DAC21 Notebook/logbook

Notebook purpose: same spirit as the FDR "live" document, basis for the EC project report. The document is shared among the whole WP2, task and activity leads, sites, experiments, experts. Everyone is invited and strongly encouraged to contribute.

Table of contents (and direct links)

DAC21 high-level goals:

- Run production **Data Management, Processing and Analysis workloads**
  - **Data Management**: acquisition, injection, replication, lifecycles
  - **Data Processing**: **Production** (= experiment “data preparation”). Use cases: reprocessing, reconstruction, calibration, etc. requiring input and output from/to the Data Lake. This includes demonstrating ability to group scientifically related observations using datasets and containers (hierarchical if necessary) and utilize them in pracitce.. Activity requiring potentially “large” resources and scale up ability: computing farms, cloud provisioning, HPC, etc. Collaboration with **WP3** for software repos, image registry, etc. Possible leverage with EOSC-ACE for punctual and at-scale compute resources.

    - **Data processing**: **User Analysis.** Use cases: put the “prepared” data at the disposal of scientists (users), teaching purposes, citizen science (WP6), etc. Activity targeting, “small” resources: analysis platforms, online notebooks, personal computers, laptops, etc. Driven together with WP5: ESAP and Data Lake integration.
    - Demonstrate **Data Lake orchestration layer sustainability** after ESCAPE (towards EOSC)
        - Leverage, integrate and use experiment and site’s dedicated installations, e.g. RUCIO and FTS.
    - Demonstrate integration and interplay of these several instances in a common Data Lake/storage infrastructure.

- **End-to-End AAI** proven for all sciences and workflows exercised at the DAC21.
    - Assessment ranging from experiments experts, to advanced users to newcomers and to sporadic web-based access (i.e. citizen science => evaluate possibility for WP6 collaboration on “people’s science” project).
    - Token-based Authentication. Is it realistic as a target to have at least one workflow running 100% on tokens?
    - Ability to “flag” and address use cases for embargoed data, e.g. CTA and CMS

Experiments plan/goals/metrics

Experiment and point of contact	**Test name**	**Description and the running method (cron, script,...)**	**Method (cron script, interactive) Cron Offset**	**Anticipated timeline Success metric**	**Estimated #Files, data volume**	**QoS requirements**	**replicas / RSEs**	**Data access/ analysis plans?**	**Completed? Link to summary of each test**
FAIR Marek Szuba	mCBM ingestion	Registration of data acquired by the mCBM detector on FAIR-ROOT	Python script, run as ~~an inotify-driven persistent service~~ a cron job + interactively	Raw mCBM data is successfully registered in the data lake.	TBC. Note that this will be zero-copy ingestion, i.e. storage will effectively expand as new files are added	N	FAIR-ROOT	N	Completed successfully
FAIR Marek Szuba, Maisam Dadkan	R3B ingestion	Ingestion and replication of simulated R3B data	Combination of shell and Python scripts, run interactively or by cron	• Data is successfully registered • Data is successfully replicated • The associated metadata is assigned • Data can be discovered using the configured ESCAPE-Rucio client	Main: ~300 files x ~5 GB Aux: ~300 files of negligible size # runs: 1 # output replicas: 2 Total: **4 TB**	Y	FAIR-ROOT + 1x QOS=SAFE	Y	Completed successfully
FAIR Marek Szuba, Ralf Kliemt	PANDA ingestion	Ingestion and replication of simulated and digitised raw PANDA fallback data	Shell script	*f*	Fallback input data: 500 GB x 1 replica	Y	FAIR-ROOT, QOS=SAFE, QOS=FAST	Y	Completed, but many files stuck in REPLICATING state
FAIR Marek Szuba, Ralf Kliemt	PANDA simulation	Particle-transport and digitisation of Monte-Carlo events. Live ingestion of	Batch jobs	Data successfully processed and sent to the data lake	Output size: 1150 TB per run # runs: 1 # output replicas: 3 Total: **3.95 TB**	Y	FAIR-ROOT, QOS=SAFE, QOS=FAST	Y	Completed, but many files stuck in REPLICATING state

		simulated data.							
FAIR Marek Szuba, Eoin Clerkin	CBM simulation	Particle-transport and digitisation of Monte-Carlo events	Cron or batch jobs	Success may be determined via the successful retrieval and upload of material. Suggest hashes of datasets are stored locally at each step and compared to determine success.	Fallback input data: 1 run - ~ 3 MB x 1 replica Output size: ~5 GB/run # runs: 84 Lifetime of output data: 48h # output replicas: 1 Total: **122 GB** peak	N	FAIR-ROOT	N	Completed Successfully
FAIR Marek Szuba, Ralf Kliemt	PANDA reconstruction	Retrieval of stored RAW data from the data-lake, processing of the data and storing the processed data back to the data lake	Batch jobs	Data successfully retrieved, processed and returned to the data lake	Fallback input data: 500 GB x 1 replica Output size: 30 MB per run # runs: 84 # output replicas: 2 Total: **505 GB**	Y	FAIR-ROOT, QOS=OPPORTUNISTIC	N	Small Test completed successfully, full run failed due to files stuck in REPLICATING state.
FAIR Marek Szuba, Eoin Clerkin	mCBM reconstruction	Retrieve raw mCBM data from the data lake, run reconstruction on it and store the results	Cron or batch jobs	Success determined by the successful completion of event reconstruction as well as the successful upload/retrieval from data lake	Fallback input data: 10 files x 10 GB x 1 replica Output size: ~1 GB/run # runs: 84 Lifetime of output data: 48h # output replicas: 1 Total: **125 GB** peak	N	FAIR-ROOT	N	Completed Successfully
FAIR Marek Szuba, Maisam Dadkan	R3B data analysis	Analyse simulated R3B data stored in the data lake, upload resulting histograms and bitmaps to the DL	JupyterHub (kernels: Python, possibly C++)	• Successful authn/authz for user • Gain access to the data lake and perform the analysis • Write results back to the data lake	Failsafe input data: 1 run - 100 GB x 2 replicas (max.) Analysis output: ~10 GB per run # runs: 30 # output replicas: 2 Total: **201 GB**	Y	*FAIR-ROOT*, QOS=CHEAP-ANALYSIS, QOS=FAST, QOS=SAFE	N	Was not able to download files from DL using OIDC. completed successfully using x509.
KM3NeT Mieke Bouwhuis	KM3NET001	Ingestion of raw data from the shore station (remote RSE) and replication	cron shell python	- Raw data are successfully registered and replicated in data lake - Daily procedure	- 4 files/day; ~3 GB/file; 5days = 60 GB - 2 replicas of the ingest data= 120 GB	Y	SAFE & FAST(x2) SARA-DCACHE-TAPE IN2P3-CC-DCACHE	Y	Completed Summaryedh

					_____ Total: 180 GB		SARA-DCACHE		
KM3NeT Mieke Bouwhuis	KM3NET002	Download raw data from data lake, calibrate, and ingest into the data lake	shell python C++	- Raw data findable in data lake - Calibrated data successfully registered in data lake - Daily procedure	- Input are the data from KM3NET001 - 20 files x 3 GB = 60 GB _____ Total: 60 GB	Y	FAST, SARA-DCACHE	N	Completed Summary
KM3NeT Mieke Bouwhuis V. Pestel	KM3NET003	Conversion of calibrated and reconstructed data to a format for high level analysis	DLaaS Jupyter	- Data findable and data conversion using DLaaS interface - Data ingest and replication via DLaaS - Executed only once per large data set	- 56 files, corresponding to 65 GB (already on CHEAP-ANALYSIS) - 1 file of ~3GB to be uploaded _____ Total: 70 GB	Y	FAST, CHEAP-ANALYSIS IN2P3-CC-DCACHE SARA-DCACHE	Y/N	Completed Summary
CTA Agustin Bruzzese Jordi Delgado	CTA001	Long-haul transfer and replication	• Python • Cron	a.Data is successfully transferred, replicated, and file deleted on the origin RSE. b.Data transfer was monitored. c.Data can be discovered using the ESCAPE or CTA-RUCIO instance	Dependent on the test. Upto 50TB with ~2GB per file	N	CTA-RUCIO @PIC: non-deterministic (La Palma) and deterministic (PIC) RSEs	N	Completed
CTA Frederic Gillardo	CTA002	Data reprocessing		a. Use of RUCIO and DIRAC integration b. The data (DL1) are findable in the datalake (CTA instance) and are updated to a new version	100 TB, ~2 GB /file	Y HOT & COLD	CTA-RUCIO @PIC	Y/N	Postponed until December
CTA Gareth Hughes	CTA004	Data analysis	Jupyterhub/mybinder via ESAP	Higher-level analysis products produced	Small MB level	N		Y	Done. With caveats. Summary
SKAO James Collinson, Rob Barnsley, Rohini Joshi, Alex Clarke	SKAO_subscriptions	Data replication	Python (cron, rucio-analysis)	Data in correct place in timely manner.	**TBD**	N	**TBD**	N	Done, Success. Summary

SKAO James Collinson, Rob Barnsley, Rohini Joshi, Alex Clarke	SKAO_long haul	Long haul data replication	Python (cron, rucio-analysis)	Data in correct place in timely manner.	100s of Gb / 3hr		SKAO Rucio (AUS/SA to UK RSEs)	N	Done. Success. Summary
SKAO James Collinson, Rob Barnsley, Rohini Joshi, Alex Clarke	SKAO_end_to_end	End-to-end proof of concept data lifecycle test	Python (cron, rucio-analysis)	Data passes through / ends up in requested storage sites	10s of Gb	Y	SKAO Rucio (AUS/SA to northern hemisphere sites)	N	Done. Success. Summary
SKAO James Collinson, Rob Barnsley, Rohini Joshi, Alex Clarke	SKAO_DLaaS	Data analysis	DLaaS (Jupyterhub)	Successfully running SKAO's science data challenge (1) pipeline using data stored in datalake.	10s of GB. Need custom jhub notebook + ~90Gb of memory allocation (or as much as poss)	N	EULAKE	Y	Done. Success. Summary
EGO/VIRGO E. Cuoco,B. Patricelli, R. Poulton, A. Petrocelli, E. Marzini, A. Staniscia, S. Vallero, F. Morawski, A. Iess	EGO_Repo	Storage Endpoint/ Data repository/data analysis	Wavefier on Cloud system	Data Visible/Available through the ESCAPE Data Lake Data Management system	~ 2 Gb	N	CNAF	Y	
CMS Diego Ciangiottini, Daniele Spiga	CMS_DASK	Multi varies Analysis Facility PoC with data access via: DASK, Marconi, and large cluster	Python Interactive, batch job submission	Plot result replicated and stored back to the lake	10s of GB	Not required, but possible to be tested	CNAF DCache	Y	Done. Success
CMS Diego Ciangiottini, Daniele Spig	CMS_Cache	Content delivery	Python Interactive, batch job submission	Plot result replicated and stored back to the lake Performance comparison with above	-	-	CNAF	Y	Done. Success
CMS Diego	CMS_Embargo	Data lake embargo data test in CNAF	Manual copy tests with tokens	Correctly managing CMS-only data	10s of GB	Not required	CNAF DCache	Y	Done. Success

Clangiottini, Daniele Spig		and DESY							
CMS Diego Clangiottini, Daniele Spig	CMS_xcprotocol	XCache Protocol Translation: xroot internal vs http external	Manual copy tests	Correctly read files with a protocol different from the original	-	Not required	-	Y	
LSST Lionel Schwarz, Fabio Hernandez	LSST_raw_replication	Simulate replication of one night's worth of raw image data between 2 Rubin data facilities, perform the exercise several times	Bash and Python scripts (using Rucio API) manually run	All the images are consistently replicated to the destination	Each iteration is composed of 15TB, 800k files, ideally replicated in 12 hours or less	Not required	EULAKE → IN2P3_CC_LSST_DEST	N	Done. Success
ATLAS Arturo Sánchez	Data multiplication and replication.	Exercises (data production, replication and documentation) before and during the DAC21. Include the creation of datasets for real-kind final user analysis examples using current open-access datasets	• Bash scripts executed by the user • Jupyter notebooks scripting the download and multiplication of samples	• ~200*10 = 2000 files uploaded in the Datalake • Two copies of such files (rules) into at least two RSE's	• 5-6 TB	* RSE availability. * Replica to go to QoS=SAFE * Replica to go to QoS=CHEAP-ANALYSIS	• LAPP/CNRS RSEs • NAPOLI / INFN RSEs	YES	
ATLAS Arturo Sánchez	Data analysis in multiple forms. Final user analysis.	user analysis pipeline tests on experimental particle physics using augmented open data (http://opendata.atlas.cern/software/)	• C++ compiled code • Jupyter notebooks	• Testing and validating the reading access of the samples via de Jupyter rucio extension • Running multiple analysis pipelines that can last from a few minutes to a few hours (<4hr).	• Outputs of ~100's MB	* Transition input from SAFE to CHEAP-ANALYSIS * Reading from CHEAP-ANALYSIS, write output to local RSE * Transition output from	• LAPP/CNRS RSEs • NAPOLI / INFN RSEs	YES	

						FAST to SAFE			
MAGIC Agustin Bruzzesse Jordi Delgado	MAGIC001: Long haul ingestion and replication		• Python • Cron	• Data is successfully transferred, replicated, and file deleted on the origin RSE. • Data transfer was monitored. • Data can be discovered using the CTA-RUCIO instance	TB	N	CTA-RUCIO: non-deterministic and deterministic RSEs	N	Completed
MAGIC Agustin Bruzzesse Jordi Delgado	MAGIC002: Data reprocessing		• Python • Cron • Jupyter notebooks	• Replication of DL3 file • Validating the reading access of the samples via de Jupyter rucio extension • Multiple analysis using gammapy library	Small GB level	N	deterministic RSEs	Y	Completed
LOFAR Pandey, Yan Grange	**UseCase1** - Ingestion and replication	Ingestion of LOFAR data from a remote site (non-deterministic RSE) to the data lake, transfer and replication in off-site RSEs and after successful replication the data at origin can be deleted.	Script for the initial transfer Rucio python bindings for registering the rule definitions	Three days (one for the upload, one to keep the data on fast QoS, one to check it is not on fast QoS anymore). Success metric is if manually-uploaded data is correctly registered and managed.	Around 10 files (~2TB)	FAST, SAFE (this in principle would be “disk” and “tape” in a non-QoS world)	1 Non-deterministic RSE 1 or 2 RSEs for the next copies.	No (but can be convinced :)).	Completed
LOFAR Pandey, Yan Grange	**UseCase 2-** Data processing (retrieval, processing and	The ability to process data that is in the data lake at an external location. And combine it with other astronomical	Data upload, (lifecycle rules), Data retrieval, Processing, Archiving the results. Will make use of data relations (datasets, containers) and show its	One full day. Can be in parallel to number 1 since this runs on different hardware Success metric is being	31 files, 140 GB + some data from the VO that is an order of magnitude less.	FAST, SAFE (same as above, but	1 RSE per QoS level(?)	Yes. Data access will be from datalake	Completed, Successful

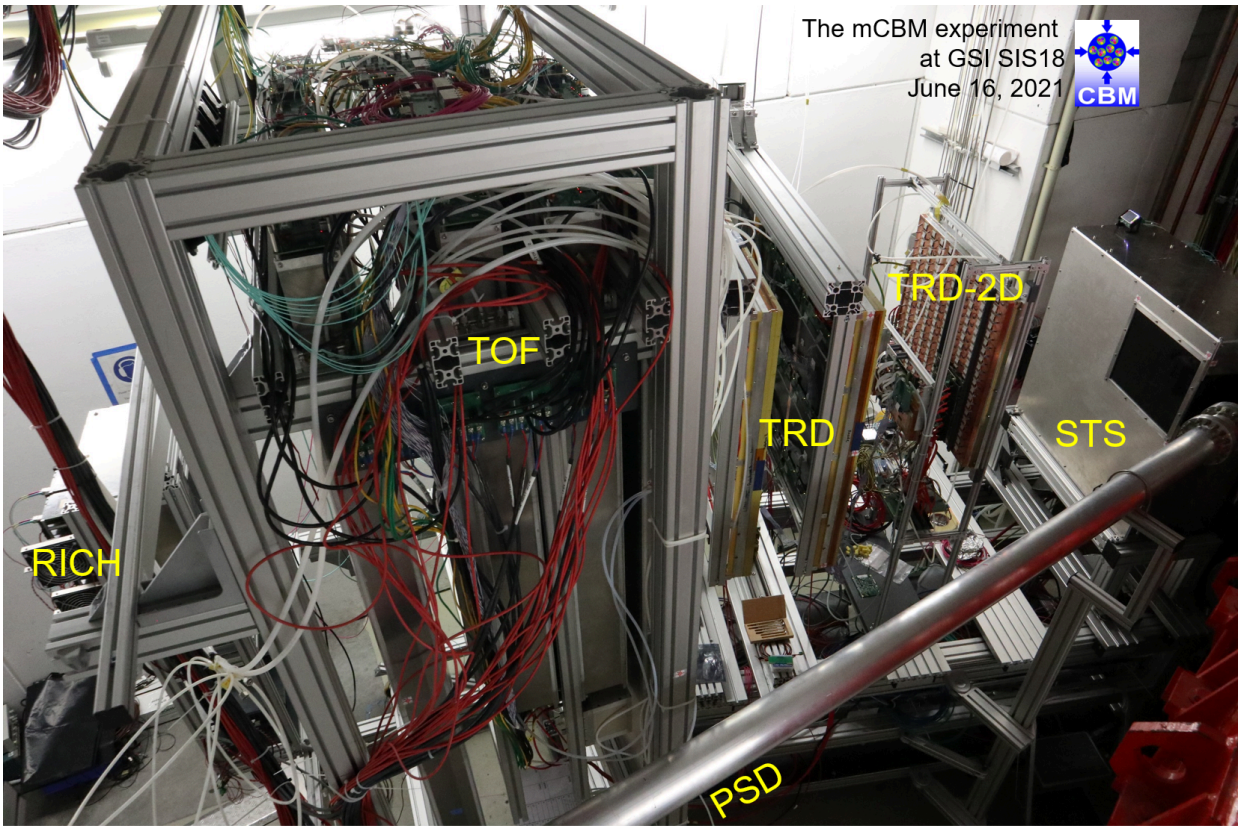
	archiving)	data to a multiwavelength image. Important to note that we aim to demonstrate grouping related observations and resulting products using concepts of datasets and containers. These grouping relations is then used in practice to process data.	applicability in practice. Also try to combine WP2, WP4, WP5. ESAP to locate data, both in Rucio as the VO. DLaaS with LOFAR software and VO tooling. Upload using DLaaS.	able to execute all the steps seamlessly and at the end having the resulting image on the DL		basically the inverse use case)		, analysis/ procesin g at external location.	
LOFAR Pandey, Yan Grange	Legacy archive, public data access RSE (**Optional** -> Actually it is very well - a POST DAC21 use case. We included it as this use case is a direct application of WP2 on a actual system)	Include a read-only RSE to a location outside the data lake. Get data from there into the DL. AAI: this could be a challenge. Will aim at the ambition “crossing virtual organisation” data but if keeping all in the same VO works that would already be great.	Rucio command line (goal of the test is mostly at higher-level systems)	Half a day. Success metric is being able to import data from the external RSE	Small ~10ish GB level	None (though this could be implemen ted by an EXTERN AL QoS level)	1 external RSE. Current LOFAR LTA seems liek a good candidate, but that could be too hard and then we create a specific one	See above	In progress Post-DAC21 activity.)
LOFAR Pandey, Yan Grange	Ingest & Replication (extension) (Post DAC21)	Extending Use Case 1 by using larger files and actually pushing to tape	Same scripts as use case1	Same as UC1	Around 1000 files (<1000) ~20-30TB	FAST, SAFE (this in principle would be “disk” and “tape” in a non-QoS world)	1 Non-deterministic RSE 1 or 2 RSEs for the next copies.	No (but can be convince d :)).	In progress

LOFAR Pandey, Yan Grange	Data processing all on DLaaS (Post DAC21)	Extending Use Case 2 by running all processing in the DLaaS, requiering a specific LOFAR software DLaaS.	DLaaS	One full day. Success metric is having an image on the DL with only using DLaaS for processing	32 files, 140 GB + some data from the VO that is an order of magnitude less.	FAST, SAFE (same as above, but basically the inverse use case)	1 RSE per QoS level(?)	Yes. Data access will be from datalake, analysis/ procesing at external location.	In progress

DAC21 Activities Logbook

FAIR

mCBM Ingestion (Marek on “ingestion” and Pierre/Eoin on “replay”)



[Fig. 1] *Above image shows the mCBM experimental setup. The beam pipe is visible transversing the lower right quadrant and the target box on the right centre of the photo. Each of the six detectors subsystems are over-labelled.*

All times are approximate and in UTC

Replay Side

- 2021-12-23, 15:00 - preparatory scripts for replay start. Several issues needed to be overcome whereby locked-in commands tried to specific ssh-keys where disabled by IT administrators. Our original plan therefore needed a work around and an alternative approach was pursued.

- The process was achieved by rsync'ing of the experimental data over Infiniband at ~150-200 MB/s, with 10 parallel copy jobs run on the GSI Green IT Cube cluster, leading to a 2 GB/s average rate. This rate was chosen to mimic the approximate high rate of mCBM data taking.

- The job array started at 18:32, most of them (9/10) finished around 18:36. One of the files had a much reduced writing speed of 800 kB/s due to an unknown reason. Writing for the final delayed file closed at 19:56.

- Infrastructure used:
    - Source: two compute nodes of the miniFles cluster (experiment side cluster of the mCBM), with each 5 HDD (4 TB) holding the original data files of the "mCBM 2021" beam campaign. The files were written in parallel so at least one file from each HDD is needed to reconstruct a segment of any mCBM run.
    - mFLES to Virgo (GSI batch cluster hosted in the GreenCube building) Infiniband backbone
    - 10 Virgo "logical nodes" (SLURM jobs with non-default ressources options), with 8 GB RAM each, each executing an rsync process to one of the mFLES HDD for the first 10 files of a "typical" mCBM run.

Ingestion Side

- 2021-11-19 - informed of last-minute decision of CBM Collaboration to postpone mCBM cosmic-ray data taking till early December. During the DAC we will instead replay acquisition of data taken by mCBM in July; from the data-lake point of view this will functionally the same as if we had data coming directly from the detector

- 2021-12-23, 15:30 - initial setup

- 2021-12-23, 17:30 - status check shows no replicas or rules having been registered with Rucio in spite of the first batch of files having already appeared in the source directory. Problem tracked down to file-system notifications not functioning as expected, began converting the script to polling mode

- 2021-12-23, 18:00 - tests of converted script repeatedly fail on connections to the CERN Rucio server being refused. Eventually discovered that recent changes to GSI network which allowed direct access to FAIR-ROOT from our compute cluster, now require rucio-clients traffic from FAIR-ROOT to the Rucio server to go through a local HTTPS proxy

- 2021-12-23, 18:30 - refactored script launched successfully, with forced delay of 60 seconds between files to avoid pile-up. Replicas begin to be added to the data lake

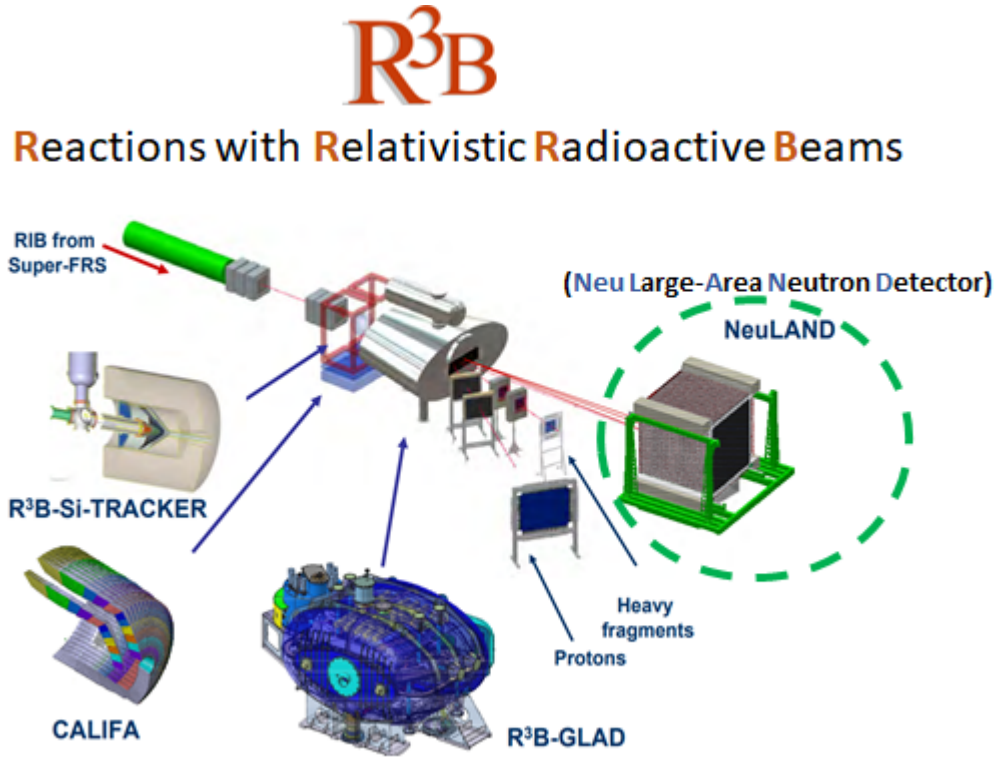
- 2021-12-24, 13:00 - status check shows 92 replicas registered successfully but owing to a typo, the script only ran once instead of periodically. Ran the script again to register the remaining 8 files
- 2021-12-24, 13:15 - final status check shows all 100 replicas registered successfully. Conducted a random sampling of associated DIDs, all files accessible

SUMMARY

- Achieved asynchronous zero-copy injection of mCBM data into the data lake

- Registering a replica and a corresponding replication rule took 30-60 s per each 4-GB file (having subtracted the aforementioned forced delay), i.e. comparable to the rate at which the data has been replayed

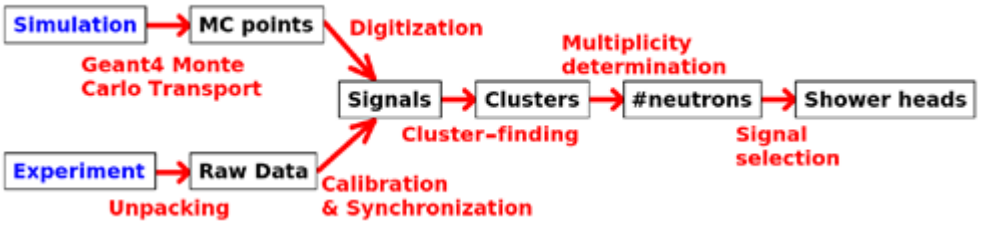
- Current bottleneck: calculation of Adler32 checksums (which obviously has to be done at least once, although with a bit of care one can avoid repeating the calculations for unchanged data) on the client side prior to registration of new replicas. This would ideally use checksums calculated by the underlying file system at the time of data being stored (if available and possible to be extracted at file rather than inode level), however Adler32 appears to be supported by rather few modern file systems and may not be available even when supported (e.g. at GSI - Lustre does support Adler32 but our cluster uses CRC-32C for performance reasons)
    - *Possible future development in Rucio:* add support for more modern checksum algorithms (e.g. xxhash as “fast” hash and SHA256/BLAKE2B as “strong” hash), ideally featuring a transition path from the current “Adler32+MD5” scheme



[Fig. 2] *An illustration of the R3B setup. The NueLAND is the modular neutron detection system that is used to detect outgoing neutrons from the reaction of the radioactive beam with the target.*

Introduction:

The R3B setup is a multi-purpose experimental setup used to study nuclear structure properties of short-lived isotopes through inverse kinematic reactions. The heart of the R3B setup consists of a fixed target, where the secondary beam from the Super-FRS is shot on. The general goal of the R3B experiment is to then provide a kinematically complete reconstruction of all particles participating in the reaction, so that the nuclear structure of the beam isotope can be studied. In order to accomplish this, the fixed target is surrounded by many different detector systems (see Fig. 2). Each type of the outgoing particles are detected by one of these detectors. The neutrons produced at the target, which are generally also very forward-boosted, are detected by NeuLAND (Neu Large-Area Neutron Detector). NeuLAND is a Time-of-Flight spectrometer that is meant to detect fast neutrons in the range 200 MeV-1000 MeV. The problem of finding the shower head among all the scintillator signals in Neu-LAND is challenging. Especially in the situation where multiple neutrons have to be detected in coincidence, solutions are far from trivial because of two reasons: 1) it is not (always) known a priori how many neutrons were detected and 2) showers from distinct neutrons tend to overlap quite often.Two analysis approaches have been developed to analyze the data of the NueLAND: 1) Technical Design Report (TDR) 2) Deep Neural Network (DNN). The R3B setup is under construction and, therefore, these two methods are applied to the Monte Carlo data from the specially designed simulation and data analysis framework of this experiment (R3BRoot). The data analysis workflow of R3B is almost the same for both simulation and the real data (see Fig. 3).



[Fig. 3] *The analysis workflow of the NeuLAND detector.*

Ingestion:

For DAC21, we used part of the available simulated data of 12 and 23 double-planes in three different energies of 200, 600, and 1000 MeV. The following steps was made to upload the data to the data lake:

- A portion of data including monte carlo points and other pre-processed data files (all in the .root format) after digitization step with a volume of 2.6 TB were chosen to upload.
- As a test, the Rucio-JupyterLab docker image with X509 Auth/z was used to upload a test dataset but it gave some unfamiliar warning. Therefore, we switched to the official rucio client docker image and it worked fine.
- The ingestion was started on 17/11/2021 at 16:00 using Rucio python client in a docker container. The first portion (~ 70 GB) was uploaded to the FAIR-ROOT rse. Due to work on having the GSI firewall allow access to FAIR-ROOT from the GSI batch farm having (temporarily) blocked external access to that RSE, the remaining data were uploaded to the GSI-ROOT and the EULAKE-1 rses.
- Due to the voms proxy expiration, the uploading failed two times. The ingestion was successfully completed on 18/11/2021 at 18:30.
- In total 1850 root files were uploaded to the data lake. All the files have a 9 months lifetime and are attached to the r3b_neuland_dataset.
- Attaching files to the dataset while uploading using rucio upload client in python did not work properly. The rules information of the files disappear after attaching. Therefore, we did file to the dataset attachment using rucio CLI after the files were uploaded.
- The data replication was started on 23/11/2021 at 5:00 using the QoS CHEAP-ANALYSIS label and 1 replica with 14 days lifetime. All the files were successfully replicated.

PANDA Ingestion (Ralf)
Panda simulations of background events at 15 GeV/c have been performed with the default detector setup. A total of 12500 jobs of 1000 events each were simulated, once in advance as fallback data and once in live mode. All data was sent from inside a container running on the Virgo cluster at GSI.

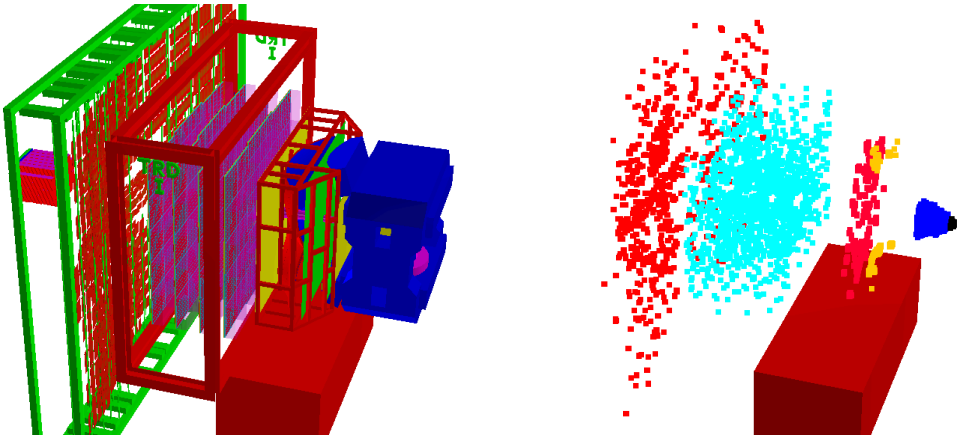
Task partially completed (2021-11-24):

- A small set of test data was sent successfully to the DataLake

- Fallback data (37500 files, ~2.3 TB) was sent by 125 parallel batch jobs from the Virgo Cluster
- Live data (37500 files, ~2.3 TB) was sent as the simulation jobs were ready directly from the cluster nodes. Almost all jobs were running at the same time, potentially leading to peaking requests to the DataLake
- A large number of files are stuck in “REPLICATING” state

CBM Simulation (Eoin + Paul)

The future CBM experiment at FAIR in its electron configuration with a MVD, STS, RICH, TRD, TOF and PSD detectors was simulated. (See Fig. 2)



[Fig.4] *Left shows the CBM simulation geometries for the future CBM experiment at FAIR. Consists of a blue magnet yoke on the right, followed by the RICH detector, the TRD, the TOF and PSD detector on the left. The beam enters from the right. The STS detector is contained inside the magnet. Right shows hits after transport from the demo scripts used during the DAC21 challenge.*

Task Completed during (Thu 2021-11-25) and (Fri 2021-11-26)

- Several docker images were prepared by Paul and Eoin. CBMROOT, our simulation software developed at GSI/FAIR for the simulation of the CBM experiments is built upon FairRoot which in turn is built upon FairSoft software packages. Docker images for each of these steps, starting from the most recent stable docker-hub Debian release were produced. Finally a fourth docker image containing the necessary python, gfal, rucio, java, voms-client and xrootd was built on top.

- Task, therefore, could be completed using a MacBook connected to a network external to GSI running the docker container.

- Initial Monte Carlo data was uploaded to the data lake and followed by an extraction from the data lake. File consistency was verified via md5sum before and after uploads.

- A second demo script (demo2.sh) simulates the transport of particles through the CBM experiment in its electron setup. Three data files are uploaded to the data lake containing parameters, geometry and transport data.

- It was decided that the demo script would run as a cron job running every 5 minutes which started on Thursday evening and continued through to Friday midnight.

- A third script (demo3.sh) extracted some of this transport data and generated digitalisation data which was run ad-hocly during the two days.

- Demonstration was sporadically successful. It should be noted however that interaction with the datalakes gave error messages due to server side issues. Additionally, timeout of voms proxy was also an issue.

PANDA Reconstruction (Ralf)

The goal was to have batch jobs read from the DataLake and upload the processed data.

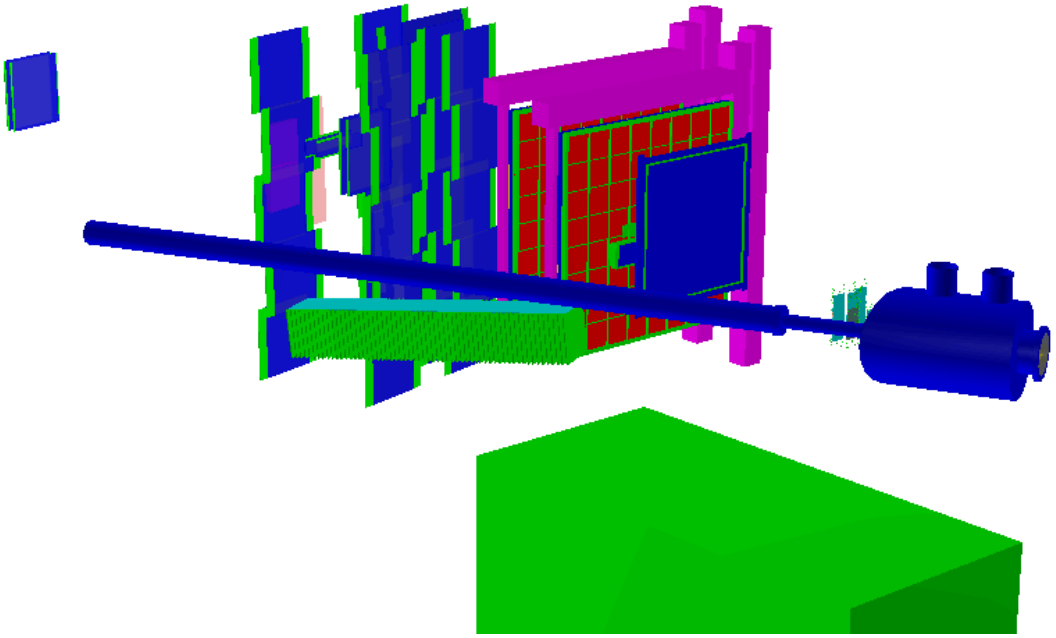
Task partially completed (2021-11-25 – 2021-11-26)

- A small set of test data was successfully processed and the results were uploaded
- The large scale job was not successful due to unavailable data / fallback data in REPLICATING state
- A solution could not be found within the remaining time.

mCBM Reconstruction (Eoin)

Goal of this task was to reconstruct recent mCBM (July 2021) data interacting with the data lake. Task was completed inside a docker container run on a Gentoo linux

desktop machine on the internal GSI network.



[Fig. 5] *Shows the simulation geometries of the July mCBM setup. The figure is comparable to the photo shown in Fig.1*

2021-11-22 It was noted that much of our demo environment developed for this task suddenly became obsolete due to changes in the development branch of CBMROOT which required updated versions of FairSoft and FairRoot. To make matters worse, these were major rather than incremental changes which required modification of the installation process. Bleeding-edge cbmroot was needed as software tools for reconstruction of July data was being developed actively. The necessary CBMROOT software was not available to complete this task on this day.

2021-11-23 Development of new Docker Images built on latest stable debian, building a fairsoft image, on which a fairoot image was built followed by a cbmroot image. Finally an image with rucio, gfal libraries, xrootd and all necessary additions to CBMROOT necessary to complete this task was built on top of the cbmroot image.

2021-11-24 Although unpacking of mCBM had been working for a week prior to DAC21, unpackers had made it into cbmroot the week before by Pierre, several developments in the reconstruction data still had not been submitted to the CBMROOT development branch. It is therefore difficult to prepare for this DAC21 task ahead of time. On this day, our CBMROOT software was still able to not reconstruct the 2021 data and we seriously planned to switch to plan B which meant reconstructing 2020 data instead.

2021-11-25 Good news! Tracking Reconstruction Code merged to the development branch of cbmroot by Valentina. These new CBMROOT macros allow reconstruction of the real experiment 2021 data from the

SIS18 experiment shown in Fig.1. The simulation geometries shown in Fig.3 is also used during the reconstruction process.

2021-11-26 13:00 New CbmRoot pulled and rebuilt within the running Docker container. This was also done so as to not have to rerun the docker image which was being used for the task.

2021-11-26 15:00-19:00 Several demonstration scripts tested and completed our task although in an ad-hoc fashion. Decided a systematic approach is warranted.

2021-11-26 18:47 Some raw mCBM data from run 1588 taken during July 2021 is uploaded to the data lake using a rucio-upload command. This forms the basis for later reconstruction.

2021-11-26 20:05 Script (demo6.sh) pulls this mCBM data. If pull is successful then continues, otherwise failure 1 is output to the log file. Next the script reconstructs the data using standard mCBM reconstruction macros merged into CBMROOT the day before. Upload reconstructed data to data lake otherwise declare failure 2 to the log files. Only full completion is considered a success.

2021-11-26 20:05 In order to get some success/failure statistics, this demonstration was run one hundred with the script (demo6.sh) in a “for” loop with a 60 second sleep between runs.

Statistics trial started at 21:04:50 and ended at 23:25:50. Of the 100 cases, 0 had ‘failure 1’, i.e. reading the raw mcbm data from the data lake using a rucio-get command, 75 had ‘failure 2’, i.e writing the reconstructed mCBM data the data lake using a rucio-upload command, and 25 were deemed fully successful having no failures reported. The failure rates were high, as one of the protocols failed on the clients side. The target RSE supported both ‘root://’ and ‘davs://’ as protocols. When contacting the RSE via the rucio upload command, the rucio client primarily chose ‘davs://’ as the protocol.
The ‘davs://’ protocol was not supported in the client container due to a minor error in the configured paths. gfal2, using its https plug-in, couldn't resolve the libdavix dependencies correctly. It therefore reported that the protocol is not supported.
The error message also contained "The requested service is not available at the moment", which is misleading and lets a user believe that the service is at fault.

mCBM Reconstruction (Eoin)
Goal of this task was to reconstruct recent mCBM (July 2021) data interacting with the data lake. Task was completed inside a docker

container run on a Gentoo linux desktop machine on the internal GSI network.

[Fig. 5] Shows the simulation geometries of the July mCBM setup. The figure is comparable to the photo shown in Fig.1

2021-11-22 It was noted that much of our demo environment developed for this task suddenly became obsolete due to changes in the development branch of CBMROOT which required updated versions of FairSoft and FairRoot. To make matters worse, these were major rather than incremental changes which required modification of the installation process. Bleeding-edge cbmroot was needed as software tools for reconstruction of July data was being developed actively. The necessary CBMROOT software was not available to complete this task on this day.

2021-11-23 Development of new Docker Images built on latest stable debian, building a fairsoft image, on which a fairoot image was built followed by a cbmroot image. Finally an image with rucio, gfal libraries, xrootd and all necessary additions to CBMROOT necessary to complete this task was built on top of the cbmroot image.

2021-11-24 Although unpacking of mCBM had been working for a week prior to DAC21, unpackers had made it into cbmroot the week before by Pierre, several developments in the reconstruction data still had not been submitted to the CBMROOT development branch. It is therefore difficult to prepare for this DAC21 task ahead of time. On this day, our CBMROOT software was still able to not reconstruct the 2021 data and we seriously planned to switch to plan B which meant reconstructing 2020 data instead.

2021-11-25 Good news! Tracking Reconstruction Code merged to the development branch of cbmroot by Valentina. These new CBMROOT macros allow reconstruction of the real experiment 2021 data from the SIS18 experiment shown in Fig.1. The simulation geometries shown in Fig.3 is also used during the reconstruction process.

2021-11-26 13:00 New CbmRoot pulled and rebuilt within the running Docker container. This was also done so as to not have to rerun the docker image which was being used for the task.

2021-11-26 15:00-19:00 Several demonstration scripts tested and completed our task although in an ad-hoc fashion. Decided a systematic approach is warranted.

2021-11-26 18:47 Some raw mCBM data from run 1588 taken during July 2021 is uploaded to the data lake using a rucio-upload command. This forms the basis for later reconstruction.

2021-11-26 20:05 Script (demo6.sh) pulls this mCBM data. If pull is successful then continues, otherwise failure 1 is output to the log file. Next the script reconstructs the data using standard mCBM reconstruction macros merged into CBMROOT the day before. Upload reconstructed data to data lake otherwise declare failure 2 to the log files. Only full completion is considered a success.

2021-11-26 20:05 In order to get some success/failure statistics, this demonstration was run one hundred with the script (demo6.sh) in a “for” loop with a 60 second sleep between runs.

Statistics trial started at 21:04:50 and ended at 23:25:50. Of the 100 cases, 0 had ‘failure 1’, i.e. reading the raw mcbm data from the data lake using a rucio-get command, 75 had ‘failure 2’, i.e writing the reconstructed mCBM data the data lake using a rucio-upload command, and 25 were deemed fully successful having no failures reported. The failure rates were high, as one of the protocols failed on the clients side. The target RSE supported both ‘root://’ and ‘davs://’ as protocols. When contacting the RSE via the rucio upload command, the rucio client primarily chose ‘davs://’ as the protocol.

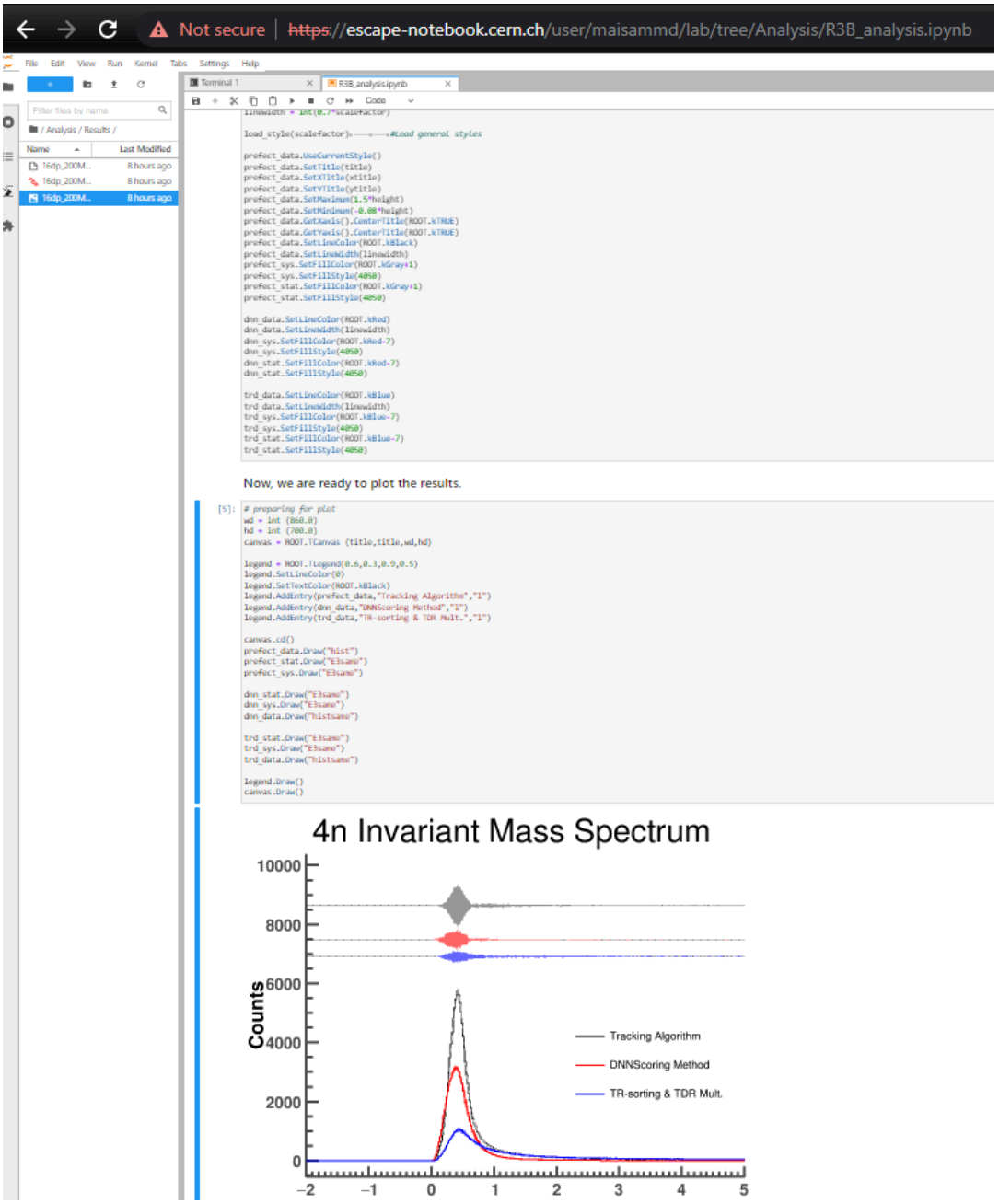
The ‘davs://’ protocol was not supported in the client container due to a minor error in the configured paths. gfal2, using its https plug-in, couldn't resolve the libdavix dependencies correctly. It therefore reported that the protocol is not supported.

The error message also contained "The requested service is not available at the moment", which is misleading and lets a user believe that the service is at fault.

R3B Data Analysis (Maisam M. Dadkan)

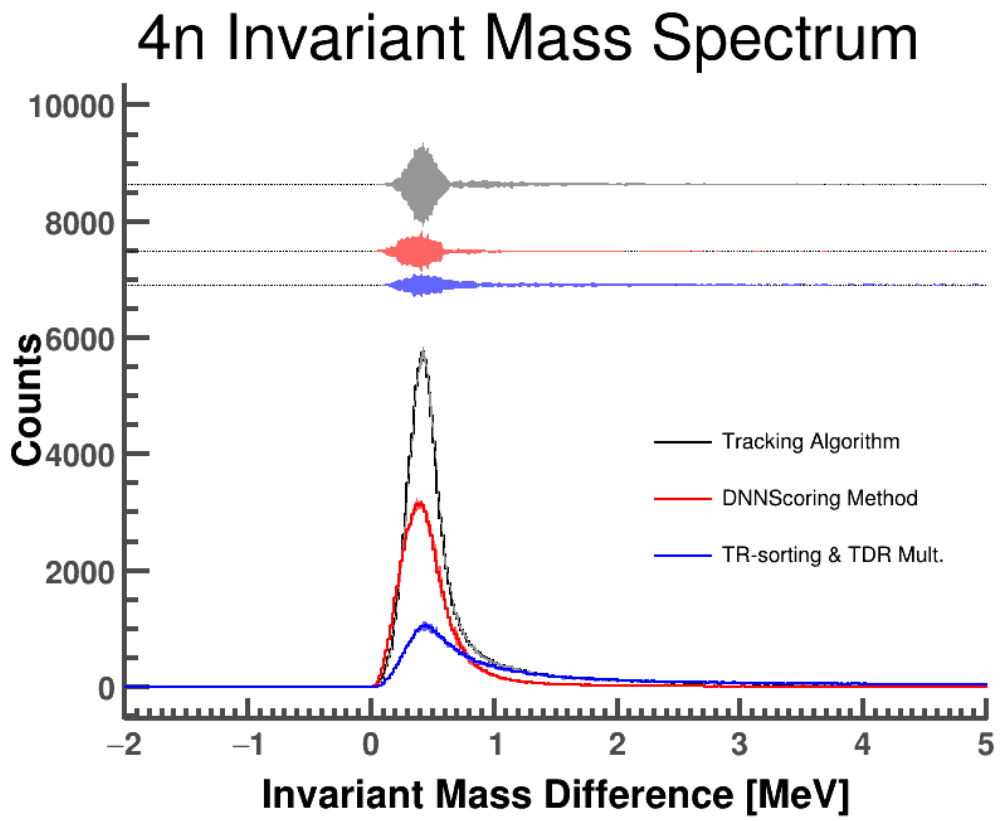
For the analysis on the R3B data, a PyRoot code in the notebook format was developed. The following steps were made to perform the analysis tasks:

- We aimed to use DLaaS for the data analysis using the ROOT environment.
- Get the required files of the R3B (TetraNeutron_InvMass_23dp_600MeV) for different Geant physics lists from the data lake using rucio.download.client. The test was successful using x509 but it did not work on the DLaaS using OIDC. Therefore, we used x509 in DLaaS to download the data.



[Fig. 6] *A screenshot of the analysis code of the R3B-NeuLAND in the DLaaS ROOT environment.*

- The spectrum of invariant mass difference was reconstructed for three different analysis methods for 23 double planes and 600 MeV energy (see Fig. 7).



[Fig. 7] *The result of the invariant mass difference resulted from three different analysis methods. It is clear that the DNN method gives better efficiency compared to the TDR method.*

KM3NeT

Use Case KM3NET001:
This exercise was done daily in the period 22-26 Nov. Every night at 04:20 a script was started via a cron job in one of the KM3NeT shore stations, that would ingest raw data files (corresponding to a full day of data taking) into the data lake.
Initially the procedure was not successful in ingesting and replicating the datafiles. We figured that this could be due to the fact that the same data files were used during DAC21 as the ones that were used during the MockDAC period, and erased after the end of MockDAC. Maybe a SCOPE:NAME can't be re-used.
Also the replication to QOS=FAST was not successful (STATE=STUCK) because of issues with LAPP-WEBDAV. After changing the replication rules to replicate to QOS=OPPORTUNISTIC, and by using unique data files, the tests were successful.
Each night the data ingestion involved only 3-4 GB, and the exercise took about 10 minutes.
We have not found a way to make this daily procedure fully automatic, as we have to renew the x509 certificate proxy in the shore station manually every day.

Use Case KM3NET002:
This exercise was done twice, at the end of the DAC21 week. In this exercise we wanted to download data from the datalake that were ingested from the shore station the night before (Use Case KM3NET001), then process them, and ingest the processed data into the datalake. However, since Use Case KM3NET001 was only successful towards the end of the week, this Use Case was performed only twice.
The findability of the data in the datalake and the download were ok, the merging of the KM3NeT software with the rucio data handling methods was ok, and the ingestion of the processed data into the datalake were ok.
We noticed that the file permissions of the files differ depending on the RSE, although the file upload was done in the same way for all files: all files were replicated to QOS=OPPORTUNISTIC. After download from "rse":QOS=OPPORTUNISTIC we saw that the files that were retrieved from LAPP-DCACHE and EULAKE-1 had executable permissions for user, group and others, while the file that was retrieved from GSI-ROOT had no executable permissions at all.

Use Case KM3NET003:
The Use Case KM3NET003 went well during the DAC21 week. The main activity was to retrieve 54 ORCA data files from the datalake and convert them into "open-data-format" (as this doesn't formally exist, I was creating a simplified version of the data files).
The files, grouped in a dataset, were made available with the jupyter-hub RUCIO integration (DLaaS). The generated files were later uploaded to RUCIO from the notebook itself, using the RUCIO python module.
The "**Minimal environment"** server config was used, the rest of the environment was installed with pip from a notebook.
From the technical point of view, I never succeeded in using username/password authentication in the RUCIO integration, and I finally used the x509 proxy.
Also, I didn't find how (but haven't thoroughly looked into) I could do more than just querying the file with the RUCIO integration (i.e. using the created variable pointing to the files to automatically retrieve meta-data etc ...). Apart from these 2 minor points, I find the system fairly easy to use.

CTA

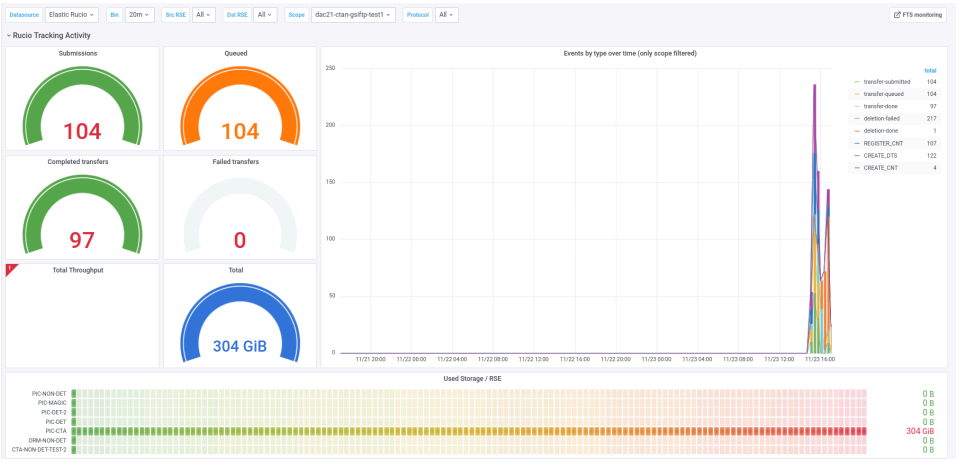
Use case 1 (Agustin B)

2021-11-23 - Test 1:

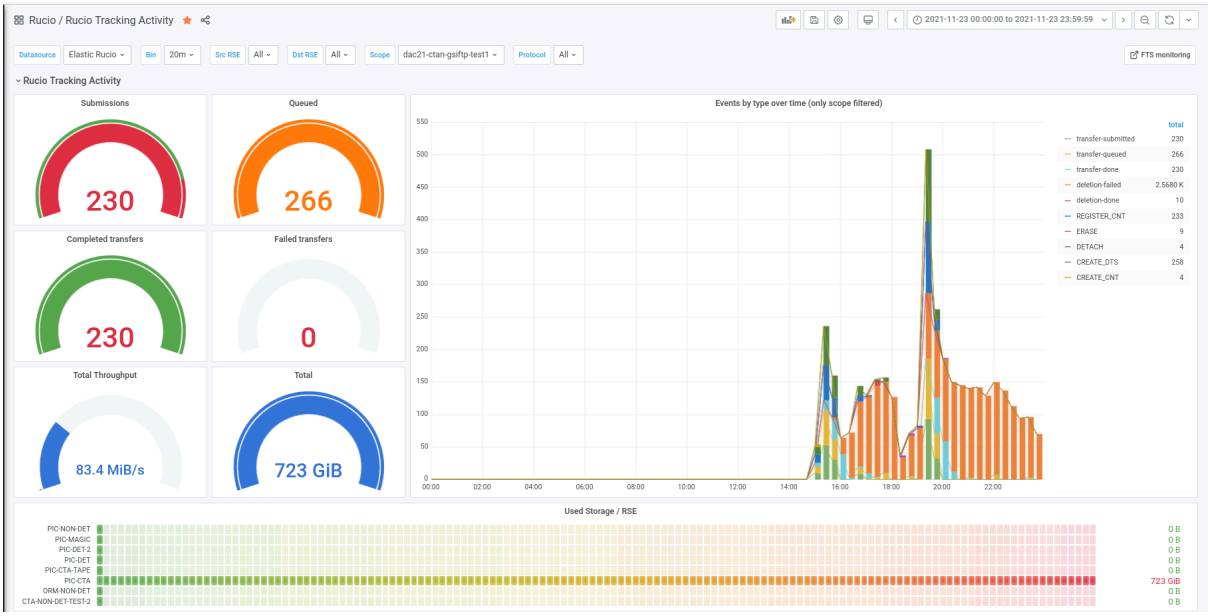
- 15:27: We had to delay the tests because we did not have the dataset provided by CTAO. At 15:25, we started. For the moment, we can observe the automatic discovery of files to be replicated, we

add them to the rucio database, also metadata,then it is replicated to the PIC. We don't see any problems at the moment.

- 16:44: In other words, 300 GB of LST data (DL0) has been transferred from La Palma to the PIC. Even so, we observed an inability to delete the files at the source.
    - Error: The requested service is not available at the moment. Details: An unknown exception occurred. Details: globus_xio: Unable to connect to datatransfer.ctan.cta-observatory.org:2811 globus_xio: System error in connect: Connection timed out globus_xio: A system call failed: Connection timed out
    - Reason: This is an error whose solution is due to the fact that the k8s cluster where the PIC's Rucio server is located, is not able to reach the IP of CTA's IT container. This requires modifying the range of IPs that can access the CTA IP.



- Finished successfully Test 1



2021-11-24 - Test 2:

- 00:11: Suddenly, it seems that the CTA IT container at La Palma does not accept my certificates.
- Error: WARNING Tue, 23 Nov 2021 23:31:45 +0100; Timeout stopped
- ERR Tue, 23 Nov 2021 23:31:45 +0100; Non recoverable error: [13] TRANSFER globus_ftp_client: the server responded with an error 530 530-Login incorrect. : globus_gss_assist: Gridmap lookup failure: Could not map /DC=org/DC=terena/DC=tcs/C=ES/L=Cerdanyola del Valles/O=Institut de Fisica d. Altes Energies/CN=pic01-rucio-server.pic.es 530- 530 End.
- Link
- 9:12: The problem was that there was a deployment of an endpoint that conflicted with the port of our gsiftp endpoint. The problem has been resolved

2021-12-03 - Test 2

- Solved all problems with the xrootd and https protocol at CTA's IT container:

  [bruzzese@mic01 ~]$ gfal-ls root://datatransfer.ctan.cta-observatory.org:1094//fefs/test/data_transfer/rucio_tmp/DL0/2021-11-25/obs115862643/moon/gamma_20deg_180deg_run140___cta-prod5b-lapalma_desert-2158m-LaPalma-moon.simtel.zst
  root://datatransfer.ctan.cta-observatory.org:1094//fefs/test/data_transfer/rucio_tmp/DL0/2021-11-25/obs115862643/moon/gamma_20deg_180deg_run140___cta-prod5b-lapalma_desert-2158m-LaPalma-moon.simtel.zst

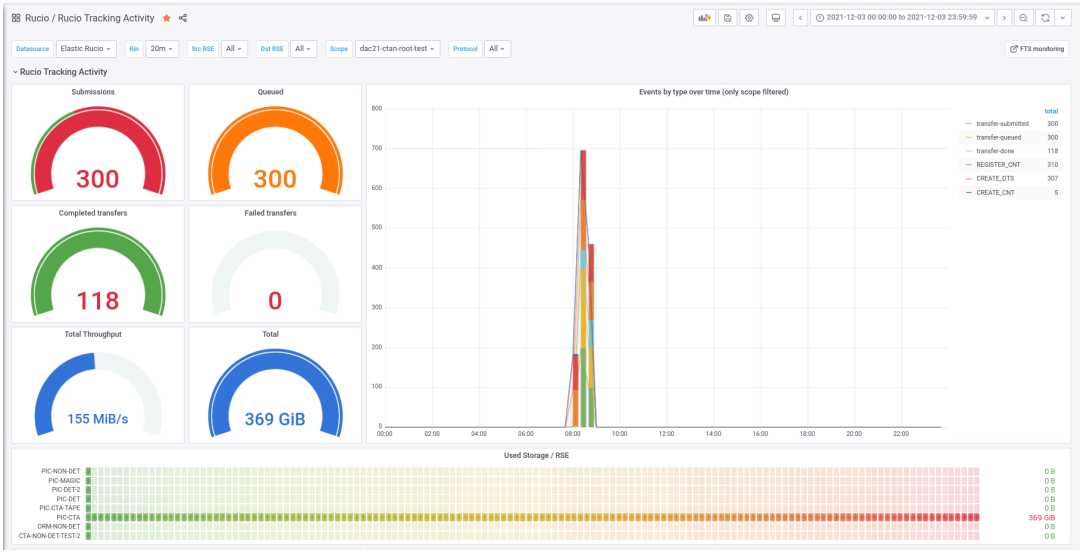
  [bruzzese@mic01 ~]$ gfal-ls https://datatransfer.ctan.cta-observatory.org:8000//fefs/test/data_transfer/rucio_tmp/DL0/2021-11-25/obs115862643/moon/gamma_20deg_180deg_run140___cta-prod5b-lapalma_desert-2158m-LaPalma-moon.simtel.zst
  https://datatransfer.ctan.cta-observatory.org:8000//fefs/test/data_transfer/rucio_tmp/DL0/2021-11-25/obs115862643/moon/gamma_20deg_180deg_run140___cta-prod5b-lapalma_desert-2158m-LaPalma-moon.simtel.zst

- We add the xrootd protocol of the CTAN IT controller to the CTA rucio instance

[bruzzese@rucio03 ~]$ rucio-admin rse info CTAN-NON-DET-ROOT
Settings:
=========
  availability_delete: True
  availability_read: True
  availability_write: True
  credentials: None
  delete_protocol: 1

```
  deterministic: False
  domain: ['lan', 'wan']
  id: b8633c7649694803b78c26d41dd1799c
  lfn2pfn_algorithm: identity
  qos_class: None
  read_protocol: 1
  rse: CTAN-NON-DET-ROOT
  rse_type: DISK
  sign_url: None
  staging_area: False
  third_party_copy_protocol: 1
  verify_checksum: True
  volatile: False
  write_protocol: 1
Attributes:
===========
  CTAN-NON-DET-ROOT: True
  fts: https://fts01.pic.es:8446
Protocols:
==========
  root
        domains: '{"lan": {"read": 1, "write": 1, "delete": 1}, "wan": {"read": 1, "write":
1, "delete": 1, "third_party_copy": 1}}'
        extended_attributes: None
        hostname: datatransfer.ctan.cta-observatory.org
        impl: rucio.rse.protocols.gfal.Default
        port: 1094
        prefix: //fefs/test/data_transfer/rucio_tmp
        scheme: root
Usage:
======
  rucio
        files: 318
        free: None
        rse: CTAN-NON-DET-ROOT
        rse_id: b8633c7649694803b78c26d41dd1799c
        source: rucio
        total: 1070697936007
        updated_at: 2021-12-03 08:03:23
        used: 1070697936007
```

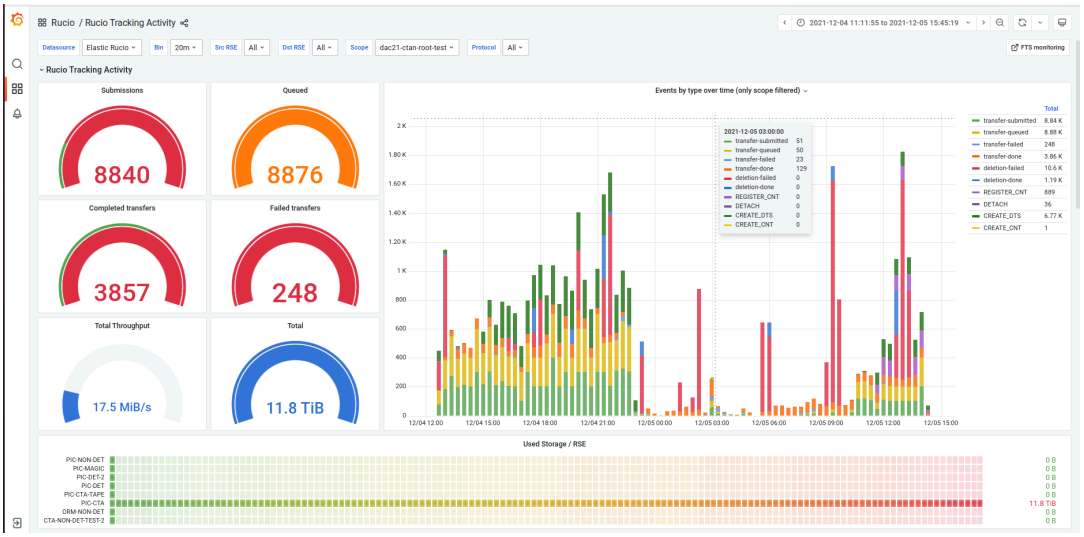
- Then we begin the 300 GB transfer from CTAN IT container to PIC



- Finished successfully Test 2

2021-12-05 - Extended Test 2

- We have transferred 10 TB with the xrootd protocol from the IT container to PIC in 24h



2021-11-24 - Test 3:

- Finished successfully Test 3

2021-11-25 - Problem with the IT at CTA when running Test 4:

Dear all,

we have a total outage of the cooling in the IT container.
Therefore, we are switching off all the machines now.

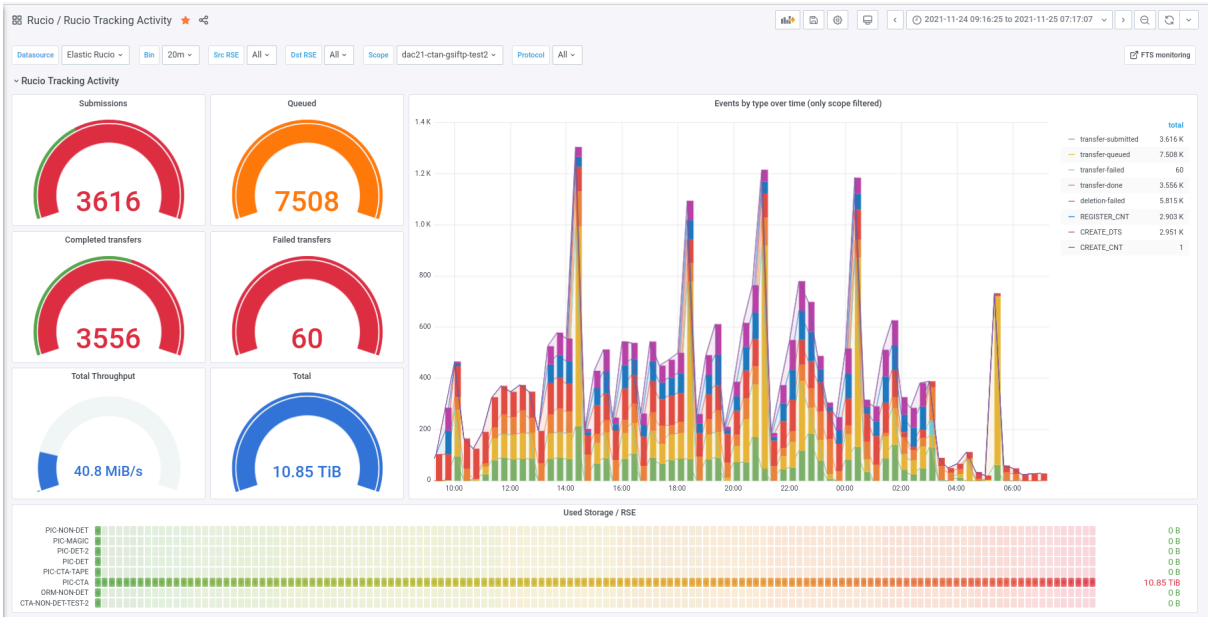
We will inform you once the IT center is up and running again.
It may take a while since Europe is sleeping now.

Sorry for the inconvenience!

Best regards,

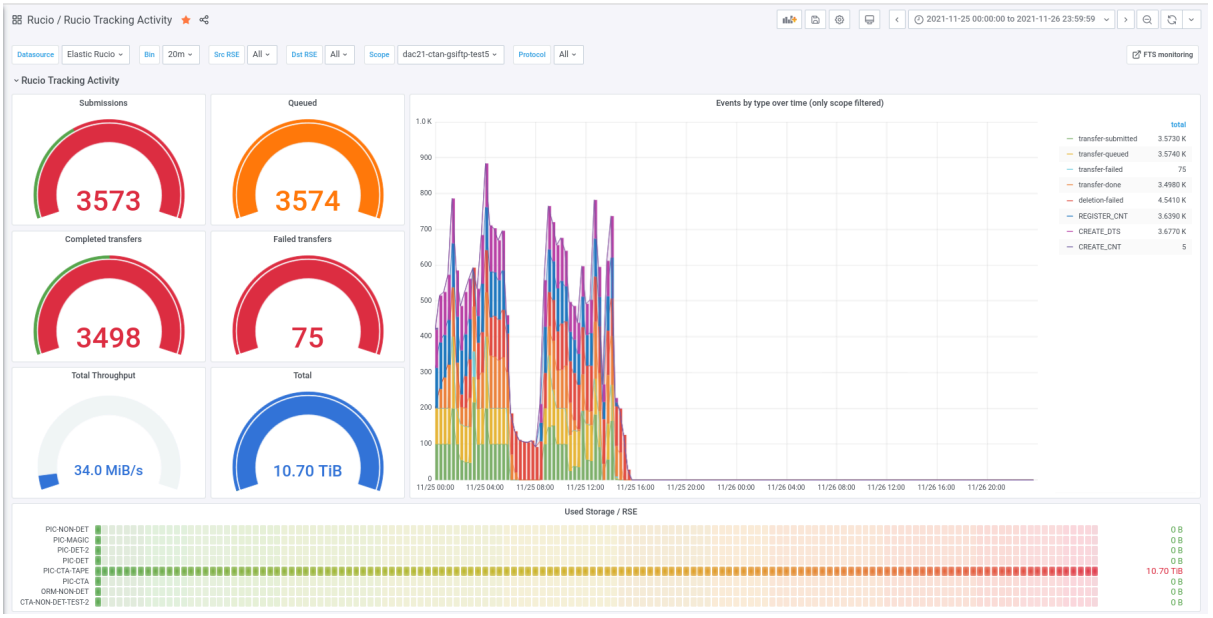
Daniela for the IT team

This message was circulated at 2am in the morning. The system was restored in the next hours, but the Singularity GFTP image hosting the RSE at the CTA site, was not running. We restarted the image in the morning at 9:30am. Then we resumed the transfers



2021-11-25 - Test 5

- Finished successfully Test 5



2021-11-26 - Tests 2 and 6

Test 2 worked in the test environment (internal images and RSE at PIC) so functionally speaking all RUCIO configurations are correct, and failed due to the network configuration of the XROOTD at the CTA IT site. Still not sure what is the reason, but looks like some network rule in the firewall is preventing the

connection. We will check again after DAC21.

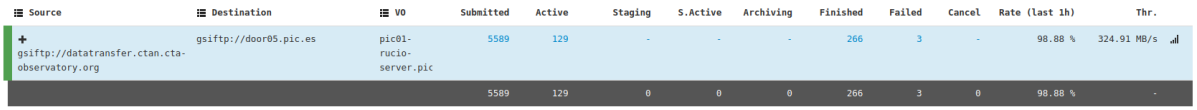
Anyhow, this test was an optional one, since the same test using GridFTP (Test 1) worked well.

Test 6 worked in the test environment (internal images and RSE at PIC) when defining the replication policies of the datasets, so functionally speaking all RUCIO configurations are correct. We couldn't run the test on production environment since we are experiencing problems to connect from PIC to CNAF due to an update of Grid certificates in both sides, changes in the CA. CNAF support and Adrea Cecantti updated the DN and certificate for the user in charge of connecting to CNAF but didn't work. Need to debug in more detail to understand the situation, but looks like something related to Vomses configuration and IAM.

2021-11-26 - Test 4

Test 4 is running from 2021-11-25. It was interrupted due to the issue reported from the CTAN IT Admins, then resumed the 26th in the morning. Since it is a quite big dataset of 40TB, we don't expect to finish in the time window of 1 day. We didn't expect that despite the issue reported in the morning. We expect that it will finish on 27th?

The execution is running with a parallel execution of multiple data streams by FTS, and we can observe 324MB/s over a connection of 1Gbps. With an expected and dedicated connection of 10Gbps the scenario would change significantly.



Source	Destination	VO	Submitted	Active	Staging	S.Active	Archiving	Finished	Failed	Cancel	Rate (last 1h)	Thr.
gsiftp://datatransfer.ctan.cta-observatory.org	gsiftp://door05.pic.es	pic01-rucio-server.pic	5589	129	-	-	-	266	3	-	98.88 %	324.91 MB/s
			5589	129	0	0	0	266	3	0	98.88 %	-

Use Case 4 (Gareth)

Data:

4x 500kb DL3 level files. Which represents the output of CTA Use Case 2.
Data initially uploaded to CTA_LAPP_FERDERIC scope using Rucio docker container.
> rucio upload --lifetime 1210000 --rse LAPP-DCACHE --scope CTA_LAPP_FREDERIC <FILENAME>

Software:

gammapy (version 0.19)

gammapy is not yet fully onboarded into the OSSR. This was due to the fact that the main gammapy dev in ESCAPE moved on. The gammapy group (external to CTA and CTAO) then decided to wait for the next software release before finishing

the onboarding processes. Therefore the metadata or codemeta.json file does not exist and cannot therefore be found by ESAPs interactive menu.

Therefore an example notebook was uploaded to the Astron gitlab which could then be hardcoded into the ESAP, this simulated full onboarding:
https://git.astron.nl/astron-sdc/escape-wp5/workflows/cta-example/

When pointed to a mybinder the Docker file in this repository builds the jupyterlab environment that includes gammapy

An example notebook was created:
https://git.astron.nl/astron-sdc/escape-wp5/workflows/cta-example/-/blob/master/spectral_analysis-Copy1.ipynb

Execution:

mybinder:
The environment could be generated on mybinder, the JIVE mybinder but not on the SKA mybinder instance. In the last instance the image was created but got stuck at “pushing image”.

DLaaS:
It was not possible to install on-the-fly (pip install) as the gammapy software on a DLaaS instance as gcc is required.

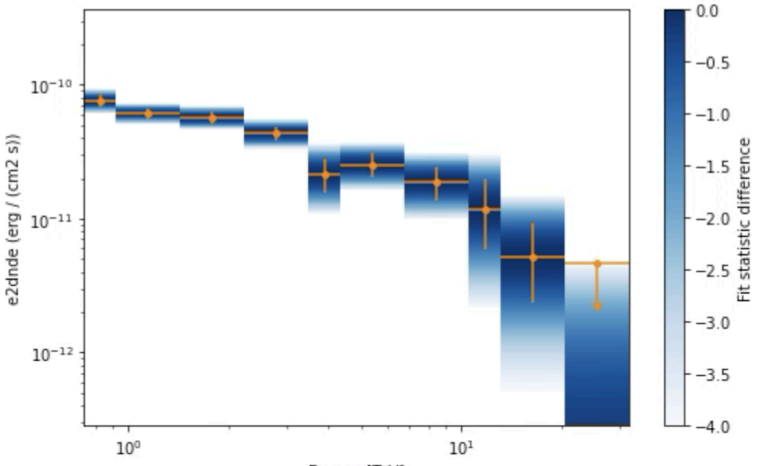
Accessing the Data:

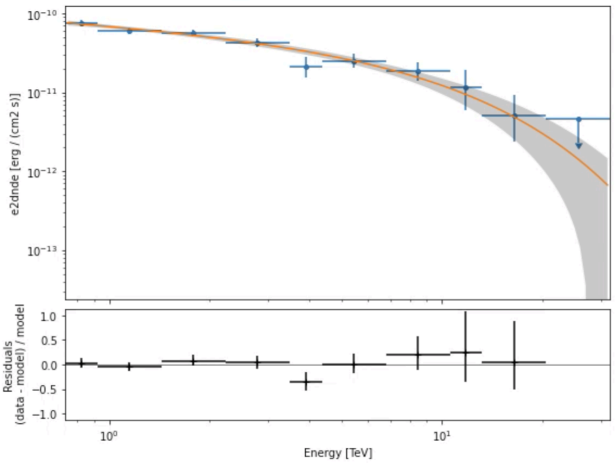
From mybinder I was able to access the shopping basket and obtain the json block stating where the data is stored. However it is not possible at this time to download the file. One solution might be some form of download function to the REST API. Or the CTA data should be made available in a way that can be parsed using the pandas framework setup in the shopping basket.
For the mybinder example I proceeded by downloading the data from an alternative source using functions in gammapy.

From DLaaS I was able to make the data available.

Results:

Higher level data products: Spectral Energy Distribution of the Crab Nebula in the TeV regime.





Conclusion:
The main aim of the Use Case "Higher-level analysis products produced" was achieved. However, data could not be accessed from the ESAP launched binderhubs. Two possible solutions are: a Rucio REST API download method/function or CTA data made available on a service separate from the DL - update. If a gammapy container was available on DLaaS this would not be an issue, however we would lose the ability to dynamically change the analysis - update.

Update 1.12.2021:

Agustin and Alba added an image to DLaaS that enables the installation of gammapy. After logging into DLaaS you now how the option of an image with gcc. Once launched, open a terminal.

Open a terminal:
> conda install -c conda-forge gammapy
> conda install -c conda-forge iminuit

Exercise complete using DLaaS. See image below.

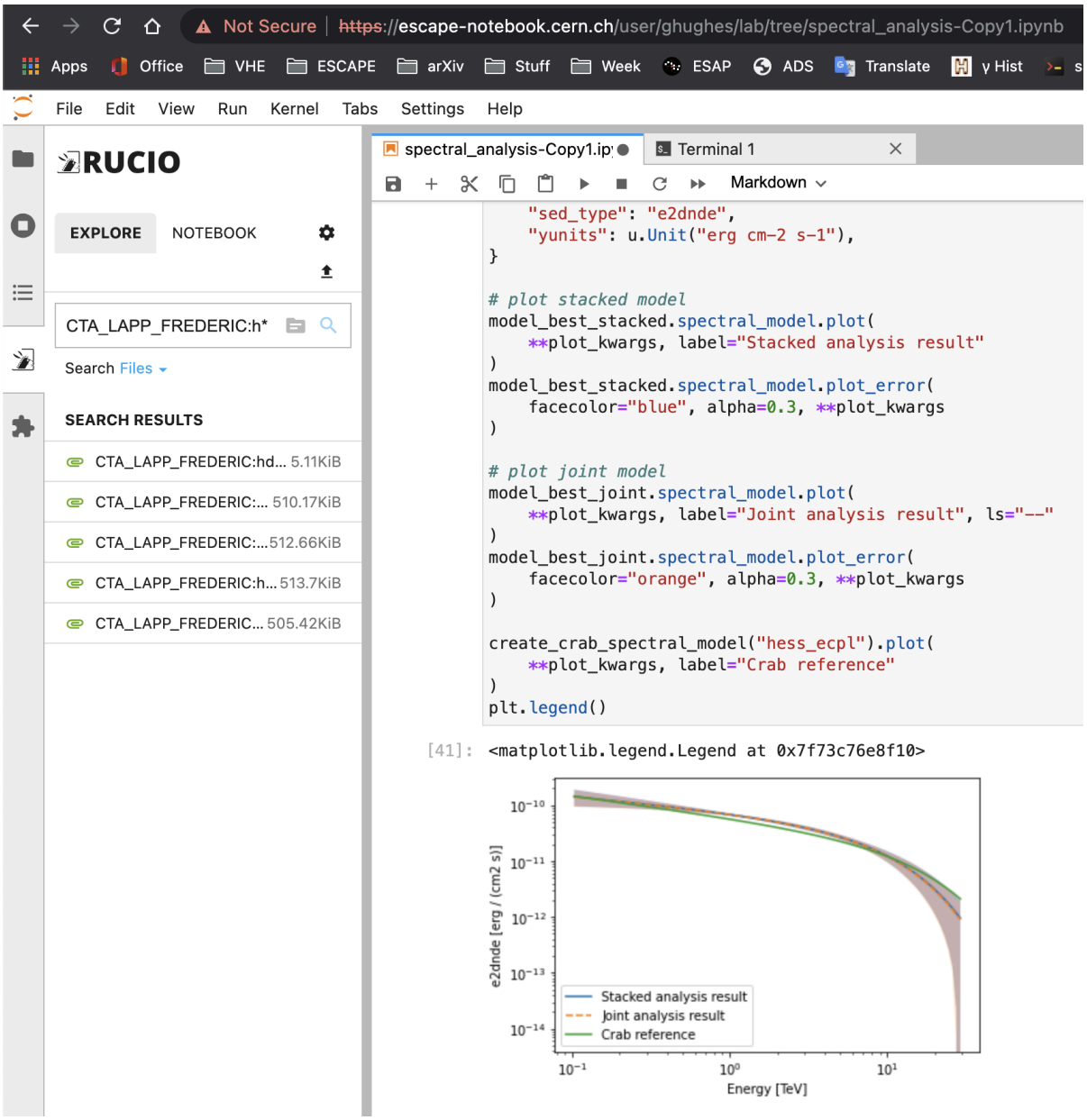
Update 3.12.2021:

Rizart asked about this on the Rucio Slack:

"Hello, as far as I understand, neither the download or the upload functionality is possible via using the REST interface, but one has to use directly the clients. Is this intentional? Probably this was avoided as not to burden the server with many download/upload streams? Thanks!

Martin Barisits:
Mostly correct. You do not strictly need to use the rucio clients to download, you can use something else (E.g. get a metalink with the replicas via rucio server rest, and then download the replicas yourself) - there is also the option of a redirector, which http redirects you to the replicas.
but there is no way to stream things, via rest, from the rucio server directly"

This is an interesting thing to pursue.



CTA Summary

Use Case 1 Long-haul Transfer:

- Six tests defined 4-5 completed successfully (see table below).
- Amount of data transferred in <1day is comparable to the volume required by CTA.
- The inclusion of Tape storage meets an important goal in the CTA bulk data management plan.
- Technical difficulties in deletion at the source and replication in CNAF.
- More tests planned next year, including some new tests regarding file size and priority data.

Use Case 2 Reprocessing:

- Delayed due to initial tech difficulties and scheduling.
- The number of methods available in DIRAC to manage the catalogue is much larger than that of Rucio. It may be necessary to implement new missing methods.
- Understanding the cache in dCache will also be important wrt to accessing the data on Tape.
- Upgrade of Rucio from v1.23 to v.1.26 may solve some issues.
- Work planned to continue in January.

Use Case 4 Analysis:

- Analysis completed on both ESAP (via “OSSR” & Jupyterlab/mybinder) and DLaaS
- ESAP shopping basket would need to be adapted to accommodate files of this type, replicate to http download service or the workflow should change towards IVAO TAP like services.
- DLaaS requires a preloaded container or container that can install software on the fly.

Final thought:

- The results of UC1 are already very interesting from a CTAO construction perspective.
- Several important requirements have been met. Several more are within reach.
- We will continue next year with further tests, which will aid in the construction project.
- Within striking reach of a telescope-to-user chain, using ESCAPE tech.

Test name	Protocol	Estimated #Files, data volume	STORAGE	Replicas/RSEs	Results	Observations
Test 1	GSIFTP	300 GB	Disk	CTA-RUCIO: non-deterministic and deterministic RSEs	Completed successfully.	
Test 2	XROOTD	300 GB	Disk		Completed successfully.	
Test 2.1	XROOTD	10 TB	Disk		Completed successfully.	Initially not planned, is a mix of Tests 2 and 3
Test 3	GSIFTP	10 TB	Disk		Completed successfully.	
Test 4	GSIFTP	40 TB	Disk		Failed. Completed in more time than stated	Problems with the Rucio server side transferring large number of files.
Test 5	GSIFTP	10 TB	Tape		Completed successfully.	
Tes 6	GSIFTP	10 TB	Disk		Failed. Completed internally.	We couldn't run the test on production environment since we are experiencing problems to connect from PIC to CNAF due to an update of Grid certificates in both sides, changes in the CA

SKAO

Log book

22-11-21

- 11:00 Cron job kicked off subscription creation test and new data being uploaded every hour. Subscription and relevant rule created as expected.
- 11:00 conveyor-submitter daemon had fallen over over the weekend. Screenshot shows the dashboard that monitors daemon liveness in sec (time from last log timestamp). Note large number for conveyor-submitter.

  Deleted the pod to force recreation. Fts delegated credentials had expired, ran manual job from the cronjob. It is catching up on old transfers from the weekend now.
- 12:45 Still working on old transfers. Subscription transfers not done yet.
- 14:00 Still working on old transfers. Subscription transfers not done yet.
- Following the issue above, a lot of files were waiting to be submitted to FTS by conveyor-submitter. Upped both the number of threads (1->4) and pods (1->2) to help process the backlog.

-
- Data being uploaded and registered as part of the dataset being subscribed (transferred) from early in the day. Conveyor-submitter issues were resolved, and submitter worked through its backlog to do these transfers by around 1600. After this, the pending deletions were done in bulk (red). The rest of the day the hourly transfers and deletions progressed as expected.

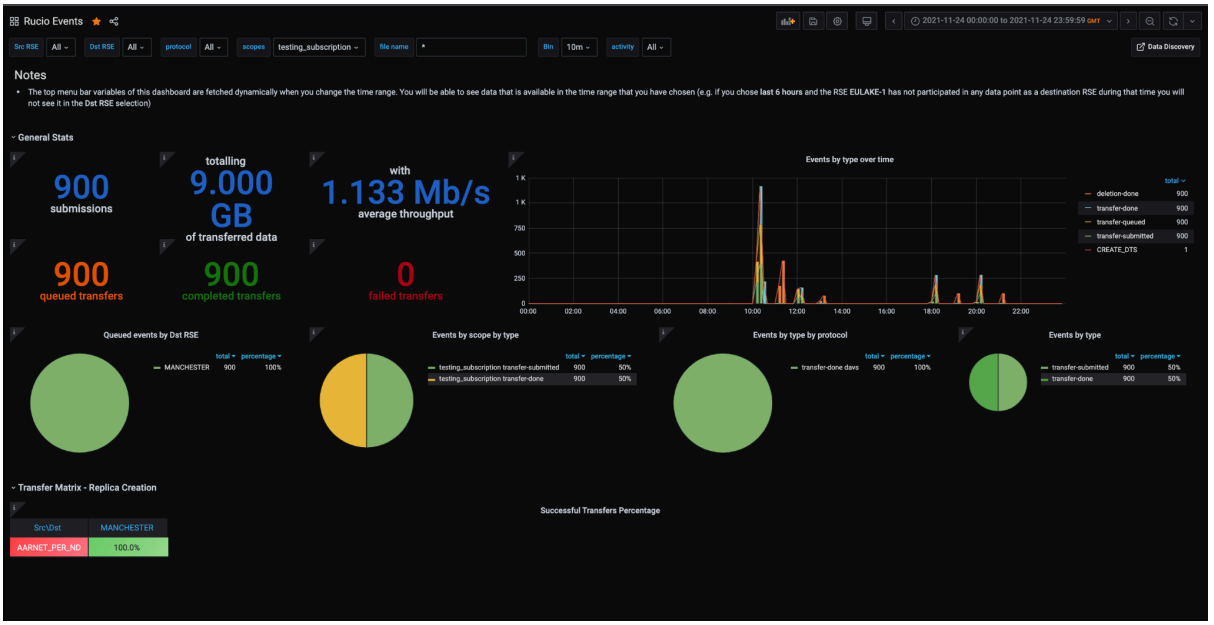
23-11-21

- Data lifecycle test on SKAO Rucio conducted from IDIA (South Africa) to QOS=FAST (MANCHESTER).
- Issue with using gfal-ls. As part of upload/register tests, gfal-ls used to find filelists created in upload section of test. Logic in test hadn't accounted for current directory (.) and parent directory (..) listing and was crashing out. Fixed by skipping these.

- Some issues with the yaml, resolved by around 16:00
- Dashboard view of the day:

24-11-21

- Data lifecycle test on SKAO Rucio conducted from AARNET (Australia) to QOS=FAST (MANCHESTER).
- Issues with the extra slash needed for the root protocol. Fixed by introducing scheme dependent logic to PFNs, i.e. if scheme == root, add extra ‘/’.
- Existing filelists had to be manually downloaded/searched/replaced/reuploaded to fix, done by 10:30.
- Dashboard view of the day:



25-11-21

Long haul transfer tests ran successfully on the whole. Need to investigate the cause of failed transfers which seem to be dominated by the 50 MB file size tests.

Test summaries

SKAO_DLaaS (22-11-21)

Demo can be viewed here.

We wished to evaluate the functionality enabled by the ESCAPE data-lake-as-a-service (DLaaS) platform, which facilitates the staging of data in a Jupyter notebook environment. We have developed an example SKAO workflow to test interactive data analysis platforms by running a source finding and classification routine on some synthetic SKAO images. This was the challenge set to participants in the SKAO Science Data Challenge 1 (SDC1), and our solution workflow has previously been containerised, and built on top of the jupyter-singleuser notebook environment, allowing us to demonstrate this use case on both JupyterHub and BinderHub previously.

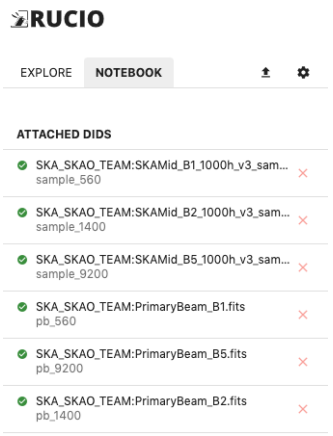
However, data staging remains a challenge, and has previously involved downloading the data into the interactive environment prior to processing. This is where the DLaaS adds significant value, by delegating the data staging to the ESCAPE Rucio instance. We thus modified our existing SDC1 solution workflow environment to extend the datalake-singleuser image, allowing it to be deployed on DLaaS.

From this point, we were able to launch our notebook servers with access to files in the data lake and the software needed to generate insight from that data. The RAM available to each notebook's user pod was not sufficient to analyse the full 4GB images set for SDC1 (~100 GB RAM required), but for a proof of concept we were able to use sample images, and demonstrate staging these in the notebook environment using the Rucio tab in the JupyterLab user interface.

Once the environment was available for use our SKAO end-to-end test was as follows:

1) Use the Rucio JupyterLab element to search for the SKA_SKAO_TEAM:SDC1 data and select 'make available' to replicate this to EULAKE-1 (previously stored at DESY-DCACHE).

2) Assign variables to the DIDs representing the sample images, the primary beam images and the input (‘truth’) catalogues:



3) Develop and run a notebook to perform image preprocessing (primary beam correction and training area cut-out). This also creates symlinks to the data files in the notebook home space, since the preprocessing stages by default output the generated data products in the same directory as the input images, which in the case of the attached RSE storage, is read-only.
4) Run a script which deploys the source finding software PyBDSF to identify sources in the images, outputting these catalogues to disk.
5) Upload the source catalogues back into the SKA_SKAO_TEAM:SDC1 dataset (this in practice would mean further development could continue from this point, without needing to run the expensive source finding step. This had to be done using the Rucio CLI in a terminal, as the Rucio JupyterLab interface did not work at this stage.
6) Train a random forest classifier on the training area sources and truth catalogue, before using it to make predictions about the unseen sources in the full sample image.
7) Upload notebooks and scripts into the SKA_SKAO_TEAM:SDC1 dataset to facilitate reproduction.

The main functionality under test here - the Rucio-JupyterHub integration - worked well in general. We have noted above that the user interface to upload from the notebook was not working during the DAC21 week, but uploading via command line (to CNAF-STORM) functioned correctly.

We positively noted the benefit of defining variables for the SingleItemDID objects in notebooks; once the notebooks are shared between users in the DLaaS these variables persist, which favours reproducibility.

We also note that there is currently no way to configure the length of time for which a DID is replicated at EULAKE-1 from the DLaaS Rucio interface; by default this is seven days, but in future it may be worth enabling the DLaaS user to configure this, in the event that they need the data staged for longer than this.

Subscription test (22-11-21)

We wanted to test Rucio's basic subscription functionality to show metadata-based data movement. This highly parameterised method of data movement has several use cases, including moving Observatory-level data products to the SRCs.
The rucio-analysis test creates a subscription by passing parameters including name, lifetime, scope, and a set of metadata. It attaches this metadata to a dataset and uploads files to the dataset which triggers data movement. As more data products are uploaded into the 'subscribed' dataset, this new data is also replicated for the duration of the subscription lifetime. Hourly cron jobs were used to automatically upload and register data into the dataset at Lancaster from 12 am and the one-time subscription creation ran at 9 am to replicate to Manchester. A summary of the test can be seen below. [TODO replace dashboard screengrab, check all times]

New data products were uploaded hourly and the subscription created successfully. Shortly after, we discovered that the conveyor-submitter Rucio daemon had crashed over the weekend. We restarted it at around 10:00, and it began to work through the accumulated backlog of events. This daemon is responsible for submitting transfers to FTS. To help it with the extra payload, we bumped up the number of threads in the k8s pod running the daemon and also deployed an additional conveyor-submitter pod. Having worked through the backlog, the first batch of transfers corresponding to this test happened around 16:00. Once replications were complete, the pending deletions were done in bulk (seen as a spike in red). For the rest of the day the hourly transfers and deletions progressed as expected.
Thus, despite some minor hiccups the test ran largely successfully, and rucio demonstrated the ability to recover from the daemon failure.

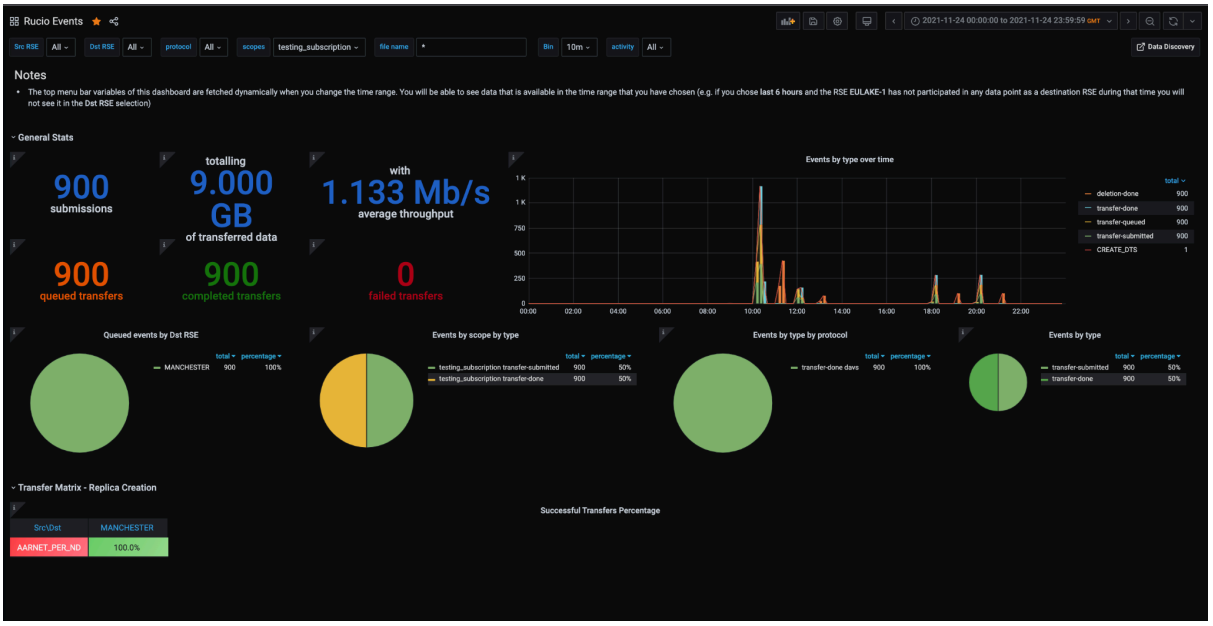
End to end data lifecycle test - IDIA (23-11-21)

An extension of the subscription test above, this test uploads data to an RSE in South Africa (IDIA_ND) and replicates it via a QoS-based subscription to Manchester. Again, the main use case is to emulate moving Observatory-level data products to the SRCs. Data has a short lifetime at source to signify a valuable

storage space on the Observatory side where data will be staged to be moved off into the network of SRCs as quickly as possible. The subscription was created via a one-off cron job at 0900, and new data was being uploaded every four hours.
Although the data was being *uploaded* successfully, the rucio-analysis test hadn't accounted for current directory (.) and parent directory (..) listing and was crashing on the last step of registering data as part of the 'subscribed' dataset. The way in which files and directories are listed varies from site to site and depends on how the storage admins have configured their storage. We identified and fixed this issue as well as a small typo with the yaml file used to deploy the test. This combined with the fact that new data was uploaded every 4 hours, the test generated events at/after 16:00 to catch up with the accumulated data of the day. The 20:00 run to upload and register data as well as the corresponding deletion events to delete the short lived data at source occurred smoothly.

End to end data lifecycle test - AARNET (24-11-21)

Similar to the lifecycle test above, this test uploads data to an RSE in Australia (AARNET_PER_ND) and replicates it via a QoS-based subscription to Manchester. Again, the main use case is to emulate moving Observatory-level data products to the SRCs. Data has a short lifetime at source to signify a valuable storage space on the Observatory side where data will be staged to be moved off into the network of SRCs as quickly as possible. The subscription was created via a one-off cron job at 0900, and new data was being uploaded every four hours.
The AARNET_PER_ND RSE currently has only the root protocol configured. During this test we ran into an unforeseen manifestation of a known issue which is that the root protocol requires an extra slash when using it to define storage URLs and/or physical file names (PFNs). The issue was identified and fixed in the morning and the test ran smoothly for the rest of the day.



Nice to have: It would be useful to be able to set replication rule lifetimes based on a timestamp instead of a lifetime in seconds. This would allow them to be absolute lifetimes and not relative to the time of rule creation.

Long haul transfer tests (25-11-21 to 26-11-21)

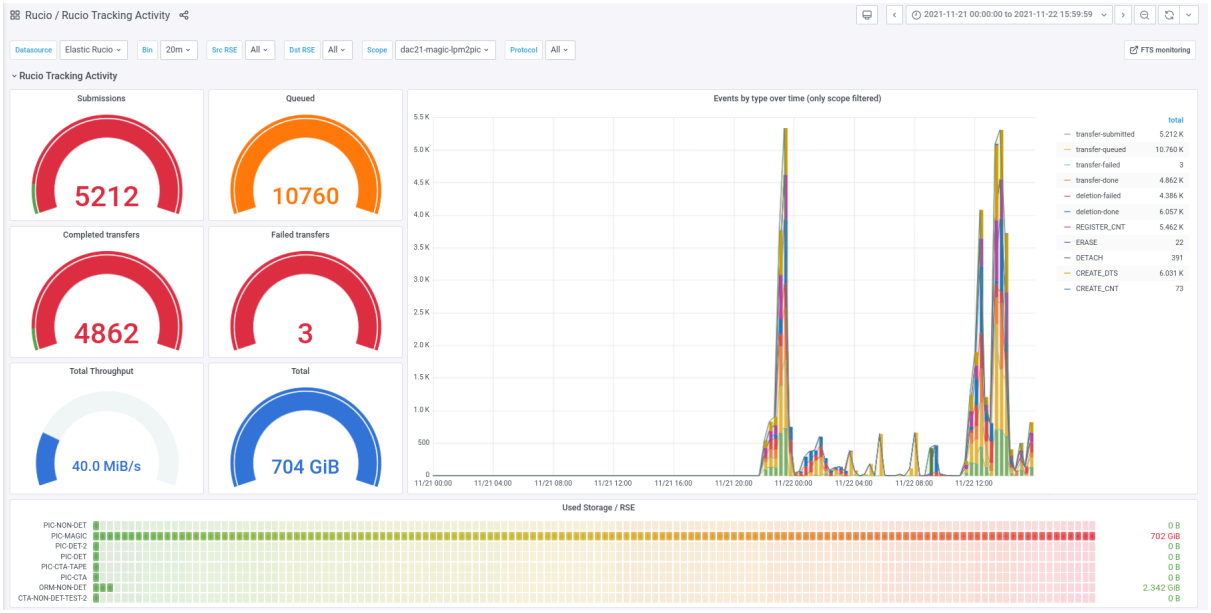
Establishing reliable long haul connectivity from the telescope host countries to the SRC network is going to be a crucial aspect of SRC functionality for SKAO. With this set of tests, we aim to exercise the existing network links available to us by pushing large amounts of data across these links continuously. By cycling through 3 file sizes (50 MB, 500 MB and 5 GB), we can get an idea of which file size is more performant across these long haul links. This test bulk uploads data to long haul RSEs (AARNET_PER and IDIA) and replicates them to Manchester, and Lancaster. (TODO: add DESY if we can also add an explanation for why desy has more failed transfers than the UK sites).
Hourly cron jobs run this bulk upload test and they cycle through the three file sizes mentioned. In other words, the 50 MB version of the test runs every 3 hours as does the 500 MB and 5 GB one, both with a unique offset. The number of files is inversely proportional to the file size such that the total volume uploaded every hour is 50 GB irrespective of individual file size.
TODO: Add overview dashboard
TODO: Add filesize-throughput impact dash

On the whole, the test ran largely successfully.

Follow-up needed?

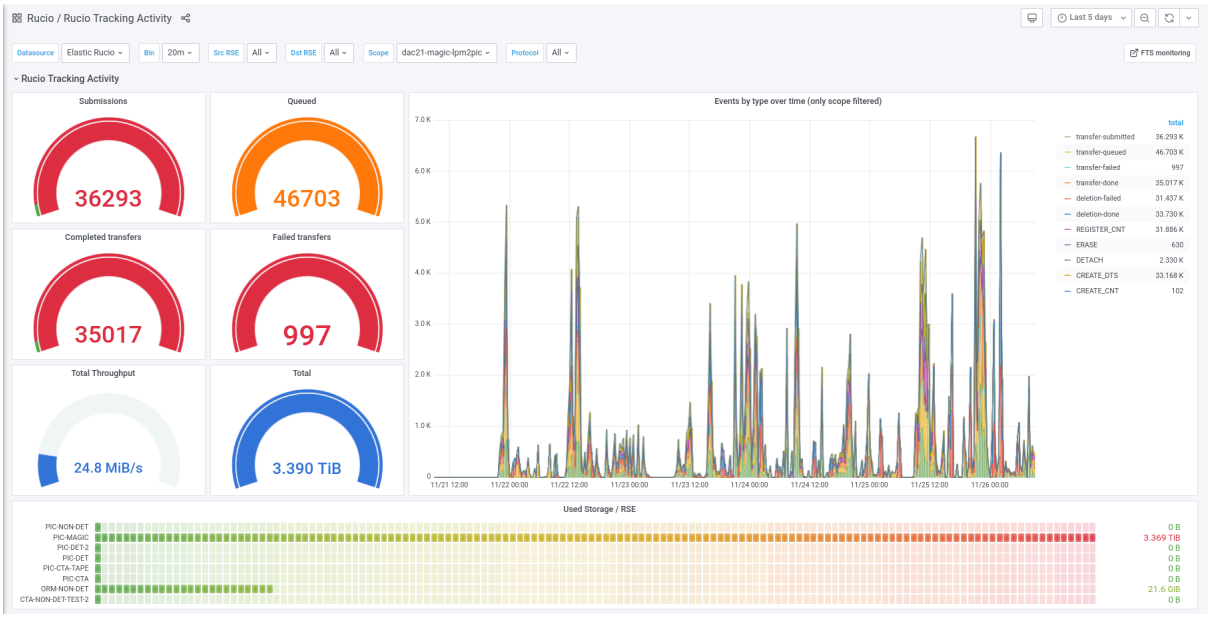
MAGIC

22/11/21 - Test1:



25/11/21 - Test2:

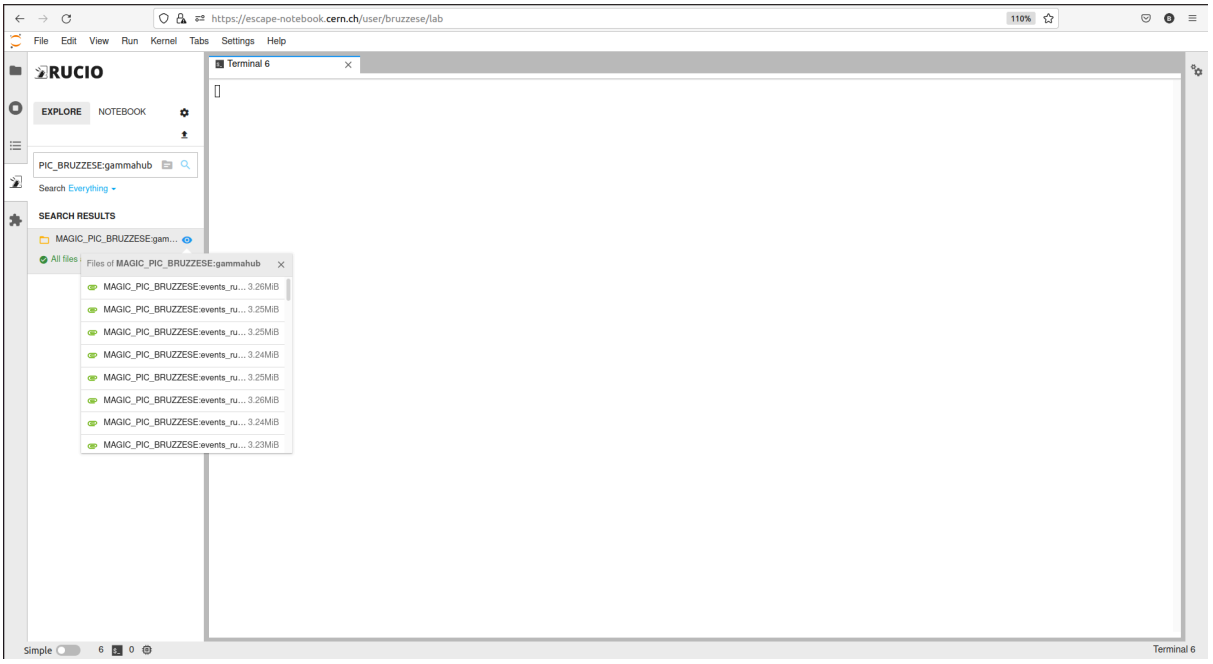
- Finished successfully Test 2



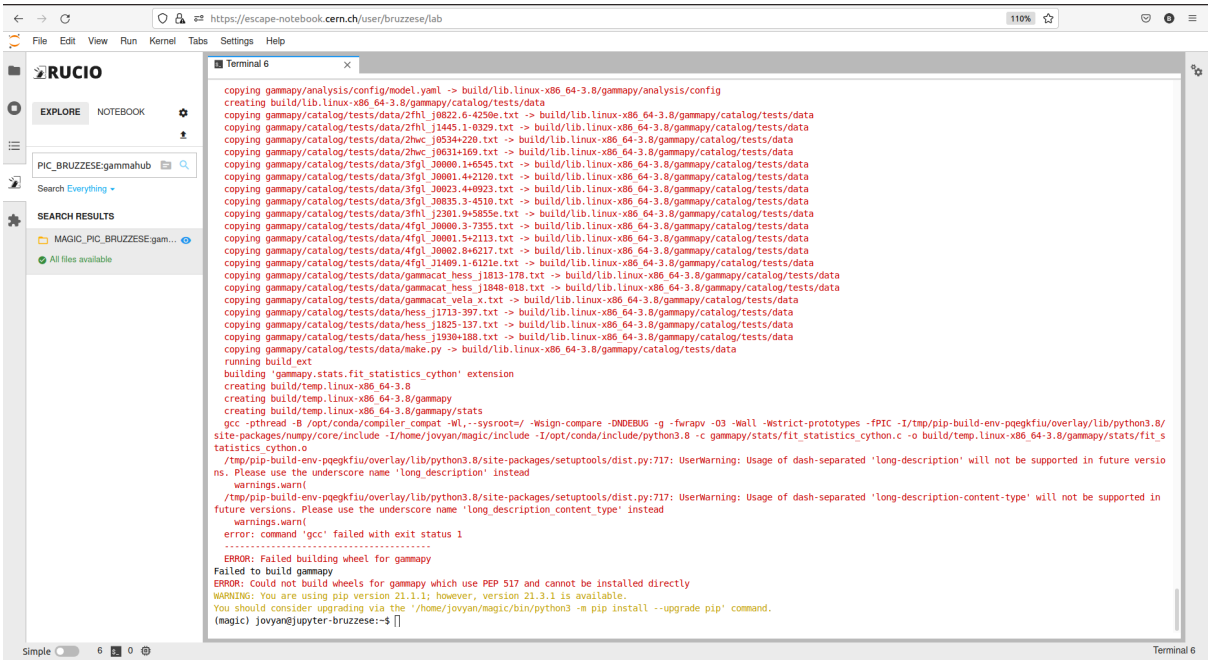
26/11/21 - Usecase 2:
For the magic usecase2, we could not get the files earlier, because the simulation of these has to be done with specific values. So we received them today. We have uploaded dl3 simulated files to datalake (on 26/11/21).

Specifically, the destination RSE has been PIC-DCACHE. The upload of the files has been successful.

Once inside the jupyter (DlaaS), we were able to source the dataset containing the dl3 files as expected.

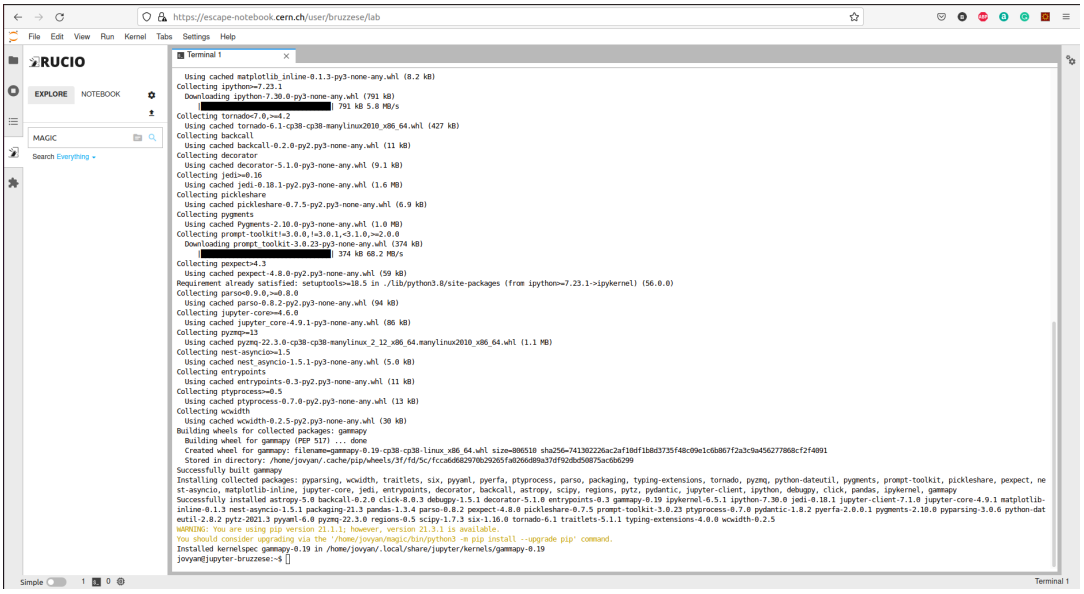


The drawback that prevented us from completing this test is due to the lack of gcc and was not possible to install with pip or by python setup install of the tar.gz. The gammapy software on a DLaaS instance required gcc.



(magic) jovyan@jupyter-bruzzese:~/magic$ gcc --version
bash: gcc: command not found

30/11/21 - Alba V. solved the problem, now installing gammapy is possible.

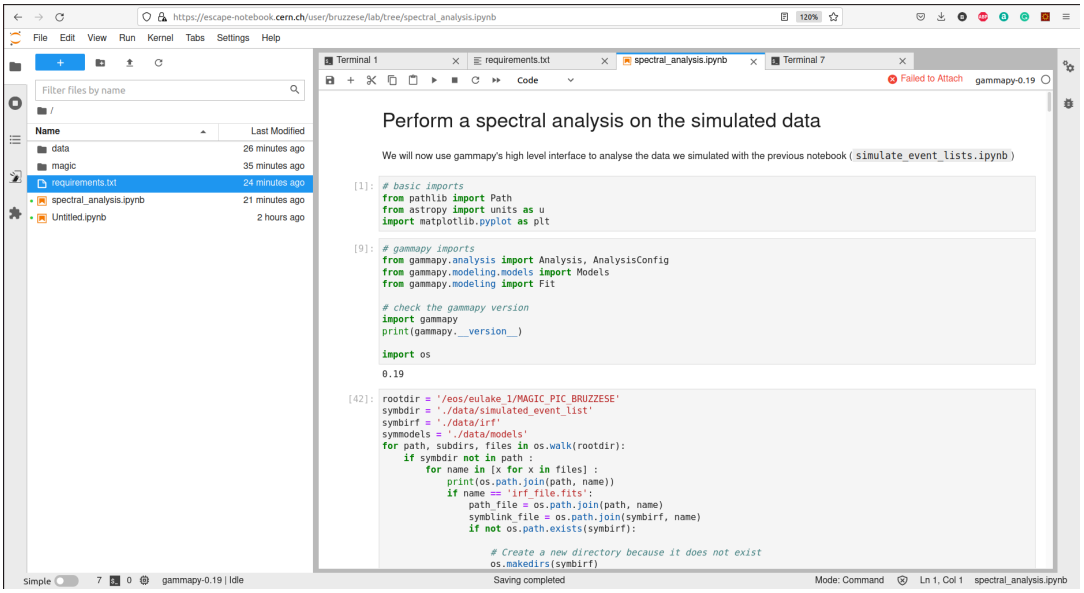


1/12/21 -

During these past days, we wanted implement a use case associated with GammaHub, an initiative that IFAE-PIC is developing in the context of WP5 and that aims to offer interactive analysis tools using the Gammapy package and the emerging standard in Gamma-ray astronomy, the DL3 format files.

These files were simulated following strict guidelines, otherwise the notebook would not work.

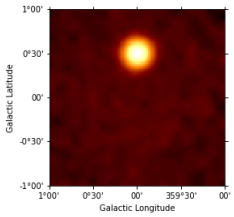
As explained above, the method by which they were uploaded to the datalake was by a simple upload of local files to a deterministic rse of the ESCAPE datalake.
Once done, they were logged into the https://escape-notebook.cern.ch application via openid and the files were made accessible in EULAKE1.



[37]:
```
# let's check just the first observations
counts = analysis.datasets["stacked"].counts
counts.smooth("0.05 deg").plot_interactive()
```

Select energy: 1.00e+02 GeV - 1.58e+02

Select stretch: linear / sqrt / log

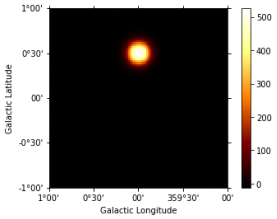


[38]:
```
# we can also compute the significance of our source
analysis.get_excess_map()
analysis.excess_map["sqrt_ts"].plot(add_cbar=True);
```

```
Computing excess maps.
Position <SkyCoord (Galactic): (l, b) in deg
    (0., 0.)> is outside valid IRF map range, using nearest IRF defined within
```


[38]:
```
# we can also compute the significance of our source
analysis.get_excess_map()
analysis.excess_map["sqrt_ts"].plot(add_cbar=True);
```

```
Computing excess maps.
Position <SkyCoord (Galactic): (l, b) in deg
    (0., 0.)> is outside valid IRF map range, using nearest IRF defined within
```

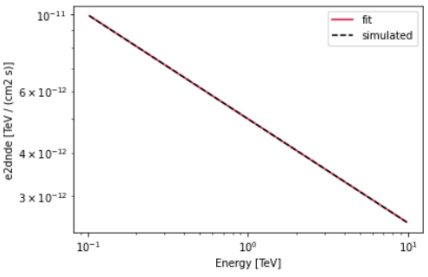



perform the fit

As a final step we fit the spectrum of the source, and we compare to the one we actually used for simulation

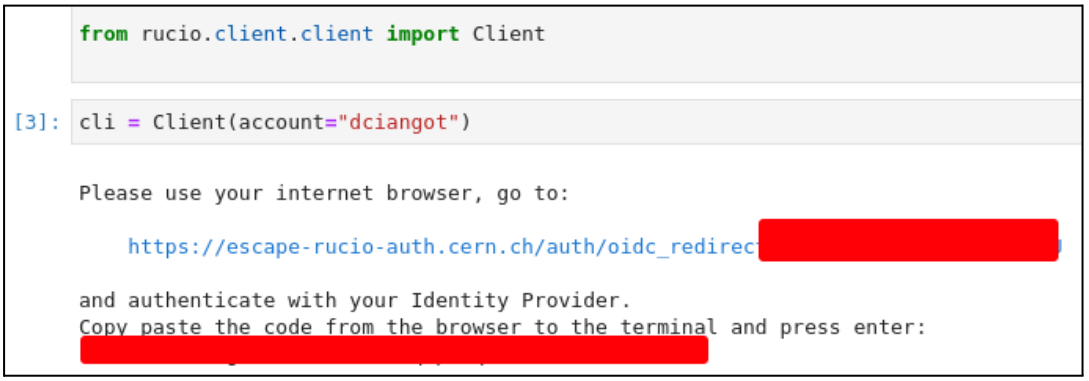
[43]:
```
# let us load the model we used for the simulation
models = Models.read("./data/models/point-source-pwl.yaml")
# let us create a copy of the spectral model for later comparison
original_spectral_model = models[0].spectral_model.copy()
```

[47]:
```
plot_kwargs = {
    "energy_bounds": [0.1, 10] * u.TeV,
    "sed_type": "e2dnde",
    "yunits": u.Unit("TeV cm-2 s-1"),
    "xunits": u.TeV,
}

analysis.models[0].spectral_model.plot(**plot_kwargs, color="crimson", label="fit")
analysis.models[0].spectral_model.plot_error(**plot_kwargs, color="crimson", alpha=0.3)
original_spectral_model.plot(**plot_kwargs, label="simulated", color="k", ls="--")
plt.legend()
plt.show()
```



- Finished successfully MAGIC usecase 2

LOFAR



Use Case 1: Ingestion and replication

The goal of this use case is to simulate a realistic ingest work flow comparable to what we now do in LOFAR for a single Long Term Archive. The goal was to upload the data to a non-deterministic RSE, register it in Rucio and apply relevant life cycle rules. The first rule is to move the data to a quickly-accessible QOS level (in the current LOFAR design this would be dis k) and keep it there for a set amount of time (e.g. a week) so that users could download the data that was recently observed for them. The second rule is to also copy the data on a SAFE QOS-level tier (in the current LOFAR design this would be tape). In a realistic scenario the data will remain there indefinitely, but for DAC21 we applied a lifetime that is longer than the disk life time (e.g. more than a week). In the ideal case this use case should be fully executed using OIDC authentication, but we would accept the access to the storage by X509 if this would have been the only blocker.

We started by using **PIC-INJECT** (several MB/s max; leading to transfers of 60GB-sized files of several hours) as the non-deterministic RSE in our workflow. It was however quickly clear that the network connectivity was not adequate to support the data sizes we were transferring.

The network connectivity to DESY was good, so in order to solve the problem, a non-deterministic RSE at DESY (DESY-DCACHE-NDR) was set up (Thanks to Paul). The use case was attempted there. We uploaded our files there. The bandwidth to DESY was much higher making the files transferred in ~10 minutes. After fixing some minor AAI issues that made OIDC access impossible we were able to execute the full use case successfully.

(datarate achieved ?, amount of data transferred? were the lifecylce rules placed?)

The code that we have generated to automate this use case can be found here: https://git.astron.nl/astron-sdc/escape-wp2/dac21-ingest

Conclusion

We have not been able to simulate the upload of a large data set but the proof of concept that the use case is doable is definitely successful (shall we attempted in near future). Also we have been able to execute this use case using only OIDC-based AAI which was another aim.

Ideally a workflow would be adopted where the data is streamed through a tool that computes the checksum, into the storage. We have experimented with the tooling we use in LOFAR, combined with rclone. However we found some behaviour of the AAI that sometimes just made the token invalid that made it very hard to upload too many files. Also the rclone streaming seems to be buffering significant data on disk, which on one hand seems to defy the purpose of streaming at all, and on the other hand leads to issues on the node we used for upload (due to lack of disk space). This disk-space problem will be partly mitigated as we have now another node as a resource. Thus parallel transfers from two nodes with additional diskspace can also be attempted. Although buffering of data on the disk is not a preferred scenario in a streaming case.

However all in all, the main goals of this use case have been achieved. It would still be nice to actually do a full-scale experiment to see if some of the issues we found could be mitigated, but we consider that a post-DAC21 exercise.

- LTA figure?
- We have been monitoring system statistics all week so we should be able to plot any performance number we mention here…

Use Case 2: Data upload, (lifecycle rules), Data retrieval, Processing, Archiving the results. This includes demonstrating ability to group scientifically related observations using datasets and containers (hierarchical if necessary) and utilize them in pracitce.. For Astronomy use case this is quite an useful aspect. Apart from its usefulness in practical day to day processing, this concept may have the potential to address making required changes to a single observation only in the files where needed. (This aspect needs to be thought and deliberated further in future).
This usecase also combines some aspects of ESCAPE work packages 2, 4 and 5. (Step 9 onwards). This is by using ESAP, VO, DLaas notebook (including installing relevant softwares).

In this we had a total of 32 Subbands (SB along frequency) of an Observed Data. Total size is about 125GB (~3,9 GB per SB).

We did not create several replicas because all those capabilities had been demonstrated in the previous exercise of Data lake in 2020 in our use case.

The Observations are present at an external location.

Step 1: Group the observations in three groups of 12, 12 and 8 SBs. The idea is to treat the observations within the groups as a single entity when needed. This concept is useful as then the user can simply use a group to download all relevant data (or address in for other uses) rather than having to treat each file. It has also further subsequent usages later when we are dealing with calibrator observations (source which is needed for calibrating the target data) and target observations (source to be imaged). In principle there could be several calibrators observed at different times in different orders. Using the concepts of groups (dataset in rucio terms) and later containers one can logically use the relations so as handling and

processing of the data does become much more simpler, practical and logical to address in a real life case.

We call these three groups as:

```
LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB000_SB011_Set1
LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB012_SB024_Set2
LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB025_SB031_Set3
```

The nomenclature can be further improved but for this exercise this is sufficient.

Step2: The observations were uploaded as groups. So three groups.
--register-after-upload was useful in case things do go wrong during the upload process.
The upload was carried out for the three datasets (groups) separately. The upload did take a total of about 9mins.

After the successful upload we checked the contents:

```
rucio list-content  LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB000_SB011_Set1
+-------------------------------------------------------------+--------------+
| SCOPE:NAME                                                  | [DID TYPE]   |
|-------------------------------------------------------------+--------------|
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB000_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB001_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB002_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB003_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB004_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB005_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB006_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB007_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB008_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB009_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB010_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB011_uv.MS.15ch2s.dppp.tar | FILE       |
+-------------------------------------------------------------+--------------+
```

```
rucio list-content  LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB012_SB023_Set2
+-------------------------------------------------------------+--------------+
| SCOPE:NAME                                                  | [DID TYPE]   |
|-------------------------------------------------------------+--------------|
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB012_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB013_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB014_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB015_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB016_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB017_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB018_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB019_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB020_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB021_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB022_uv.MS.15ch2s.dppp.tar | FILE       |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB023_uv.MS.15ch2s.dppp.tar | FILE       |
+-------------------------------------------------------------+--------------+
```

```
rucio list-content  LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB024_SB031_Set3
+-------------------------------------------------------------+--------------+
```

```
| SCOPE:NAME                                                    | [DID TYPE]   |
|---------------------------------------------------------------+--------------|
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB024_uv.MS.15ch2s.dppp.tar | FILE         |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB025_uv.MS.15ch2s.dppp.tar | FILE         |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB026_uv.MS.15ch2s.dppp.tar | FILE         |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB027_uv.MS.15ch2s.dppp.tar | FILE         |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB028_uv.MS.15ch2s.dppp.tar | FILE         |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB029_uv.MS.15ch2s.dppp.tar | FILE         |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB030_uv.MS.15ch2s.dppp.tar | FILE         |
| LOFAR_ASTRON_PANDEY:L432696_SAP000_SB031_uv.MS.15ch2s.dppp.tar | FILE         |
+---------------------------------------------------------------+--------------+
```

Step 3: The rule to limit the lifetime of about 10 days was added, and activity flag DAC21 was added as well.

Step 4: The three datasets were grouped together as a container and later we checked the contents.

```
rucio list-dids  LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB000_SB031_Set1-2-3
--recursive
+-------------------------------------------------------------------+--------------+
| SCOPE:NAME                                                        | [DID TYPE]   |
|-------------------------------------------------------------------+--------------|
| LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB000_SB011_Set1    | DATASET
|
| LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB012_SB023_Set2    | DATASET
|
| LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB024_SB031_Set3    | DATASET
|
| LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_SB000_SB031_Set1-2-3 |
CONTAINER    |
+-------------------------------------------------------------------+--------------+
```

Lifetime and activity flags were appropriately added.

Step 5: (External location and independent user needs to process this data). First step is to download/retrieve it from data lake.

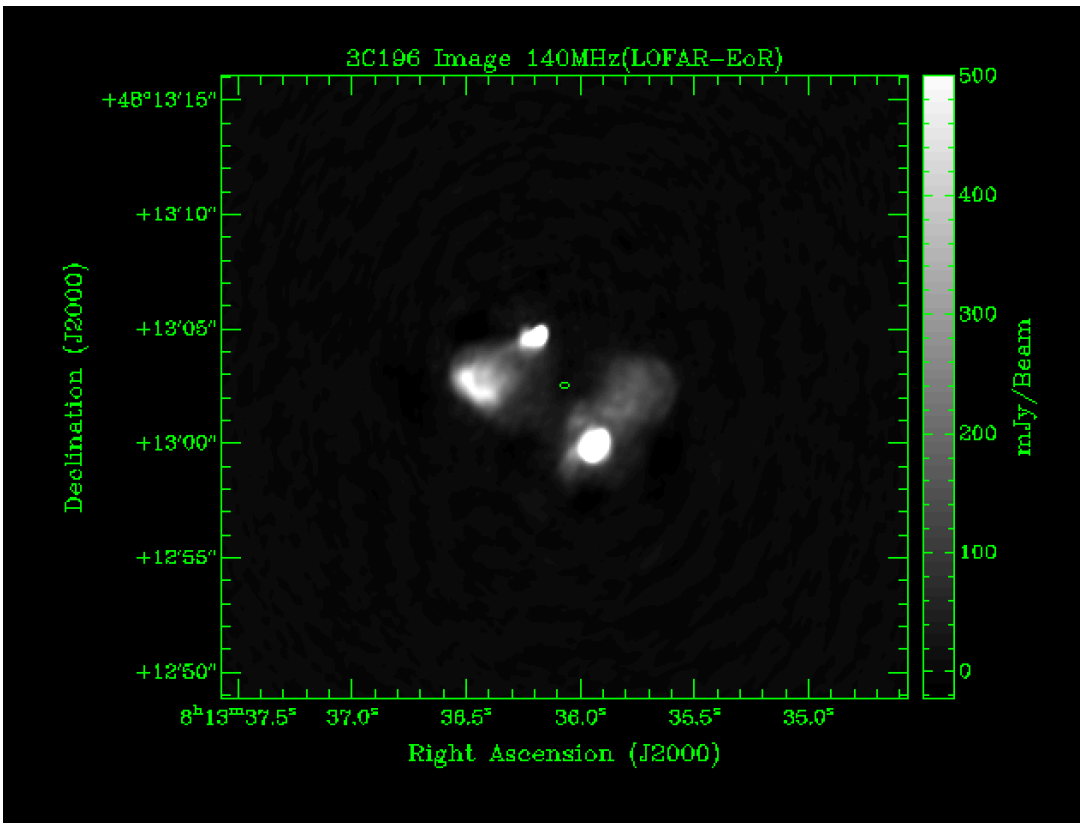
Now from an external location a user wants to process data. He has all the required data in the container. He is given the name of the container which as all the data related to this observations. He can simply download the container with all its associated data without being aware of all the details of the observations. In this example while we have a quite simple grouping, in principle we could have a complicated relationship but all that can be consumed by careful grouping of the datasets including using the hierarchy of relations.

```
rucio -vvv download LOFAR_ASTRON_PANDEY:UseCase2_Dataset_3C196IBL_
SB000_SB031_Set1-2-3
```

The download step took about 9min20seconds.

Step 6:

The next step was to process this data by the user at an external location. This was done using the Docker based software image.



Processing on an external location : 2hrs 10min. The high resolution 3C196 image shown above is using the observations which were carried out including the international baselines of LOFAR (maximum baseline ~2000km). The beam size is shown as a green ellipse at the center which clearly shows that 3C196 (which is about 5 arc seconds, is so well resolved.

3C196 is a quasar, the central core (which is usually in the between the radio components) is not visible in radio observations but in optical or higher frequency images. One of the reasons could be that the central engine emitting mostly at shorter wavelengths due to high energies where as the old electrons (in the outer lobes) emitting mostly synchrptron radiation in higher wavelengths.

Step 7: The resulting image(s) was archived back to the datalake as a dataset.

LOFAR_ASTRON_PANDEY:UseCase2_Imageset_3C196IBL_SB000_SB031_Set1-2-3

Step 8: The image was also attached to the container of the original data set.

Step 9: For this step we wanted to demonstrate combining the results (data) in the data lake with public data from elsewhere.

For this we first used ESAP to locate the data in the data lake and added it to the shopping basket. Then, we used external tooling to find optical data (from the Hubble Space Telescope) in the Virtual Observatory and sent the table to ESAP, which places it in the shopping basket too. In essence, this use case combines ESCAPE work packages 2, 4 and 5. This is an useful aspect of this exercise.

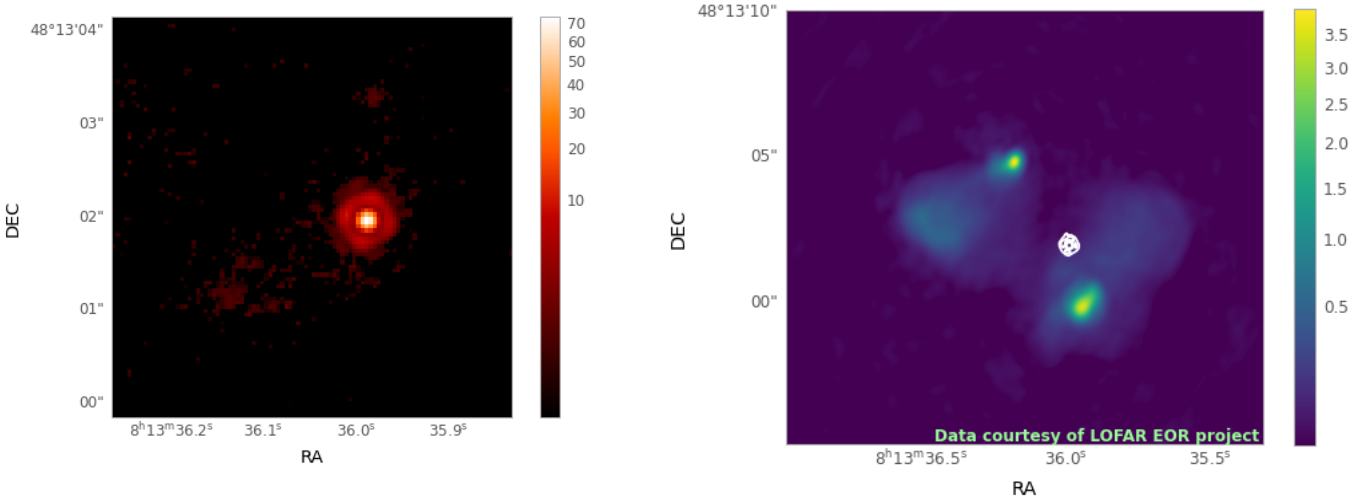
The next step was to open the DLaaS notebook and install both the shopping basket client and the astropy packages that can be used to process FITS images.

We then read out the data from the shopping basket and staged the data from the Datalake to the right location so that it would be accessible from within the notebook.

The notebook can be found on this repostory:
https://git.astron.nl/astron-sdc/escape-wp2/dac21-imagecomb

The first cell of the notebook installs the relevant software after which the function is defined that can be used to download the data into the right RSE for processing. Then several steps are taken to read the two data sets from the shopping basket, bring the Rucio data online and download the data from the Virtual Observatory. We then use the astropy tooling to stack the optical images (to increase the signal to noise ratio), genrate contours and apply the contours to the image downloaded from the data lake.

The last step is taking the png image and uploading it back into the datalake.
The figure shows the optical image on the left, and the combined image (LOFAR with optical contours) on the right.



Conclusion:
This use case was very successful. The Data upload along with adding relevant lifecycle rules, Data retrieval, Processing, Archiving the results was successfully carried out. This also included demonstrating ability to group scientifically related observations using datasets and containers (hierarchical if necessary) and utilize them in a practical manner. Thus another potential interested user can simply access the entire relevant information from all connected raw observations, upto the end resulting images in a straight forward manner (a single command so to speak). The integration between ESAP en DLaaS demonstrated possible although

more thought is needed to explore what would be an ideal integration. Being able to stage data for usage in the DLaas notebook directly from the shopping basket, basically having the code written for that either in the DLaaS environment or in the shopping basket handler. Also, the query functionality of ESAP could be based more on the metadata of the underlying data for specific scopes. This could for instance make the datalake a backend for a multi-archive query in ESAP for positions on the sky in case of astronomical instrumentation.
In future, It may also be nice/useful to demonstrate that the DLaaS can be deployed at multiple sites (with different storage backends) and that multiple ESAP installations could communicate with multiple DLaaS deployments.

- Could add screen shots of the notebook

Use Case 3: Legacy archive, public data access RSE
TBD: This depends a bit on the actual feasability of connecting an RSE from external VO in read-ony mode,
We have not been able to execute this (optional) use case within DAC21. It needs more time and and will be a post-DAC21 activity.

LSST

Infrastructure “bootnotes”

Robustness

On 2021-11-22 (Monday) several problems were found with the dCache instance at CC-IN2P3. These came predominantly from the relatively small size of the LSST files (many are approx 10 MiB). Almost all transfers failed. There were two main failure modes. The first failure mode allowed FTS to recover (using HTTP-TPC PUSH from EULAKE-1), the second failure mode meant FTS could not recover. This gave a somewhat distorted picture: the data was being transferred, but not without problems.

With the help from the IN2P3-CC admins (Adrien) and dCache developers (Paul) several hot fixes were developed to address these problems and deployed at CC-IN2P3 on Monday.

The problems fixed were:

- Removed bottleneck caused by mutex/monitor that resulted in the door only processing one incoming request or transfer finalisation at a time. Now, any number of new transfers (up to configured limits) and any number of transfer finalisation (up to configured limits) may be processed.
- Move transfer finalisation process off of the dCache-internal message thread. This is to avoid blocking the message-processing thread, allowing greater throughput.
- Fix bug that allowed the transfer status to change from failed to success, if dCache-internal message timeout coincided with the transfer completing

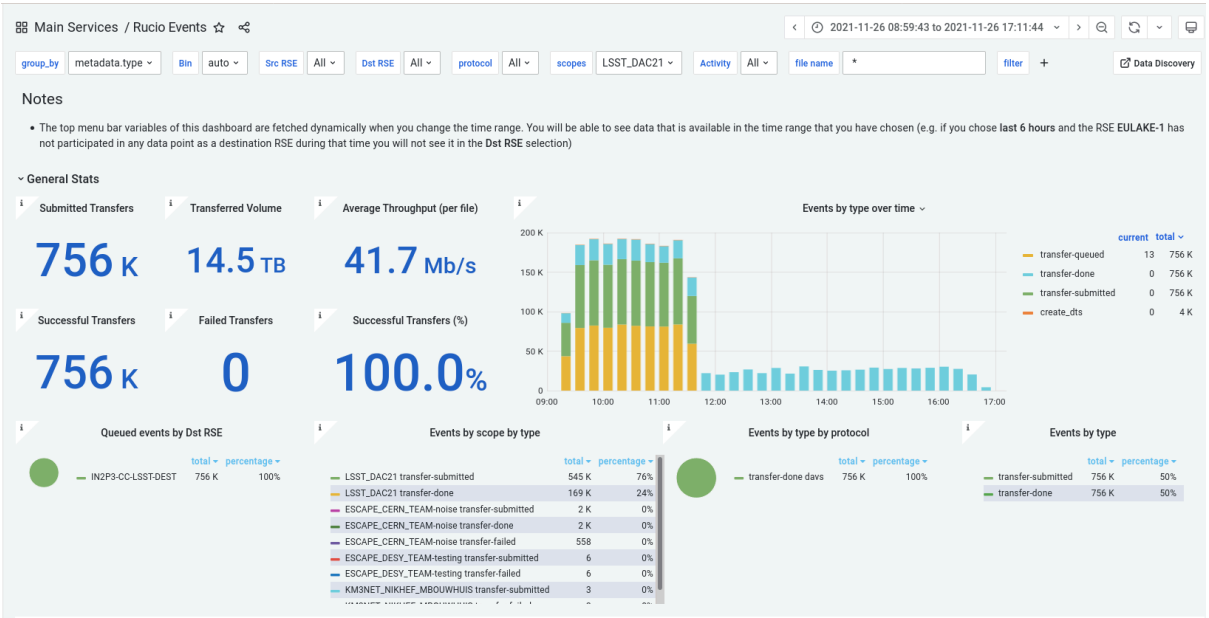
successfully. This is normally an extremely rare problem, but was triggered by the large build-up of dCache-internal messages.

- Fix race condition between WebDAV door and transfer manager. This is normally an extremely rare problem, but was triggered by a build-up of dCache-internal messages.

During 2021-11-23 (Tuesday), the majority of transfers succeeded; however, a non-trivial fraction of the transfers were still failing. These failures followed the first failure mode, from which FTS could recover (issuing HTTP-TPC PUSH request to `EULAKE-1`). Therefore, the Rucio monitoring dashboard showed ~100% success rate, despite the failure. During Tuesday, two new patches were developed to improve the throughput and fix the final set of errors.

With these further patches deployed, dCache provided (near) 100% successful transfer rate. Of some 588K transfers, only 99 (that's 0.02%) failed with an error. dCache reported these errors as a problem requesting data from the source server (`EULAKE-1`), claiming the server returned `500 Internal Server Error` in response to dCache's GET request.

By Wednesday, transfers were 100% successful at IN2P3 and FTS was no longer using HTTP-PUSH requests to `EULAKE-1` as a recovery procedure.



The final transfer campaign, on Friday (2021-11-26) from CERN (EULAKE-1) to CC-IN2P3 (IN2P3-CC-LSST-DEST).

Performance

One of the hot patches (developed and deployed on Tuesday) was to address the low performance issue. This yielded a noticeable performance improvement (values?).

dCache was configured to support 300 concurrent HTTP-TPC transfers. However, manual inspection suggests this number was never realised in practice.

The small files result in short transfer time. This means that internal latencies within dCache *may have* dominated the overall performance; however, this point is conjecture and further investigation is needed. However, it seems likely that at least twice the supported transfer are needed in order to keep the pools "busy": one transfer is "active" while the other is "queued up", ready to replace the currently active transfer once it completes. Moreover, due to the latencies, the optimal number of concurrent transfers may be more than this.

The FTS queue limited the number of concurrent transfers to 100. Given the latencies involved and the short transfer times for these files, this value was too low. Unfortunately, this was discovered too late to allow the FTS configuration to be updated.

Currently, it's not clear what may be done within dCache to reduce latency: this is the subject of future investigations. A first step in this direction would be to enable better measurement of the various latencies and introspection from where they are coming.

In a related note the combination of Rucio and FTS appeared to be unable to maintain the configured 100 concurrent transfers within dCache, despite having a considerable back-log of activity. Manual checking the number of active transfers (some 28 times) showed an average of 52 concurrent transfers, but with considerable variance. At several times there were zero active transfers. At one time there were 211 concurrent transfers.

Concluding remarks

Leading out of DAC21, all but one of the dCache "hot fixes" have passed through code-review and will be included in the next bug-fix dCache release cycle, scheduled for 2021-11-30.

Additional patches to improve monitoring are also being developed.

ATLAS

- 22/11/21
    - Testing more uploads and transfer of datasets in the DataLake
    - Issues in the LAPP-WEBDAV site will need more replications than those planned
        - Solution: move samples to other sites
        - Experts in QoS meetings helped me to make correct decisions and know how to move between RSEs
    - Re-installing updated personal certificate. Following the docs. All good as a new-normal user.

        - But we need to add more details in the docs when you want to renew a certificate.
- 23/11/21
    - *<information to be inserted between the 29/30 Nov 2021>*
    - *<activity>*
    - *<RSE used>*
    - *<issues>*
- 24/11/21
    - *934 8799 4617<information to be inserted between the 29/30 Nov 2021>*
- 25/11/21
    - *<information to be inserted between the 29/30 Nov 2021>*
- 24/11/21
    - *<information to be inserted between the 29/30 Nov 2021>*

CMS

22-11-21:
Opendata loaded in the datalake already as preparation step - **SUCCESS no problem to report**

Name	Account	RSE Expression	Creation Date	Remaining Lifetime	State
CMS_INFN_DCIANGOT:ESCAPE-CMS-Opendata	dciangot	CNAF-STORM	2021-12-03T08:58:26.000Z	90d	OK
Name	Account	RSE Expression	Creation Date	Remaining Lifetime	State

Name	Account	RSE Expression	Creation Date	Remaining Lifetime	State
CMS_INFN_DCIANGOT:Run2012C_DoubleElectron.root	dciangot	DESY-DCACHE	2021-11-15T09:27:26.000Z	4d	OK
CMS_INFN_DCIANGOT:ZZTo4mu.root	dciangot	DESY-DCACHE	2021-11-15T09:22:13.000Z	4d	OK
CMS_INFN_DCIANGOT:ZZTo4e.root	dciangot	DESY-DCACHE	2021-11-15T09:21:52.000Z	4d	OK
CMS_INFN_DCIANGOT:ZZTo2e2mu.root	dciangot	DESY-DCACHE	2021-11-15T09:21:26.000Z	4d	OK
CMS_INFN_DCIANGOT:SMHiggsToZZTo4L.root	dciangot	DESY-DCACHE	2021-11-15T09:20:34.000Z	4d	OK
CMS_INFN_DCIANGOT:Run2012C_DoubleMuParked.root	dciangot	DESY-DCACHE	2021-11-15T09:16:30.000Z	4d	OK
CMS_INFN_DCIANGOT:Run2012B_DoubleElectron.root	dciangot	DESY-DCACHE	2021-11-15T09:11:36.000Z	4d	OK
CMS_INFN_DCIANGOT:Run2012B_DoubleMuParked.root	dciangot	DESY-DCACHE	2021-11-15T09:09:17.000Z	4d	OK

Accessed opendata from the lake with CMS analysis workflow
- Jupyterlab notebook
    - Authentication to rucio via TOKEN - **SUCCESS no problem to report**
    - Reading and upload results via TOKEN - **SUCCESS no problem to report**

[7]:
```
file_list = []
for repl in cli.list_replicas([{"scope":"CMS_INFN_DCIANGOT", "name": "ESCAPE-CMS-Opendata"}], schemes=["davs"]):
    file_list.append(list(repl["pfns"].keys())[0])

print(file_list)
```

```
['davs://dcache-door-doma01.desy.de:2880//escape/wp2_rucio_testbed/desy_dcache/CMS_INFN_DCIANGOT/72/27/Run2012B_Double
ucio_testbed/desy_dcache/CMS_INFN_DCIANGOT/df/47/Run2012B_DoubleMuParked.root', 'davs://dcache-door-doma01.desy.de:288
12C_DoubleElectron.root', 'davs://dcache-door-doma01.desy.de:2880//escape/wp2_rucio_testbed/desy_dcache/CMS_INFN_DCIAN
esy.de:2880//escape/wp2_rucio_testbed/desy_dcache/CMS_INFN_DCIANGOT/64/e4/SMHiggsToZZTo4L.root', 'davs://dcache-door-d
ANGOT/0b/5f/ZZTo2e2mu.root', 'davs://dcache-door-doma01.desy.de:2880//escape/wp2_rucio_testbed/desy_dcache/CMS_INFN_DC
scape/wp2_rucio_testbed/desy_dcache/CMS_INFN_DCIANGOT/42/5d/ZZTo4mu.root']
```

[27]:
```
import os
os.environ["BEARER_TOKEN_FILE"] = "/tmp/token-rucio"
df = ROOT.RDataFrame("Events",
                     file_list[4:7])
```

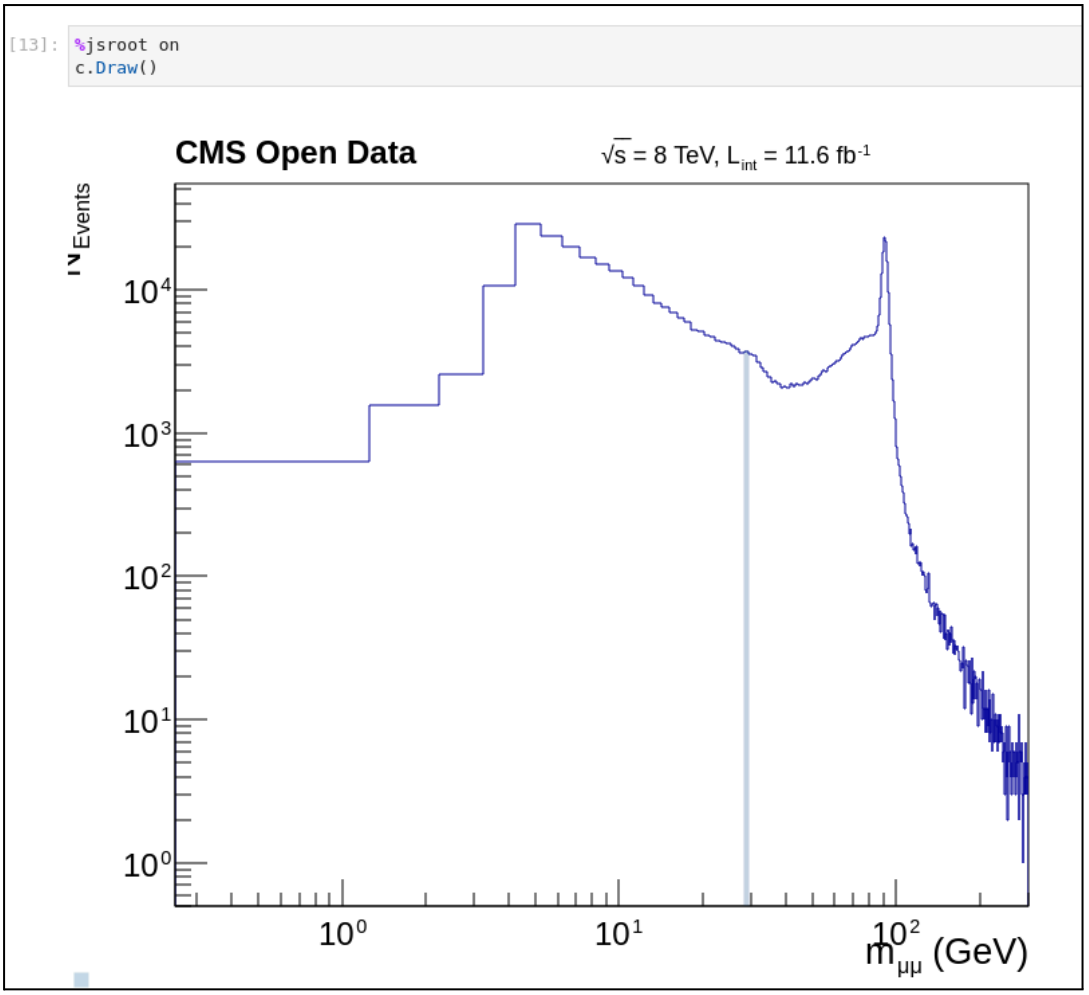
[9]:
```
df_2mu = df.Filter("nMuon == 2", "Events with exactly two muons")
df_oc = df_2mu.Filter("Muon_charge[0] != Muon_charge[1]", "Muons with opposite charge")
```

[10]:
```
df_mass = df_oc.Define("Dimuon_mass", "ROOT::VecOps::InvariantMass(Muon_pt, Muon_eta, Muon_phi, Muon_mass)")
```

[11]:
```
nbins = 300
low = 0.25
up = 300
h = df_mass.Histo1D(("Dimuon_mass", "Dimuon_mass", nbins, low, up), "Dimuon_mass")
```

[12]:
```
%%time

ROOT.gStyle.SetOptStat(0); ROOT.gStyle.SetTextFont(42)
c = ROOT.TCanvas("c", "", 800, 700)
c.SetLogx();
c.SetLogy()

# This triggers the distributed computation
h.SetTitle("")

h.GetXaxis().SetTitle("m_{#mu#mu} (GeV)")
h.GetXaxis().SetTitleSize(0.04)
h.GetYaxis().SetTitle("N_{Events}")
h.GetYaxis().SetTitleSize(0.04)
h.Draw()

label = ROOT.TLatex()
label.SetNDC(True)
label.SetTextSize(0.040)
label.DrawLatex(0.100, 0.920, "#bf{CMS Open Data}")
label.SetTextSize(0.030)
label.DrawLatex(0.500, 0.920, "#sqrt{s} = 8 TeV, L_{int} = 11.6 fb^{-1}");
```

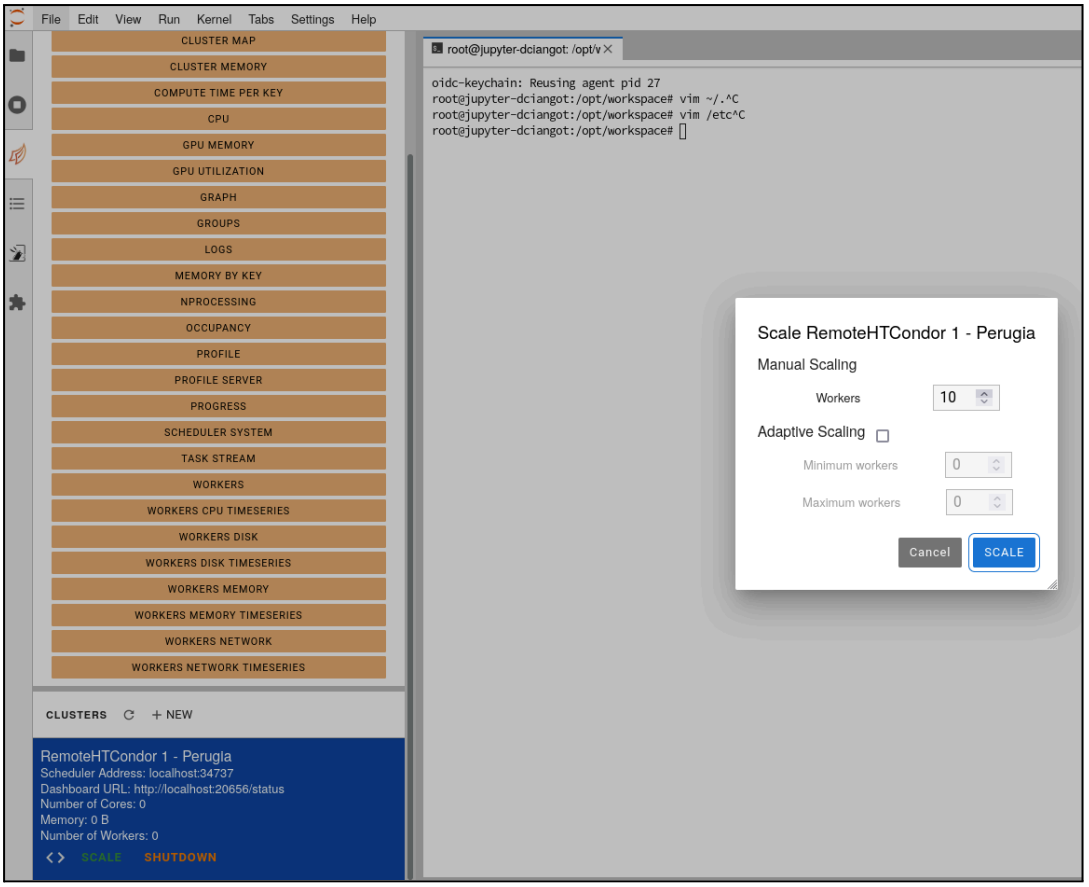
```
CPU times: user 16.1 s, sys: 928 ms, total: 17.1 s
Wall time: 33 s
```



23-11-21:

HTCondor test and scaleout:

- Reading via cache setup at cloud @CNAF - **SUCCESS no problem to report**
  - The cache communicate with the lake via x509
  - Client side trusts all connections coming from a HTCondor WN
- HTCondor job - **SUCCESS no problem to report**
  - Still work on getting a significant screenshot for this part
- Jupyterlab DASK interactively over HTCondor (10 nodes) - only a subset of the inputs - **SUCCESS no problem to report**

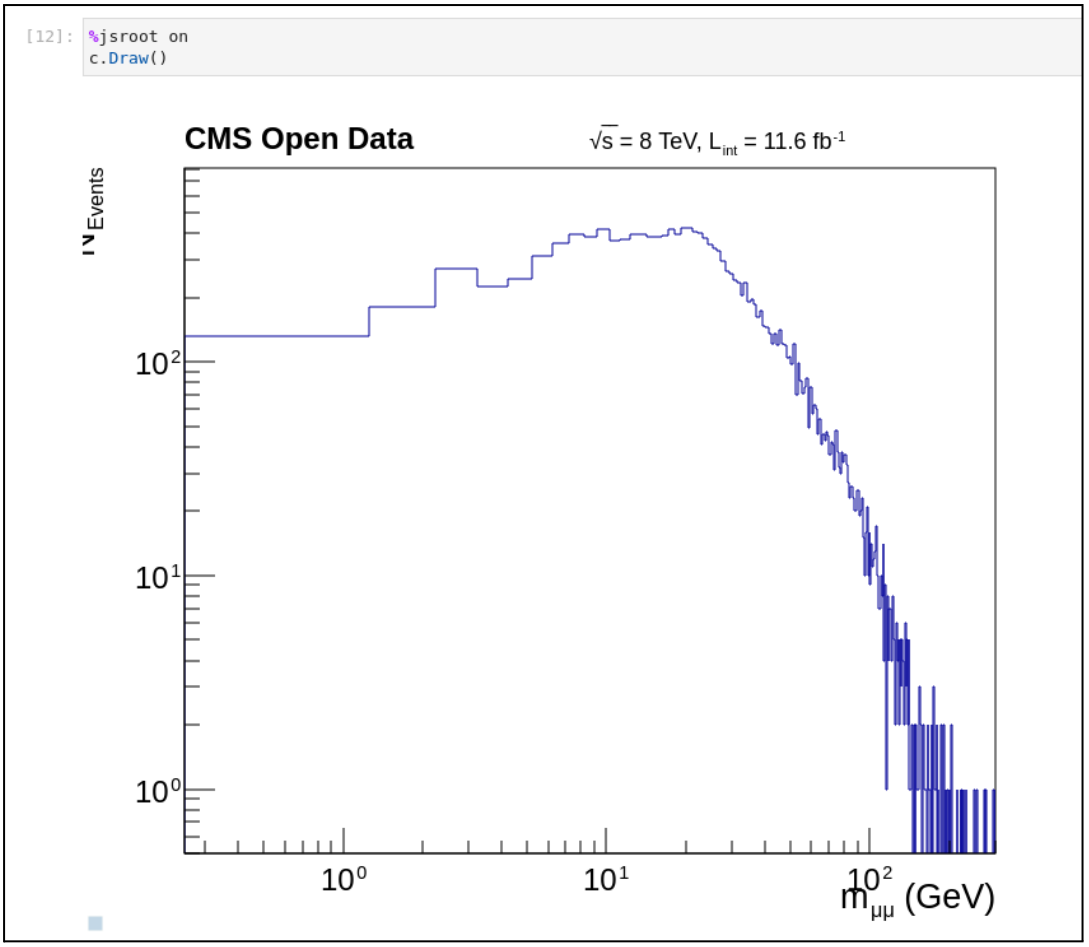



```
root@jupyter-dciangot:/opt/workspace# condor_q -totals


-- Schedd: 131.154.96.124.myip.cloud.infn.it : <131.154.96.124:31618?... @ 12/03/21 09:13:11
Total for query: 22 jobs; 11 completed, 0 removed, 1 idle, 10 running, 0 held, 0 suspended
Total for dciangot: 22 jobs; 11 completed, 0 removed, 1 idle, 10 running, 0 held, 0 suspended
Total for all users: 39 jobs; 28 completed, 0 removed, 1 idle, 10 running, 0 held, 0 suspended
```

Create a Dask client

First we will create a Dask `Client` object to connect to the cluster we will use in this example.

[2]:
```
from dask.distributed import Client

client = Client("localhost:33674")
client
```



- RUCIO upload results via TOKEN - **SUCCESS no problem to report**

<To be added>: a screenshot of jlab cell with the upload of the output file.

24-11-21:
Embargoed data

- Rucio upload of embargoed data to CNAF-STORM:
  - Using a dedicated scope CMS_EMBARGOED_DATA
  - Only IAM CMS group members can read and write in there
  - Via x509 - **SUCCESS no problem to report**
  - Via token - **SUCCESS no problem to report**

rucio upload --scope CMS_EMBARGOED_DATA --rse CNAF-STORM --lifetime 90000 --summary --name ZZTo4mu.root ZZTo4mu.root

Not completed during Dac week:
test on HPC Cineca. We are working on setting things up to be ready for a very quick functionality test.

EGO/VIRGO

22-11-21:

- Managed to configure X.509 certificate to get access to the datalake
- Set out plan to have data streaming from rucio to a Kafka producer to distribute the data to the Consumers

23-11-21

- Setup container to download data from Rucio
- Created another container to get the data from the above container and have them streamed out via Kafka Producer

24-11-21

- Attempted to resolve authentication issues in uploading data to the datalake which turned out to be missing certificates (see below)
- There were further issues in the authentication since it would not accept the grid certificate which was resolved by the creation of a proxy certificate
- Did a rucio upload of the required data

25-11-21

- Did run tests of the container that:
    - Downloads gravitational wave data from Rucio
    - Resizes the data so it can be streamed in 1 second intervals
    - Writes the data into a shared volume
- Another container was also setup to read the data from the above container and publish it via kafka to the consumers.

26-11-21

- Used a ROBOT certificate to enable the downloading of data from the datalake to a kubernetes cluster
- Setup the full wavefier pipeline from downloading data from the datalake to processing triggers running on the Kubernetes cluster at CNAF

CENTRAL SERVICES

- 22-11-21@10:20: IAM central services experiencing issues. Test robots having some failures since Saturday, points to some changes introduced by CI/Jenkins changes. Andrea and his team are chasing this up.
- 22-11-21@10:22: IAM is back up running, token-based testsuite still down, under investigation.
- 22-11-21@10:47: IAM testsuite problem understood (expired certificate used for the datalake X.509 testing). New certificate requested, testsuite should be back up as soon as the new cert is used.

- 22-11-21@16:25: Token-based authz testsuite is running again.

- 24-11-21: Missing certificates in EULAKE-1 grimap file, issue identified and solved manually
- 24-11-21: DLaaS local shared space for users was filled, we extended the space to 800GiB (from 500GiB) and the users cleaned their large files. A reminder to store any big files to the scratch space was given.
- 25-11-21: The addition of a new RSE exposed a bug in the CRIC-Rucio mapping, this was fixed relatively fast and the RSE was integrated
- General: Rules stuck in replicating state and in a deadlock state where they would never be accomplished, could be connected with FTS-pilot instance degradation, still inspecting. Fixed temporarily by changing the rule state to “STUCK” so that Rucio will try to repair them again.

PIC/IFAE

LAPP

22-11-21@10:10: Failures for X509 based transfers on LAPP webdav endpoint, token based transfers ongoing.

IN2P3-CC

22-11-21@10:45: IN2P3_CC_LSST: some failures on writing with “TRANSFER [2] DESTINATION CHECKSUM HTTP 404 : File not found”. Reminder to everyone not to use these RSE.
22-11-21@14:21: QOS of IN2P3_CC_LSST_* RSEs set to null so should not be used by anyone

GSI

CERN

22-11-21@10:15**:** Local RSE for notebook purposes showing up on the Data Lake transfers matrix. This RSE meant to local usage but popping up on the global matrix when an upload rule is created (maybe need to be filtered out for global DL data movement/replication purposes)

23-11-21@11:00: Repeated failures to upload files from LOFAR tests, eros shown in the singularity container running the workflow hinting to a failed x509 cert mapping. Root cause found: New robot certificate used by ASTRON not

propagated to EOS storage head node (xrootd) and the http and griftp doors.
Solution: grid mapfile sync scripts ran manually to force new certificate to be added into the /etc/grid-security/gridmapfile file.

-

DESY

2021-11-24 15:00:03+01 Installed an updated dCache RPM on one of the head nodes. This allowed the DESY-DCACHE RSE to fully support token-based authentication, as per the current test suite. There was a short interruption of a few minutes at this time.

SURF/SARA

RUG

The report of contribution of University of Groningen to DAC21 is discussed under the FAIR section since our use case, R3B, is part of the FAIR-GSI facility.

INFN

ASTRON