Exploring The Milky Way

(Or Any Other NDim Dataset)

Francesc Alted / @FrancescAlted

The Blosc Development Team / @Blosc2

SciPy 2023 Conference, Austin, TX, USA
July 12th 2023

Agenda

The Gaia Dataset

Blosc2 NDim and NDArray Objects

Automatic Compression Tuning with Btune

Exploring the Milky Way with Blosc2

Conclusions

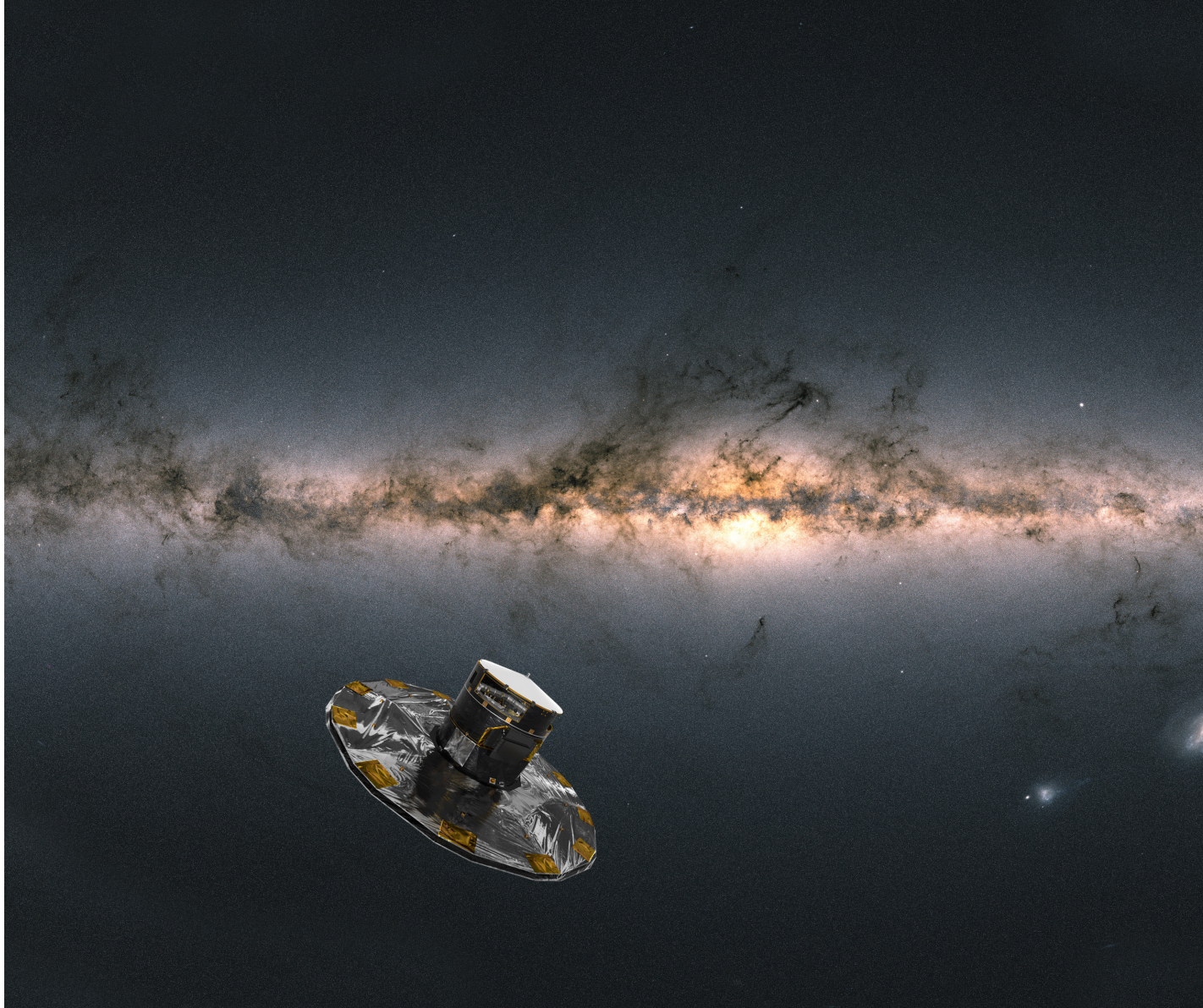
Disclaimer

I am not attached to the Gaia collaboration at all.

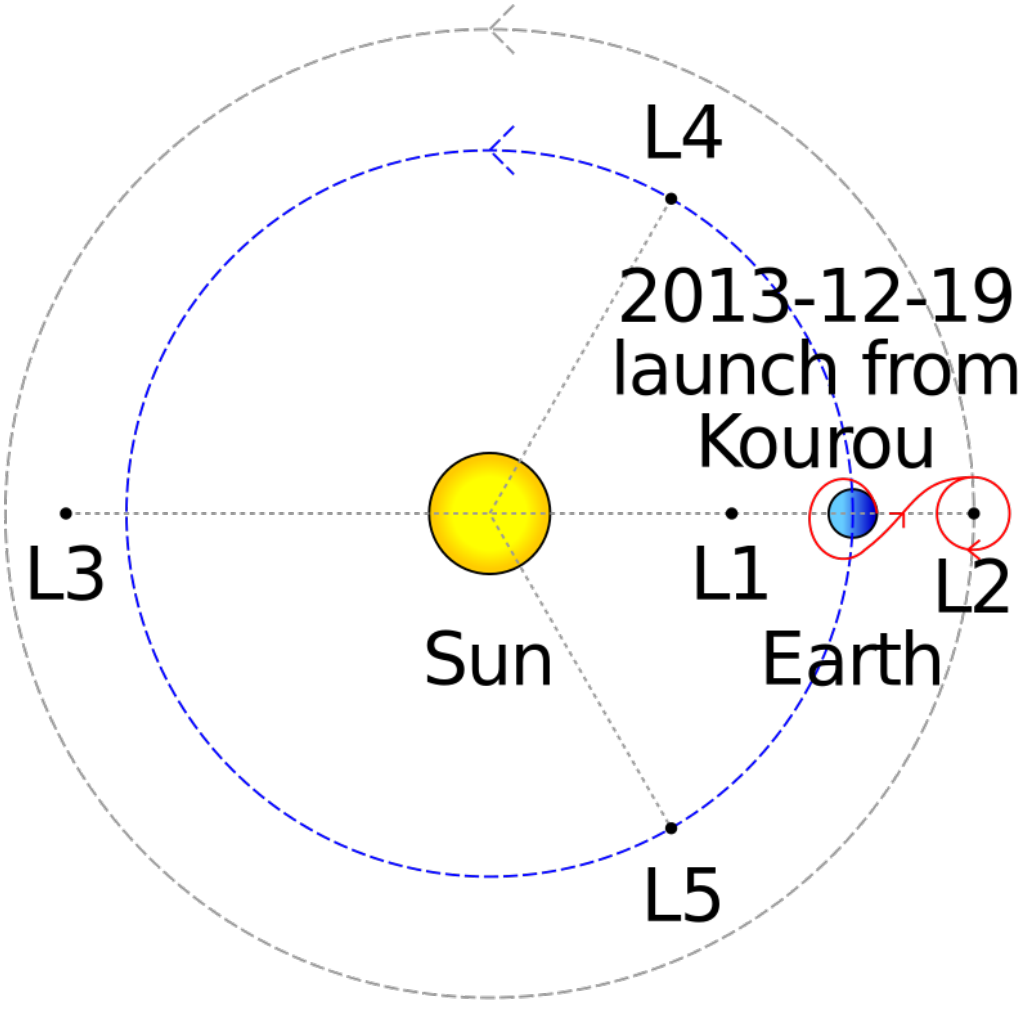
Any statement said here about scientific facts might be plain wrong!

Documentary Seen at Planetarium of My Home Town (Castelló de la Plana)

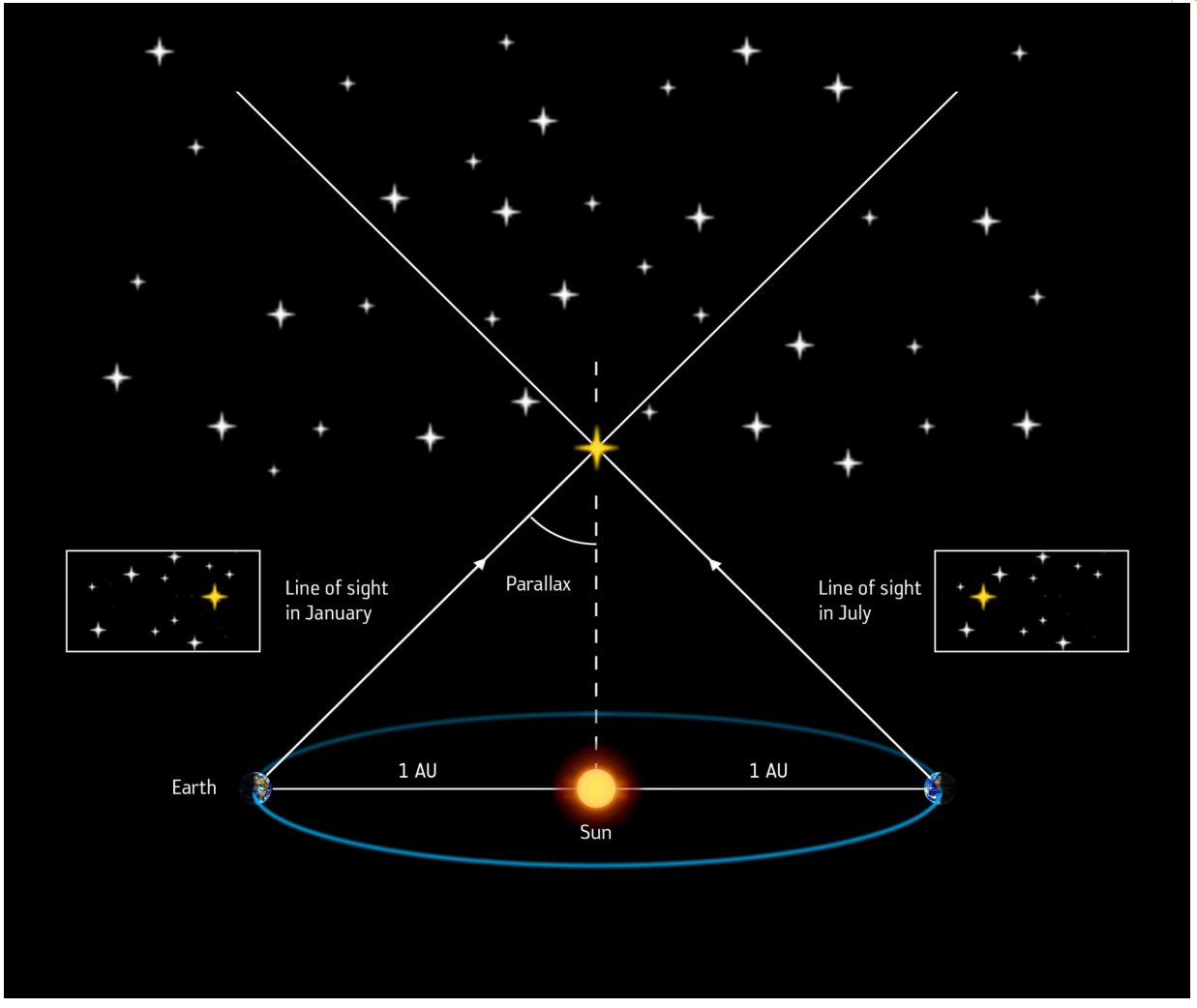
Gaia mission: measure the position of stars in the Milky Way (and a lot of other info)



Gaia telescope orbits around L2 (recently James Webb joined too)

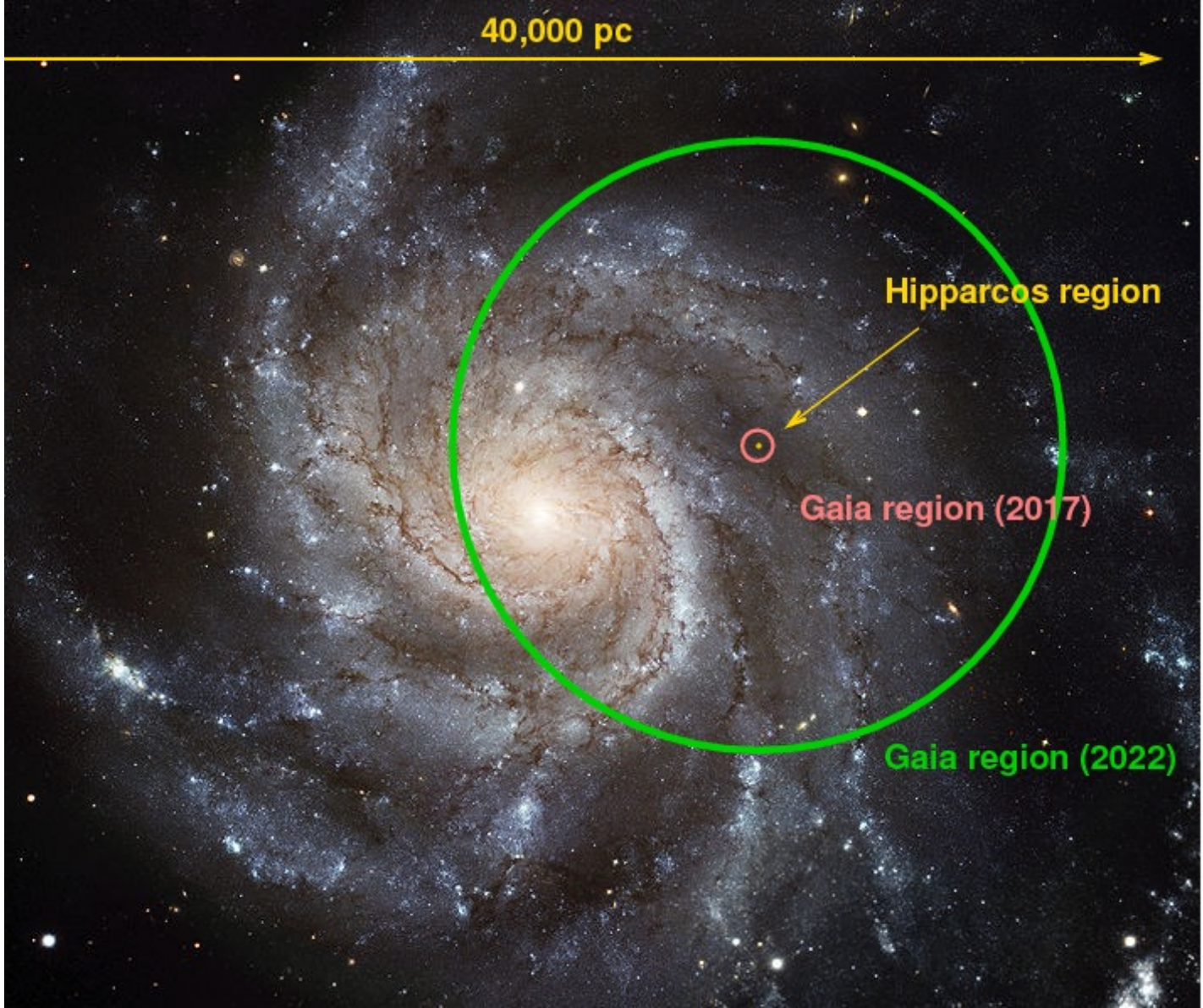


Gaia uses Parallax to measure distance to nearby stars

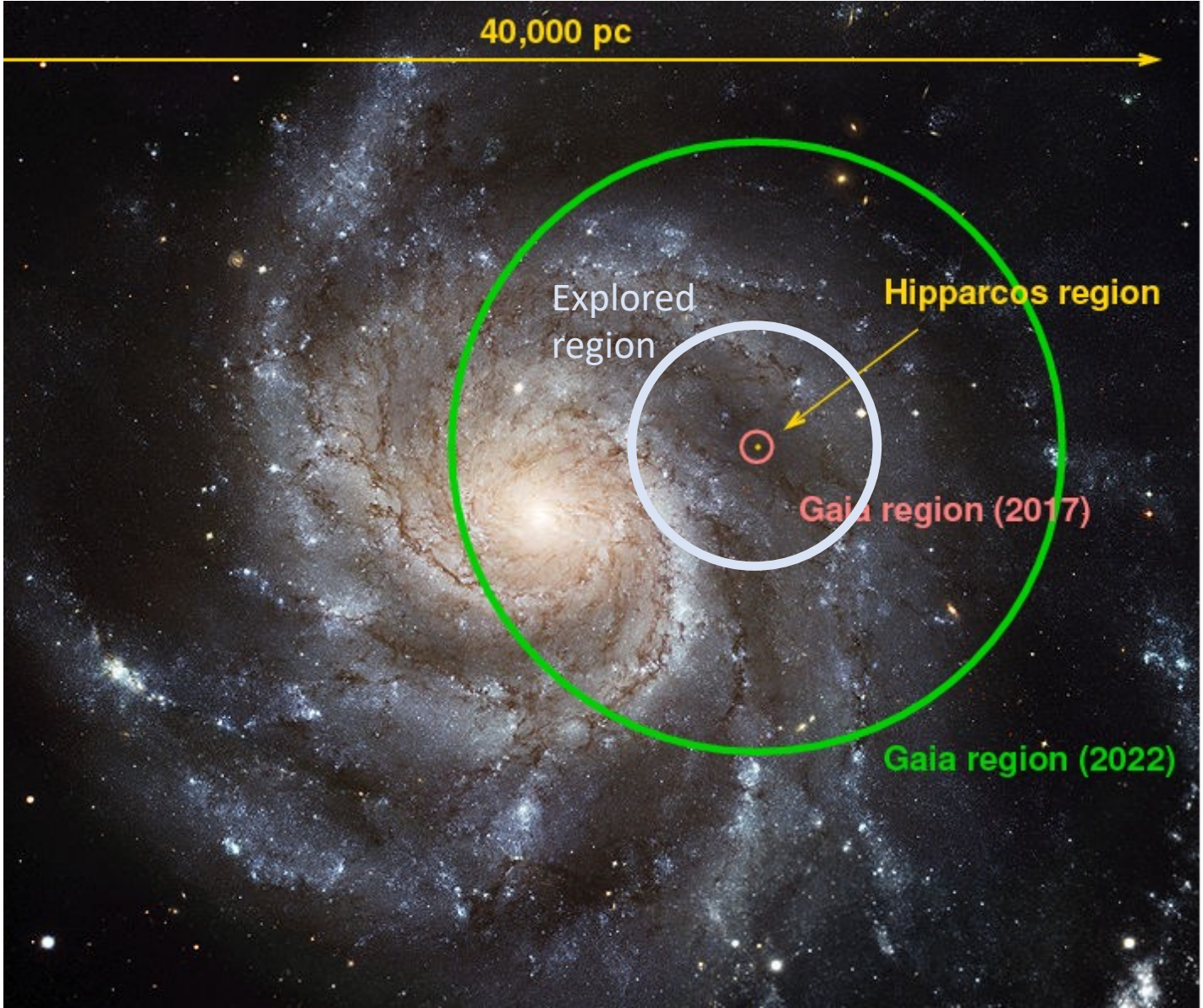


The Gaia Dataset

Gaia can observe larger regions as more observations cumulated



We will restrict ourselves to a region which is 10,000 light years (3,000 parsec) away from Gaia



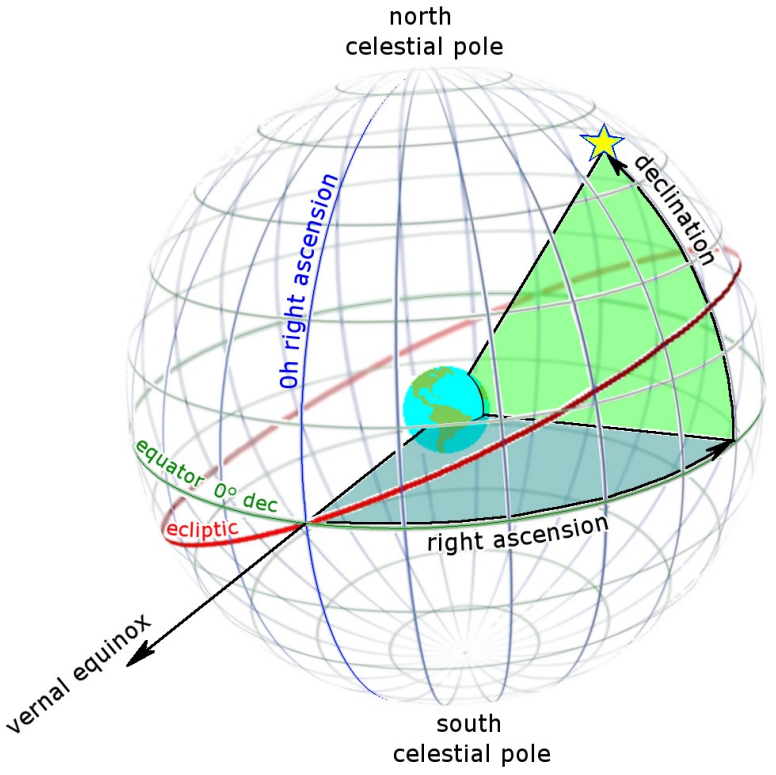
Gaia Main Source Catalog

- A set of public CSV files with many star parameters
- We are interested mainly in:
  - **ra** : Right ascension (double, Angle[deg])
  - **dec** : Declination (double, Angle[deg])
  - **distance_gspphot**: Distance from GSP-Phot Aeneas (float, Length & Distance[parsecs])
- From these, and some elemental spherical geometry, we can read and filter the stars in a radius of 10,000 light years.

https://gea.esac.esa.int/archive/documentation/GDR3/Gaia_archive/chap_datamodel/sec_dm_main_source_catalogue/ssec_dm_gaia_source.html

Spherical to Cartesian Coordinates

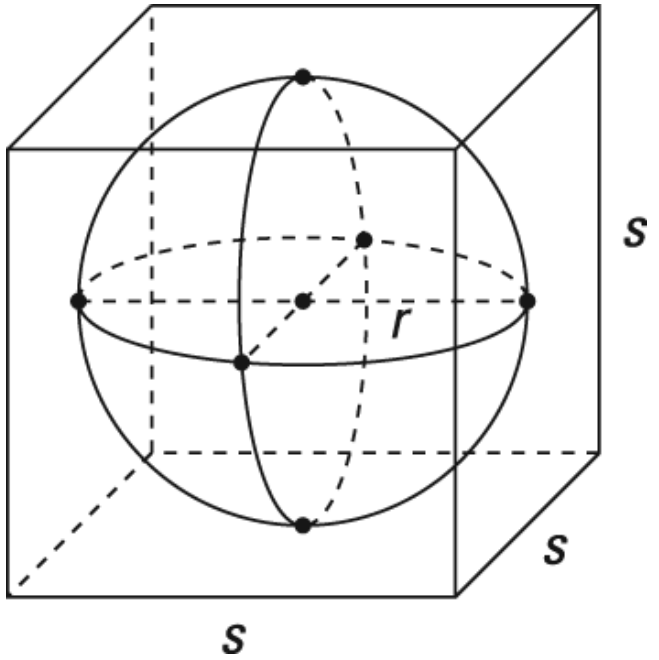
$$x = \rho \sin\theta \cos\varphi$$
$$y = \rho \sin\theta \sin\varphi$$
$$z = \rho \cos\theta$$



Easy to convert.
(But beware, angles must be in radians,
whereas Gaia raw data provides degrees)

The Exploration Cube

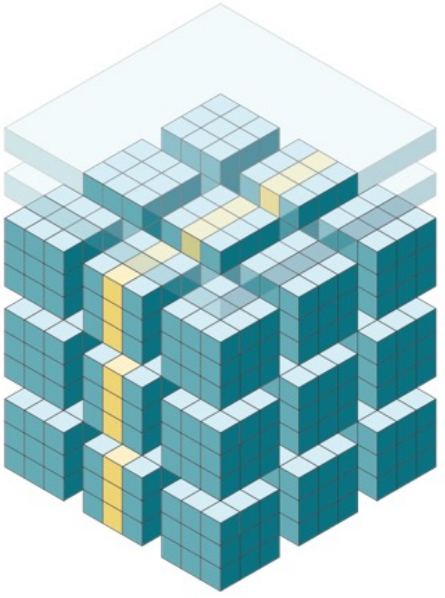
- Radius of the inscribed sphere:
  r = 10,000 light years
- Length of the cube side:
  S = 20,000 light years
- Every cell in the cube is 1 cubic light year
- 8 trillion cells (7.3 TB!)



How to explore a multi-TB dataset on a laptop with 8 GB of RAM and 256 GB (with just 50 GB free!) of SSD?

We Need Compression (and an effective one)

- The **number of stars** in the sphere of radius 10,000 light years is **around 0.5 billion**.
- **Sparsity is 1 in 10,000** cells (very high).
- The solution must handle sparse data **effectively**
- If the final goal is **real-time exploration**, it has to support **fast multidimensional slicing**

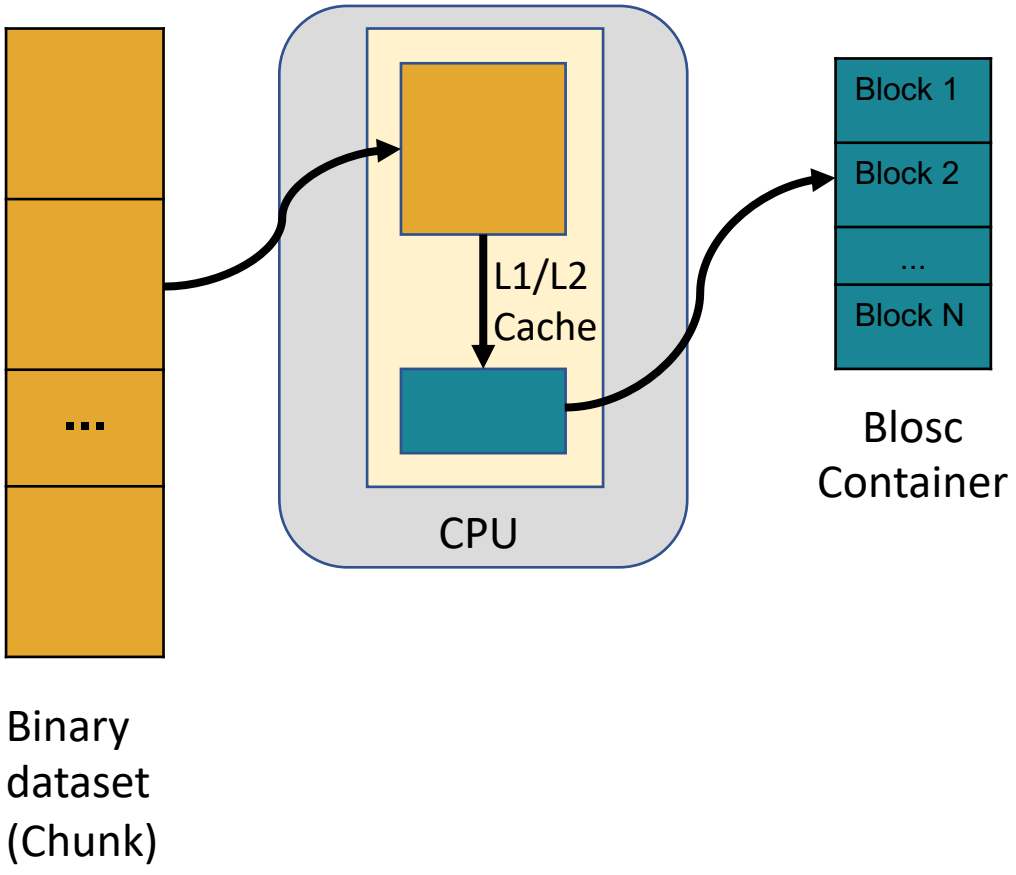


Enter Blosc2 NDim

A highly effective library (C and Python) for handling multdimensional, and potentially sparse datasets

What is Blosc?

- 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)
- Optimized for binary data



Where is Blosc used?

Blosc is used in many places in the PyData ecosystem:

- HDF5 / h5py (via hdf5plugin)

- HDF5 / PyTables (native)

- Zarr (via numcodecs)

- ironArray (Blosc2)

Lots of terrabytes compressed (and decompressed) on a daily basis!

Blosc (Francesc Alted) Winner of Google's Open Source Peer Bonus in 2017

"To recognize and celebrate external contributors to the open source ecosystem Google depends on."

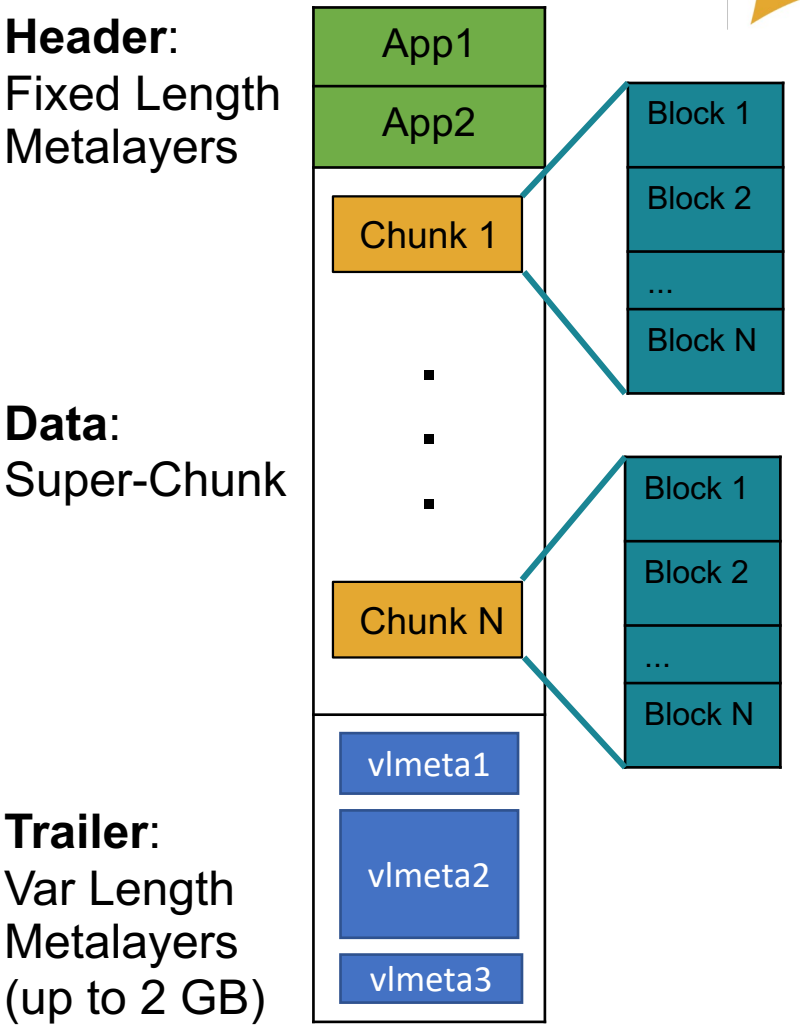
Some of the projects that won the award the same year:

- SQLite (Dan Kennedy, Joe Mistachkin, Richard Hipp)
- NumPy (Sebastian Berg)
- Ffmpeg (Michael Niedermayer)
- Flask (Armin Ronacher)

Google Open Source

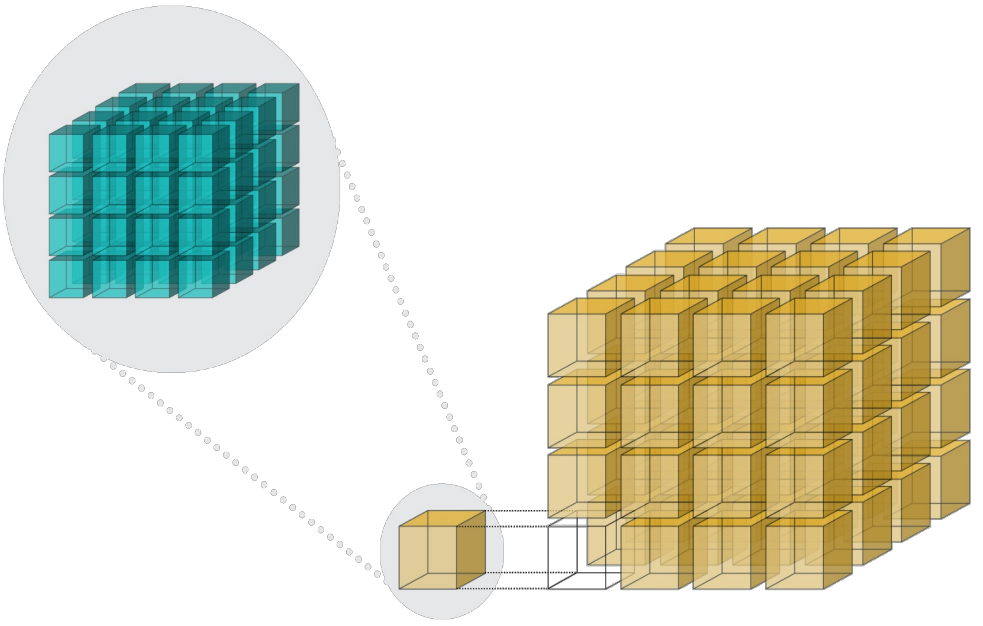
What is Blosc2?

- ✓ Next generation of Blosc
- ✓ 63-bit containers
- ✓ Enhanced support for sparse data
- ✓ Fully multidimensional double partitioning
- ✓ Metalayers for adding info for apps and users



C-Blosc2 NDim: Multidimensions for C

- ✓ Each NDim array is split in chunks
- ✓ Each chunk is split in blocks
- ✓ All the partitions are multidimensional
- ✓ AFAIK, no other library implements this



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

NDArray: Blosc2 NDim for Python

```python
import blosc2

a = blosc2.full((4, 4), fill_value=9)
a.resize((5, 7))
a[3:5, 2:7] = 8
print(a[:])
```

Output:

```
[[9 9 9 9 0 0 0]
 [9 9 9 9 0 0 0]
 [9 9 9 9 0 0 0]
 [9 9 8 8 8 8 8]
 [0 0 8 8 8 8 8]]
```

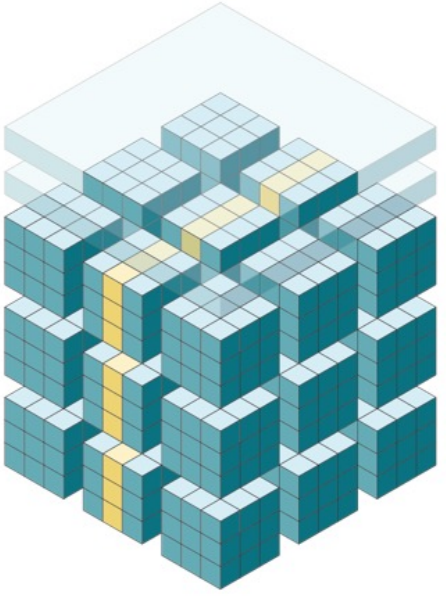
Features:

- Create arrays in memory or on disk
- Flexible resize (including shrinking)
- Efficient conversion from/to NumPy
- Mimic NumPy API

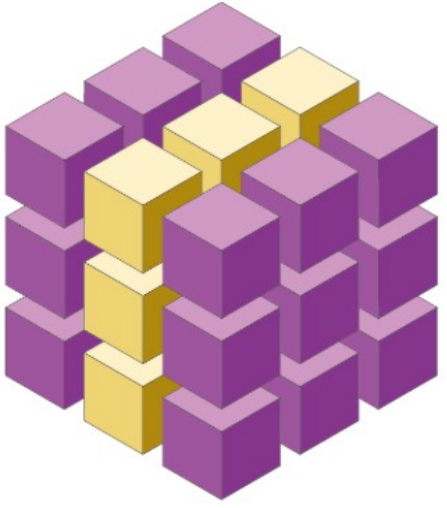
https://www.blosc.org/python-blosc2/reference/ndarray_api.html

Leveraging the second partition in Blosc2 NDim

Much more selective and faster queries!

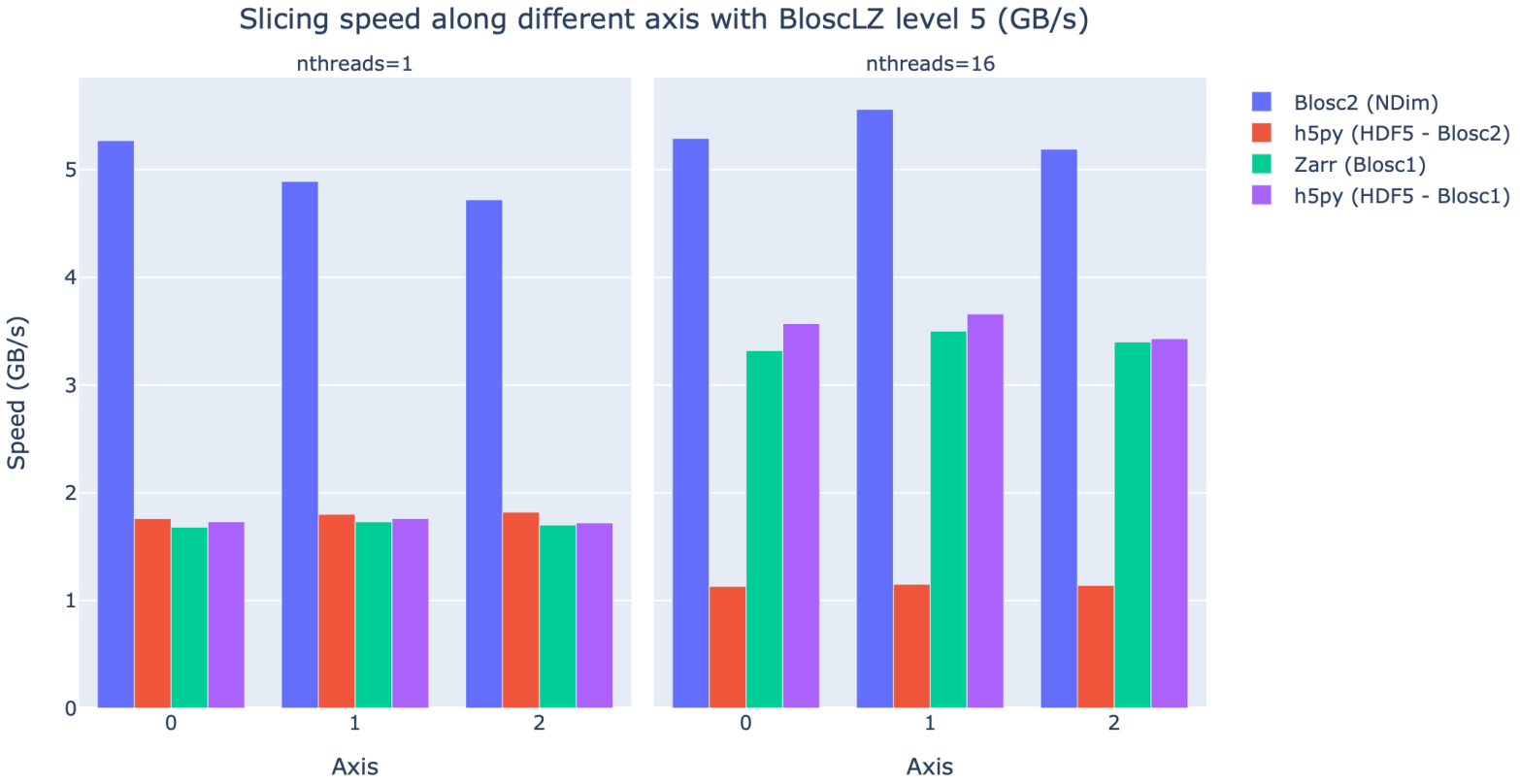


Blosc2 NDim



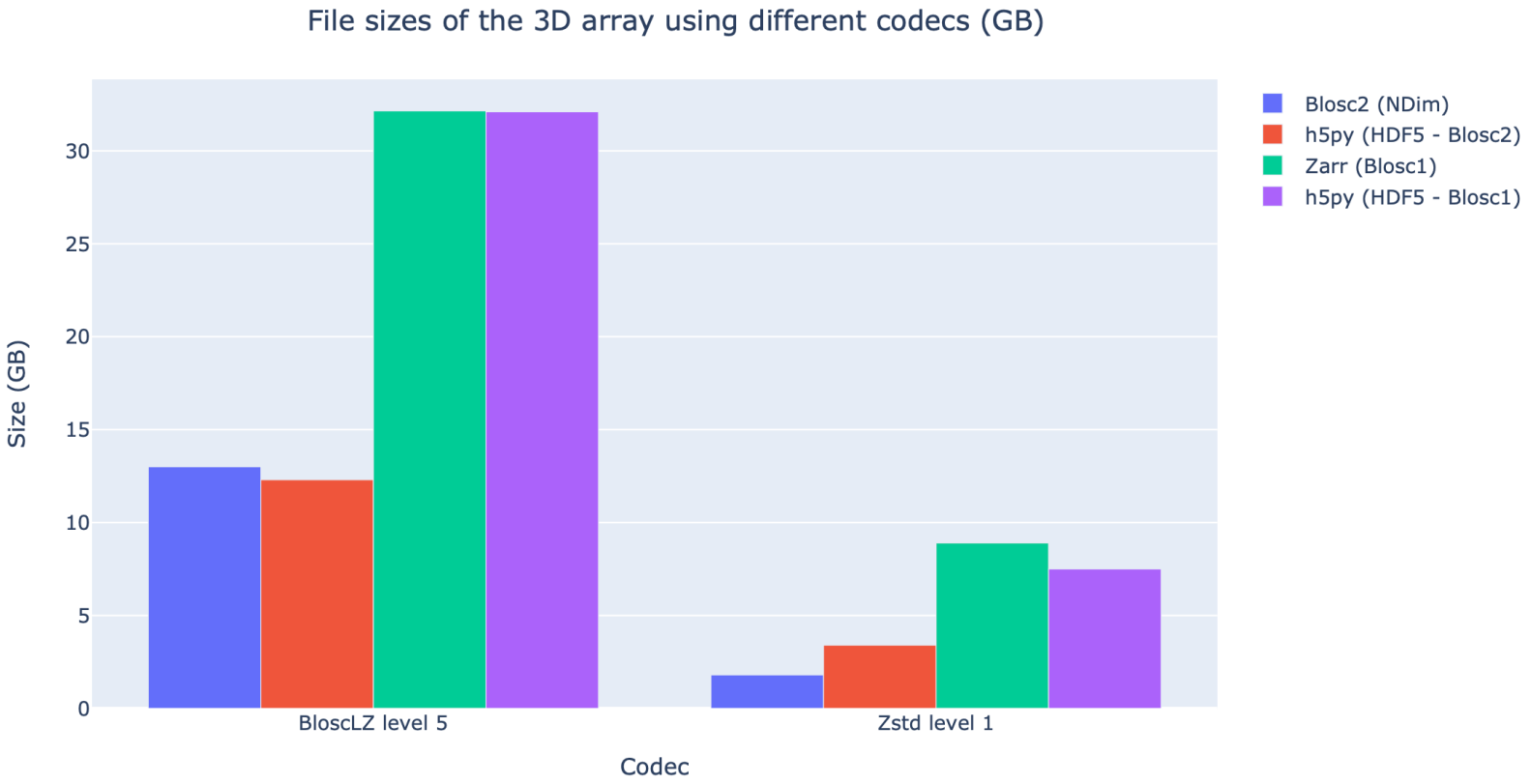
HDF5 / Zarr / others

Blosc2 NDim partial read performance



- Better sparse support for Blosc2 makes it faster
- Higher data selectivity in double partitioning

Blosc2 Ndim File Sizes



- Better sparse support for Blosc2 produces smaller files
- Blosc2 + Zstd packs the entire 3D Gaia array in less than 2 GB (4000x !)




The Blosc Development Team

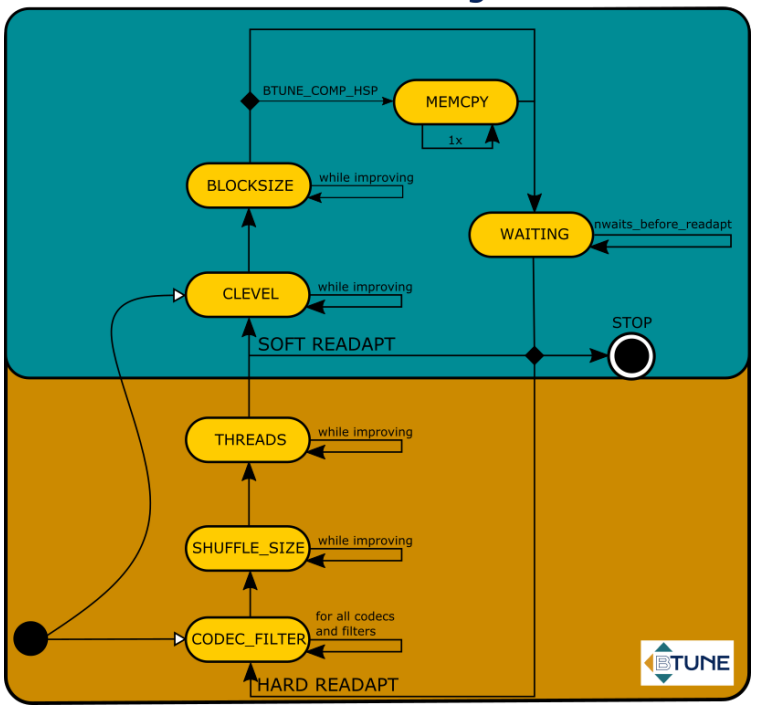
Marta Iborra
Aleix Alcacer
Francesc Alted
J. David Ibáñez
Ivan Vilata
Oscar Guiñón
Sergio Barrachina
Alberto Sabater




BTUNE

Making compression better

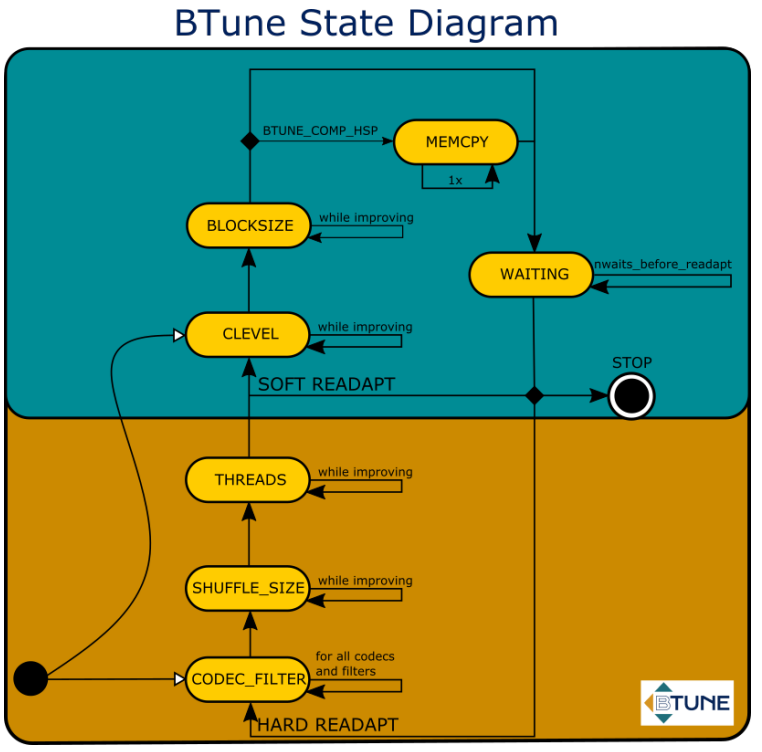
BTune State Diagram



Fine Tuning Compression Performance

- 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.



https://btune.blosc.org

Btune Operation Modes

- **Btune Free**: Use the dynamic Btune plugin directly.

- **Btune Models (AI)**: The Blosc Development Team helps you find a Neural Net Model adapted to your datasets for faster operation.

- **Btune Studio**: Use the training package locally to generate your own models for your datasets by yourself.

Installing & Using the Btune Plugin

We provide with binary wheels:

```
$ pip install blosc2-btune
```

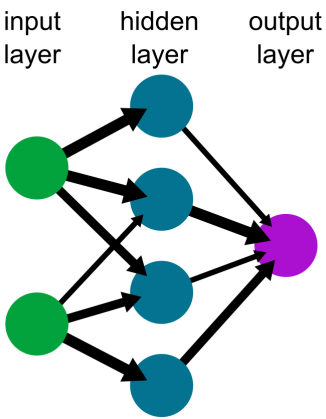
Using it is easy:

```
$ BTUNE_TRADEOFF=0.5 BTUNE_TRACE=1 python
examples/schunk.py
```

Btune Models

- Btune is trained for your datasets and can infer, in real time, the right combination of codec and filter that suits the requirements: favor speed, favor compression ratio, or a trade-off.

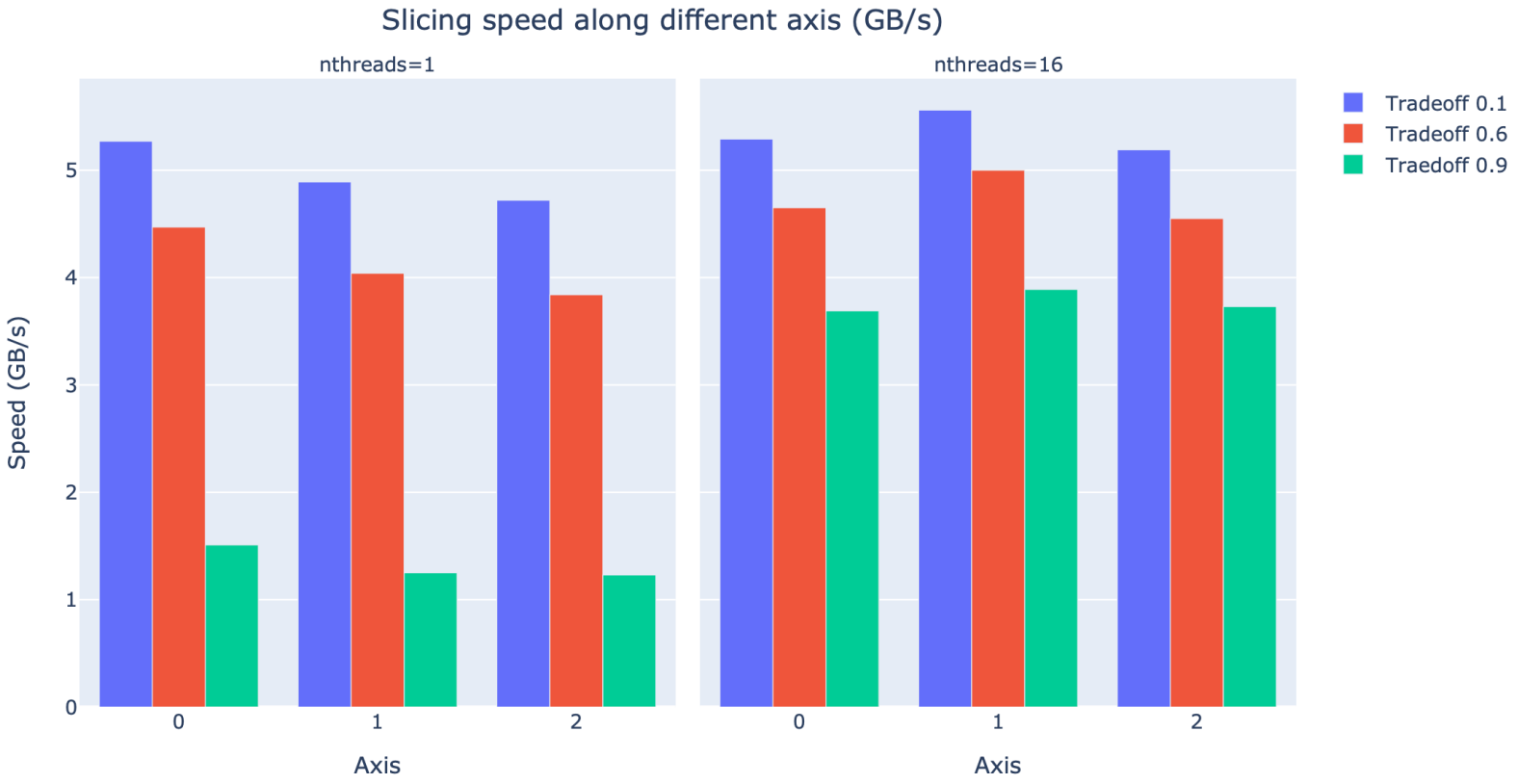
Neural Network Model



Predictions for Gaia dataset (decomp speed)

Tradeoff	Most predicted	Cratio	Cspeed	Dspeed
0.0	blosclz-nofilter-5	786.51	106.86	91.04
0.1	blosclz-nofilter-5	786.51	106.86	91.04
0.2	blosclz-nofilter-5	786.51	106.86	91.04
0.3	blosclz-nofilter-5	786.51	106.86	91.04
0.4	blosclz-nofilter-5	786.51	106.86	91.04
0.5	blosclz-nofilter-5	786.51	106.86	91.04
0.6	zstd-nofilter-9	8959.6	8.79	59.13
0.7	zstd-nofilter-9	8959.6	8.79	59.13
0.8	zstd-nofilter-9	8959.6	8.79	59.13
0.9	zstd-bitshuffle-9	10789.6	3.41	12.78
1.0	zstd-bitshuffle-9	10789.6	3.41	12.78

Btune Models



- Performance for different tradeoffs for decompressing
- Single threading is fine for tradeoffs favoring speed

Testimonials

"Blosc2 and Btune are fantastic tools that allow us to efficiently compress and load large volumes of data for the development of AI algorithms for clinical applications. In particular, the new NDArray structure became immensely useful when dealing with large spectral video sequences."

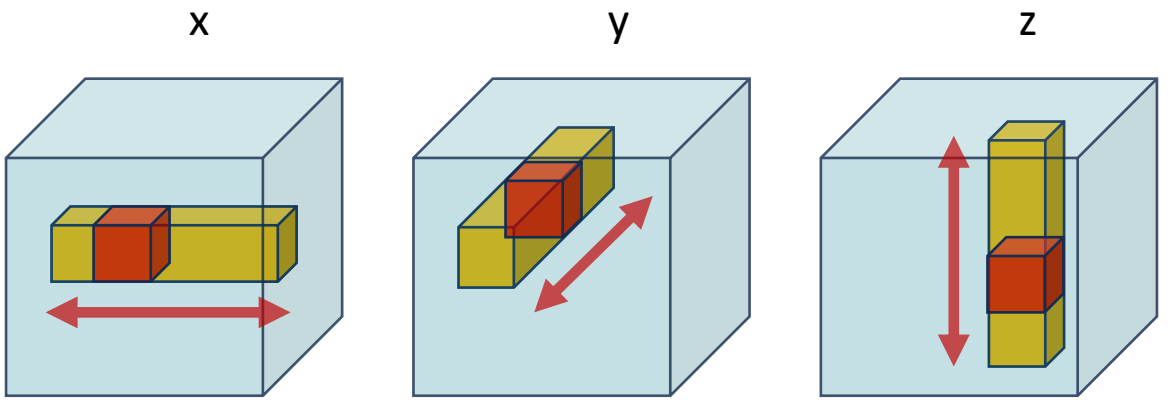
-- Leonardo Ayala, Div. Intelligent Medical Systems, German Cancer Research Center (DKFZ)

dkfz. DEUTSCHES KREBSFORSCHUNGSZENTRUM IN DER HELMHOLTZ-GEMEINSCHAFT

"Btune is a simple and highly effective tool. We tried this out with @LEAPSinitiative data and found some super useful spots in the parameter space of Blosc2 compression arguments! Awesome work, @Blosc2 team!"

-- Peter Steinbach, Helmholtz AI Consultants Team Lead for Matter Research @HZDR_Dresden

HELMHOLTZAI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT



Exploring Gaia Data

Visualize a 3D datagrid with 8 trillion cells and 0.5 billion of stars

Find the scripts and notebooks here: https://github.com/Blosc/exploring-milky-way

Conclusion

Blosc2 and the Multidimensional Milky Way

- **Blosc2 Ndim and NDArray** can be used to easily **handle huge sparse matrices** representing large spatial volumes (in this case, 8 trillion cells)

- **Double partition** allows to explore them **effectively**

- **Btune** allows for **automatically selection** of the best Blosc2 **compression parameters**

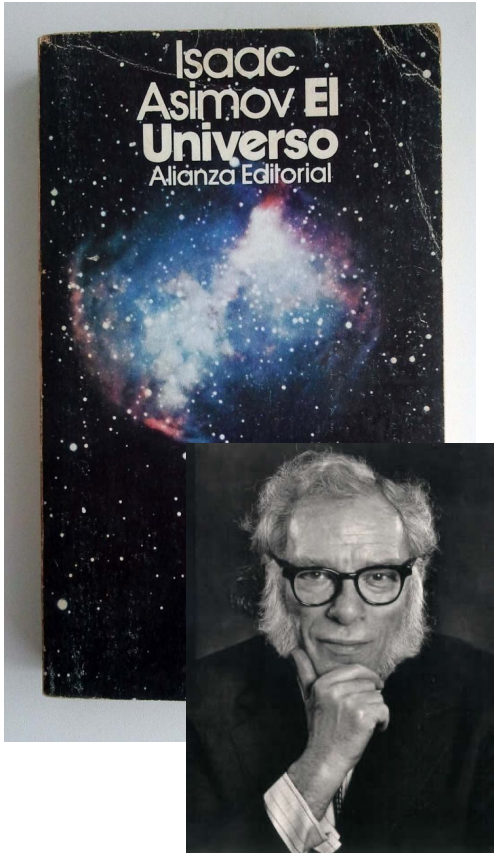
Blosc2 represents a highly efficient and flexible tool for **compressing your data, your way**

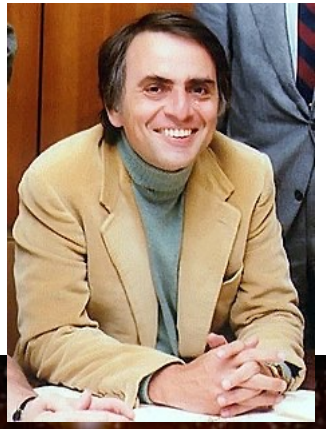
Thanks to donors & contracts!

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 2.10.0.

Thanks for Inspiration!





Thank you! Questions?








We make compression better