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 interactive analysis for future colliders

Adelina D'Onofrio, Elvira Rossi, Antimo Cagnotta, Gianluca Sabella, Bernardino Spisso
on behalf of WP2.5

Spoke 2 Annual Meeting, December 18-20th 2023,
CINECA, Casalecchio di Reno (BO)

Outline

- Motivations
- Test infrastructure
- Use case example, in a future collider context
  - Scalability results
- Conclusions

16:55 **Developing and testing of a flexible and scalable high rate analysis platform**
Speaker: Gianluca Sabella (Istituto Nazionale di Fisica Nucleare)

17:05 **Benchmark interactive analysis at future colliders** 10m
Speaker: Adelina D'Onofrio (Istituto Nazionale di Fisica Nucleare)

17:15 **Quasi interactive analysis of big data with high throughput - Initial steps and future perspectives**
Speakers: Francesco Giuseppe Gravili (Istituto Nazionale di Fisica Nucleare), Tommaso Diotalevi (Università e INFN, Bologna)

09:50 **Evolving High Rate Analysis infrastructure with seamless offloading on different type of providers**
Speaker: Tommaso Tedeschi (Università e INFN Perugia)

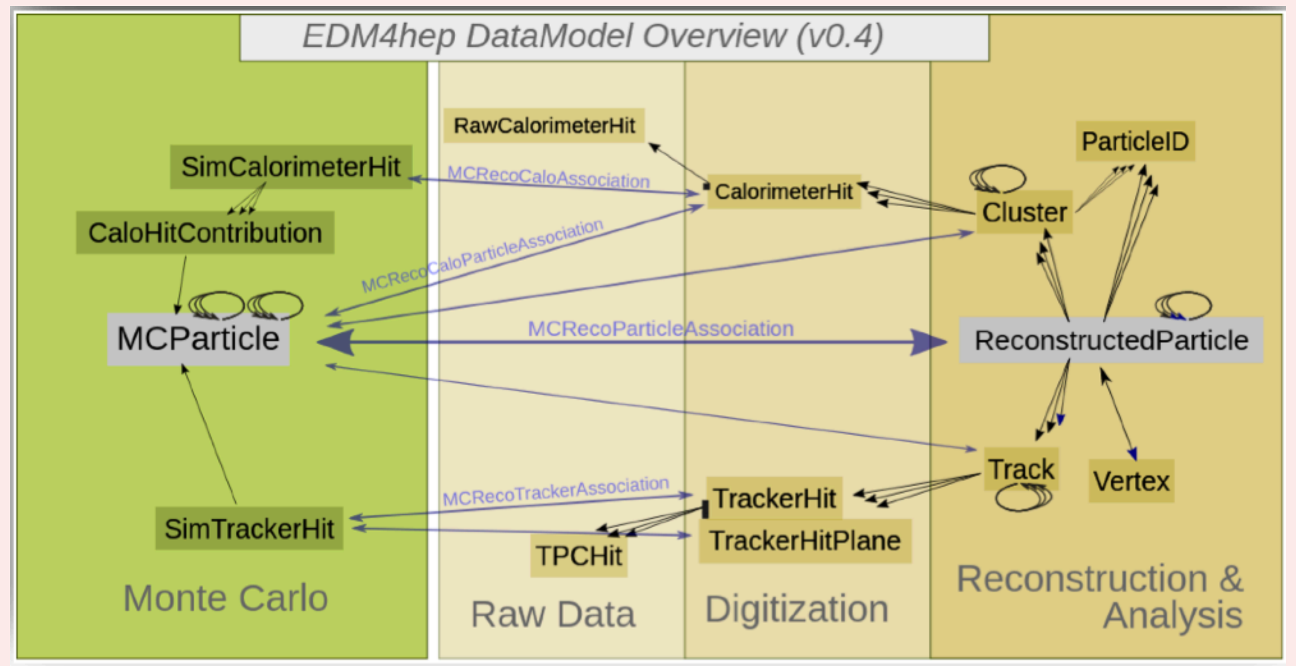
Motivations

- **Challenges of the future** e^+e^- colliders are pushing to **re-think the HEP computing models**
  - Impact on several aspects, from software to the computing infrastructure
- From the software perspective, **interactive/quasi interactive analysis** is a promising paradigm
  - User-friendly environment
  - Adopting open-source industry standards: *Dask, Jupyter Notebooks* and *HTCondor*
  - Validating new frameworks (e.g. *ROOT RDataFrame* with multi-threading)
  - More in ***Francesco Gravili & Tommaso Diotalevi's talk***, WP2
- **Preliminary feasibility studies** exploiting **future** e^+e^- **colliders pseudo-data**
  - Local testbed **infrastructure for high throughput data analysis**
  - The local deployment is based on the *Open-Stack IaaS* paradigm
  - The cluster is made up of 2 identical virtual machines, each equipped with 12 cores and 64GB
  - Rocky Linux 8.6 is the operating system

Gianluca's talk

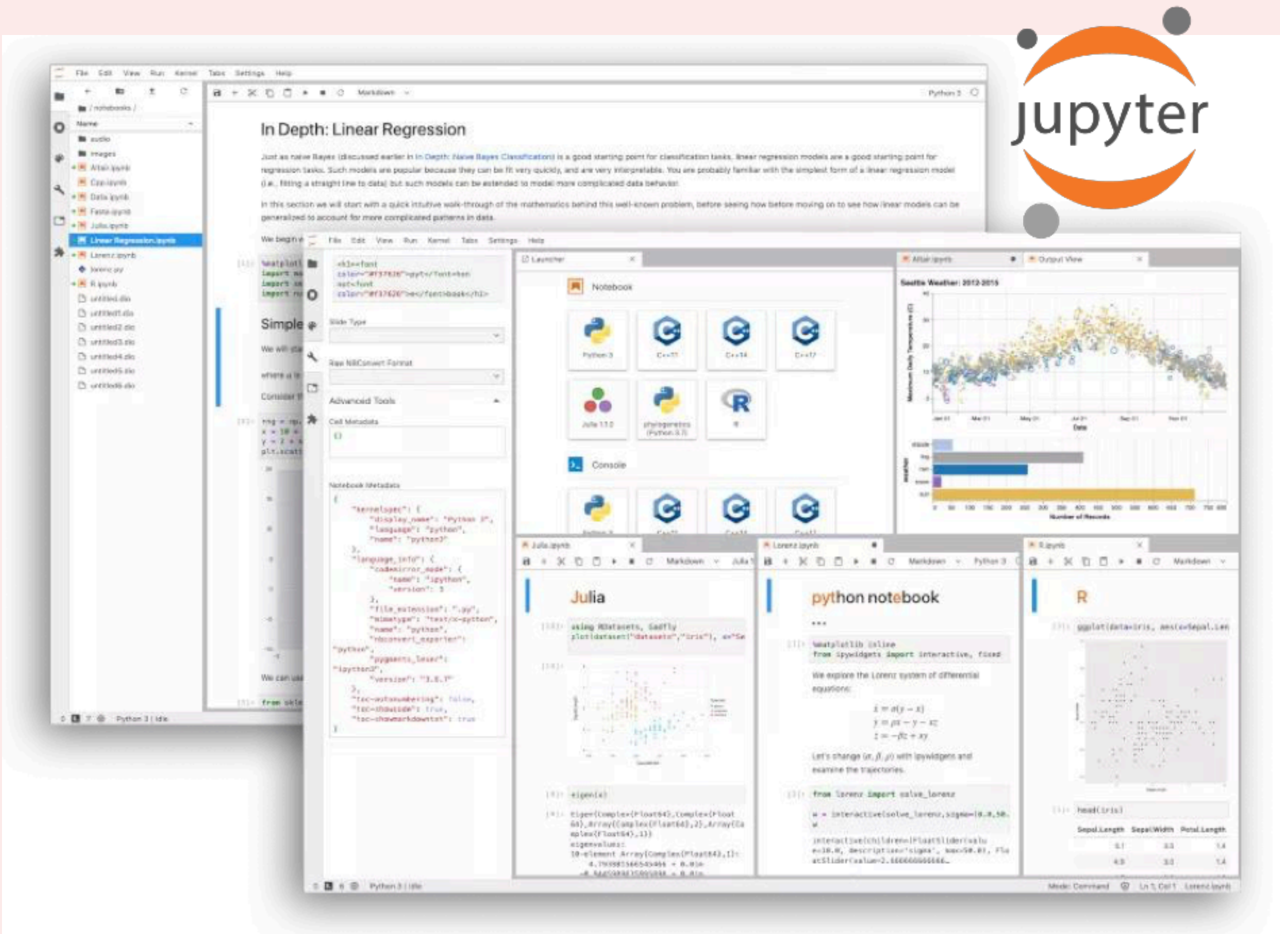
Use case: Future colliders

EDM4hep input data format



flat input ntuples

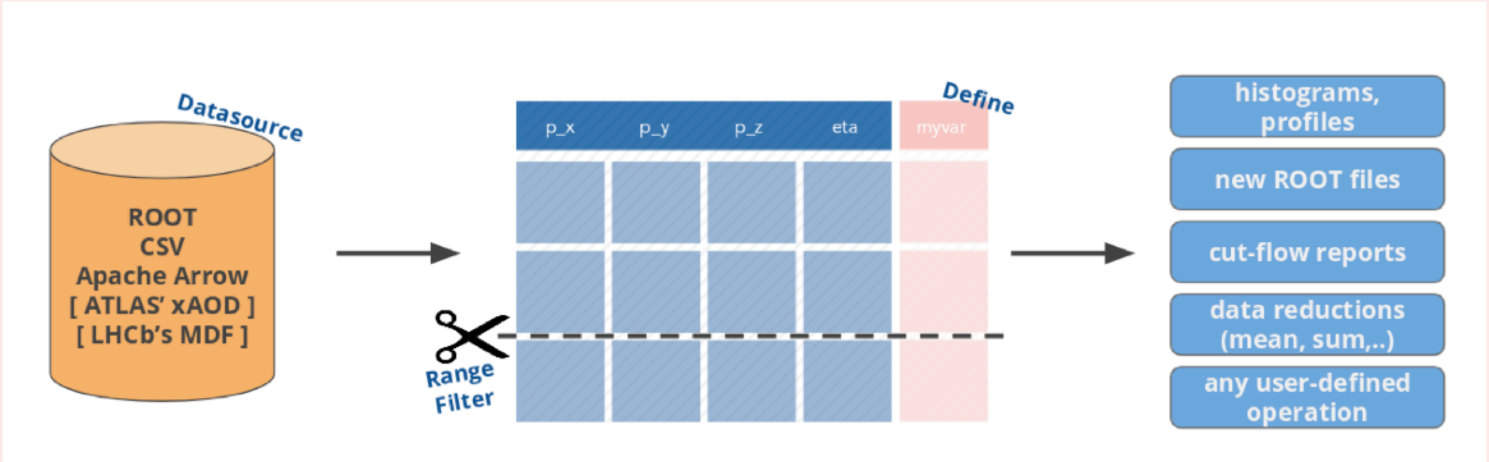
New approach to data analysis



Selection and histogramming interactively via RDataFrame on JupyterHub

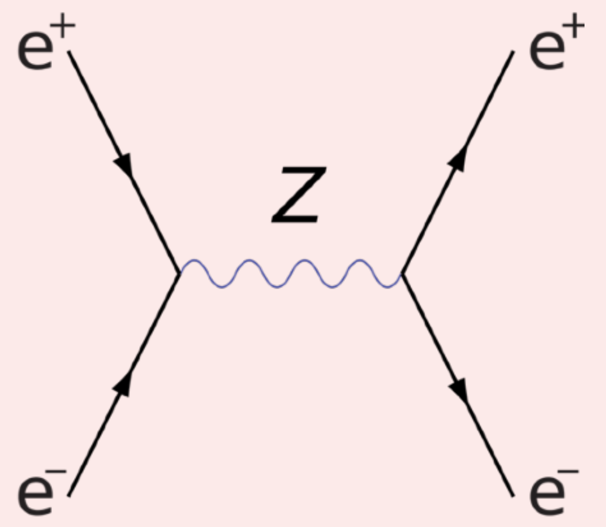
+

[RDataFrame]

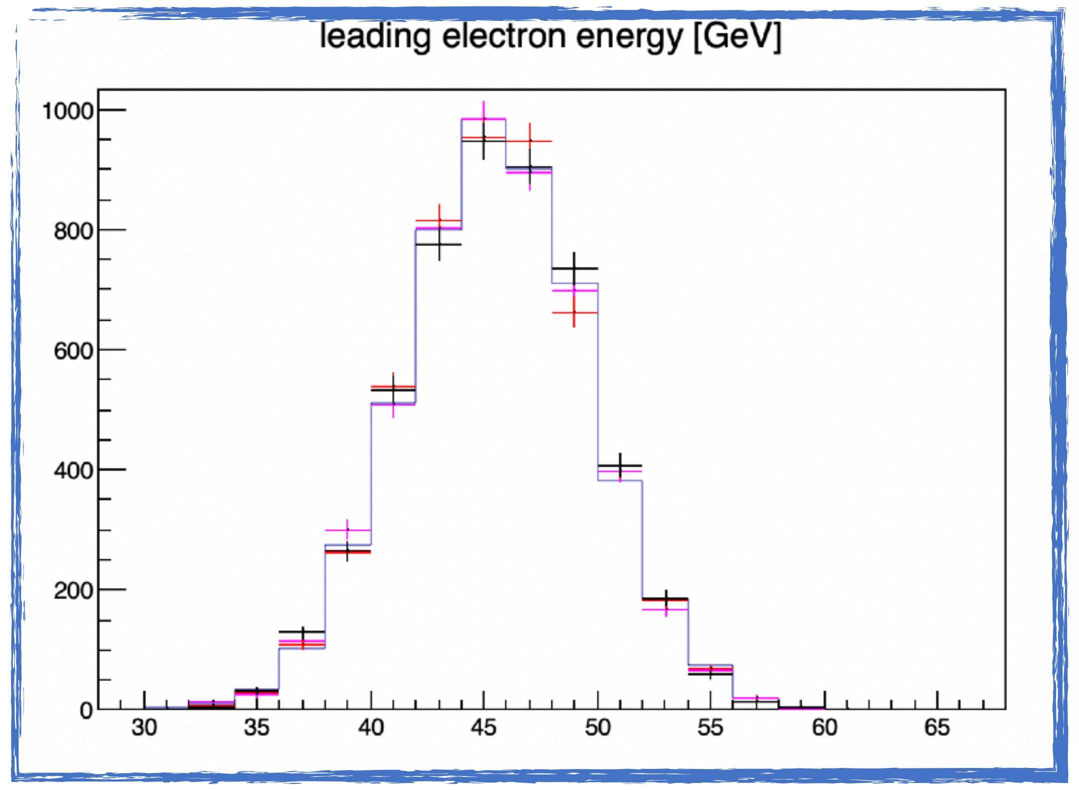


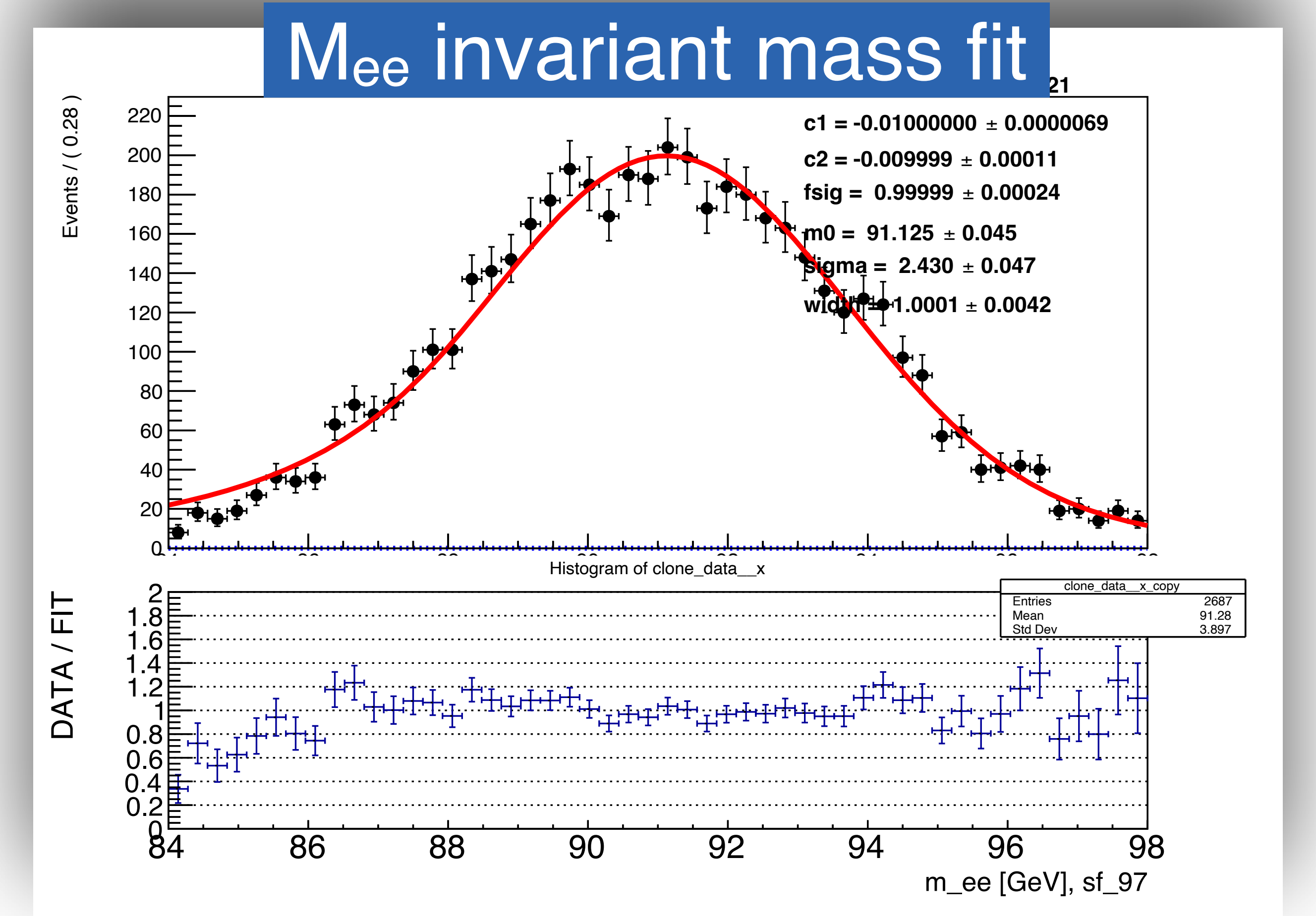
+ dask used as backend

Feasibility study



:





Mimic systematic variations: e^+e^- energy gaussian smearing

ECFA presentation [link]

[github link to the code]

Preliminary results: local client

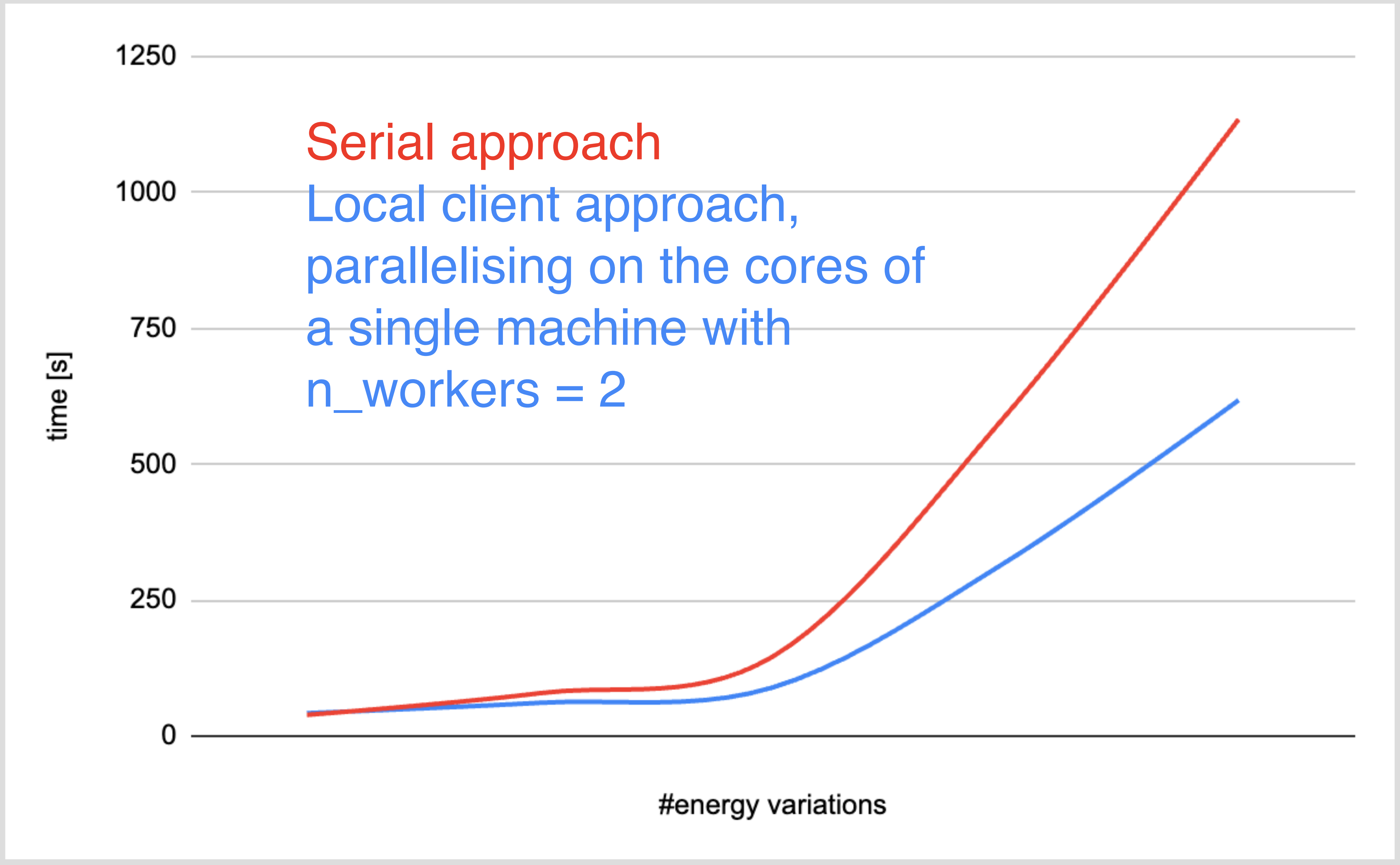

```
from dask.distributed import LocalCluster, Client
if distributed == True:
    RDataFrame = ROOT.RDF.Experimental.Distributed.Dask.RDataFrame
    ROOT.RDF.Experimental.Distributed.initialize(my_initialization_function)
else:
    RDataFrame = ROOT.RDataFrame
    my_initialization_function()
```

} Parallel

} Serial

⋮ **No changes required to the rest of the code**

```
df = df.Define('w_nominal', '1')
df = df.Define("m_e","0.0005124") #GeV
df_ge = df.Define("goodelectrons", "Particle.charge[0]*Particle.charge[1] < 0.").Filter("goodelectrons > 1")
```



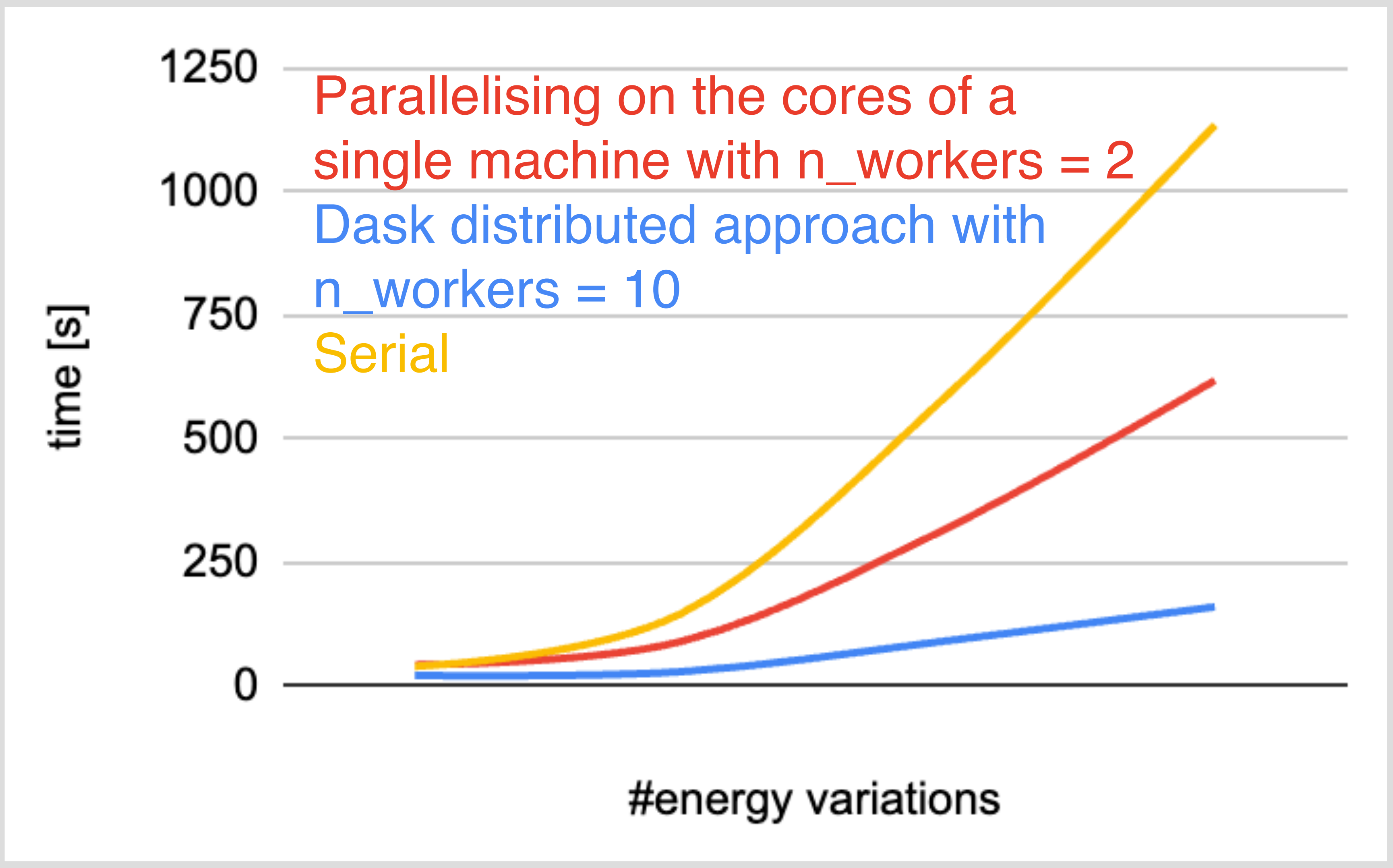
How to compare the performance?

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

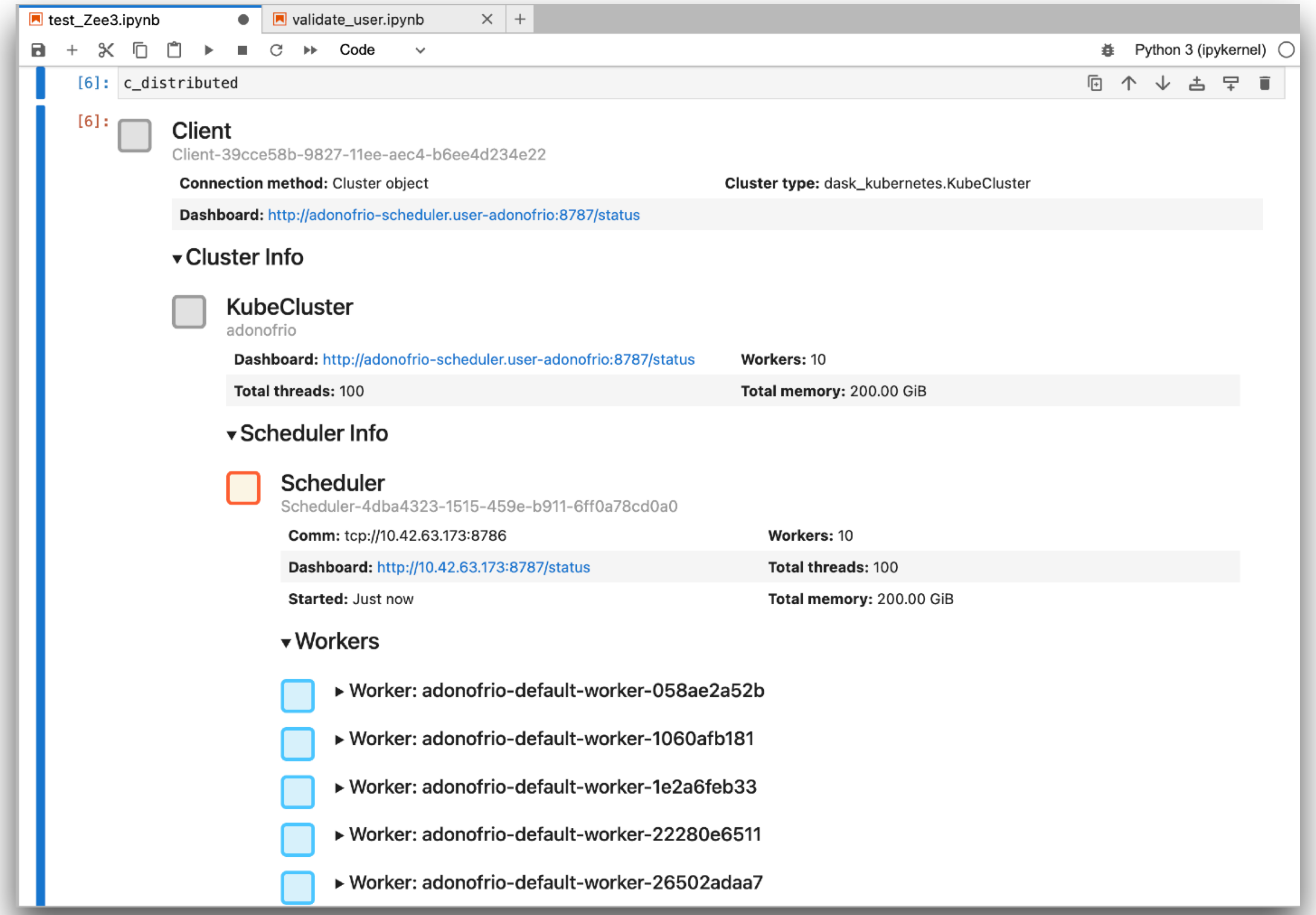
- Exploiting the local client approach, the execution time improves *wrt* the standard/serial approach if we iterate over a significative number of energy variations (> 10)

Preliminary results: distributed cluster

- Kubernetes infrastructure: 5+1 virtual machines (5 Kubernetes workers & 1 Kubernetes master) on *Open-stack*



# iterations	Serial approach	Local client Dask	Distributed Dask
50	590 s	320 s	75 s
100	1135 s	618 s	138 s



- Moving to a distributed Dask model and **scaling resources, the performance improves**
- Advantage: use this use case as simple test for who wants to benefit from the **WP5** infrastructure

Conclusions & Next Steps

- Interactive analyses feasibility studies on the local testbed infrastructure succeeded
  - Performance evaluated using Dask on the local cluster or distributed, *wrt* original implementation
- Very productive collaboration with other work packages

➡ **Short term goals:**

- Deploy of the code & relative instructions to allow other users to test quasi interactive high throughput data analysis platform
  - Benchmark studies with local performance evaluation

➡ **Medium-long term goals:**

- Automate the high throughput data analysis deployment exploiting the ICSC computing resources
- Evaluate scalability and simultaneous performance with increasing number of workers

- More in ***Tommaso Tedeschi's talk*** !

Thank you!

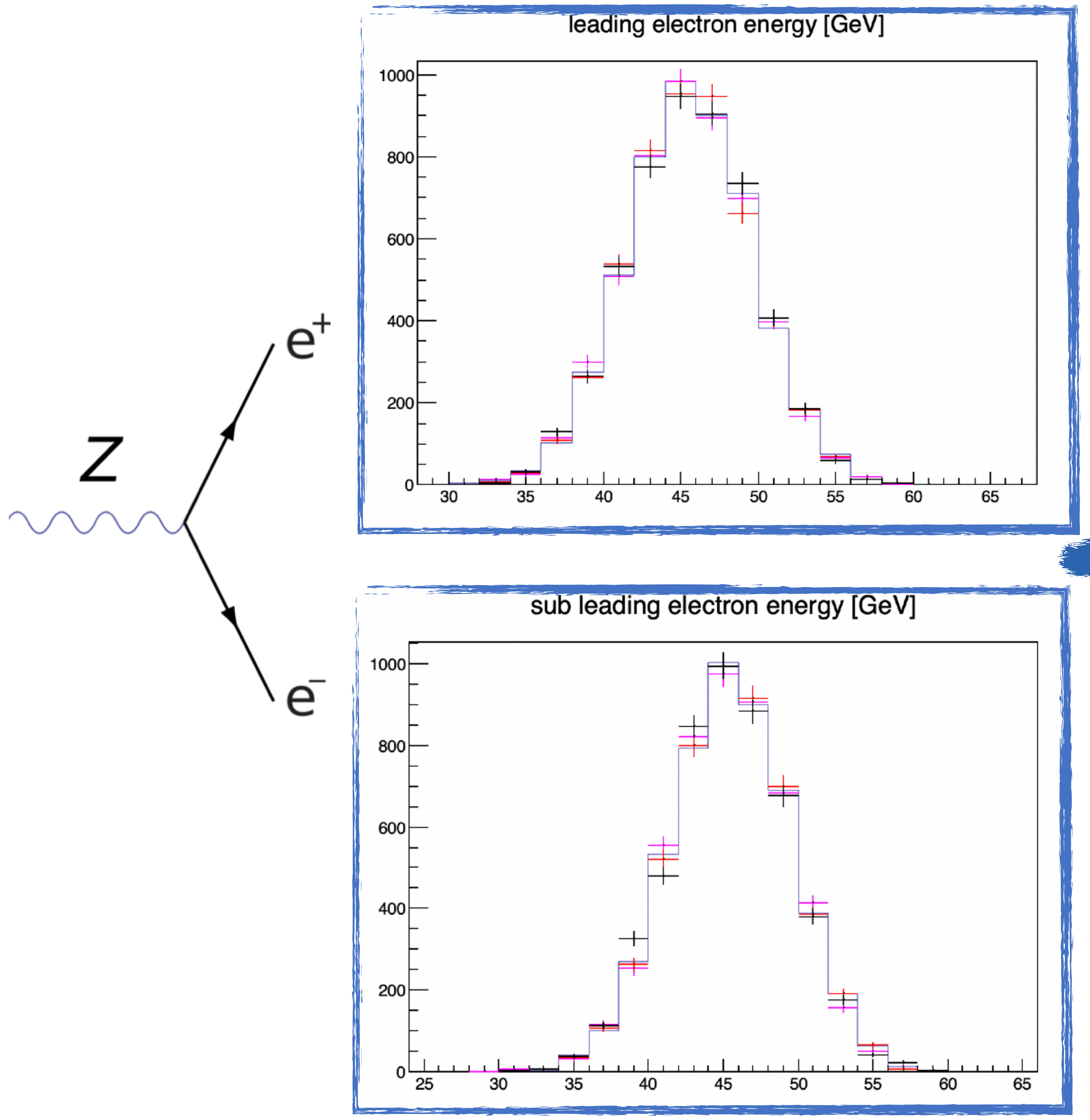
Back-up

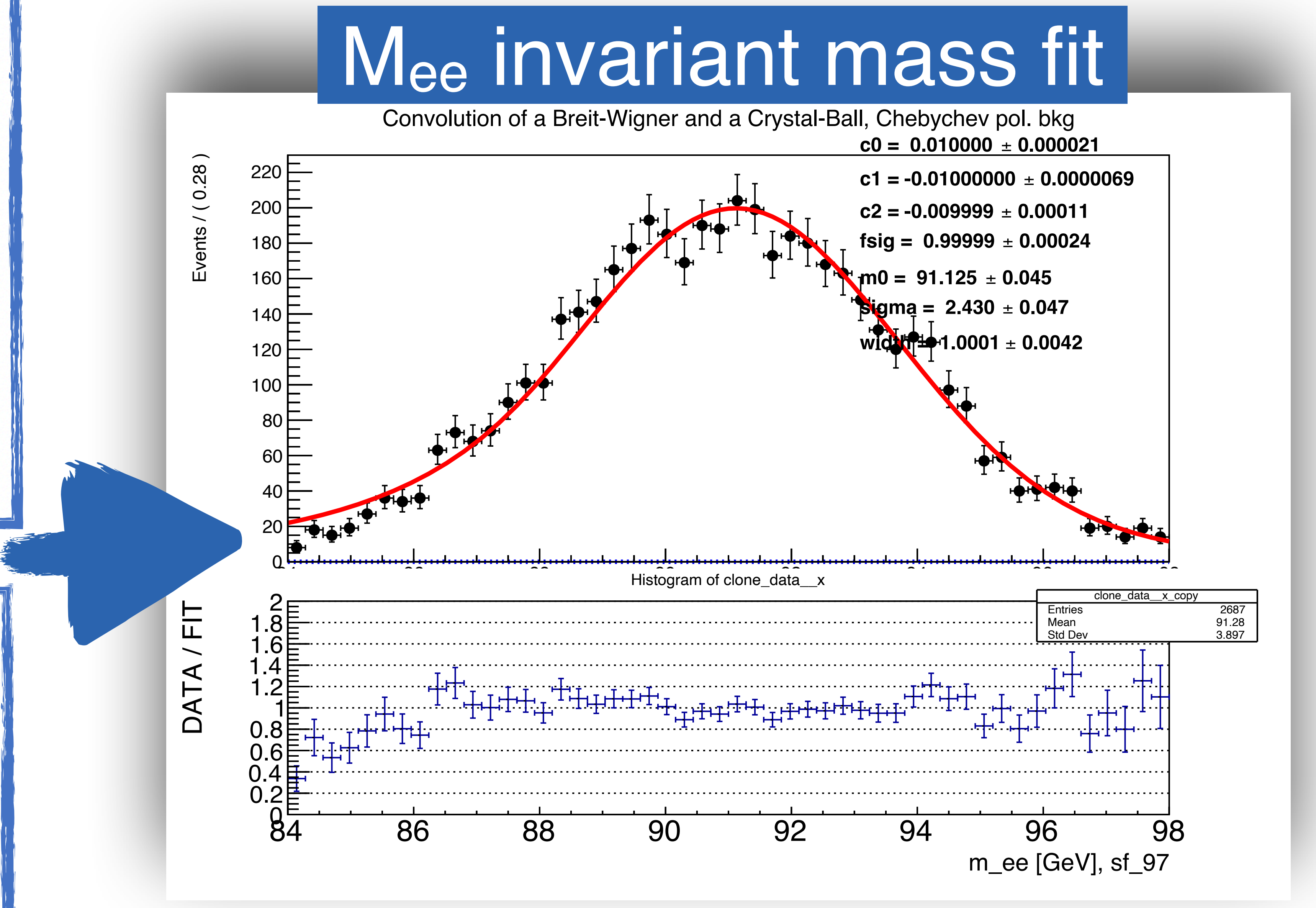
Playground infrastructure at Naples (INFN)

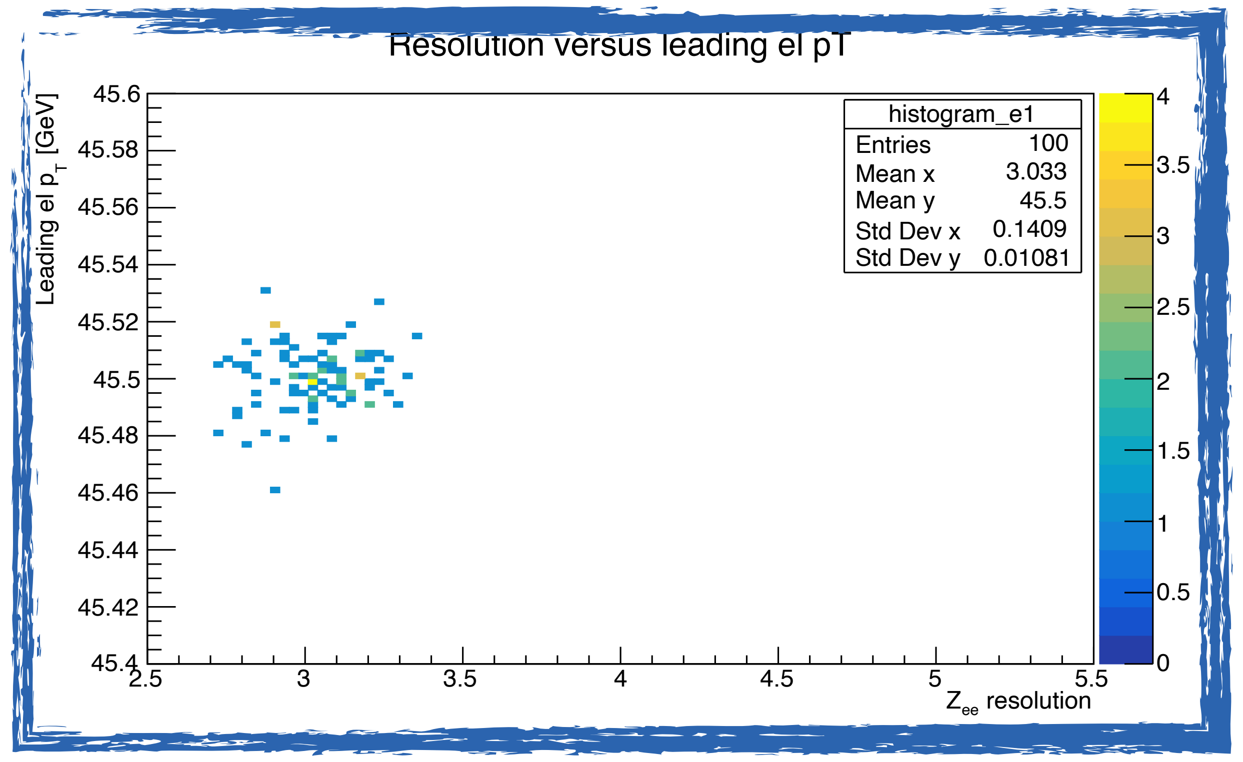
- Our group developed a local testbed infrastructure in INFN Naples (Italy)
- The local deployment is based on the *Open-Stack IaaS* paradigm
- Starting from the already existing *I.Bi.S.CO* installation, several updates were performed
- The cluster is made up of 2 identical virtual machines, each equipped 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

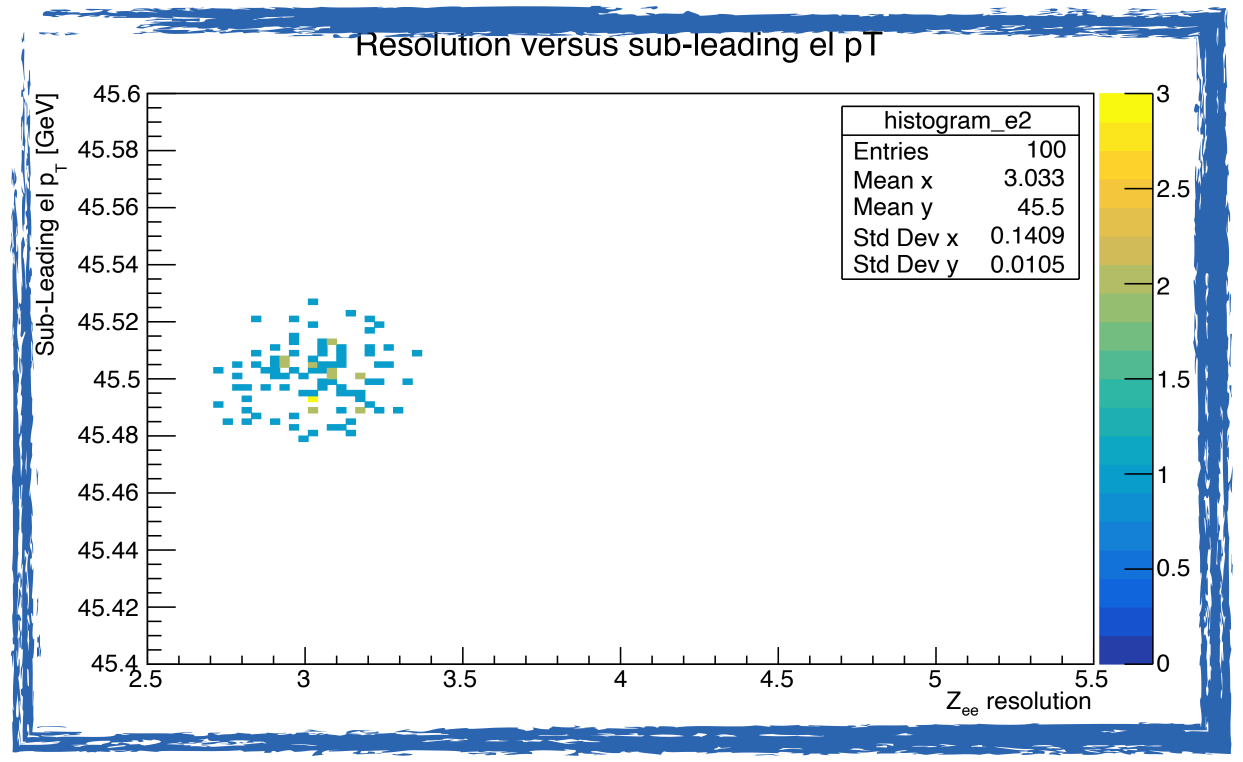
Simple test

- Simulation exploited:
  - 5k events, scaled to 1M events replicating the available dataset
  - Idea: mimic systematic variations, gaussian smearing the electrons energy to compute M_{ee} resolution









Selection and histogramming interactively via RDataFrame on JupyterHub

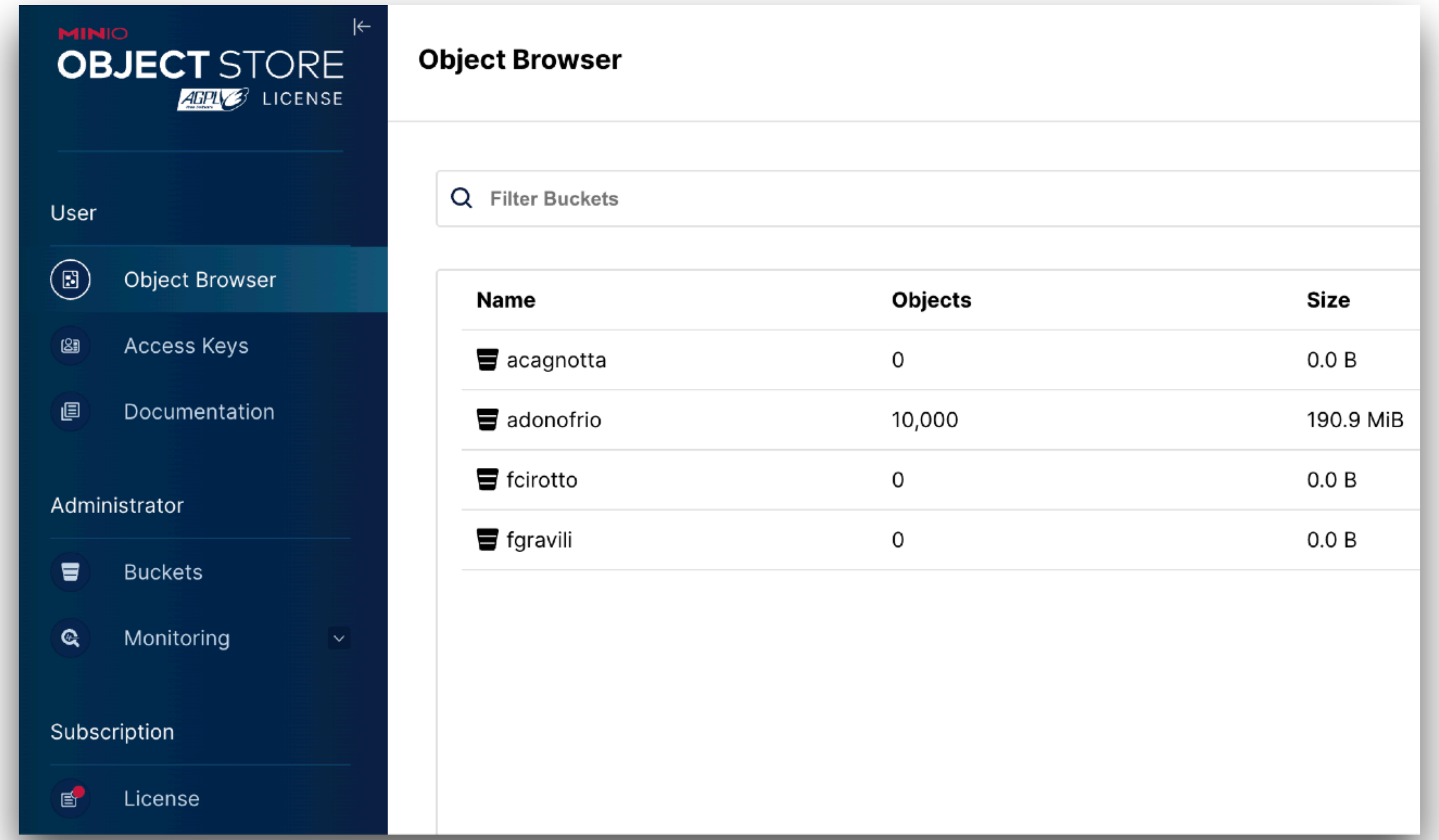
github link to the code

Efficient & user friendly infrastructure

- 2 nodes equipped with **Docker** (20.10) for containerisation and **Kubernetes** (1.26.3) for orchestration

MinIO

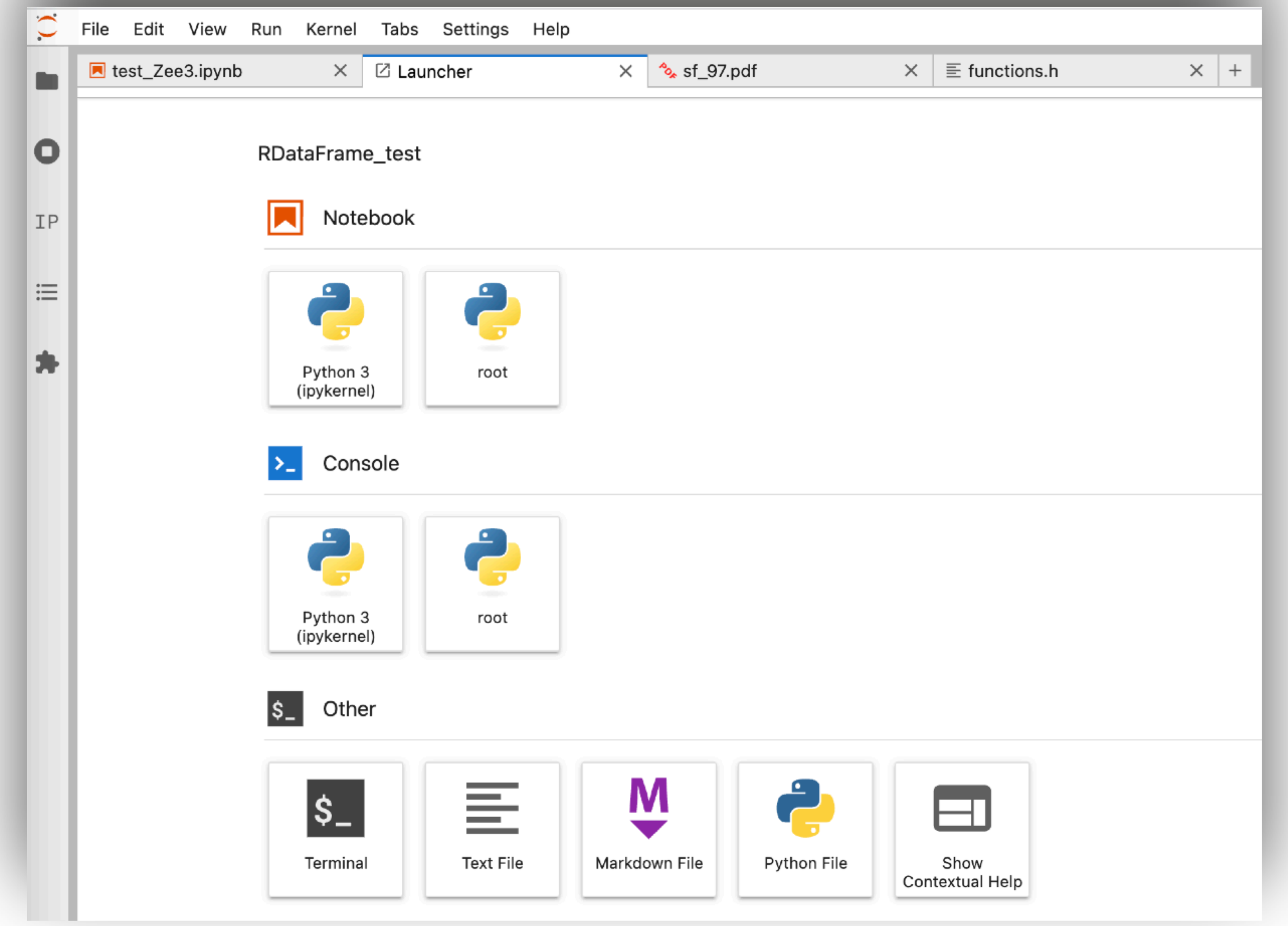
An object storage instance where users can store data



Gianluca’s presentation *link*

Jupyter

The JupyterLAB environment allows users to exploit data science python libraries and to scale them over the cluster

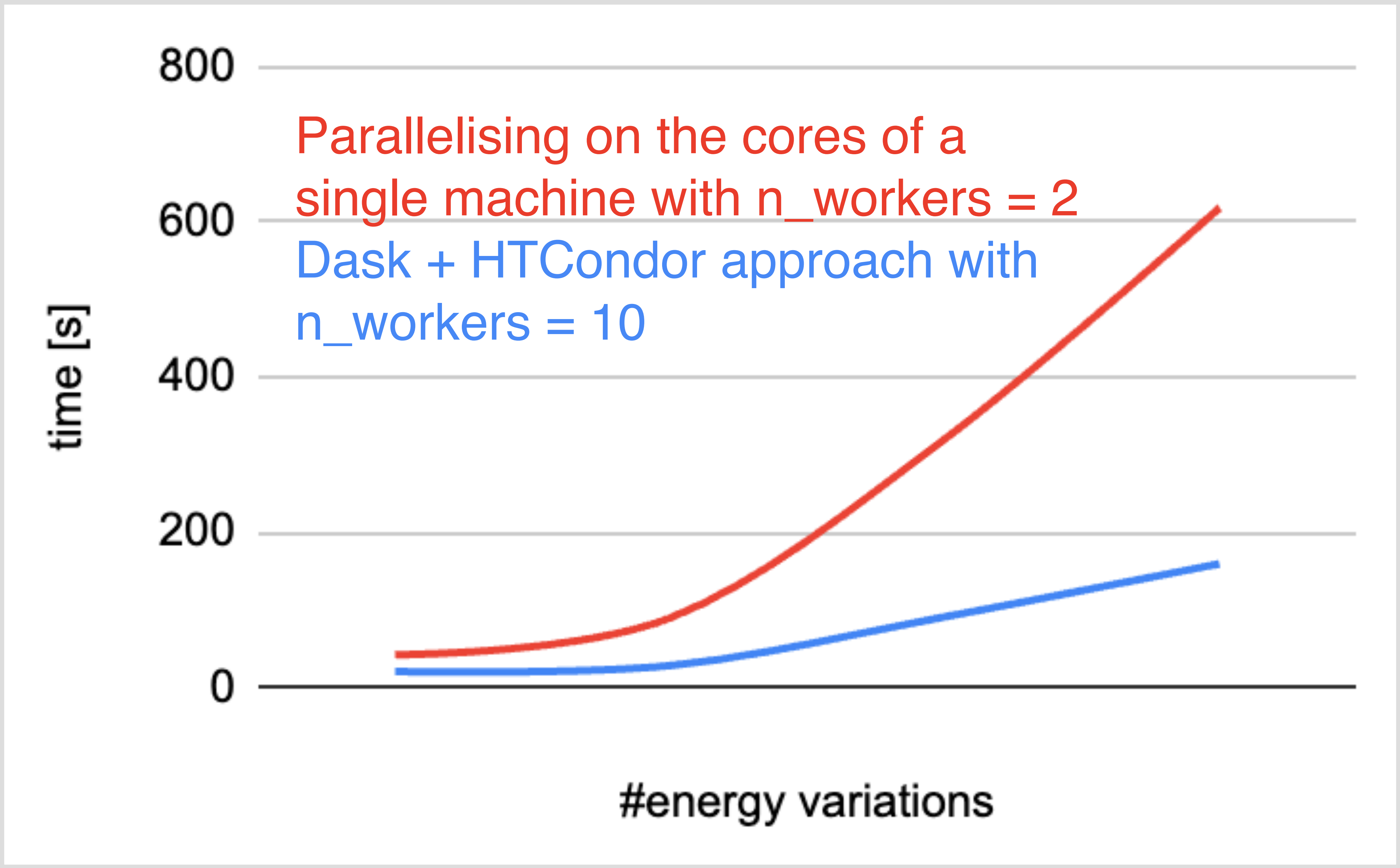


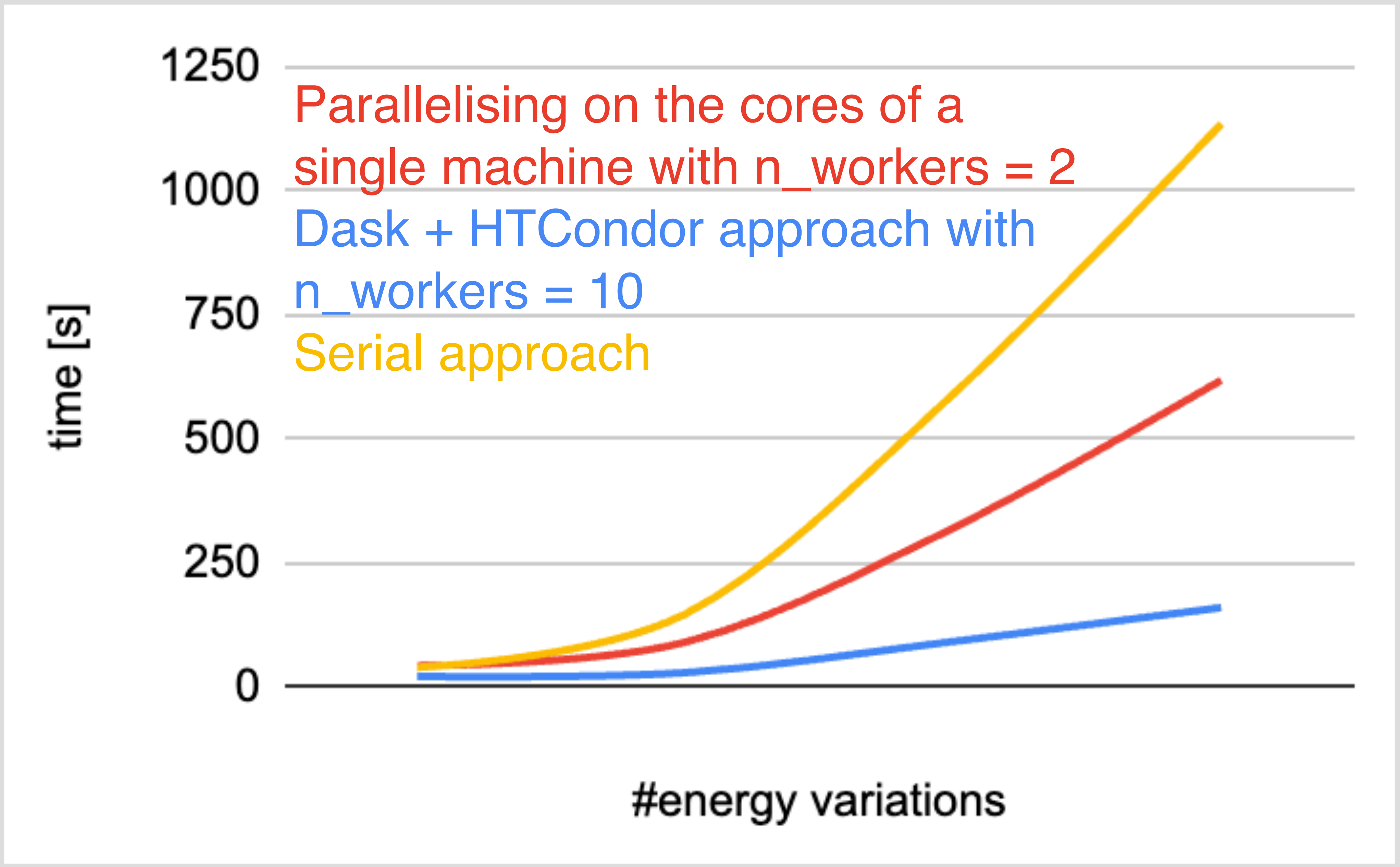
Dask

A python library to scale python code from multi-core local machines to large distributed clusters in the cloud

- Jupyter interface includes:
  - Terminal
  - Notebook implementation
    - Completely exportable and replicable

Towards a Dask + HTCondor model





- Exploiting the distributed approach, the execution time halves wrt the local approach
- Moving to a Dask+HTCondor model, we gain up to another factor 2
  - Increasing the number of workers, the execution time further improves