Lucio Anderlini
Matteo Barbetti
Giulio Bianchini
Diego Ciangottini
Stefano Dal Pra
Diego Michelotto
Rosa Petrini
Daniele Spiga

DEVELOPING ARTIFICIAL INTELLIGENCE IN THE CLOUD: THE AI_INFN PLATFORM

Abstract *The INFN CSN5-funded project* AI_INFN *("Artificial Intelligence at INFN") aims to promote ML and AI adoption within INFN by providing comprehensive support, including state-of-the-art hardware and cloud-native solutions within INFN Cloud. This facilitates efficient sharing of hardware accelerators without hindering the institute's diverse research activities.* AI_INFN *advances from a Virtual-Machine-based model to a flexible* Kubernetes*-based platform, offering features such as JWT-based authentication, JupyterHub multitenant interface, distributed file system, customizable conda environments, and specialized monitoring and accounting systems. It also enables virtual nodes in the cluster, offloading computing payloads to remote resources through the Virtual Kubelet technology, with* InterLink *as provider. This setup can manage workflows across various providers and hardware types, which is crucial for scientific use cases that require dedicated infrastructures for different parts of the workload. Results of initial tests to validate its production applicability, emerging case studies and integration scenarios are presented.*

Keywords artificial intelligence, analysis facilities, high energy physics

Citation Computer Science 26(SI) 2025: 9–28

1. Introduction

The National Institute for Nuclear Physics (INFN) is the Italian research agency dedicated to studying the fundamental particle constituents of the universe and the laws governing them, under the supervision of the Ministry of Universities and Research (MUR). INFN conducts theoretical and experimental research in subnuclear, nuclear, and astroparticle physics, with applied physics emerging as a significant research direction, including medical physics and material sciences. In particular, high-scale computing has become essential, leading to the deployment of a large distributed computing infrastructure with 10 major sites nationwide. This infrastructure is a key part of the national computing backbone (the *Datalake*) within the NextGenerationEU project "National Center for Big Data, HPC and Quantum Computing."

To fulfill its mission, INFN adopts cutting-edge technologies and instruments developed in its laboratories and in collaboration with industries. In that context, Artificial Intelligence (and Machine Learning in particular) is a key area and at the same time a research direction. Its implications in INFN activities involve three areas: technological research towards the management of a large-scale distributed and heterogeneous computing infrastructure capable of provisioning hardware accelerators, the development of the foundational aspects of algorithms intended for scientific research, and the application of Artificial Intelligence to the INFN research domains with attention to induced biases and ethics.

The AI_INFN initiative, funded by INFN, aims to connect the scientific communities developing infrastructures, algorithms, and applications. It provides tools, computing infrastructures, and social connections to foster collaboration and facilitate the adoption of AI-powered computing technologies within INFN research fields.

AI_INFN is organized into four work packages: procuring and operating computing infrastructure with hardware accelerators, organizing training events for Machine Learning adoption, building a community of ML experts and developers across INFN units, and developing the competence to profit from hardware accelerators beyond GPUs, such as FPGAs and Quantum Processors, for AI model training and inference.

The AI_INFN Platform is a cloud-native toolset developed in collaboration with the *DataCloud* initiative, operating INFN Cloud, to support the activities of the four work packages of AI_INFN. In this document we present the AI_INFN platform and discuss its relevance to the outlined scopes.

2. INFN Cloud Infrastructure for Artificial Intelligence

The software stack enabling Artificial Intelligence is designed to ease an efficient and effective acceleration of tensor algebra with Graphics Processing Units (GPUs), which arose central in most developments and applications of Artificial Intelligence.

A large number of computing problems relevant to scientific research, from numerical computing on lattice to data analysis, can be formulated in terms of tensor algebra to increase the computation efficiency and port it to GPUs or other hardware

accelerators specialized for Artificial Intelligence. Often, such a reformulation is also a mandatory prerequisite to develop and benefit from machine learning algorithms.

Both the reformulation of the scientific computation problems and their integration with machine learning toolkits require combining expensive hardware and the highest degree of domain knowledge. Following this reasoning, the AI_INFN infrastructure was designed to provide researchers with a reliable access to a pool of shared hardware accelerators, for the time needed for the development and the validation of the techniques by the domain experts, trying to eliminate the administrative, bureaucratic, economic and technical burdens otherwise associated to the setup of on-premise HPC infrastructures.

The AI_INFN platform is a *Software-as-a-Service* provided by INFN Cloud. The underlying specialized and dedicated hardware was designed for High Performance Computing tasks and is hosted and managed at INFN CNAF in Bologna, few steps away from the pre-exascale supercomputer CINECA Leonardo. The infrastructure comprises four servers, acquired and installed between 2020 and 2024 as the demand for Cloud-accessible HPC computing resources increased:

- Server 1 (2020), with 64 CPU cores, 750 GB of memory, 12 TB of nVME disk, eight nVidia Tesla T4 GPUs and five nVidia RTX 5000 GPUs;
- Server 2 (2021), with 128 CPU cores, 1024 GB of memory, 12 TB of nVME disk, two nVidia Ampere A100 GPU, one nVidia Ampere A30 GPU, two AMD-Xilinx U50 boards and an AMD-Xilinx U250 board;
- Server 3 (2023), with 128 CPU cores, 1024 GB of memory, 24 TB of nVME disk, three nVidia Ampere A100 GPUs and five AMD-Xilinx U250 boards;
- Server 4 (2024), with 128 CPU cores, 1024 GB of memory, 12 TB of nVME disk, one nVidia RTX 5000 GPUs and two AMD-Xilinx Versal V70 boards.

The four servers are clustered in a tenancy of the OpenStack infrastructure of CNAF, which is federated in INFN Cloud.

Before AI_INFN started its activity in January 2024, the farm was maintained by another INFN initiative named ML_INFN that developed a provisioning model relying on Virtual Machines assigned to groups of users developing a data analysis or machine learning study [26]. In the ML_INFN initiative, automations were created to install nVidia drivers for CUDA [43], the JupyterHub [39] software stack, and monitoring tools like Prometheus and Grafana [36]. A TOSCA [27] template was developed to integrate with the INFN Cloud orchestrator [45], allowing hardware resource configuration, including GPU model selection. An Ansible [27] playbook automates the installation and configuration of software packages to interface JupyterHub with JWT-based authentication and authorization from the DataCloud initiative. INFN Cloud uses Indigo IAM [30] for identity and authorization management. Deploying a VM with JupyterHub requires an administrative role in INFN Cloud IAM, making users administrators of the VM and its web service. JupyterHub users are identified through INFN Cloud IAM as well, but do not need administrative roles, as they manage their own containers, only. The JupyterLab instance holds an identification token

of the user, who can use it to interact with other *DataCloud* services (such as a PSI ELOG [6] instance, or the object storage service) without any further authentication.

During the late period of AI_INFN, however, an increase in the user base highlighted some limitation to the efficiency of this provisioning model. For example, since VMs and GPUs were statically assigned to specific projects, with fixed resources for each. Reassigning GPUs to other projects requires destroying the associated VM, resulting in the loss of the file system and potentially user data. Similarly, increasing the resources for a project requires re-creating the VM, which hindered the ability to run collaborative programming sessions. Secondly, batch jobs using GPUs required dedicated VMs that competed with interactive resource access. Ideally, these jobs could take advantage of time slots when interactive development is less frequent, such as nights and weekends, but separate VMs with different administrators can hardly be coordinated to process batch jobs without interfering with interactive development. Finally, the need for many users to become administrators in order to be autonomous in the configuration of the their virtualized environment resulted in a significant increase in critical administrative burden across many similar, but not identical deployments managed by a number of different researchers, often with a data science background and limited system administration experience. The security risks grew to an unacceptable level and called for the introduction of an alternative model enabling users to tune the resources provisioned to their cloud-based computing environment, without becoming administrators of a multi-user web service.

2.1. Software-as-a-Service provisioning model, the platform

In order to overcome the limitations of a provisioning models based on VMs assigned to projects, a second provisioning model is being developed as a managed Software-as-a-Service (SaaS) application. In order to improve the flexibility in the allocation of the resources to different users and projects, user data must be decoupled from the hardware allocation. In other words, deallocating compute resource should not imply removing user data. Decoupling compute and storage resources is also effective to develop, validate and test the same application on different hardware, since the number of cores, the memory and the hardware accelerators allocated to a task can be modified without risks for user data integrity.

Several container schedulers and orchestrators, such as *Apache Airflow* [2], *Docker Swarm* [5] or *HashiCorp Nomad* [16], would be suitable for deploying JupyterHub on Cloud resources, with Kubernetes emerging as de-facto standard for deploying applications in INFN Cloud.

The AI_INFN platform was deployed on a Kubernetes cluster spanning on at least three VMs within the dedicated OpenStack tenancy providing part of the storage resource, the monitoring infrastructure and the Kubernetes control plane. A minimal amount of compute resources is also provisioned to make it possible for users to access their data on the platform anytime, but the compute to perform data analysis, machine learning and artificial intelligence studies are added to the cluster by deploying

additional VM, possibly providing access to the hardware accelerators and let them join the cluster.

The same authentication and authorization workflow described above for the VM-based provisioning model is used. Once authenticated, users can configure and spawn their JupyterLab instance using JupyterHub. Similar to other scientific platforms discussed in the literature [25, 47, 48], the authenticated JupyterLab session serves as a secure gateway to Cloud resources using token-based identity delegation, without introducing novelty compared to the VM-based model. In order to ensure consistency between the compute resources allocated to the platform and the configuration options presented to the landing users, the JupyterHub resource-configuration page was customized to query the Kubernetes API server at each page refresh.

To enable transparently moving the user's session from one machine to another, for example to change GPU model, user's homes are stored in a distributed NFS file system. In addition, nVME volumes are made available to the containers as ephemeral, high-performance file system to be used for training of machine learning models and for data analysis. The file systems made available the platform users is discussed in Section 2.1.3.

The VMs providing compute resources to the cluster can be removed and used as stand-alone virtual machines running the Ansible playbook to install the software stack, or assigned to a different cluster in the same tenancy.

The resulting software stack allows for a clear separation of concerns and identify different administrative and user roles. The configuration of the hardware and its integration in the hypervisor (in particular, OpenStack) requires a *site administrator*, with access to the configurations of the computing site and its infrastructure. The *tenancy administrators* or, in our particular case the AI_INFN *administrators*, are in charge of deploying the VMs. The same level of authorization is needed to create project-dedicated stand-alone VMs and VMs that will be part of a Kubernetes cluster. The deployment of the VMs is usually achieved through the INFN Cloud orchestrator and its dashboard, which is an INFN service maintained by INFN Cloud *administrators*. The configuration of the Kubernetes cluster where the platform is deployed requires a further role, named *platform administrator*. Finally, users are granted with administrative privileges within the perimeter of their container, which can be customized as discussed in Section 2.1.4.

The default OCI (Open Container Initiative) image includes the configurations to mount the user's bucket in the INFN object storage as a virtual file system, using a JWT-compliant (JSON Web Token) version of the rclone-mountpackage [33]. In addition to the identity management service, the platform interfaces with several INFN Cloud services, including Ceph and RadosGW for object storage, the OCI registry for publishing custom JupyterLab images, and the orchestrator. Integration with remote compute backends for heavy computing tasks is under development and is discussed in Section 2.1.7.

The software stack of the AI_INFN platform and the roles enabled to configure its components are represented in Figure 1.

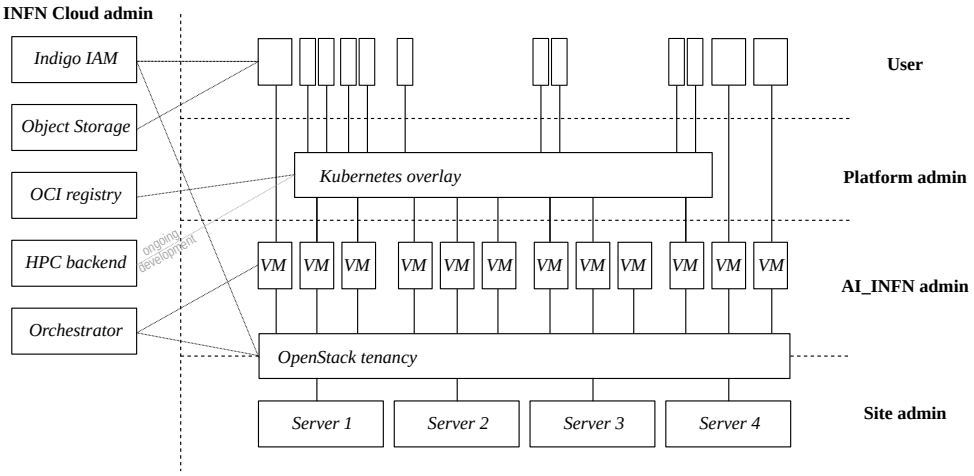

Figure 1. Structure and design of the AI_INFN platform. The role to configure the various components of the software stack are also indicated

2.1.1. GPU management

While in the VM-based provisioning model the GPU drivers are defined in custom Ansible playbook and TOSCA template, in the AI_INFN platform the Kubernetes GPU operator [18] developed and maintained by nVidia is used. The nVidia GPU Operator on Kubernetes streamlines the management and deployment of nVidia GPU resources by automating the installation and configuration of required components within a Kubernetes cluster. It uses the Kubernetes Operators pattern to oversee the life-cycle of GPU drivers, Kubernetes device plugins, the nVidia Container Runtime, and monitoring tools such as the nVidia DCGM Exporter, as shown in Figure 2. In general, the GPU Operator is designed to simplify the management of clusters with a large number of GPU accelerators, enabling efficient scaling and resource optimization for GPU-accelerated workloads such as AI, machine learning, and data processing. In the AI_INFN platform, the GPU operator is preferred over the management based on Ansible and TOSCA for its closer integration with Kubernetes, which enables modifying the configuration of the GPU drivers by editing the metadata of the node, and for a standardization of the naming conventions for the GPU models enabling better portability of the platform itself.

The nVidia GPU Operator also provides an effective way to manage GPU partitioning. Indeed, the nVidia A100 GPUs can be partitioned into multiple multi-instance GPUs (MIG) [15]. Each partition operates independently, ensuring consistent performance and optimised resource allocation. In the AI_INFN platform, the

GPU partitions are allocated to different users, who run their workloads independently in an environments isolated at hardware level. Note that the number of available PCIe slots on the motherboards of the servers limits scaling the number of GPUs and hence the number of concurrent users connected to the platform. Using MIG, the number of users per PCIe slot can increase up to a factor 7. In addition, the partitioning can be easily reconfigured by editing the Kubernetes node metadata to flexibly shape the fleet of hardware accelerators selectable by the users based on the demand.

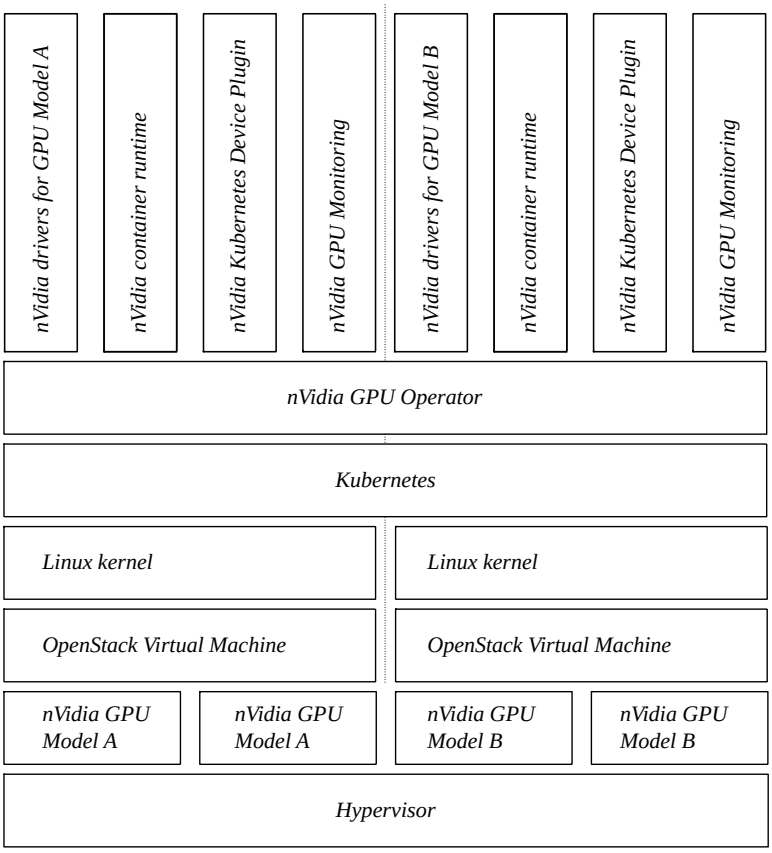

Figure 2. Outline of the structure and composition of the nVidia GPU Operator

2.1.2. Project-based resource allocation and feature gating

From the user's perspective, an important advantage of the VM-based provisioning model over the Kubernetes platform is that all the users of the VM are member of the same project. The project data can be shared with all the users of the VM, in some shared directory of the VM file system without any additional care for who is entitled to access what. To provide users with similar functionalities, the AI_INFN platform

relies on Jupyter groups, which can be managed directly through the JupyterHub Administration panels since version 4. To avoid confusion the IAM groups, which are used to determine who, among the INFN Cloud users, has access to the AI_INFN platform and whom has administrative privileges, we will refer to the JupyterHub groups used to define users contributing to the same activity as *poleoject groups* or simply *projects*. A separate volume per project is automatically generated by JupyterHub and mounted within the JupyterLab instances of its members. Project groups can be associated to extraordinary resource allowances for their members. They will become entitled to select a number of cores, an amount of memory and GPU models otherwise unavailable to the unprivileged users.

In addition, entire Kubernetes nodes can be *tainted* with a reservation keyword that prevents users without the requested membership to spawn their notebooks there. This mechanism has been introduced to grant access to full nVidia A100 accelerators to specific projects requiring large GPU memory for development purposes for a limited amount of time. In the future, it could be used to grant reserved access to some resources to projects financially contributing to the hardware infrastructure.

Finally, Jupyter groups are also used to define beta-testers for new features while they are added to the platform. These *feature gates* are used to gradually widen the user basis before a new feature is publicly released. Examples of features introduced following this procedure include the experimental adoption of a cloud-native virtual file system discussed in Section 2.1.3 and the submission of jobs to platform queues managed with Kueue as presented in Section 2.1.6.

2.1.3. File systems

The main platform file system is distributed through the containers through NFS. One of the platform nodes, runs an NFS server in a Kubernetes pod and exports data to the containers spawned by JupyterHub. At spawn time, JupyterHub is configured to create the user's home directories and the project shared volume. JupyterHub users are not mapped to file system users and access control to files relies on mount policies. The resulting weak isolation between users might be strengthened with Kerberos authorization in the future. A special directory of the platform file system is reserved for distributing managed environments that users can use directly or clone and extend in their home or project directories. That directory is mounted in read-only mode for all users except for the environment maintainers who are entitled to create and maintain template environments.

While significantly less performing than a local file system, NFS performs sufficiently well on large amounts of small files, which is a relatively common scenario when configuring software environments using virtual environments such as *conda* [4] or *Python venv* [20].

The platform file system is subject to regular encrypted backup. Backup data is stored in a remote Ceph [49] volume provisioned by INFN Cloud using the *BorgBackup* package [3] to ensure data deduplication.

Large datasets must be stored in a centralized object storage service based on Rados Gateway [21] and centrally managed by *DataCloud*. To ease accessing the datasets with the Python frameworks commonly adopted for machine learning developments, a patch of the rclone package was developed [33] to enable mounting the user's bucket in the JupyterLab instance using the same authentication token used to access JupyterHub. The mount operation is automated at spawn time.

To overcome the bandwidth limitations of a virtual file system with a remote backend, heavily impacting on most iterative training and data analysis procedures requiring to process the whole dataset multiple times, the AI_INFN platform provides an ephemeral file system, mapped directly to a logical volume defined in the nVME storage of the underlying hypervisors. The data needed to test and validate the analysis and machine learning techniques developed on the platform are usually copied in this volume by the users at the beginning of each new session. Ephemeral fast volumes are also appreciated as a cache to store intermediate results or to extend the available RAM memory through memory mapping.

In principle, virtual file systems can be mounted on various computing resources, enabling the sharing of notebooks and user-defined computing environments across multiple compute backends. Specialized file system for serverless computing are being considered for this specific task,with JuiceFS [10] emerging as the most suitable intermediate solution between a platform file system and a virtual file system mounted with rclone. JuiceFS is a cloud-based, high-performance, POSIX-compliant distributed file system specifically designed for multi-cloud and serverless computing. It decouples data and metadata delegating these tasks to highly optimized third party projects, combining a metadata engine implemented with either key-value databases (such as Redis [34]) or relational database management systems (such as PostgreSQL) with storage systems accessed through S3, WebDAV or other high-throughput protocols. In the ongoing evaluation, Redis and MinIO [14] are used.

Finally, as is standard in most OCI runtimes, users can install or upgrade packages in their containers. Installing new software will introduce ephemeral modifications in OverlayFS layer on top of the container file system. This feature is particularly used when developing an OCI image intended for the platform as it enables checking possible side effect in the installation of new packages before going through the build and upload procedures.

A more effective and popular alternative to installing packages in the container, is to rely on the binaries distributed through the CERN VM file system (`cvmfs`). CVMFS, used by the LHC experiments and other communities to distribute software through the nodes of the WLCG, is made available to the platform users through a Kubernetes installation that shares the caches among different users and sessions.

A comparative summary of the file systems made available to the platform users is given in Table 1. A possible configuration of the default container spawned in the platform, relying on some of those file systems, is depicted in Figure 3.

Table 1
File systems available in the AI_INFN platform, their intended usage and the most important limitations

File system	Intended usage	Main limitations
Platform (NFS)	User's home directories, project's code and notebooks, small datasets, software environments...	Cannot be exported, does not scale beyond few terabytes
Cloud storage (`rclone`)	Medium-sized and large datasets, Apptainer images...	Unsuitable for small files
Overlay (`containerd`)	Temporary modifications to the container file system	Ephemeral, (small) metadata overhead
Temporary	Caching large datasets and intermediate results, memory mapping	Ephemeral
Multi-site (`JuiceFS`)	Sharing user's code, environments and small to medium-sized dataset towards remote compute backends	Slower than NFS, less robust than cloud storge
CVMFS	Accessing software libraries and executables specific for the simulation and data processing of the CERN experiments	Read-only

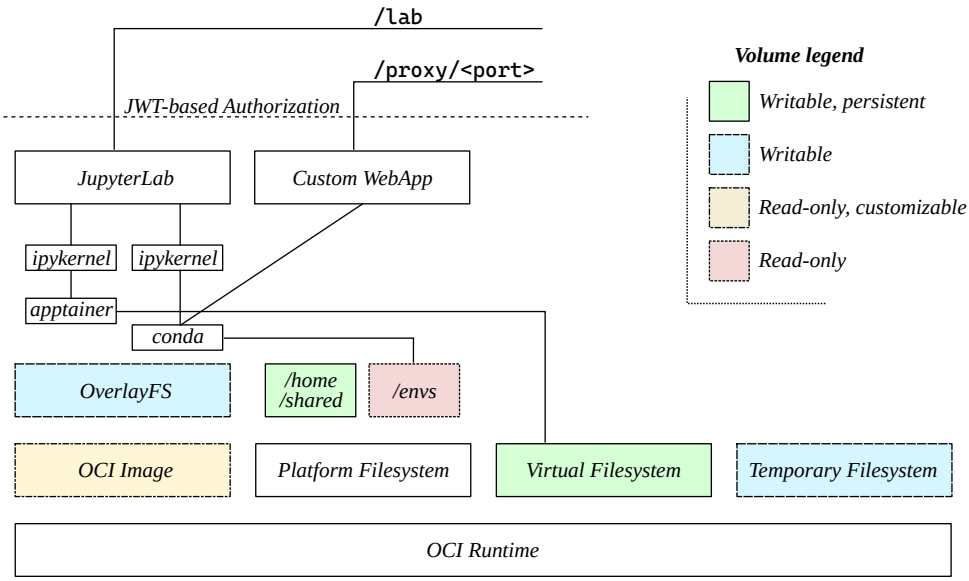

Figure 3. A possible configuration of the default JupyterLab container spawned in the AI_INFN platform

2.1.4. Templated software environments

One of the most frequent requests of support with the VM-based provisioning model concerned the configuration of a GPU-accelerated Python software stack. While the TOSCA template and the Ansbile playbook were sufficient to install the nVidia driver and runtime, the choice and the installation of the Python libraries providing the software dependencies for the developed application was left to the users.

In general, managing the dependencies of a data science project is a responsibility of the developers and analysts. For some projects, relying on the most updated libraries is crucial to get new features, but for others, the ability of running a 10-years old software stack is a strict requirement. In addition, many analysis projects need more than one computing environment. For example, one may need an environment to prepare the data and a different one to train and validate a machine learning model. When introducing the AI_INFN platform, special care was devoted to make user sessions as modifiable and adaptable as possible, by defining mechanisms for users to create and maintain their own computing environments. The most radical customization option is to build and pick a custom OCI image. Communities and single users can extend the default OCI image by adding system libraries or software packages modifying the JupyterLab service itself. A common motivation to define custom OCI images is to include web-based dashboards or single-user web applications to be served through Jupyter Server Proxy [11].

While users typically prefer conda for defining custom software environments for analysis and machine learning applications, Apptainer images are gaining traction. Conda environments consist of thousands of small files, whereas Apptainer uses SquashFS [22], a compressed read-only file system, to encapsulate the entire environment in a single file. This allows Apptainer images to be more effectively share and distributed through object stores compared to conda environments. AI_INFN users are provided with documentation to export conda environments as Apptainer images and use them as Jupyter kernels for interactive notebook processing. Although user management of these environments is essential, the process of defining computing environments capable of utilizing hardware accelerators from scratch can be laborious. Therefore, the availability of pre-configured environments, which can be extended with project-specific dependencies, is highly desirable. The common file system available in the AI_INFN platform offers the opportunity of providing the users with conda environments and Apptainer images with the versions of the software packages to accelerate the most common Machine Learning frameworks matching the GPU driver and runtime installed in the underlying VMs. Users are then provided with instructions to clone those environments and add the project specific dependencies, which are usually related to data loading and visualization, and independent of the GPU software stack.

A notable exception is represented by the software environment to develop Quantum Machine Learning (QML), featuring Python modules that simulate the effect of

quantum operators on GPU and therefore requiring the same attention as other GPU-accelerated ML libraries to match the versions of the underlying software.

At the time of writing, the conda software environments managed centrally include:

- TensorFlow [24] 2.14, with Keras [32] 2;
- TensorFlow 2.16, with Keras 3;
- Torch [44] 2.3;
- TensorFlow 2.16, with JAX and PennyLane [28];
- Torch 2.3, with PennyLane.

Finally, Apptainer images specialized for the data processing of the LHC experiments can be obtained via CVMFS.

2.1.5. Monitoring and accounting

As mentioned above, a dedicated monitoring and accounting system has been set up for the platform in order to effectively control the use of all the platform's resources and in particular of the GPUs.

Several metric exporters have been configured to collect the information of interest and then expose it to a Prometheus [50] instance running in the platform. Some of these exporters are already available as Free and Open Source Software, such as Kube eagle [46], which manages information about the use of the cluster's CPUs and memory resources by the various components of Kubernetes, or nVidia GPU DCGM exporter [17]. Other exporters were developed on purpose, for example to monitor the usage of storage resources. All the metrics collected by Prometheus are then made visible and accessible through a Grafana dashboard. Grafana is run in a VM independent of the platform cluster and is used to monitor other VMs in the AI_INFN OpenStack tenancy. It also hosts a PostgreSQL [19] database for the accounting metrics, updated at regular intervals by averaging the metrics obtained from the monitoring Prometheus service.

2.1.6. Job queueing for opportunistic access to the resources

Once an analysis or model development is mature, analysts may want to scale it to more resources for example to have longer training time than is available in the interactive session, or to free up interactive resources for development, or to run multiple trials in parallel.

For this reason, a batch system was designed that would allow non-interactive workloads to be executed opportunistically, making effective use of the cluster's resources without adversely affecting the performance of interactive sessions. A local, cluster-internal batch system was therefore set up to run the jobs as a secondary priority to the JupyterLab sessions, taking advantage of the platform's least busy times, such as nights and weekends, to run the jobs. In addition, the system was configured so that if there is a high demand for interactive sessions and the resources are occu-

pied by running jobs, these are evicted and the resources are immediately freed to allow new JupyterLab sessions to be spawned.

Kueue [12] is used to effectively manage batch workloads. Kueue is a set of APIs and a controller designed to simplify and improve job queue management in Kubernetes, providing a solid and scalable infrastructure for job queue management. Kueue natively integrates with Kubernetes resources and features, leveraging the cluster's orchestration and management capabilities. Thanks to its dedicated controller, Kueue simplifies job state monitoring and allows resources to be automatically scaled based on workload.

To enable users to make the most of the platform's batch system, a microservice, `vk-dispatcher`, was developed to run as a sidecar container in the same Kubernetes Pod the user's JupyerLab session and translate the latter into a Kubernetes job, by matching the JupyterHub user system with the Kueue job queues. At spawn time, `vk-dispatcher` is informed by JupyterHub with the project groups the user belong to. At job submission time, `vk-dispatcher` determines the subset of job queues available to the users based on her or his groups. In order to decouple the job submission syntax from the current GPU availability in the cluster, `vk-dispatcher` jobs define the minimum amount of GPU memory which is necessary to the job to succeed. At submission time, `vk-dispatcher` converts the memory requirement into a set of priorities for the available GPU models, excluding those with insufficient GPU and assigning higher priority to smaller GPUs and MIG partitions. A proof-of-concept controller to redefine the MIG partitioning of some nodes based on the pressure on the job queues has been tested, but because of the limited number of MIG-capable devices, combined with the need to drain the corresponding node from running jobs and interactive user sessions, a static configuration of the MIG partitions was preferred.

On the user side, job management is facilitated by a REST API, which serves as the primary interface for job submission, orchestration and management. This API abstracts the complexity of Kubernetes, providing a simplified and consistent way to interact with the system. To further streamline interactions, a dedicated command-line tool is provided, allowing users to manage jobs using intuitive commands without requiring in-depth knowledge of Kubernetes. The Command Line Interface (CLI) simplifies operations such as job submission, status monitoring, log retrieval and job cancellation by encapsulating the complexity of API calls. This design ensures that the system is accessible to users with varying levels of expertise, while maintaining the scalability and robustness of a Kubernetes-based infrastructure.

2.1.7. Ongoing development: offloading workloads to remote backends

The resources of the platform cluster on which the batch jobs can be executed are limited and subject to instant eviction due to the higher priority of the Jupyter interactive sessions. For this reason, an offloading system is being implemented to enable batch jobs to be transparently executed on computing resources outside the

cluster from which they are launched. In addition, this will contribute to demonstrate a provisioning mechanism to make batch resource available to Kubernetes workload managers.

The offloading mechanism relies on the concept of *Virtual Kubelet* [23]. A service within the platform is configured to interact with the Kubernetes API server mimicking a physical node (kubelet). The pod are then addressed to the underlying node to be delegated towards a remote computing backend, defined by the *Virtual Kubelet Provider*. The InterLink [8] provider is being developed by the EU-funded *InterTwin* [9] initiative to enable the creation of an ecosystem of remote plugins (e.g. SLURM [51] or HTCondor [29]) through the introduction of an common interface. Hence, *de facto*, InterLink provides an abstraction layer for running Kubernetes pods on any remote resource that can manage the lifecycle of a container.

An InterLink virtual node has been added to the platform to submit Kueue jobs to remote backends. For testing purpose, small Kubernetes clusters in other sites federated with INFN Cloud were used as compute backends. Successful offloading of the Kubernetes jobs managed with Kueue was also obtained using a simple Virtual Machine equipped with a GPU and using docker runtime as compute backend.

JuiceFS was used to create a common POSIX-compatible file system through the platform and the remote compute backends. Access control over the JuiceFS file system is explicitly disabled to enable symmetric access for different user IDs associated to different remote data centers. At submission time, jobs without dependencies for files in the platform file system can be marked with a Kubernetes toleration indicating they are suitable for being processed in the remote backend. An appropriate Kueue resource flavor is defined to prioritize the submission of the jobs to the remote, batch-dedicated compute backend first, and use the local, opportunistic resources then. This setup, while not mature enough for massive productions, enabled successful scalability tests up to a hundred of concurrent jobs.

As mentioned above, however, the ultimate goal of the InterLink provider in the context of AI_INFN is to enable running generic Kubernetes-defined jobs in SLURM and HTCondor compute backends. Indeed, with the increasing availability and financial support for High-Performance Computers (HPCs), the ability of executing Kubernetes-defined jobs in managed batch-system arises as a key enabling technology to grant interoperability between sites and computing platforms. In particular, CINECA Leonardo [13] is a next-generation supercomputer developed in Italy as part of the European EuroHPC project, which aims at providing Europe with a network of high-performance supercomputers. Preliminary tests to submit simple Kubernetes jobs from CNAF to CINECA Leonardo through InterLink were successful. Full integration between the AI_INFN platform and CINECA Leonardo represents the focus of the current development activities.

In addition, CINECA Leonardo is equipped with nVidia A100 GPUs, managed by CINECA itself. In this setup, InterLink plays a crucial role in facilitating communication between the platform and the HPC system. It is responsible for advertising

the availability of GPU resources to the cluster and ensuring appropriate mapping between the Kubernetes extended resources and the GPUs configured by the remote center, enabling effective utilization in the offloading scenario when administrative privileges are not granted.

2.1.8. Future development: developing ML on FPGA, in the Cloud

Specialized accelerators, possibly deployed on Field-Programmable Gate Array (FPGA) devices, can provide hardware acceleration for Artificial Intelligence workloads at a significantly reduced latency and energy consumption.

In the High Energy Physics community, the development of machine learning algorithms for FPGA devices is of particular interest for real-time applications requiring fixed latency. Initiatives sparked within the HEP community such as hls4ml [35] and BondMachine [42], and commercial tools such as Intel OpenVINO [7] or AMD-Xilinx VitisAI [1] aim at reducing the gap between the communities of machine learning experts and hardware engineers. To reduce the gap further, it is important to provide the machine-learning community with a shared access to FPGAs dedicated to firmware and software development. Once the development reaches a sufficiently advanced stage, it should be made easy to export and deploy the machine learning models to production facilities. Under this perspective, the role that the AI_INFN platform plays in provisioning GPU resources for development activities is also desirable for provisioning FPGAs.

In addition, effectively operating hardware accelerators in distributed computing infrastructures is a strategic research area in itself and is particularly relevant in the DataLake paradigm adopted by INFN, enabling the transparent execution of hardware accelerated scientific tasks in multiple sites of the national computing backbone.

As of today, FPGAs are available through VM-based provisioning model similar to what was developed by ML_INFN for GPUs. Integration of FPGAs in the AI_INFN Kubernetes-powered platform or in a similar service will be investigated in a later part of the project.

3. Resource provisioning at the fifth ML_INFN hackathon

Schools and training events focusing on the development of GPU-accelerated software often need to set limits to the number of participants due to the number of interactive sessions simultaneously available.

Between 2021 and 2023, the ML_INFN initiative organized five training events called *hackathons* in two formats: an entry level course providing the participants with the basics of machine learning and exercising on CPU-only resources; and an advanced level focusing on more computing-intensive applications requiring GPU acceleration [26]. During the last ML_INFN advanced hackathon, organized in November 2023 in Pisa, the first proof-of-principle version of the AI_INFN platform was deployed to provision up to 42 MIG partitions to the participants. To ensure some redundancy two equal and independent versions of the platforms were deployed, one in

the AI_INFN OpenStack tenancy at CNAF and one in the HPC farm of the ReCaS-Bari computing center. The two platforms shared the same authentication workflow based on an Indigo IAM instance centrally managed by INFN Cloud, and implement unidirectional synchronization of the NFS file system to propagate the datasets and the school material, while user data was managed separately on the two platforms.

The experiment was successful, with both platforms succeeding in provisioning the necessary resources for the development of the proposed exercises. At the end of the school, the user data at ReCaS was transferred to the CNAF-hosted platform, the ReCaS-hosted platform was destroyed and the GPUs of the CNAF-hosted platform were reclaimed and reassigned to research use-cases through the VM-based provisioning model. The school participants have been able to access their data for two weeks after the end of the event, before the platform was completely deallocated and the platform file system destroyed. The platform was deployed for research purpose on CNAF resources in February 2024.

4. Conclusion

Machine Learning and Artificial Intelligence have been reshaping the landscape of data processing and data analysis applications, making it easier for data analysts and data scientist to accelerate a variety of computing-intensive tasks on GPUs. The AI_INFN initiative is developing and serving a highly customizable development platform, integrated in the service portfolio of INFN Cloud and provisioning different GPU models, possibly partitioned with the Multi-Instance GPU technique. It features a shared file system and templated software environments managed with conda and Apptainer and is designed to strengthen the collaborations among users working on the same project aiming at reducing the time to become productive. The integration with the national INFN identity management enables immediate interaction between physics-domain experts and experienced machine learning practitioners, contributing to foster the adoption of modern techniques in a variety of INFN-related fields, ranging from high-energy physics data analysis [41] to solid-state detector modeling [38] and from theoretical physics [31] to nuclear medicine [40].

Since it was deployed in February 2024, 59 users logged in as developers and 9 new projects were instantiated. On average, 40 vCPU, 60 GB of memory, three 10GB MIG partitions of one nVidia A100 GPU, three nVidia RTX5000 GPUs, and one nVidia A100 (not partitioned) were allocated simultaneously for interactive usage. The two projects contributing to the development of batch job functionalities as beta testers submitted 11876 jobs since June 2024. Since the introduction of the platform, no new AI_INFN administrator was nominated, and project-dedicated VMs have only been deployed for technological research and development on the computing infrastructure itself.

Ongoing developments focus on integrating CINECA Leonardo as a remote computing backend through the InterLink *Virtual Kubelet* provider and extending the supported accelerators to include FPGAs.

In parallel, INFN Cloud is evaluating the adoption of *KubeFlow* [37] as a partial replacement of JupyterHub. As it happened for Jupyter, *KubeFlow* will be evaluated in a single-user environment first, and possibly promoted to a *Software-as-a-Service* at a later tage.

Although primarily a research and development project, the AI_INFN platform is gaining recognition from several small experiments within the AI_INFN research lines as a potential provider of GPU-accelerated computing resources, and is poised to play a trailblazing role in the future landscape of the INFN computing infrastructure.

Acknowledgements

The research presented in this paper was partially supported by "Ente Cassa di Risparmio di Firenze" under the grant CLOUD_ML. DC and GB are funded by the Italian Center for Super Computing (ICSC) as part of the NextGenerationEU plan.

References

[1] AMD Vitis™ AI Software, 2024. https://www.amd.com/en/products/software/vitis-ai.html. Accessed: 15/09/2024.
[2] Apache Airflow, 2024. https://airflow.apache.org/. Accessed: 12/12/2024.
[3] Borg, 2024. https://borgbackup.readthedocs.io/en/stable/#. Accessed: 15/09/2024.
[4] Conda, 2024. https://conda.io. Accessed: 15/09/2024.
[5] Docker Swarm Mode, 2024. https://docs.docker.com/engine/swarm/. Accessed: 12/12/2024.
[6] ELOG, 2024. https://elog.psi.ch/elog/. Accessed: 15/09/2024.
[7] Intel© Distribution of OpenVINO™ Toolkit, 2024. https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html. Accessed: 15/09/2024.
[8] InterLink, 2024. https://intertwin-eu.github.io/interLink/. Accessed: 15/09/2024.
[9] InterTwin, 2024. https://www.intertwin.eu/. Accessed: 15/09/2024.
[10] JuiceFS, 2024. https://juicefs.com/en/. Accessed: 15/09/2024.
[11] Jupyter Server Proxy, 2024. https://jupyter-server-proxy.readthedocs.io/en/latest/. Accessed: 15/09/2024.
[12] Kueue, 2024. https://kueue.sigs.k8s.io/. Accessed: 15/09/2024.
[13] Leonardo, 2024. https://leonardo-supercomputer.cineca.eu/. Accessed: 15/09/2024.
[14] MinIO, 2024. https://min.io/. Accessed: 15/09/2024.
[15] Multi-instance GPU, 2024. https://www.nvidia.com/it-it/technologies/multi-instance-gpu/. Accessed: 15/09/2024.
[16] Nomad by HashiCorp, 2024. https://www.nomadproject.io/. Accessed: 12/12/2024.

[17] nVidia DCGM Exporter, 2024. https://docs.nvidia.com/datacenter/cloud-native/gpu-telemetry/latest/dcgm-exporter.html. Accessed: 15/09/2024.
[18] NVIDIA GPU Operator, 2024. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/index.html. Accessed: 15/09/2024.
[19] PostgreSQL, 2024. https://www.postgresql.org/. Accessed: 15/09/2024.
[20] Python venv, 2024. https://docs.python.org/3/library/venv.html. Accessed: 15/09/2024.
[21] Rados Gateway, 2024. https://docs.ceph.com/en/reef/radosgw/. Accessed: 15/09/2024.
[22] Squashfs, 2024. https://www.kernel.org/doc/Documentation/filesystems/squashfs.txt. Accessed: 15/09/2024.
[23] Virtual Kubelet, 2024. https://virtual-kubelet.io/. Accessed: 15/09/2024.
[24] Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G.S., *et al.*: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. https://www.tensorflow.org/. Software available from tensorflow.org.
[25] Adamec M., Attebury G., Bloom K., Bockelman B., Lundstedt C., Shadura O., Thiltges J.: Coffea-casa: an analysis facility prototype, *EPJ Web Conferences*, vol. 251, 02061, 2021. doi: 10.1051/epjconf/202125102061.
[26] Anderlini L., Boccali T., Dal Pra S., Duma D., Giommi L., Spiga D., Vino G.: ML_INFN project: Status report and future perspectives, *EPJ Web of Conferences*, vol. 295, 2024. doi: 10.1051/epjconf/202429508013.
[27] Antonacci M., Salomoni D.: Leveraging TOSCA orchestration to enable fully automated cloud-based research environments on federated heterogeneous e-infrastructures, *PoS*, vol. ISGC&HEPiX2023, 020, 2023. doi: 10.22323/1.434.0020.
[28] Bergholm V., Izaac J., Schuld M., Gogolin C., Ahmed S., Ajith V., Alam M.S., *et al.*: PennyLane: Automatic differentiation of hybrid quantum-classical computations, 2022. https://arxiv.org/abs/1811.04968.
[29] Bockelman B., Livny M., Lin B., Prelz F.: Principles, technologies, and time: The translational journey of the HTCondor-CE, *Journal of Computational Science*, vol. 52, 101213, 2021. doi: 10.1016/j.jocs.2020.101213. Case Studies in Translational Computer Science.
[30] Ceccanti A., Hardt M., Wegh B., Millar A., Caberletti M., Vianello E., Licehammer S.: The INDIGO-Datacloud Authentication and Authorization Infrastructure, *Journal of Physics: Conference Series*, vol. 898(10), 102016, 2017. doi: 10.1088/1742-6596/898/10/102016.
[31] Chen S., Glioti A., Panico G., Wulzer A.: Boosting likelihood learning with event reweighting, *Journal of High Energy Physics*, vol. 2024, 117, 2024. doi: 10.1007/JHEP03(2024)117.
[32] Chollet F., *et al.*: Keras, https://keras.io, 2015.
[33] Ciangottini D.: rclone, 2022. https://github.com/DODAS-TS/rclone.
[34] Eddelbuettel D.: A Brief Introduction to Redis, 2022. https://arxiv.org/abs/2203.06559.

[35] FastML Team: fastmachinelearning/hls4ml, 2023. doi: 10.5281/zenodo.1201549.

[36] Grafana Labs: Grafana Documentation, 2018. https://grafana.com/docs/.

[37] Grant T., Karau H., Lublinsky B., Liu R., Filonenko I.: *Kubeflow for Machine Learning*, O'Reilly Media, 2020. https://books.google.it/books?id=YLICEAAAQBAJ.

[38] Janssens D., Brunbauer F., Flöthner K., Lisowska M., Muller H., Oliveri E., Orlandini G., *et al.*: Studying signals in particle detectors with resistive elements such as the 2D resistive strip bulk MicroMegas, *Journal of Instrumentation*, vol. 18(08), C08010, 2023. doi: 10.1088/1748-0221/18/08/C08010.

[39] Kluyver T., Ragan-Kelley B., Pérez F., Granger B., Bussonnier M., Frederic J., Kelley K., *et al.*: Jupyter Notebooks – a publishing format for reproducible computational workflows. In: F. Loizides, B. Schmidt (eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pp. 87–90, IOS Press, 2016.

[40] Lizzi F., Postuma I., Brero F., Cabini R., Fantacci M., Oliva P., Rinaldi L., *et al.*: Quantification of pulmonary involvement in COVID-19 pneumonia: an upgrade of the LungQuant software for lung CT segmentation, *The European Physical Journal Plus*, vol. 138, 2023. doi: 10.1140/epjp/s13360-023-03896-4.

[41] Mariani S., Anderlini L., Di Nezza P., Franzoso E., Graziani G., Pappalardo L.L.: A neural-network-defined Gaussian mixture model for particle identification applied to the LHCb fixed-target programme, *Journal of Physics: Conference Series*, vol. 2438(1), 012107, 2023. doi: 10.1088/1742-6596/2438/1/012107.

[42] Mariotti M., Magalotti D., Spiga D., Storchi L.: The BondMachine, a moldable computer architecture, *Parallel Computing*, vol. 109, 102873, 2022. doi: 10.1016/j.parco.2021.102873.

[43] NVIDIA, Vingelmann P., Fitzek F.H.: CUDA, release: 10.2.89, 2020. https://developer.nvidia.com/cuda-toolkit.

[44] Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., *et al.*: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[45] Salomoni D., Campos I., Gaido L., de Lucas J.M., Solagna P., Gomes J., Matyska L., *et al.*: INDIGO-DataCloud: a Platform to Facilitate Seamless Access to E-Infrastructures, *Journal of Grid Computing*, vol. 16(3), pp. 381–408, 2018. doi: 10.1007/s10723-018-9453-3.

[46] Schneppenheim M.: Kube eagle, 2020. https://github.com/cloudworkz/kube-eagle.

[47] Stetzler S., Jurić M., Boone K., Connolly A., Slater C.T., Zečević P.: The Astronomy Commons Platform: A Deployable Cloud-based Analysis Platform for Astronomy, *The Astronomical Journal*, vol. 164(2), 68, 2022. doi: 10.3847/1538-3881/ac77fb.

[48] Tejedor E., Bocchi E., Castro D., Gonzalez H., Lamanna M., Mato P., Moscicki J., *et al.*: Facilitating Collaborative Analysis in SWAN, *EPJ Web Conferences*, vol. 214, 07022, 2019. doi: 10.1051/epjconf/201921407022.

[49] Weil S.A., Brandt S.A., Miller E.L., Long D.D.E., Maltzahn C.: Ceph: a scalable, high-performance distributed file system. In: *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, pp. 307–320, OSDI '06, USENIX Association, USA, 2006.

[50] Winikoff M., Padgham L.: The Prometheus Methodology. In: F. Bergenti, M.P. Gleizes, F. Zambonelli (eds.), *Methodologies and Software Engineering for Agent Systems*, pp. 217–234, Springer, Boston, 2004. doi: 10.1007/1-4020-8058-1_14.

[51] Yoo A.B., Jette M.A., Grondona M.: SLURM: Simple Linux Utility for Resource Management. In: D. Feitelson, L. Rudolph, U. Schwiegelshohn (eds.), *Job Scheduling Strategies for Parallel Processing*, pp. 44–60, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

Affiliations

Lucio Anderlini
Istituto Nazionale di Fisica Nucleare, Sezione di Firenze, via G. Sansone 1, Sesto Fiorentino (FI), 50019, Italy

Matteo Barbetti
Istituto Nazionale di Fisica Nucleare, CNAF, Viale Berti Pichat 6/2, Bologna (IT), 40127, Italy

Giulio Bianchini
Istituto Nazionale di Fisica Nucleare, Sezione di Perugia, via A. Pascoli, Perugia (PG), 06123, Italy

Diego Ciangottini
Istituto Nazionale di Fisica Nucleare, Sezione di Perugia, via A. Pascoli, Perugia (PG), 06123, Italy

Stefano Dal Pra
Istituto Nazionale di Fisica Nucleare, CNAF, via G. Sansone 1, Sesto Fiorentino (FI), 50019, Italy

Diego Michelotto
Istituto Nazionale di Fisica Nucleare, CNAF, Viale Berti Pichat 6/2, Bologna (IT), 40127, Italy

Rosa Petrini
Istituto Nazionale di Fisica Nucleare, Sezione di Firenze, via G. Sansone 1, Sesto Fiorentino (FI), 50019, Italy, rosa.petrini@fi.infn.it

Daniele Spiga
Istituto Nazionale di Fisica Nucleare, Sezione di Perugia, via A. Pascoli, Perugia (PG), 06123, Italy

Received: 10.03.2025
Revised: 12.03.2025
Accepted: 12.03.2025