CSD3 to Echo S3 Data Transfer

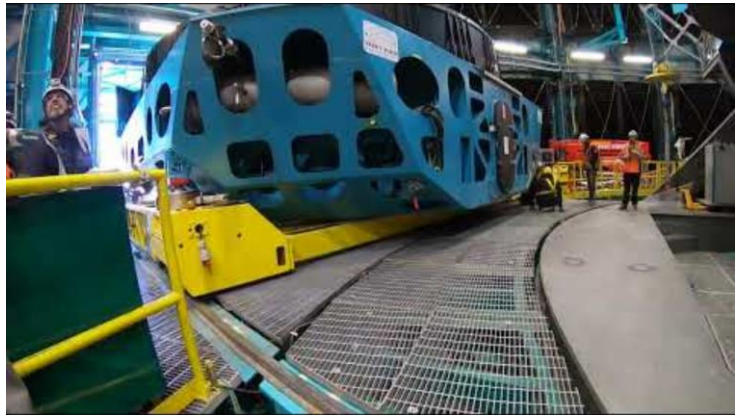
Curation of LSST:UK DEV Outputs

csd3-echo-somerville

Dr Dave McKay, Architect, EPCC

The Vera C. Rubin Telescope and LSST

- Located in Cerro Pachón, Chile
- 8.4 m Simonyi reflecting telescope
- 3.2 gigapixel CCD – the world's largest digital camera – LSSTCam
- Legacy Survey of Space and Time (LSST)
  - conceived in 2001
  - 10-year survey starting early 2025
  - covering 20,000 sq.-deg. in u,g,r,i,z,y bands
  - observing each field 800+ times during survey
  - producing 20 TB data per night
  - around 10M alerts per night
- Has reached Commissioning Phase and begun taking calibration images with smaller ComCam CCD

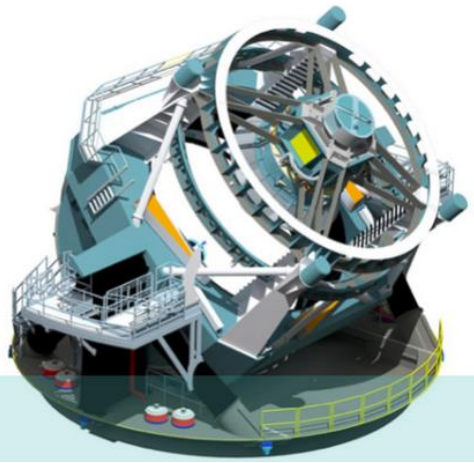


24th October 2024

The Vera C. Rubin Telescope and LSST

Raw Data: 20TB/night

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



LSST SCIENCE PIPELINES

Prompt Data Products

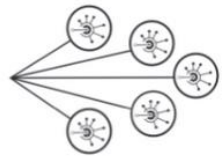
- Alerts incl. science, template and difference image cutouts — 60s — via Alert Streams
- Catalogs of detections incl. difference images, transient, variable & solar system sources — 24h — via Prompt Products
- Raw & processed visit images (PVIs), difference images — 80h — via Image Services

Data Release Data Products

Final 10yr Data Release:

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

via Data Releases



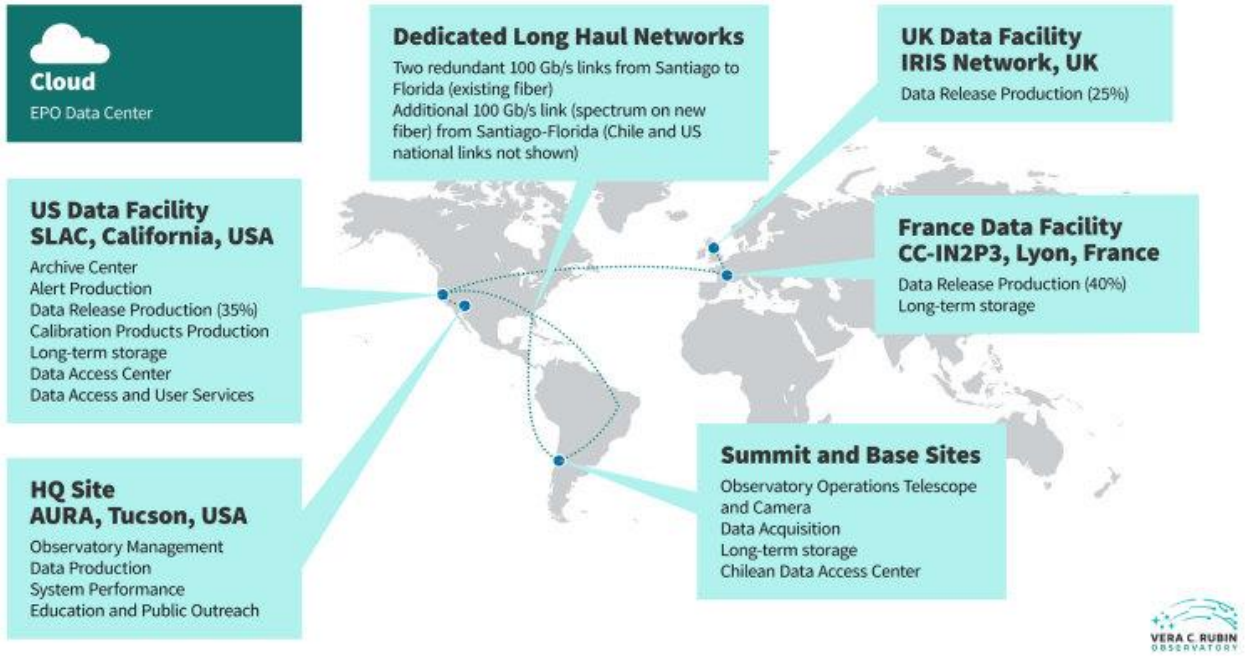
Community Brokers

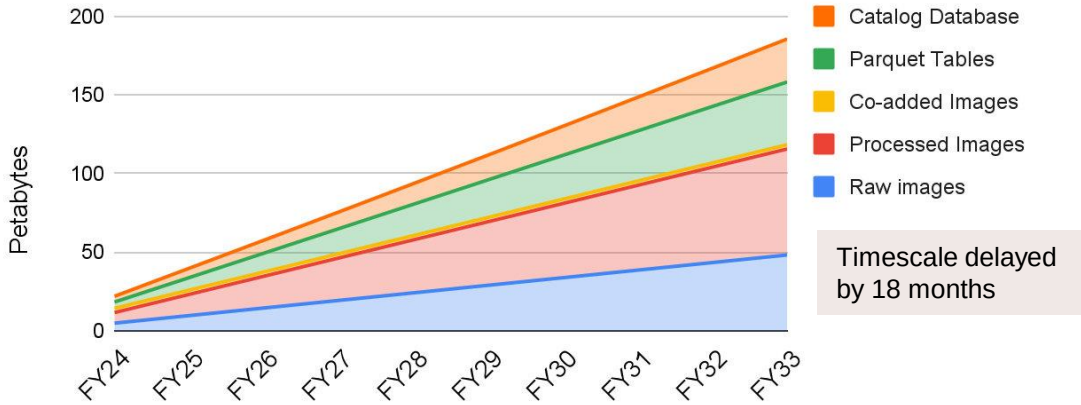
Rubin Data Access Centres (DACs)

USA (USDF)
Chile (CLDF)
France (FRDF)
United Kingdom (UKDF)

Independent Data Access Centers (IDACs)

The Vera C. Rubin Telescope and LSST





Timescale delayed by 18 months

LSST:UK – 16 in-kind Contributions – IDAC, DRP and DEV

DRP

- The UK's contribution to Data Release Processing (25% of all LSST data) is performed at Lancaster University and RAL

IDAC

- The UK's Independent Data Access Centre
  - hosted on Somerville
  - provides an RSP
  - plus data and services from in-kind contributions including Lasair Community Broker, Near-IR Data Fusion, multi-survey crossmatch

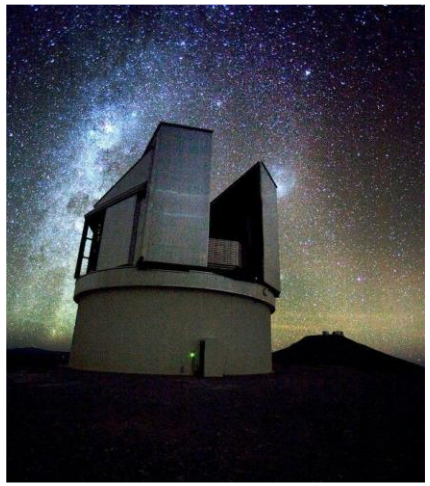
DEV

- Develop software (to work with LSST Science Pipeline) and services for UK priority science areas
- Produce the in-kind contribution data for the IDAC

LSST:UK Consortium

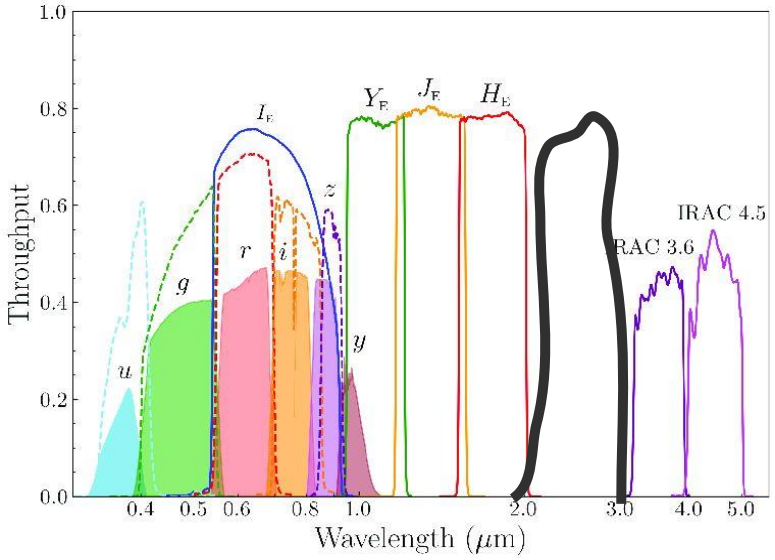
epcc

LSST:UK – DAC-DEV for LSST Near-IR Data Fusion



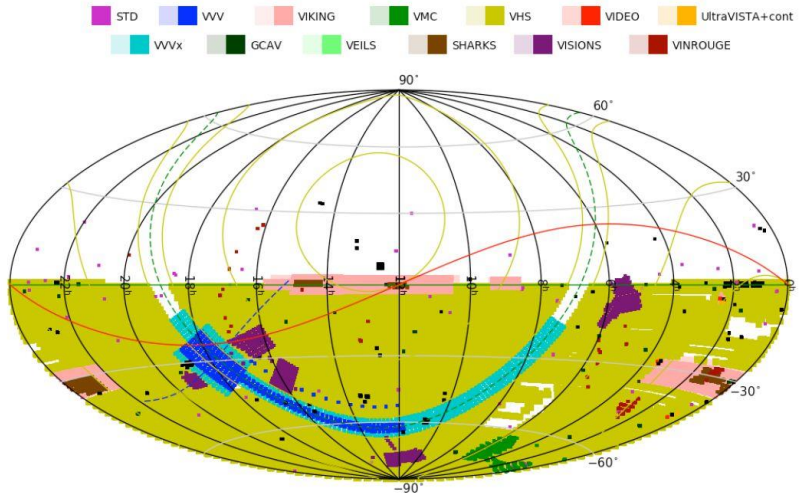
VISTA

- wide-field reflecting telescope with a 4.1m mirror, located in Antofagasta, Chile
- fitted with a near-IR camera producing 67 MP images



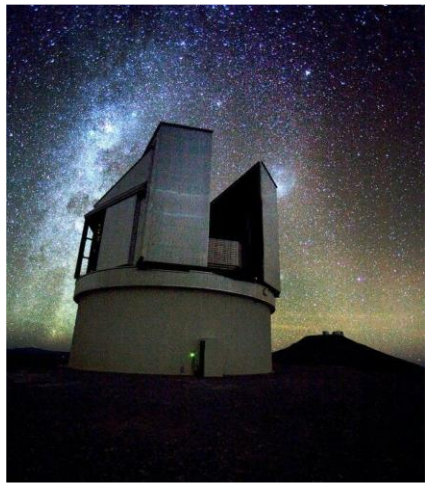
LSST and HSC: optical
VISTA: Z, Y, J, H and K_s bands

VHS	20000	20
VIKING	1500	21.2
VVV	520	19.9
SHARKS	300	22.7
VMC	184	22.1
KEDFS	20	23.5
VIDEO	12	23.5
VEILS	9	23.3
UltraVista	0.73	25.6



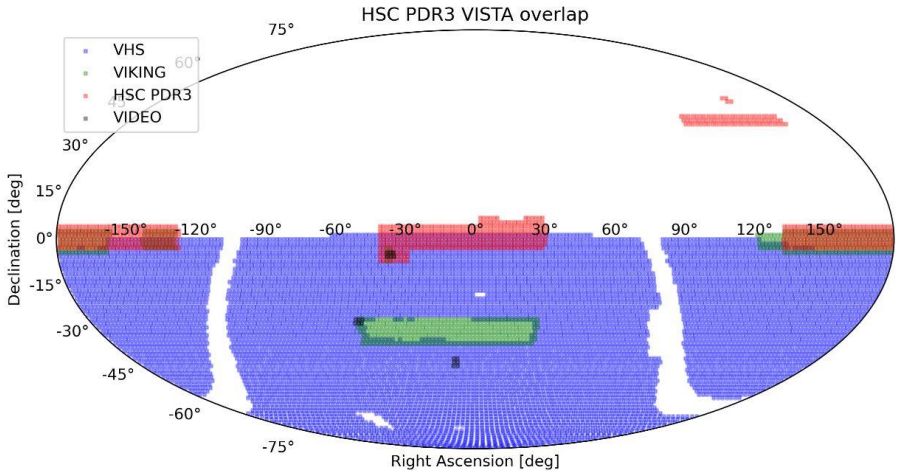
University of Southampton: Elham Saremi, Raphael Shirley, Shenli Tang, Manda Banerji

LSST:UK – DAC-DEV for LSST Near-IR Data Fusion



VISTA/HSC Survey Overlap

- since no LSST data is available yet, HSC data can be used to develop data fusion
- the LSST Science Pipeline allows “obs” packages to be added so other surveys can be processed together with HSC data, for which an obs package exists
- VISTA obs package:
  - `github.com/lsst-uk/obs_vista`
- processing repo:
  - `github.com/lsst-uk/lsst-ir-fusion`
- data fusion products to be ingested into IDAC
- products and runs to be backed up to RAL Echo S3
  - so far using ~700 TB storage on CSD3



Red squares: HSC D/UD patches
Blue squares: VIDEO patches

University of Southampton: Elham Saremi, Raphael Shirley, Shenli Tang, Manda Banerji

The Data “Butler”

Butler

- part of the LSST Stack
- bespoke database of observations
- used by data processing (DRP and derived products), data transfer (Rucio policy-driven data synchronisation – Chile/US/UK/France, then beyond), data access services (all DACs)

Butler datasets (on CSD3)

- large total size (100s of TB)
- millions of small files (20 KB ~ 100 MB)
- deep folder structures (around 3 × more folders than files)

(Near-IR Data Fusion) User working data

- as above, but also occasional large individual files (100s of GB)

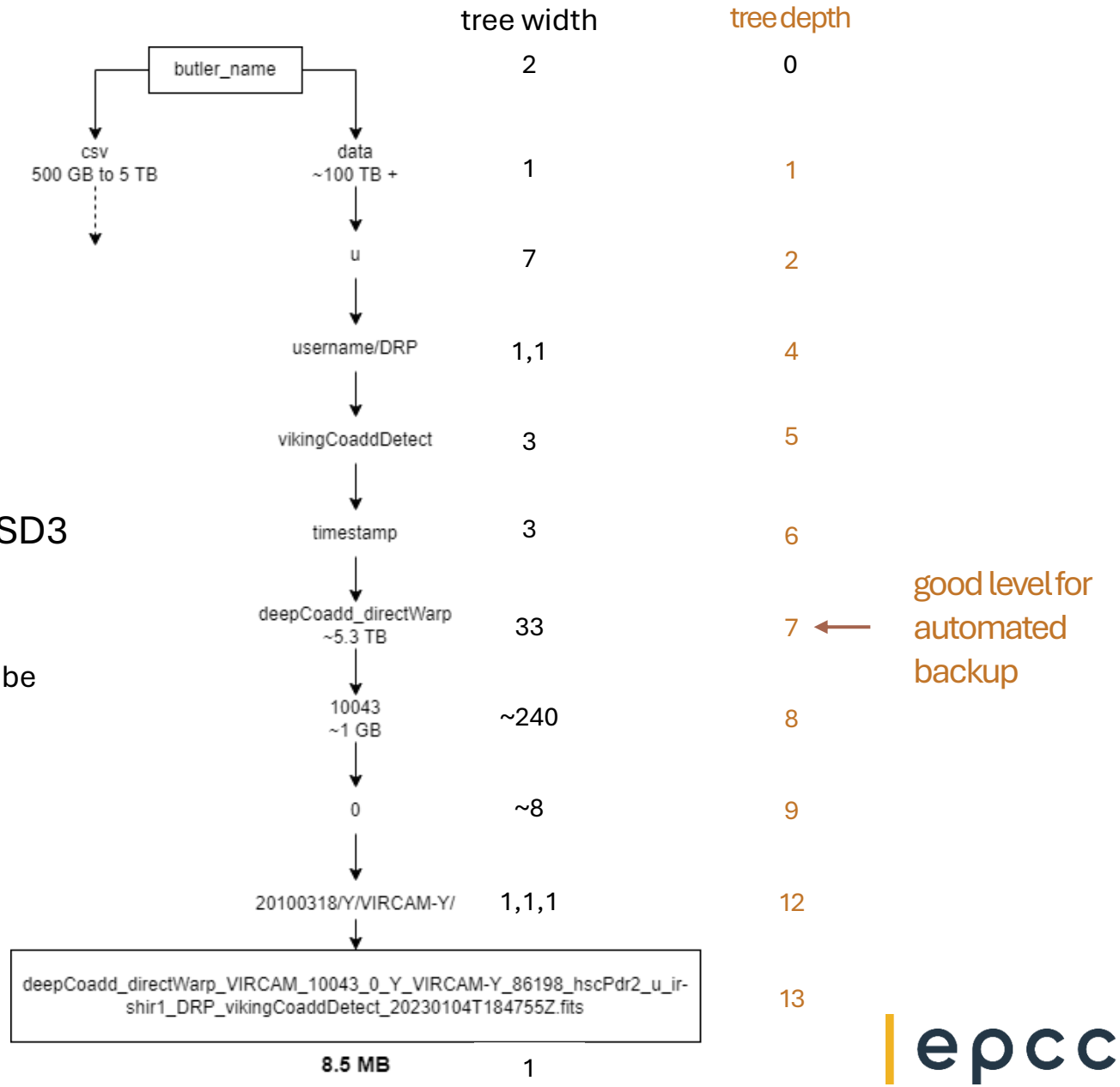
Butler Datasets

Butler dataset directory tree structure

- example file (taken at random)
- 8.5 MB (if average(?), 12.3 million files)
- up to 13 folders deep, 240 wide

Other considerations

- a large amount of “data” in the Butler folders on CSD3 are symlinks to another filesystem
  - need to follow symlinks to produce complete Butler dataset
  - but need to retain knowledge of symlinks to allow dataset to be restored from backup
- file-count-limited processes are slow: du requires compute-node CPU time
- avoid hitting storage limit – avoid local I/O
- practical automated backup strategy



File Transfer

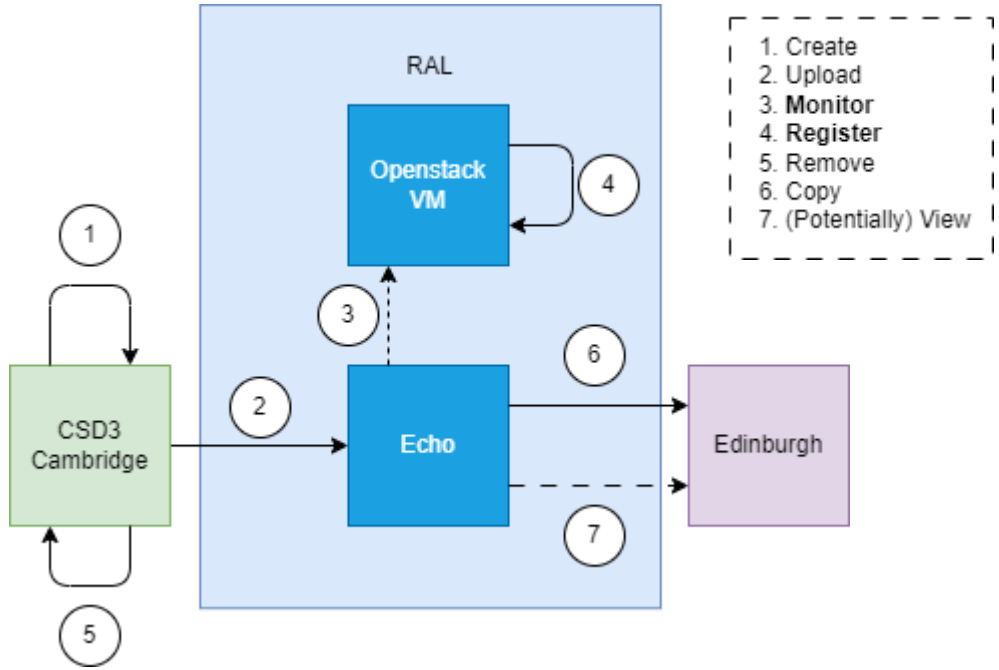
Why not use Rucio?

- Rucio used by the project generally – attempted for this data by Mathew Sims (STFC)
- however, ideally runs between Rucio endpoints (of which CSD3 is not one)
- expects dedicated, not shared system
- in sub-optimal environment, only Rucio upload client could be used
- registering and checksumming become blocking
- prohibitively slow, > 2s per file, full transfer would take months.

New solution

- combine “basic” upload from CSD3 with registering at RAL
- key outcome: general applicability to HPC → S3 transfer

1. Create (DEV activity)
2. Upload (not clever)
3. Monitor (VM local to Echo verifies uploads)
4. Register (Butler on the VM)
5. Remove (from CSD3); Echo holds gold copy
6. Copy, or
7. View data from DAC



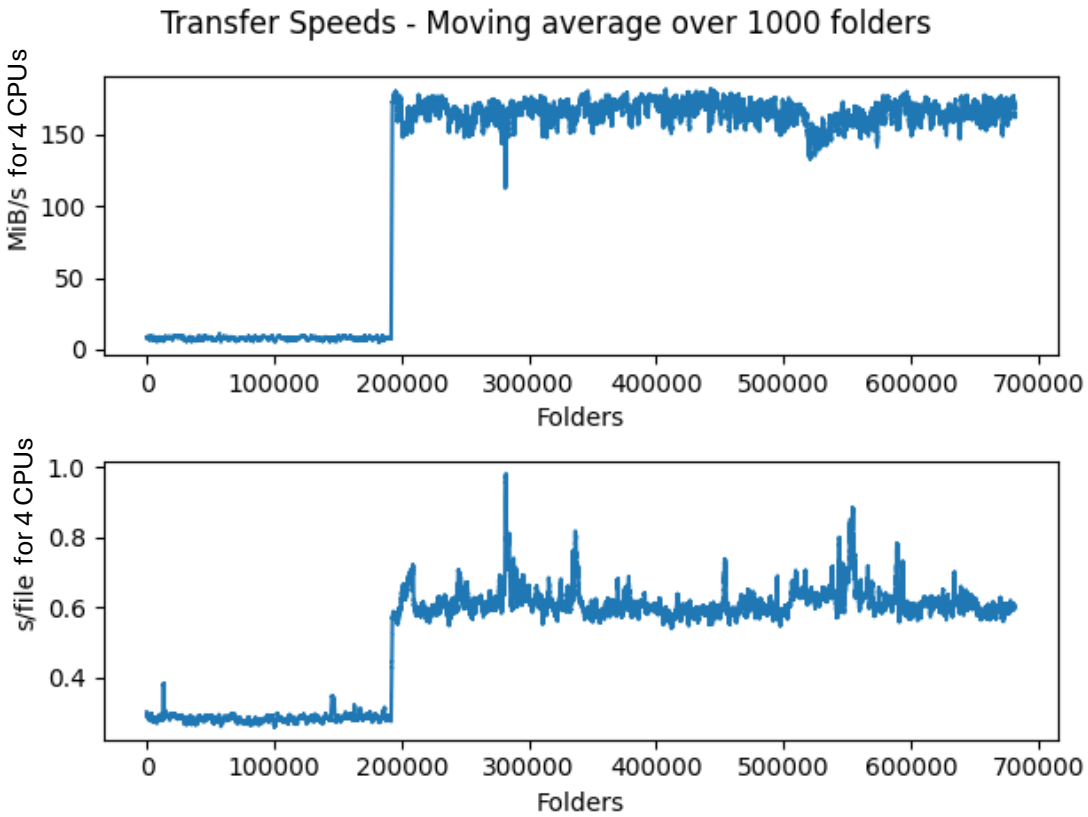
Transfer from Login Nodes

Python scripts

- written to use the AWS-developed boto3 library
- simple filesystem traversal
- upload each file to object with the absolute path as name
- fairly easy to make embarrassingly parallel

Other considerations

- required Watchdog exemption
- relatively low bandwidth to Internet
- keep CPU usage low (use ≤4 cores)
- keep memory usage low (how?)
- shared system – be `nice`



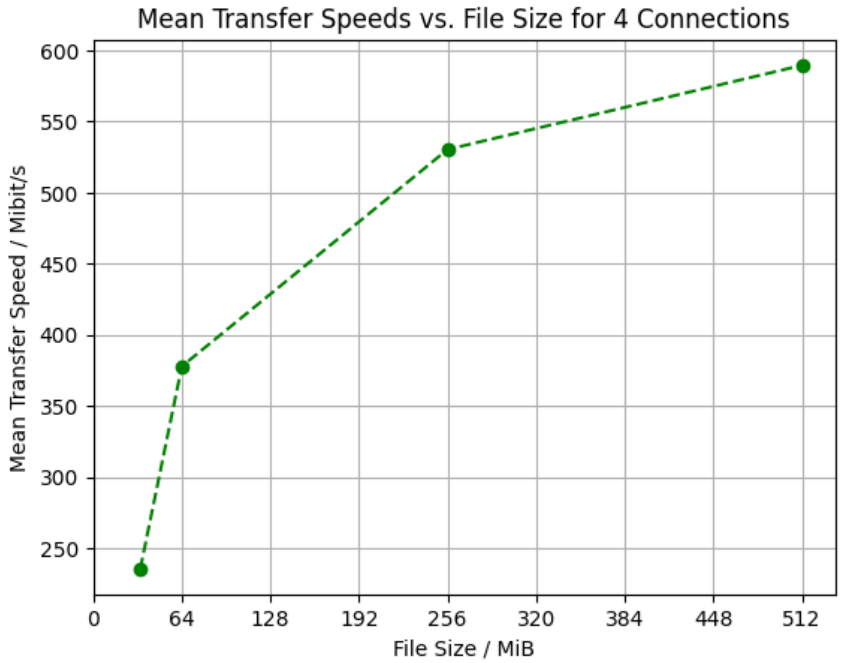
~ 160 MiB/s (1.2 Gbit/s) overall

~ 40 MiB/s (0.3 Gbit/s) per CPU

SKAO – UKSRC to the rescue!

UKSRC DTNs

- Offered testing use of DTN nodes at CSD3 funded by UKSRC – thanks Richard Hughes-Jones *et al.*
- 2 nodes with 100 Gbit Ethernet links from CSD3 to JANET
- Login-node-like Cascade Lake setup (56 cores, 192 GB RAM (3.4 GB/CPU), direct SSH, MFA)



Mean transfer speed per process for files of a given size, using 4 processes for upload.

- Test runs with dummy data
- High speeds of up to 600 Mibit/s per processes
- 2.3 Gbit/s overall
- But only for the largest files

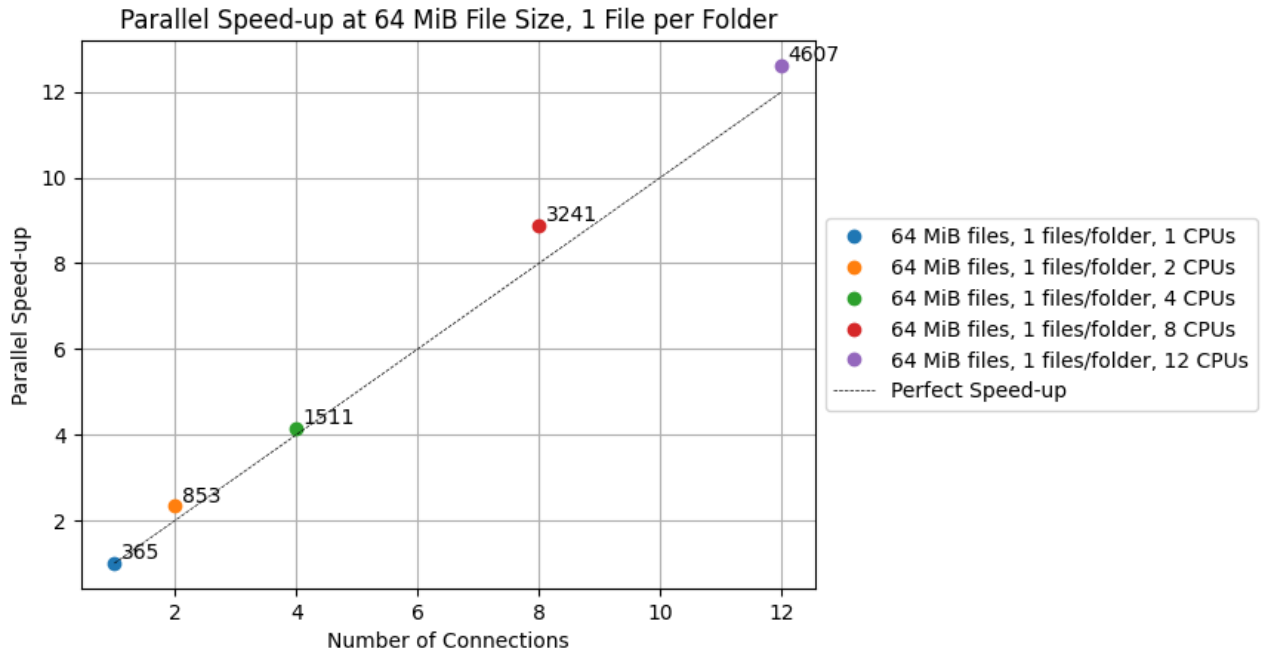
Note: timings include establishing a connection for each file being uploaded.

SKAO – UKSRC to the rescue!

UKSRC DTNs

- Offered testing use of DTN nodes at CSD3 funded by UKSRC – thanks Richard Hughes-Jones *et al.*
- 2 nodes with 100 Gbit Ethernet links from CSD3 to JANET
- Login-node-like Cascade Lake setup (56 cores, 192 GB RAM (3.4 GB/CPU), direct SSH, MFA)

- Parallel speed-up is good – linear from 1 to 12 CPUs given 64 MiB files



Ultimately, SMALL FILES TRANSFER SLOWLY!

- connection overhead begins to take longer than transfer
- solution – zip on-the-fly

Collating (zipping) small files

Collating small files

- Need to have fewer files to reduce per-file overhead
- Counterintuitively, we want files to be large, so they transfer more quickly – do not use compression
- We can't write to local disk – zips must fit in memory
- Aim: turn millions of small files into thousands of large files on the fly

Caveats

- The backup at Echo is no longer a mirror of the data on CSD3
- In addition to monitoring, VM at RAL will discover new ZIP files, extract and upload
- Need to check these – use S3 metadata to list zip file contents
  - but S3 metadata has a character limit!
  - generate `*.zip.metadata` objects where zip contents list are long.

Collating (zipping) small files

Collating small files

- Need to have fewer files to reduce per-file overhead
- Counterintuitively, we want files to be large, so they transfer more quickly – do not use compression
- We can't write to local disk – zips must fit in memory
- Aim: turn millions of small files into thousands of large files on the fly

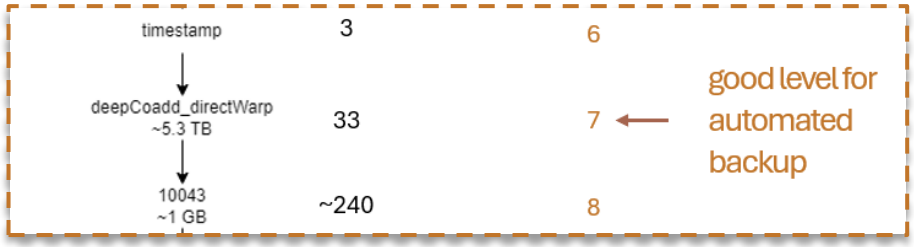
Caveats

- The backup at Echo is no longer a mirror of the data on CSD3
- In addition to monitoring, VM at RAL will discover new ZIP files, extract and upload
- Need to check these – use S3 metadata to list zip file contents
  - but S3 metadata has a character limit!
  - generate `*.zip.metadata` objects where zip contents list are long.

From simple `multiprocessing` to `dask distributed`

Python Scripts – Developed to use Dask

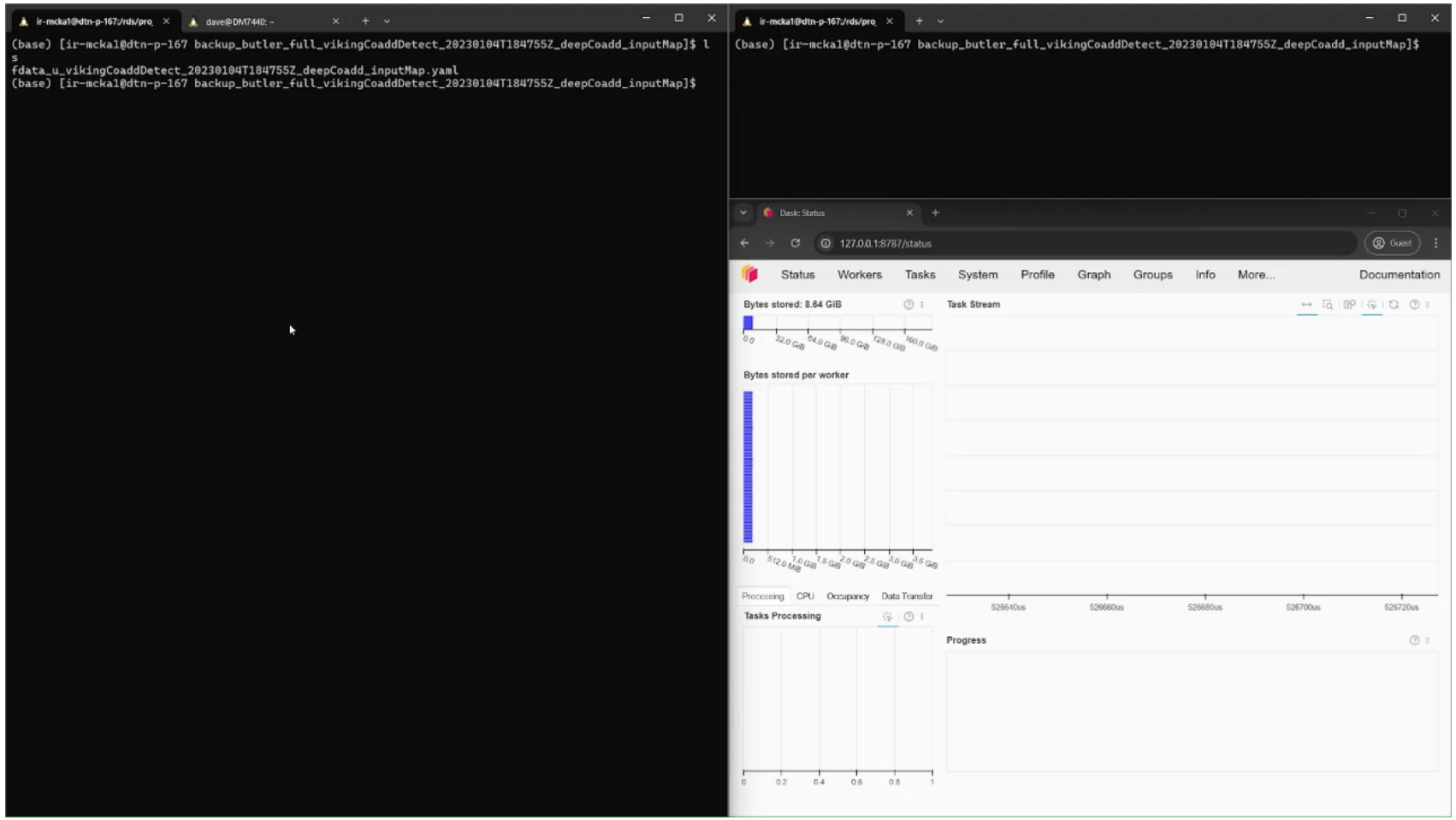
- Dask offers a simple framework for parallelisation of common Python data engineering tasks
  - Pandas-like API for data files
  - NumPy-like API for numerical data
  - distributed memory (allowing process-parallel tasks shared access)
  - scheduled workloads
  - visual task-flow dashboard
- Importantly, here:
  - built-in profiling
  - memory handling
- Pragmatic strategy
  - use 24 workers with 2 threads per worker
  - loop in shell script at first level with lots of branches



lsst_backup.py – CSD3 side backup script

Running a backup

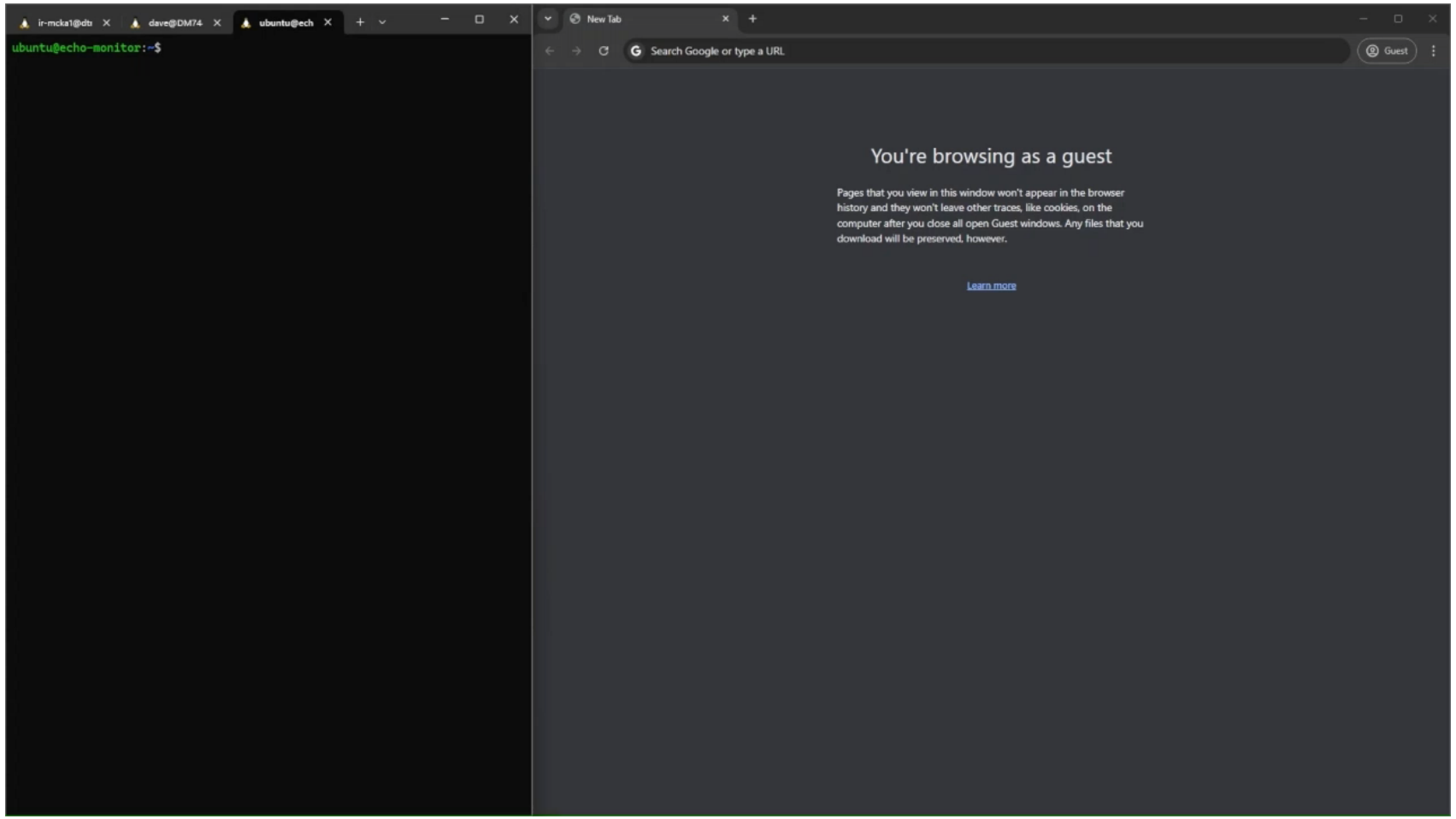
- Zipping (without compression) improves transfer rates
- example without zipping:
  - 18000 files, 0.5 MiB per file
  - ~23 it/s = 11 MiB/s = 0.1 gbps
  - 13 minutes
- with zipping (5 MiB size limit):
  - 1700 files, 5 MiB per file
  - “~250 it/s” = 130 MiB/s = 1 gbps
  - 72 seconds
  - time spent generating zip files included
- upload speed now depends on zip size, not original file size
- zip size controlled by single variable, but remains limited by folder structure



process_zips DAG – Echo-side Orchestration and Automation

Verifying and extracting zips

- VM local to Echo S3 runs Apache Airflow on microk8s
- each night, checks S3 buckets for zip files and extracts contents
- determines S3 buckets to scan at runtime from link below



https://raw.githubusercontent.com/lsst-uk/csd3-echo-somerville/refs/heads/main/echo-side/bucket_names/bucket_names.json

Next steps

Configuration testing

- Improve folder traversal to better generalise zip size control
- Determine ideal configurations for DEV users to backup outputs to Echo
- Investigate multithreading vs multiprocessing – simple due to Dask setup – currently introduce more threads if large files encountered

UK IDAC Ingestion (Somerville)

- Data Butler can ingest from local filesystem or through S3 URI, and can directly ingest zip files
- Do we:
  a. download individual files from Echo to Somerville and ingest;
  b. download zip files and configure Butler to use them;
  c. leave data on Echo and use S3 URIs on Somerville?

Acknowledgements

- George Beckett
- Bob Mann
- Amanda Ibsen
- Manda Banerji
- Raphael Shirley
- Elham Saremi
- Shenli Tang
- Timothy Noble
- Mathew Sims
- Alastair Dewhurst
- Richard Hughes-Jones
- Stuart Rankin
- HPC support