formats.

Implementation

The HDF Group

- Using special HSDS schema chunking layout: `H5D_CHUNKED_REF_INDIRECT`.
- This chunking layout is not supported by the HDF5 library.
- Developed to enable HSDS access to chunks in HDF5 files in object stores.
- Chunk information for one Zarr array is stored as an anonymous HDF5 compound dataset.
- The compound datatype has 3 fields for: byte offset (always 0), chunk object size, and chunk object URI.
- The HDF5 dataset representing the Zarr array has the `H5D_CHUNKED_REF_INDIRECT` layout and its value points to the anonymous HDF5 dataset with chunk location information.

Example Translation

The HDF Group

Zarr array

```
Name               : /zeta
Type               : zarr.core.Array
Data type          : float64
Shape              : (720, 9228245)
Chunk shape        : (10, 141973)
Compressor         : Zlib(level=6)
No. bytes          : 53154691200 (49.5G)
Chunks initialized : 4680/4680
```

HDF5 dataset with chunk info

```
Type               : h5pyd.Dataset
Data type          : compound
Shape              : (72, 65)

Value:
[[(0, 1949049, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.0')
  (0, 2911533, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.1')
  (0, 2506163, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.2') ...
  (0, 4344724, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.62')
  (0, 5696617, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.63')
  (0, 4275725, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.64')]
```

Known Limitations

The HDF Group

Zarr ➔ HDF5:

- Any HDF5 datatype that NumPy does not support.
- HDF5 data of variable-length datatype and compact datasets must be stored directly in the ReferenceFileSystem JSON.
- Advanced HDF5 features: object/region references, virtual dataset; anything that relies on some kind of file system access encoded directly in HDF5 files.

Zarr/HDF5:

- Use of unsupported compressors or filters.
- Processing effort to produce ReferenceFileSystem from HDF5 files or translate that JSON to HSDS JSON.

THANK YOU!

ajelenak@hdfgroup.org
info@hdfgroup.org