Prototype of a cloud native solution of Machine Learning as Service for HEP

L. Giommi,[a,1,*] **D. Spiga,**[b,1] **V. Kuznetsov**[c,1] **and D. Bonacorsi**[a,1]

[a] *University of Bologna and INFN section of Bologna, Bologna, Italy*
[b] *INFN section of Perugia, Perugia, Italy*
[c] *Cornell University, Ithaca, USA*
E-mail: luca.giommi3@unibo.it, daniele.spiga@pg.infn.it, vkuznet@protonmail.com, daniele.bonacorsi@unibo.it

To favor the usage of Machine Learning (ML) techniques in High-Energy Physics (HEP) analyses it would be useful to have a service allowing to perform the entire ML pipeline (in terms of reading the data, training a ML model, and serving predictions) directly using ROOT files of arbitrary size from local or remote distributed data sources. The MLaaS4HEP framework aims to provide such kind of solution. It was successfully validated with a CMS physics use case which gave important feedback about the needs of analysts. For instance, we introduced the possibility for the user to provide pre-processing operations, such as defining new branches and applying cuts. To provide a real service for the user and to integrate it into the INFN Cloud, we started working on MLaaS4HEP cloudification. This would allow to use cloud resources and to work in a distributed environment. In this work, we provide updates on this topic, and in particular, we discuss our first working prototype of the service. It includes an OAuth2 proxy server as authentication/authorization layer, a MLaaS4HEP server, an XRootD proxy server for enabling access to remote ROOT data, and the TensorFlow as a Service (TFaaS) service in charge of the inference phase. With this architecture the user is able to submit ML pipelines, after being authenticated and authorized, using local or remote ROOT files simply using HTTP calls.

41st International Conference on High Energy physics - ICHEP2022
6-13 July, 2022
Bologna, Italy

[1] For the CMS collaboration.
[*] Speaker

1. The MLaaS4HEP framework

Nowadays ML techniques are successfully used in many areas of HEP, e.g. in detector simulation, object reconstruction, identification, and Monte Carlo generation. ML will play a significant role also in the upcoming High-Luminosity LHC upgrade foreseen at CERN, when a huge amount of data will be produced by LHC and collected by the experiments, facing challenges at the exascale. A project originally started in CMS [1] is working on a ML as a Service (MLaaS) solution [2] to increase and ease the use of ML in HEP analyses, which consists of two parts. The first one, the MLaaS for HEP (MLaaS4HEP) framework [3], covers the data reading, data processing, and ML model training phases, in a completely model-agnostic fashion, directly using ROOT files of arbitrary size from local or distributed data sources. The second one, the TFaaS framework [4], can be used to host pre-trained Tensor based ML models and obtain predictions via HTTP calls.

The MLaaS4HEP framework is written in the Python programming language, it uses the Uproot [5] library to read ROOT data, and it implements a Python Generator that reads, handles, and delivers data in chunks to the ML model chosen by the user for training. Such implementation provides an efficient access to large datasets since it does not require loading the entire dataset into the RAM of the training node. The MLaaS4HEP workflow consists in two phases. In the first one, the input data are read in chunks to compute a specs file containing useful information about the ROOT files, e.g. the maximum dimension of each Jagged branch, the maximum and minimum value for each branch, and the number of events in each ROOT file. In the second one, the events are read in chunks, converted into NumPy arrays, with the necessary transformation of the Jagged/Awkward Arrays [6] dimensions, normalization of the values (based on the information computed during the first phase), and application of pre-processing operation, e.g. creation of new branches and application of cuts on the branches. These chunks are then used to train the ML model chosen by the user.

The creation of a Software as a Service (SaaS) solution for MLaaS4HEP (and therefore a MLaaS) takes place through a process of automating the platform deployment to be delivered to the user, and this work was done using DODAS as Platform as a Service (PaaS) [7]. The user can access the VM via browser and exploit the JupyterHub interface to upload all the necessary data and run MLaaS4HEP.

2. Cloud native solution for MLaaS4HEP

The MLaaS4HEP framework is not yet a service, meaning that it lacks the APIs through which a user can interact with it. To be integrated into a cloud and to be designed as a cloud native application, the final product should be developed in terms of interconnected microservices each of them in charge of different tasks. To provide such a solution we worked through a series of steps:

- we built a MLaaS4HEP server with some APIs using the (Python-based) Flask framework [8];
- we integrated an OAuth2 Proxy server [9] to manage users authentication/authorization;

- we integrated an XRootD Proxy server [10] to allow the usage of an X.509 proxy for the remote access to ROOT data;
- we connected the MLaaS4HEP server with TFaaS in a way that the ML models trained by the MLaaS4HEP server are saved in a repository from which the TFaaS service can take them for the inference phase.

We implemented a working prototype connecting the aforementioned services hosted by a VM of the INFN Cloud (see Fig. 1 for a schematic representation of the solution): the MLaaS4HEP server APIs can be reached at the following address `https://90.147.174.27:4433`, while the TFaaS ones at `https://90.147.174.27:8081`. Once the user obtains an access token from the authorization server, he/she can contact the MLaaS4HEP server or TFaaS using curl, e.g. in the following ways:

```
curl -L -k -H "Authorization: Bearer ${TOKEN_MLAAS}" -H "Content-Type:
application/json" -d @submit.json https://90.147.174.27:4433/submit

curl -L -k -H "Authorization: Bearer ${TOKEN_TFAAS}" -X POST -H "Content-type:
application/json" -d @predict_bkg.json https://90.147.174.27:8081/json
```

The former command allows to submit a MLaaS4HEP workflow (using the values of the parameters stored in the `submit.json` file) which allow to train a ML model, whereas the latter allows using this model to get the prediction on a given event (stored in the `predict_bkg.json` file).

References

[1] CMS Collaboration, *The CMS experiment at the CERN LHC*, *JINST* **3** (2008) S08004

[2] V. Kuznetsov, L. Giommi, D. Bonacorsi, *MLaaS4HEP: Machine Learning as a Service for HEP*, *Comput Softw Big Sci* **5**, 17 (2021).

[3] *MLaaS4HEP*. https://github.com/vkuznet/MLaaS4HEP

[4] *TFaaS*. https://github.com/vkuznet/TFaaS

[5] *DIANA-HEP Scikit-hep uproot library. Minimalist ROOT I/O in pure Python and NumPy*. https://github.com/scikit-hep/uproot4

[6] *Awkward Array*. https://awkward-array.readthedocs.io/en/latest/

[7] L. Giommi, V. Kuznetsov, D. Bonacorsi, D. Spiga, *Machine Learning as a Service for High Energy Physics on heterogeneous computing resources*, in proceedings of *International Symposium on Grids and Clouds*, Proceedings of Science SISSA, 2021.

[8] L. Giommi, D. Spiga, V. Kuznetsov, D. Bonacorsi and M. Paladino, *Cloud native approach for Machine Learning as a Service for High Energy Physics*, in proceedings of *International Symposium on Grids and Clouds*, Proceedings of Science SISSA, 2022.

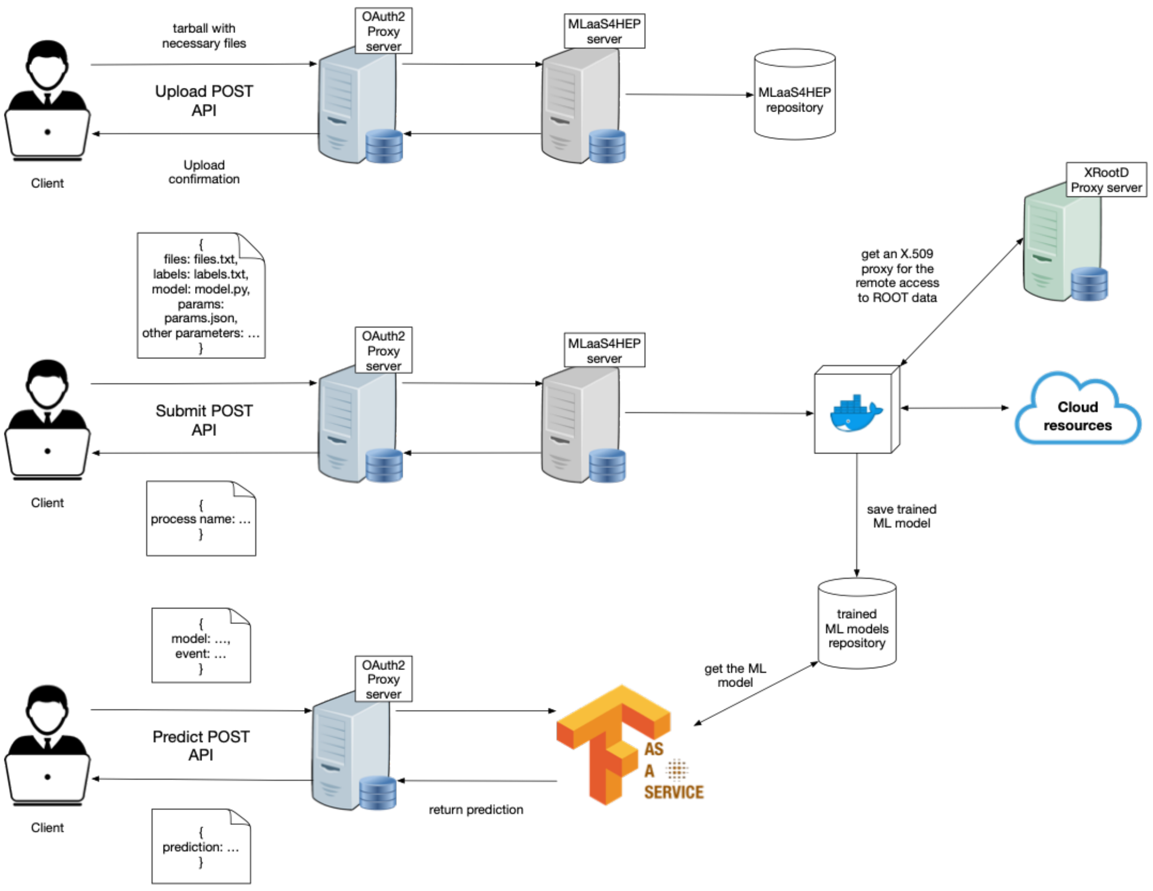

Figure 1: Solution connecting OAuth2-Proxy server, MLaaS4HEP server, XRootD Proxy server, and TFaaS. Firstly, a client uploads a tarball with the necessary files to the MLaaS4HEP server, then submits a MLaaS4HEP workflow which consists in running a Docker container in the server, which can read remote ROOT files using valid X.509 proxies, and that trains and saves the ML model. This model is accessed by the TFaaS service that uses it to make inference. In front of the MLaaS4HEP server and TFaaS server, two OAuth2-Proxy servers are used to handle user authentication and request authorization.

[9] *OAuth2-Proxy server*. https://oauth2-proxy.github.io/oauth2-proxy/

[10] *compose-xrootd*. https://github.com/comp-dev-cms-ita/compose-xrootd