# A Pilot Analysis Facility at CERN: Architecture, Implementation and First Evaluation

*Diogo* Castro[1], *Antonio* Delgado Peris[1], *Enrique* García García[1], *Giovanni* Guerrieri[1], *Ben* Jones[1], *Markus* Schulz[1], *Andrea* Sciabà[1], and *Enric* Tejedor Saavedra[1,*]

[1]CERN, Esplanade des Particules 1, 1211 Geneva, Switzerland

**Abstract.** Experiment analysis frameworks, physics data formats and expectations of scientists at the LHC have been evolving towards interactive analysis with short turnaround times. In preparation for HL-LHC the experiments are moving to data formats suitable for columnar analysis such as RNTuple. In addition, the Python ecosystem is becoming increasingly popular for analysis. Several sites in the community have reacted by setting up dedicated Analysis Facilities, providing tools and interfaces to computing and storage resources suitable for interactive analysis. It is expected that the demand for such facilities will increase towards the HL-LHC era and scaling out interactive processing of large datasets could become necessary.
CERN IT launched a Pilot of an Analysis Facility based on established, proven services such as SWAN, HTCondor and EOS. This facilitates the access to massive resources by enabling the use of HTCondor managed resources from SWAN, offering parallel execution via analysis interfaces, such as ROOT RDataFrame and coffea and their Dask back-ends.
In this contribution we will discuss the architecture of the Pilot Analysis Facility at CERN, giving the rationale for the decisions. For determining the next steps, evaluating the impact of different resource allocation strategies at the CERN HTCondor pool is critical. One especially interesting strategy consists in combining a set of dedicated resources for interactive analysis with the use of the general resources that are subject to experiment quotas. We will put a special focus on the feedback we received from the early testers from the experiments.

## 1 Context and Motivation

Less than 20% of the computing resources used by the LHC experiments correspond to user analysis. Unlike the other activities of the experiments, these use cases are - for most experiments - not centrally managed and are highly diverse, involving a large number of active users. Due to the more iterative approach to physics analysis, these use cases are sensitive to throughput and latency. The latter is especially relevant during the development and tuning phases of an analysis. CERN, being by far the largest site in WLCG, provides fast storage and hosts replicas of most of the analysis data. In recent years, within the scope of the preparation for the HL-LHC, the shift in analysis techniques towards data formats like RNTuple [1], supportive to columnar analysis, has started. Concurrently, analysis interfaces, like ROOT's

*e-mail: enric.tejedor.saavedra@cern.ch

RDataFrame [2] and coffea [3], have been introduced, allowing users to benefit from parallel processing. During the last decade, data analysis tools have been progressively optimised towards the use within a Python-based environment, and most students are already very familiar with this ecosystem when they arrive in the experiments. These tools can easily be used from interactive, Python-based, notebook systems, like Jupyter [4]. However, because of the data volume that has to be processed, at some stage of the analysis a shift towards batch processing is advisable.

The other fundamental change in the way analysis is conducted is related to the prevalence of the use of machine learning techniques, more specifically the use of libraries and tool chains originating outside the HEP community. These tool chains, like Keras [5] at a high level and PyTorch [6] at a low level, are most easily accessible within a Python-based environment and can benefit greatly for training and inference from access to modern GPUs.

Considering these changes, it became clear that the existing environments need to be extended to support the next generation of analysis. CERN is successfully operating a very large and mature computing infrastructure; therefore, any solution would benefit greatly from leveraging as much of the existing systems as possible.

To fully understand how an adequate production system, with a focus on analysis, should work, we decided to first build a pilot service, invite users and learn from their usage and feedback.

## 2 The CERN Analysis Facility Pilot

CERN provides both computing and storage infrastructure and a rich portfolio of services to support analysis. Some of these services are well established and have been in operation for many years: LXPLUS [7] for terminal-based interactive analysis, Batch [8] for batch analysis jobs, EOS [9] for mass storage and CVMFS [10] for software provisioning. Some other services are more recent additions and provide modern interfaces and tools to analysers: SWAN [11] for web-based interactive analysis, REANA [12] for reproducibility and workflow execution and Kubeflow [13] for machine learning workflows.

In this paper, we will focus on the integration of two services, SWAN and Batch, to scale out interactive analyses. This specific integration is what we will call the CERN Analysis Facility Pilot in this document, and it is described in detail in Section 2.1.

### 2.1 Architecture

The architecture of the CERN Analysis Facility Pilot is depicted in Figure 1. Users log in to SWAN via its web portal and are shown the JupyterLab [4] interface, which allows to open notebooks and web terminals. The SWAN user sessions (i.e. notebooks and terminals) run in dedicated resources in a Kubernetes cluster.

SWAN is integrated with the Batch service via Dask [14], a Python library for parallel and distributed computing, which can be leveraged by analysis frameworks such as ROOT RDataFrame and coffea to offload analysis computations to distributed HTCondor [15] resources. In order to exploit such resources, three steps take place:

1. The user creates a Dask cluster that submits jobs to HTCondor. This creation can either be done graphically, via the Dask JupyterLab extension [16], or programmatically from a notebook or terminal.

2. Resources are allocated in HTCondor and Dask worker jobs start running on them. Such workers connect back to the SWAN user session to notify that they are ready to accept tasks.

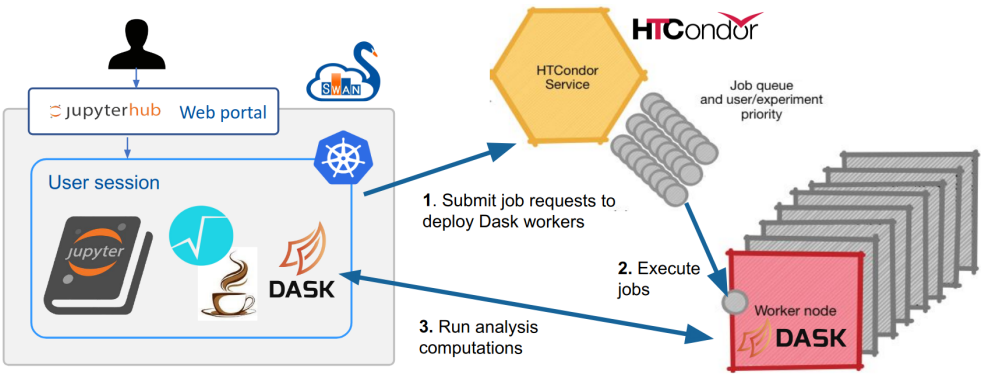


**Figure 1.** Architecture of the CERN Analysis Facility Pilot. The SWAN and the Batch services at CERN are integrated via Dask, which is used by analysis frameworks running in SWAN to allocate resources on the Batch side and submit analysis computations to those resources.

3. The user executes their analysis, which should be able to exploit the distributed HTCondor resources by offloading computations to the Dask workers.

The Pilot provides out-of-the-box access to the required software via CVMFS, including Dask, the HTCondor client and analysis frameworks (ROOT RDataFrame, coffea). The same software environment exists both on the SWAN side and on the HTCondor side to ensure the correct execution of the application.

Regarding data access, the input data for the analysis is expected to be read by the Dask workers either locally or remotely, for example from EOS via XRootD [17].

During the execution of the analysis, its progress can be monitored on the dashboard provided by the Dask JupyterLab extension. Multiple tabs with different kinds of information can be opened as part of such a dashboard: there is a task graph that represents the analysis workflow, progress bars for the different task types, a task stream that shows an execution trace of the analysis tasks, CPU / memory utilisation for the workers used, etc.

When the distributed execution finishes, the aggregated final results (e.g. histograms) are transferred back to the SWAN user session. This means that the user can display those results in the notebook and inspect them, and also store them on EOS (e.g. on EOS Home, which is the user's home directory in SWAN).

### 2.2 Resource Allocation

CERN's HTCondor batch system consists of more than 300K cores spanning diverse computing resources. It is used concurrently by hundreds of submitters for many different workflow types. The typical utilisation for analysis workloads ranges between 10% and 20%. As indicated, users can scale out from SWAN to HTCondor using the Dask library. A local Python package distributed with the Dask workers ensures that they are adapted to the specific configuration of CERN's HTCondor deployment.

To enhance interactivity, a dedicated HTCondor negotiator operates with a faster scheduling cycle, ensuring that short-lived and interactive jobs from SWAN are scheduled promptly. Moreover, a reserved fraction of around 1,200 dedicated cores within HTCondor is allocated specifically for SWAN-originated jobs, mitigating potential delays caused by contention with long-running batch workloads. Workloads exceeding this reserved buffer can allocate general HTCondor resources, subject to the standard fair-share policies in place.

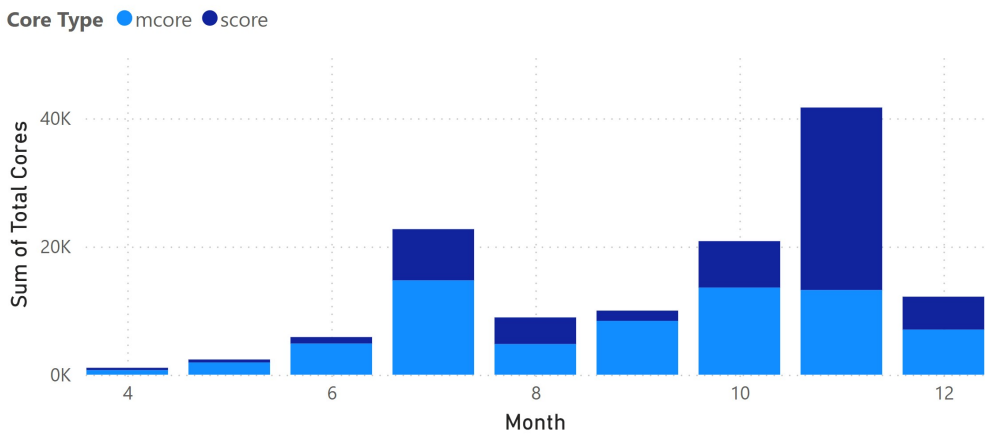


**Figure 2.** Total number of cores used by HTCondor jobs scaled-out from SWAN, by month and type of job (multi/single core).

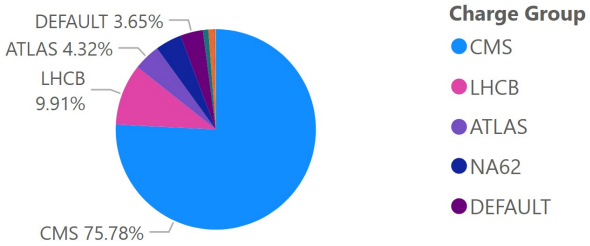


**Figure 3.** Total number of cores used by HTCondor jobs scaled-out from SWAN, by charge group.

### 2.3 Observed Usage

During 2024, more than 50 different users – distributed around the globe – have submitted jobs from SWAN to HTCondor. In total, 80K jobs were submitted, occupying 126K cores and consuming 7.92M HS23 walltime hours [18]. Figure 2 shows the evolution of utilised cores by month, while Figure 3 shows the distribution by charge group (top level organisation entity grouping users). In this initial phase, members of the CMS experiment have used the majority of the resources.

At the time of writing, between 10 and 20 SWAN users per day create a session that enables the HTCondor integration. This is still a small fraction of the total number of users that use SWAN per day (around 400).

## 3 User Feedback

One of the key objectives of the Pilot was to gain insight in the way users will use an Analysis Facilities and what their perceived needs are. Therefore, we collected feedback from the initial 23 volunteer users through dedicated meetings and a structured survey [19]. This was designed to cover aspects such as: how well the intended usage and the provided functionality match, experience with stability, monitoring, performance, scale of usage, repetitions,

convenience of the service, completeness of software and hardware spectrum, importance of transition between interactive and batch-based usage, characteristics and needs of data access, software provisioning and framework preference. In addition, the opportunity was given to share observations and requirements in text form.

The collected feedback is summarised in the next subsections.

### 3.1 Software Provisioning

When logging into the Pilot, users can load software environments in two ways. The default method consists of accessing pre-configured LCG releases [20] and experiment stacks, which in general is well appreciated by users.

In certain cases, loading custom software environments (e.g., conda) or custom container images is required; the Pilot allows installing additional software packages on the EOS volume associated with the user, to be used on top of LCG releases. This alternative is intended to be temporary, as some errors accessing the volume on the batch side have been reported.

### 3.2 Data Access

The Pilot enables access to CERNBox [21]/EOS via a FUSE mount or direct XRootD access. User credentials are automatically generated, granting access to all CERN-deployed instances, including CERNBox (homes and projects), and dedicated experiment instances. CERNBox provides the home directory for users in SWAN/JupyterLab, allowing seamless file browsing through the UI, embedded terminal, or notebooks and scripts.

Credentials and access methods propagate from SWAN to Batch nodes, enabling users to transition between local and external resources without additional configurations.

A majority of Pilot users (71%) reported accessing official experiment datasets in reduced formats (e.g., NanoAOD, PHYSLITE), while 42% worked with non-reduced or self-created/private datasets. Some users combined official reduced and non-reduced datasets or official reduced and self-created data. All users frequently run analysis on a significant amount of input data.

Among the questionnaire respondents, only 28% did not access external datasets (i.e., data stored outside CERN). Most accessed external data via remote URLs (e.g., XRootD), with one user leveraging Rucio [22] for local copies. No users manually transferred data (e.g., via xrdcp). Access is managed through X.509 proxy certificates, with one user also mentioning access to unauthenticated data.

Two-thirds of users found a UI extension for managing Rucio data directly from Jupyter useful or very useful. A similar preference was expressed for XCache integration, which has since been implemented for CMS data access.

### 3.3 User Interface

The user interface provided by the Pilot is a JupyterLab deployment. It offers the upstream web-based application that allows browsing the user's storage (located in CERN's EOS storage system), provides Jupyter notebooks with different kernels (Python, R, ROOT C++ or Julia) as well as spawning terminal sessions. This is completed by the Dask JupyterLab extension described in Section 2.1.

In general, users are satisfied with the extension's functionalities. Some users have reported that the Dask monitoring dashboard can lag behind and freeze, and the creation and/or destruction of a cluster can take the order of minutes, if many (in the order of few hundreds) workers are requested. The reason for these issues is that jobs are submitted and/or removed sequentially.

### 3.4 Execution of the Analysis

As imposed by the Pilot's requirements, the analysis workflows are mainly run within the ROOT RDataFrame and coffea frameworks, with a small fraction of users adopting custom implementations of the latter or simpler uproot [23] based and awkward [24] based code.

As mentioned in the previous sections, such frameworks enable a seamless integration with Dask, and consequently allowing the scaling of compute resources to the HTCondor pool. Users did not find significant showstoppers in executing or adapting pre-existing analysis code with the Pilot's Dask oriented architecture. For the vast majority of use cases ($\sim$ 85%), 10 to 100 Dask workers are usually allocated during an analysis; however, there are a few cases ($\sim$ 15% of the time) in which up to 1000 workers were requested.

The time necessary to instantiate Dask workers generally does not constitute a bottleneck for users' operations. A significant improvement in satisfaction was achieved after the implementation of the dedicated core buffer for SWAN-originated jobs, given that the reported execution times for analyses range from minutes to 2-4 hours, effectively indicating interactive-like workflows that benefit from quicker scheduling.

### 3.5 Stability

When using the Pilot, most users reported the system to be stable most of the time. One user even claimed it was always stable, and only one user reported finding the system being unstable several times.

A few issues with the system were reported but were promptly addressed by service managers. An initial problem with resource allocation in Batch was identified, which has since been fixed and is now stable. Another issue involved sporadic worker failures when requesting a large number of workers, which was fixed through a configuration change.

Frequent EOS read issues occurred when workers used the FUSE mount. Alternatives have been proposed, such as transferring files to nodes instead of relying on distributed filesystems. Additionally, ROOT, since version 6.28, also provides automatic conversion of FUSE paths to the XRootD access protocol, allowing users to have faster and more reliable access [25].

One user also reported instability when memory usage was very high, requiring additional RAM allocation. However, these extra resources are not used throughout all analysis steps and are tied to configurations that also increase the number of CPUs available, reducing the analysis efficiency.

### 3.6 Other Feedback and Suggestions

Of great interest for the future development of the Pilot was understanding the typical way of usage and the user's motivation. It was reported, that the code is tested and developed locally, based on a subset of files. This is done with the same code that is used on the batch system. After the analysis code is stable and performing sufficiently well, Dask can be used to process $10^9$ NanoAOD events within a few hours. During this time, the client of the interactive session closed, and after completion and reconnection, Jupyter notebooks are used for plotting and final analysis.

However, given that current users are most likely not representing the needs of all potential users, more input is needed from a larger body of users. Leveraging the results of the WLCG survey on Analysis Facilities, drafted at the WLCG/HSF Workshop 2024 [26] and later submitted by the LHCC, will be very helpful. In addition, a process has started to exchange the gained experience with other Analysis Facilities.

## 4 Summary and Next Steps

The feedback received by the initial users and the exchange with other operators of Analysis Facilities has been very useful. In order to make further progress, the most important next step is to increase the number of users. The team is now actively promoting the use of the offered service.

Nevertheless, several aspects are already clear. The ability to scale quickly to a few hundred Dask workers and a reliable, easy way to augment standard software stacks are central to the acceptance of an Analysis Facility, as is the support of access to local and remote data. For the support of the latter, use and configuration of XCache based systems will be explored. This is closely related to the seamless integration with the authentication and authorisation system required by WLCG resources.

From discussions with other providers of Analysis Facilities, it has become clear that there is great similarity in the functionality offered, but significant difference in the way these Analysis Facilities work from the user perspective. To offer users a similar level of uniformity as with WLCG resources, work is needed within the community to agree on a common interface and set of functionality for Analysis Facilities.

## References

[1] ROOT's RNTuple I/O Subsystem: The Path to Production Jakob Blomer, Philippe Canal, Florine de Geus, Jonas Hahnfeld, Axel Naumann, Javier Lopez-Gomez, Giovanna Lazzari Miotto, Vincenzo Eduardo Padulano EPJ Web of Conf. 295 06020 (2024) DOI: 10.1051/epjconf/202429506020

[2] Rene Brun and Fons Rademakers, ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86.

[3] Coffea analysis framework, https://zenodo.org/records/14781208, Accessed: 2025-04-10

[4] JupyterLab notebook interface, https://jupyter.org, Accessed: 2025-04-10

[5] The Keras Framework, https://keras.io, Accessed: 2025-04-10

[6] The PyTorch Framework, https://pytorch.org, Accessed: 2025-04-10

[7] CERN LXPLUS service, https://lxplusdoc.web.cern.ch, Accessed: 2025-04-10

[8] CERN Batch service, https://batchdocs.web.cern.ch, Accessed: 2025-04-10

[9] EOS Storage, https://eos-docs.web.cern.ch, Accessed: 2025-04-10

[10] CernVM File System, https://cvmfs.readthedocs.io, Accessed: 2025-04-10

[11] Service for Web-based Analysis, https://swan.docs.cern.ch, Accessed: 2025-04-10

[12] REANA, https://reanahub.io, Accessed: 2025-04-10

[13] Kubeflow, https://ml.docs.cern.ch, Accessed: 2025-04-10

[14] Dask, https://www.dask.org, Accessed: 2025-04-10

[15] HTCondor Software Suite, https://htcondor.org, Accessed: 2025-04-10

[16] JupyterLab Dask Extension, https://github.com/dask/dask-labextension, Accessed: 2025-04-10

[17] XRootD project page, https://xrootd.org, Accessed: 2025-04-10

[18] HEPScore: A new CPU benchmark for the WLCG, Domenico Giordano, Jean-Michel Barbet, Tommaso Boccali, Gonzalo Menéndez Borge, Christopher Hollowell et al. EPJ Web of Conferences 295 07024 (2024).

[19] CERN Analysis Facility Pilot User Survey, https://docs.google.com/forms/d/e/1FAIpQLSd7fBFZmvaJxOgnd_ARNdRGk-DQJmcGckpbgAAIG93tqG1gOA/formResponse, Accessed: 2025-04-10

[20] LCG releases, https://ep-dep-sft.web.cern.ch/document/lcg-releases, Accessed: 2025-04-10

[21] CERNBox service, https://cernbox.cern.ch, Accessed: 2025-04-10

[22] M. Barisits, T. Beermann, F. Berghaus, B. Bockelman, J. Bogado, D. Cameron, D. Christidis, D. Ciangottini, G. Dimitrov and M. Elsing, *et al.* "Rucio - Scientific data management," Comput. Softw. Big Sci. **3** (2019) no.1, 11 doi:10.1007/s41781-019-0026-3 [arXiv:1902.09857 [cs.DC]].

[23] Uproot Python library, https://uproot.readthedocs.io, Accessed: 2025-04-10

[24] Awkward Python library, https://awkward-array.org, Accessed: 2025-04-10

[25] ROOT 6.28 release notes, https://root.cern/doc/v628/release-notes.html#faster-reading-from-eos, Accessed: 2025-04-10

[26] WLCG/HSF Workshop 2024, LHCC questions draft document discussion, https://indico.cern.ch/event/1369601/contributions/5923993/, Accessed: 2025-04-10