FTS in Rubin Data Processing and Distribution

Wei Yang

Quick Intro To Vera Rubin Observatory and LSST

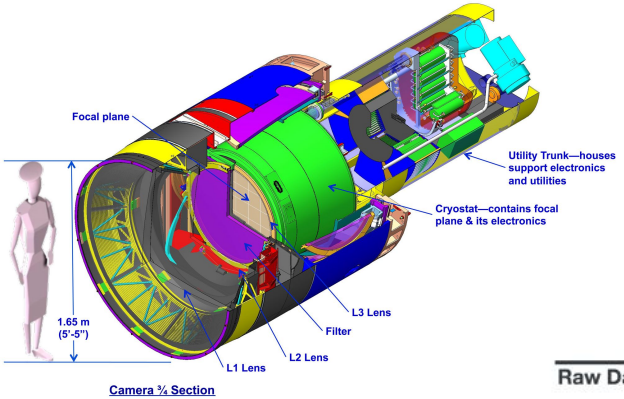
Legacy Survey of Space and Time (LSST), at the Vera Rubin Observatory:

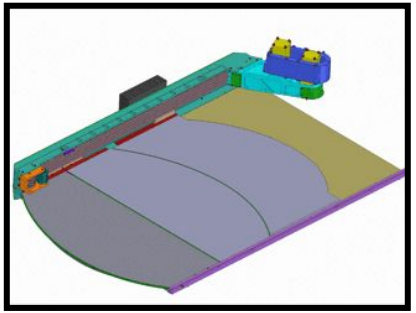
- Science themes: dark energy, dark matter, but also solar system and transient objects
- 8.4m telescope at Cerro Pachon (Chile)
- 3.2 Gpix camera
- All visible sky (~18000 deg2) in 6 bands
- 10 years survey starting from 2025

Hernandez et al. CHEP 2023

southern

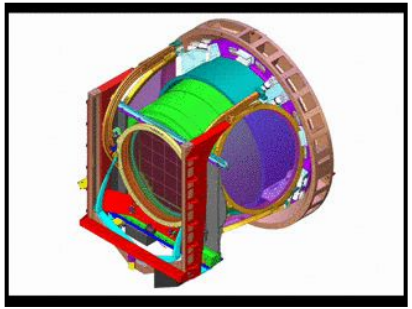
Camera, CCD, Shutter, Filters, Fast/Slow Processing

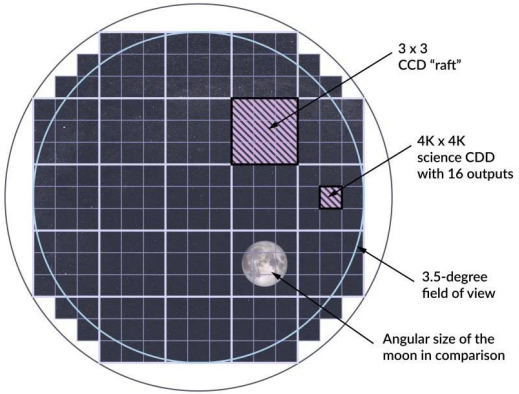




Shutter

EM-ware filters (bands)





Raw Data: 20TB/night

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

LSST SCIENCE PIPELINES

Prompt Data Products

Alerts: up to 10 million per night — 60s — via nightly alert streams

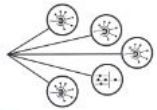
Results of Difference Image Analysis (DIA): transient and variable sources

Solar System Objects: ~ 6 million — 24h — via Prompt Products Database

Data Release Data Products

Final 10yr Data Release:
- Images: 5.5 million x 3.2 Gpx
- Catalog: 15PB, 37 billion objects

via Data Releases



Community Brokers

Alert Filtering Service

Rubin DACs (DFs & Chile)

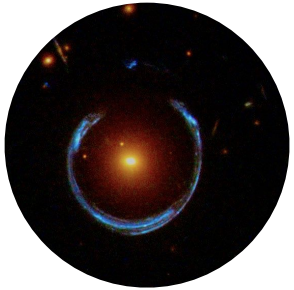
Independent DACs (iDACs)

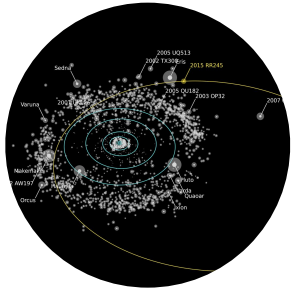
Key Science Drivers

Ivezic et al. 2019

Dark Matter, Dark Energy

- Weak Lensing
- Baryon acoustic oscillations
- Supernovae, Strong Lenses



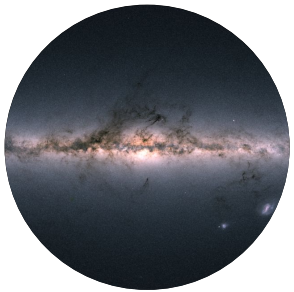


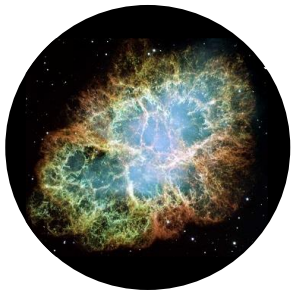
Cataloging the Solar System

- Potentially Hazardous Asteroids
- Near Earth Objects
- Object inventory of the Solar System

Milky Way Structure & Formation

- Structure and evolutionary history
- Spatial maps of stellar characteristics
- Reach well into the halo



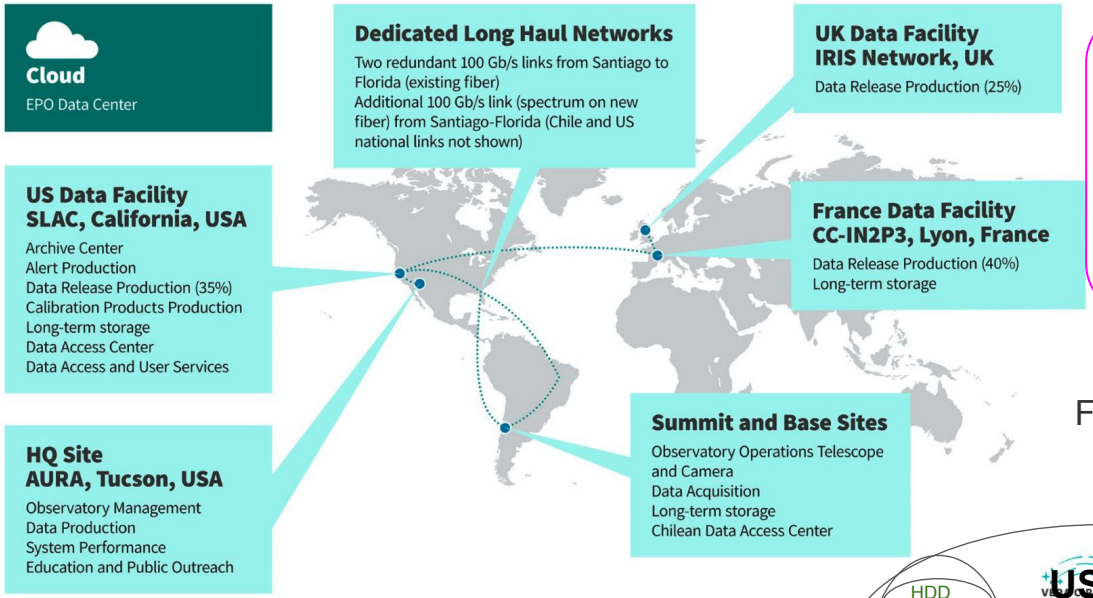


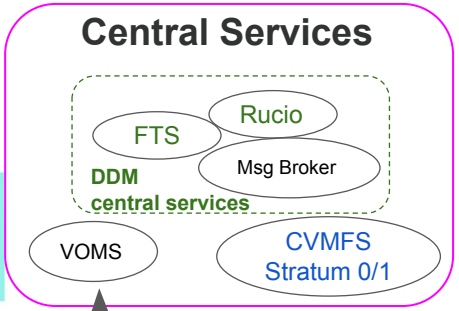
Exploring the Transient sky

- Variable stars, Supernovae
- Fill in the variability phase-space
- Discovery of new classes of transients

Rubin is a facility providing processed data to 8 independent science collaborations

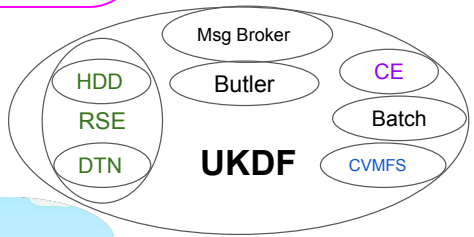
Rubin Data Facilities and Services

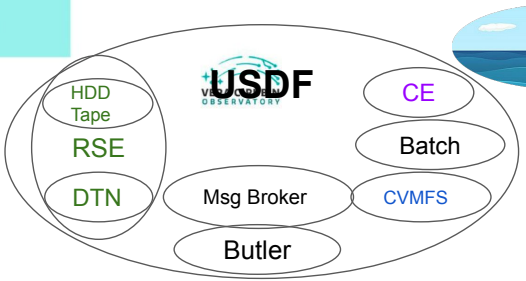


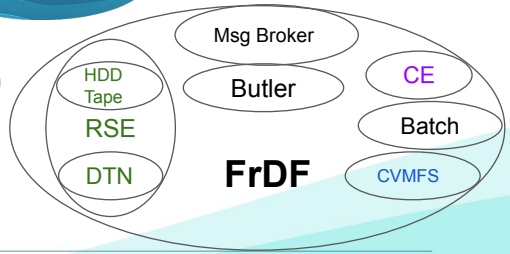


iDACs around the world with RSEs

Future: Indigo IAM







1. Rubin offline Data Release Production happens at its Data Facilities.
2. Data Products are distributed to iDACS

Both use tools from the Rucio/FTS/Xrootd/dCache

Butler: Rubin's Native Data Management Tool

It is easier to explain what Butler does by looking at the info it stores.

```
exposure:
  instrument: 'LSSTCam'
  id: 2026060700080
  day_obs: 20260607
  group: '2026-06-07T23:03:13.359'
  physical_filter: 'i_39'
  obs_id: 'MC_O_20260607_000080'
  exposure_time: 30.0
  dark_time: 30.9309
  observation_type: 'science'
  observation_reason: 'pairs_ri_15.0'
  seq_num: 80
  seq_start: 80
  seq_end: 80
  target_name: 'lowdust'
  science_program: 'BLOCK-407'
  tracking_ra: 166.21248895105293
  tracking_dec: 6.135042124461654
  sky_angle: 32.744815278944394
  azimuth: 351.7390465215113
  zenith_angle: 36.5078205626525
  has_simulated: False
  can_see_sky: True
  timespan: Timespan(begin=astropy.time.Time(2461199.0, 0.4612666340544791, scale='tai',
format='jd'), end=astropy.time.Time(2461199.0, 0.4616246296296297, scale='tai', format='jd'))
```

Butler consists of a DB and API to access the DB.

- Info in the DB helps API users to quickly find the actual image files based on the science selection criteria.
- A critical component of the LSST science pipeline software.
- Outputs are registered to Butler when produced. But not to Rucio

No running services

- No concept of multiple copies
- No service to move data.
- Very little functionality overlapping with Rucio / FTS


```
{
  "id": "409312c6-6af5-5be5-8442-e78e52520701",
  "datasetType": {
    "name": "raw",
    "storageClass": "Exposure",
    "dimensions": [
      "instrument",
      "detector",
      "exposure"
    ]
  },
  "dataId": {
    "dataId": {
      "instrument": "LSSTCam",
      "detector": 0,
      "exposure": 2026060700080
    }
  },
  "run": "LSSTCam/raw/all"
}
```

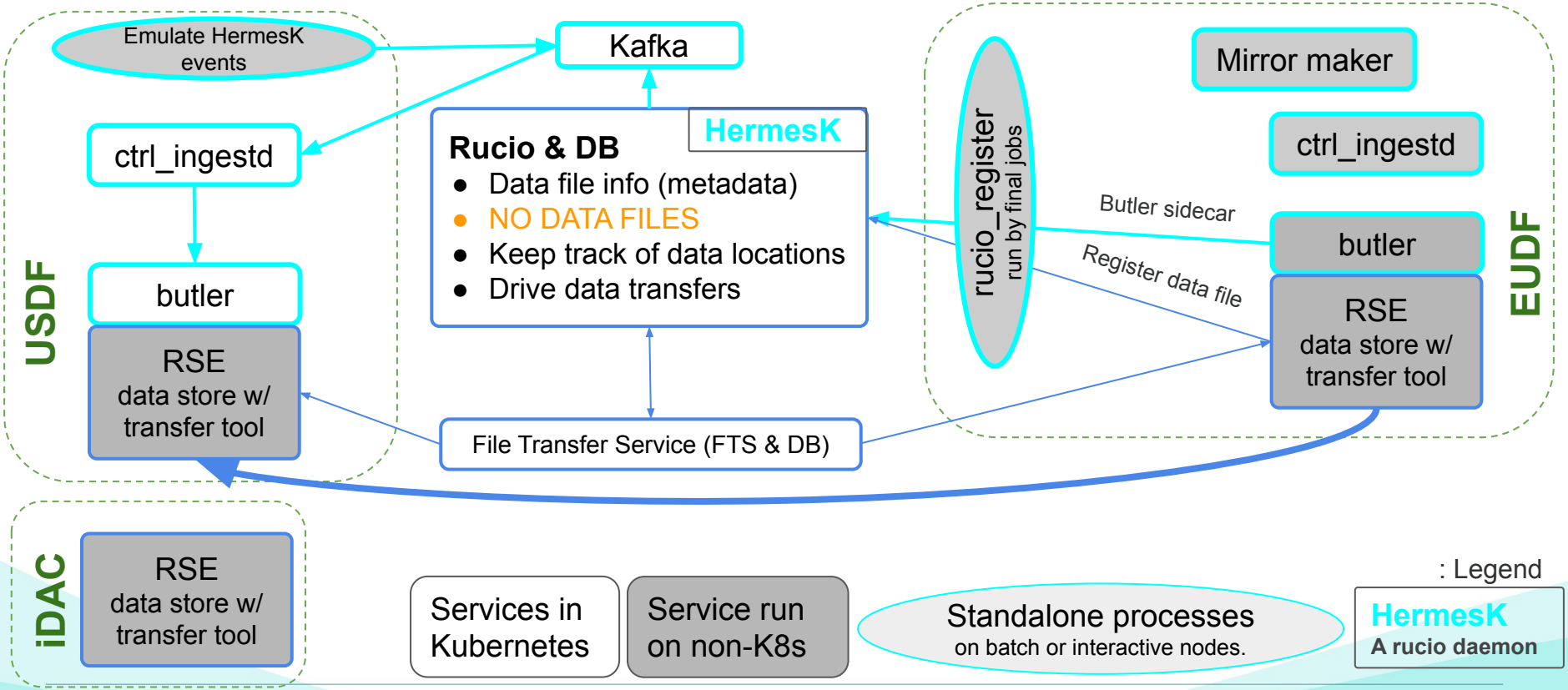

Rubin glues Butler/Rucio together using

- A set of naming conventions
- A set of services

/sdf/data/rubin/…/MC_O_20260607_000080.zip#zip-path=MC_O_20260607_000080_R01_S00.fits

Multi-site data movement

Butler dataset – Rucio file
Butler run collection – Rucio dataset
Butler chain collection – Rucio containers



FTS 3 service at the US Data Facility

Two FTS installations, a prod & a dev, in Kubernetes, based on work done at FNAL (So is Rucio setup)
Rubin also have access to RAL FTS3

Set pod CPU, RAM and maximum number of fts_url_copy based on lesson learned in operation

```
$ kubectl --context usdf-fts3 -n fts3 get pod
NAME                                READY   STATUS    RESTARTS        AGE
fts3-activemq-667989f59c-szxxk      1/1     Running   13 (34d ago)    44d
fts3-mon-7989476bc9-sq6w9           4/4     Running   0               44d
fts3-rest-fd799b6b9-7mb6c           4/4     Running   5 (21d ago)     43d
fts3-server-5dd5967f7-9t6bv         1/1     Running   0               40d
fts3-server-5dd5967f7-hvnt8         1/1     Running   0               43d
fts3-server-5dd5967f7-wxsrt         1/1     Running   0               43d
```

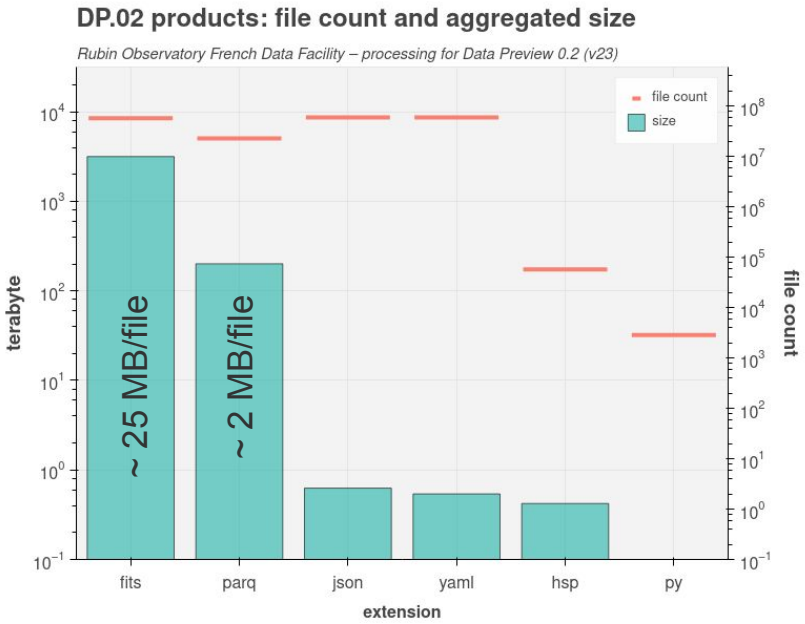
← Easy to scale up

```
$ kubectl --context usdf-fts3 -n fts3-db get pod
NAME                      READY   STATUS    RESTARTS   AGE
fts3-db-cluster-solo-0    1/1     Running   0          38d
```

Logs:

- /var/log (transfer FTS logs) and /var/lib/fts3/messages: two PVCs shared by all pods
- FTS services write logs to STDOUT, and send to Loki

Rubin's requirement and benchmark



Many small files: DP0.2 was simulated data but still reflect the real data file count and size distribution from the LSST

Files per hour and **Files per day** are better data transfer benchmark than GB/s.

Two data transfer scenarios:

- Distributed Data Release Production (DRP) at data facilities
  - DP2 (8-mo survey) has 1.1 billion files
  - In one extreme case, we may be asked to transfer 50 million files per day.
  - Optimizations are possible to reduce this number, e.g. not transfer intermediates.
  - But the tone is already set. We need to vastly increase the transfer rate (from the current 3-5 million file / day)
- Data products distribution to independent Data Access Centers (iDACs)
  - Data products total size greatly reduced (by at least 50%) when intermediates are excluded.
  - But we need to distribute to ≥ 10 iDACs

Lessons from transferring FrDF Pilot Run to USDF

- Setting: Transfer ~22.8 million small files / 256.7 TB from FrDF to USDF
  - A typical DRP data transfer scenario (but include all intermediates)
  - Average file size ~12MB
  - Many lesson learned: register data to Rucio; Rucio/FTS driven transfer; ingest data to destination butler. **Only listing FTS related lessons here**:
- Factors that increase the FTS performance
  - Increase the usage of sessionReuse
    - Both in /etc/fts3/fts3restconfig and in Rucio bulk submission size
    - (even with this, per file authentication overhead is still way to high)
  - Need to throttle the number of submitted jobs by Rucio to FTS ← seems to be a known issue
  - Some DB settings MAY help
    - Turn innodb_buffer_pool_size and innodb_flush_log_at_trx_commit
- Factors that improve reliability
  - Balance the CPU/RAM and the number of fts_url_copy
    - Otherwise FTS could sometimes loss track of completed transfer jobs.
  - Separate services into their own pods – easier to scale up
    - Single, all-in-one FTS pod would share resources negatively impact fts-mon and -rest.
- Other findings:
  - Later, we started to use ActiveMQ. It was reliable for a while but then we see memory leak.

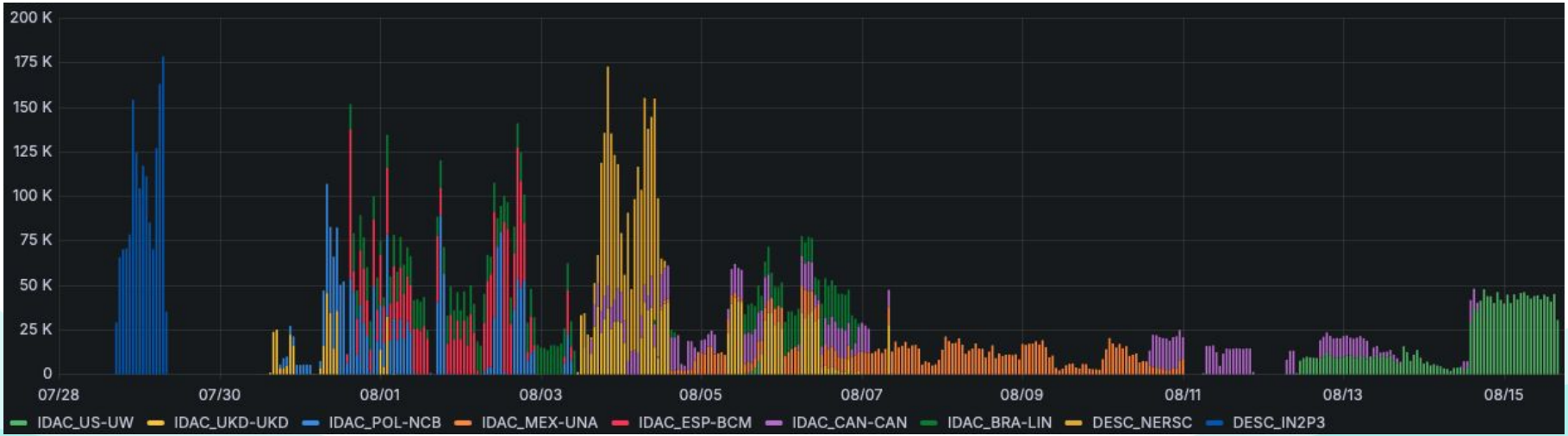
Improvement reflected in later smooth operation

Source	Destination	VO	Submitted	Active	Staging	S.Active	Archiving	Finished	Failed	Cancel	1h)	Thr.
davs://usdf-xrootd.slac.stanford.edu	davs://ccdavrubin.in2p3.fr	lsst	503	357	–	–	–	214146	14	–	99.99 %	573.73 MiB/s
			503	357	0	0	0	214146	14	0	99.99 %	–

⬆️ USDF to FrDF transfer of the same data (FrDF pilot run), > 5 million files / day, ~ 12MB/file

⬇️ Early LSST Data Preview 2 (DP2) distribution to 9 iDACs ← slowdown not caused by USDF

- Early DP2 is only 1.5 million files (~39MB/file). Full DP2 (end of 2026) will be 4-6x
- Data Release 1 (DR1) will be several times bigger than DP2.



Things that are critical and challenging to Rubin

1. Number of concurrent fts_url_copy
   - This is needed to transfer large amounts of small files
   - Affected by many other factors: number of fts-server pods, PVC IO, message bus performance, Rucio state update efficiency (conveyor-finisher)
2. Efficient transfer of small files
   - This is also needed to transfer large amounts of small files
   - **sessionReuse is essential. But per file authz is a huge waste**
     - increased the AutoSessionReuse MaxSmallFileSize
   - Rubin is working with the FTS team to find a sessionReuse replacement in FTS 4, that also don't do per file authz.
3. Resilient TPC
   - HTTP TPC is great when it works ! But is too easy to be affected by external factors
   - Latest huge memory leak happened during Rubin early DP2 distribution.
   - Lucky Xrootd TPC still works ← or we will be looking for alternatives to Rucio/FTS
   - So we still need Xrootd TPC in FTS 4
4. Coarse grained token authorization
   - Can we not have to exchange two tokens from IAM to Rucio to FTS to RSE for each file ?
   - Rubin is setting up an IAM to find out.

Conclusion

Rubin is using Rucio/FTS/Xrootd/dCache eco-system tools for production

- Probably one of the first astronomical observatories doing this
- Manys are wondering if the HEP tools will work well with the astronomy community's software stack, workflow, data characteristics and **culture**

We passed the first small scale real world test (Early DP2 distribution)
Likely will pass the second, 4-6x larger real world test (Full DP2 distribution)
Will FTS3 be stretched to handle DR1 scale DRP and distribution ?

Moving forward with Tokens, FTS4, TPC, we see challenges that need to be addressed.