

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



測度 空間 Measure Space  
Apr 28, 2019 · 5 min read

## Use Zarr to access cloud storage just like your local file system

### 1. What is Zarr?

[Zarr](#) official website describes [Zarr](#) as a Python package providing an implementation of **chunked**, **compressed**, **N-dimensional arrays**. **Chunked** indicates that Zarr can handle very large datasets and fast data access. **Compressed** means that Zarr can save files using reasonable storage size which also means with less cost. **N-dimensional arrays** reveals that Zarr can handle multi-dimension datasets just like Netcdf (e.g. geoscience datasets with time, x, y, and z four-dimensional datasets).

Some highlights as follow:

- Create N-dimensional arrays with any NumPy dtype.
- Chunk arrays along any dimension.
- Compress and/or filter chunks using any NumCodecs codec.
- Store arrays in memory, on disk, inside a Zip file, on S3, ...
- Read an array concurrently from multiple threads or processes.
- Write to an array concurrently from multiple threads or processes.
- Organize arrays into hierarchies via groups.

The most critical component of Zarr is that it can let you read and write files to cloud storage system (e.g. AWS S3) just like your local file system with the same convenience of Netcdf format.

### 2. Read a netcdf file

Here we will use the surface air temperature data from [NCEP Reanalysis Dataset](#) as an example. I first downloaded the 2019 surface air temperature file `air.sig995.2019.nc` to my laptop. Then I use [xarray](#) to read the netcdf file.

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including [cookie policy](#).



```
ds = xr.open_dataset('air.sig995.2019.nc')
ds

<xarray.Dataset>
Dimensions:      (lat: 73, lon: 144, nbnds: 2, time: 116)
Coordinates:
  * lat          (lat) float32 90.0 87.5 85.0 82.5 ... -82.5 -85.0
-87.5 -90.0
  * lon          (lon) float32 0.0 2.5 5.0 7.5 10.0 ... 350.0 352.5
355.0 357.5
  * time         (time) datetime64[ns] 2019-01-01 2019-01-02 ... 2019-
04-26
Dimensions without coordinates: nbnds
Data variables:
  air            (time, lat, lon) float32 ...
  time_bnds      (time, nbnds) float64 ...
Attributes:
  Conventions:   COARDS
  title:         mean daily NMC reanalysis (2014)
  history:       created 2017/12 by Hoop (netCDF2.3)
  description:   Data is from NMC initialized
reanalysis\n(4x/day). These...
  platform:      Model
  References:
http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reana...
  dataset_title: NCEP-NCAR Reanalysis 1

ds.air

<xarray.DataArray 'air' (time: 116, lat: 73, lon: 144)>
[1219392 values with dtype=float32]
Coordinates:
  * lat          (lat) float32 90.0 87.5 85.0 82.5 80.0 ... -82.5 -85.0
-87.5 -90.0
  * lon          (lon) float32 0.0 2.5 5.0 7.5 10.0 ... 350.0 352.5
355.0 357.5
  * time         (time) datetime64[ns] 2019-01-01 2019-01-02 ... 2019-
04-26
Attributes:
  long_name:     mean Daily Air temperature at sigma level 995
  units:         degK
  precision:     2
  GRIB_id:       11
  GRIB_name:     TMP
  var_desc:     Air temperature
  dataset:       NCEP Reanalysis Daily Averages
  level_desc:    Surface
  statistic:     Mean
  parent_stat:   Individual Obs
  valid_range:   [185.16 331.16]
  actual_range:  [198.4 314. ]
```

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy. ×

### 3. Save it as Zarr format

Now, we are going to save the data as Zarr format. Because I don't have AWS S3 account, so I'm gonna save it to my laptop. Note that you can save it to AWS S3 directly with the help of [s3fs](#) package. [s3fs](#) is a Pythonic file interface to S3. It builds on top of [boto3](#) which is the Amazon Web Services (AWS) SDK for Python..

```
import zarr
import s3fs

# Compare the data if needed
compressor = zarr.Blosc(cname='zstd', clevel=3)
encoding = {vname: {'compressor': compressor} for vname in
ds.data_vars}
# Save to zarr
ds.to_zarr(store='zarr_example', encoding=encoding,
consolidated=True)

<xarray.backends.zarr.ZarrStore at 0x31a4d1ef0>
```

Now, we have saved the data as local zarr file.

The following code can be used to save data to AWS S3 zarr format.

```
import zarr
import s3fs

# AWS S3 path
s3_path = 's3://your_data_path/zarr_example'
# Initilize the S3 file system
s3 = s3fs.S3FileSystem()
store = s3fs.S3Map(root=s3_path, s3=s3, check=False)
# Compare the data if needed
compressor = zarr.Blosc(cname='zstd', clelve=3)
encoding = {vname: {'compressor': compressor} for vname in
ds.data_vars}
# Save to zarr
ds.to_zarr(store=store, encoding=encoding, consolidated=True)
```

### 4. Access part of the Zarr file

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including [cookie policy](#). ×

can directly access the whole or part of the dataset. Imagine that all the geospatial data (e.g. NCEP Reanalysis) is saved on cloud (it could be thousands of TB or PB), you can use several lines of code to read and download the file directly from AWS S3 without going through the regular painful data downloading process. How cool is that!?

Again, because I don't have AWS S3 access, I'm gonna use local zarr file as an example.

```
# Read Zarr file
zarr_ds = xr.open_zarr(store='zarr_example', consolidated=True)
zarr_ds

<xarray.Dataset>
Dimensions:      (lat: 73, lon: 144, nbnds: 2, time: 116)
Coordinates:
  * lat          (lat) float32 90.0 87.5 85.0 82.5 ... -82.5 -85.0
-87.5 -90.0
  * lon          (lon) float32 0.0 2.5 5.0 7.5 10.0 ... 350.0 352.5
355.0 357.5
  * time         (time) datetime64[ns] 2019-01-01 2019-01-02 ... 2019-
04-26
Dimensions without coordinates: nbnds
Data variables:
  air            (time, lat, lon) float32 dask.array<shape=(116, 73,
144), chunksize=(58, 37, 72)>
  time_bnds      (time, nbnds) float64 dask.array<shape=(116, 2),
chunksize=(116, 2)>
Attributes:
  Conventions:    COARDS
  References:
http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reana...
  dataset_title:  NCEP-NCAR Reanalysis 1
  description:    Data is from NMC initialized
reanalysis\n(4x/day). These...
  history:        created 2017/12 by Hoop (netCDF2.3)
  platform:       Model
  title:          mean daily NMC reanalysis (2014)
```

See, it's easy. Actually, here zarr only read the metadata of the data file rather than loading all the real data. This is very helpful especially when the data size is large. Now, I'd like to read part of the zarr file. What should I do? For example, we can read the air temperature in January 2019.

```
import pandas as pd
```

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



```
# We'd like the read data in Januray 2019
time_period = pd.date_range('2019-01-01', '2019-01-31')
# Select part of the zarr data
zarr_Jan = zarr_ds.sel(time=time_period)
zarr_Jan

<xarray.Dataset>
Dimensions:      (lat: 73, lon: 144, nbnds: 2, time: 31)
Coordinates:
  * lat          (lat) float32 90.0 87.5 85.0 82.5 ... -82.5 -85.0
-87.5 -90.0
  * lon          (lon) float32 0.0 2.5 5.0 7.5 10.0 ... 350.0 352.5
355.0 357.5
  * time         (time) datetime64[ns] 2019-01-01 2019-01-02 ... 2019-
01-31
Dimensions without coordinates: nbnds
Data variables:
  air            (time, lat, lon) float32 dask.array<shape=(31, 73,
144), chunksize=(31, 37, 72)>
  time_bnds      (time, nbnds) float64 dask.array<shape=(31, 2),
chunksize=(31, 2)>
Attributes:
  Conventions:    COARDS
  References:
http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reana...
  dataset_title:  NCEP-NCAR Reanalysis 1
  description:    Data is from NMC initialized
reanalysis\n(4x/day). These...
  history:        created 2017/12 by Hoop (netCDF2.3)
  platform:       Model
  title:          mean daily NMC reanalysis (2014)
```

Now we have the data only in Januray 2019. Cool!

Note that the following code can be used to access AWS S3 zarr file.

```
# AWS S3 path
s3_path = 's3://your_data_path/zarr_example'
# Initilize the S3 file system
s3 = s3fs.S3FileSystem()
sotre = s3fs.S3Map(root=s3_path, s3=s3, check=False)
# Read Zarr file
ds = xr.open_zarr(store=sotre, consolidated=True)
```

## 5. The secret of fast access — `consolidated=True`

Once the data are static and can be regarded as read-only, at least for the metadata/structure of the dataset hierarchy, the many metadata objects can be

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including [cookie policy](#).



## 6. Summary

Using Zarr, we can easily read and write files to cloud storage system (e.g. AWS S3). This is extremely helpful for large geospatial data access using cloud storage system. The **compress** and **consolidated** functionality of Zarr can also help use save storage cost and increase data access speed. N-dimensional data reading and writing is a very hot topic right now. Except Zarr, there are some other packages developing now. So far, Zarr is the leading one.

[AWS](#)[Zarr](#)[Xarray](#)[Python](#)

# Medium

[About](#) [Help](#) [Legal](#)

Get the Medium app

