CATERVA

A Compressed And Multidimensional Container For ~~Big~~ Medium Size Data

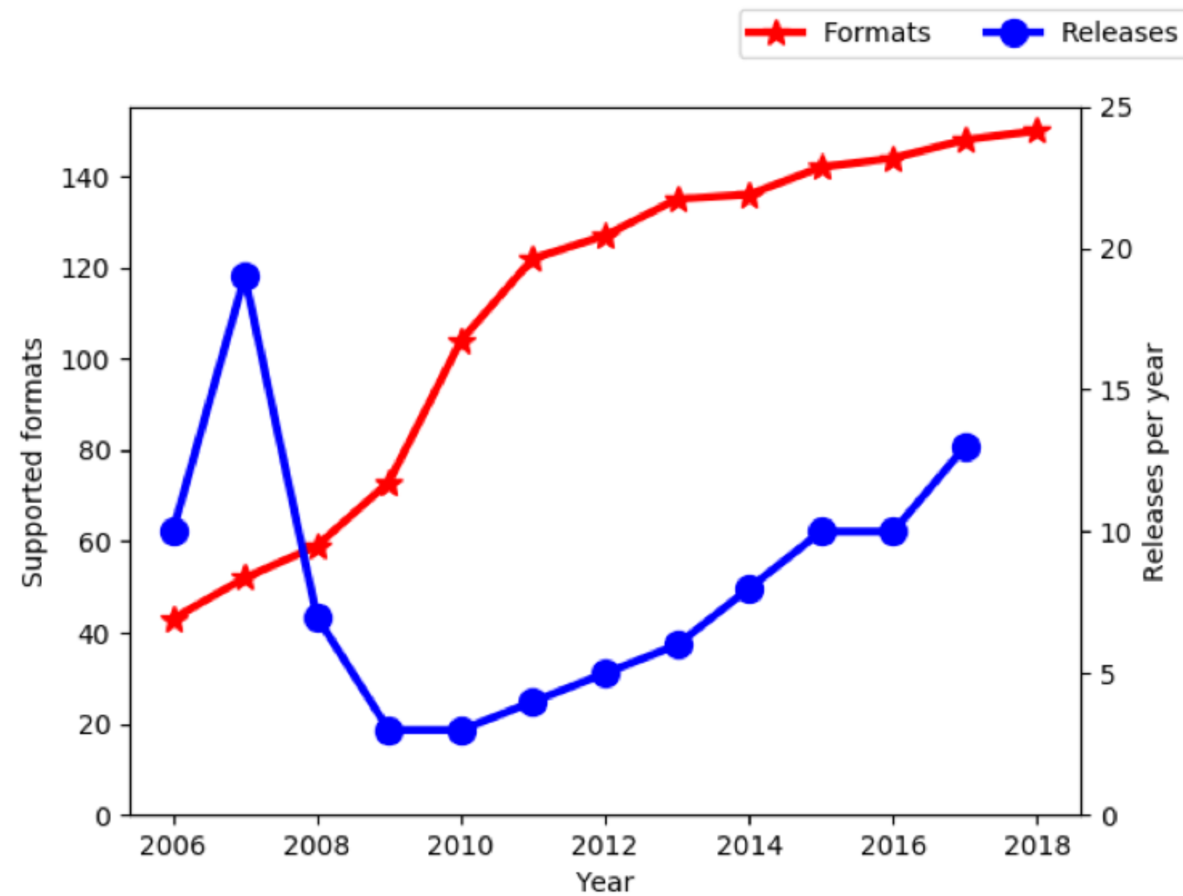
@FrancescAlted
Freelancer

Sponsored by NUMFOCUS OPEN CODE = BETTER SCIENCE

A Member of the BLOSC Family

The Big Debate (I)

Evolution of proprietary
file formats by Bio-Formats
Source: Euro-Bioimaging Industry Board



"Put simply, the format landscape has scaled beyond a manageable level."

–OME's position regarding file formats

https://blog.openmicroscopy.org/community/file-formats/2019/06/25/formats/

The Big Debate (II)

- My position on adding new formats is that we absolutely need them for adapting not only to different use cases but also to the rapidly evolving storage technology.

- In particular, I claim that we need more **in-memory compressed formats** because I see memory more and more as another storage layer, and high-speed compression can help on using memory more efficiently.

- Having said that, we *absolutely need* to adopt **open formats** for reducing the maintenance burden and facilitate its adoption more quickly.

What is Caterva?

- It is an open source C library and a format that allows to store large multidimensional, chunked, compressed datasets.

- Data can be stored either in-memory or on-disk, but the API to handle both versions is the same.

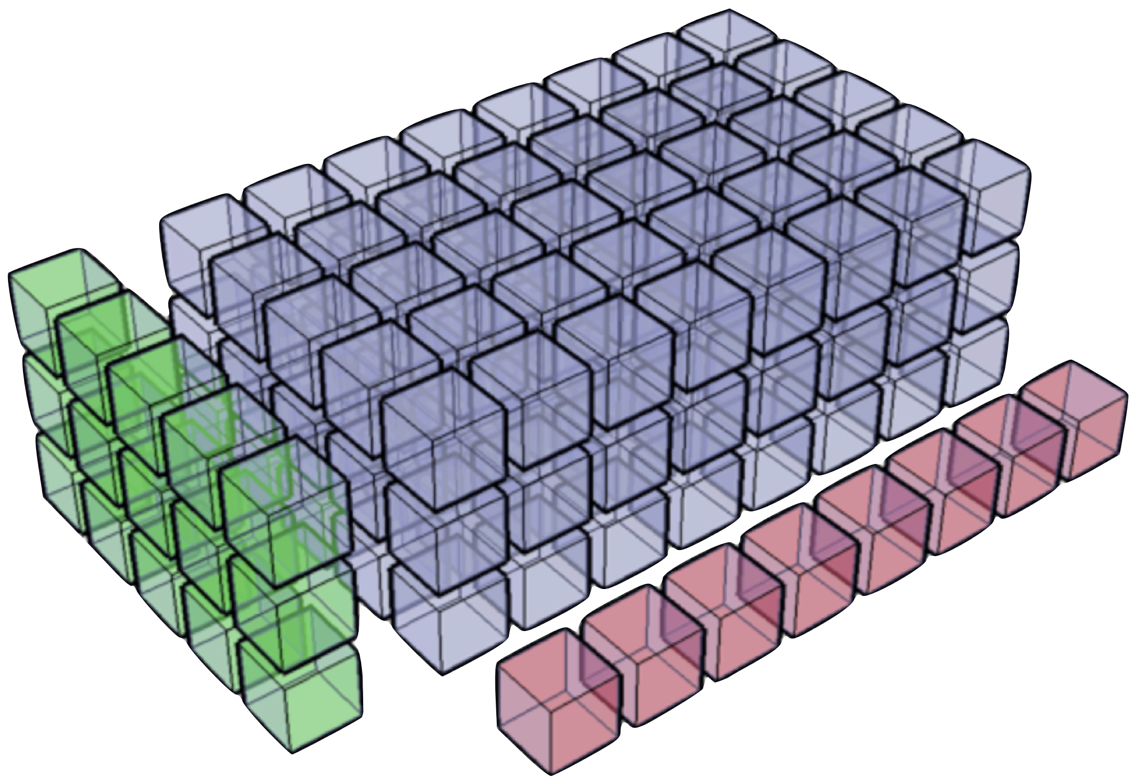
- Compression is handled transparently for the user by adopting the Blosc2 library.

- cat4py is a thin wrapper for Python.

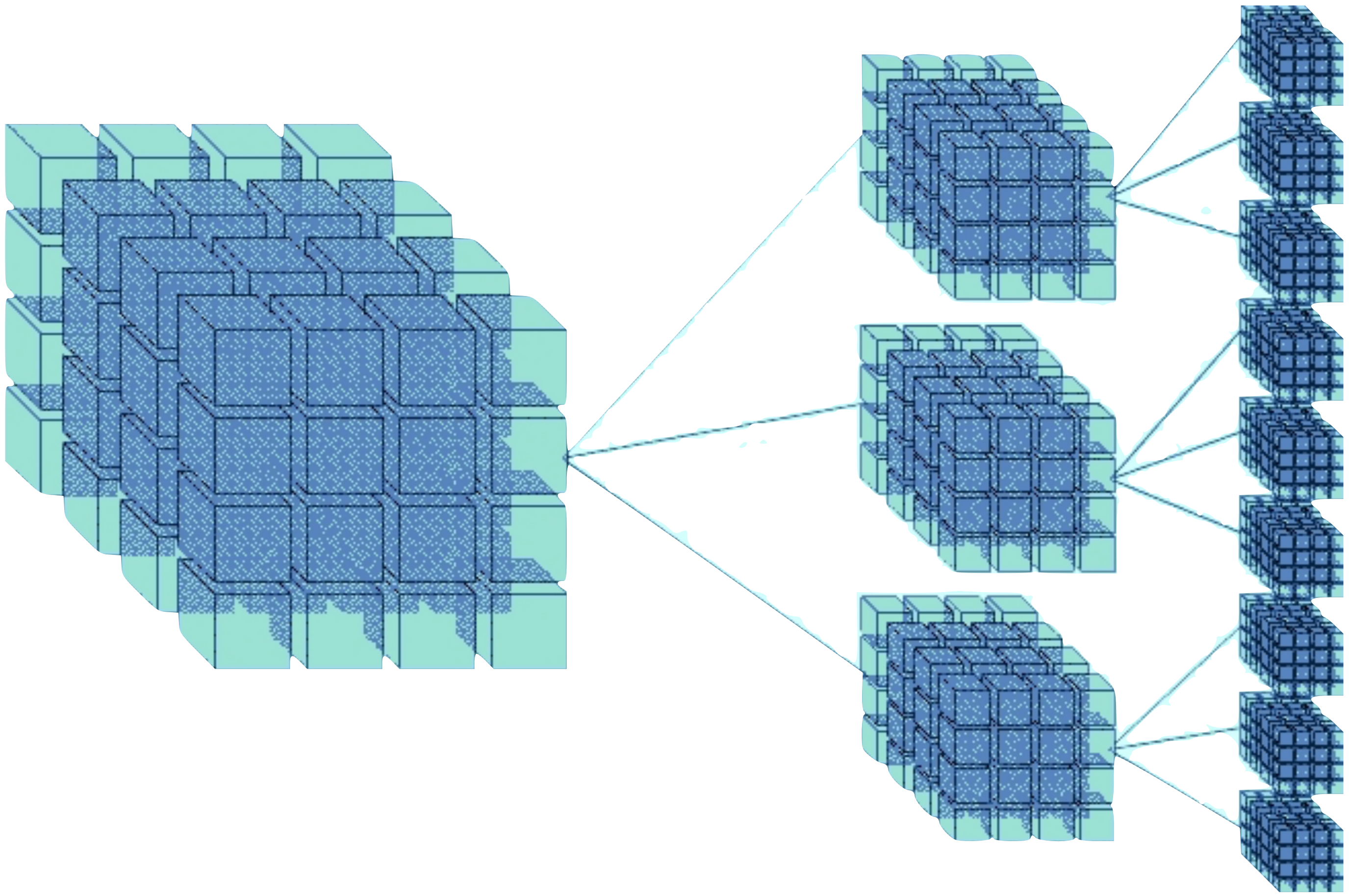
Caterva Brings Powerful Slicing Capabilities

- Caterva's main feature is to be able to extract all kind of slices out of high dimensional datasets, efficiently.

- Resulting slices can be either Caterva containers or regular plain buffers (for better interaction with e.g. NumPy).



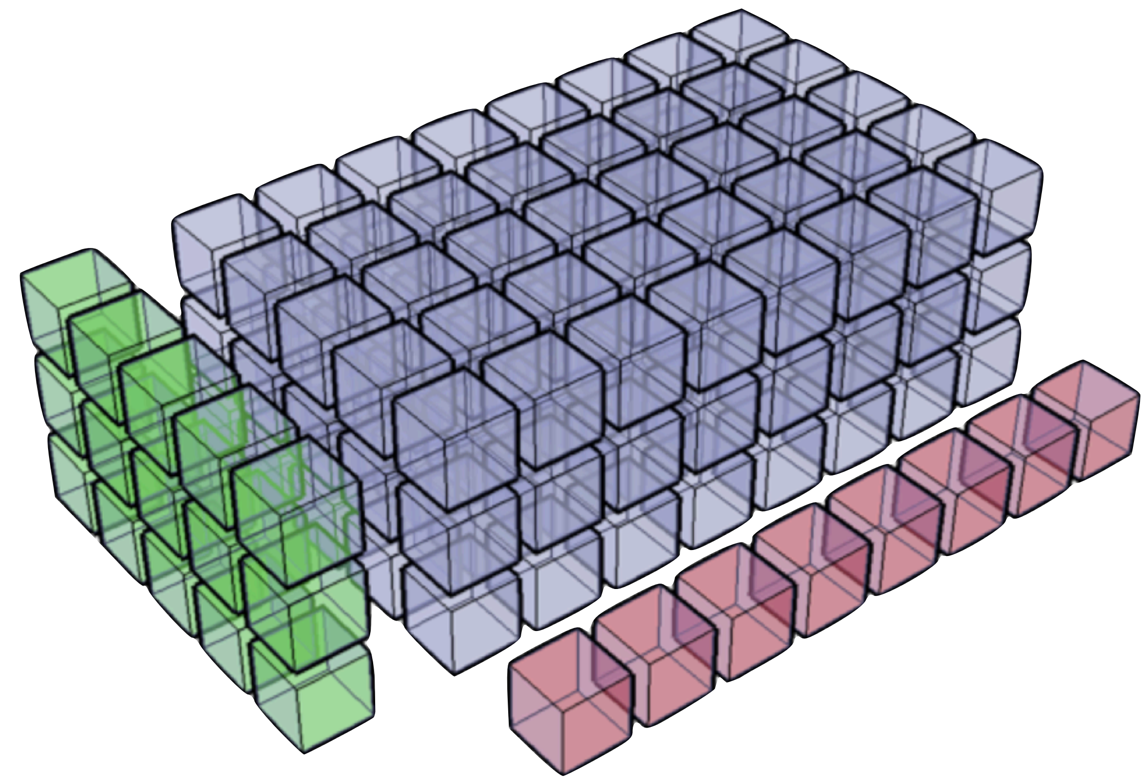
Compression and Partitions



- Caterva normally splits the dataset into so-called **partitions** (aka chunks in Blosc idiom).
- Such partitions can be compressed individually.

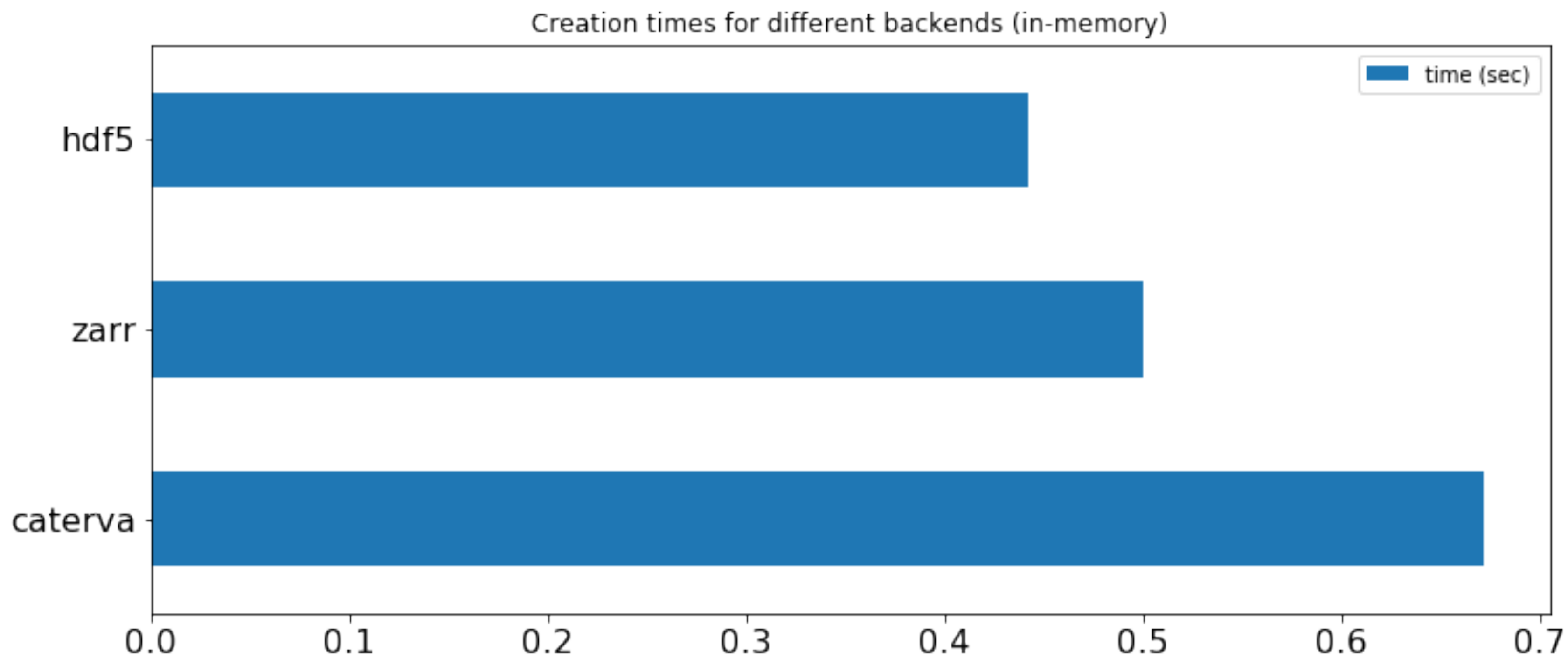
Accessing Chunked Datasets

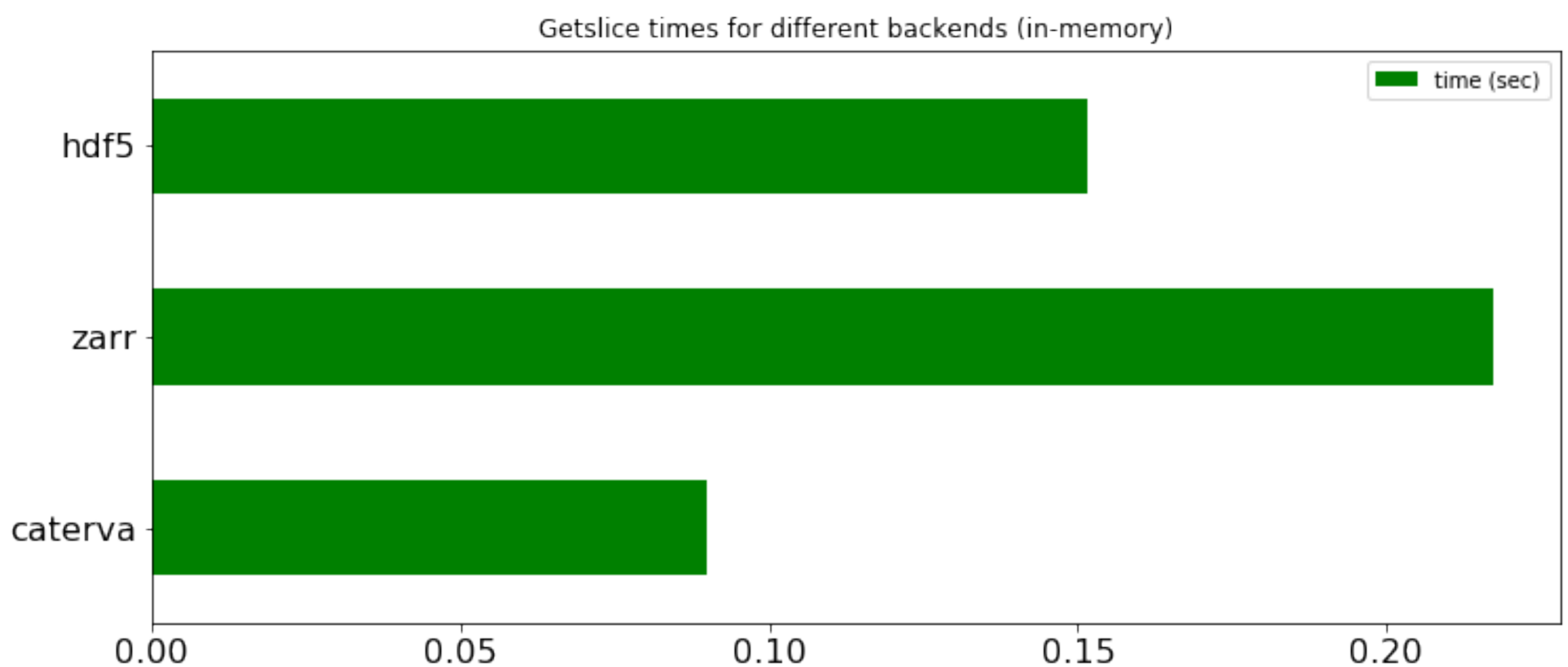
Those used to manipulate chunked multidimensional arrays know how critical choosing the partition size is.



https://github.com/Blosc/cat4py/blob/master/notebooks/compare_getslice.ipynb

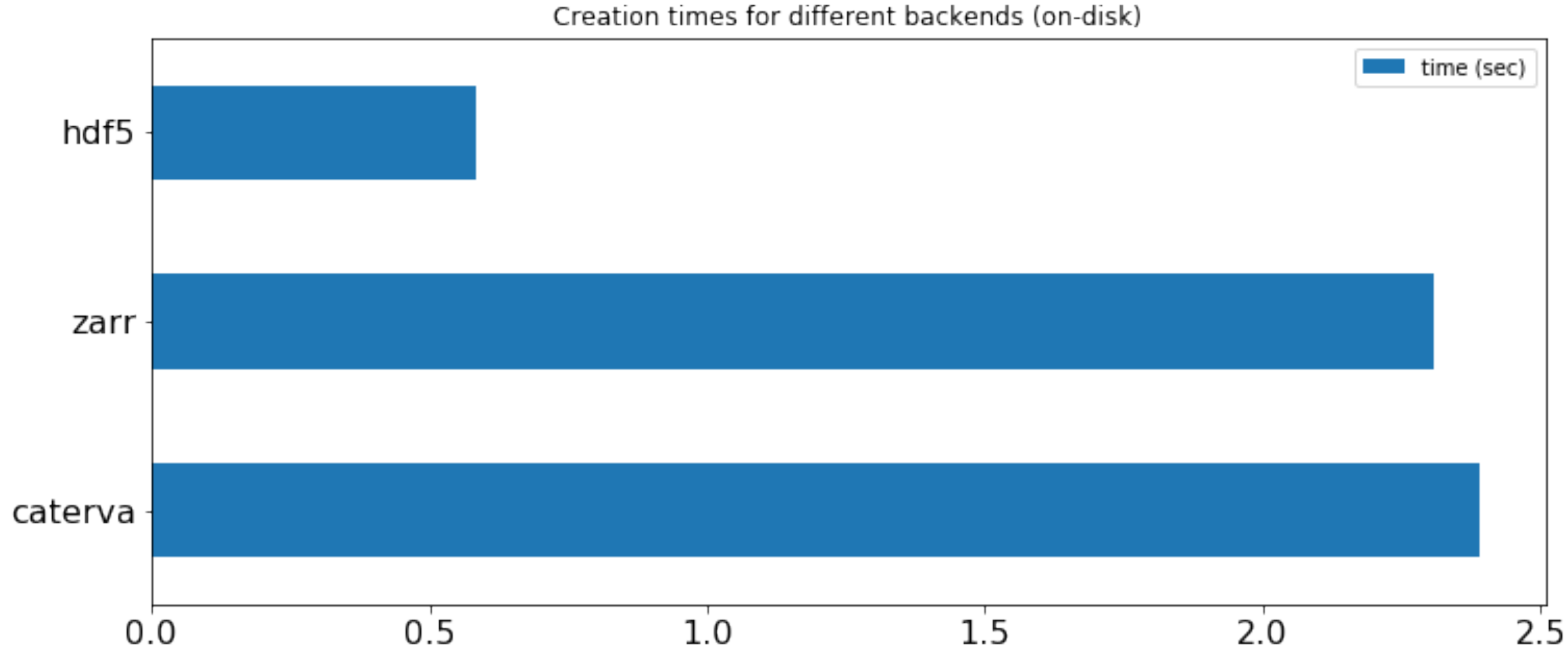
Performance In-Memory

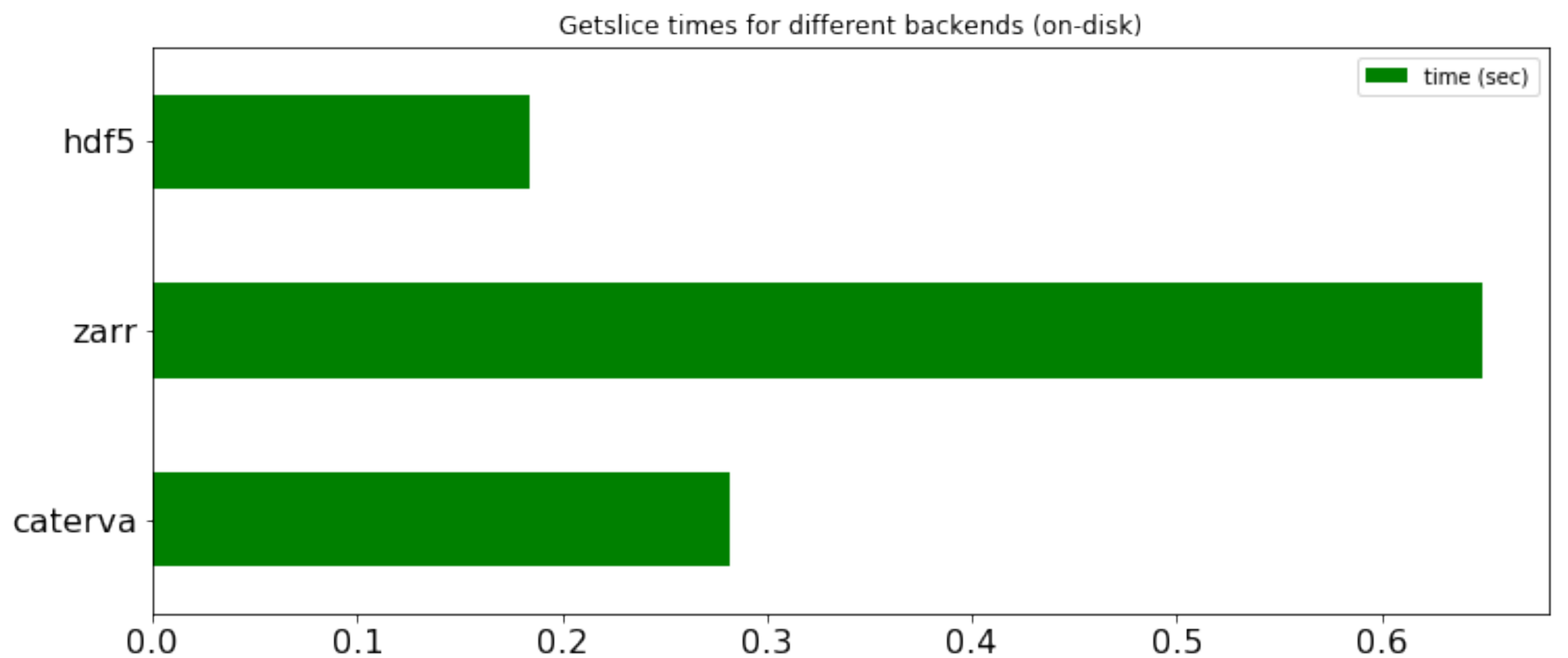




Caterva is meant to read data very fast!

Performance On-Disk





Caterva still has room for optimization when reading from disk!

Brief Comparison Against Well Known Chunked Containers

	HDF5	Zarr	Caterva
Solid	Yes	No (1 file per chunk)	Yes
Mature	Yes	Yes	In process
In-memory version?	Yes (sequential?)	Yes (sparse)	Yes (sequential / sparse)
Hierarchical	Yes	Yes	No (use the filesystem)

Blosc2

- Blosc2 is the next generation of the well-known Blosc (aka Blosc1).
- New features:
  - Enlargeable 64-bit containers: in-memory or on-disk
  - New compression codecs
  - New filters
  - Metalayers
  - User metadata

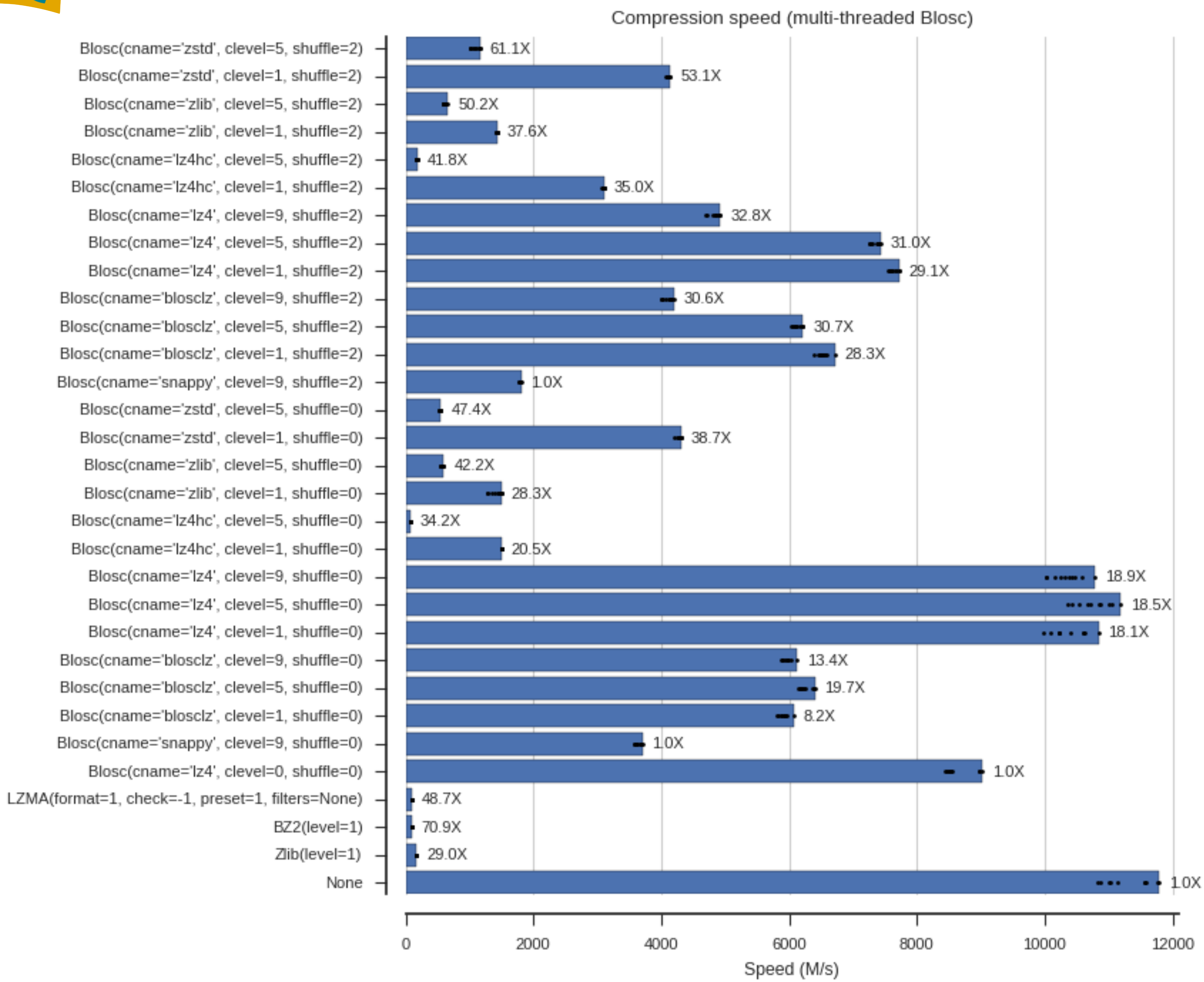
A Few Words About Blosc1

- Blosc1 (or just Blosc) was the original meta-codec that was meant to be used to accelerate operation in PyTables (HDF5), but it became stand-alone pretty soon.

- Blosc1 is distributed with different codecs and filters so that it is very easy to be adopted by third-party libraries.

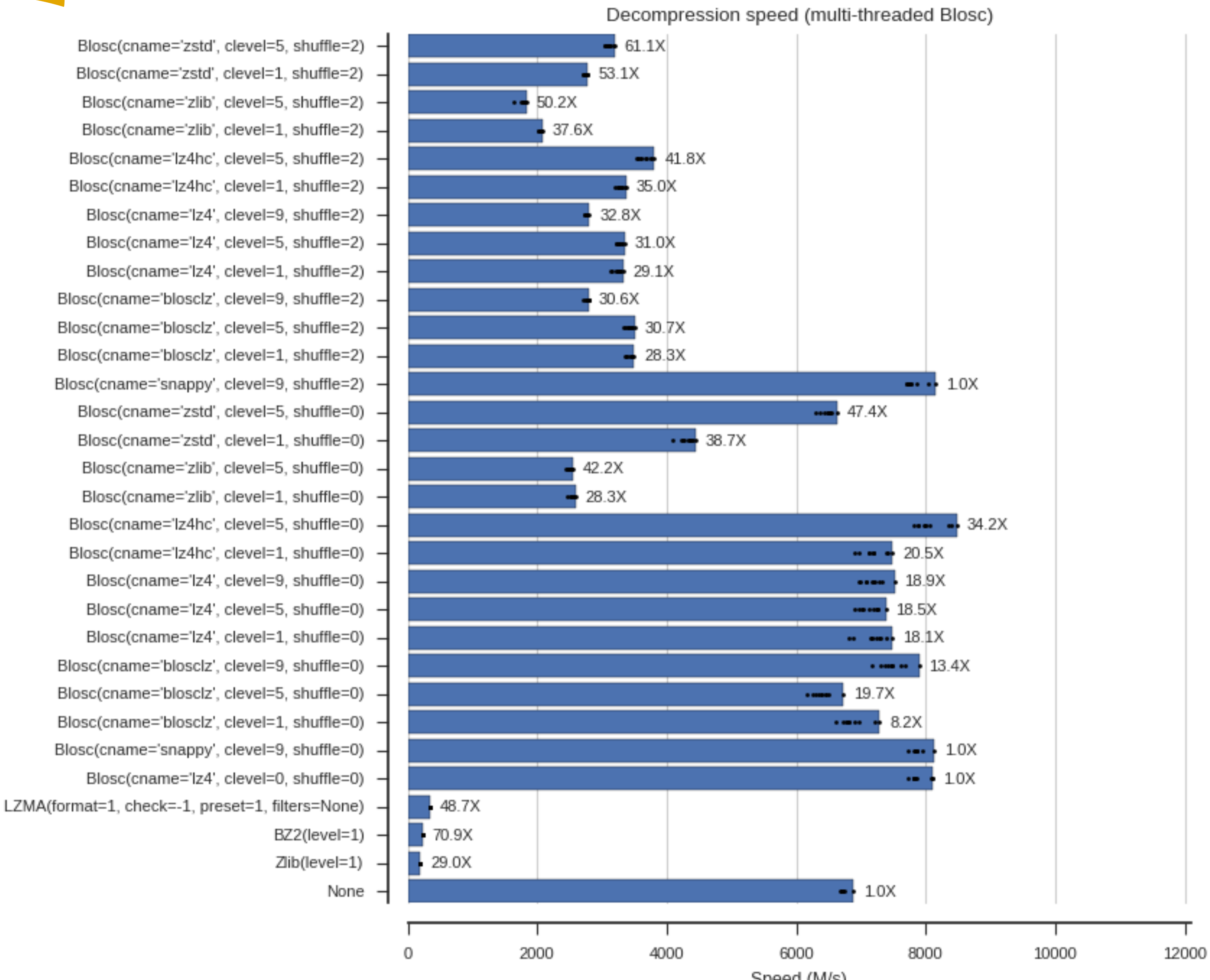
- It is already 10 years old, and since its inception it has seen a broad adoption in different libraries implementing data containers.

Compression Speed



http://alimanfoo.github.io/2016/09/21/genotype-compression-benchmark.html

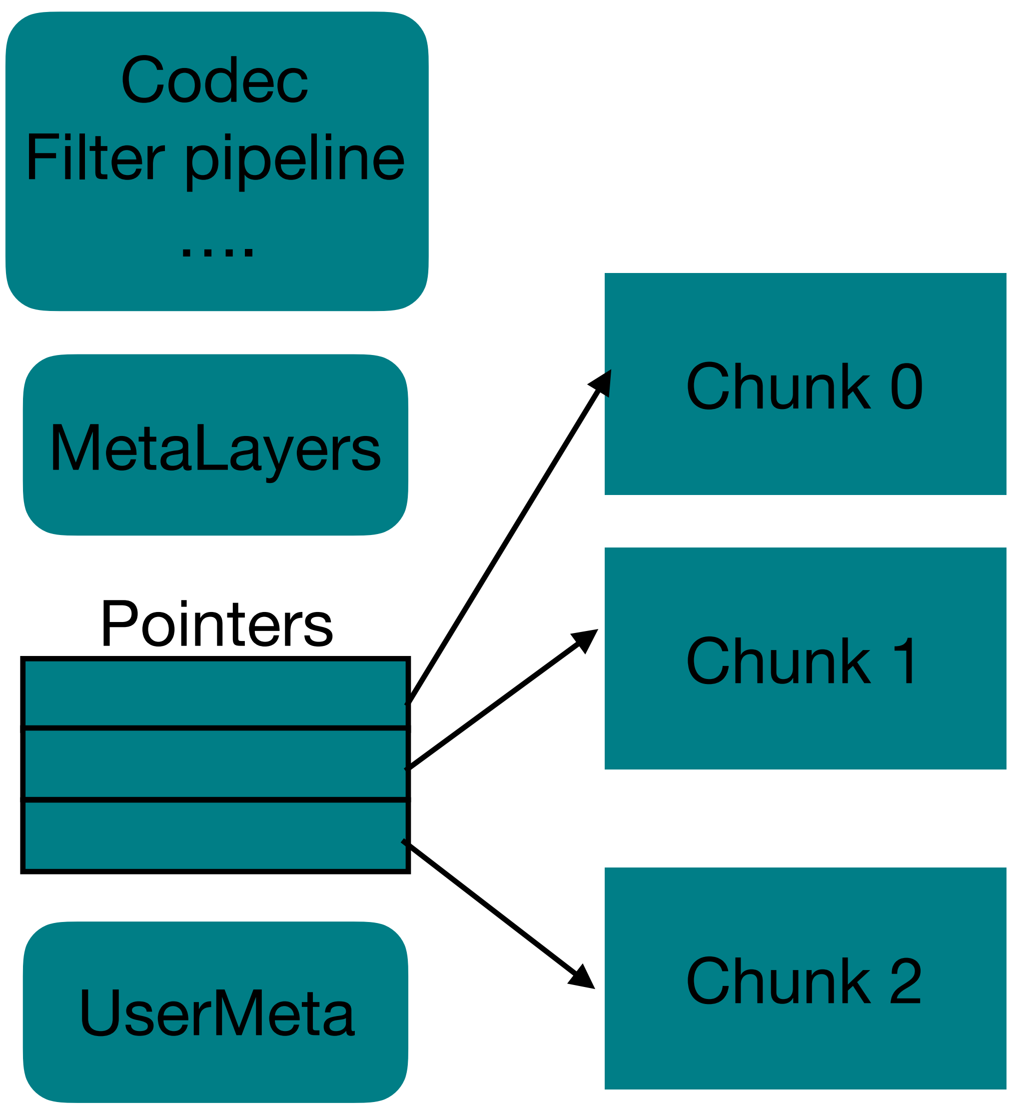
Decompression Speed



http://alimanfoo.github.io/2016/09/21/genotype-compression-benchmark.html

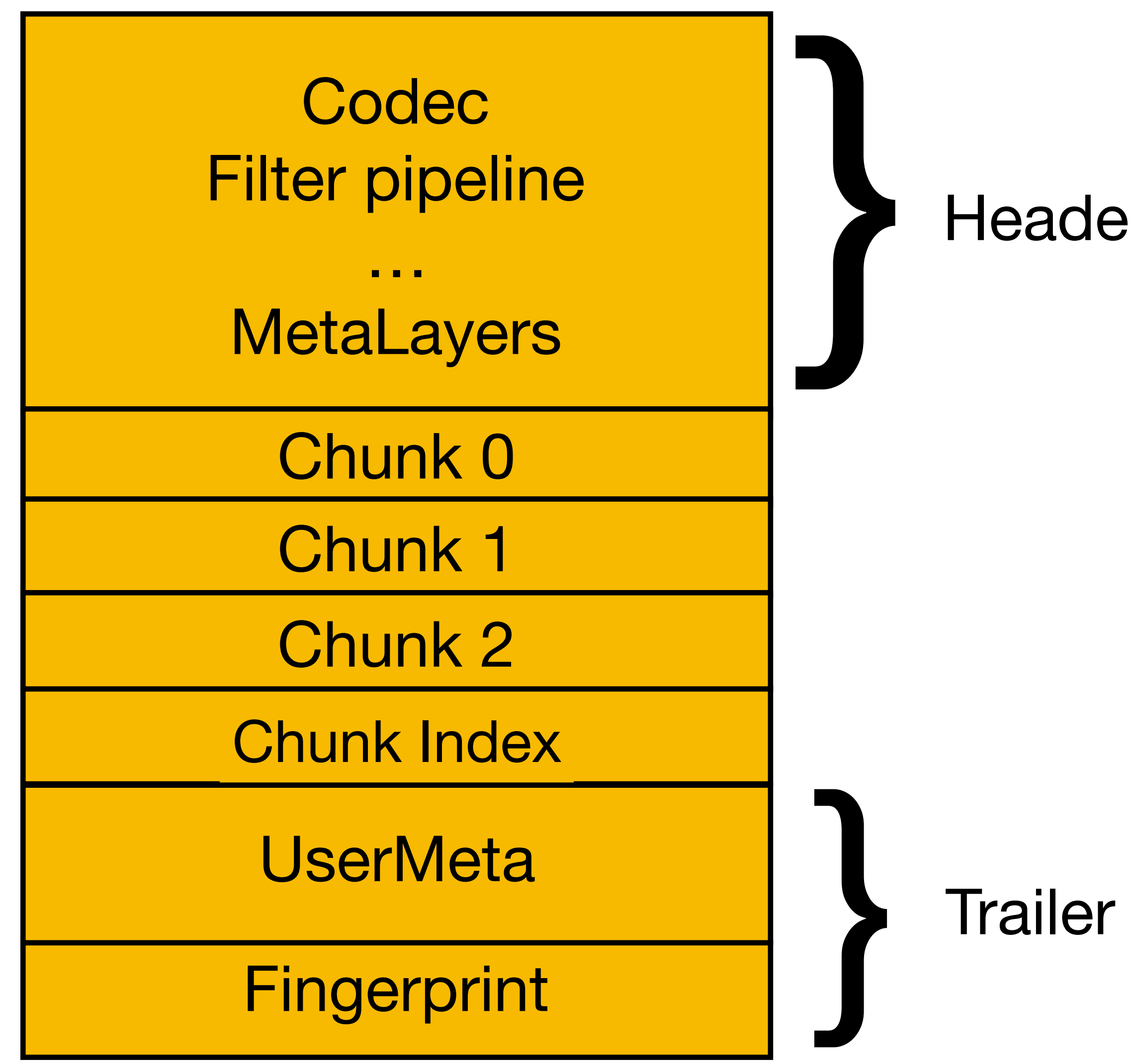
Containers in Blosc2

Super-chunk



- Sparse
- In-memory

Frame

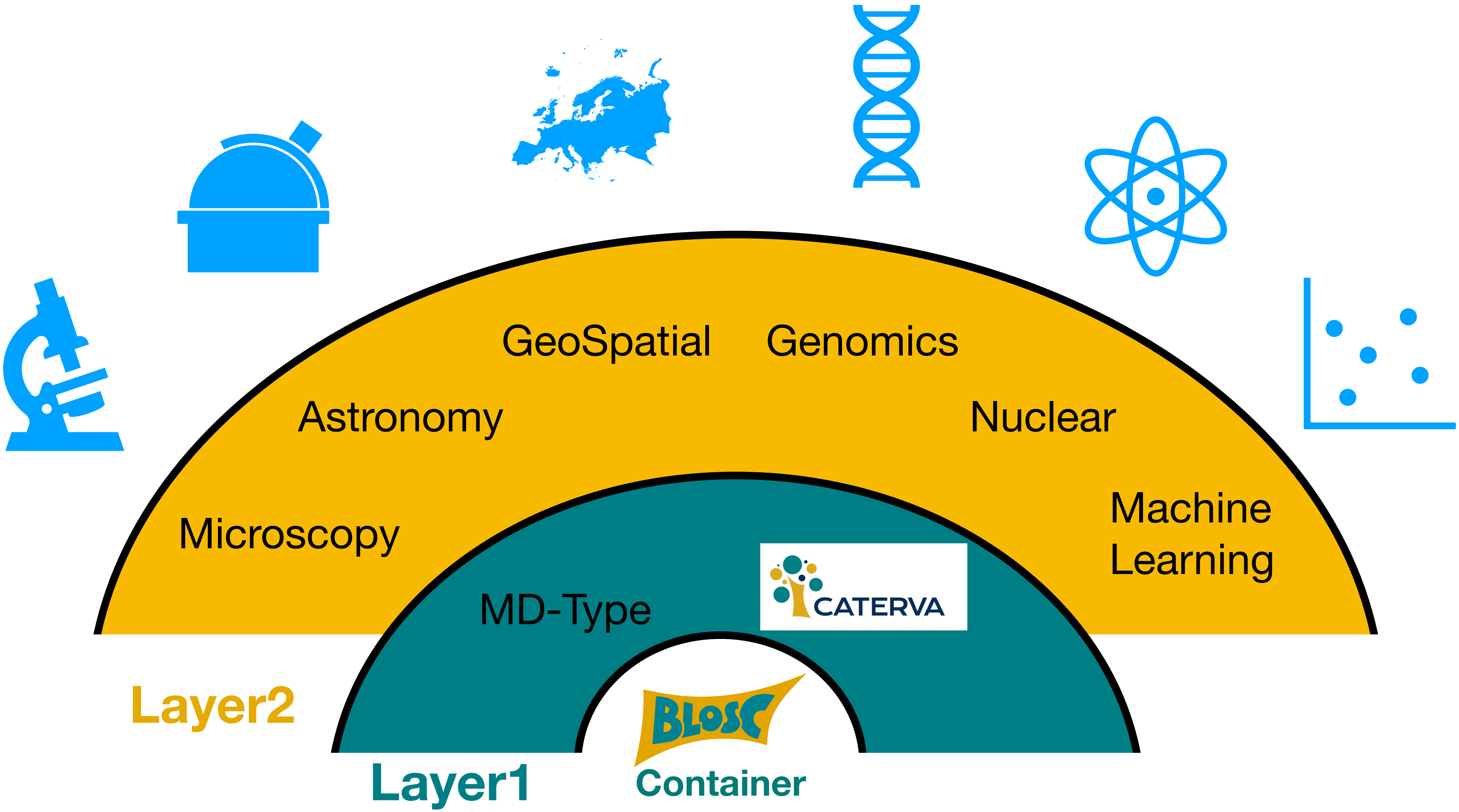


- Sequential
- In-memory / On-disk

MetaLayers in Blosc2

- Metalayers are small metadata for informing about the kind of data that is stored on a Blosc2 container.
- Example: Caterva defines a metalayer for multidimensional data, but another library can add another metalayer for specifying the type. [https://github.com/Blosc/cat4py/blob/master/notebooks/array-metalayer.ipynb]
- They are handy for defining layers with different specs: multi-dimensions, types, geo-spatial…

MetaLayers in Blosc2



Multiple layers to target different data aspects

Caterva MetaLayer

Caterva specifies a metalayer on top of a Blosc2 container for storing multidimensional information:

```
typedef struct {
    int8_t ndim;
    //!< The number of dimensions
    uint64_t dims[CATERVA_MAXDIM];
    //!< The size of each dimension
    int32_t pdims[CATERVA_MAXDIM];
    //!< The size of each partition dimension
} caterva_dims_t;
```

Such metalayer can be modified so that the shapes can be updated (e.g. an array can grow or shrink).

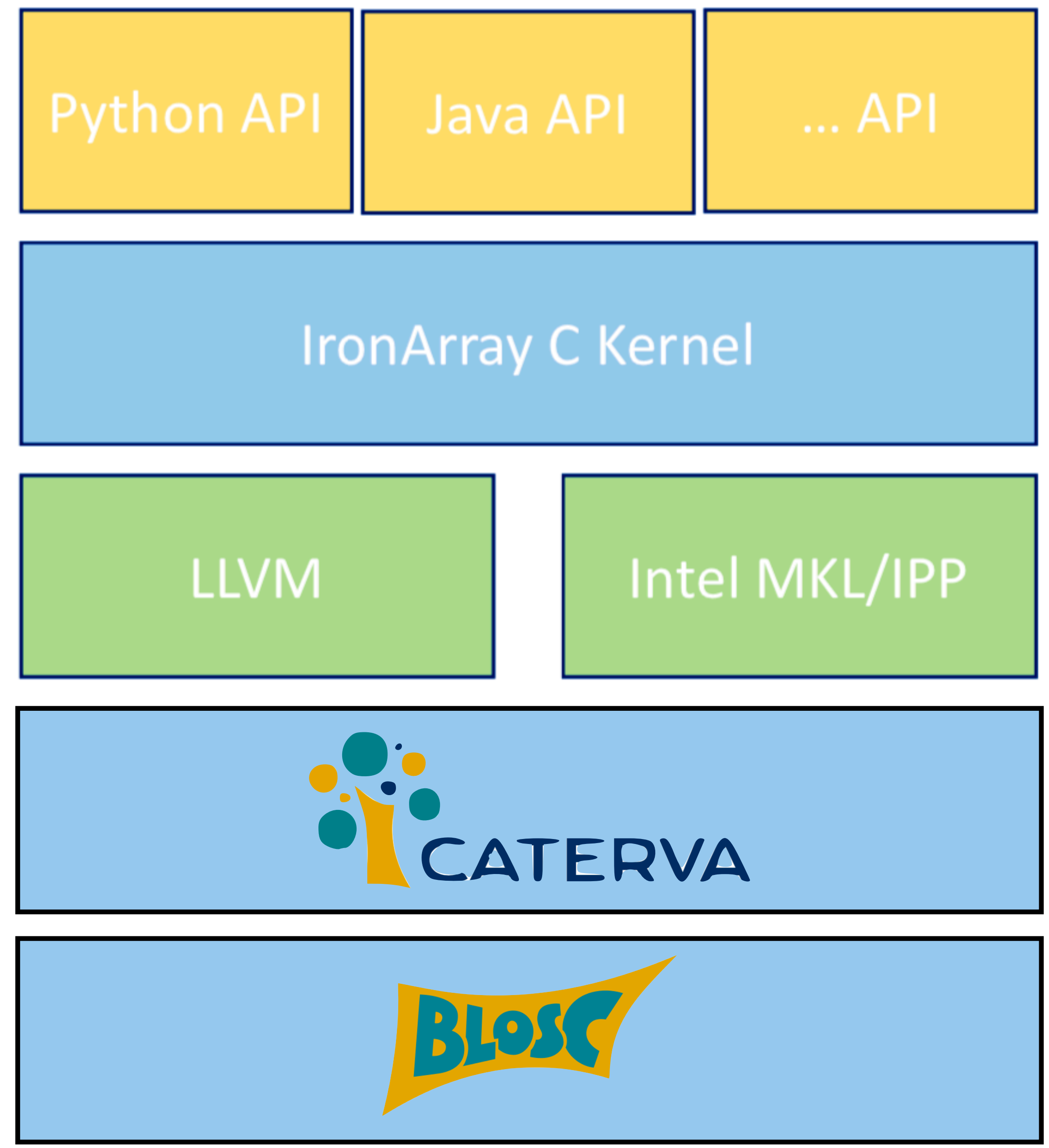
Frame Format and MetaLayers Specs

- The format for a Blosc2 frame is completely specified at:
  - https://github.com/Blosc/c-blosc2/blob/master/README_FRAME_FORMAT.rst
- The format for a Caterva metalayer:
  - https://github.com/Blosc/Caterva/blob/master/README_CATERVA_METALAYER.rst

Everything specified in the msgpack format.

IronArray

- High-performance matrix and vector computations.
- Optimized for compressed in-memory multi-dimensional data.
- Language specific API's leveraging a powerful C kernel.



Example of IronArray API for Python

```
import iarray as ia

shape = [10 * 1000 * 1000]
pshape = [100 * 1000]
cparams = dict(clib=ia.LZ4, clevel=1, nthreads=2)

xa = ia.linspace(ia.dtshape(shape=shape, pshape=pshape_), 0., 10., **cparams)

# Evaluate a polynomial
ya = ((xa - 1.35) * (xa - 4.45) * (xa - 8.5)).eval()
```

- There are wrappers for Java too.
- An R wrapper could also be interesting in the future.

IronArray Availability

- A beta version is planned for November.
- It will follow a production-ready version later in the year.
- Commercially available, with a possible Community Edition.

Stay tuned!

Overview

- Caterva is a C library and a metalayer for handling multidimensional data on top of Blosc2 containers.

- cat4py leverages Caterva and Blosc2 for building multidimensional containers that can be very large.

- You can use metalayers for adapting Blosc2/Caterva containers to your own needs.

    - **Beware**: Do not reinvent the wheel and build on top of existing metalayers!

Help Needed!

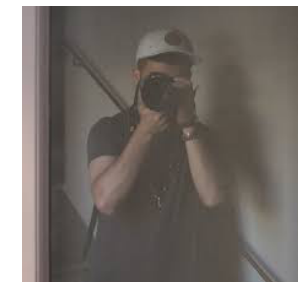
- Caterva and specially cat4py are quite new and still need a fair amount of love in terms of documentation, API consistency and testing in general before they are apt for general consumption.

- Please join us in our sprint on Friday morning if you are interested in helping us to reach production quality as soon as possible.

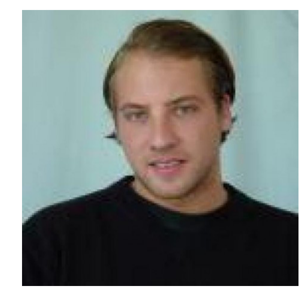
You are welcome!

Acknowledgements

- First and foremost to Aleix Alcacer who contributed most of the code behind Caterva.



- Christian Steiner, for suggestions and improvements on Blosc2 / Caterva projects.



- Pepe Aracil, for his proposal for using msgpack for serializing Blosc2 containers.

- Last but not least, NumFOCUS for providing funding for developing Blosc2 and Caterva.

NUMFOCUS
OPEN CODE = BETTER SCIENCE

Thank You!

Questions?

CATERVA