Zarr vs. HDF5

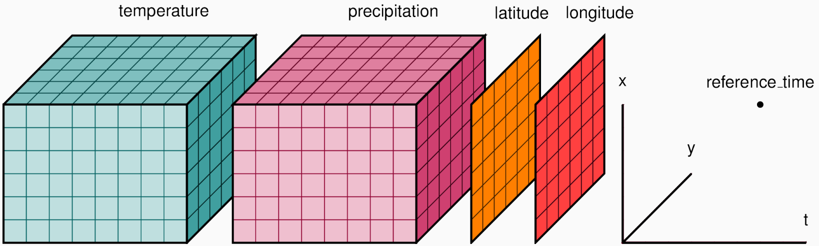
Joe Jevnik

November 4th, 2019

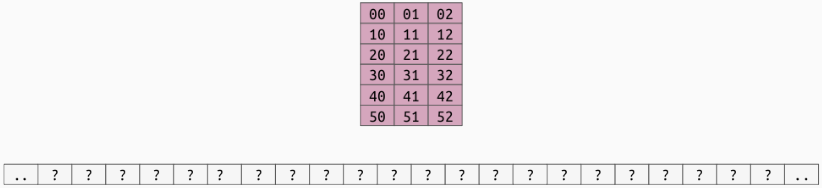
Boston Python

Core Concepts

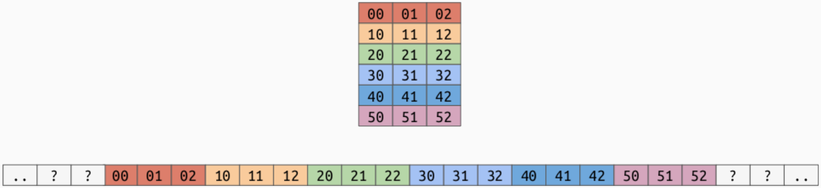
Multidimensional Data



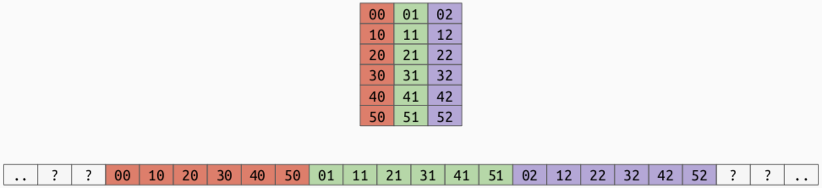
Multidimensional Data



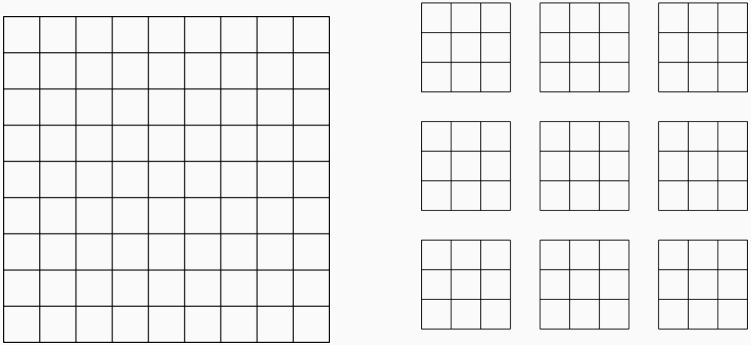
Row Order


00 01 02
10 11 12
20 21 22
30 31 32
40 41 42
50 51 52
.. ? ? 00 01 02 10 11 12 20 21 22 30 31 32 40 41 42 50 51 52 ? ? ..

Column Order



Chunks



Chunks

reduce io

Chunks

reduce io

facilitate caching

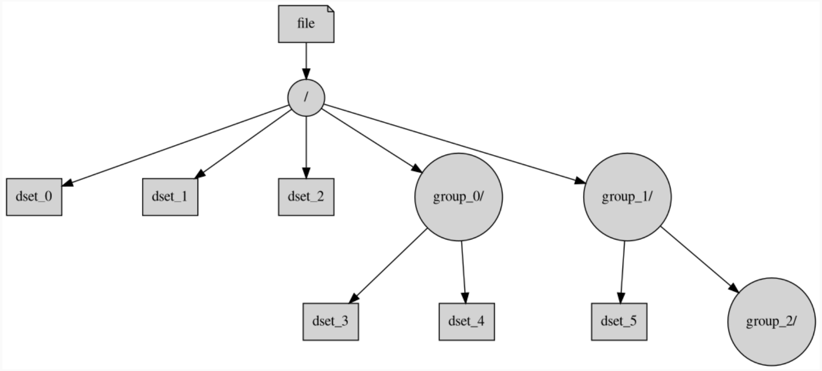
Chunks

reduce io

facilitate caching

allow extending the shape of the dataset

Hierarchy


file
/
dset_0
dset_1
dset_2
group_0/
group_1/
dset_3
dset_4
dset_5
group_2/

Nodes

Definition (Dataset)

a multidimensional array

leaves of a Zarr or HDF5 tree

Nodes

Definition (Dataset)

a multidimensional array

leaves of a Zarr or HDF5 tree

Definition (Group)

collection of **datasets** or **groups**

Nodes

Definition (Dataset)

a multidimensional array

leaves of a Zarr or HDF5 tree

Definition (Group)

collection of **datasets** or **groups**

Definition (Node)

either a **dataset** or **group**

Attributes

Definition (Attributes)

key-value data

Attributes

Definition (Attributes)

key-value data

property of each **node**

Python Interface

General

nested dictionaries

General

nested dictionaries

leaves are *array-like*

General

nested dictionaries

leaves are *array-like*

supports numpy-style indexing

General

nested dictionaries

leaves are *array-like*

supports numpy-style indexing

NOTE: describes `h5py`, not `pytables`

zarr

```
>>> import zarr
>>> f = zarr.open('file.zarr')
>>> f
<zarr.hierarchy.Group '/'>
```

zarr

```
>>> import zarr
>>> f = zarr.open('file.zarr')
>>> f
<zarr.hierarchy.Group '/'>

>>> f['dset'] = np.arange(20 * 5).reshape(20, 5)
>>> f['dset']
>>> <zarr.core.Array '/dset' (20, 5) int64>
```

zarr

```
>>> import zarr
>>> f = zarr.open('file.zarr')
>>> f
<zarr.hierarchy.Group '/'>

>>> f['dset'] = np.arange(20 * 5).reshape(20, 5)
>>> f['dset']
>>> <zarr.core.Array '/dset' (20, 5) int64>

>>> f['dset'][10, 3]
53
>>> f['dset'][:]
array([...]])
```

h5py

```
>>> import h5py
>>> f = h5py.File('file.h5')
>>> f
<HDF5 file "file.h5" (mode r+)>
```

h5py

```
>>> import h5py
>>> f = h5py.File('file.h5')
>>> f
<HDF5 file "file.h5" (mode r+)>

>>> f['dset'] = np.arange(20 * 5).reshape(20, 5)
>>> f['dset']
<HDF5 dataset "dset": shape (20, 5), type "<i8">
```

h5py

```
>>> import h5py
>>> f = h5py.File('file.h5')
>>> f
<HDF5 file "file.h5" (mode r+)>

>>> f['dset'] = np.arange(20 * 5).reshape(20, 5)
>>> f['dset']
<HDF5 dataset "dset": shape (20, 5), type "<i8">

>>> f['dset'][10, 3]
53
>>> f['dset'][:]
array([...]])
```

Extra Functionality

`Node.attrs` to get access to attributes as dict-like object

Extra Functionality

`Node.attrs` to get access to attributes as dict-like object

`Group.create_dataset` to set chunk shape and compression

Extra Functionality

`Node.attrs` to get access to attributes as dict-like object

`Group.create_dataset` to set chunk shape and compression

`Group.create_group` to create sub-groups

Extra Functionality

`Node.attrs` to get access to attributes as dict-like object

`Group.create_dataset` to set chunk shape and compression

`Group.create_group` to create sub-groups

`Dataset.read_direct` to read into existing buffers

Making a Decision

HDF5

- over 20 years old

HDF5

- over 20 years old
- excellent cross language support

HDF5

- over 20 years old
- excellent cross language support
- lots of existing software

HDF5

- over 20 years old
- excellent cross language support
- lots of existing software
- written in (very clean) C

HDF5

- over 20 years old
- excellent cross language support
- lots of existing software
- written in (very clean) C
- can be made thread safe, not thread optimal

HDF5

- over 20 years old
- excellent cross language support
- lots of existing software
- written in (very clean) C
- can be made thread safe, not thread optimal
- extensible in C

Zarr

- first release in 2015, 1.0 on May 17 2016

Zarr

- first release in 2015, 1.0 on May 17 2016
- written in Python, Python oriented

Zarr

- first release in 2015, 1.0 on May 17 2016
- written in Python, Python oriented
- has specification which could be reimplemented

Zarr

- first release in 2015, 1.0 on May 17 2016
- written in Python, Python oriented
- has specification which could be reimplemented
- multithreading support

Zarr

- first release in 2015, 1.0 on May 17 2016
- written in Python, Python oriented
- has specification which could be reimplemented
- multithreading support
- extensible in Python

Extensions

filters and compressors

Extensions

filters and compressors

storage backends

Extensions

filters and compressors

storage backends

which extensions come as part of the library itself?

Extensions

filters and compressors

storage backends

which extensions come as part of the library itself?

how to extend the libraries for non-standard use cases?

Extensions

filters and compressors

storage backends

which extensions come as part of the library itself?

how to extend the libraries for non-standard use cases?

ease of distributing extensions

Filters

Filters

Definition (Filter)

a function that sits between the raw data and the storage

Filters

Definition (Filter)

a function that sits between the raw data and the storage

converts data on read and write

Filters

Definition (Filter)

a function that sits between the raw data and the storage

converts data on read and write

act on one chunk at a time

Filters

Definition (Filter)

a function that sits between the raw data and the storage

converts data on read and write

act on one chunk at a time

composable

Filters

Definition (Filter)

a function that sits between the raw data and the storage

converts data on read and write

act on one chunk at a time

composable

compressors

Filters

Definition (Filter)

a function that sits between the raw data and the storage

converts data on read and write

act on one chunk at a time

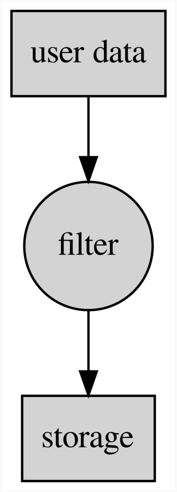
composable

compressors

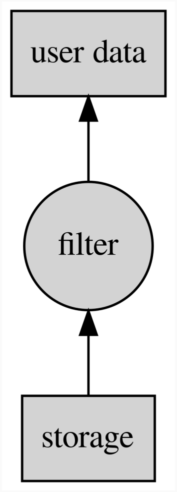
checksumming

Filters

write

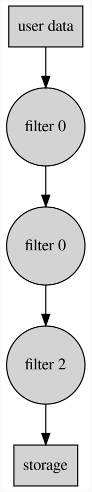


read

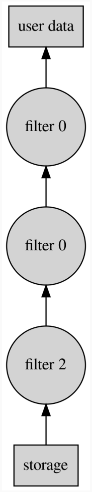


Filter Pipelines

write



read



Writing an HDF5 Filter

```
size_t dod_filter(unsigned int flags,
                  size_t cd_nelmts,
                  const unsigned int cd_values[],
                  size_t nbytes,
                  size_t *buf_size,
                  void **buf);
```