Processing Every Image: Data Release Production at Rubin Observatory

Colin Slater
Rubin Data Management

Rubin Observatory

3.2 GPix camera, taking a new image every ~40 seconds, all night. 600-800 images per night = ~10 TB/night raw data.

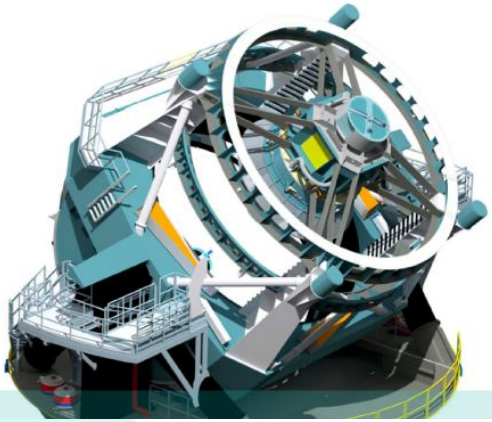
We took the first images with the full camera in April 2025. Since then, we have been commissioning and ramping up in prep for start of the official survey (LSST).

Right now, collected data volume is testing our ability to scale; we've reached the hard phase for data management.

Two different timescales

Raw Data: 10TB/night

Sequential 30s images covering the entire visible sky every few days



Prompt Data Products

- Alerts incl. science, template and difference image cutouts

60s via Alert Streams

- Catalogs of detections incl. difference images, transient, variable & solar system sources

24h via Prompt Products

- Raw & processed visit images (PVIs), difference images

80h via Image Services

Data Release Data Products

Final 10yr Data Release:

- Images: 5.5 million x 3.2 Gpixels
- Catalog: 15PB, 37 billion objects

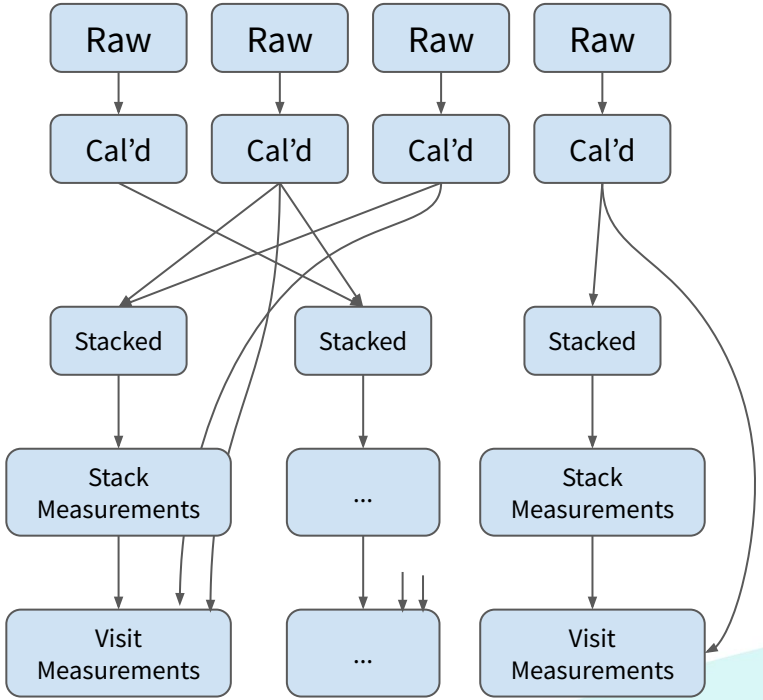
via Data Releases

Data Release Goals

Annually, we will publish our best dataset for the users, incorporating **everything** we've observed to date, and taking ~6 months to process.

This data release production requires:

- Calibrating every single image
- Combining all images that overlap a given patch of sky ("stacking")
- Measuring every object on that combined image
- THEN going back to each individual image to measure the history of each object.

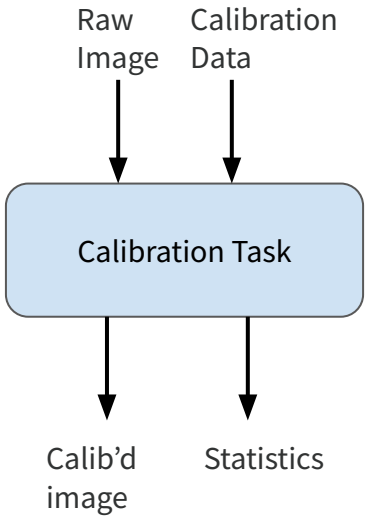


Processing Abstractions

This structure is clearly a DAG. Key step was separating the algorithmic code from the DAG-construction; all our algorithm developers need to do is declare their inputs and outputs, and our middleware inspects that + available data to build the DAG.

We add to rearrange tasks *so much* during development — being able seamlessly accommodate new DAG organization has been a huge win for us.

Made a big effort to NOT bake in anything Rubin-specific; two other observatories are using the same system today.

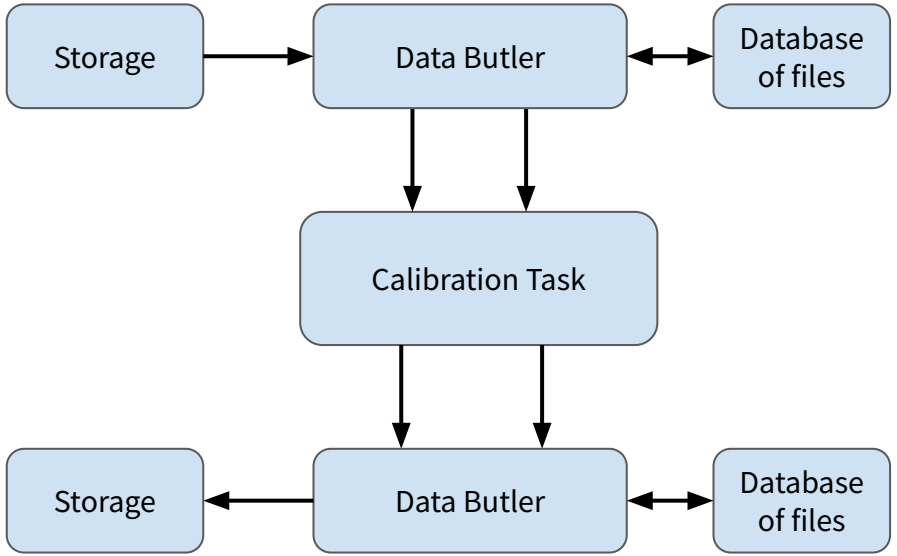


IO Abstraction

We also abstracted IO away from algorithmic code from day 1.

Algorithm developers generally don't have to think about how the data are stored or how the IO is performed.

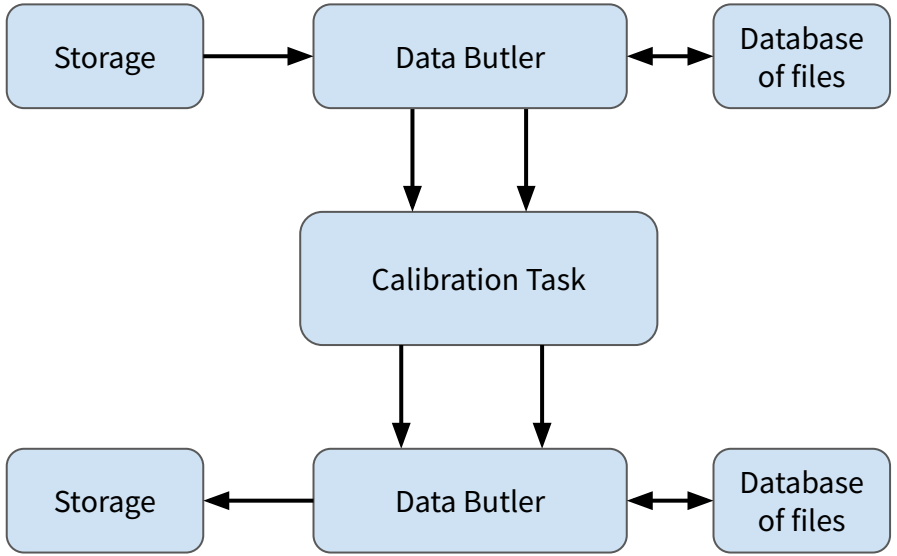
Corollary: All communication between DAG nodes is via long-lived files on disk.



Scheduling Granularity

These algorithmic “Tasks” are also the level of granularity for:

- Job Scheduling & Parallelism (one Task=one core, but we can also cluster Tasks to look like a single HTCondor Job)
- Data storage – Any output file is the result if a single task
- Monitoring – We get really great data on workflow status and performance via HTCondor

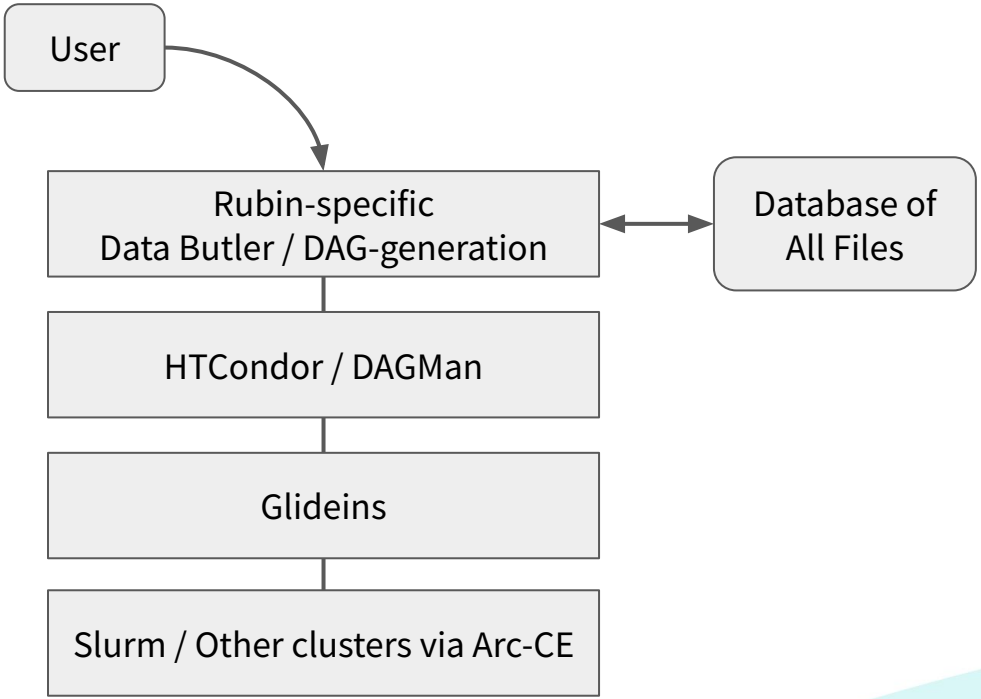


Processing Stack

The DAG then gets passed to HTCondor for execution.

We don’t actually have a “bare metal” HTCondor cluster; we rely on Glideins running on datacenter-provided Slurm clusters. This works well!

More details about implementation in Greg Daues’s talk in the next session.

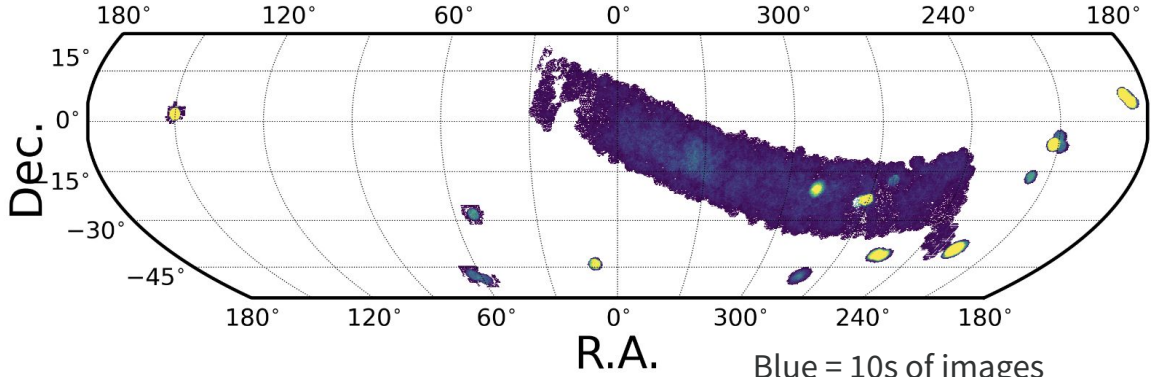


What’s difficult

Rubin went on sky April 2025, we were in great shape to process the data, we got through the early rush of algorithmic fixes to handle the new camera.

Started processing for our first “preview” data release in December:

- 30k images.
- 3M core-hours.
- 120TB input data,
- ~320TB of processed data
- Along the way, generated ~7+ PB of intermediate data products, >100M files.



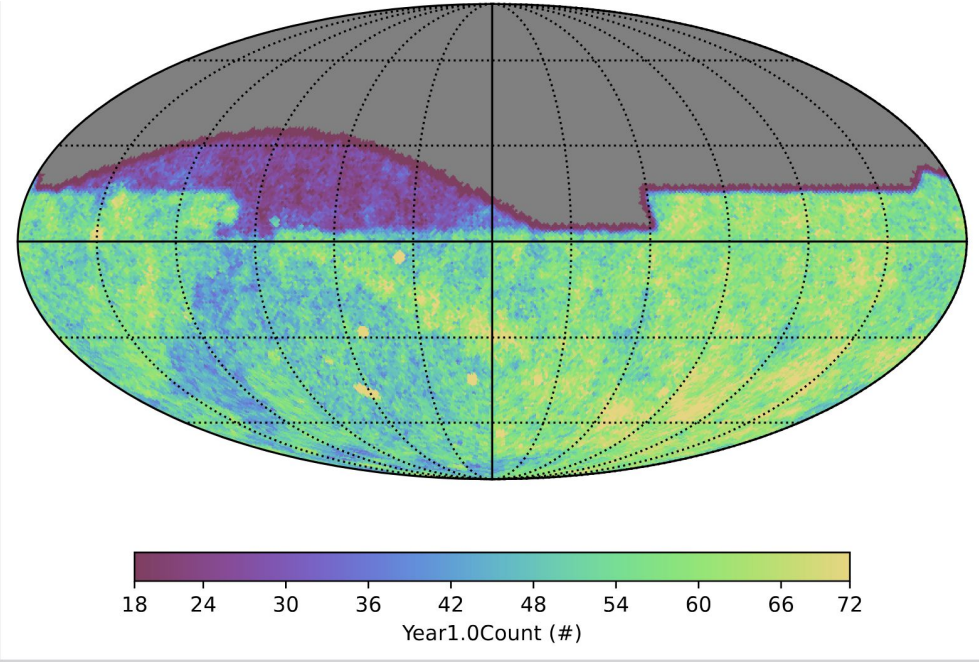
Blue = 10s of images
Yellow = 100s of images

What’s difficult (2)

Our first “Full” data release will be ~5-10x the size of the preview.

- 0.7 PB input
- 1.6 PB Output
- 47 PB intermediates – !?!

Our second data release is another year of observing, 2x the input! Even more the next year!



Storage is difficult

- We have previously treated disk space as a “free” resource for intermediate products while focusing on algorithm development – clearly that era is over.
- The “preview” processing did stress our storage systems, learned a lot from this.
- We accidentally filled up our ~30PB storage cluster, which halted processing for >a week.
- Turns out it is a lot harder to delete a few PB of data than we thought! Deletions have their own set of performance bottlenecks.

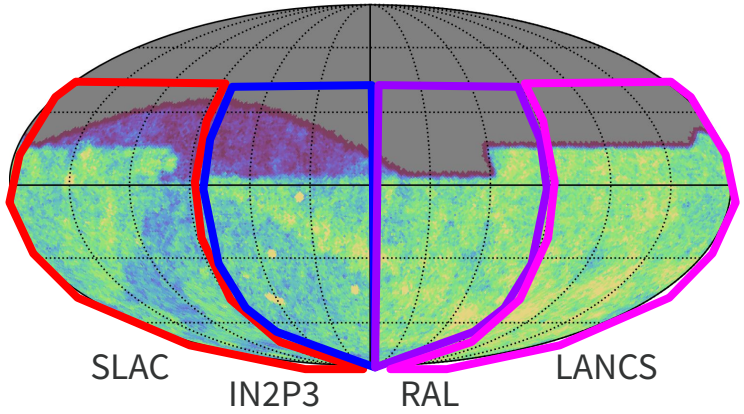
Where we're headed next

For this next data release, we are relying on four different compute sites to handle the load: SLAC (California), IN2P3 (Lyon), Rutherford-Appleton Lab, Lancaster (Both UK).

We're currently running HTCondor jobs at all sites from a central Access Point at SLAC. Works well!

The challenging part is data movement; we can *mostly* partition the sky so that each site operates on an independent set of images, but we expect to need more ad hoc flexibility too.

Made up example partitioning:



Open questions

Choices that we're still grappling with, would love to hear how other projects have addressed these:

- Centralize the knowledge of what data are where? Or decentralize so each site can operate independently?
- How to manage the tradeoff between granular data access vs. bookkeeping costs of 100M+ files
- Pre-stage data that each compute site will need, or transfer data just-in-time to start each compute job?
- How do we reduce the overall burden on the storage system?

Thanks from Rubin

HTCondor processed this data