Report on improvements in the HDF5/Blosc2 integration

Francesc Alted / @FrancescAlted@masto.social

The Blosc Development Team / @Blosc2@fosstodon.org

CEO ironArray / francesc@ironArray.io

LEAPS INNOV Meeting -- Kraków, Poland
April 8th 2024

Agenda

Plugins for **JPEG2000**

Support for **Blosc2 Ndim in HDF5**

Btune: Predicting the **best codecs and filters**

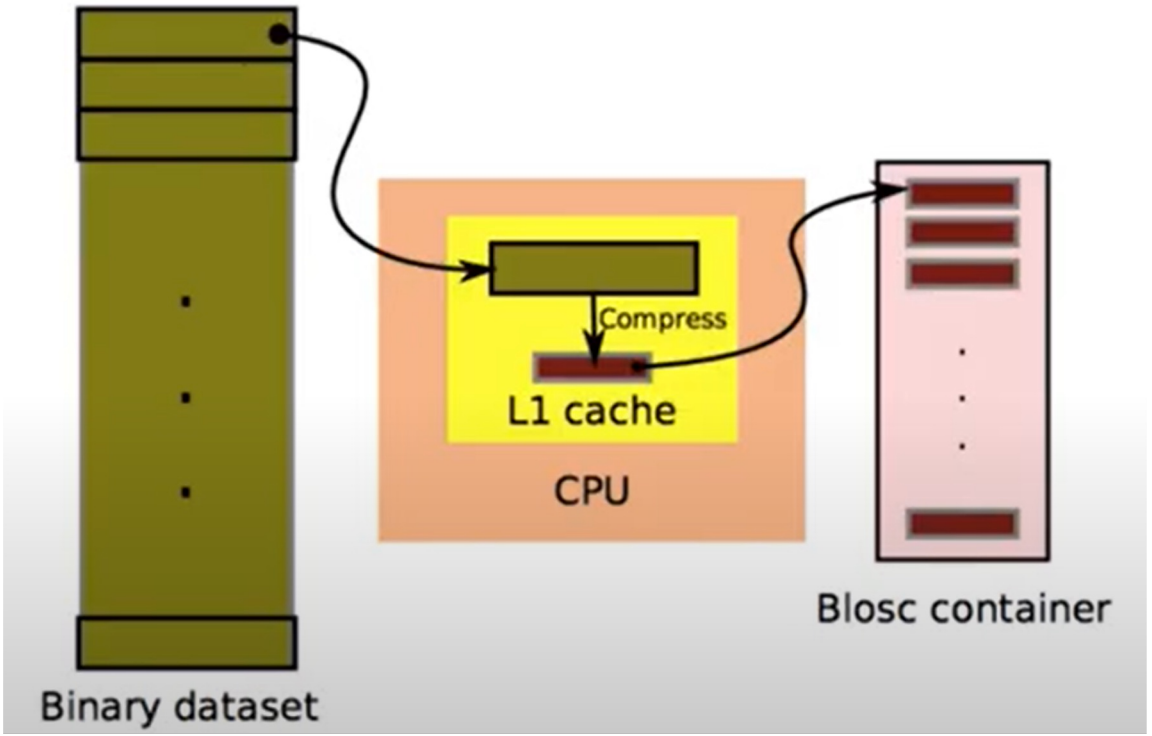
Handling **sparse datasets** with Blosc2

Caterva2: On-demand access to local/remote Blosc2/HDF5 data repositories

Intro

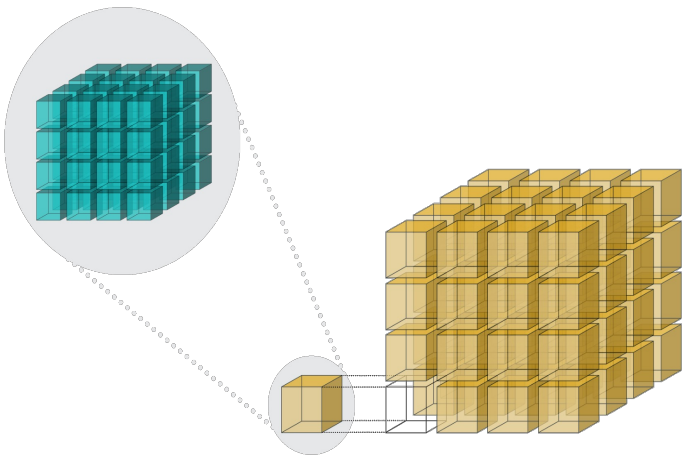
What is Blosc?

- A collection of codecs and filters for compressing binary data
- Goal: sending data from memory to CPU (and back) faster than *memcpy()*.
- Combining chunking and blocking: divide and conquer.



What is Blosc2?

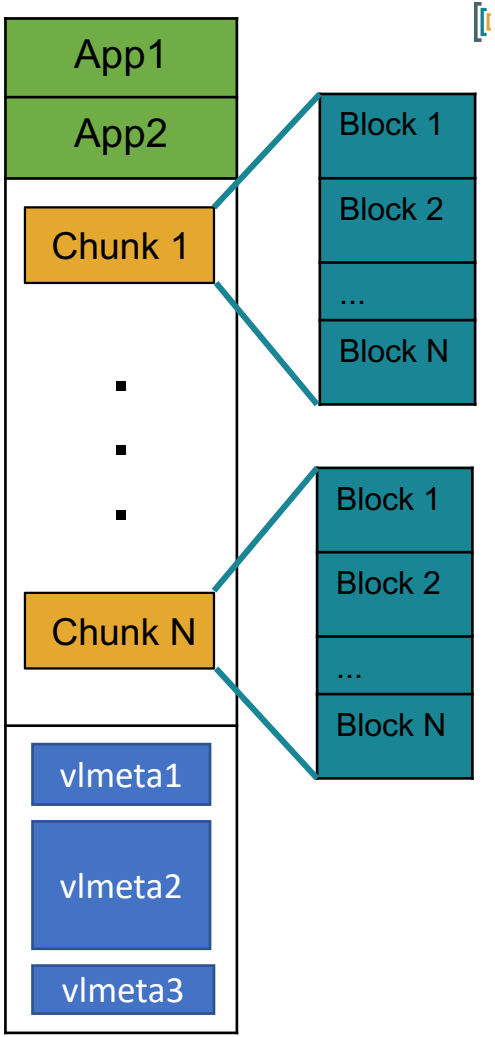
- ✓ Adds 63-bit containers.
- ✓ Metalayers for adding info for apps and users.
- ✓ Multidimensional blocks and chunks.



Header:
Fixed Length
Metalayers

Data:
Super-Chunk

Trailer:
Var Length
Metalayers
(up to 2 GB)



Who is ironArray SLU?

- We are the developers of PyTables, numexpr and Blosc ecosystems
- Team of experts empowering you to harness the full potential of compression for big data: we are here to help!





ironArray

https://ironarray.io

Plugins for JPEG 2000

Introducing grok and OpenHTJ2K dynamic plugins

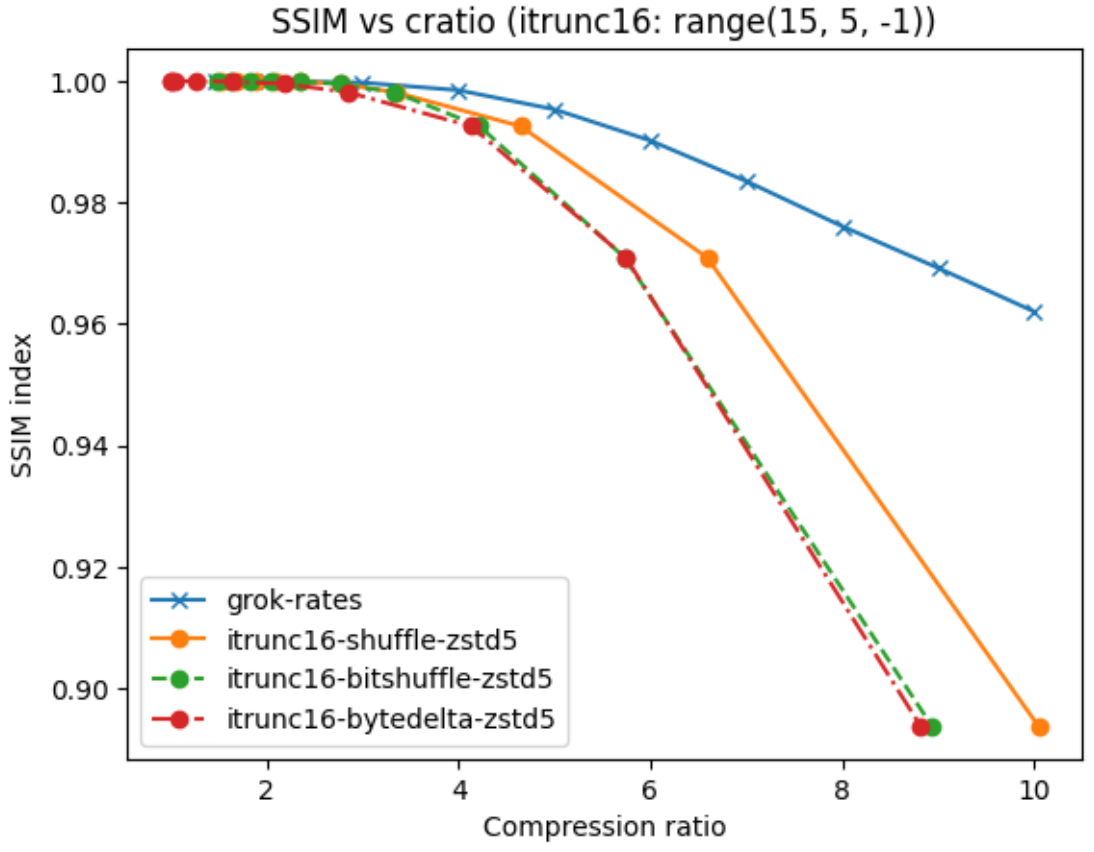
- OpenHTJ2K, an open source HTJ2K implementation by Osamu Watanabe.
- Grok, another free implementation for HTJ2K by Grok Image Compression Inc.
- Packed and distributed as Python wheels:
  - `$ pip install blosc2-openhtj2k`
  - `$ pip install blosc2-grok`

Grok supports 16-bit gray images, while OpenHTJ2K is only 12-bit

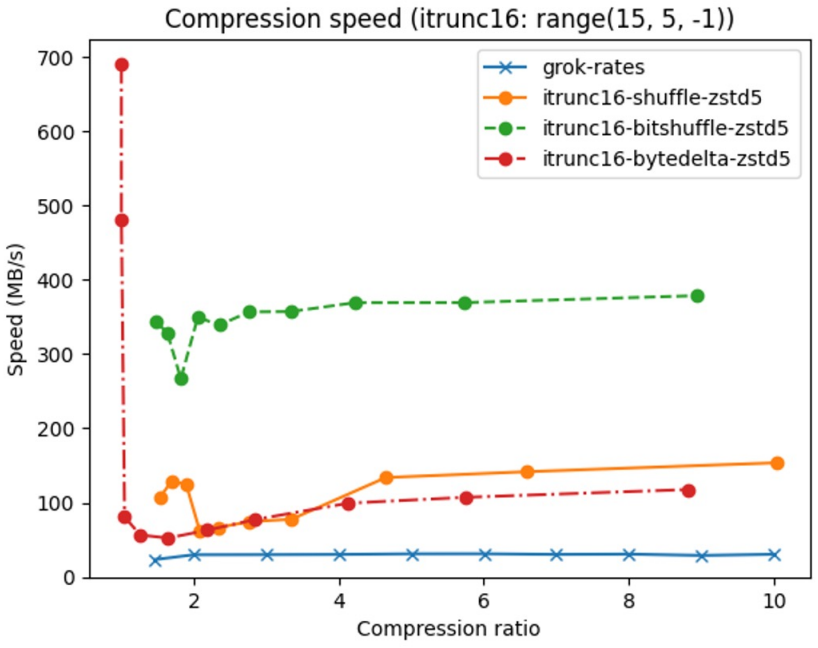
Lossy compression with grok and itrunc+zstd

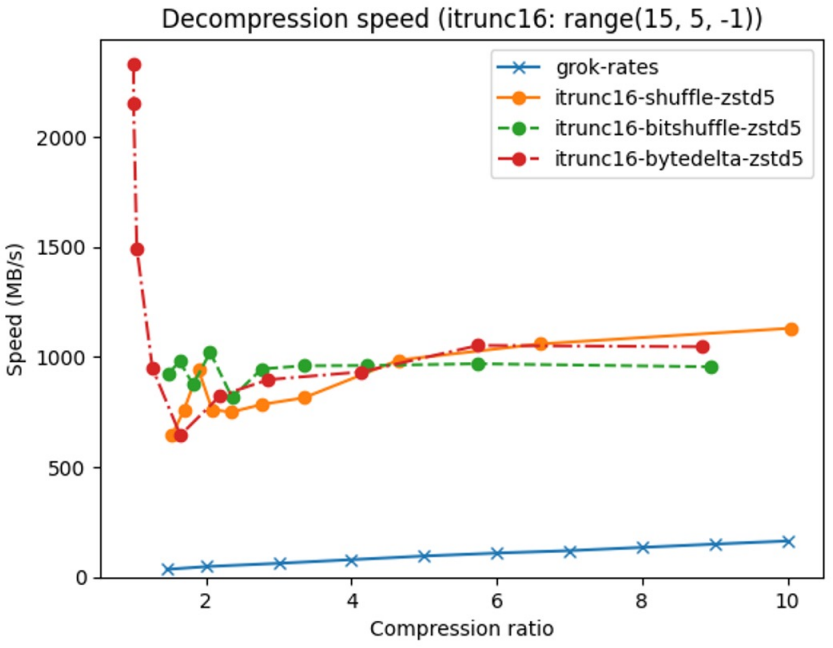
- JPEG 2000 can achieve much better quality for the same compression ratio.
- For low compression ratios, itrunc can provide similar quality.

Dataset:
http://www.silx.org/pub/leaps-innov/tomography/lung_raw_2000-2100.h5



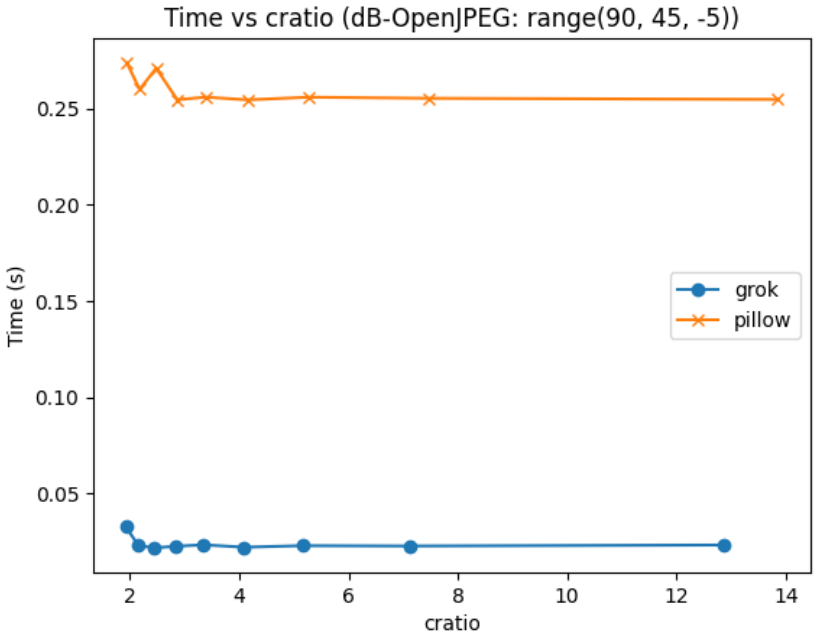
Lossy compression with grok and itrunc+zstd





Speed-wise, itrunc+bitshuffle+zstd is very competitive

JPEG 2000 in grok is still very fast!



Same order of cratio than OpenJPEG, but 10x faster

New: Ability to link with C/C++ Apps

- We recently added the possibility to use the blosc2-grok plugin with C/C++ applications.
- You can **tweak almost all the params** that grok allows: https://github.com/Blosc/blosc2_grok?tab=readme-ov-file
- This allows JPEG 2000 to be used in scenarios where C/C++ is the main language (e.g. acquisition devices).
- See example using HDF5 + Blosc2 + grok at:

https://github.com/Blosc/leaps-examples/tree/main/c-compression

Future Work

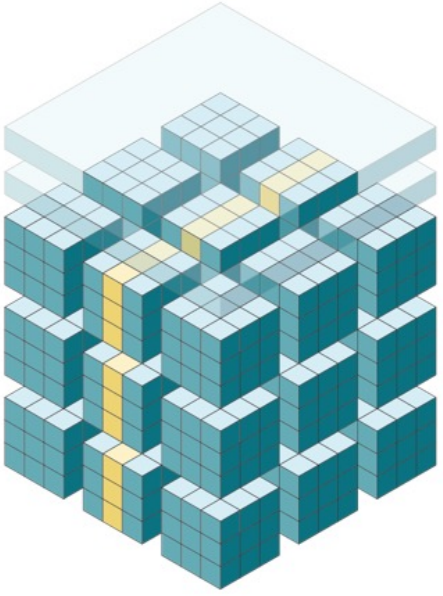
- WebAssembly
  - JPEG 2000 has a lot of potential to be sent to a browser and be decompressed in-place (much less data to send).
  - **Blosc2 (+ plugins) in the browser** (see demo on Caterva2 later)

- Better **interaction with hdf5plugin** for setting different parameters (cratio, dB...). For now, this is possible via HDF5 direct chunking.

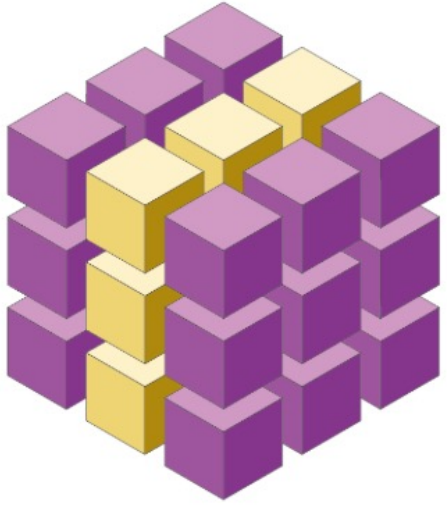
Support for Blosc2 NDim in h5py / HDF5

Leveraging the second partition in Blosc2 NDim

Much more selective and hence, faster queries!



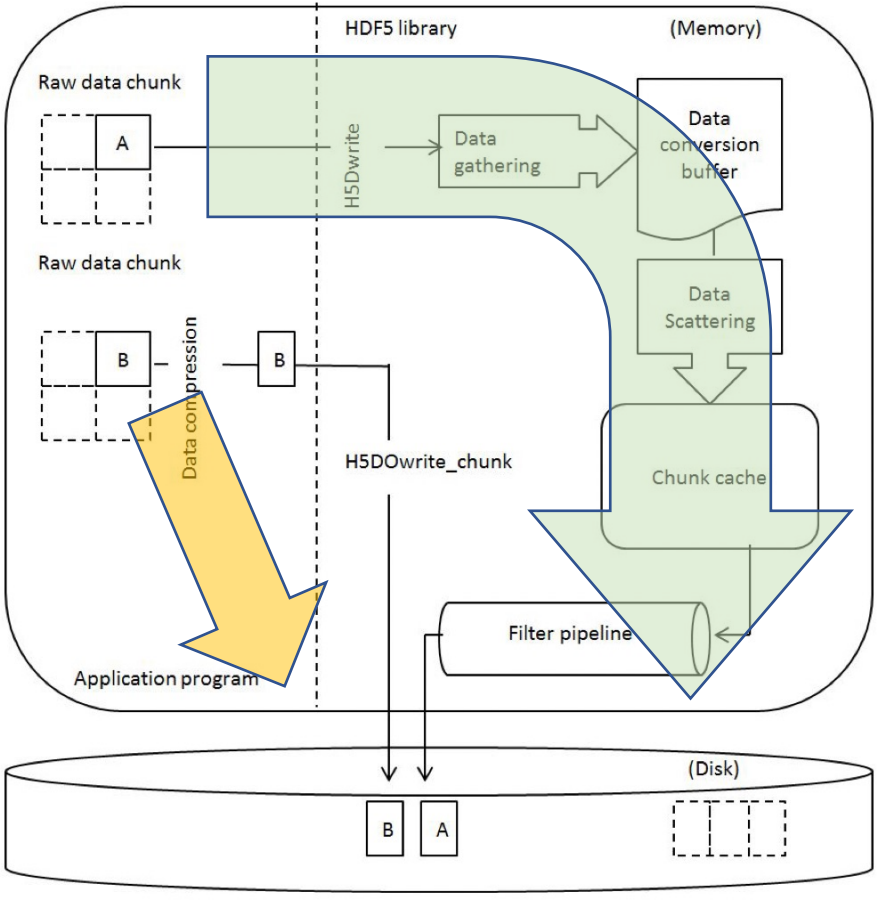
Blosc2 NDim



HDF5 / Zarr / others

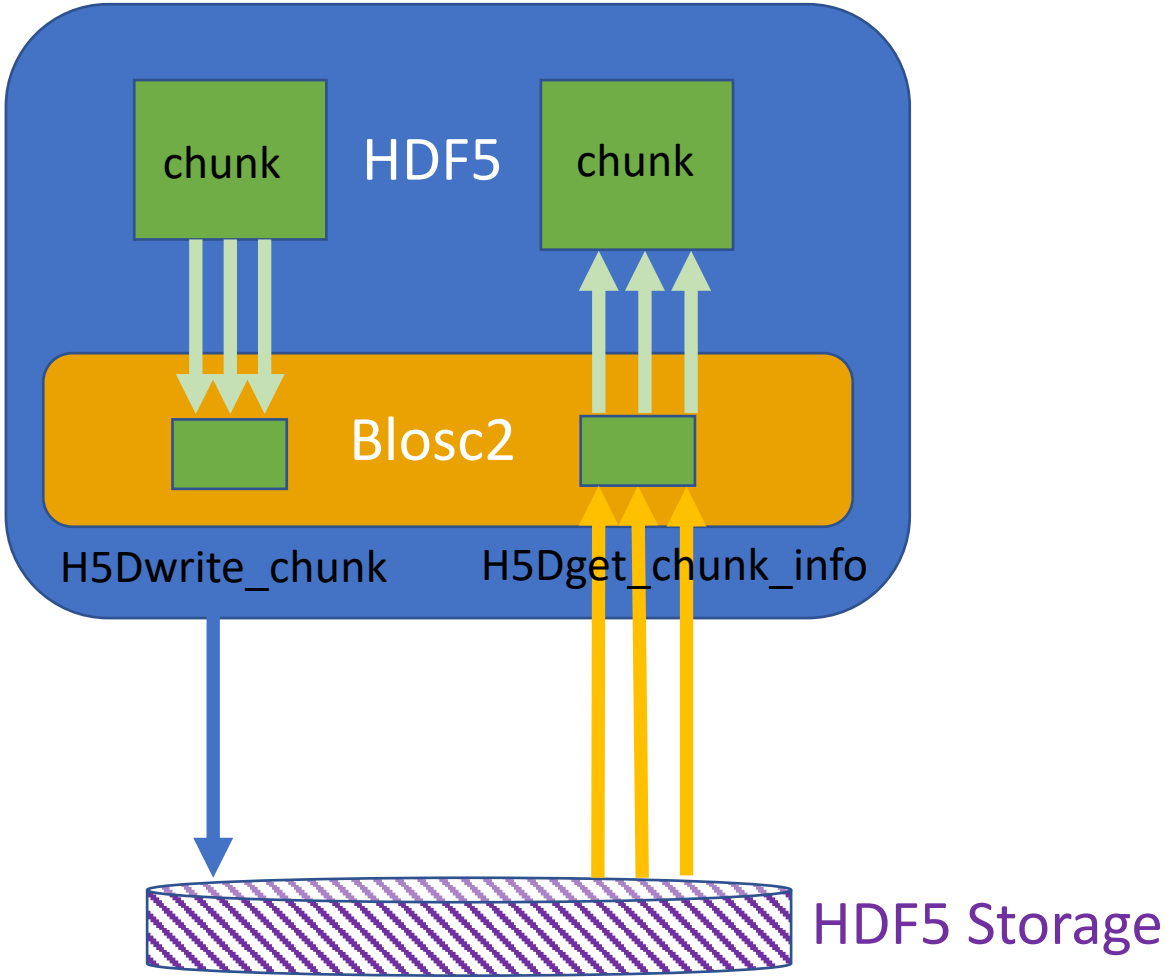
Bypassing the HDF5 pipeline: Direct Chunking

- HDF5 pipeline implementation is powerful but known to be slow.
- This can be bypassed using direct chunking in HDF5. **Integrated in new b2h5py**.
- New version of Blosc2 plugin for HDF5. It has been **included in hdf5plugin**.



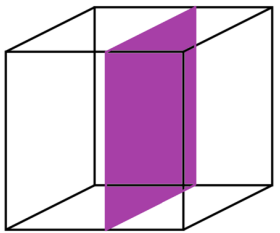
b2h5py: Use Blosc2 Inside Direct Chunking

- All compression and decompression executed in parallel via Blosc2!
- Blosc2 can do parallel I/O for reads
- Blosc2 can do chunk reads with enhanced selectivity from disk
- Data can still be read with hdf5plugin and h5py.

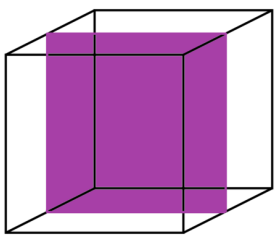


https://github.com/Blosc/b2h5py

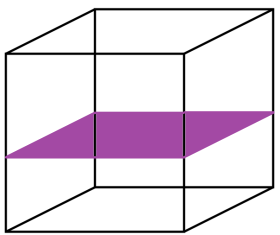
HDF5 pipeline vs direct chunking: Reading orthogonal slices with b2h5py



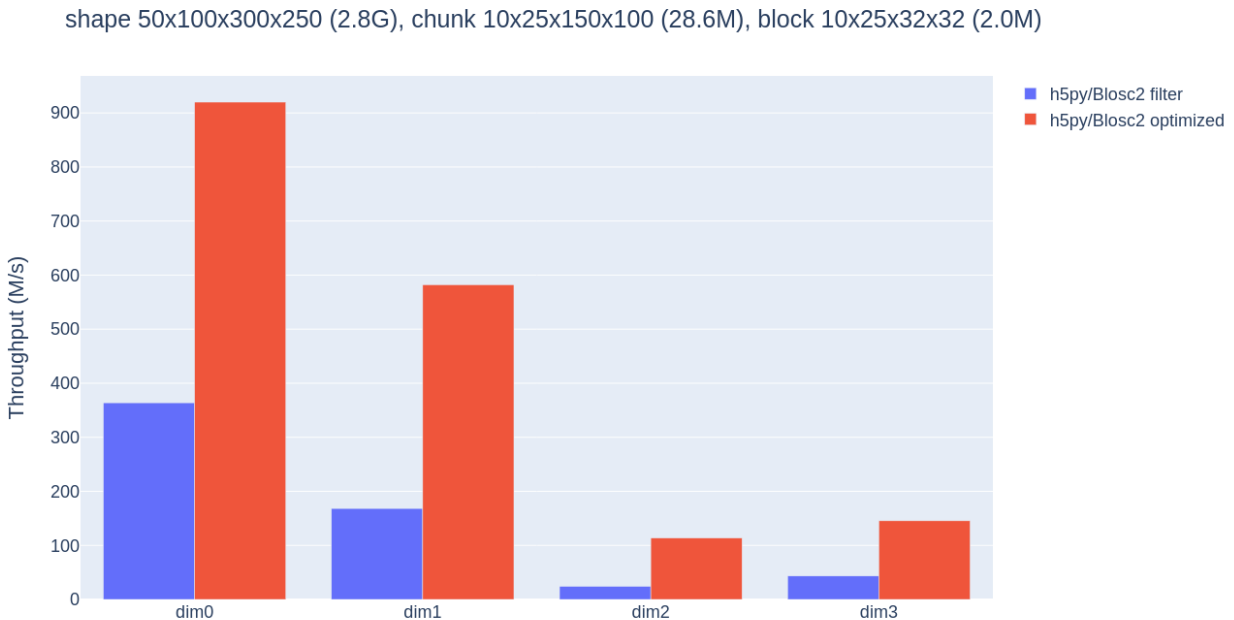
constant dim0



constant dim1



constant dim2



Faster slicing due to higher data selectivity in double partitioning

Btune: automatic selection of the best codecs / filters

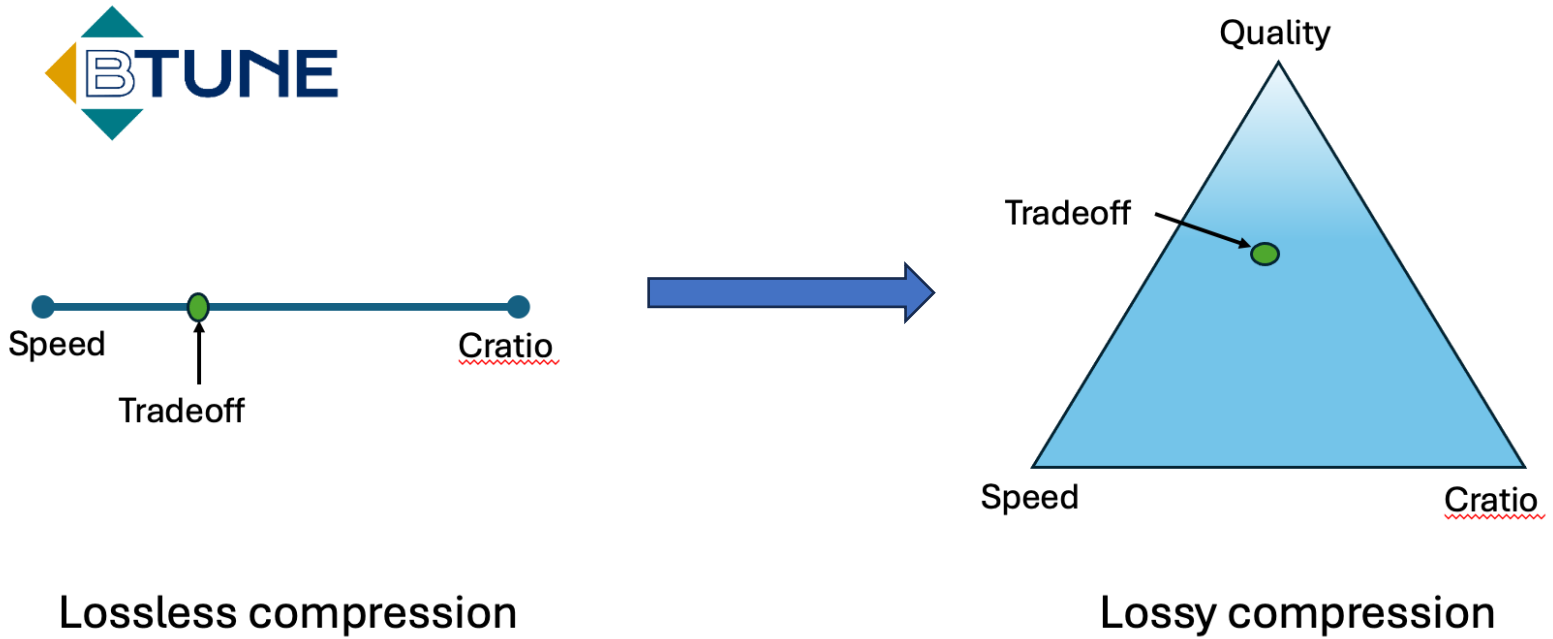
Allowing selection of Btune params programmatically

```
kwargs = {
    "tradeoff": 0.3,
    "perf_mode": blosc2_btune.PerformanceMode.DECOMP,
    "models_dir": f"{base_dir}/models/"}
blosc2_btune.set_params_defaults(**kwargs)
```

With that, and after a training, Btune predicts the best parameters **per chunk**

https://btune.blosc.org

New Lossy Mode in Btune



Works by combining neural networks and heuristics

Example of Prediction of Lossy Codecs

Example with tradeoff `(cratio=0.7, speed=0.2, quality=0.1)`

```
(btune_arm64) martaiborra@MacBook-Air examples % BTUNE_TRADEOFF="(0.7, 0.2, 0.1)" BTUNE_TRACE=1 python quality_mode.py
Performing compression using Btune
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Btune version: 1.1.2
Performance Mode: COMP, Compression tradeoff: (0.700000, 0.200000, 0.100000), Bandwidth: 20 GB/s
Behaviour: Waits - 0, Softs - 5, Hards - 10, Repeat Mode - STOP
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
TRACE: time load model: 0.000294
|    Codec    | Filter | Split | C.Level | C.Threads | D.Threads |  S.Score  |   C.Ratio    |   Btune State   | Readapt | Winner
|       grok |      0 |     0 |       5 |         4 |         4 |   0.0328 |           8x |    CODEC_FILTER |    HARD | W
|       grok |      0 |     0 |       5 |         4 |         4 |   0.0543 |           8x |    CODEC_FILTER |    HARD | W
|       grok |      0 |     0 |       5 |         4 |         4 |   0.0554 |           8x |    CODEC_FILTER |    HARD | W
|       grok |      0 |     0 |       5 |         4 |         4 |   0.0547 |           8x |    CODEC_FILTER |    HARD | -
|       grok |      0 |     0 |       5 |         4 |         4 |   0.0552 |           8x |    CODEC_FILTER |    HARD | -
Cratio:  8.001620890749567
Compression speed (GB/s):  0.04857454307398124
Minimum ssim:  0.908711549595501
```

In this case, cratio was important, but quality not that much, so grok with a cratio 8x is being predicted per every chunk.

Challenges for Btune

- It does not have a good (and fast) estimator for the image quality

  (This is why we are using heuristics here)

- There is **great potential on finding image quality estimator**

  Nice (and quite challenging) project for the future

Handling sparse data

Compressing sparse data with Blosc2

Blosc2 has many provisions for compressing sparse data:

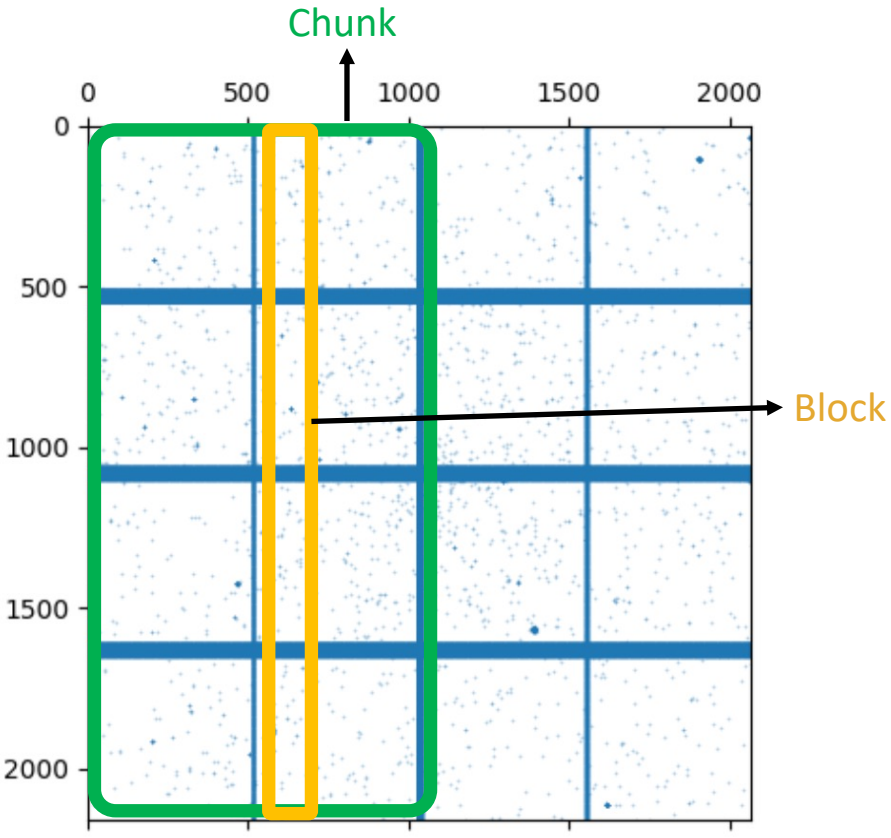
- Blocks of zeros can be represented by just 4 bytes
- Chunks of zeros can be represented by just 8 bytes
- Sequence of several chunks of zeros can be represented with 8 bytes.

Automatic zero detection:

- Such runs (blocks or chunks) of zeros can be detected automatically, but you can provide chunks of zeros explicitely too.

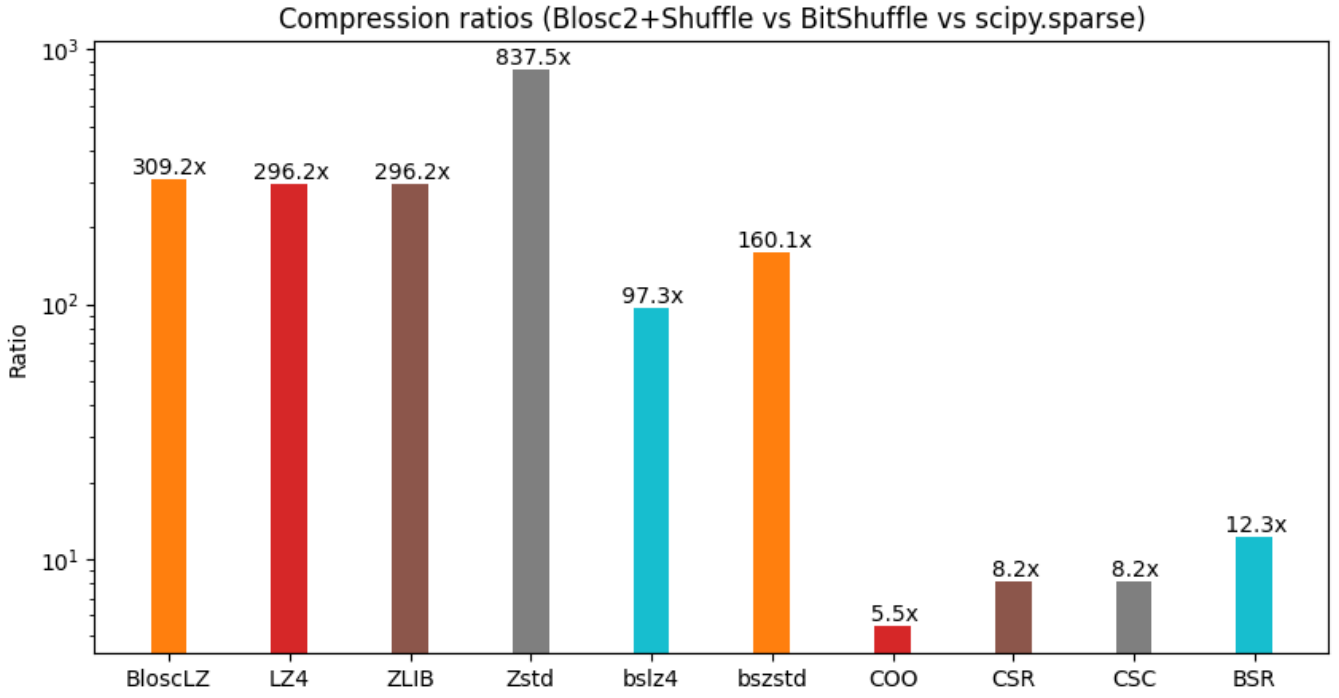
Example: X-ray diffraction

- A sample image. A tomography can be formed by 1000's of them.
- When compressing, it is important to be able to specify different partitions: this can make a **huge difference** in compression ratio, or speed.
- Blosc2 allows to do that in two-level, multidimensional partitions.



http://www.silx.org/pub/leaps-innov/sparse_image_stack.h5

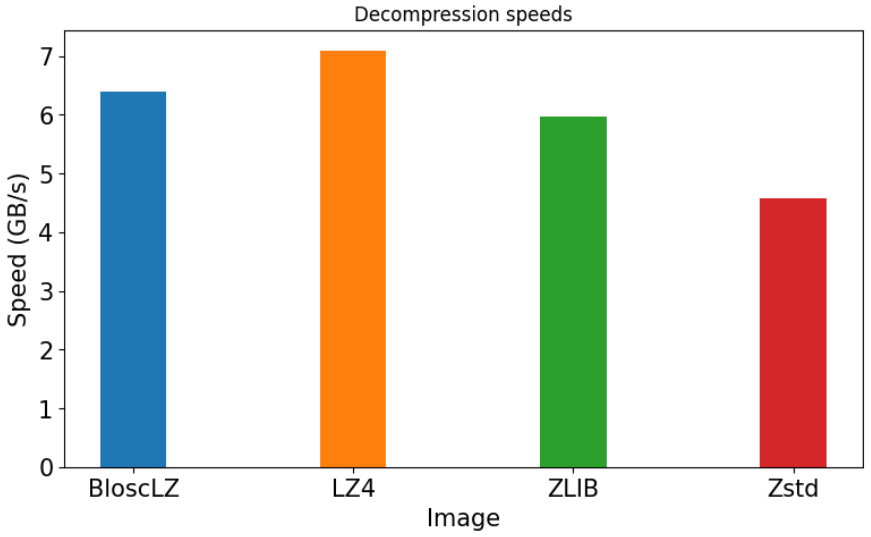
Example: X-ray diffraction



Blosc2+Shuffle+Zstd shines with this sparse dataset

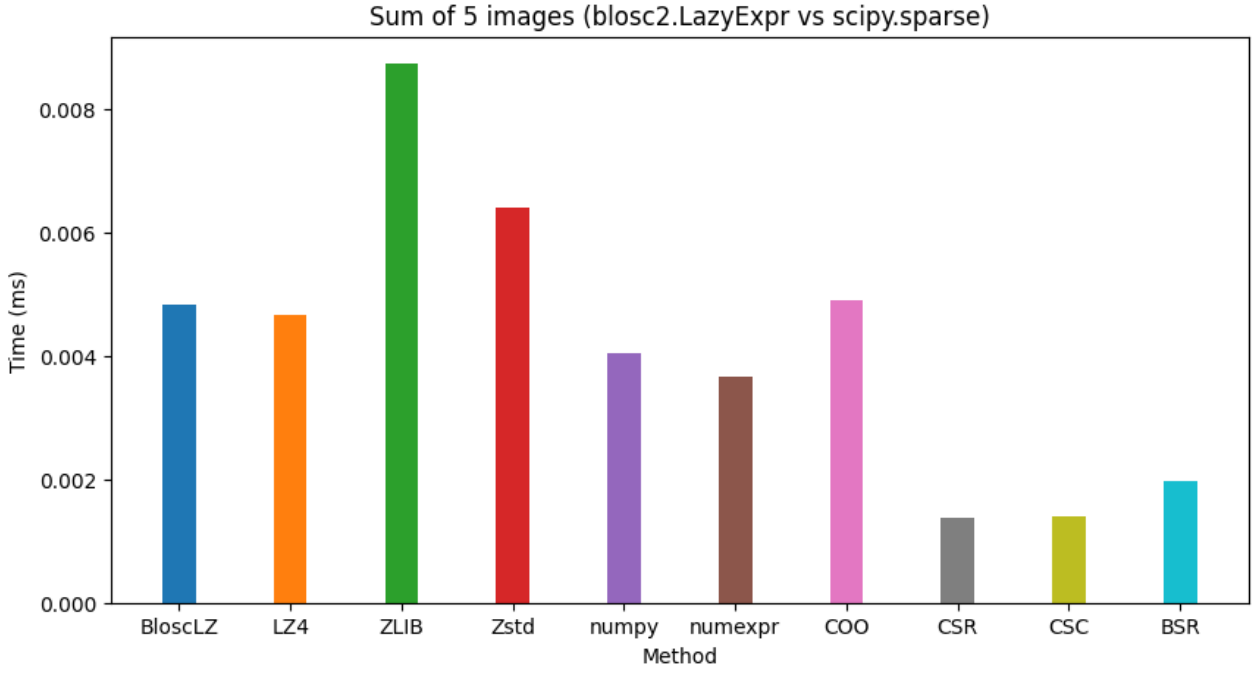
https://github.com/Blosc/leaps-examples/tree/main/sparse

Example: X-ray diffraction



Blosc2+Shuffle+LZ4 shows good balanced speed (~1000 fps)

Computing with sparse data



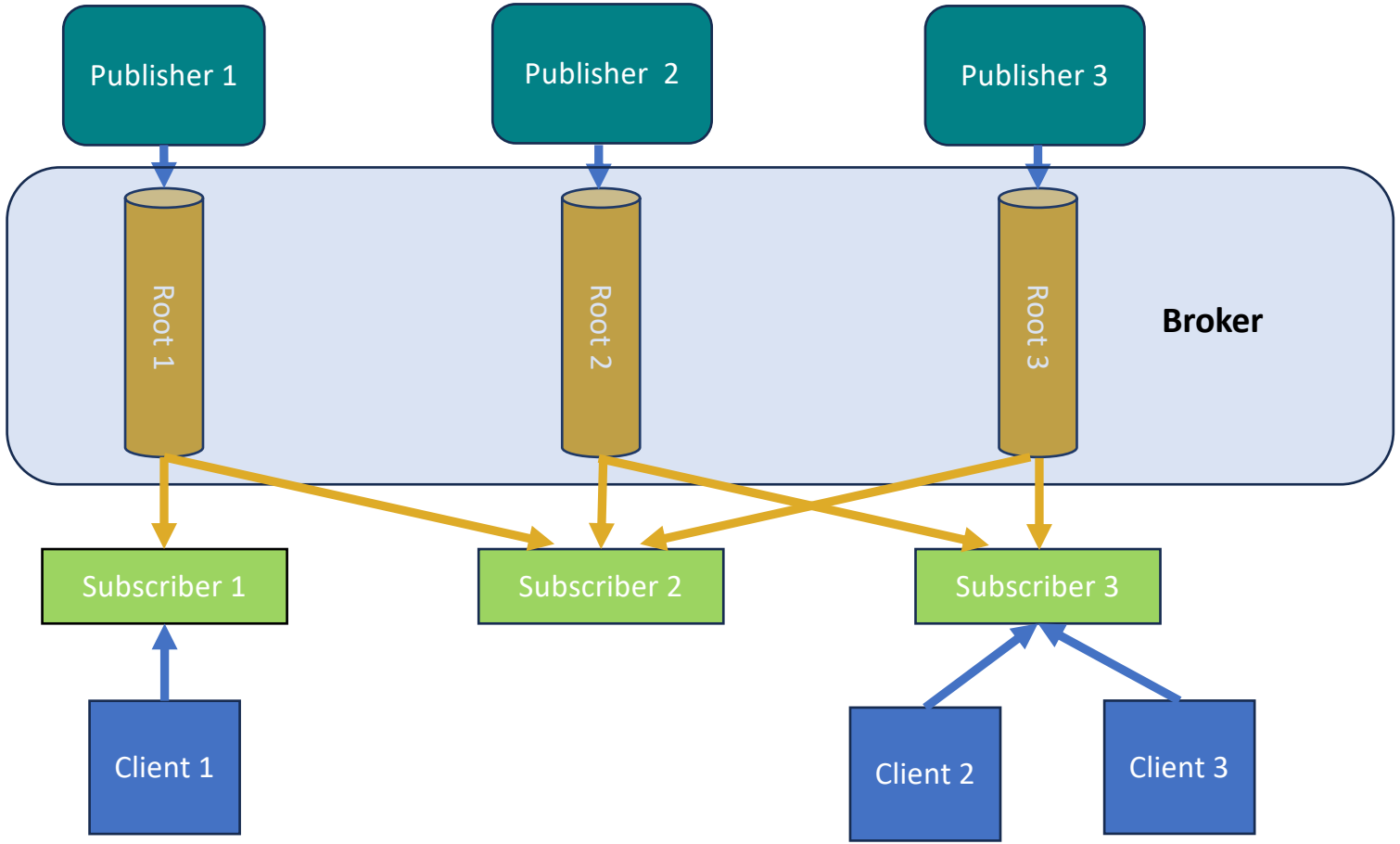
New LazyExpr computation engine in Blosc2:
summing at 1000 fps

Work for the future

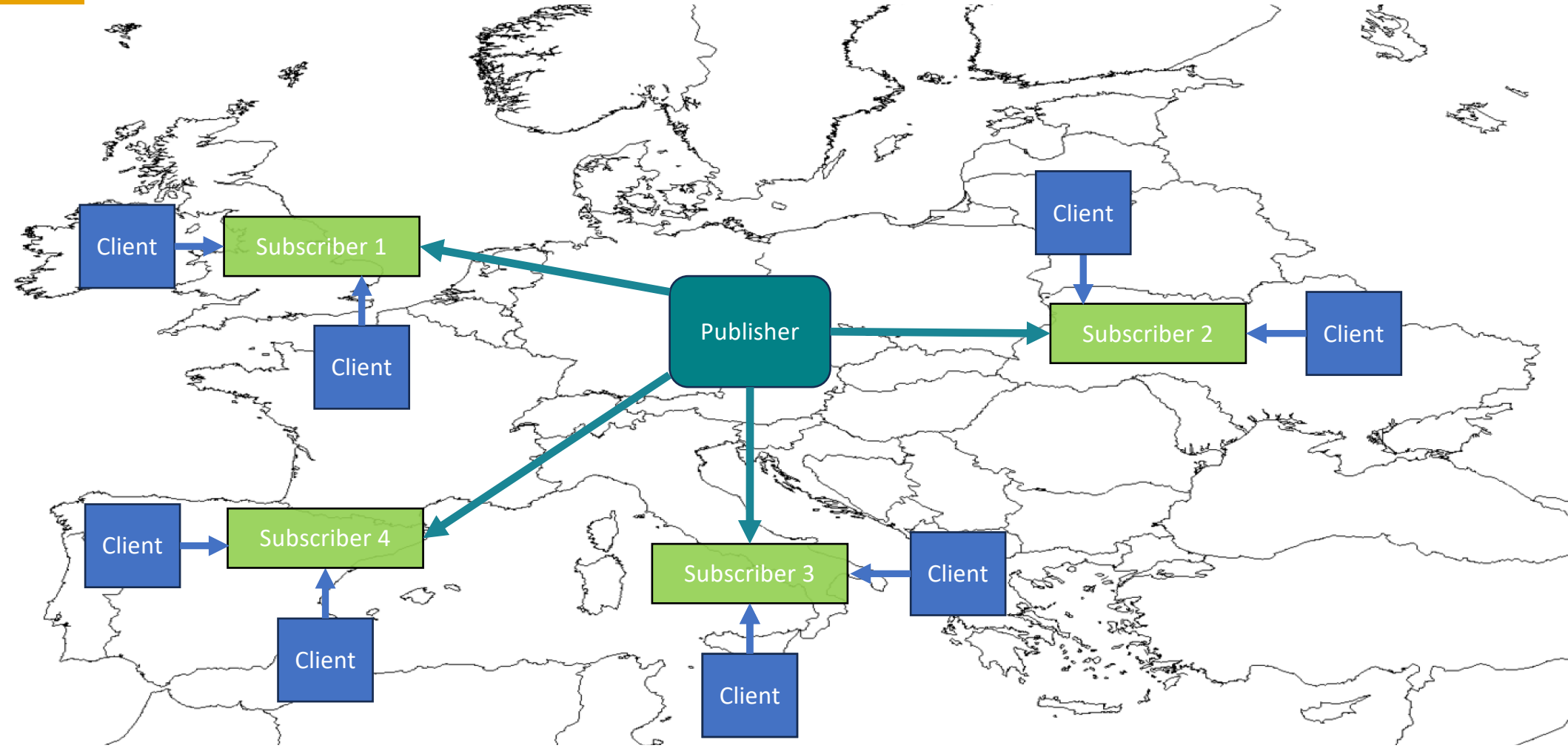
- It should be nice to **skip computations on blocks/chunks that are full of zeros**
- Add **linear algebra** computations to blosc2.NDArray instances
- Other functionality (FFTs)?

Caterva2: On-demand access to local/remote Blosc2/HDF5 datasets

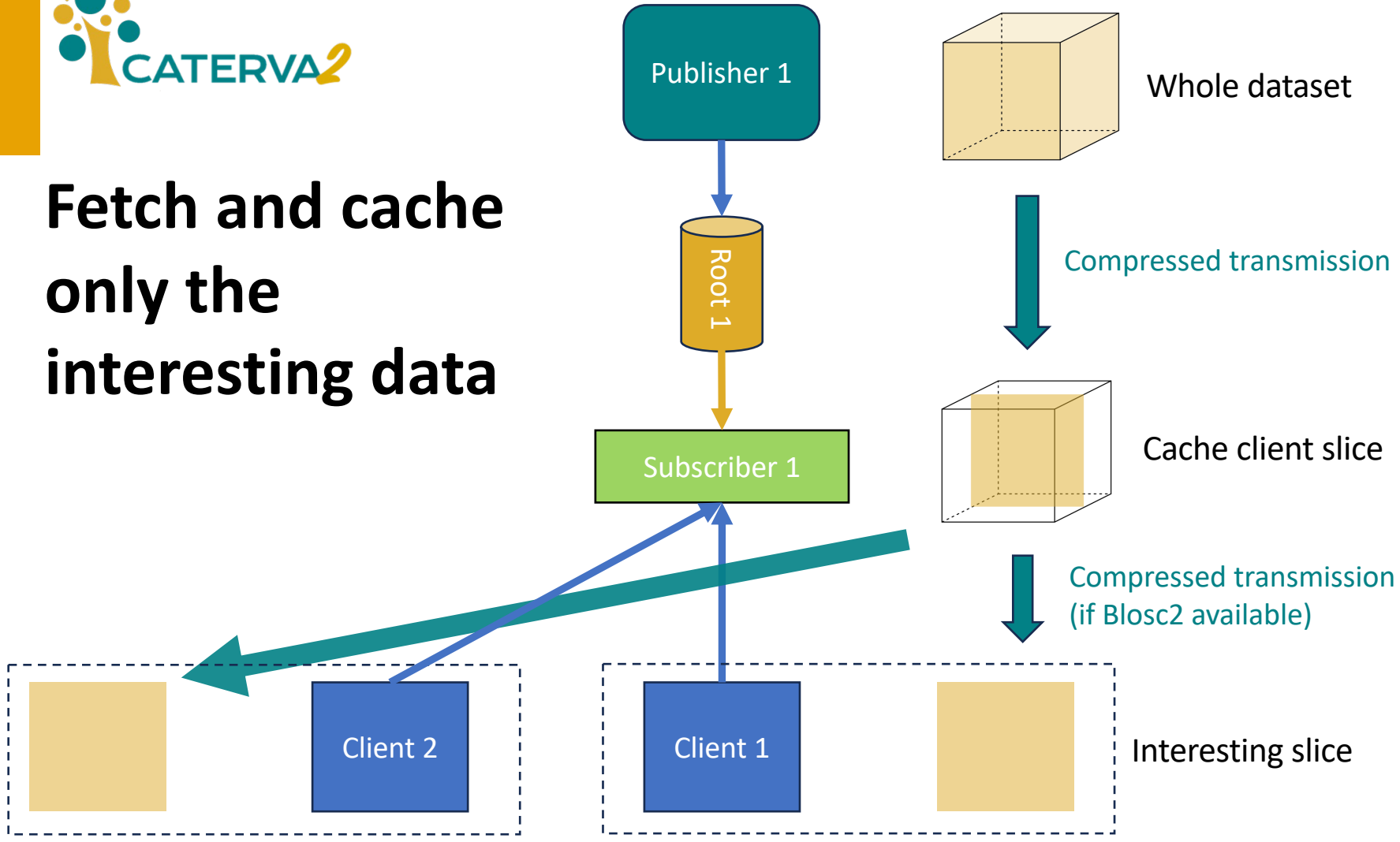
PubSub Data Flow



Putting data closer to the user



Fetch and cache only the interesting data



Demo time

- Go to demo.caterva2.net and try the interface by yourself.
- The demo box is a cheap 8 GB RAM, 64 GB disk and 4 cores, running Ubuntu 22.04 and in aarch64.
- Provider is hetzner.com in Nuremberg, Germany (so near to Krakow).

Work for the future

- Integrate LLM in the search box
- More plugins (on demand; suggestions?)
- Increase stability
- Make cache eviction more fine grained (now all the dataset is thrown away when it changes in the publisher)

Conclusion

Progress made in integrating Blosc2 with HDF5

- **Plugins for High Throughput JPEG 2000**
- Implemented **native support for Blosc2 NDim in HDF5**, bypassing the HDF5 pipeline
- **Btune, has got support for lossy compression** when predicting the best Blosc2 parameters
- **Caterva2, making Blosc2/HDF5 data generally available with easy and efficiency.**

Blosc2: a highly efficient and flexible tool for
compressing your data, your way

Koniec and thanks! Questions?

contact@ironarray.io

We make compression better