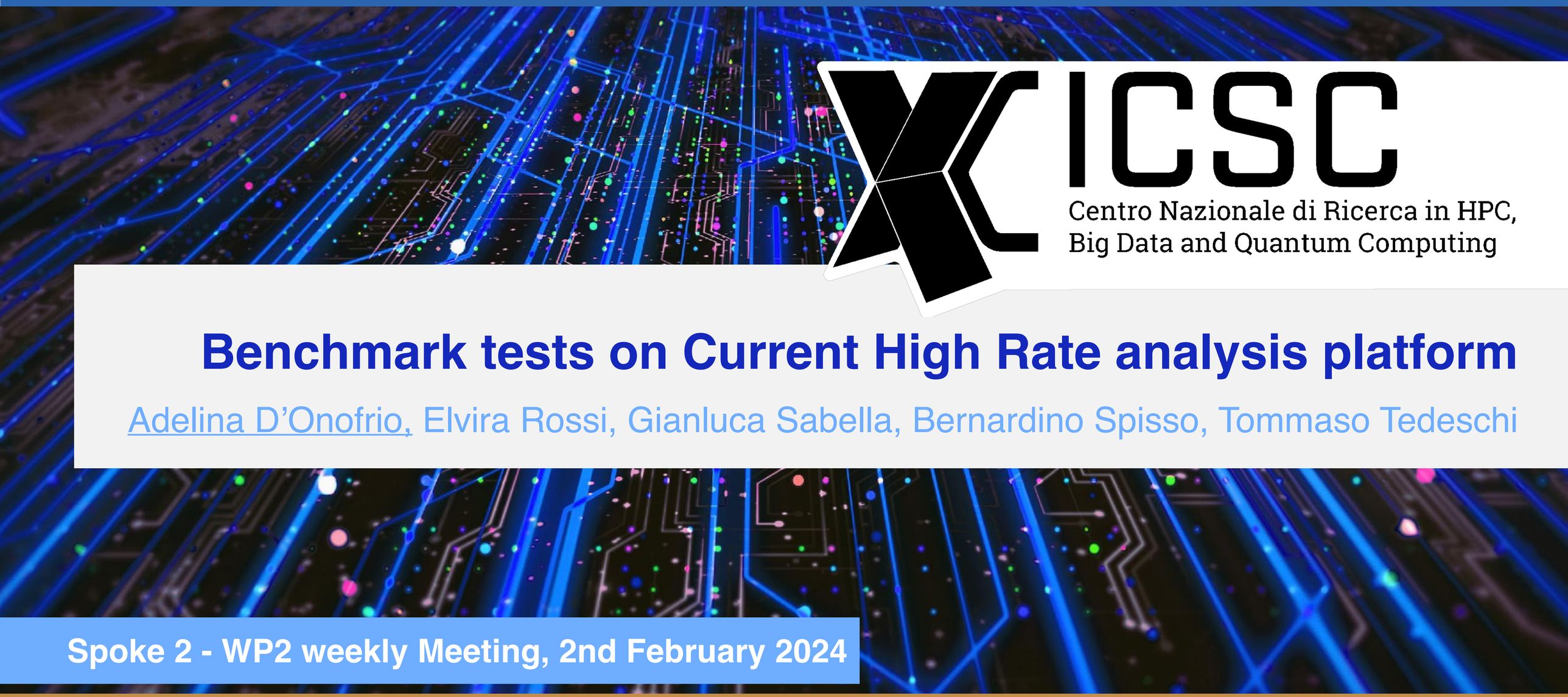
Finanziato dall'Unione europea
NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing



ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Benchmark tests on Current High Rate analysis platform

Adelina D'Onofrio, Elvira Rossi, Gianluca Sabella, Bernardino Spisso, Tommaso Tedeschi

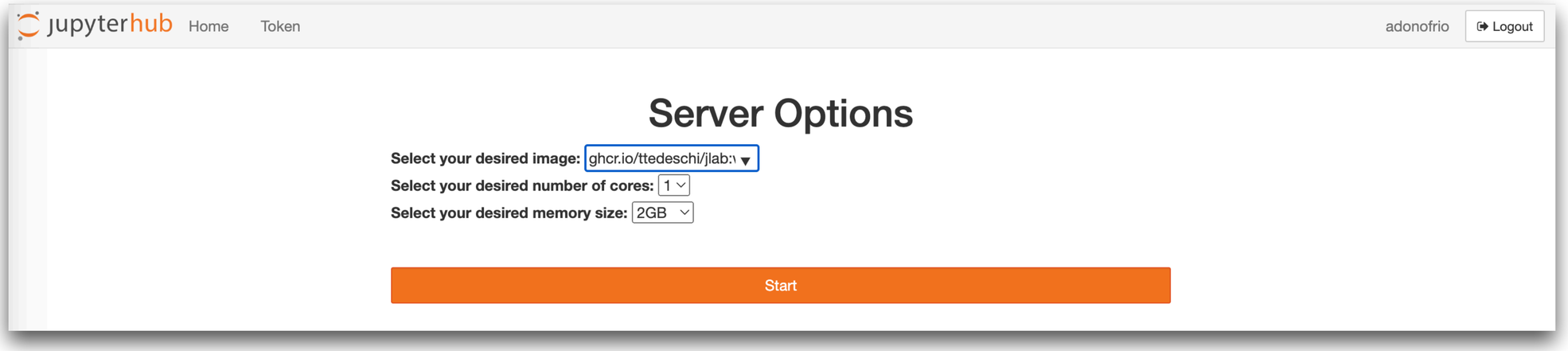
Spoke 2 - WP2 weekly Meeting, 2nd February 2024

Outline

- **Goal**: replicate the simple benchmark use case on the current high rate platform
- Compare the performance with the local infrastructure used previously (more in back-up)
- Use case tested:
  - FCCee: simple test on Zee samples
  - Scaling with #cpu and memory on h.r.p.
- First documentation ideas/efforts

Resources used

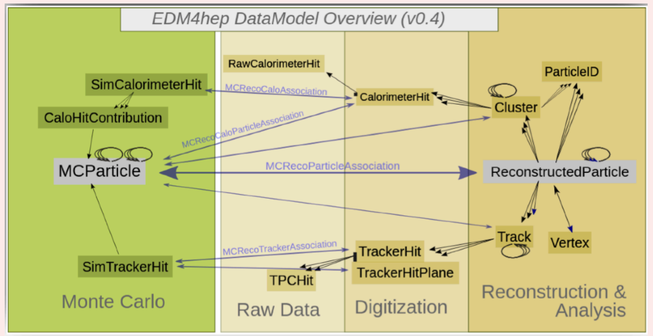
- High rate platform shown by ***Tommaso in the last meeting***, thanks a lot!
- Entrypoint: https:/hub.192.135.24.49.myip.cloud.infn.it
- Login via IAM DEMO: https:/iam-demo.cloud.cnaf.infn.it/
- We used one of the available ready-to-use JupyterLab images:
  - ghcr.io/ttedeschi/jlab:wp5-alma8-0.0.40 (almalinux8 + python3.11 + Dask + ROOT 6.30)



FCCee use-case

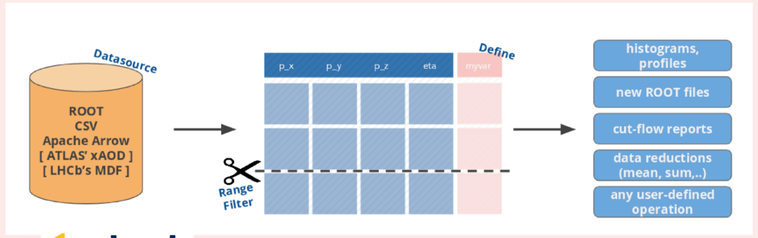
Workflow

EDM4hep input data format



flat input ntuples

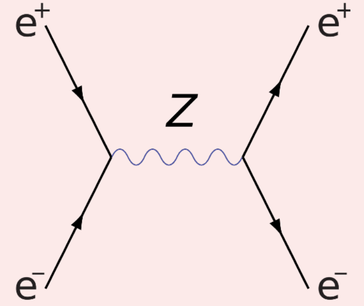
RDataFrame

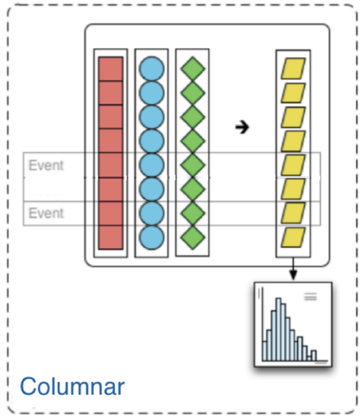


+ dask used as backend

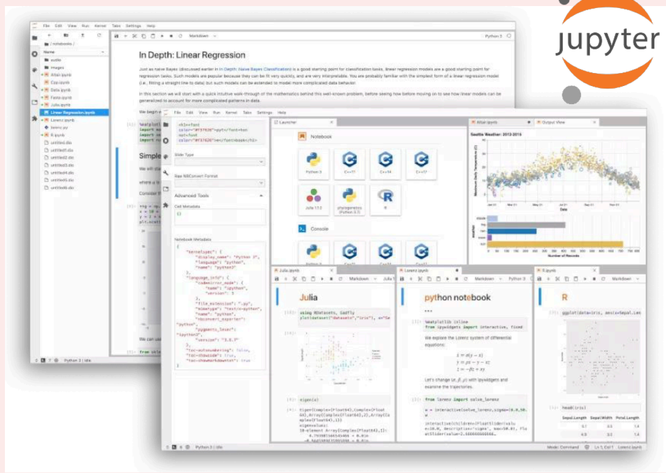
Standard test

Use case





New approach to data analysis



Feasibility study & Preliminary performance evaluation

13/10 WP2.5 presentation *link*

Preliminary results: local client

```
from dask.distributed import LocalCluster, Client
if distributed == True:
    RDataFrame = ROOT.RDF.Experimental.Distributed.Dask.RDataFrame
    ROOT.RDF.Experimental.Distributed.initialize(my_initialization_function)
else:
    RDataFrame = ROOT.RDataFrame
    my_initialization_function()
```

Parallel

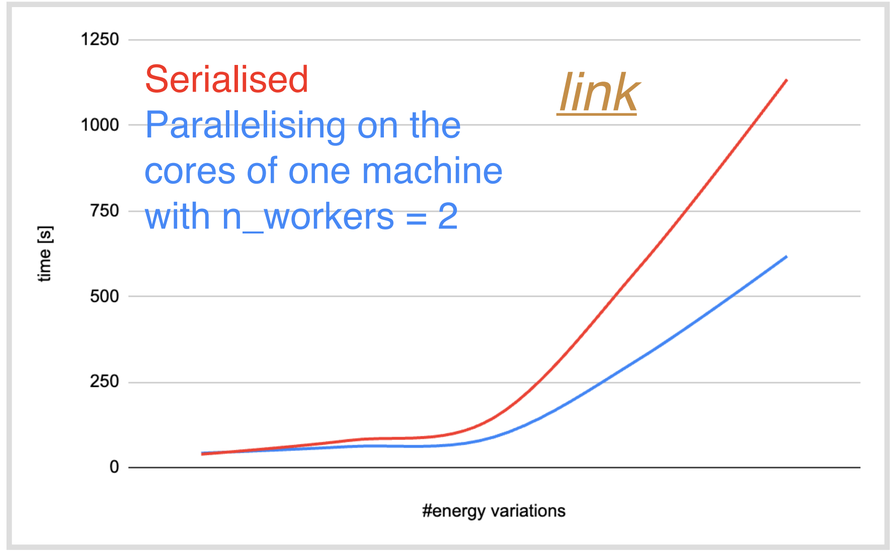
Serialised

Local client

```
cluster = LocalCluster(n_workers=2, threads_per_worker=1, processes=True, dashboard_address=':8786')
client = Client(cluster)
```

Defined Metric	
Overall execution time	Time elapsed from the start of the execution (execution triggered) to the end of execution

INFN Napoli infrastructure



High rate platform, 100 iterations

# cpu	memory [GB]	execution time
1	2 GB	482 s
2	4 GB	341 s
4	8 GB	258 s
8	16 GB	245 s

hyper threading?

- Performance comparable between the 2 infrastructures exploited
- Advantage: use this use case as simple test for who wants to benefit from the WP5 infrastructure

First attempt: distributed client

```python
from dask_kubernetes.operator import KubeCluster

client = _get_global_client()

if client is not None:
    client.shutdown()

cluster = KubeCluster(
    name=f'{username}',
    namespace=f'user-{username}',
    image='ghcr.io/ttedeschi/jlab:wp5-alma8-0.0.40',
    n_workers=10,
    worker_command=[
        'dask-worker',
        '--name=$(DASK_WORKER_NAME)',
        '--local-directory=temp_dir',
        '--nthreads', '8',
        '--death-timeout', '60',
        '--memory-limit','16Gi'
    ],

    resources={
        "requests": {
            "memory": "16Gi",
            "cpu": "8"
        },
        "limits": {
            "memory": "16Gi",
            "cpu": "8"
        }
    }
)
c_distributed = Client(cluster)
```

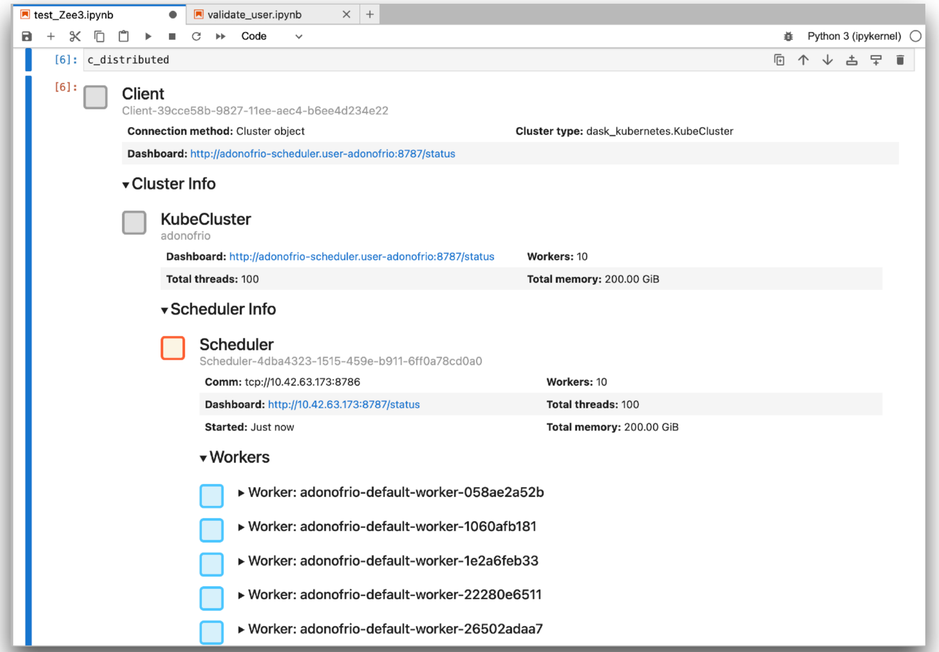
```
RuntimeError: Failed to create DaskCluster resource.Are the Dask Custom Resource Definitions installed? https://kubernetes.dask.org/en/latest/operator.html#installing-the-operator
```

High rate platform

- What's going wrong?
- Any suggestion?

```
RuntimeError
Cell In[4], line 8
      5 if client is not None:
      6     client.shutdown()
----> 8 cluster = KubeCluster(
```

INFN Napoli infrastructure



- 5 Kubernetes workers & 1 Kubernetes master on *Open-stack*

# iterations	Serial approach	Local client Dask	Distributed Dask
100	1135 s	618 s	138 s

Documentation efforts on gitlab

➡ **Admin's point of view:**

- Providing a detailed guide for users who have obtained access to the national cloud and wish to build their own infrastructure using the INFN Cloud interface
- The guide will cover every essential step, from configuration to usage, enabling each user to fully harness the potential of the cloud platform:
  - access to INFN Cloud
  - resource selection
  - infrastructure deployment
  - access and description of the infrastructure

➡ **Our goal is to complete the guide within 15-20 days**

➡ **User's point of view:**

- Exploit the Z_{ee} use case as benchmark also for the documentation
- Guide the user through the code highlighting which functionality of the infrastructure we use step by step

Conclusions & Next Steps

- Simple tests performed on the high rate platform
- Performance in line with the local infrastructure previously used
- Distributed client, to be investigated
- First ideas for the documentation

Thank you!

Back-up

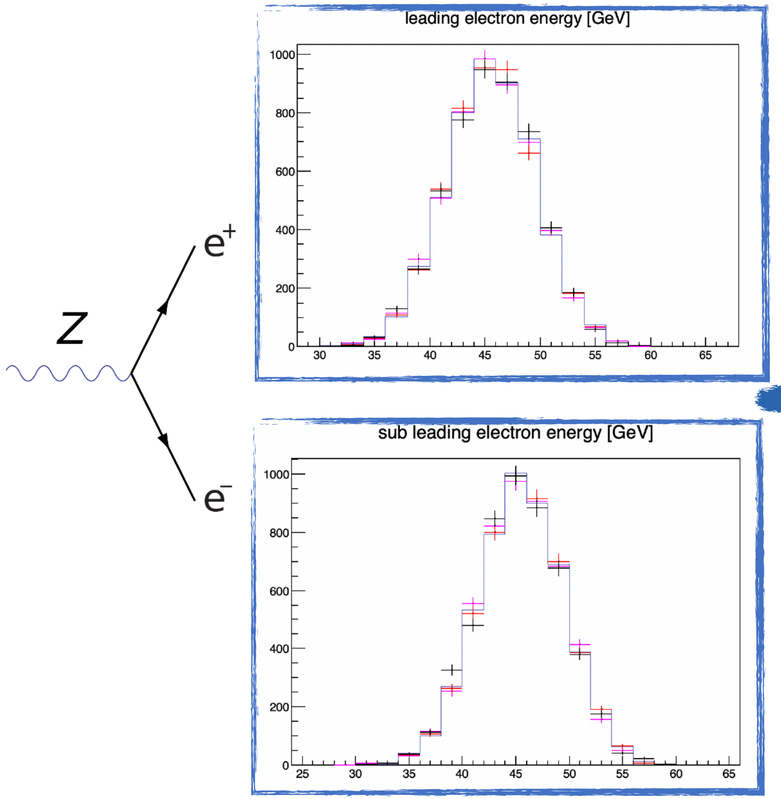
INFN Napoli infrastructure

- The local deployment is based on the *Open-Stack IaaS* paradigm
- Starting from the already existing *I.Bi.S.CO* installation, several updates were performed
- The cluster is made up of 2 identical virtual machines, each equipped with 1CPU quadCore and 8GB RAM, currently expanded up to 12 cores and 64GB
- Rocky Linux 8.6 is the operating system
- 2 nodes are equipped with **Docker** (20.10) for containerisation and **Kubernetes** (1.26.3) for the orchestration
  - 📌 One node plays as controlplane, etcf & worker; the other node acts as a plain worker
- The cluster is equipped with **JupyterHub** & **JupyterLAB** where the user can play with **Python, ROOT & Dask** libraries

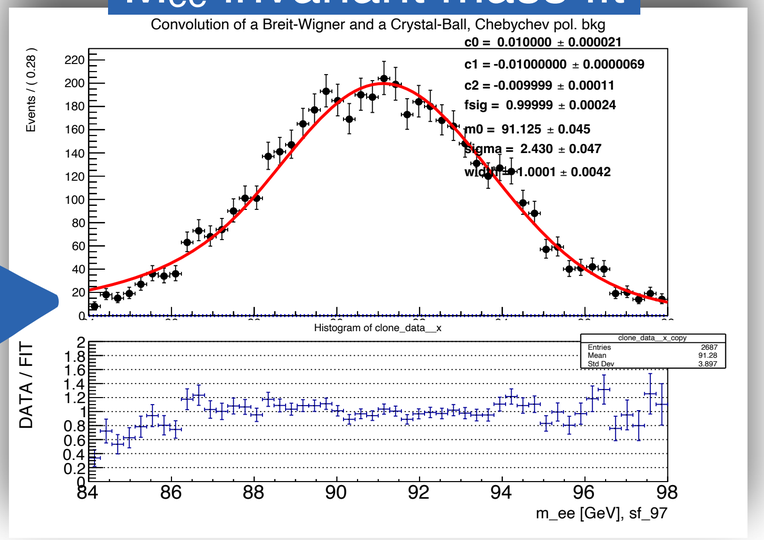
13/10 WP2.5 presentation *link*

Simple test

- FCCee simulation: /eos/experiment/fcc/ee/tmp/ee_Z_ee_EDM4Hep.root
  - 5k events, scaled to 1M events replicating the available dataset
  - Mimic systematic variations, gaussian smearing the electrons energy to compute M_{ee} resolution



M_{ee} invariant mass fit



Selection and histogramming interactively via RDataFrame on JupyterHub

github link to the code