

HDF5 & Blosc2

A Proposal For Working As A Team

Francesc Altèd / [@FrancescAltèd](#)

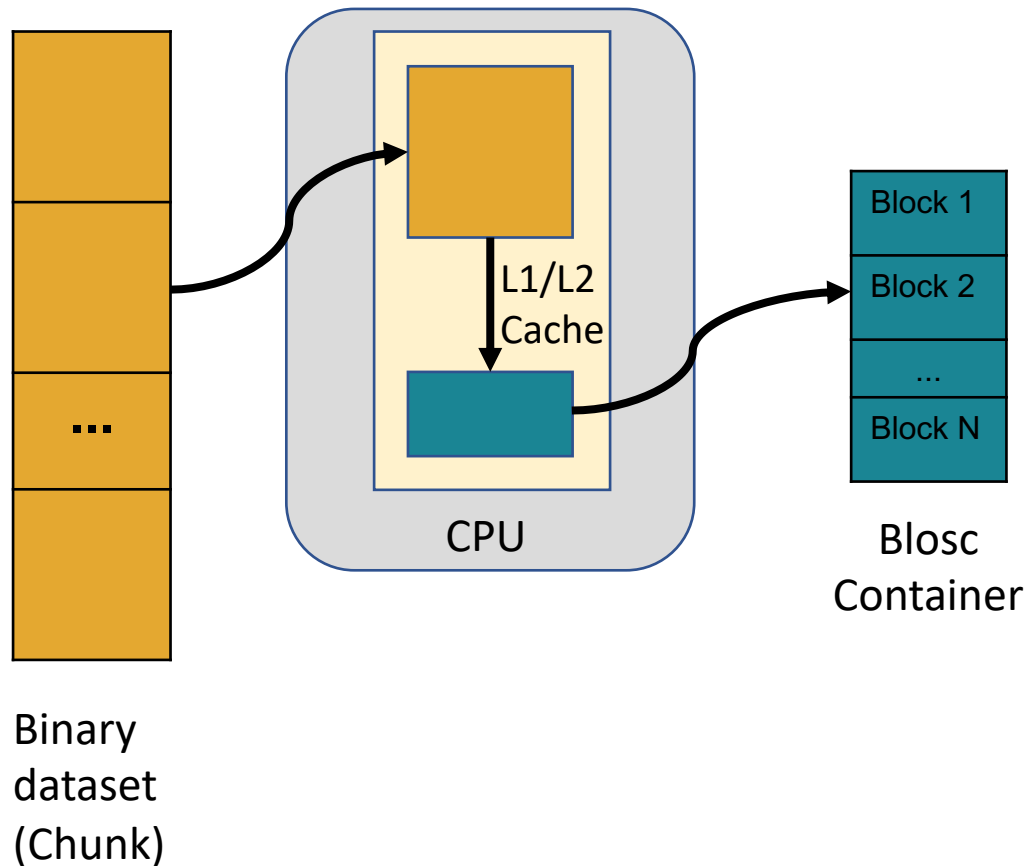
The Blosc Development Team / [@Blosc2](#)

CEO  **iron**Array / [@ironArray](#)

LEAPS Innov WP7 (data reduction and compression) meeting
May 4th 2022

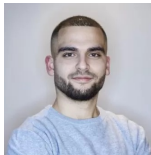
What is Blosc?

- ✓ Sending data from CPU to memory (and back) faster than *memcpy()*.
- ✓ Split in blocks for better cache use: divide and conquer.
- ✓ It can use different filters (e.g. shuffle, bitsuffle) and codecs (e.g. LZ4, Zlib, Zstd, BloscLZ).



A solid orange square located in the top-left corner of the slide.

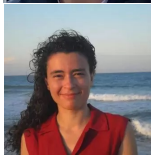
The Blosc Development Team



Aleix Alcacer



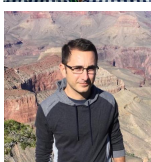
Oscar Guiñón



Marta Iborra



Alberto Sabater



Nathan Moinvaziri



Francesc Alted (BDFL)

Origins of Blosc

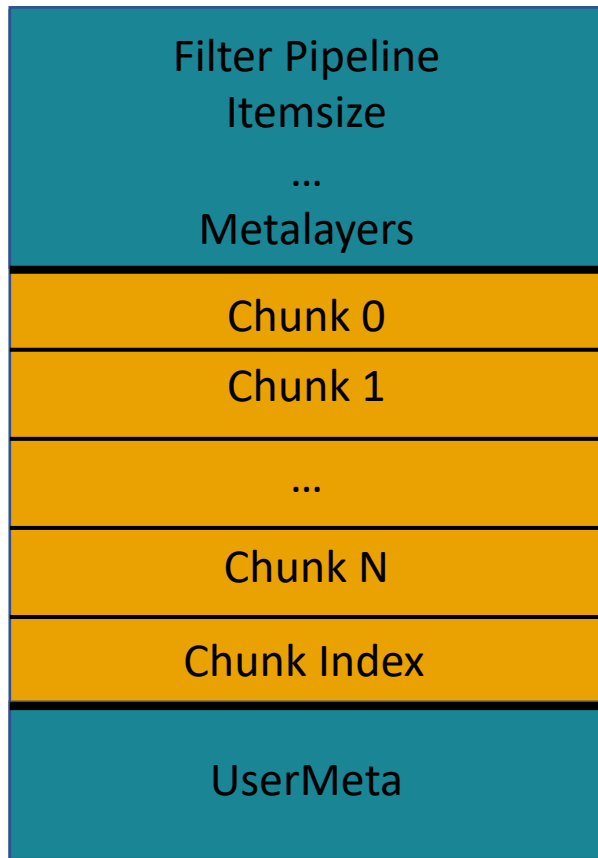


- **2009:** it was very clear that compression was slowing down storage in PyTables/HDF5 a lot. Work began.
- **2010:** Blosc 1.0 was ready for production. Innovations:
 - Shuffle filter was optimized for SSE2 (*much* faster)
 - Multithreaded operation
- **2013:** Blosc gained multi-codec (LZ4, Snappy and Zlib where included)
- **2015:** hdf5-blosc plugin for HDF5 was released (hdf5plugin took over!)
- **2021:** Blosc2 appeared with **lots** of new features.

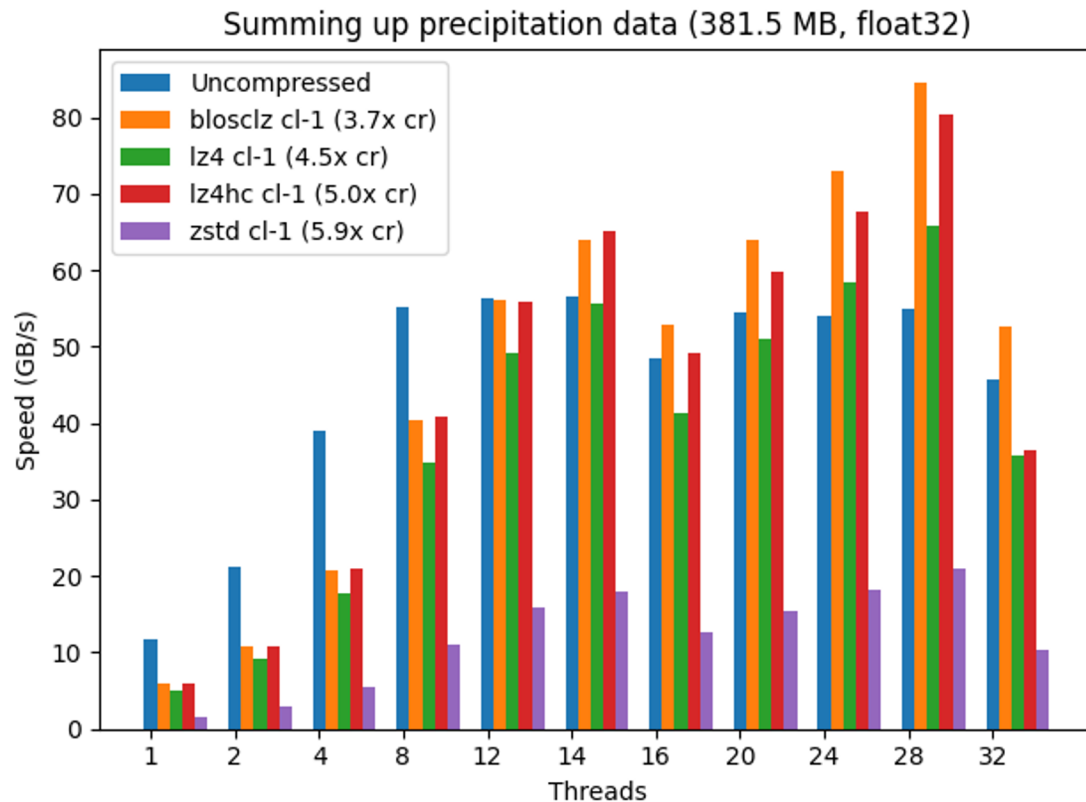
What is Blosc2?

- ✓ Blosc2 is the next generation of Blosc1.
- ✓ New 63-bit containers (frames) that expand over the existing 31-bit containers (chunks) in Blosc1.
- ✓ Metalayers for adding info for applications.
- ✓ Area for adding metadata for users (variable length).

Blosc2 Frame



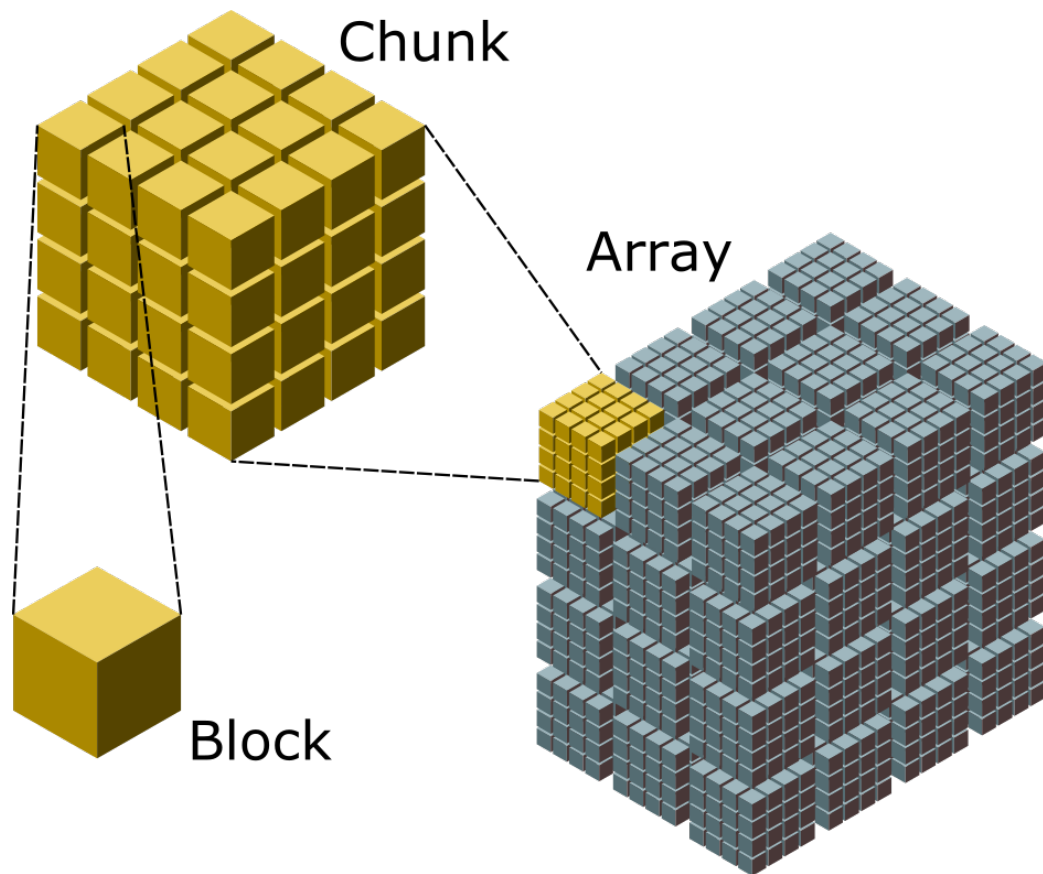
Example of Decompression Speed



<https://www.blosc.org/posts/breaking-memory-walls/>

Caterva: Blosc2 Goes Multidimensional

- ✓ Metalayer representing multidimensionality
- ✓ Each Caterva array is split in chunks
- ✓ Each chunk is split in blocks
- ✓ All the partitions are multidimensional!



HDF5: Multidimensions and Chunking

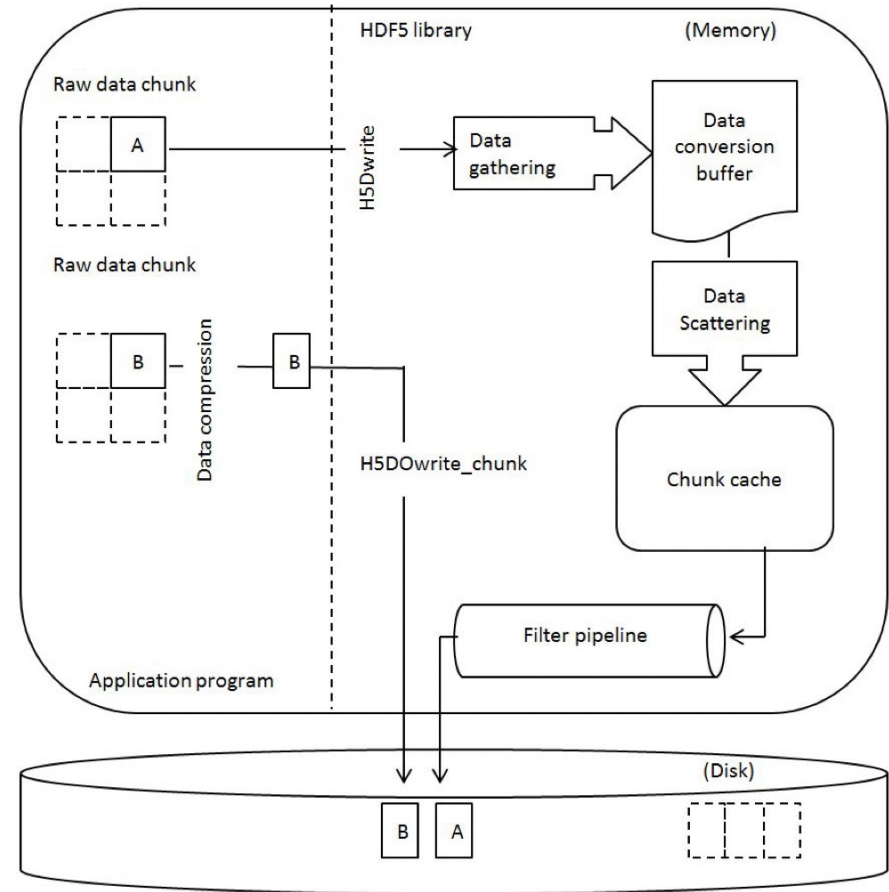


- Data can be stored in hypercubes, making retrieval very convenient.



- But there is a price to pay for this flexibility: HDF5 is known to be **slow** when retrieving (hyperslabs of) data.

- Allow the application to handle the chunk I/O and bypass the powerful (but slow!) chunk handling machinery in HDF5.
- The result is that data can be handled up to about 10x faster, provided efficient pre and post processing.



Proposal 1: Use Blosc2 Inside Direct Chunk



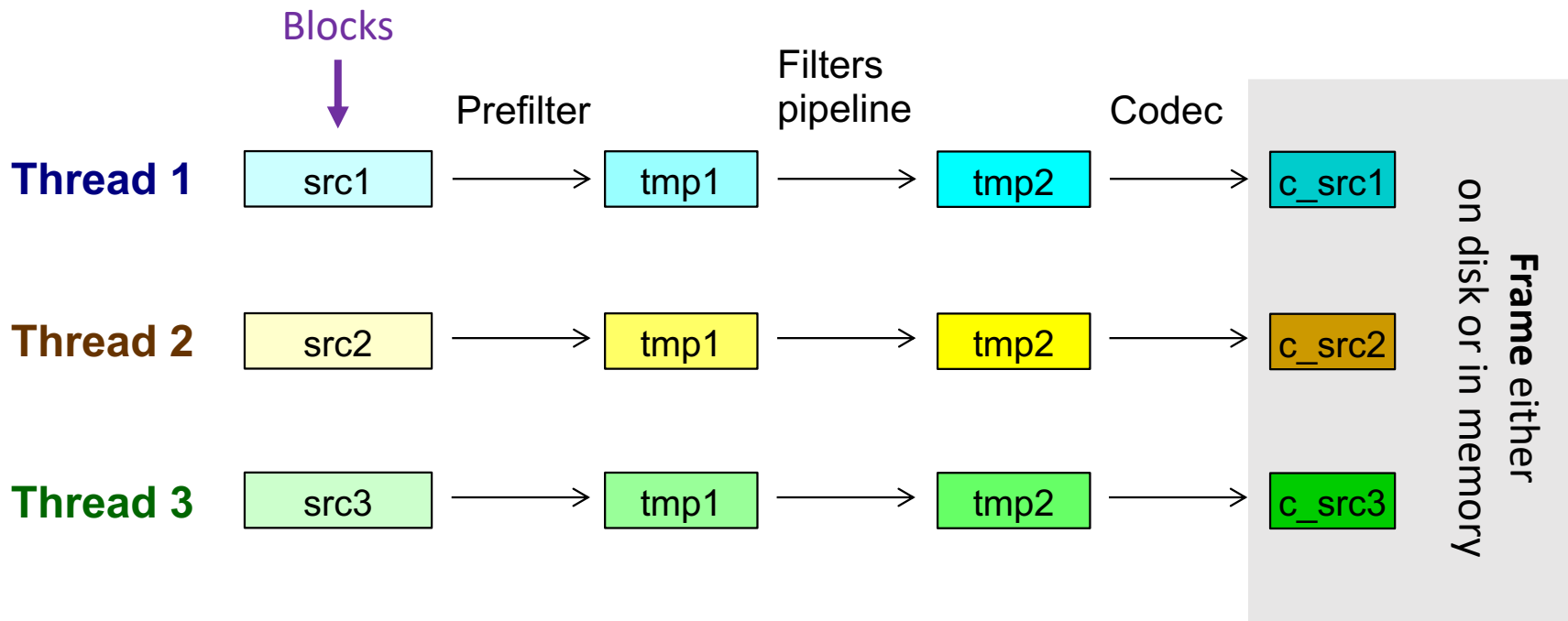
Make Blosc2 to pre- and post-process chunk data for:

- Handle double partitioning
- Multithreaded compression/decompression
- Parallel I/O (important to achieve higher IOPS in SSDs)
- When second partition fits well in L1/L2 CPU caches => speed!
- In addition, if the Caterva layer is used => multidim partitions (this can be useful for ZFP, SZ or JPEG codecs)

Blosc2 Advantages

Blosc2: Fine Tuned Cache Usage

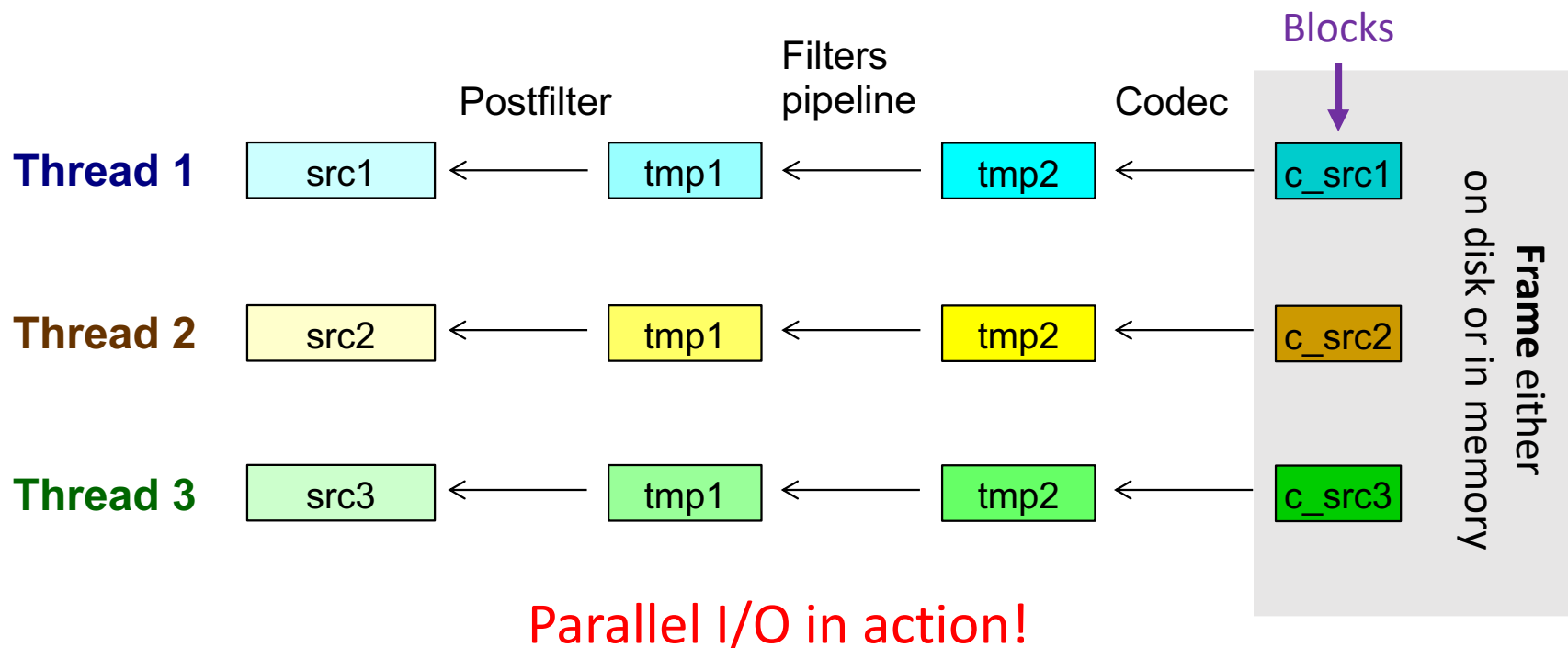
Compression: chunks are split in blocks for CPU cache sake



Buffers are reused **inside** CPU caches -> speed!

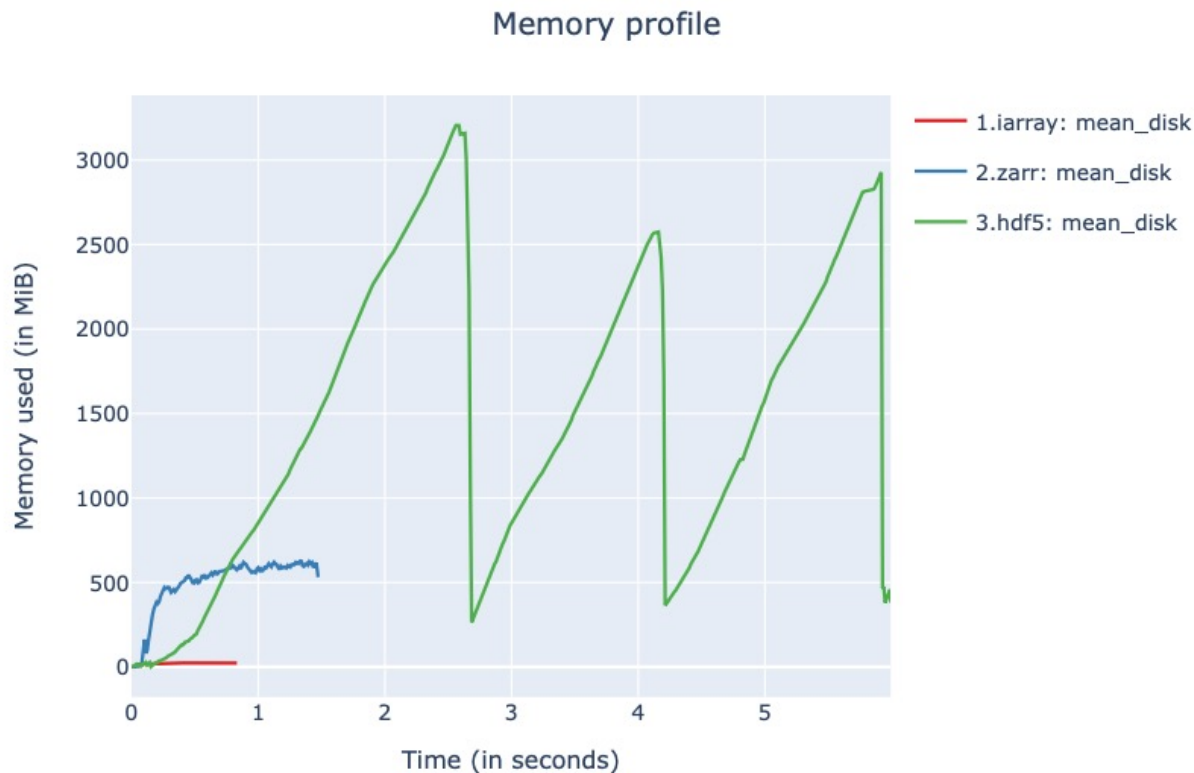
Blosc2: Leveraging I/O Parallelism

Decompression: blocks are read in parallel from storage



Blosc2: Parallelism and Efficiency

- In the plot: 3 compressed arrays are decompressed, operated, and the result is compressed again.
- ironArray is using Blosc2.
- When handled correctly, parallelism can buy not only speed, but also less memory resources!



Mean of 3 arrays of 3 GB each (on disk)

Adaptability: Plugins in Local Registry

Filters registry

BLOSC_SHUFFLE	1
BLOSC_BITSHUFFLE	2
BLOSC_DELTA	3
...	
BLOSC_NDCELL	32
BLOSC_NDMEAN	33
...	
urfilter1	160
urfilter2	161
...	

- Blosc official registered filters
- User local filters

User defined filter:

```
int urfilter2(
    blosc2_filter *filter) {
    ...
}
```

To register locally:

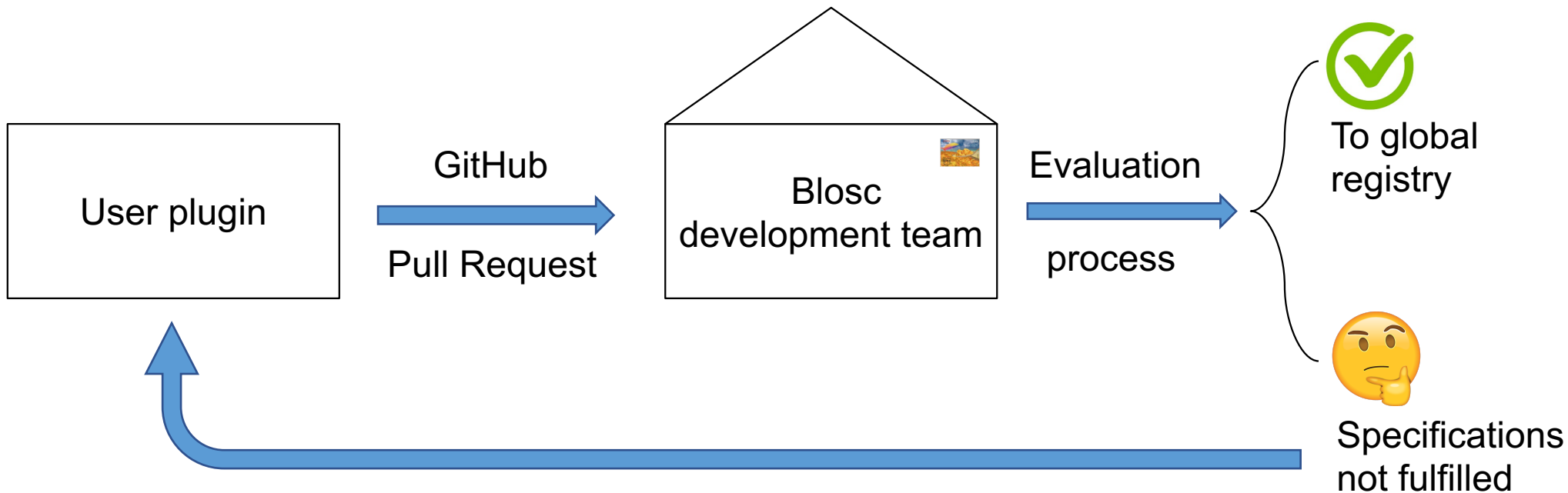
```
blosc2_register_filter(
    urfilter2) →
```

Can be used now:

```
→ cparams.filters[4] = 161;
```

(Similar functionality to the plugin interface in HDF5)

Registering Global Plugins in Blosc2



Specs: <https://github.com/Blosc/c-blosc2/blob/main/plugins/README.md>



Deploying Plugins in Central Registry

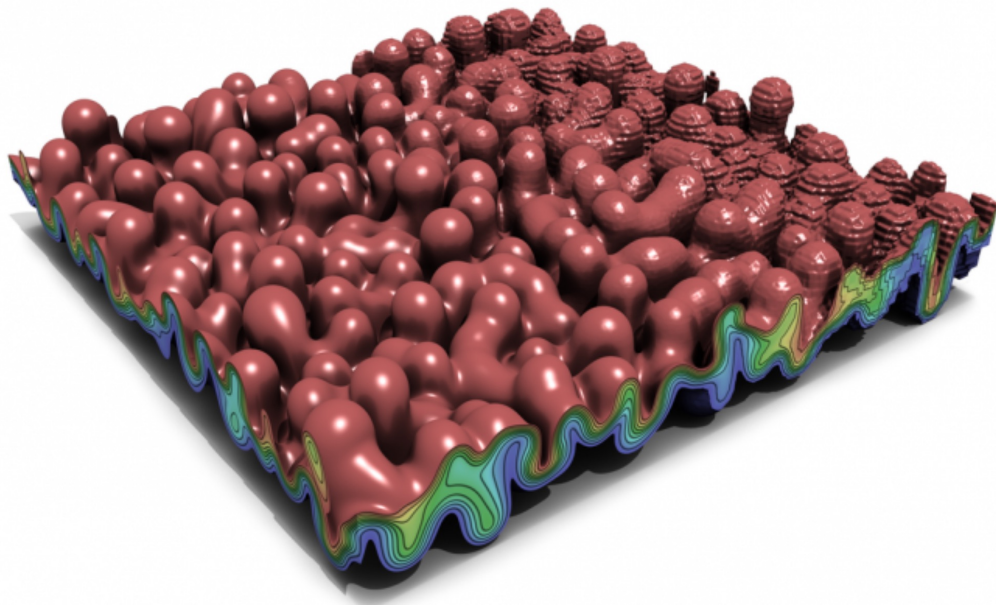
Central registered plugins are **included** and **distributed** within the Blosc2 library. Can be installed using the Python wheels:

```
bash-3.2$ pip install blosc2 --no-cache-dir
Collecting blosc2
  Downloading blosc2-0.2.0-cp39-cp39-macosx_10_9_x86_64.whl (4.0 MB)
    | ████████████████████ | 4.0 MB 3.4 MB/s
Installing collected packages: blosc2
Successfully installed blosc2-0.2.0
```

Very convenient in making your filter/codec accessible for everybody!

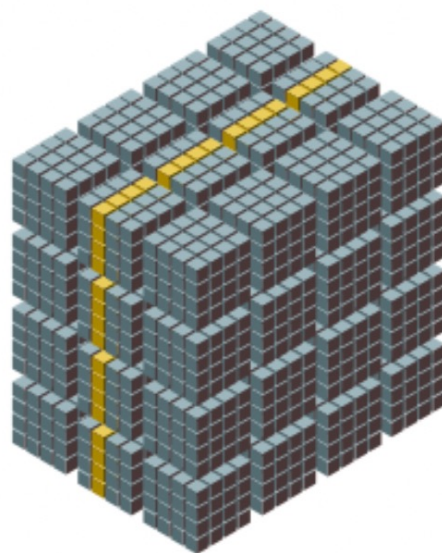
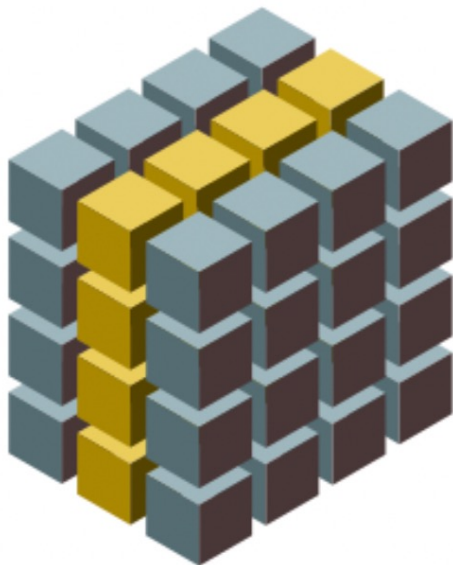
Benefits of Adding the Caterva Layer

- Get **improved compression ratio** because data is packed in a way that can show higher spatial locality.
- Also, get **improved hyperslab query speed**, i.e. some blocks can be masked out so as to not read them.



ZFP: supported as a registered plugin

Masked & paralel I/O in multidim datasets

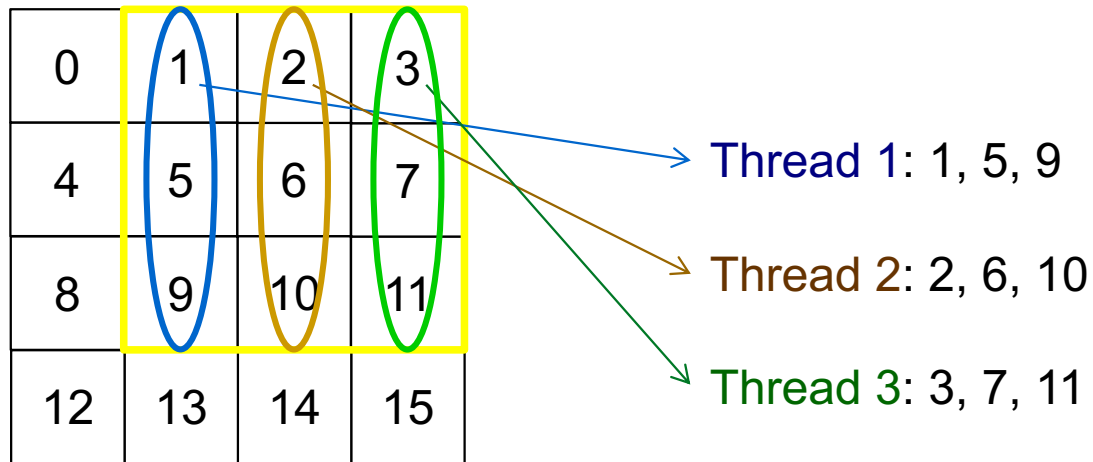


Much more selective and faster queries!

[Caterva](https://github.com/Blosc/caterva) (<https://github.com/Blosc/caterva>) and [ironArray](https://ironarray.io) (<https://ironarray.io>)

Block Masks and Parallel I/O

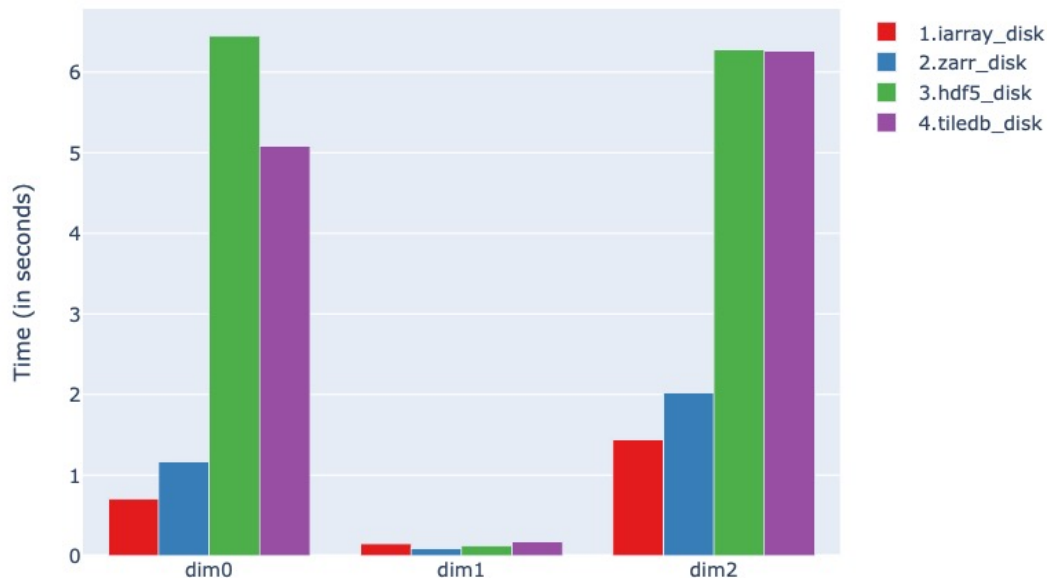
Block maskout	F	T	T	T	F	T	T	T	F	T	T	T	F	T	T	T
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



Specially effective when retrieving slices of multidim datasets.

Masked & Parallel I/O in Multidim Datasets

Slicing Performance on disk (with an optimized dimension)



Better performance in general
(except for dimensions where retrieving a chunk is already optimal)

https://ironarray.io/docs/html/tutorials/03.Slicing_Datasets_and_Creating_Views.html

Proposal 2: Help in Determining Optimal Compression Pipelines

We are offering a service for adapting to the user data, and determining:

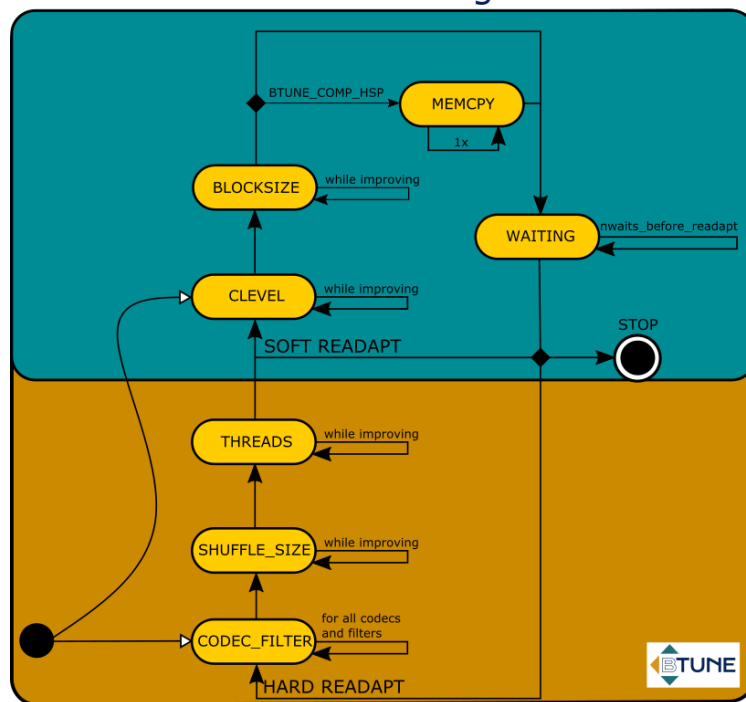
- Set of most useful codecs to be used
- Set of most useful filters to be used

We produce **specific versions** of **BTune**, a machine learning tool for selecting the best pipeline candidate on a **chunk by chunk** basis, that adapts to the needs of the user.

Fine tuning performance with BTune

- BTune can fine tune the different parameters of the underlying Blosc2 storage to perform as best as possible.
- Active during the compression pipeline.
Automatically learns the best parameters on the go.

BTune State Diagram



Conclusion

Blosc2 Helps Saving Resources



Blosc2 orchestrates a **rich set of codecs and filters** for:

- **CPU parallelization** via multithreading
- Reuse and sharing internal buffers for **optimal memory consumption**
- **Parallel I/O**
- **Selective** hyperslab **selections**

In addition, **new filter & codec can be registered.**

The result is a highly efficient tool for
compressing and accessing your data your way

Proposal Summary



1. Use Blosc2 in combination with HDF5 direct chunking mechanism for efficient compression and parallel I/O.
2. Help in determining optimal compression pipelines by adapting to user data and using machine learning techniques.

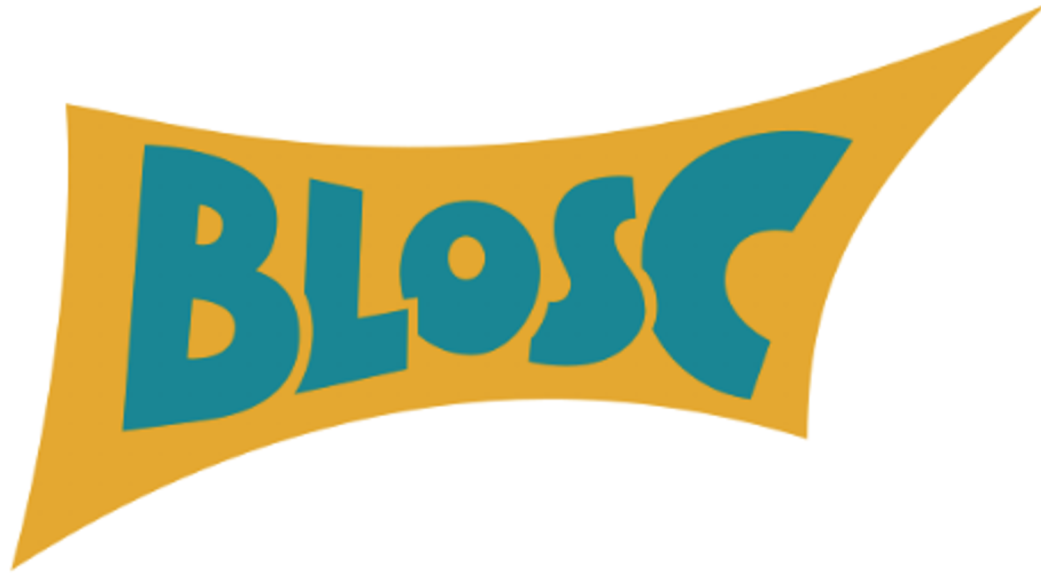
Thanks to donors & contractors!



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 2.1.0.

Enjoy data!



<https://blosc.org/>