Recent improvements in the HDF5/Blosc2 plugin systems

Francesc Alted / @FrancescAlted

The Blosc Development Team / @Blosc2

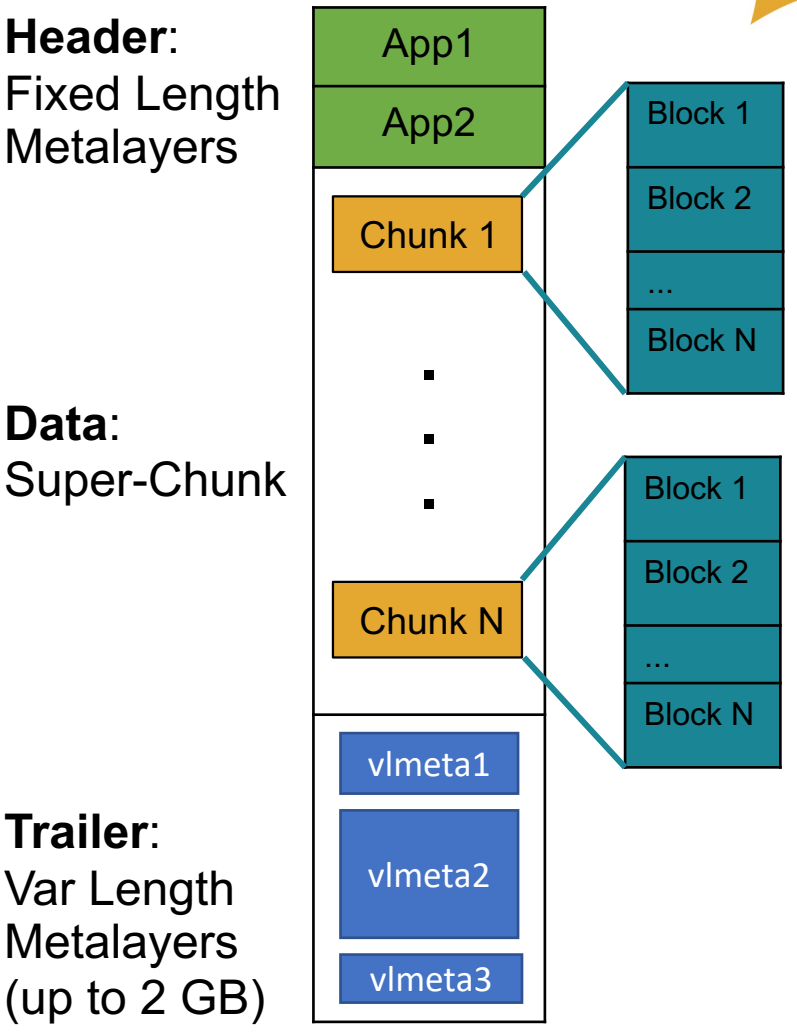
CEO ironArray / @ironArray

2023 European HDF User Group (HUG) plugins and data compression summit

September 19th 2023

What is Blosc2?

- Next generation of Blosc(1), a high performance compressor.
- Blosc2 adds 63-bit containers that expand over the existing 31-bit containers (chunks) in Blosc1.
- Supported in **hdf5plugin** (still experimental)






The Blosc Development Team

Marta Iborra
Aleix Alcacer
Francesc Alted
J. David Ibáñez
Ivan Vilata
Oscar Guiñón
Sergio Barrachina
Alberto Sabater




Agenda

A new **dynamic plugin** system for Blosc2

A new plugin for **HT JPEG2000**

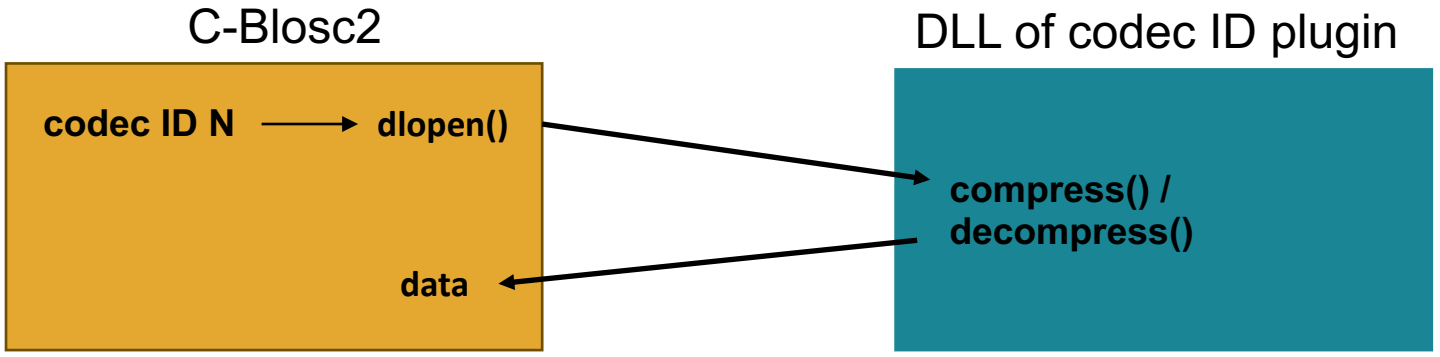
Support for **Blosc2 Ndim in HDF5**

Btune: AI tool for automatic selection of the **best codecs and filters**

A new dynamic plugin system for Blosc2

Loading plugins dynamically
How does it work?

Whenever C-Blosc2 receives a request for using **dynamic codec ID N**, it will **dynamically load** its **DLL library** using **dlopen()/LoadLibrary()** calls:



https://www.blosc.org/posts/dynamic-plugins/

Dynamic plugins as Python wheels

- Wheels are not only a great idea, but they are well implemented too:
  - Great backward compatibility with really old OS
  - Smooth user interface (via pip) for dealing with different platforms
  - Mostly compatible with conda environments

- We are actively using wheels for distributing binary versions of the C plugins (e.g. OpenHTJ2K; see later).

- Anybody can create their own plugins, and distribute them in binary form.

Pros and cons of dynamic plugins

Pros

- Very easy to install:

```
$ pip install blosc2-openhtj2k
```

- Do not bloat the C-Blosc2 library or other plugin managers (hdf5plugin, numcodecs...)
- Support for C, C++, Rust plugins. Only requisite is to expose a C API!

Cons

- Somewhat more work to create. But we are providing an example with detailed instructions:

  https://github.com/Blosc/blosc2_plugin_example#readme

A new plugin for High Throughput JPEG 2000

High Throughput JPEG 2000 (HTJ2K)

HTJ2K preserves almost all the rich feature set of JPEG 2000, except for quality scalability, offering an order of magnitude increase in throughput (i.e., vastly lower computational complexity) at the cost of a 5-10% reduction in coding efficiency.

Table 2: Throughput and objective compression performance, measured via PSNR, for unweighted encoding with the irreversible CDF 9/7 wavelet transform with 64x64 code-blocks.

	ENCODING (fps)		DECODING (fps)		PSNR (dB)	
bpp	J2K-1	HTJ2K	J2K-1	HTJ2K	J2K-1	HTJ2K
1	13.2	84 (6.4x)	31.2	117 (3.8x)	41.41 dB	40.92 dB
2	8.5	76 (8.9x)	16.7	112 (6.7x)	45.54 dB	44.83 dB
4	5.3	64 (12x)	8.6	91 (11x)	51.18 dB	50.10 dB

https://ds.jpeg.org/whitepapers/jpeg-htj2k-whitepaper.pdf

Introducing the Blosc2-OpenHTJ2K dynamic plugin

- It is using OpenHTJ2K, an open source HTJ2K implementation by Osamu Watanabe.
- **Release 0.1.2 is out! Please review.**
- Packed and distributed as a Python wheel:
  - `$ pip install blosc2-openhtj2k`

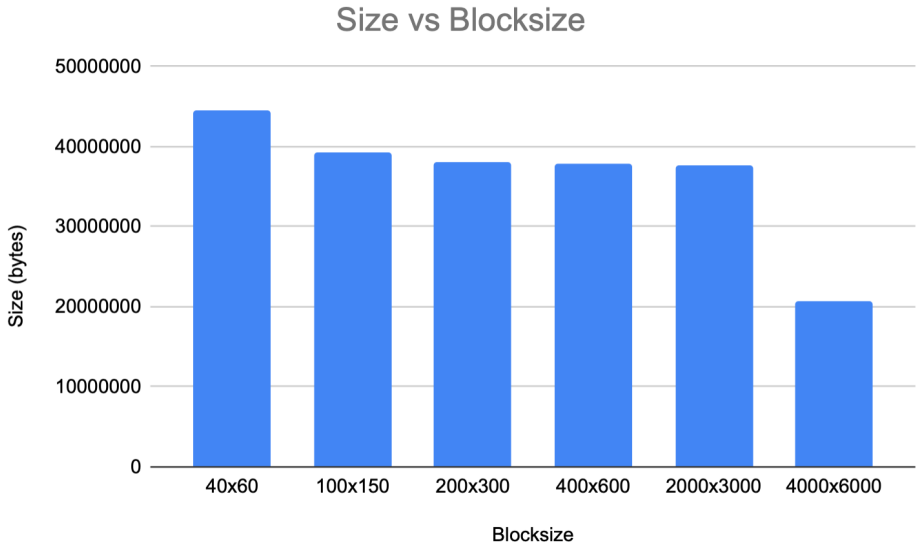
https://github.com/Blosc/blosc2_openhtj2k

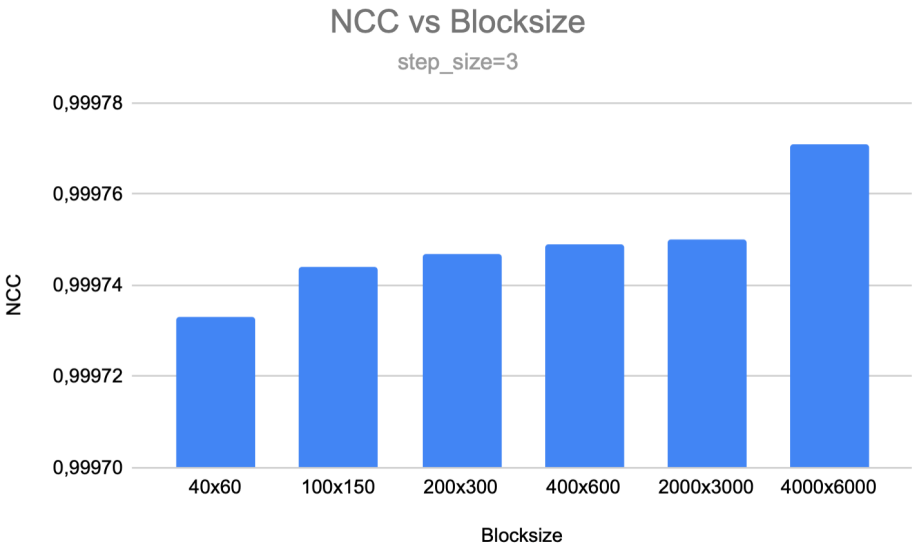
Lossless vs lossy compression with OpenHTJ2K

OpenHTJ2K lossless (634K)

OpenHTJ2K lossy (330K)

Effect of blocksize on size and quality





Better compression when blocksize equals to image size

Better quality when blocksize equals to image size

OpenHTJ2K works better when blocks are of the same size than images

Parameters for OpenHTJ2K

```
params_defaults = {
    'blkheight'             : 4,
    'is_max_precincts'      : True,
    'use_SOP'               : False,
    'use_EPH'               : False,
    'progression_order'     : 0,
    'number_of_layers'      : 1,
    'use_color_trafo'       : 1,
    'dwt_levels'            : 5,
    'codeblock_style'       : 0x040,
    'transformation'        : 1,
    <snip>
}
```

And then some more: https://github.com/osamu620/OpenHTJ2K#options

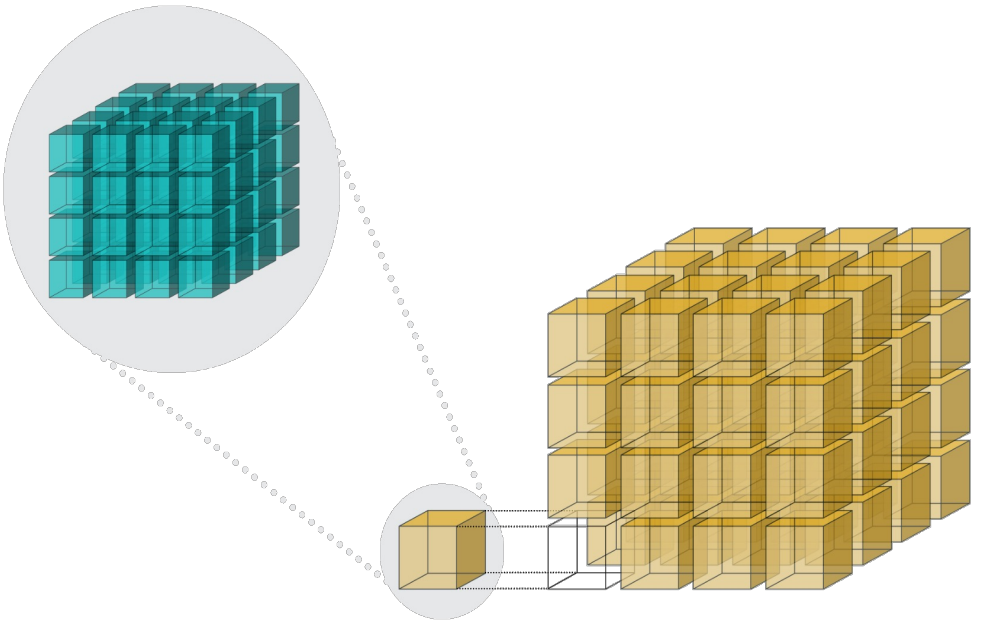
Work in progress

- **It cannot leverage Blosc2 multithreading yet** (OpenHTJ2K not threading-safe)
    - One can still use the internal one provided by OpenHTJ2K (still WIP)
- **Support for just x86_64**. ARM64 should be interesting, specially for Macs.
- Make the **Blosc2 plugin for HDF5, as well as hdf5plugin, aware of the parameters** needed for OpenHTJ2K. As there are many of them, we better be creative on how to pass them:
    - Good enough defaults?
    - Passing params via environment variables?

Support for Blosc2 NDim in PyTables / HDF5

C-Blosc2 NDim: Multidimensions for C

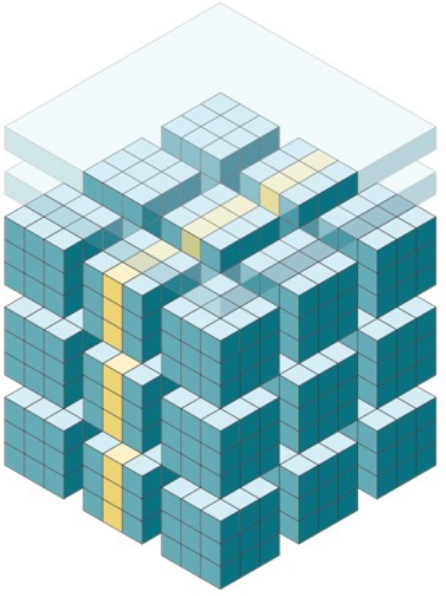
- Each NDim array is split in chunks
- Each chunk is split in blocks
- **Both partitions are multidimensional**
- Metalayer representing both **multidimensionality** and **data types**



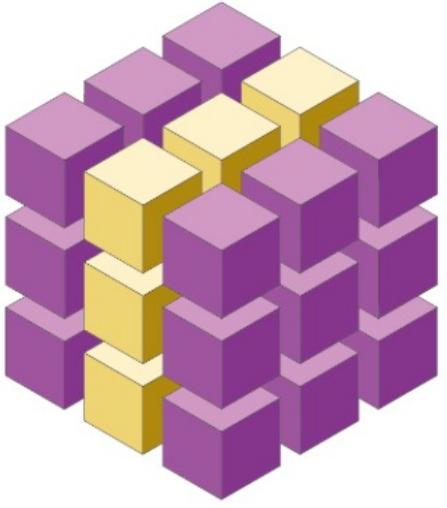
https://www.blosc.org/c-blosc2/reference/b2nd.html

Leveraging the second partition in Blosc2 NDim

Much more selective and hence, faster queries!



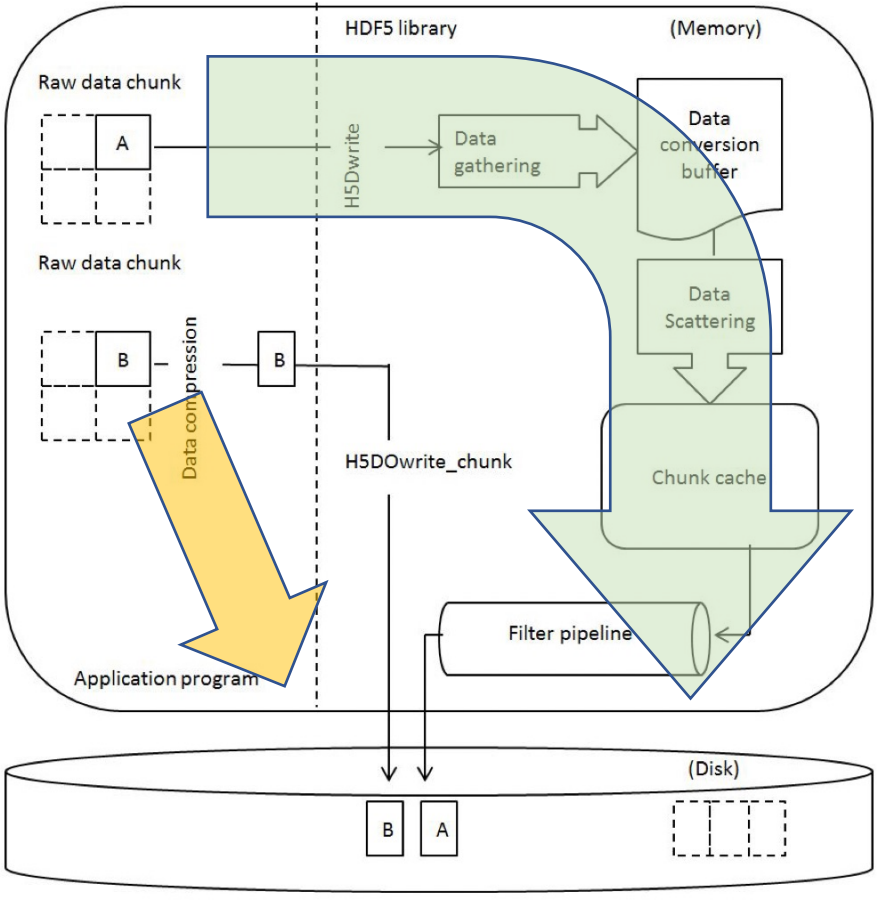
Blosc2 NDim



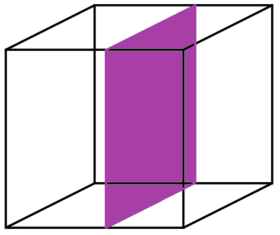
HDF5 / Zarr / others

Bypassing the HDF5 pipeline: Direct Chunking

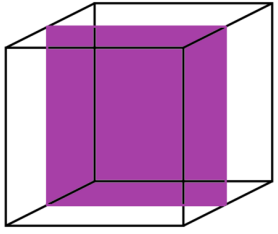
- HDF5 pipeline implementation is powerful but known to be slow
- PyTables has support for bypassing it via the H5Dwrite_chunk / H5Dread_chunk
- Unleash the full I/O parallel in Blosc2



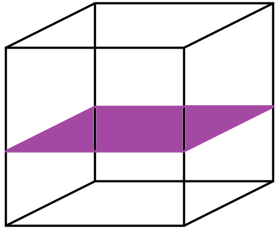
HDF5 pipeline vs direct chunking: Reading orthogonal slices



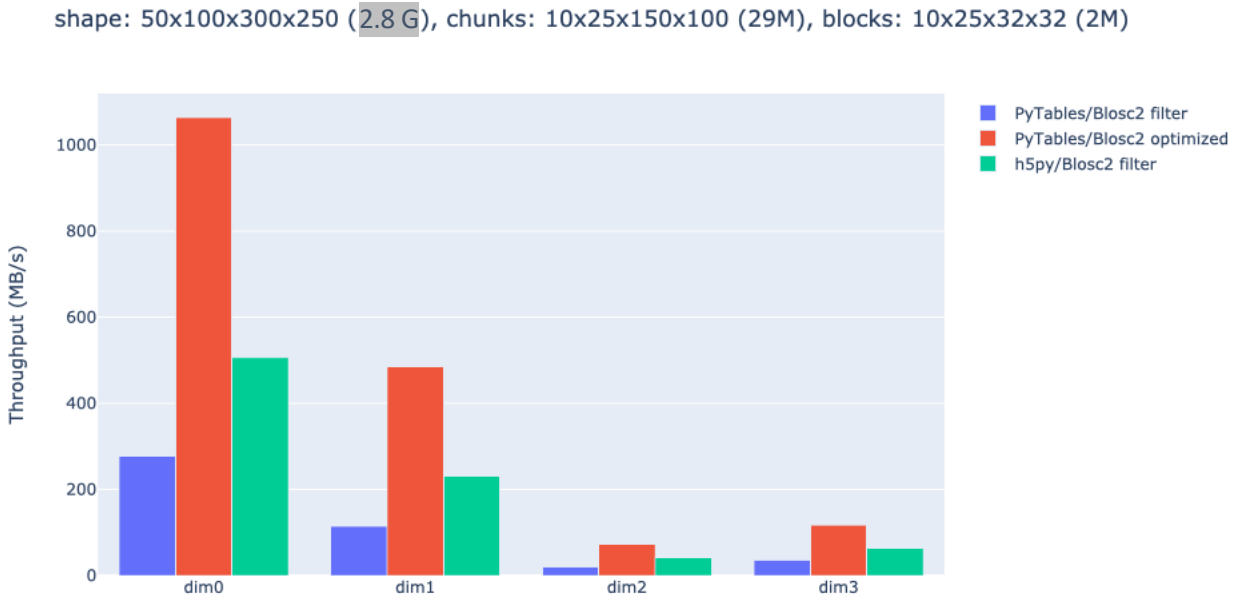
constant dim0



constant dim1



constant dim2



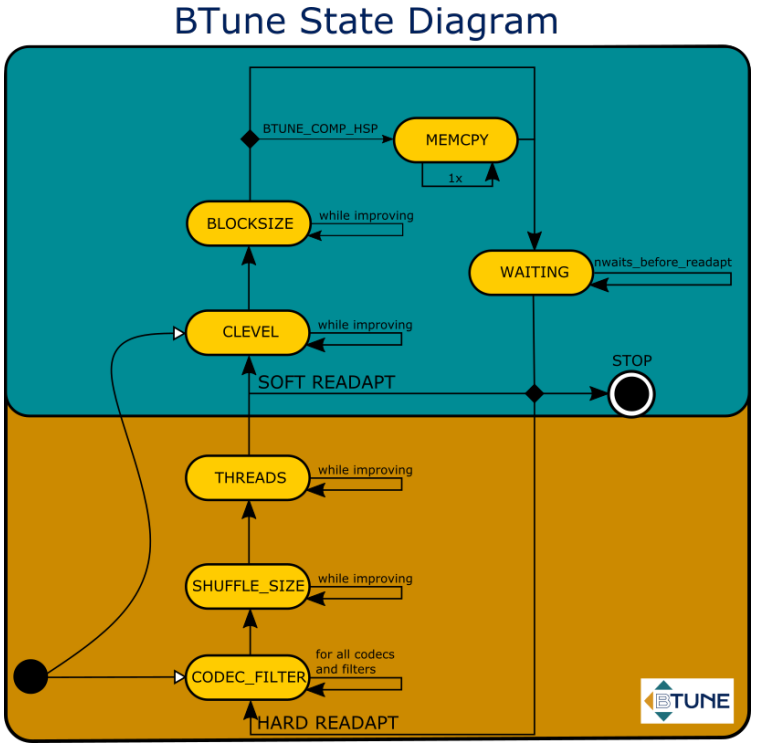
Faster slicing due to higher data selectivity in double partitioning

Btune: automatic selection of the best codecs / filters

Fine Tuning Performance with BTune

https://btune.blosc.org

- BTune can fine tune the different parameters of the underlying Blosc2 storage to perform as best as possible.
- Can be trained to find the best codec & filter with **deep learning**.
- **Btune ready for production!**



Different tiers of support for Btune

- **Genetic (Btune Free):** test different combinations by brute force
- **Trained (Btune Models):** Blosc team train datasets for you:
  - More accurate predictions for all chunks (specially first ones!)
- **Fully managed (Btune Studio):** you can train on your own

Tutorial on Thursday!

https://btune.blosc.org

Conclusion

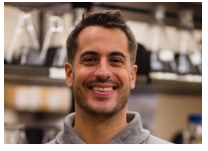
Blosc2 is making rapid progress

The Blosc2 development team has recently implemented:

- **Dynamic plugin** support
- **Plugin for High Throughput JPEG 2000**
- Implemented **native support for Blosc2 NDim in HDF5**, bypassing the HDF5 pipeline
- **Btune,** an AI tool for automatically selecting the best Blosc2 parameters, is **in production mode**

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

Thanks to donors & contracts!



Jeff Hammerbacher

Without them, we could not have possibly put Blosc2 into production status: Blosc2 2.0.0 came out in June 2021; now at 2.10.3.

Thank you! Questions?








blosc@blosc.org

We make compression better