HiPSCat: Enabling Storage and Analytics of Large-Scale Catalogs

Mario Juric
DiRAC Institute Director | LINCC FW Co-I
Professor of Astronomy, University of Washington

with Sam Wyatt, Sean McGuire, Melissa DeLucchi, Max West, Doug Branton, Neven Caplar, Steven Stetzler, Colin Slater, Jeremy Kubica
and the LINCC Frameworks Analytics Group

DATA INTENSIVE RESEARCH IN ASTROPHYSICS AND COSMOLOGY

AST-2003196

The LINCC Frameworks Project

LSST Interdisciplinary Network For Collaboration And Computing

To collaboratively develop open computing systems and algorithms needed for large-survey analyses.

- Data analysis infrastructure (this talk)
- Solar System exploration
- Time domain science
- Extragalactic astronomy

Two LINCC-FW hubs:

- Carnegie Mellon University
- University of Washington

The Legacy Survey of Space and Time

Deep synoptic optical survey, coming in 2025.

Repeated imaging of the visible sky to ~24th mag.
10 years of operation.
60 PB of raw data.
40 billion stars, galaxies, asteroids.
30 trillion observations.

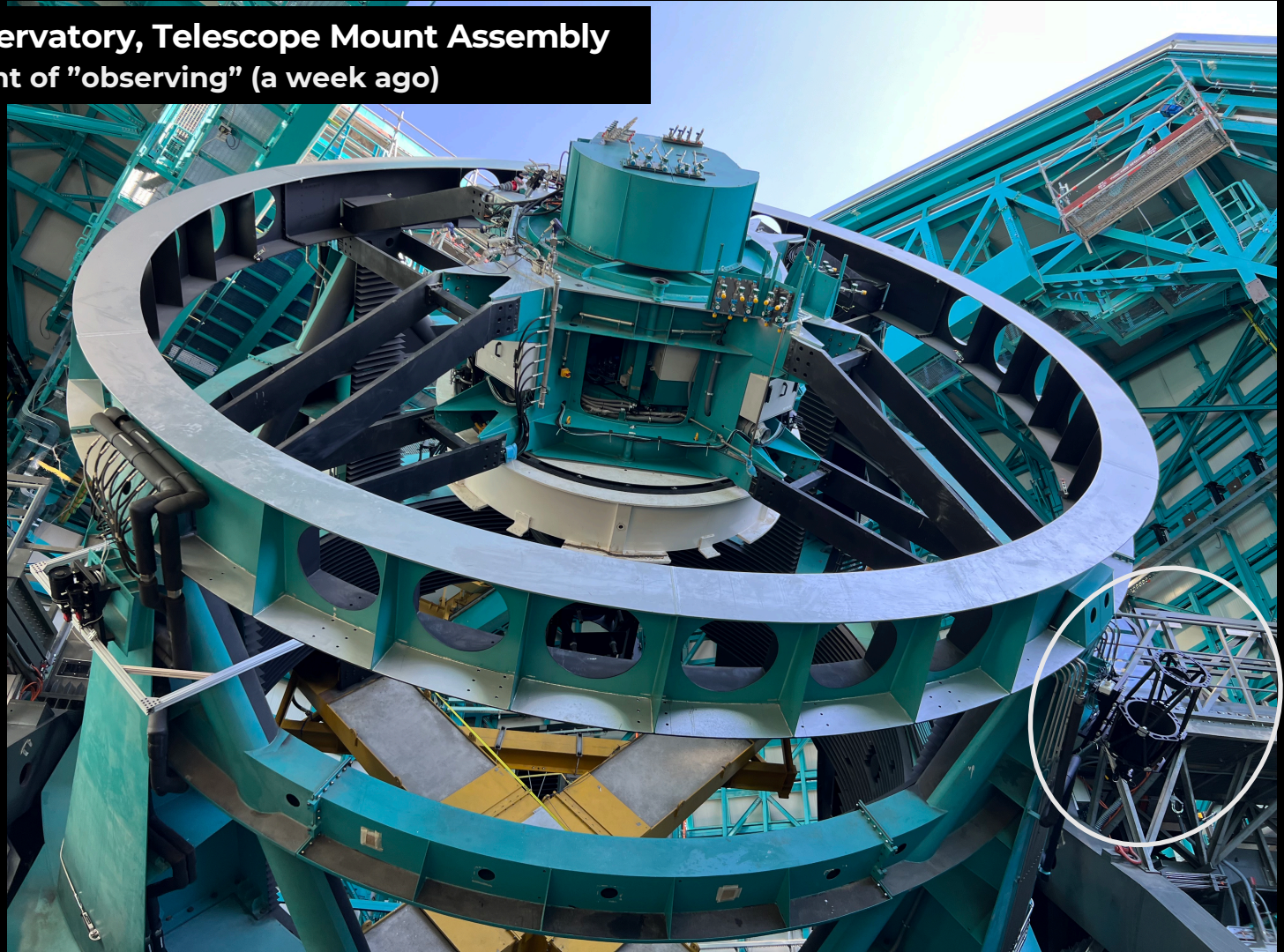
SCHMIDT FUTURES

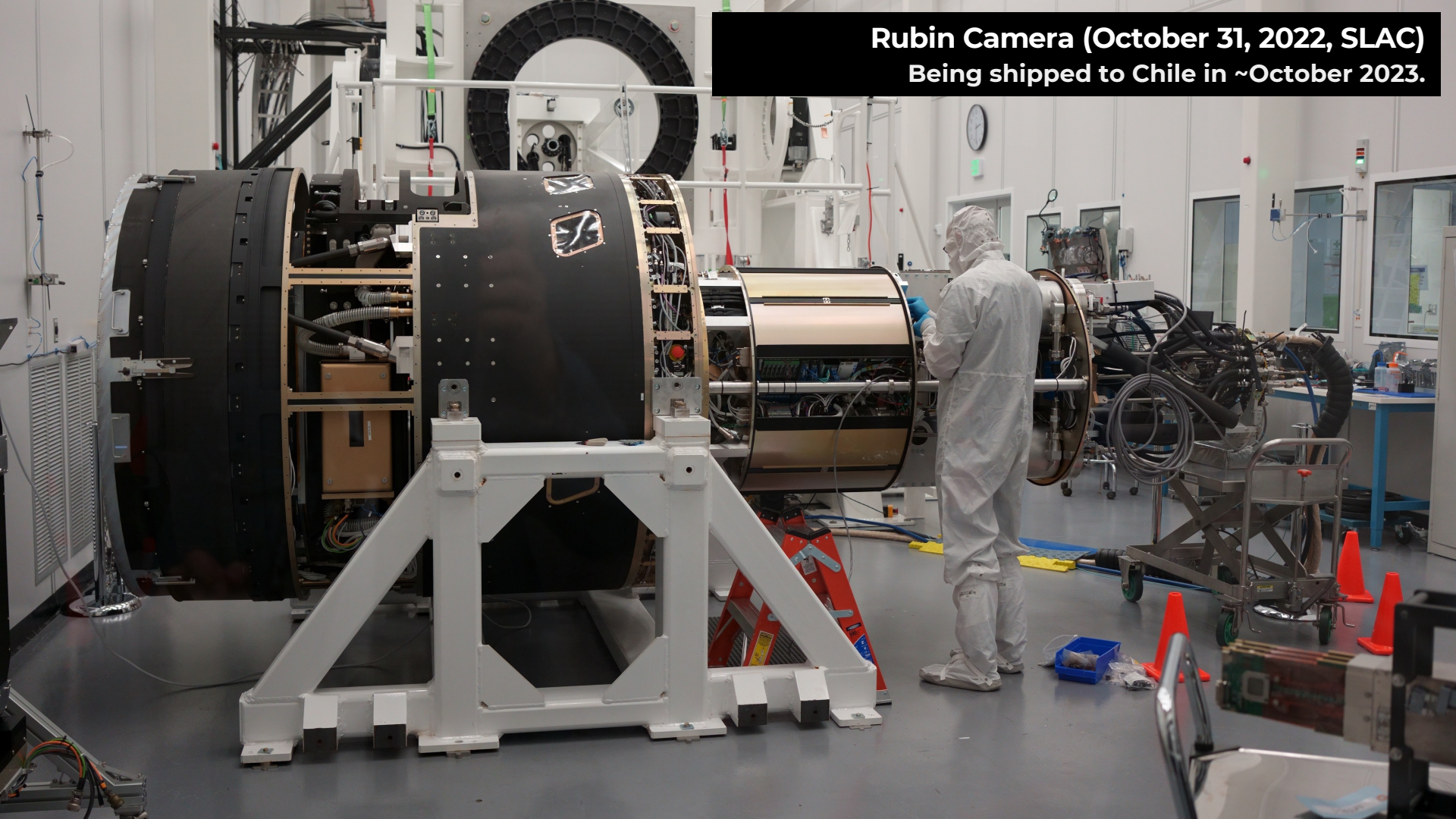
Rubin Observatory, March 15, 2023.

Cerro Pachon, Chile

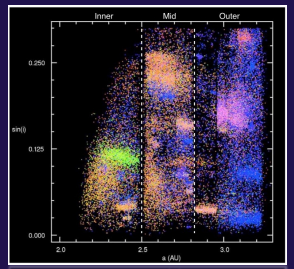
Rubin Observatory, Telescope Mount Assembly

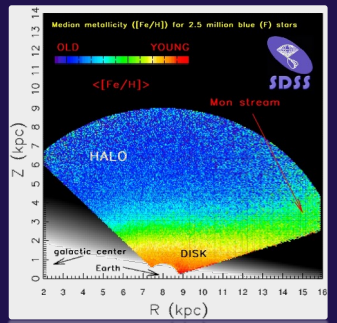
Ran full night of ”observing” (a week ago)

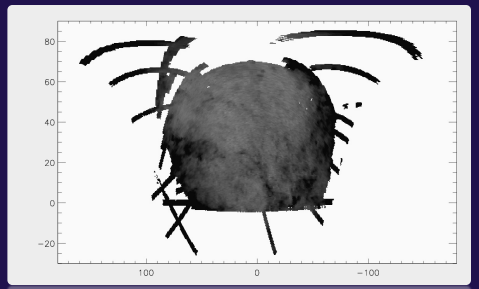



Rubin Camera (October 31, 2022, SLAC)
Being shipped to Chile in ~October 2023.

One of the major things the community will want to do with Rubin is <u>whole dataset science.</u>







Scale of the problem

Rubin Year #1 dataset:

- 10Bn objects
- 100 obsv/object == 1T observations
- 100 bytes/obsv == **100 TB**

Not just a Rubin Problem:

- Gaia, DES, ZTF, WISE, PS, Euclid, Rubin, Roman, SphereX, …
- Each one of these is Bn+ objects (w. many more measurements)

Fellow Travelers on the Quest

Sam Wyatt (Product Owner), Sean McGuire, Melissa DeLucchi (Project Manager)
Max West, Doug Branton, Neven Caplar, Steven Stetzler, Jeremy Kubica

Vandana Desai (IPAC), Troy Raen (IPAC), Dave Shupe (IPAC), Brigitta Sipőcz (IPAC)
Gregory Dubois-Felsmann (IPAC & Rubin)
Colin Slater (DiRAC & Rubin)
Sharon Shen (STScI)
Susan Mullally (STScI)
Rick White (STScI)
Bernie Shiao (STScI)
Travis Berger (STScI)
Erik Tollerud (STScI)
Josh Peek (STScI)
Tess Jaffe (HESARC)

+ YOU (join the party!)

Large-dataset Analytics: Partitioned Files

- Relational databases are not ideal for this type of work. Poor UX, too many bottlenecks.
- Industry state of-the-art is to use distributed analytics tools operating on files.
- Distributed computation achieved through partitioning.

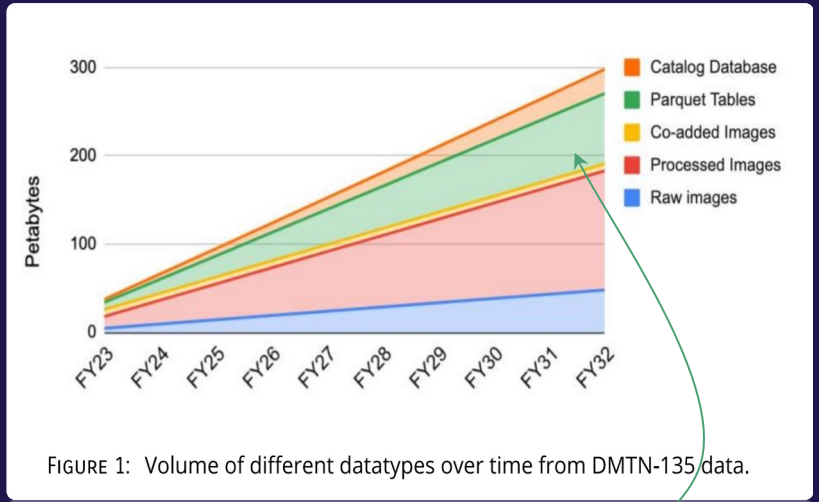
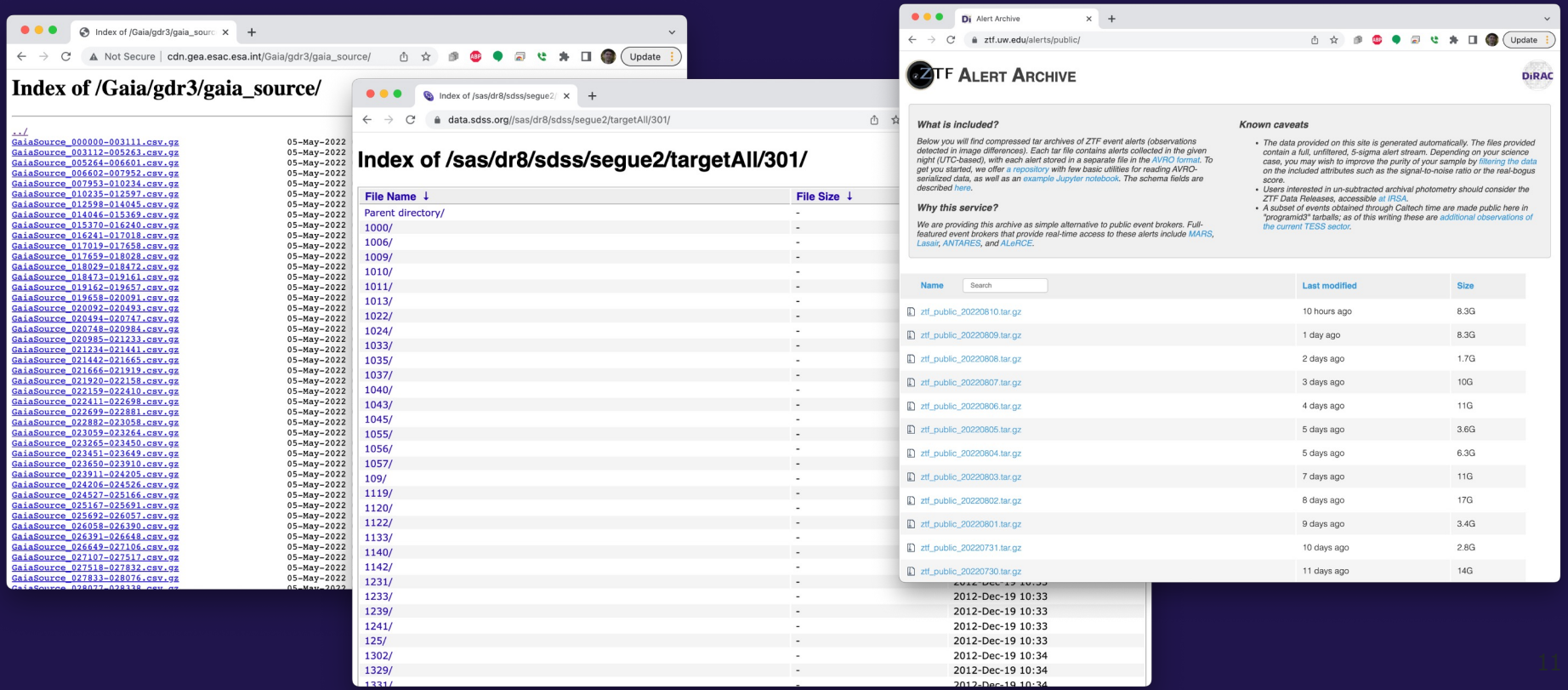


FIGURE 1: Volume of different datatypes over time from DMTN-135 data.

```
import pandas as pd
pd.read_parquet('example_pa.parquet', engine='pyarrow')
```

How to partition? Historically, we haven't generally given this much thought…



There’s value in thinking this through, and standardizing

- Users know what to expect and how to handle the dataset
- High quality, shared, analytics tools can be written
- Multi-dataset analytics can supported
- Pre-staging/ETL may be avoided

- Bulk export files == bulk analytics files
- Easier to generate and support for providers
- Can share code and infrastructure (e.g. mirroring, caching)

Think of all the wonderful tools and ecosystems that sprung up around HiPS!

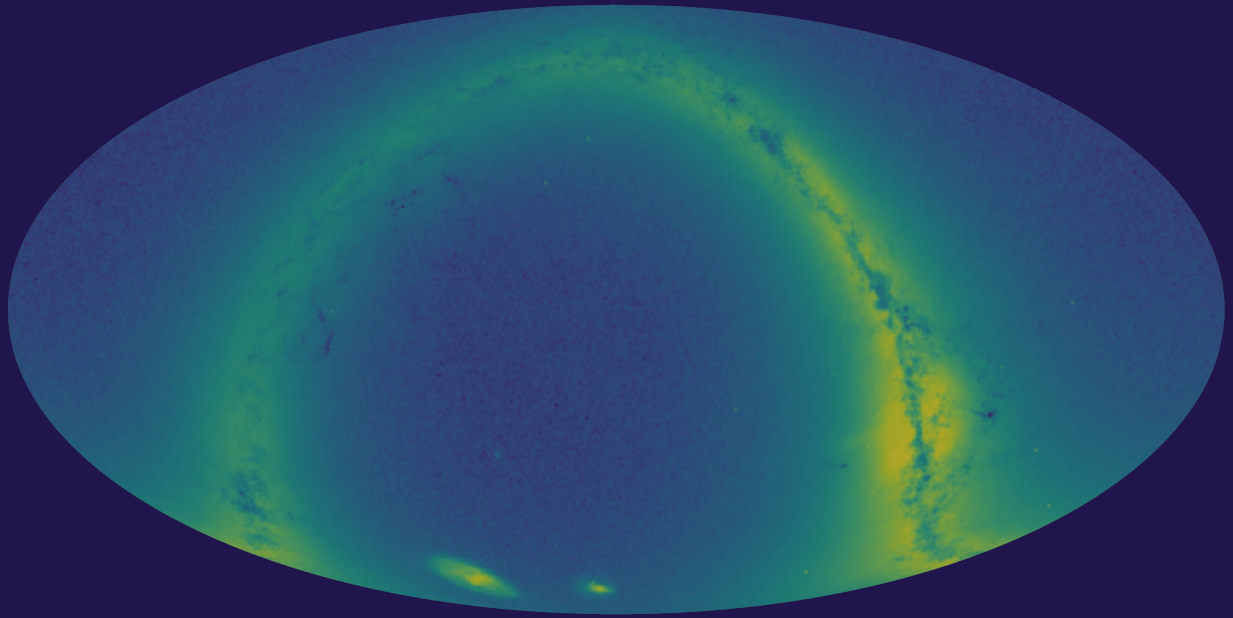
Imagine a dataset with approximately 1,811,709,771 sources…

… how could we store it?

Gaia DR3

The 1.8Bn sources released in Gaia DR3

A single ASCII file would be about ~680GB in size, (gzip compressed).



1. Partitioning: HEALPix

Partition the sky into NSIDE=1 (order=0) HEALPix tiles, map tiles to files.

Example:

```
Norder0-Npix0.tsv.gz
Norder0-Npix1.tsv.gz
Norder0-Npix2.tsv.gz
Norder0-Npix3.tsv.gz
Norder0-Npix4.tsv.gz
Norder0-Npix5.tsv.gz
Norder0-Npix6.tsv.gz
Norder0-Npix7.tsv.gz
Norder0-Npix8.tsv.gz
Norder0-Npix9.tsv.gz
Norder0-Npix10.tsv.gz
Norder0-Npix11.tsv.gz
```

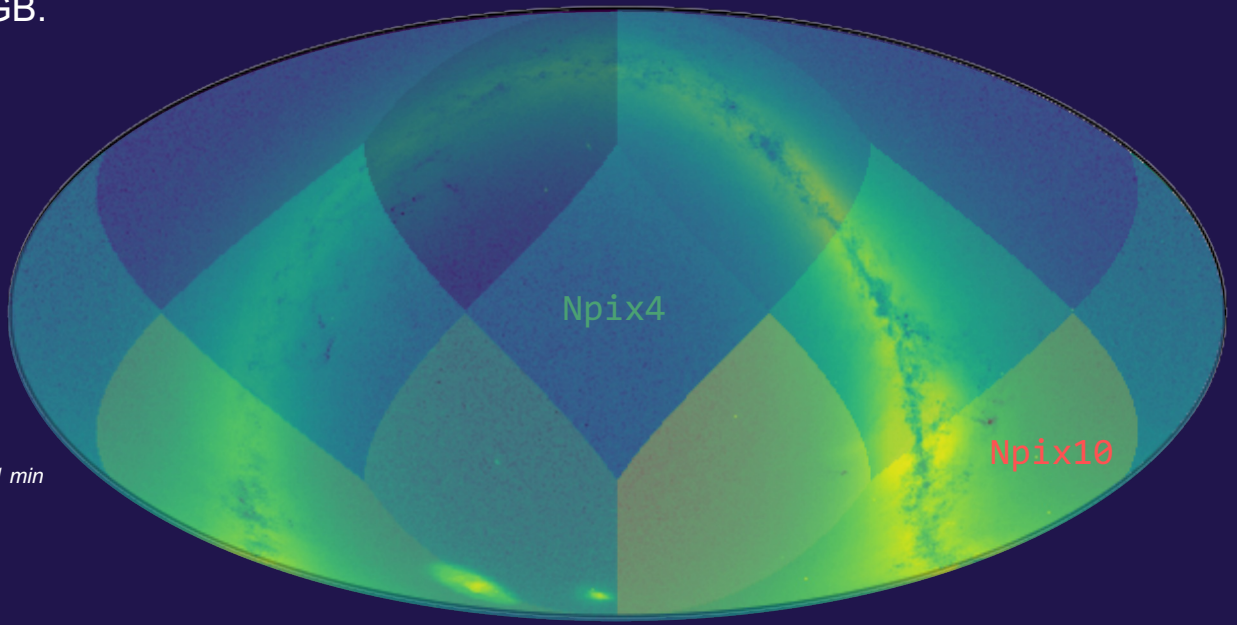
Problem: Severely unbalanced file sizes

Pixel 4 (Galactic pole) ~ 20GB
Pixel 10 (Galactic center) ~ 400GB.

Simple file-based parallelization fails.

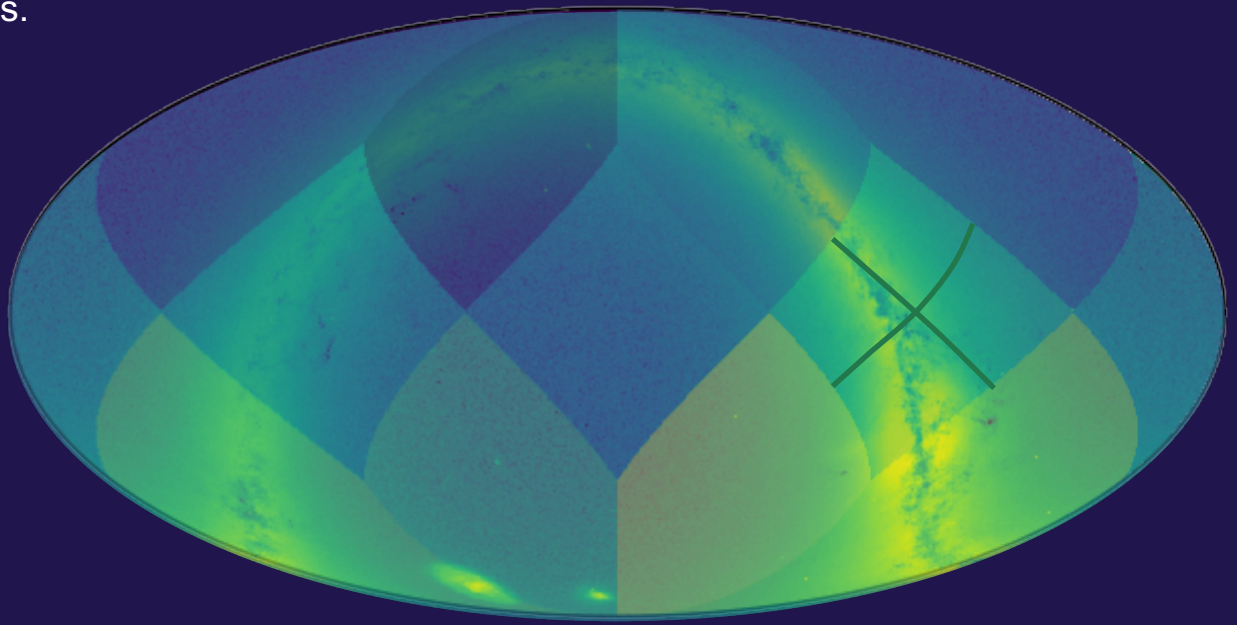
Example

```
Norder0-Npix0.tsv.gz
Norder0-Npix1.tsv.gz
Norder0-Npix2.tsv.gz
Norder0-Npix3.tsv.gz
Norder0-Npix4.tsv.gz   → 1 min
Norder0-Npix5.tsv.gz
Norder0-Npix6.tsv.gz
Norder0-Npix7.tsv.gz
Norder0-Npix8.tsv.gz
Norder0-Npix9.tsv.gz
Norder0-Npix10.tsv.gz  → 20 min
Norder0-Npix11.tsv.gz
```



Solution: Partition Hierarchically

If too many sources fall into a pixel,
split it into four higher order pixels.

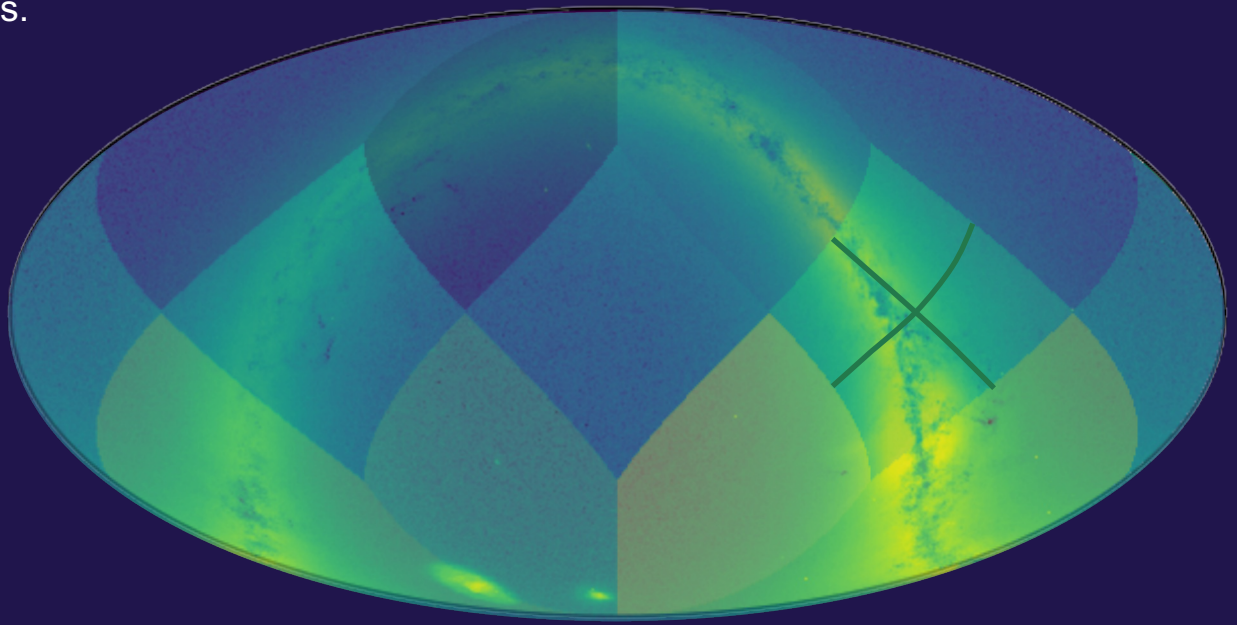


Example

```
Norder0-Npix0.tsv.gz
…
Norder0-Npix7.tsv.gz
…
Norder0-Npix11.tsv.gz
```

1. Partition Hierarchically

If too many sources fall into a pixel,
split it into four higher order pixels.



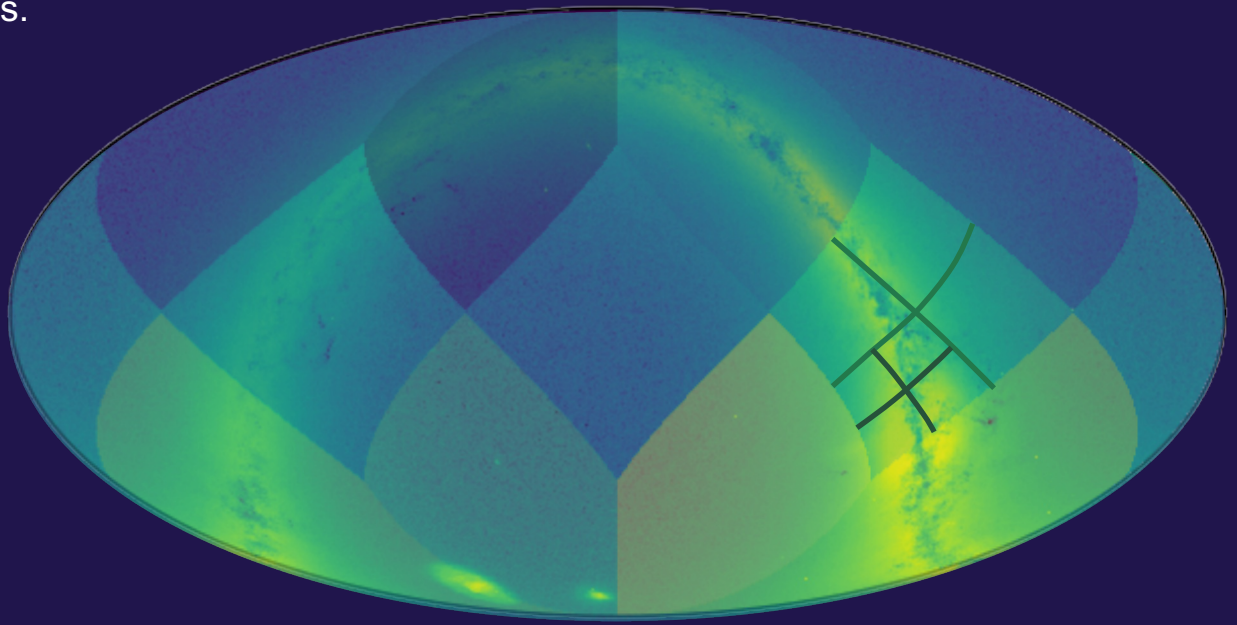
Example

```
Norder0-Npix0.tsv.gz
…
Norder1-Npix28.tsv.gz
Norder1-Npix29.tsv.gz
Norder1-Npix30.tsv.gz
Norder1-Npix31.tsv.gz
…
Norder0-Npix11.tsv.gz
```

1. Partition Hierarchically

If too many sources fall into a pixel,
split it into four higher order pixels.

Repeat.



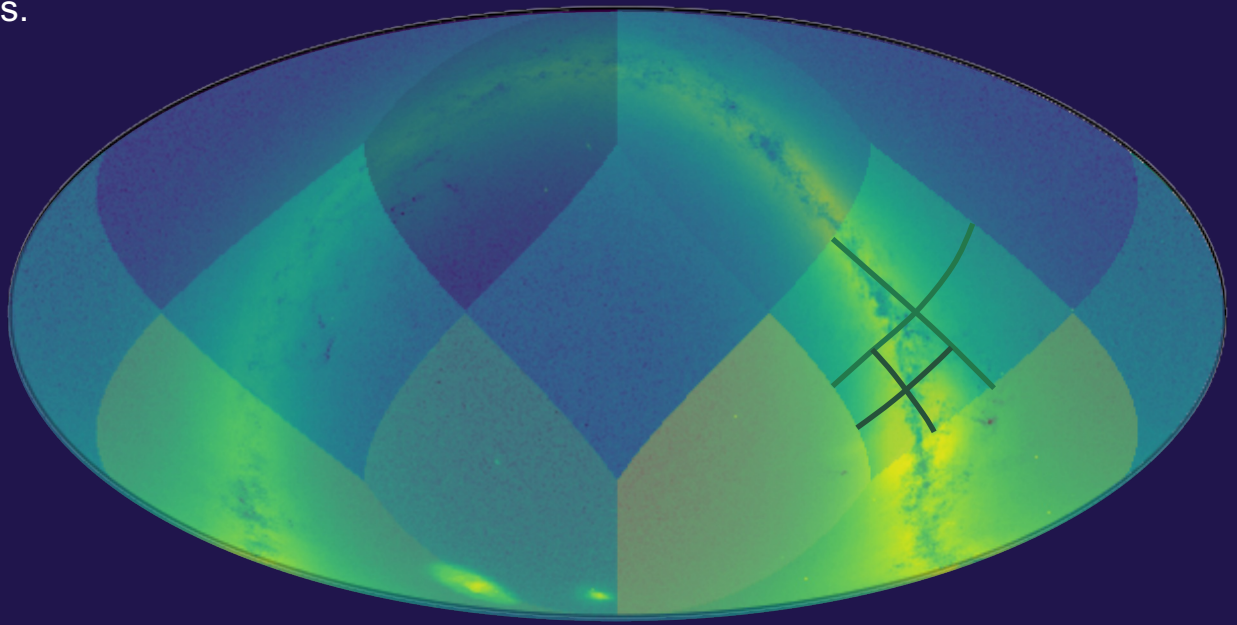
Example

```
Norder0-Npix0.tsv.gz
…
Norder1-Npix28.tsv.gz
Norder1-Npix29.tsv.gz
Norder1-Npix30.tsv.gz
Norder1-Npix31.tsv.gz
…
Norder0-Npix11.tsv.gz
```

1. Partition Hierarchically

If too many sources fall into a pixel,
split it into four higher order pixels.

Repeat.



Example

```
Norder0-Npix0.tsv.gz
…
Norder1-Npix28.tsv.gz
Norder1-Npix29.tsv.gz
Norder1-Npix30.tsv.gz
Norder2-Npix112.tsv.gz
Norder2-Npix113.tsv.gz
Norder2-Npix114.tsv.gz
Norder2-Npix115.tsv.gz
…
Norder0-Npix11.tsv.gz
```

1. Partition Hierarchically

If too many sources fall into a pixel, split it into four higher order pixels.

Repeat until each file size is beneath some pre-defined threshold.

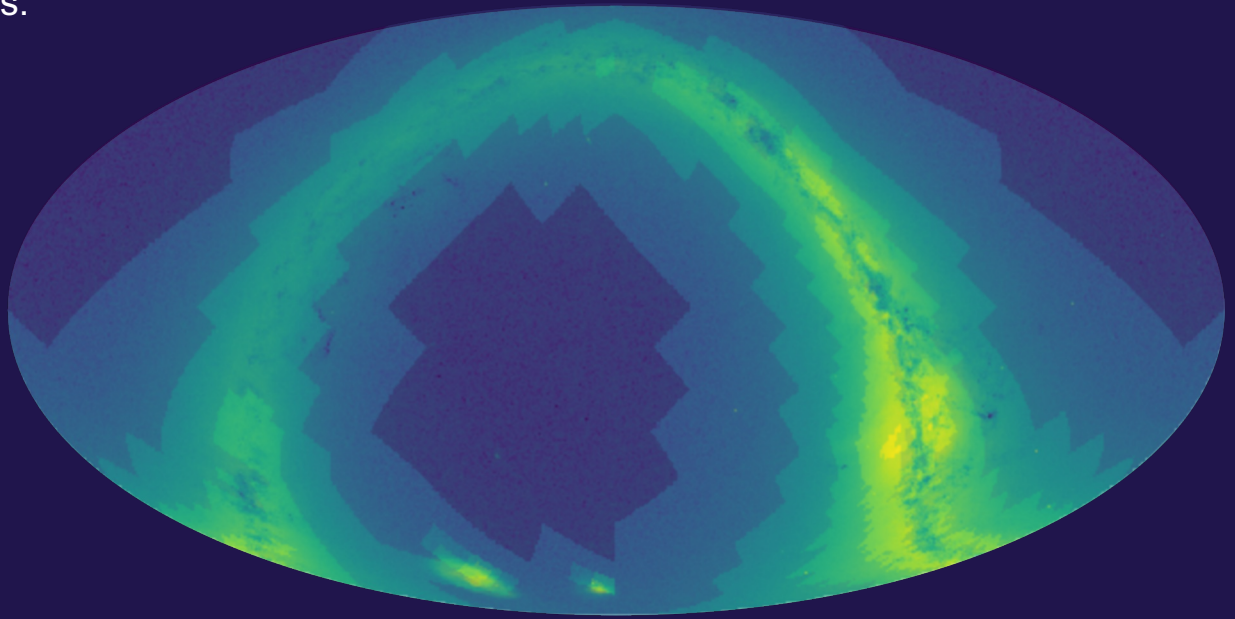


Figure: an overlay of Gaia counts and the partitioning map, taking MAXOBJECTS=1e6

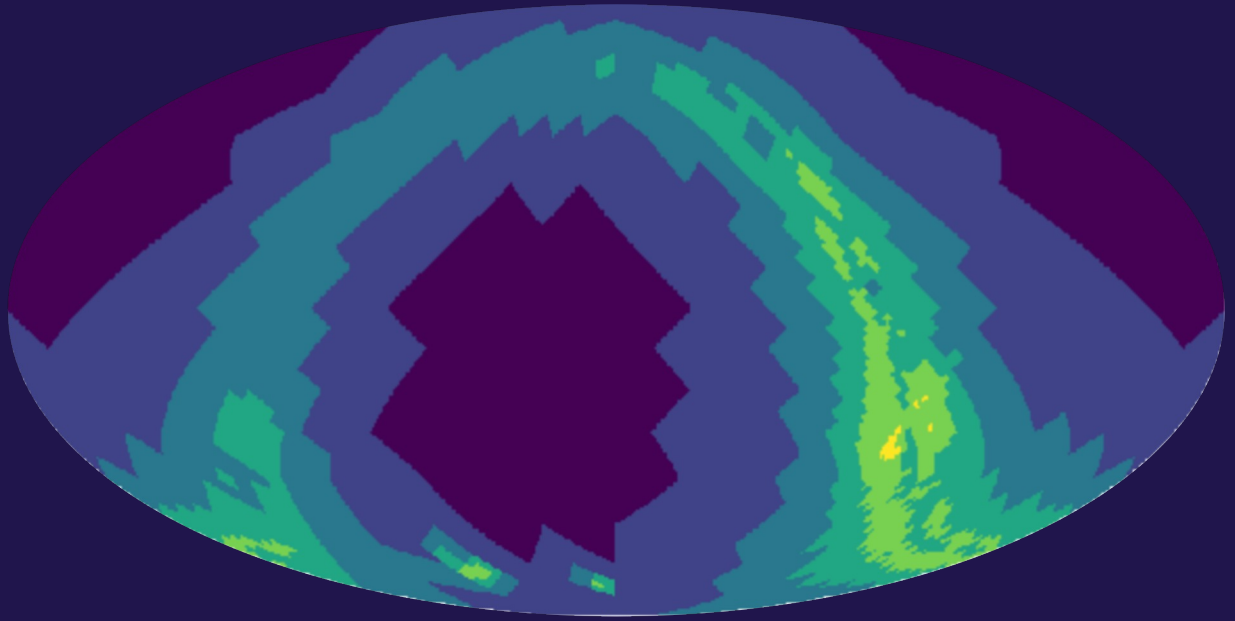
order:

3993 partitions for Gaia DR3, with 1M object/partition threshold

2. On-disk organization

Holding everything in a single directory is unwieldy (at best).

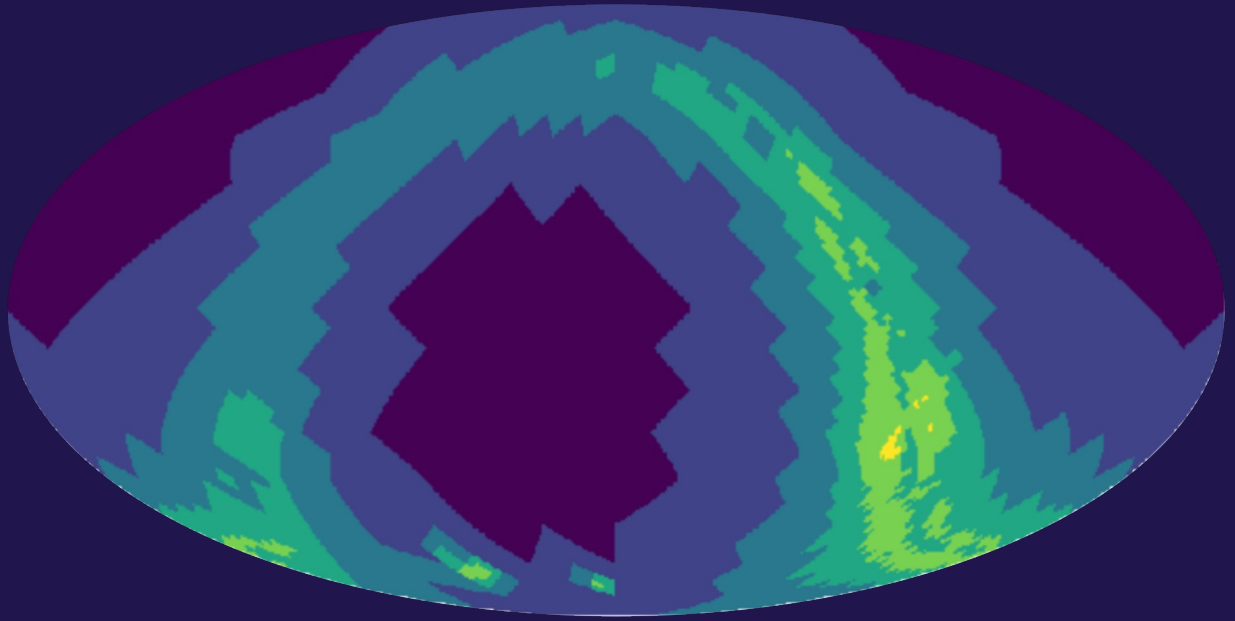
A directory structure encoding the hierarchy would be helpful.



```
Norder0-Npix0.tsv.gz
…
Norder1-Npix28.tsv.gz
Norder1-Npix29.tsv.gz
Norder1-Npix30.tsv.gz
Norder2-Npix112.tsv.gz
Norder2-Npix113.tsv.gz
Norder2-Npix114.tsv.gz
Norder2-Npix115.tsv.gz
…
Norder0-Npix11.tsv.gz
```

2. On-disk organization: HiPS-like Directories

Fortunately, we have a precedent in VO on how to organize hierarchically partitioned HEALPix data: HiPS.

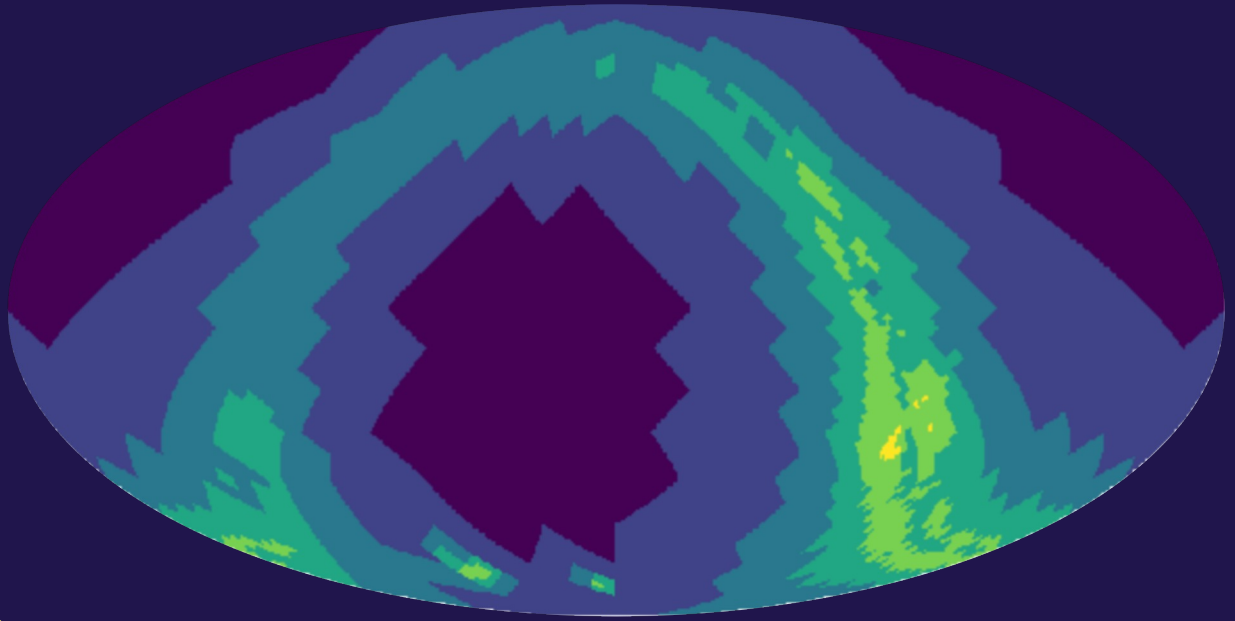


```
Norder0/Dir0/Npix0.tsv.gz
…
Norder1/Dir0/Npix28.tsv.gz
Norder1/Dir0/Npix29.tsv.gz
Norder1/Dir0/Npix30.tsv.gz
Norder2/Dir0/Npix112.tsv.gz
Norder2/Dir0/Npix113.tsv.gz
Norder2/Dir0/Npix114.tsv.gz
Norder2/Dir0/Npix115.tsv.gz
…
Norder0/Dir0/Npix11.tsv.gz
```

Note: Hierarchical ~~Progressive~~ Surveys

While we use a HiPS-like directory structure, *we only store data at the leaf nodes*.

I.e., there are no lower-resolution files at lower orders.



```
Norder0/Dir0/Npix0.tsv.gz
…
Norder1/Dir0/Npix28.tsv.gz
Norder1/Dir0/Npix29.tsv.gz
Norder1/Dir0/Npix30.tsv.gz
Norder2/Dir0/Npix112.tsv.gz
Norder2/Dir0/Npix113.tsv.gz
Norder2/Dir0/Npix114.tsv.gz
Norder2/Dir0/Npix115.tsv.gz
…
Norder0/Dir0/Npix11.tsv.gz
```

e.g., there's no Norder1/Dir0/Npix31.csv

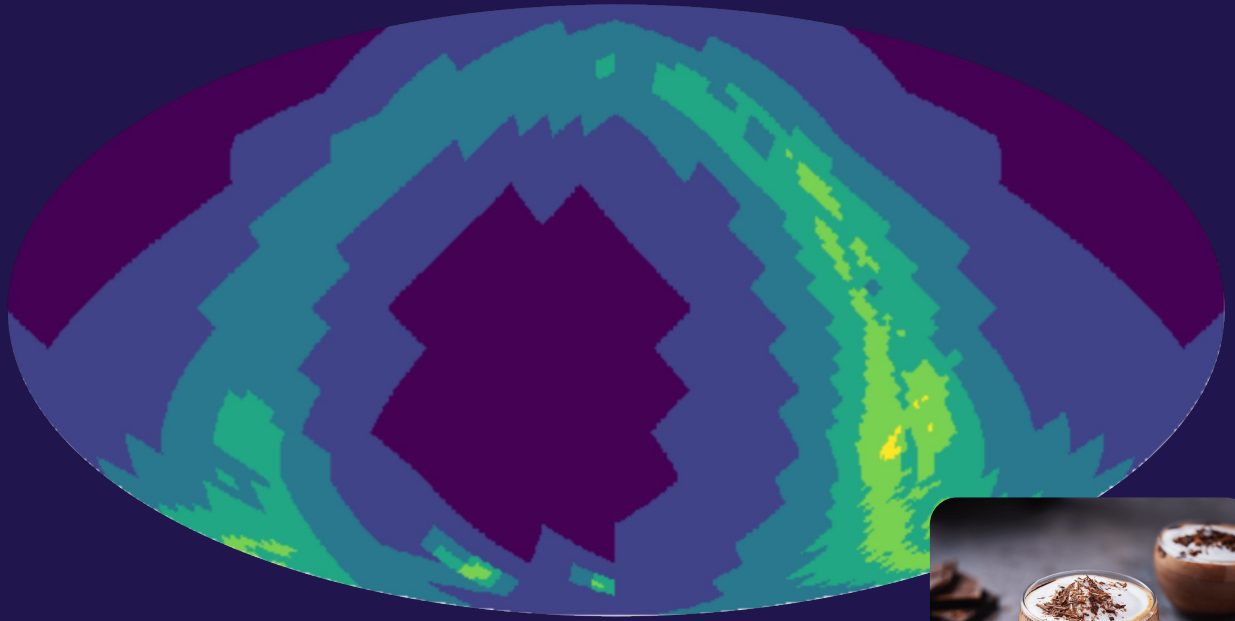
Note: Hierarchical ~~Progressive~~ Surveys

While we use a HiPS-like directory structure, *we only store data at the leaf nodes*.

I.e., there are no lower-resolution files at lower orders.

```
Norder0/Dir0/Npix0.tsv.gz
…
Norder1/Dir0/Npix28.tsv.gz
Norder1/Dir0/Npix29.tsv.gz
Norder1/Dir0/Npix30.tsv.gz
Norder2/Dir0/Npix112.tsv.gz
Norder2/Dir0/Npix113.tsv.gz
Norder2/Dir0/Npix114.tsv.gz
Norder2/Dir0/Npix115.tsv.gz
…
Norder0/Dir0/Npix11.tsv.gz
```

This is really a MOC
(multi-order coverage map; https://ivoa.net/documents/MOC/)



HiPSCat → MOCHA? (e.g. Multi-order Catalogs in HEALpix for Astronomy? Bacronym suggestions welcome!)

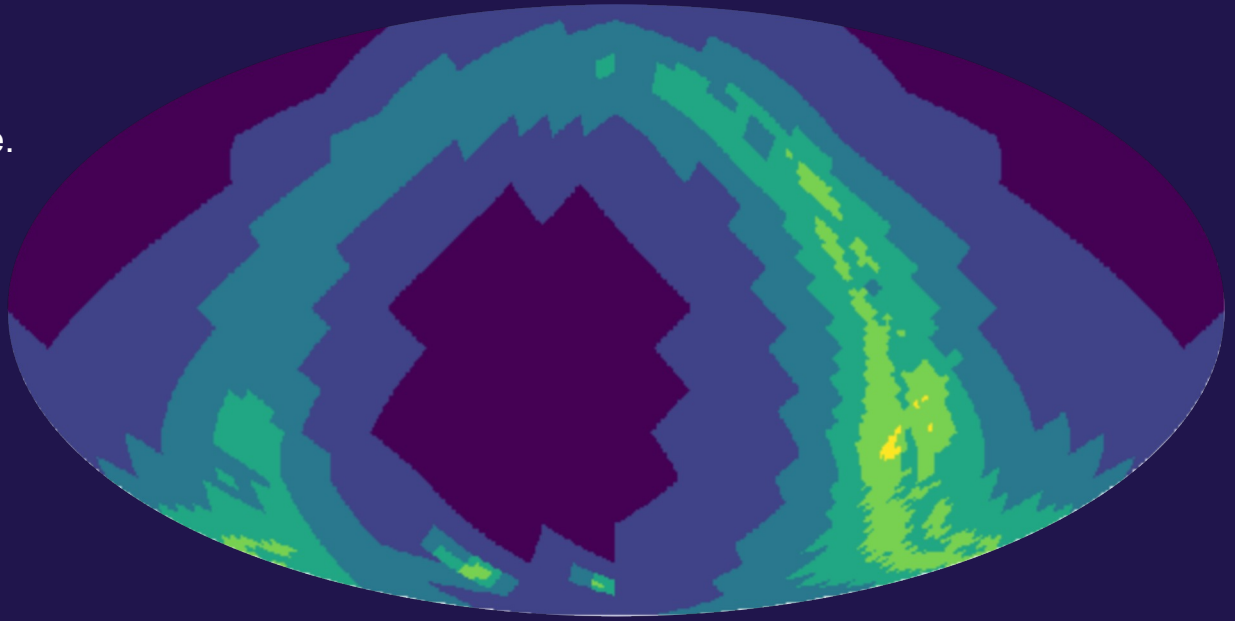
3. Serialization

TSV is not ideal for large catalog storage and analytics.

Time-consuming to parse and (de)compress. Also not seekable.

FITS? HDF5?

```
Norder0/Dir0/Npix0.tsv.gz
…
Norder1/Dir0/Npix28.tsv.gz
Norder1/Dir0/Npix29.tsv.gz
Norder1/Dir0/Npix30.tsv.gz
Norder2/Dir0/Npix112.tsv.gz
Norder2/Dir0/Npix113.tsv.gz
Norder2/Dir0/Npix114.tsv.gz
Norder2/Dir0/Npix115.tsv.gz
…
Norder0/Dir0/Npix11.tsv.gz
```



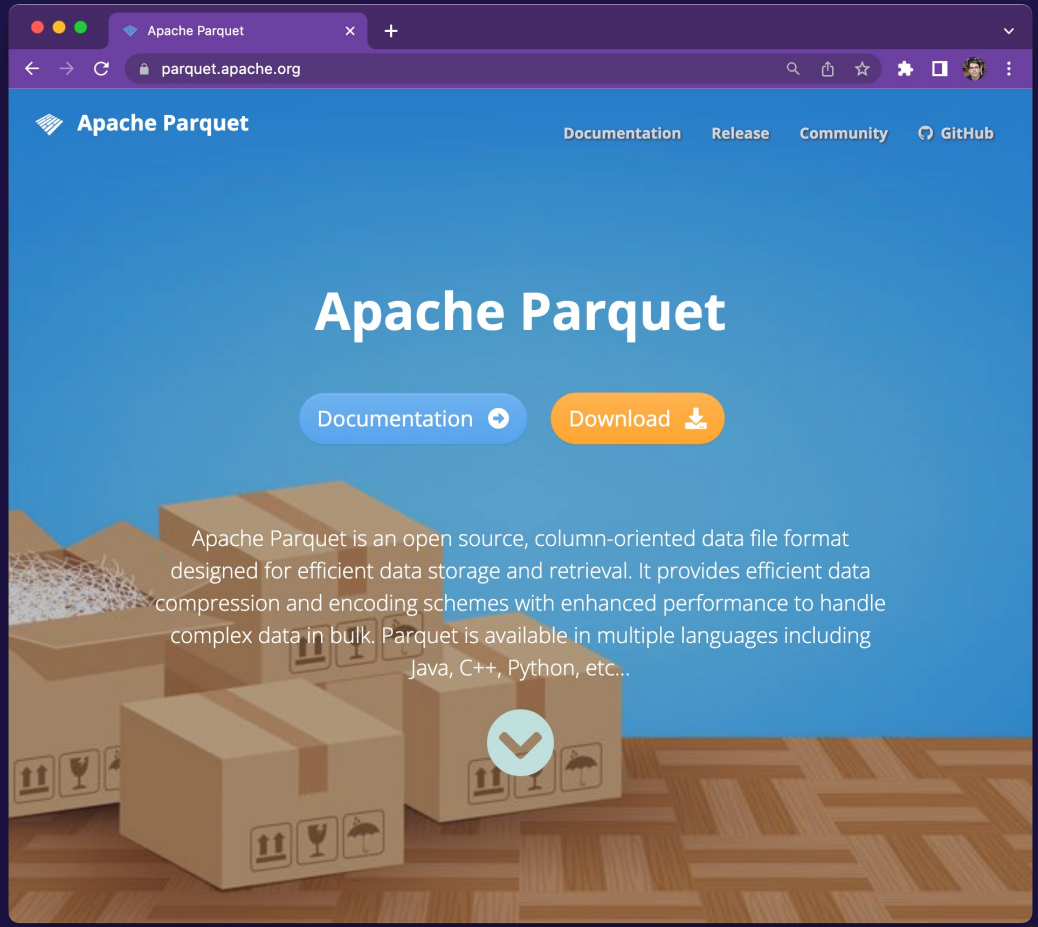
3. Serialization: Parquet

TSV is not ideal for large catalog storage and analytics.

We use Parquet.

<u>Key features:</u>

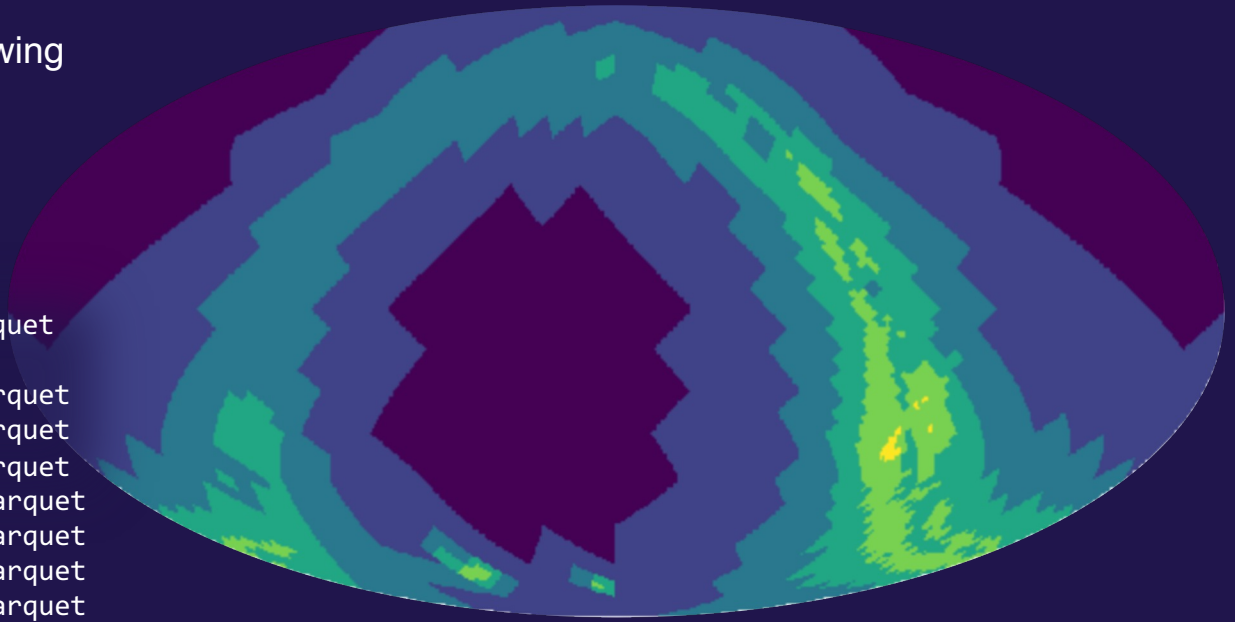
- ✓ Designed for storage of large tables
- ✓ Columnar
- ✓ Efficient (binary)
- ✓ Transparent compression
- ✓ Data Integrity (checksums)
- ✓ Partitioning
- ✓ Broad multi-language support
- ✓ Broad tool support
- ✓ Strong industry backing
- ✓ Open source



3. Serialization: Parquet

Parquet readers natively support reading partitioned datasets if they're stored in directories following `<key>=<value>` naming format.

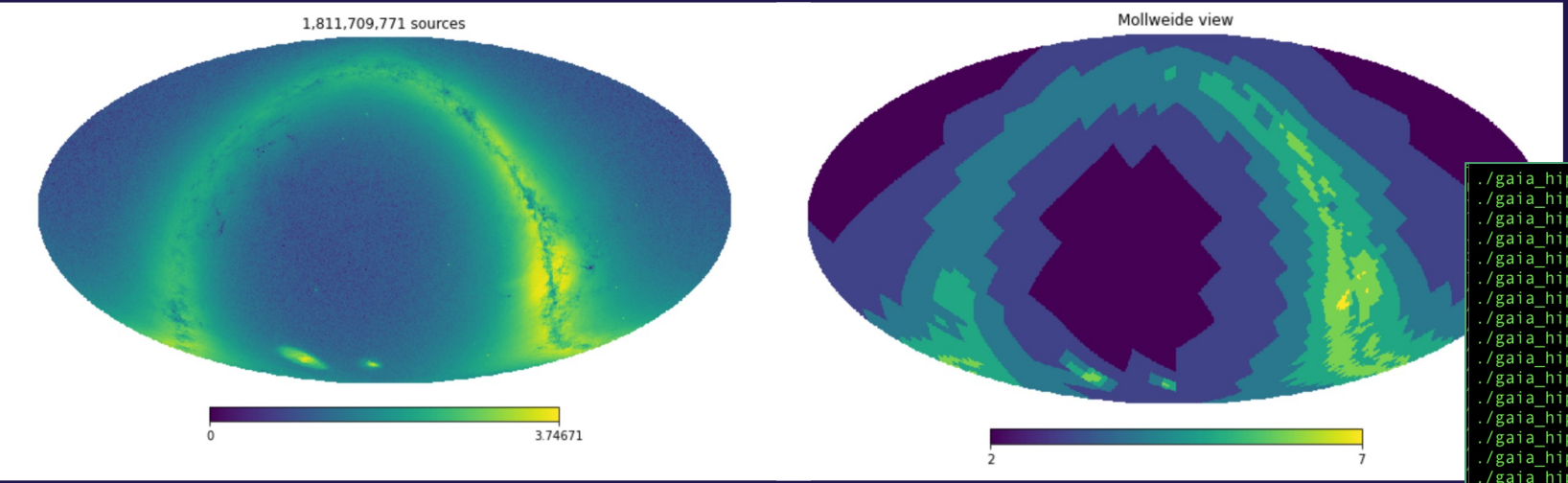
We make that small tweak…



```
Norder=0/Dir=0/Npix=0/catalog.parquet
…
Norder=1/Dir=0/Npix=28/catalog.parquet
Norder=1/Dir=0/Npix=29/catalog.parquet
Norder=1/Dir=0/Npix=30/catalog.parquet
Norder=2/Dir=0/Npix=112/catalog.parquet
Norder=2/Dir=0/Npix=113/catalog.parquet
Norder=2/Dir=0/Npix=114/catalog.parquet
Norder=2/Dir=0/Npix=115/catalog.parquet
…
Norder=0/Dir=0/Npix=11/catalog.parquet
```

(*) We may need to change the name to avoid confusion with true HiPS catalogs

All together: HiPSCat*



Gaia DR2 Catalog Counts (log scale)

Visualization of file storage (color = healpix level)
3933 partitions of similar size (128-256 MB)

Layout on "disk": 👇

```
./gaia_hipscat/Norder5
./gaia_hipscat/Norder5/Npix0
./gaia_hipscat/Norder5/Npix1
./gaia_hipscat/Norder5/Npix10
./gaia_hipscat/Norder5/Npix100
./gaia_hipscat/Norder5/Npix1000
./gaia_hipscat/Norder6/Npix10000
./gaia_hipscat/Norder6/Npix10001
./gaia_hipscat/Norder6/Npix10002
./gaia_hipscat/Norder6/Npix10003
./gaia_hipscat/Norder5/Npix10004
./gaia_hipscat/Norder5/Npix10005
./gaia_hipscat/Norder5/Npix10006
./gaia_hipscat/Norder5/Npix10007
./gaia_hipscat/Norder5/Npix10008
./gaia_hipscat/Norder5/Npix10009
./gaia_hipscat/Norder5/Npix1001
./gaia_hipscat/Norder5/Npix10010
./gaia_hipscat/Norder5/Npix10011
./gaia_hipscat/Norder7/Npix10012
./gaia_hipscat/Norder7/Npix10013
./gaia_hipscat/Norder7/Npix10014
./gaia_hipscat/Norder7/Npix10015
./gaia_hipscat/Norder5/Npix10016
./gaia_hipscat/Norder5/Npix10017
./gaia_hipscat/Norder5/Npix10018
./gaia_hipscat/Norder5/Npix10019
```

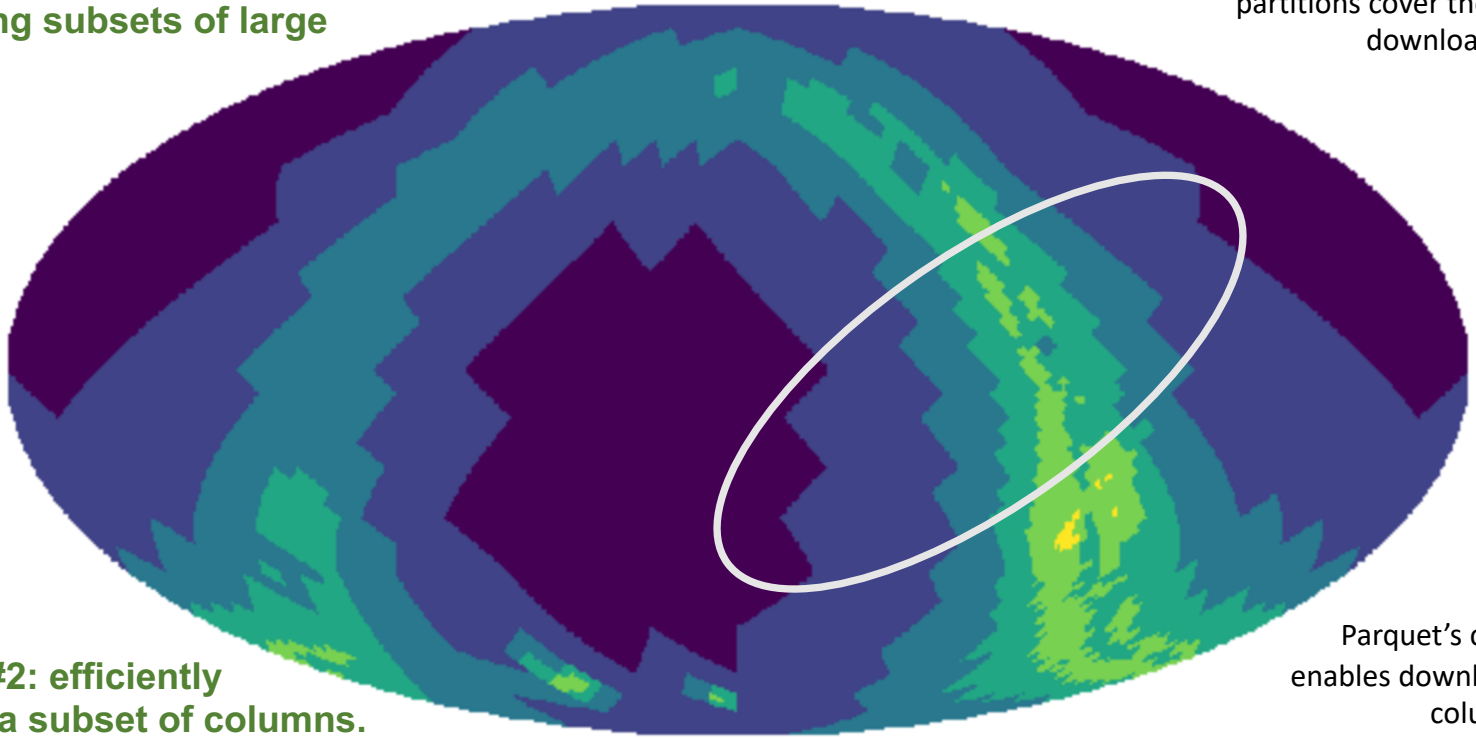
Inspired by the widely used IVOA HiPS standard. Standing on the shoulders of giants (Pierre, Mark, Thomas et al; REC-HIPS-1.0-20170519).

What can we do with this?

Download spatial subsets of a catalog

Use case #1: downloading overlapping subsets of large catalogs.

Given a region of the sky, it's straightforward to find which partitions cover the region (and download those files)



Use case #2: efficiently download a subset of columns.

Parquet's columnar layout enables downloading only the columns of interest

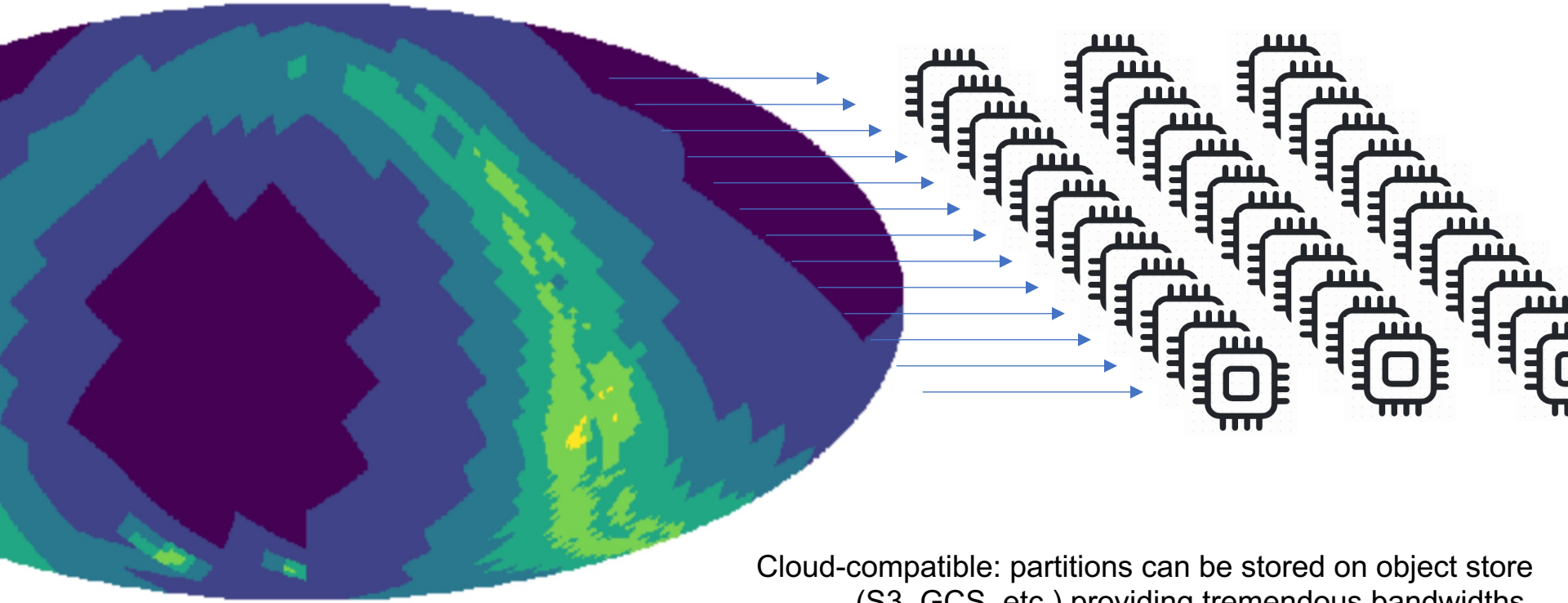
order = 2
pixel size size = 14.7deg

order = 7
pixel size size = 0.46deg

Straightforward Parallel Whole-Catalog Computation

Use case #3: complex searches, feature computation, spatial processing (clustering)

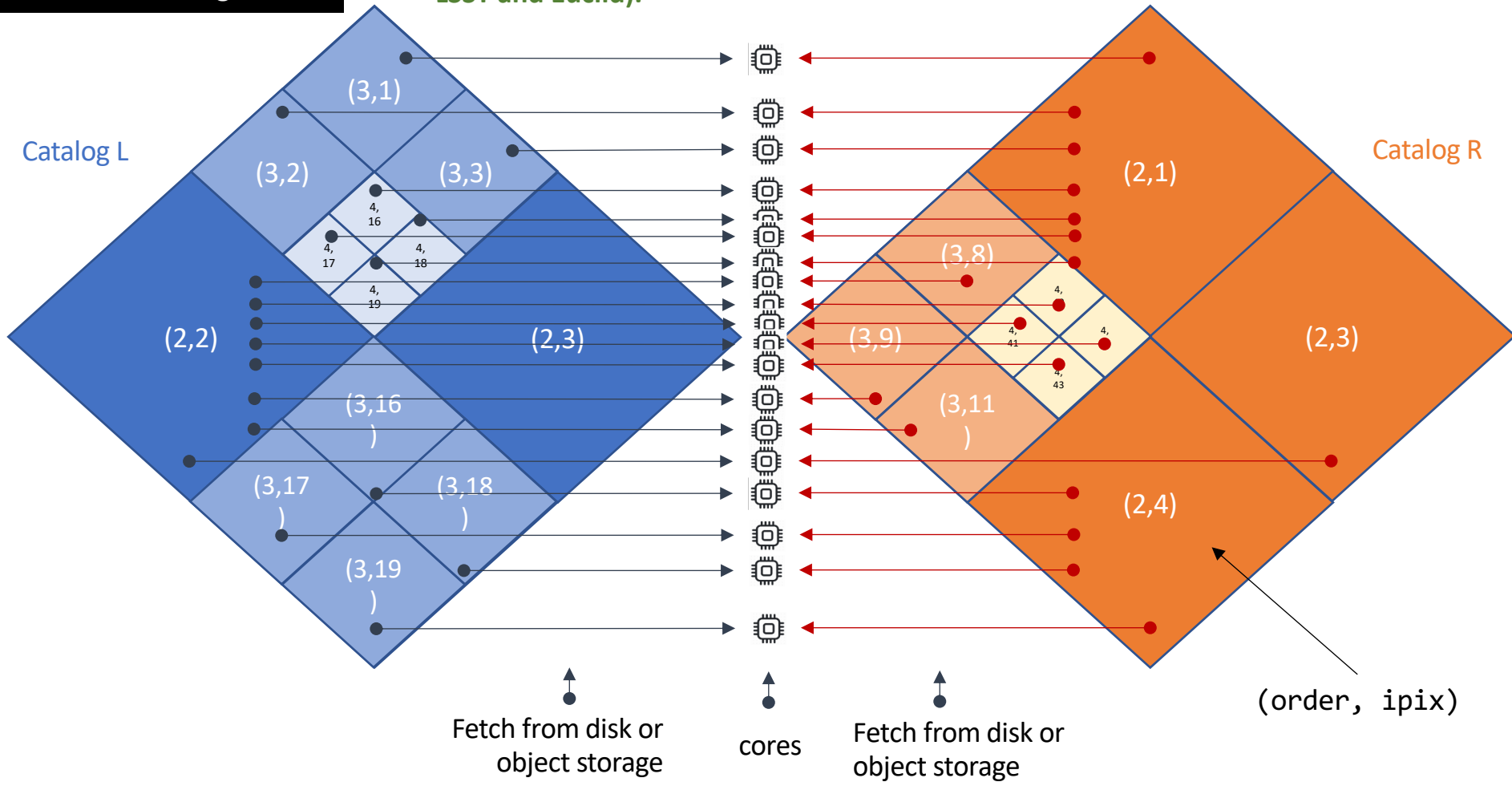
Enables very simple parallel computation schemes: per-file parallelization.



Cloud-compatible: partitions can be stored on object store (S3, GCS, etc.) providing tremendous bandwidths.

Efficient, parallel, joins and crossmatching

Use case #4: distributed analysis on data from two catalogs (example: LSST and Euclid).



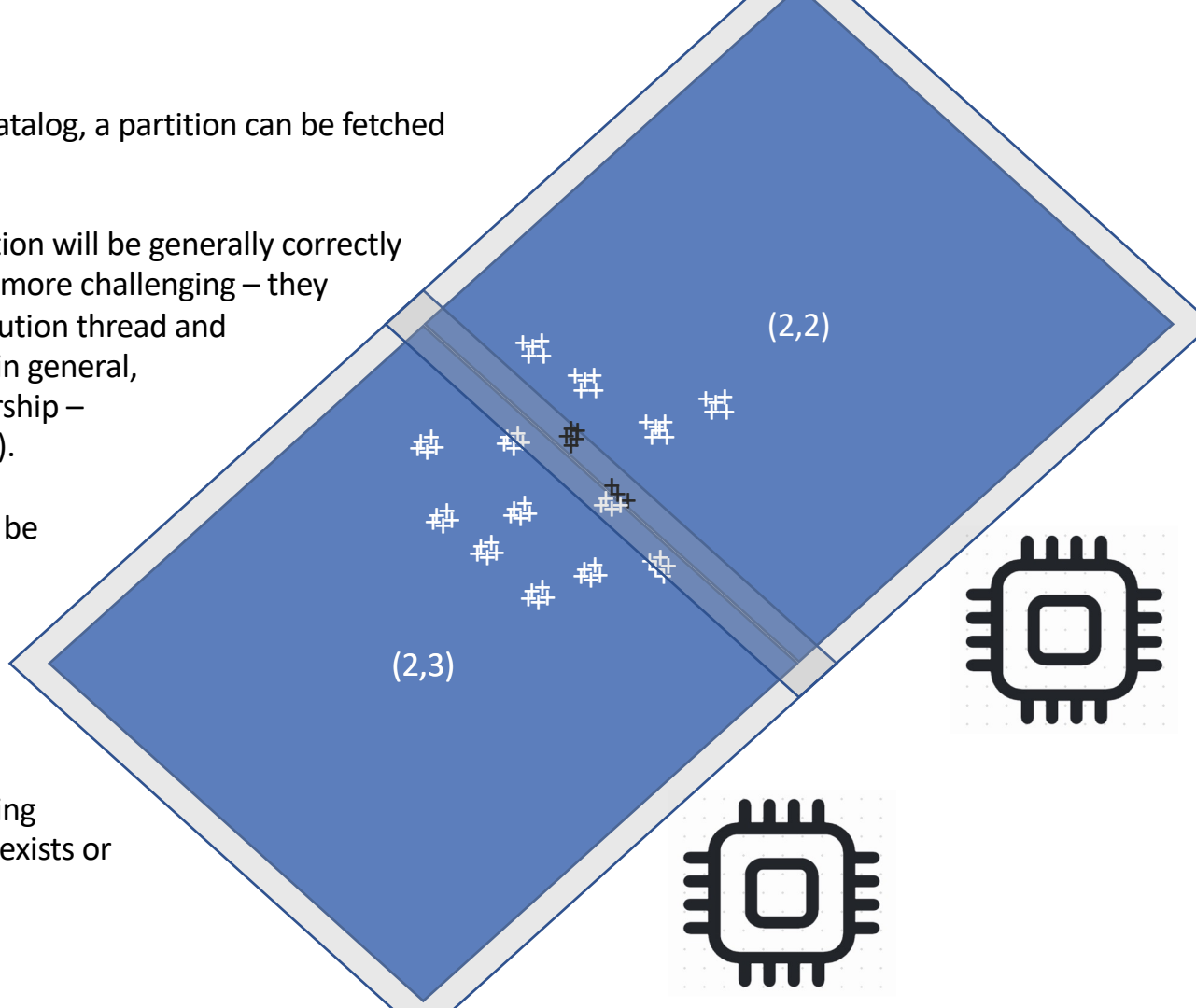
Clustering

Clustering algorithms: Given a source catalog, a partition can be fetched and a clustering algorithm run on it.

Clusters close to the center of the partition will be generally correctly found. Clusters near the edges become more challenging – they can be found by by more than one execution thread and will need to be deduplicated/resolved (in general, they won't even be identical in membership – distributed clustering is a *hard* problem).

Simple 1-pass clustering algorithms can be made to work, and complex algorithms can be implemented in two passes where the 2nd pass only requires O(~1%) of the data to exchange.

The use case here is creation of object catalog from source catalogs via clustering (of interest where no reference catalog exists or cannot be utilized for whatever reason)



Analytics Tools

How do we expose these capabilities to the user and enable science?

Any Tool that Understands Parquet can Read HiPSCat

- Parquet underpins much of modern large-dataset analytics …
- … and HiPSCat is a valid, Hive-partitioned, Parquet dataset.

- We can immediately use existing tools.
- Spark, Dask, Ray, Pandas*, Hive, Hadoop, ….

- For all of these (except for Pandas) analysis execution is inherently distributed, with an API that hides the complexity from the user.

- Still, we would like an ”astronomy-aware” layer…

LSDB: Python Analytics for HiPSCat

- LSDB: Large Survey Database
- Enable Pandas-like analysis on trillions of observations with thousands of cores
- Build on existing tools: Dask (looking at Ray).
- Full HiPSCat awareness: spatial queries, cross-matching, timeseries, multi-dataset joining.
- Very much in pre-alpha/prototype phase; expect usable alphas in the next few weeks

```
img = gaia
      .query("pm > 10")
      .crossmatch(ztf)
      .join(ztf_sources)
      .for_each(varstar_classify)
      .query("pRRLy > 0.95")
      .skymap()

hp.mollview(img)
```

LSDB target APIs: The API center science. Multi-processing, autoscaling, fail-over, etc. are all implicit. Good user experience.

Wyatt et al. (2023)
https://github.com/astronomy-commons/lsdb

Quick Demo

(see it in full on YouTube at https://dirac.us/opj)

ivoa_demo.ipynb U ●

examples > ivoa_demo.ipynb > #viewing imported hipscats

+ Code + Markdown | Run All Clear All Outputs Restart | Variables Outline ··· lsdb-test (Python 3.10.11)

```
import numpy as np
import healpy as hp
from lsd2 import registry as reg
from dask.distributed import Client
```

[1] ✓ 4.2s Python

```
#viewing imported hipscats
HIPS_DIR ='/data3/epyc/projects3/ivoa_demo/'
lsd2_registry = reg.Registry(location=HIPS_DIR)
```

[ ] Python

```
#loading a hipscat
```

[ ] Python

```
#basic catalog visualization
```

[ ] Python

```
#print the gaia columns
```

(see it in full on YouTube at https://dirac.us/opj)

Status

- Ad-hoc collaboration of scientists/engineers from LINCC, Rubin, MAST, IRSA, HESARC, LINeA (Brazil). *JOIN US!*

- Still heavily prototyping (the format isn't yet static)
- Expecting to stabilize the format a bit by ~June.
- Building tools in parallel (dataset import, end-user analytics)

- Will convert a large number of datasets, run science user tests.
- Gather feedback, iterate until we have a solid solution.

- Draft for winter Interop (?)

Mailing list: https://groups.google.com/g/hipscat-wg
Repositories: https://github.com/astronomy-commons
Meetings: *10am PT, June 2nd (then every other wk)*

Much work remains

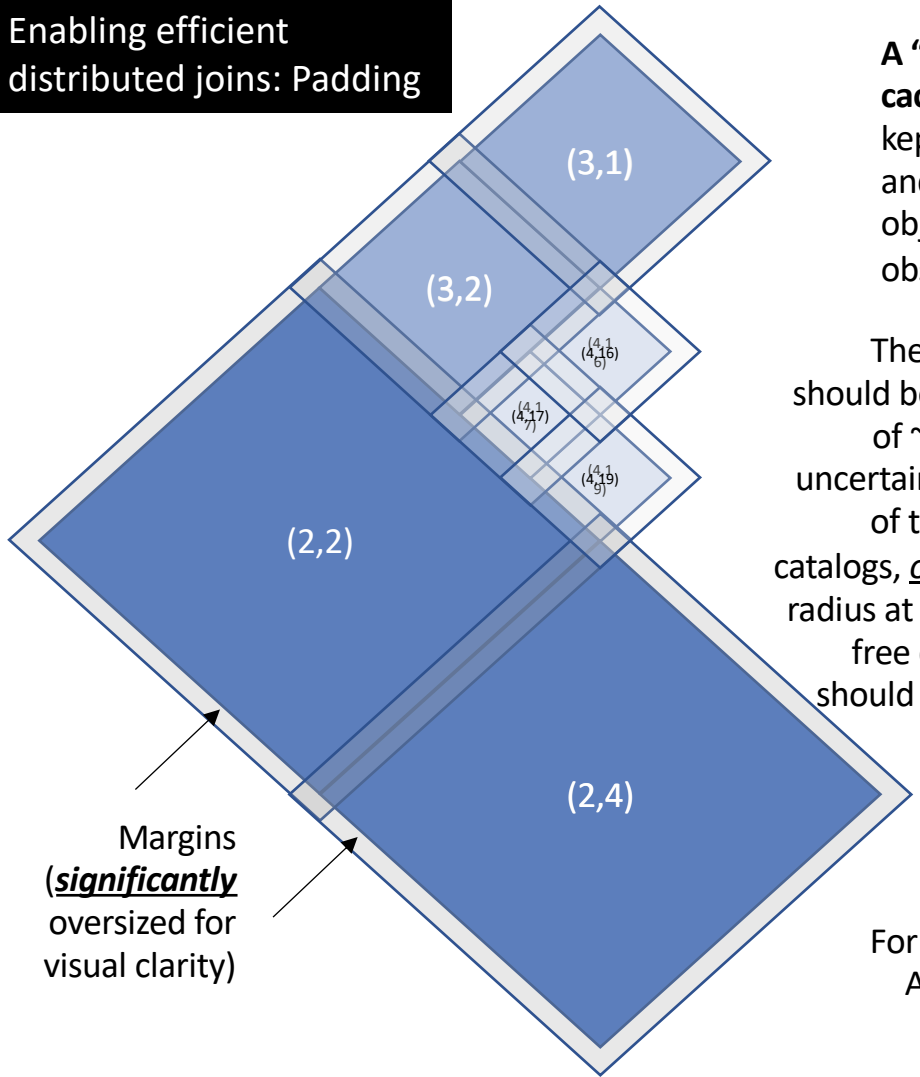
- Format
  - Better integrate with VO standards
  - Supporting variable neighbor margins (for cross-matching)?
  - Supporting efficient joins on tables of moving objects?
  - Temporal partitioning?
  - Supporting spectra?
  - Future transaction support?
- Tooling
  - Core HiPSCat libraries
  - Developing LSDB
  - Spark on HiPSCat
  - Rust, Java, C/C++ libraries
  - …
- Science
  - Importing a variety of catalogs
  - Science case tests
  - Deploy, collect user feedback
  - …

Mailing list: https://groups.google.com/g/hipscat-wg
Repositories: https://github.com/astronomy-commons
Meetings: *10am PT, June 2nd (then every other week)*


LINCC
DiRAC
Collaboratively advancing data-intensive astronomy.
A Universe Understood Through Data-Intensive Discovery
Thank You !
Contact: mjuric@uw.edu
UNIVERSITY of WASHINGTON

Backups

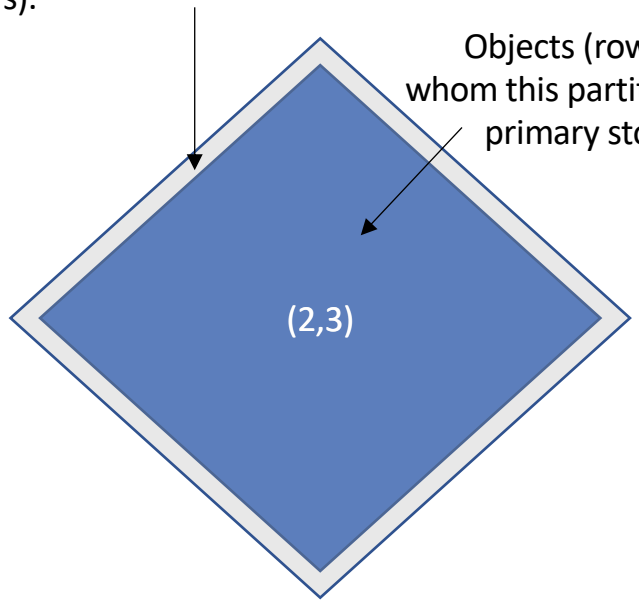
Enabling efficient distributed joins: Padding



Margins (***significantly*** oversized for visual clarity)

A "padding" or "overlap" or "neighbor margin" or "neighbor cache". A *copy* of the rows whose *(ra, dec)* fall into this area will be kept with this partition. This enables distributed, shuffle-free, joins and crossmatches of catalogs where *(ra, dec)* of same objects/sources match only approximately (any catalog with observational errors).

The margin width should be on the order of ~few times the uncertainty in *(ra, dec)* of the rows in the catalogs, *or*, close to the radius at which shuffle-free cross-matches should be supported.

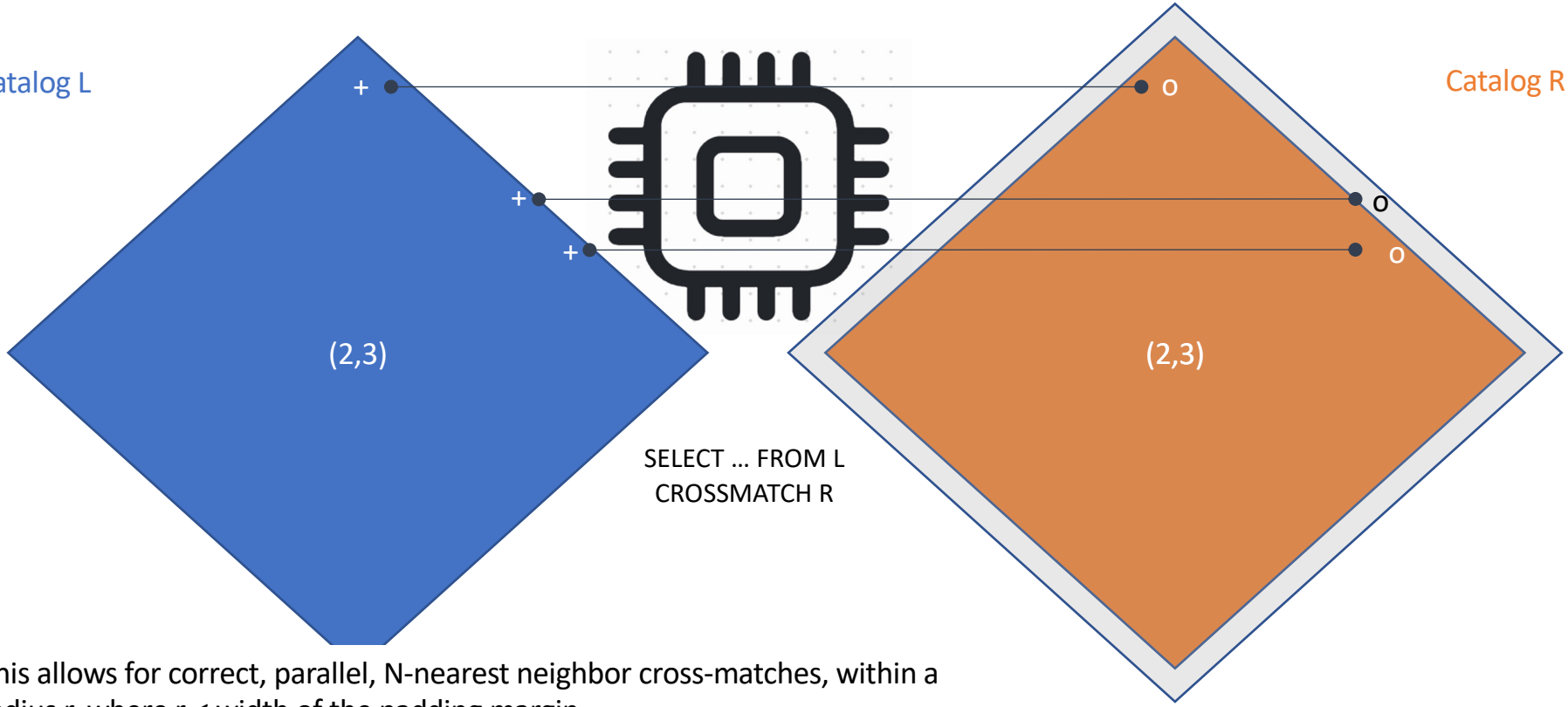


Objects (rows) for whom this partition is primary storage.

For a typical optical large dataset this may be **on order of ~10arcsec**. Assuming a (very) conservative medium partition size of ~0.46deg (1650 arcsec), **the extra storage overhead is roughly ~2.4%**.

How distributed crossmatching works

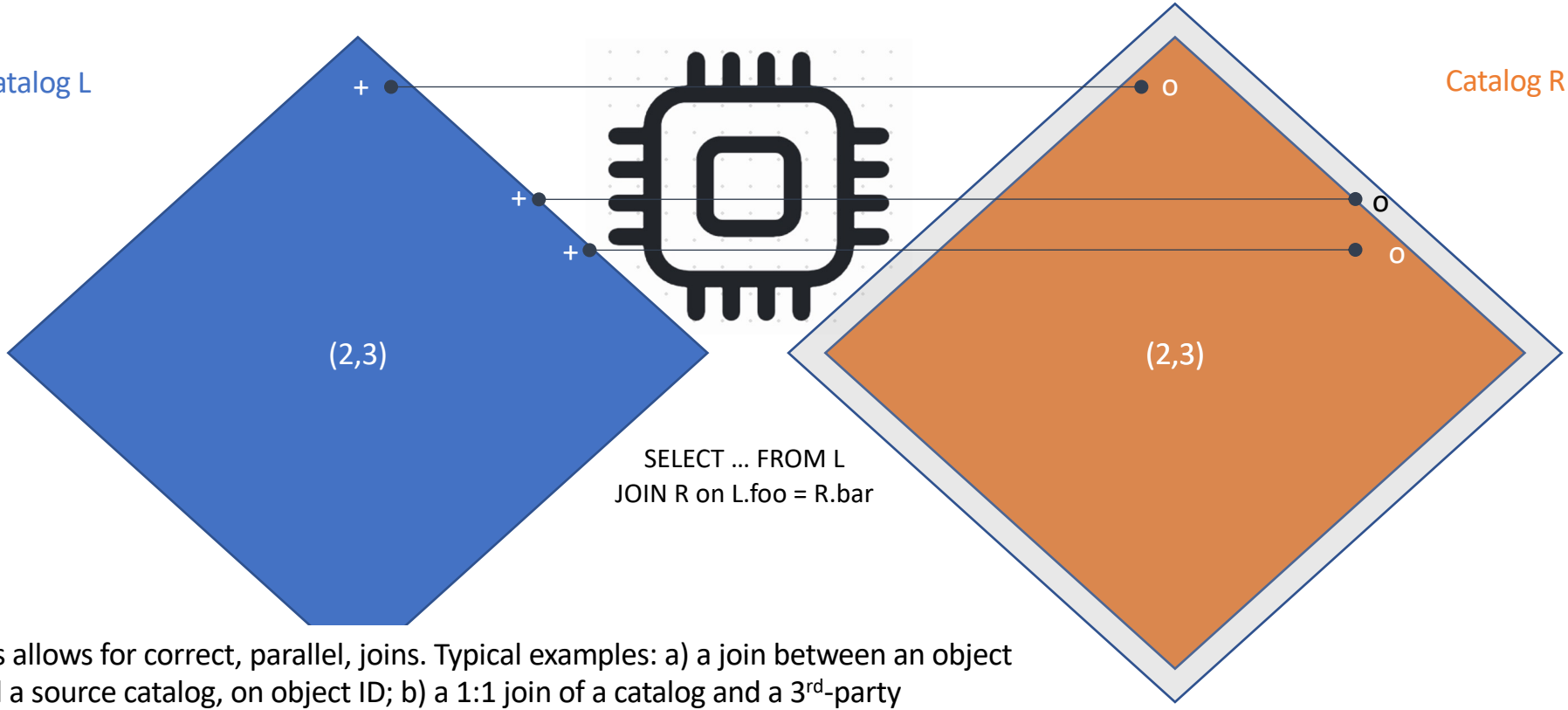
Crossmatching algo: Fetch the coordinate data from both partitions. Make sure to also download the padding of the partition to the right. For each row on the left, find nearest neighbor(s) in the table to the right. In some cases, the NN can be right across the partition boundary, and thanks to the padding, it will be found.



This allows for correct, parallel, N-nearest neighbor cross-matches, within a radius r, where r < width of the padding margin.

How distributed joins work

Joining on **co-partitioned columns** just works(™). Co-partitioned columns are columns for whose values it's guaranteed they will *all* be in the same partitions (plus margins) on the left and right side of the join. For such columns, a global "SELECT … FROM L JOIN R ON L.foo = R.foo" type query can be executed on a per-partition basis.



This allows for correct, parallel, joins. Typical examples: a) a join between an object and a source catalog, on object ID; b) a 1:1 join of a catalog and a 3rd-party produced catalog of added-value columns (e.g., object classifications).