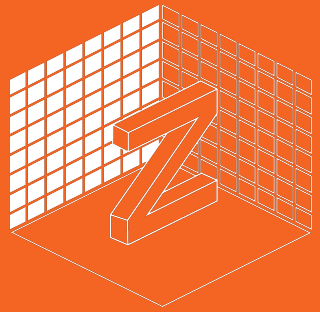


Zarr

Zarr
A Storage Format for Chunked, Compressed ND-Array

@ PyCon DE & PyData Berlin 2023 🇩🇪

~Sanket Verma

About Me

~ Takes care of the community & OSS @ Zarr

~ Chair @ PyData Delhi and Global ‘20 & ‘21

~ Worked with forensics, startups, organisations

~ Sometimes I play the Violin

/msankeys963

Slides and code 👇

What I'm gonna show you?

- ➔ What is Zarr? How Zarr works?
- ➔ Some Code
- ➔ Why and When to use Zarr?
- ➔ How Zarr is different?
- ➔ What is the Zarr specification?
- ➔ Zarr Implementations
- ➔ Zarr Community & Future Goals

Inception of Zarr! 👶

~ Created by Alistair Miles (Oxford Geneticist; @alimanfoo) in 2015

~ Rapidly growing user-base and industry ready

~ Currently has 21 core-devs for Zarr-Python

~ Fiscally sponsored by NumFOCUS

~ Funded by CZI under EOSS

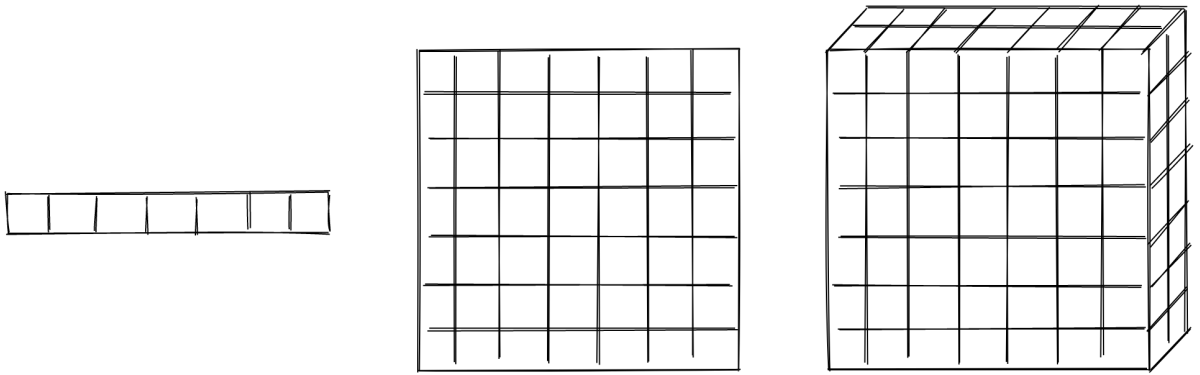
 /zarr_dev  /zarr-developers  https://zarr.dev/

What is an array?

Before Zarr, let's have a look at what array/tensor is?

What is an array/tensor?

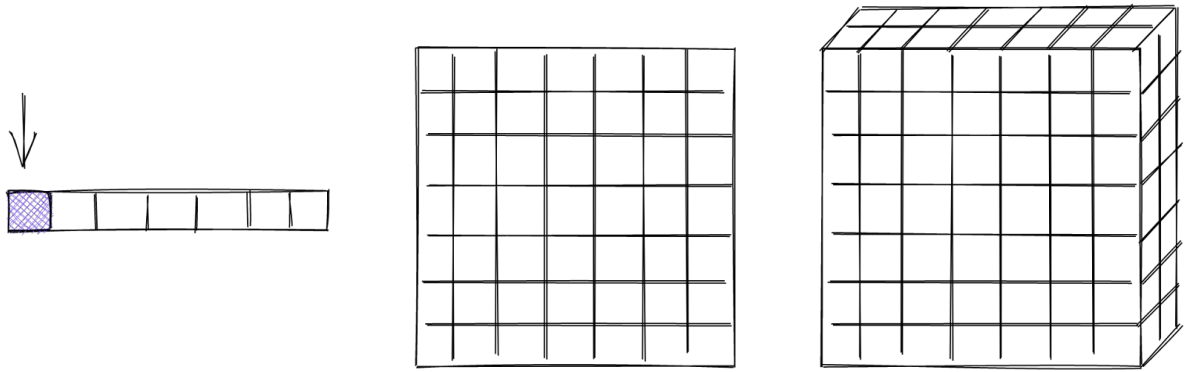
A container of items

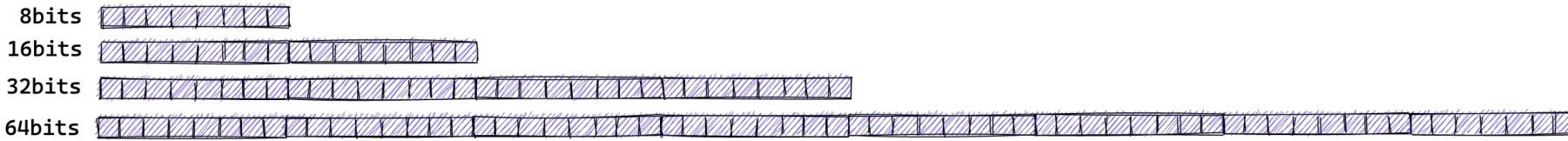


Credits: Trevor Manz (@trevmanz)

What is an array/tensor?

A container of items of the same data-type and size (in bits)

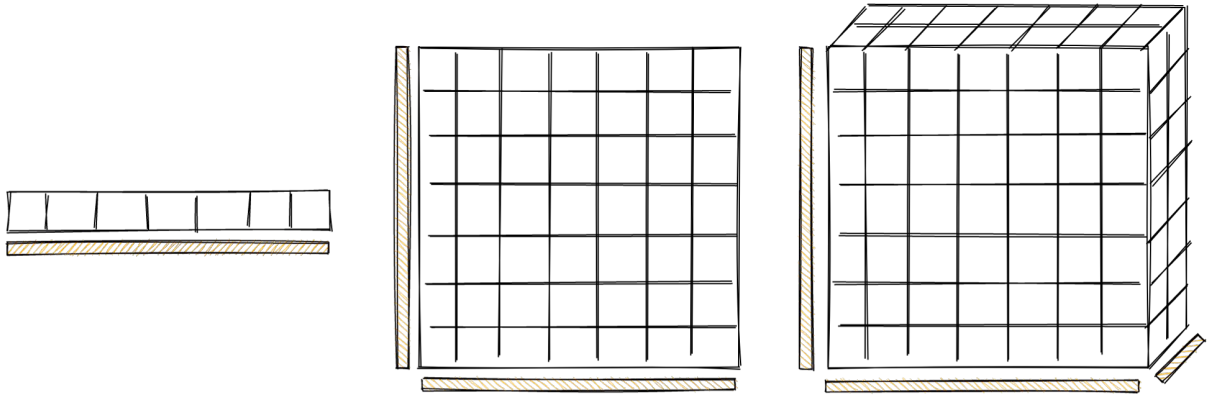




Credits: Trevor Manz (@trevmanz)

What is an array/tensor?

The number of dimensions and items in container are described by the shape



shape	(7,)	(7,7)	(2,7,7)
# dimensions	1D	2D	3D
# items	7	7 * 7	7 * 7 * 2

What is Zarr?

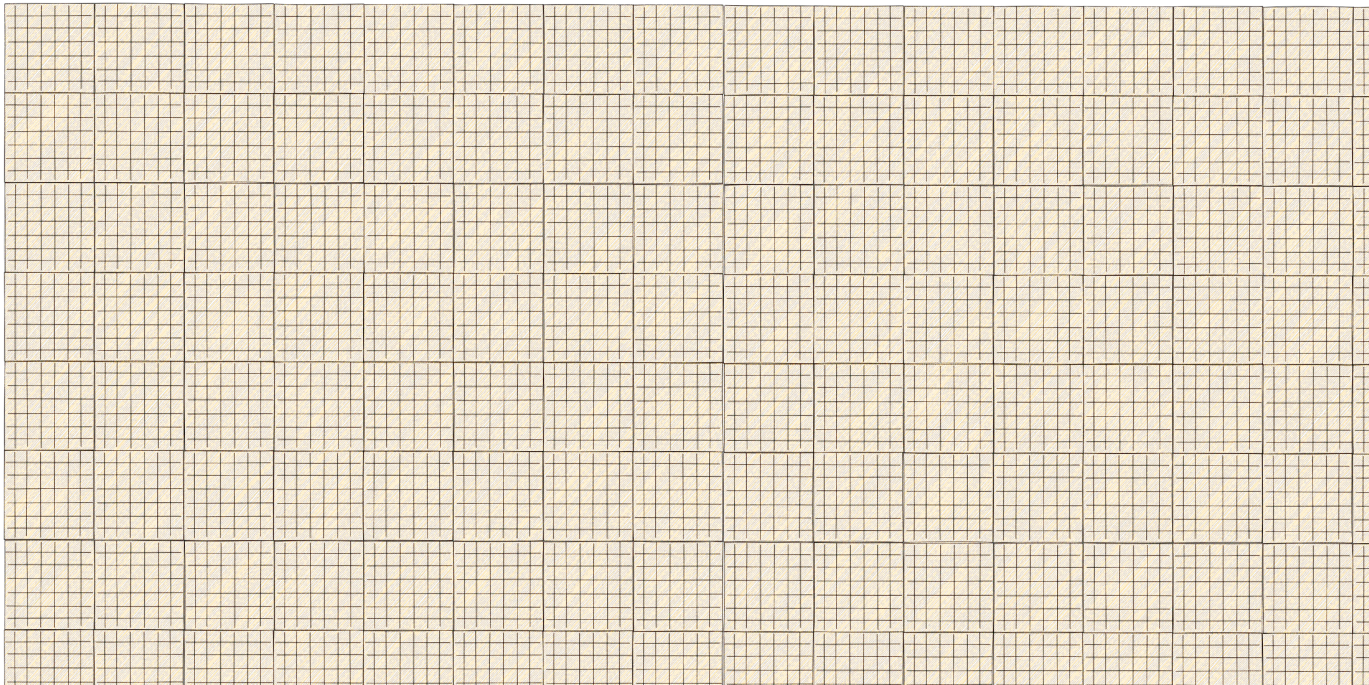
Zarr is open-source specification format for storing chunked, compressed, N-dimensional arrays.

How Zarr works? 🤔

Credits: Trevor Manz (@trevmanz)

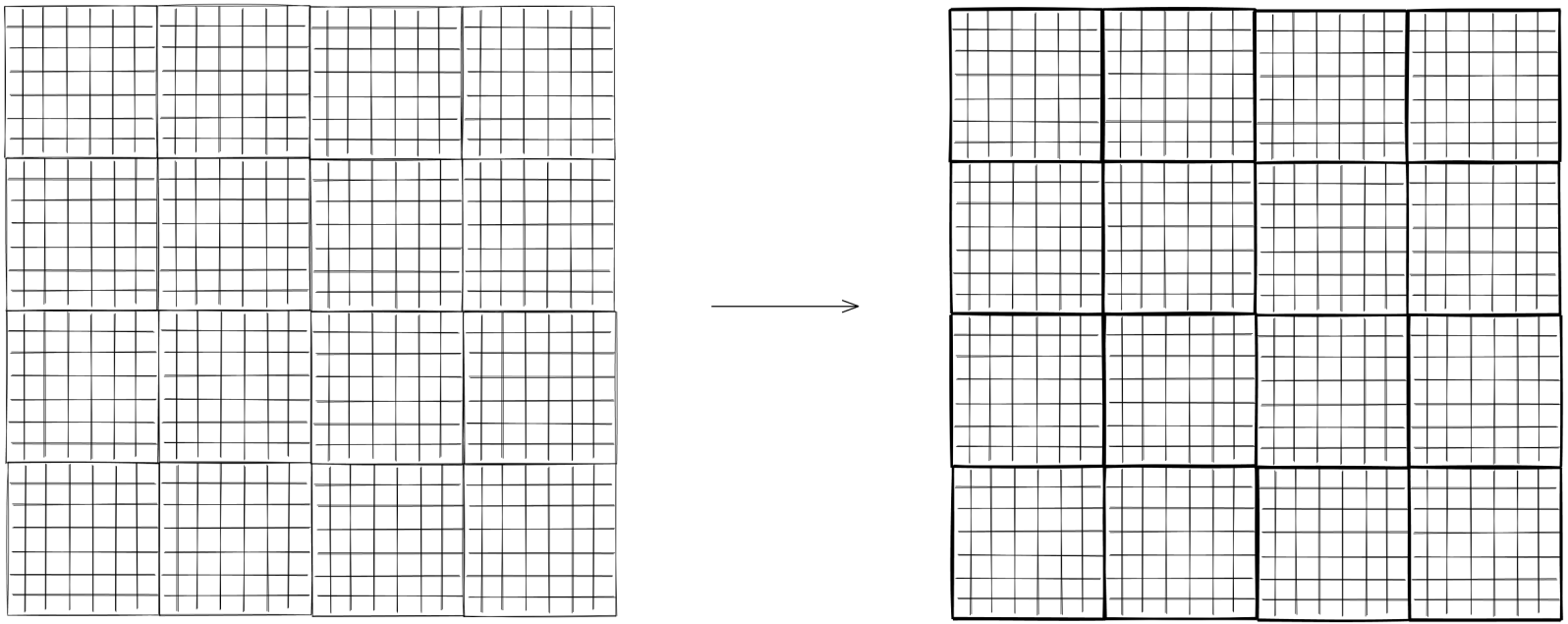
How Zarr works? 🤔

What if the data is too big to fit in memory



How Zarr works? 🤔

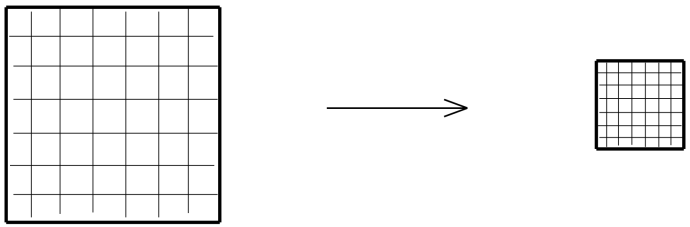
Divide array into chunks



Credits: Trevor Manz (@trevmanz)

How Zarr works? 🤔

Compress each chunk

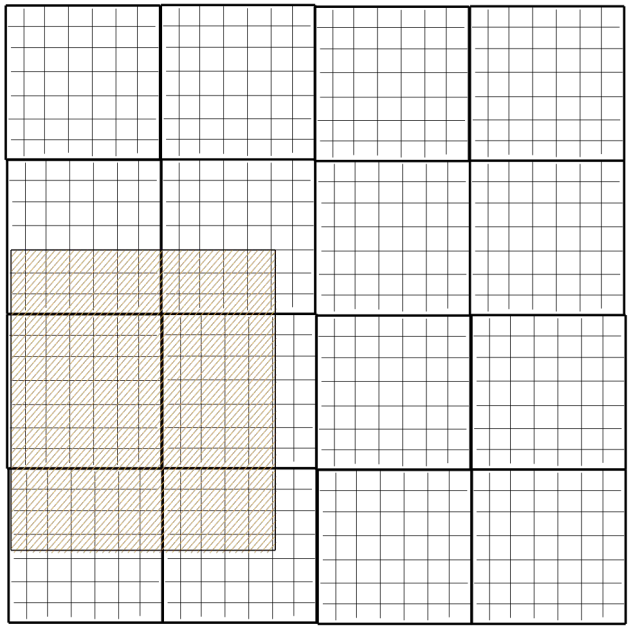


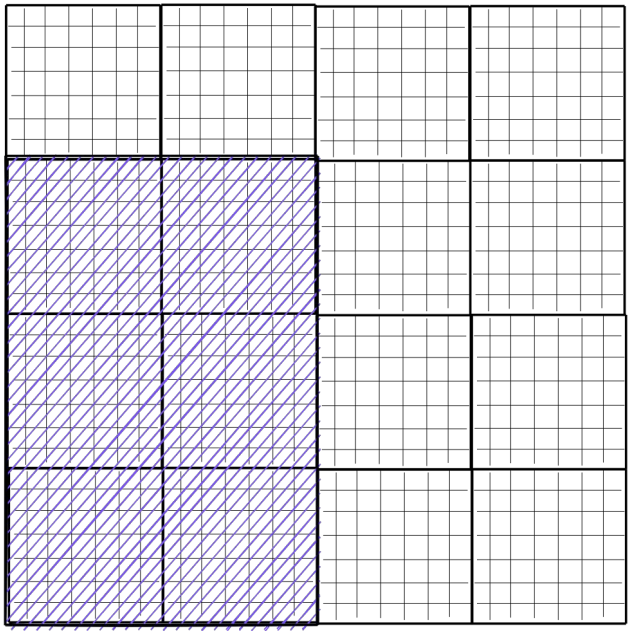
Over 20 supported compressors (BLOSC, Zstd, Zlib, etc)
https://numcodecs.readthedocs.io/en/stable/

Credits: Trevor Manz (@trevmanz)

How Zarr works? 🤔

Retrieve chunks only when needed

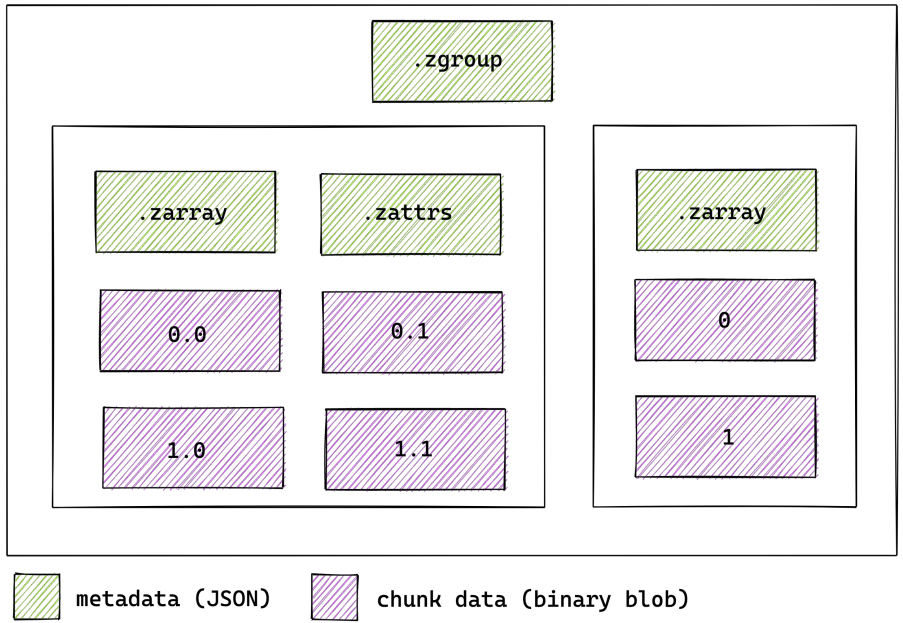




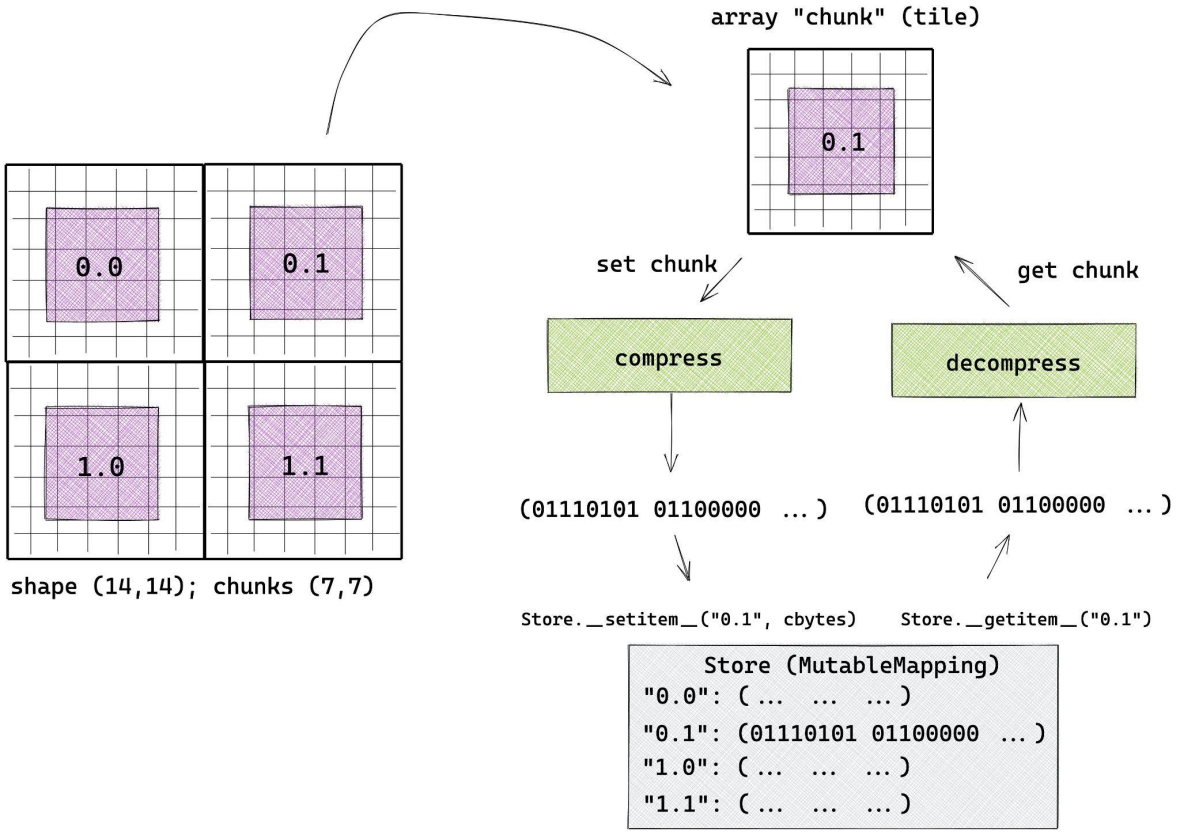
Credits: Trevor Manz (@trevmanz)

How Zarr works? 🤔

Multiple arrays can be organized in hierarchies of groups



How Zarr works? 🤔



Compressors and File Storage in Zarr-Python 

Numcodecs

Numcodecs is a Python package providing buffer compression and transformation codecs for use in data storage and communication applications. These include:

Compression codecs, e.g., Zlib, BZ2, LZMA, ZFPY and Blosc.

Pre-compression filters, e.g., Delta, Quantize, FixedScaleOffset, PackBits, Categorize.

Integrity checks, e.g., CRC32, Adler32.

(https://numcodecs.readthedocs.io/en/stable/)

Compressors and File Storage in Zarr-Python 

~ Over 12+ file storage system supported in Zarr-Python

~ Any object implementing the MutableMapping interface from the collections module in Python can be used as a Zarr array store

~ Can read and write to Amazon S3, GCS, HDFS and Azure Cloud storage

Let's see some code! 

Documentation 📖

Why use Zarr? 🤓

~ Chunk arrays along any dimension

~ Arrays can be read & written concurrently from multiple threads/processes

~ Easy to extend, append, etc.

~ Extensive compression support + easy to plug-in new compressors

~ Support local and cloud storage systems (any key-value store)

~ Simple and based on open specification which allows it to be hackable!

When to use Zarr? 🤓

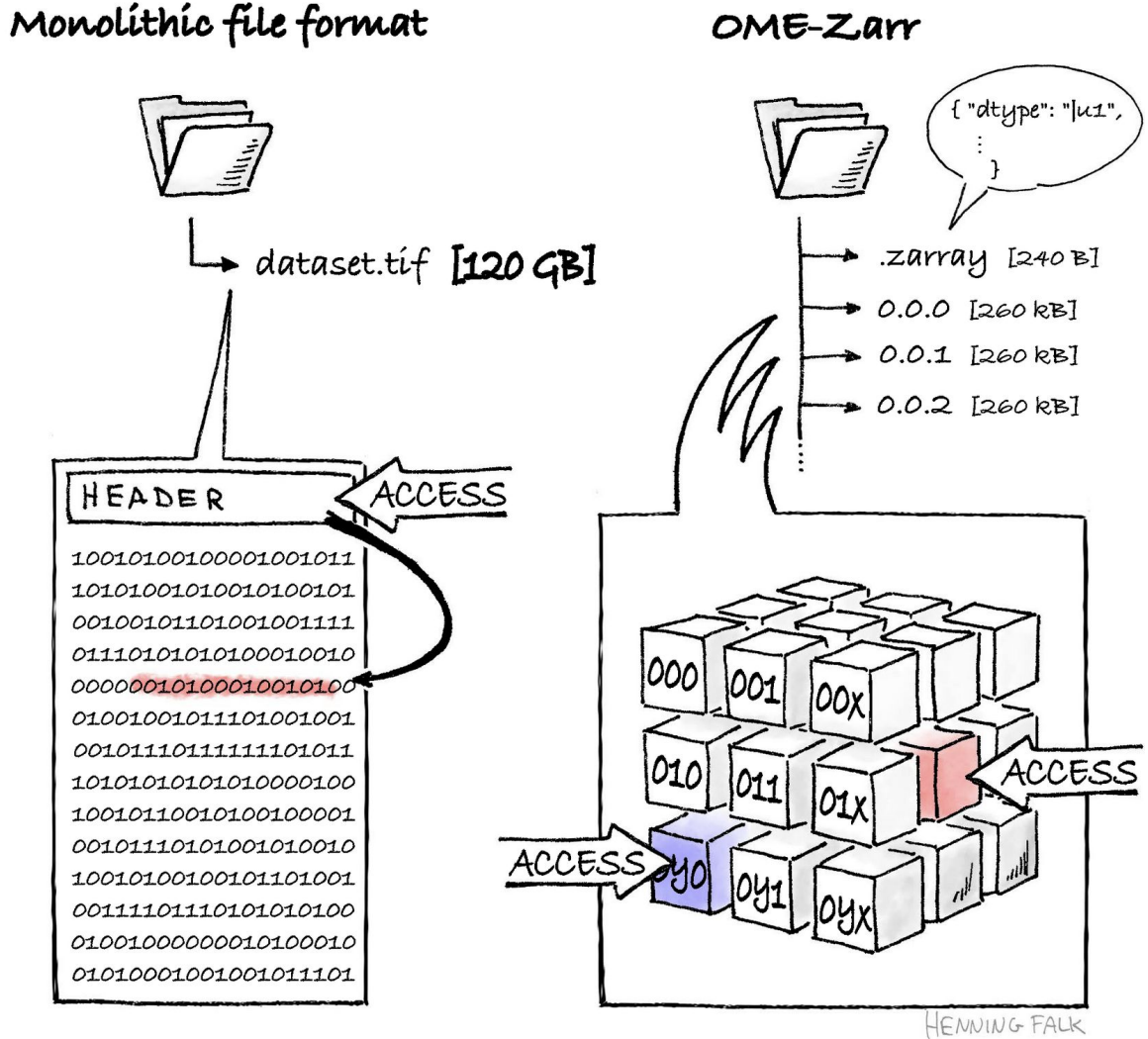
~ When you have absurdly large datasets!

~ When you want to share big datasets across multiple points (having key-valued chunks really helps this)

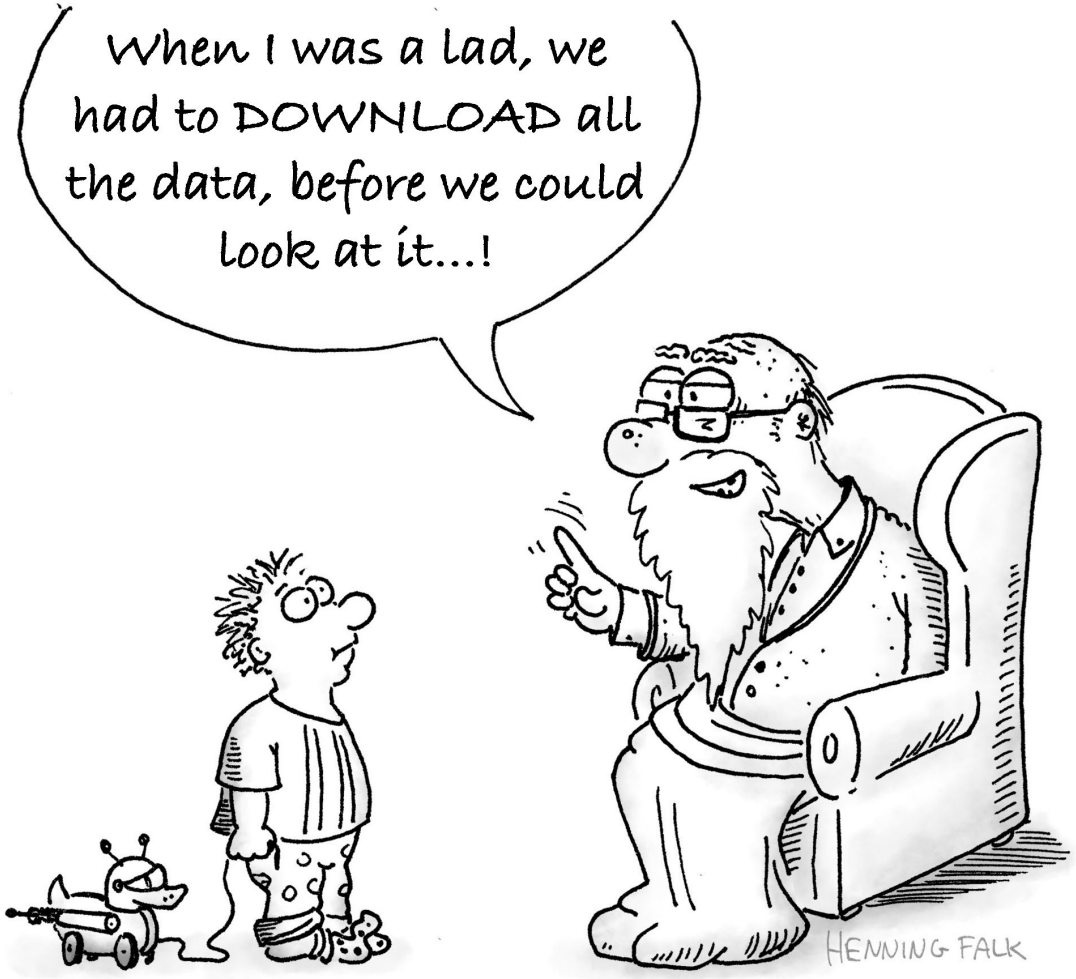
~ If you’re looking for a transparent OS data format with an awesome community!

~ And, to be cool among your friends and colleagues! 😎

How Zarr is different? 🤔


Monolithic file format
OME-Zarr
dataset.tif [120 GB]
HEADER
ACCESS
{ "dtype": "|u1",
}
.zarray [240 B]
0.0.0 [260 kB]
0.0.1 [260 kB]
0.0.2 [260 kB]
000
001
00x
010
011
01x
ACCESS
ACCESS
0y0
0y1
0yx
HENNING FALK


INSTITUT
SSD
10TB
10TB
JUPE
FLASH
4TB
HENNING FALK


When I was a lad, we had to DOWNLOAD all the data, before we could look at it...!
HENNING FALK

More illustrations 👇

The Zarr Specification 🧐

https://zarr-specs.readthedocs.io/

~ Provides a technical specification of the protocol and format for storing Zarr arrays

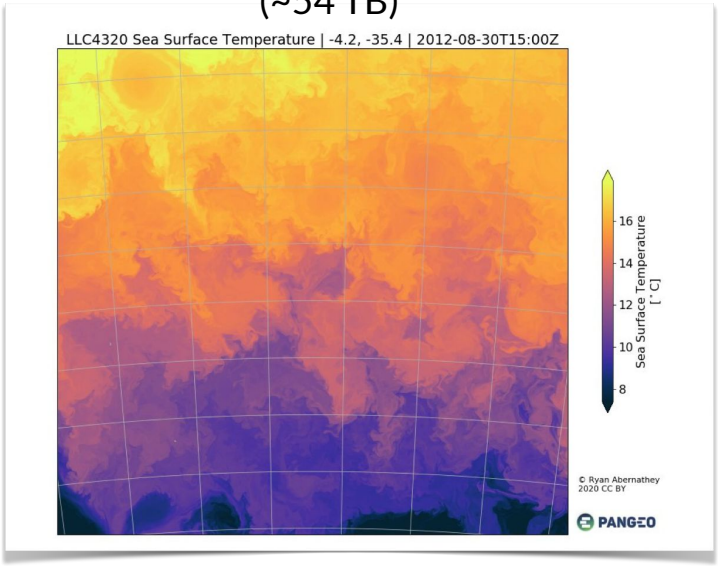
~ Currently V2 is in effect

~ Tells how hierarchies are organised, arrays are chunked and grouped, how metadata should look like etc.

~ V3 is coming out very soon, see ZEP0001

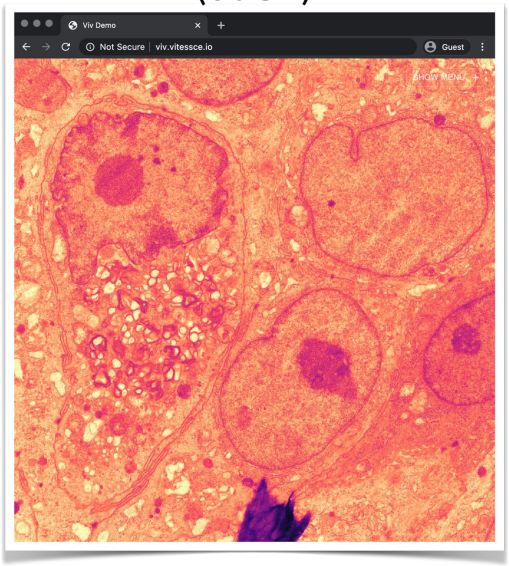
Zarr is a *Generic* Format For Scientific Array Data

Oceanography
(~54 TB)



https://twitter.com/LLC4320Bot/status/1274862822778982402

Microscopy
(30GB)



https://twitter.com/trevmanz/status/1265377097981329423

Interoperability! 🔗

~ Read and write to Zarr Stores using: Dask, Xarray, Anndata etc.

xarray.open_zarr

```
xarray.open_zarr(store, group=None, synchronizer=None, chunks='auto',
decode_cf=True, mask_and_scale=True, decode_times=True,
concat_characters=True, decode_coords=True, drop_variables=None,
consolidated=None, overwrite_encoded_chunks=False, chunk_store=None,
storage_options=None, decode_timedelta=None, use_cftime=None,
**kwargs)                                                    [source]
```

xarray.Dataset.to_zarr

```
Dataset.to_zarr(store=None, chunk_store=None, mode=None,
synchronizer=None, group=None, encoding=None, compute=True,
consolidated=None, append_dim=None, region=None, safe_chunks=True,
storage_options=None)                                        [source]
```

Interoperability! 🔗

~ Dask 👇

dask.array.from_zarr

```
dask.array.from_zarr(url, component=None, storage_options=None, chunks=None, name=None, inline_array=False, **kwargs)
```
[source]

Load array from the zarr storage format

dask.array.to_zarr

```
dask.array.to_zarr(arr, url, component=None, storage_options=None, overwrite=False, region=None, compute=True, return_stored=False, **kwargs)
```
[source]

Save array to the zarr storage format

Zarr Implementations! 😎

~ Python: https://github.com/zarr-developers/zarr-python

~ C++: https://github.com/constantinpape/z5

https://github.com/xtensor-stack/xtensor-zarr

~ NetCDF-Java: https://github.com/Unidata/netcdf-java

~ NetCDF-C: https://github.com/Unidata/netcdf-c

~ Julia: https://github.com/JuliaIO/Zarr.jl

~ Javascript: https://github.com/freeman-lab/zarr-js

~ and several more… *It's relatively easy to implement Zarr from scratch*

Zarr Community & Future! 👨‍👩‍👧‍👧

~ What is the Zarr community and how you can join us?

~ ZEP: A community feedback process to implement changes to the Zarr spec

(https://zarr.dev/zeps/)

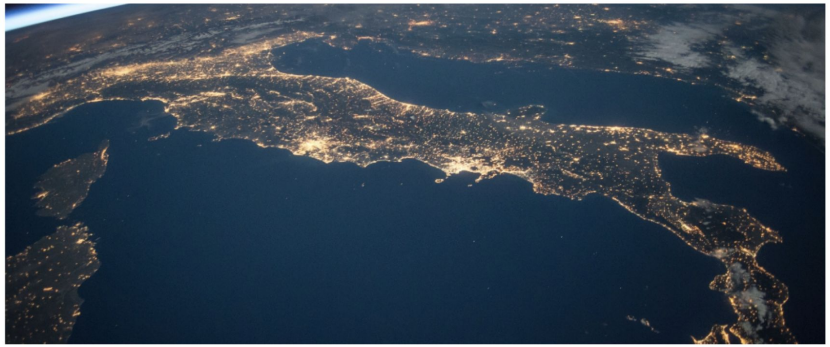
~ What does V3 looks like? Features?

~ Gitter: https://gitter.im/zarr-developers/community/

Example Zarr Cloud Datasets

DATA ANALYTICS

New climate model data now in Google Public Datasets



Shane Glass
Program Manager, Google Cloud Public Dataset Program

December 9, 2019

Exploring public datasets is an important aspect of modern data analytics, and all this gathered data can help us understand our world. At Google Cloud, we maintain a collection of public datasets, and we're pleased to collaborate with the Lamont-Doherty Earth Observatory (LDEO) of Columbia University and the Pangeo Project to host the latest climate simulation data in the cloud.

CMIP6 dataset processed from ESGF NetCDF
>1 PB of data on AWS & GCS
>500K individual Zarr stores

Registry of Open Data on AWS aws

Multi-Scale Ultra High Resolution (MUR) Sea Surface Temperature (SST)

climate | earth observation | environmental | natural resource | oceans | satellite imagery | sustainability | water | weather

Description

A global, gap-free, gridded, daily 1 km Sea Surface Temperature (SST) dataset created by merging multiple Level-2 satellite SST datasets. Those input datasets include the NASA Advanced Microwave Scanning Radiometer-EOS (AMSR-E), the JAXA Advanced Microwave Scanning Radiometer 2 (AMSR-2) on GCOM-W1, the Moderate Resolution Imaging Spectroradiometers (MODIS) on the NASA Aqua and Terra platforms, the US Navy microwave WindSat radiometer, the Advanced Very High Resolution Radiometer (AVHRR) on several NOAA satellites, and in situ SST observations from the NOAA iQuam project. Data are available from 2002 to present in Zarr format. The original source of the MUR data is the NASA JPL Physical Oceanography DAAC.

Update Frequency

The temporal extent of the Zarr store is 2002-06-01 to 2020-01-20.

License

There are no restrictions on the use of these data.

Documentation

https://podaac.jpl.nasa.gov/dataset/MUR-JPL-L4-GLOB-v4.1

Managed By

FARALLON INSTITUTE

See all datasets managed by Farallon Institute.

L4 Sea Surface Temperature
One single big Zarr store
54 TB total

How Zarr is different? 🤔

~ Active community, rapid development

~ Implementations in several programming languages

~ Versatile data format; data conventions like GeoZarr, OME-Zarr, NCZarr

~ Interoperable with other libraries in the PyData Stack

~ Adopted by many groups → https://zarr.dev/adopters/

Thank you! 🙌

You are welcome at our community meetings and office hours!

(https://zarr.dev/community-calls/)
(https://zarr.dev/office-hours/)

 /zarr_dev  /zarr-developers  https://zarr.dev/