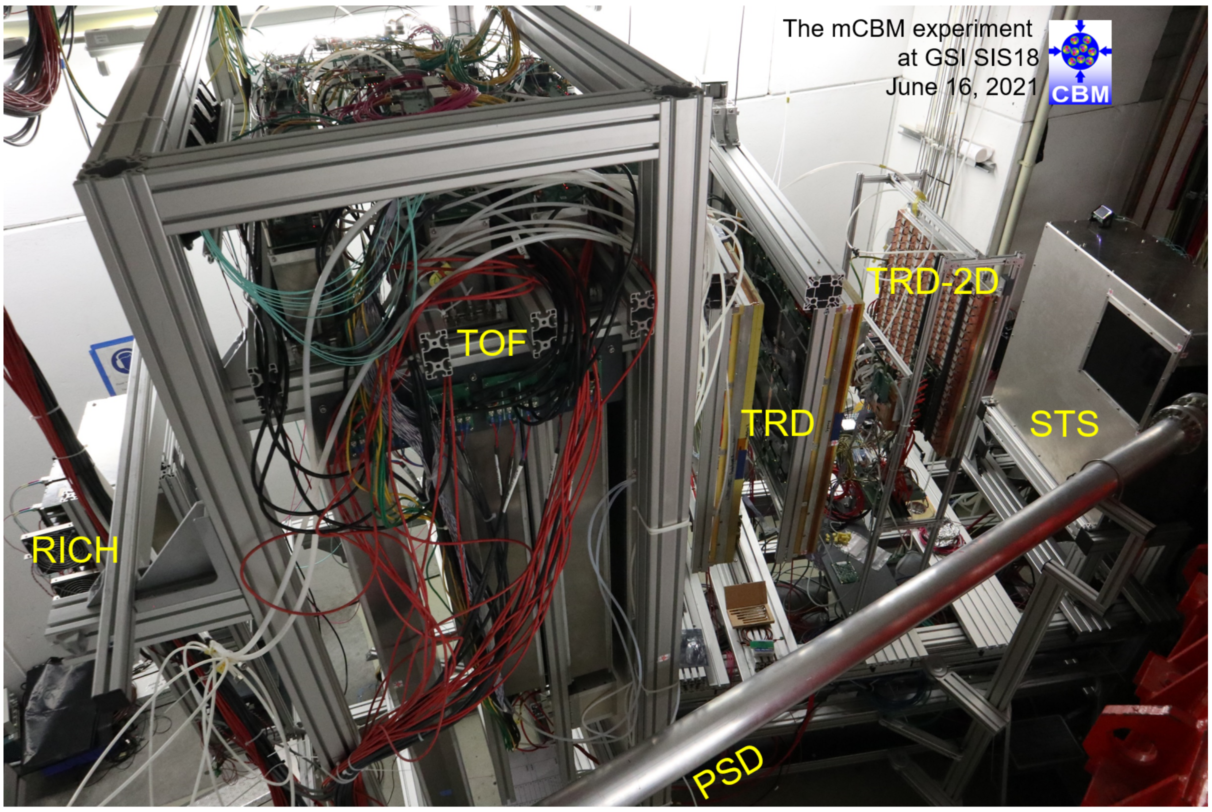
FAIR

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.

<u>Ingestion Side</u>

- 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

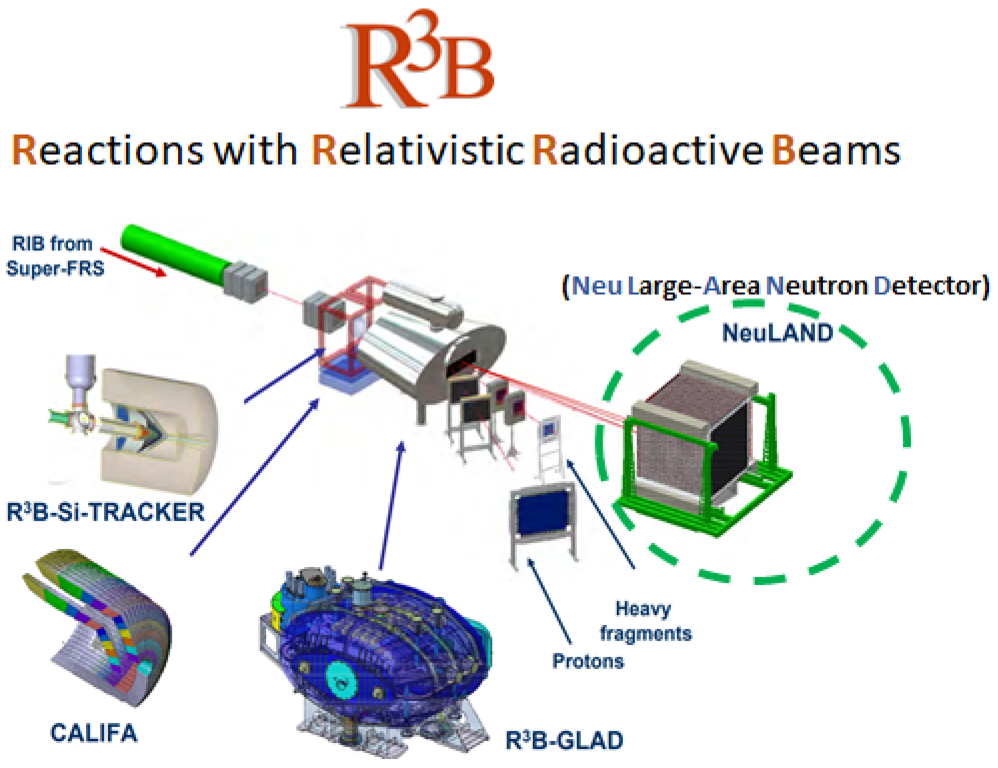
SUMMARY

- 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

R3B Ingestion (Maisam M. Dadkan)

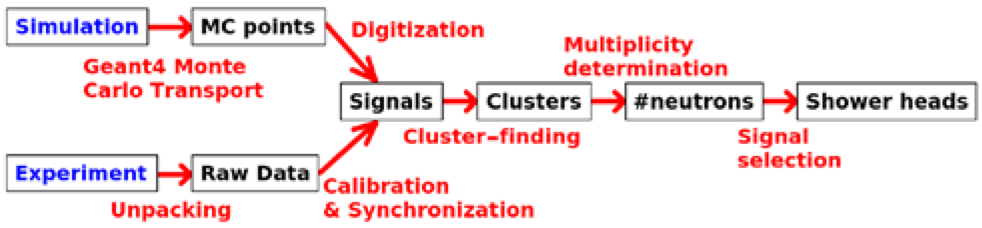

R3B
Reactions with Relativistic Radioactive Beams
RIB from Super-FRS
(Neu Large-Area Neutron Detector)
NeuLAND
R3B-Si-TRACKER
Heavy fragments
Protons
CALIFA
R3B-GLAD

[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.

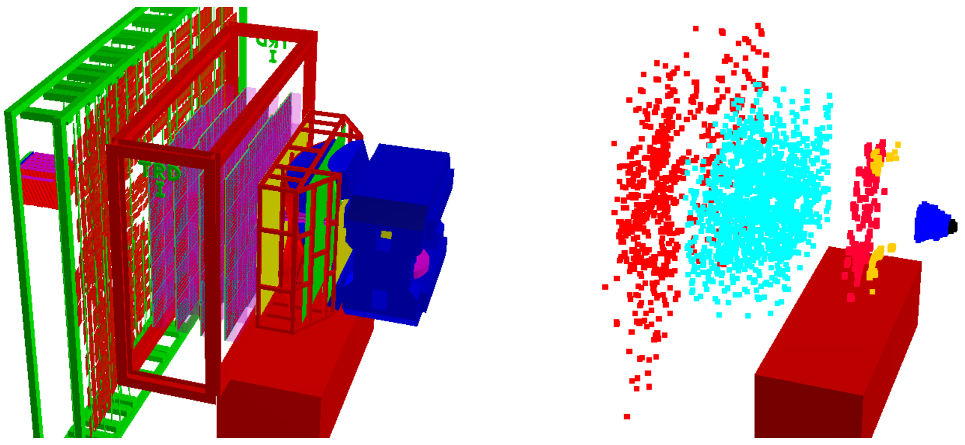
PANDA Ingestion (Ralf)
Panda simulations of background events at 15 GeV/c have been performed with the default detector setup. A total of 12500 jobs of 1000 events each were simulated, once in advance as fallback data and once in live mode. All data was sent from inside a container running on the Virgo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

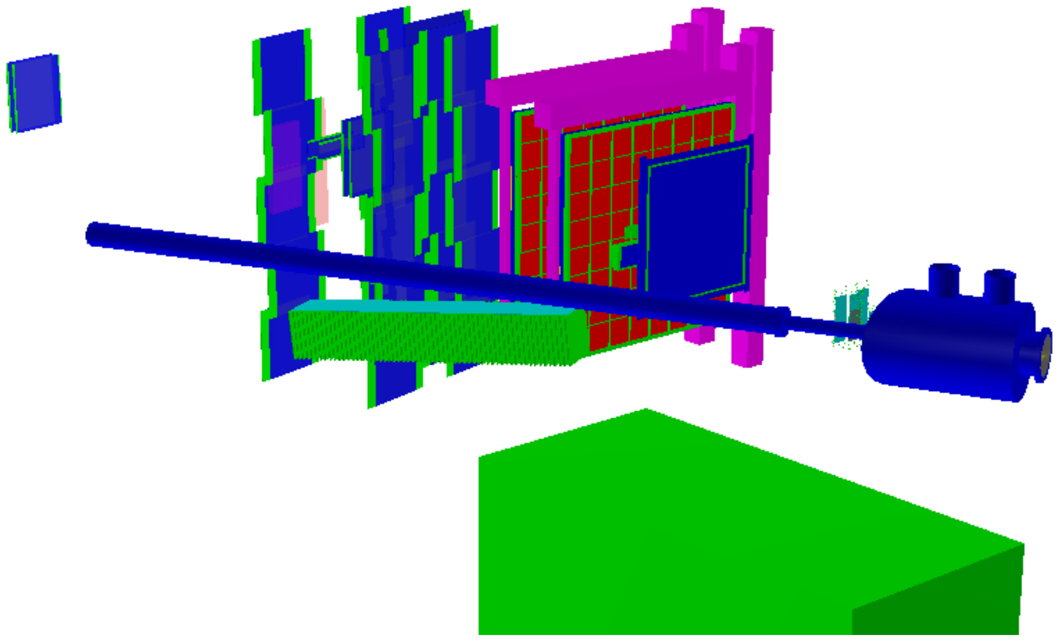
The goal was to have batch jobs read from the DataLake and upload the processed data.

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.

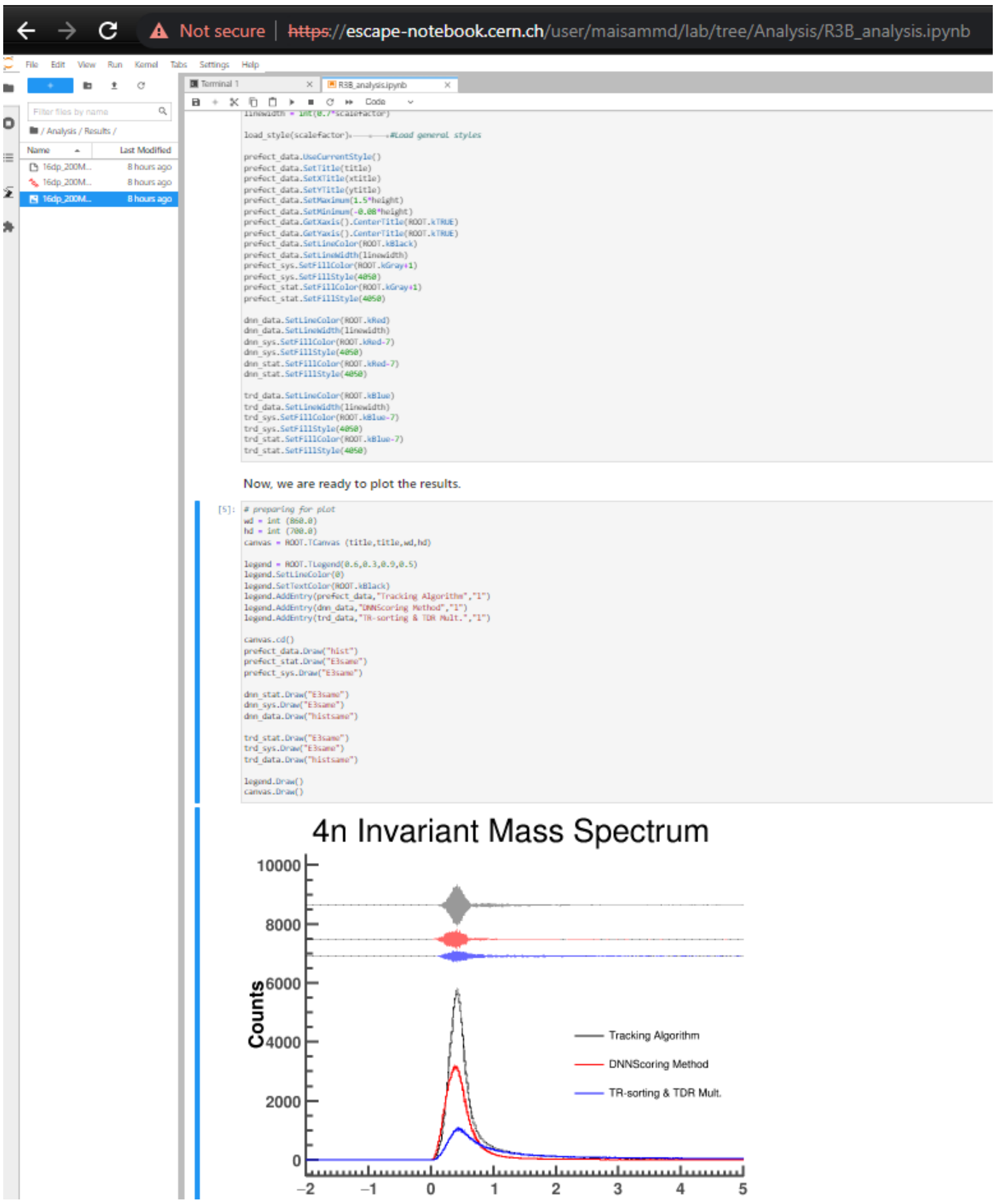
Statistics trial started at 21:04:50 and ended at 23:25:50. Of the 100 cases, 0 had ‘failure 1’, i.e. reading the raw mcbm data from the data lake using a rucio-get command, 75 had ‘failure 2’, i.e writing the reconstructed mCBM data the data lake using a rucio-upload command, and 25 were deemed fully 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.

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.



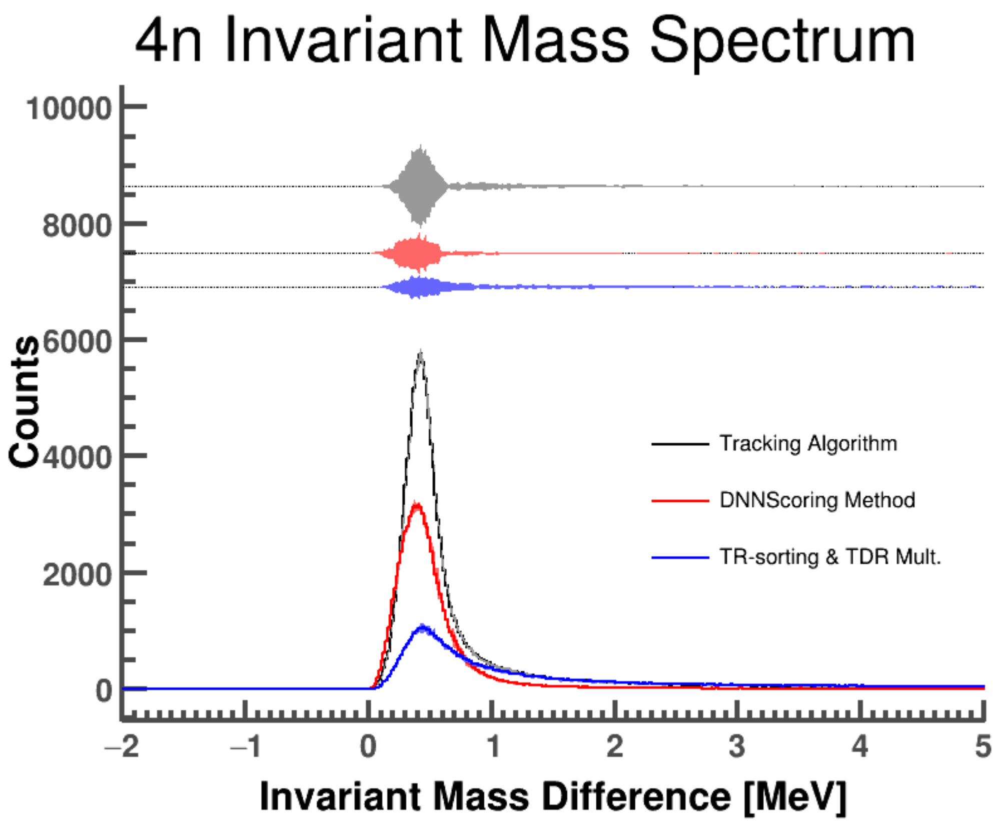
```
linewidth = int(0.7*scalefactor)

load_style(scalefactor)          #Load general styles

prefect_data.UseCurrentStyle()
prefect_data.SetTitle(title)
prefect_data.SetXTitle(xtitle)
prefect_data.SetYTitle(ytitle)
prefect_data.SetMaximum(1.5*height)
prefect_data.SetMinimum(-0.08*height)
prefect_data.GetXaxis().CenterTitle(ROOT.kTRUE)
prefect_data.GetYaxis().CenterTitle(ROOT.kTRUE)
prefect_data.SetLineColor(ROOT.kBlack)
prefect_data.SetLineWidth(linewidth)
prefect_sys.SetFillColor(ROOT.kGray+1)
prefect_sys.SetFillStyle(4050)
prefect_stat.SetFillColor(ROOT.kGray+1)
prefect_stat.SetFillStyle(4050)

dnn_data.SetLineColor(ROOT.kRed)
dnn_data.SetLineWidth(linewidth)
dnn_sys.SetFillColor(ROOT.kRed-7)
dnn_sys.SetFillStyle(4050)
dnn_stat.SetFillColor(ROOT.kRed-7)
dnn_stat.SetFillStyle(4050)

trd_data.SetLineColor(ROOT.kBlue)
trd_data.SetLineWidth(linewidth)
trd_sys.SetFillColor(ROOT.kBlue-7)
trd_sys.SetFillStyle(4050)
trd_stat.SetFillColor(ROOT.kBlue-7)
trd_stat.SetFillStyle(4050)
```

Now, we are ready to plot the results.

```
[5]: # preparing for plot
wd = int (860.0)
hd = int (700.0)
canvas = ROOT.TCanvas (title,title,wd,hd)

legend = ROOT.TLegend(0.6,0.3,0.9,0.5)
legend.SetLineColor(0)
legend.SetTextColor(ROOT.kBlack)
legend.AddEntry(prefect_data,"Tracking Algorithm","l")
legend.AddEntry(dnn_data,"DNNScoring Method","l")
legend.AddEntry(trd_data,"TR-sorting & TDR Mult.","l")

canvas.cd()
prefect_data.Draw("hist")
prefect_stat.Draw("E3same")
prefect_sys.Draw("E3same")

dnn_stat.Draw("E3same")
dnn_sys.Draw("E3same")
dnn_data.Draw("histsame")

trd_stat.Draw("E3same")
trd_sys.Draw("E3same")
trd_data.Draw("histsame")

legend.Draw()
canvas.Draw()
```

[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.*