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

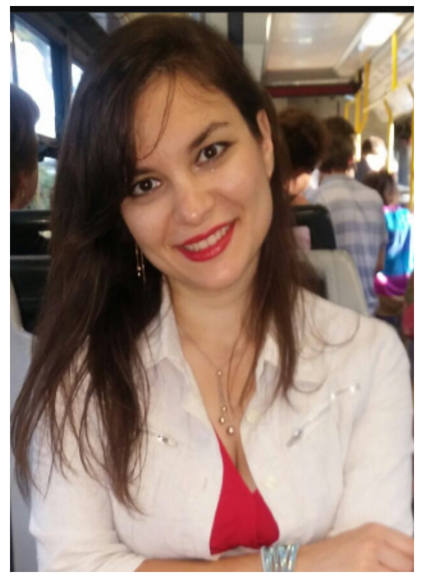
# Spoke 2 Activities Report

Adelina D'Onofrio, INFN sez. Napoli

Spoke 2 INFN Meeting, XXXXXX 2024,
Online

# Introducing myself

## Curriculum Vitae



### Personal information

| | |
|---|---|
| Name / Surname | **D'Onofrio Adelina** |
| Personal Email | donofrioadele@gmail.com - adelina.d'onofrio@cern.ch |
| ORCID | orcid.org/0000-0002-0343-6331 |
| Nationality | Italian |
| Date of birth | 5 June 1988 |
| Gender | Female |

### Awards

| | |
|---|---|
| Dates | **15/07/2020** |
| Prize | Chung-Yao Chao Fellowship 2020, granted by the Center for Excellence in Particle Physics and the Collaborative Innovation Center for Particles and Interactions of the Chinese Academy of Science (CAS) |

### Work experience

| | |
|---|---|
| Dates | **01/07/2023 - today** |
| Occupation or position held | Tecnologo III livello, contratto a Tempo Determinato |
| Name and address of the employer | Istituto Nazionale di Fisica Nucleare (INFN), Sezione di Napoli |
| Main Topic | PNRR - ICSC the National Research Centre for High Performance Computing, Big Data and Quantum Computing, funded by European Union - NextGenerationEU Spoke 2: fundamental research and space economy |

INFN Istituto Nazionale di Fisica Nucleare

## Current activities focussed on ICSC-Spoke 2

- **WP2:** Design and development of tools and algorithms for Experimental HEP
- **WP5:** Support for Data Management on the Distributed CN infrastructure

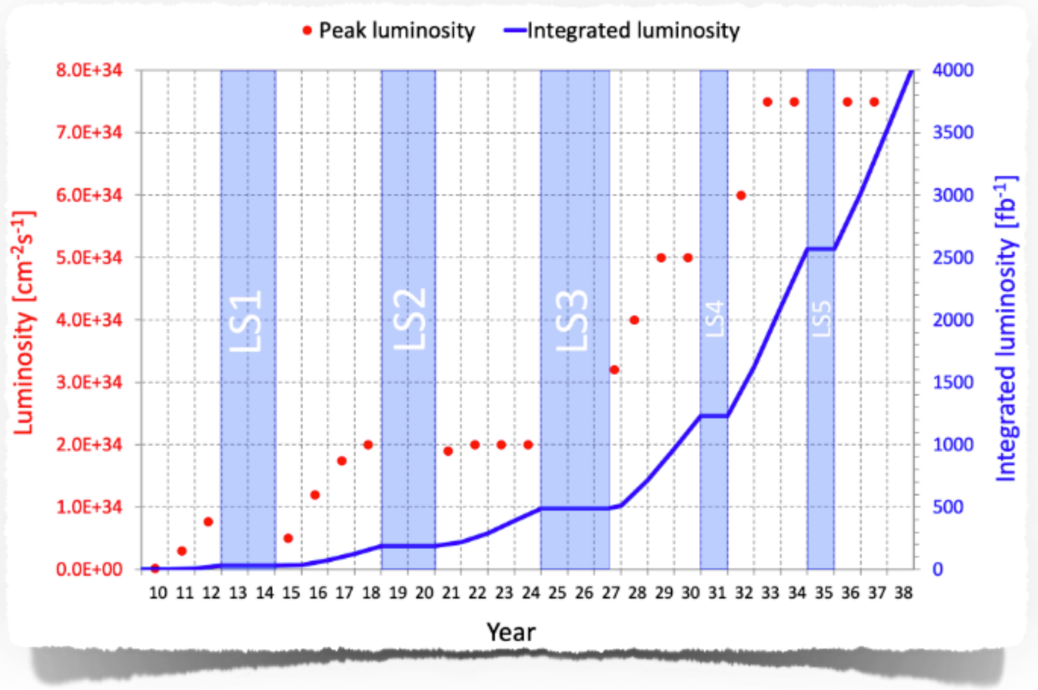
**Target:** benchmarking interactive analyses with the INFN high rate platform

# Outline

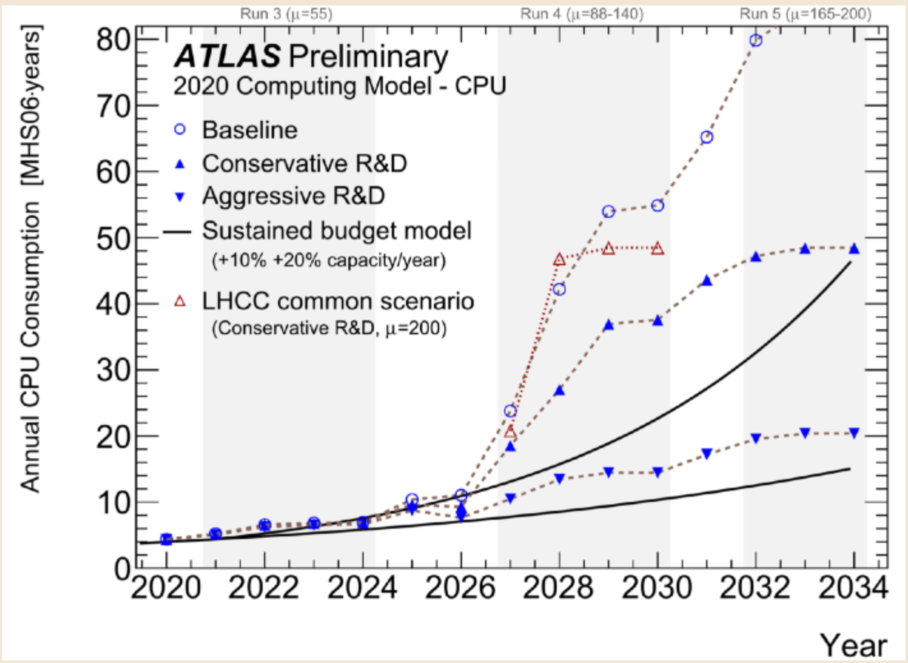
- Motivations
- Test infrastructure
- Use case examples:
  - In a future collider context
  - ATLAS Experiment use cases
- Scalability results
- Miscellanea
- Conclusions

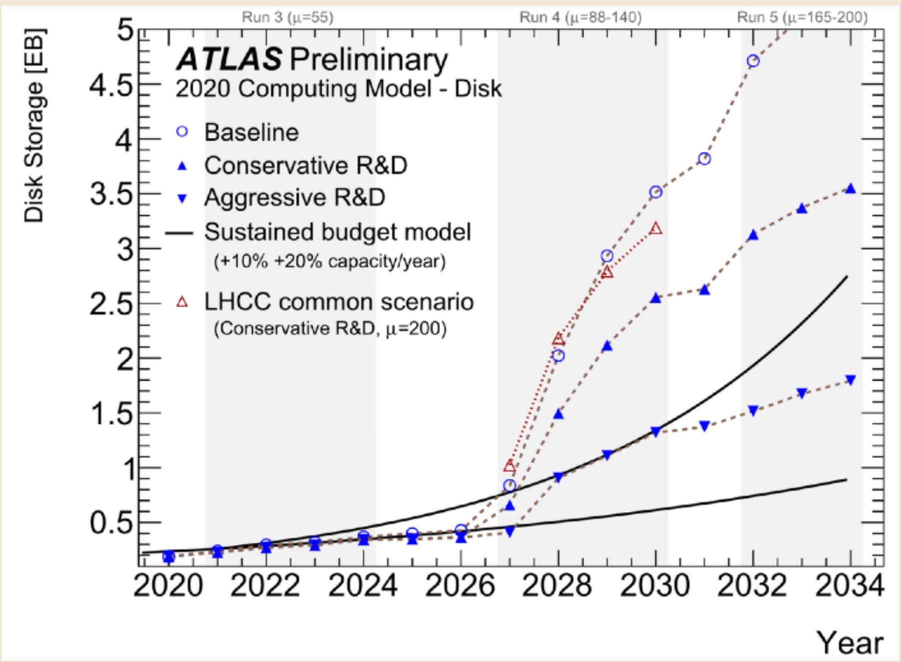
# Motivations

- **Challenges of LHC, HL-LHC and Future Colliders** push to **re-think the HEP computing models**
  - Impact on several aspects, from software to the computing infrastructure




## Similar trends for ATLAS and CMS HL-LHC projections




**Higher rates of collision events** → **Higher demand for computing and storage resources**

To better analyse this increasing amount of Big Data:

- Optimize the usage of CPU and storage;
- Promote the usage of better data formats;
- **Develop new analysis paradigms**!

- New software based on declarative programming and interactive workflows;
- Distributed computing on geographically separated resources

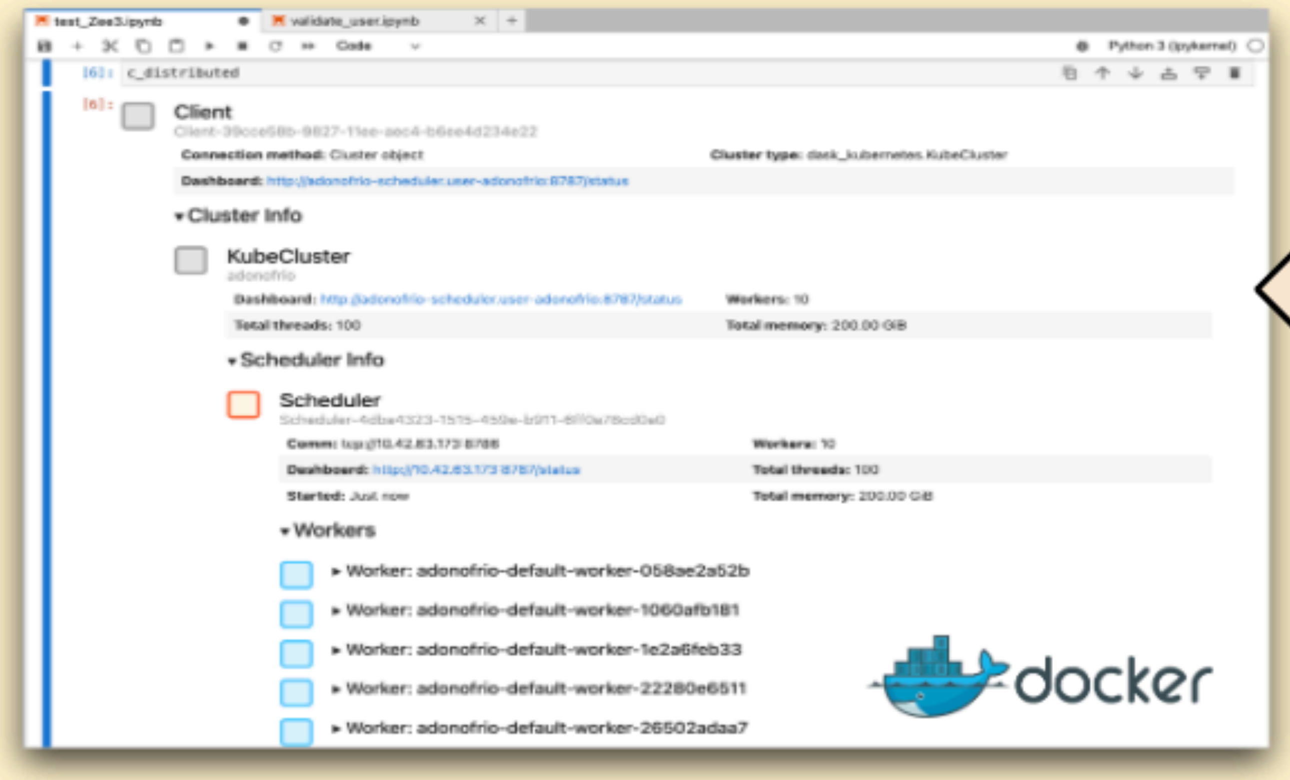
# High throughput data analysis platform

## Access and security

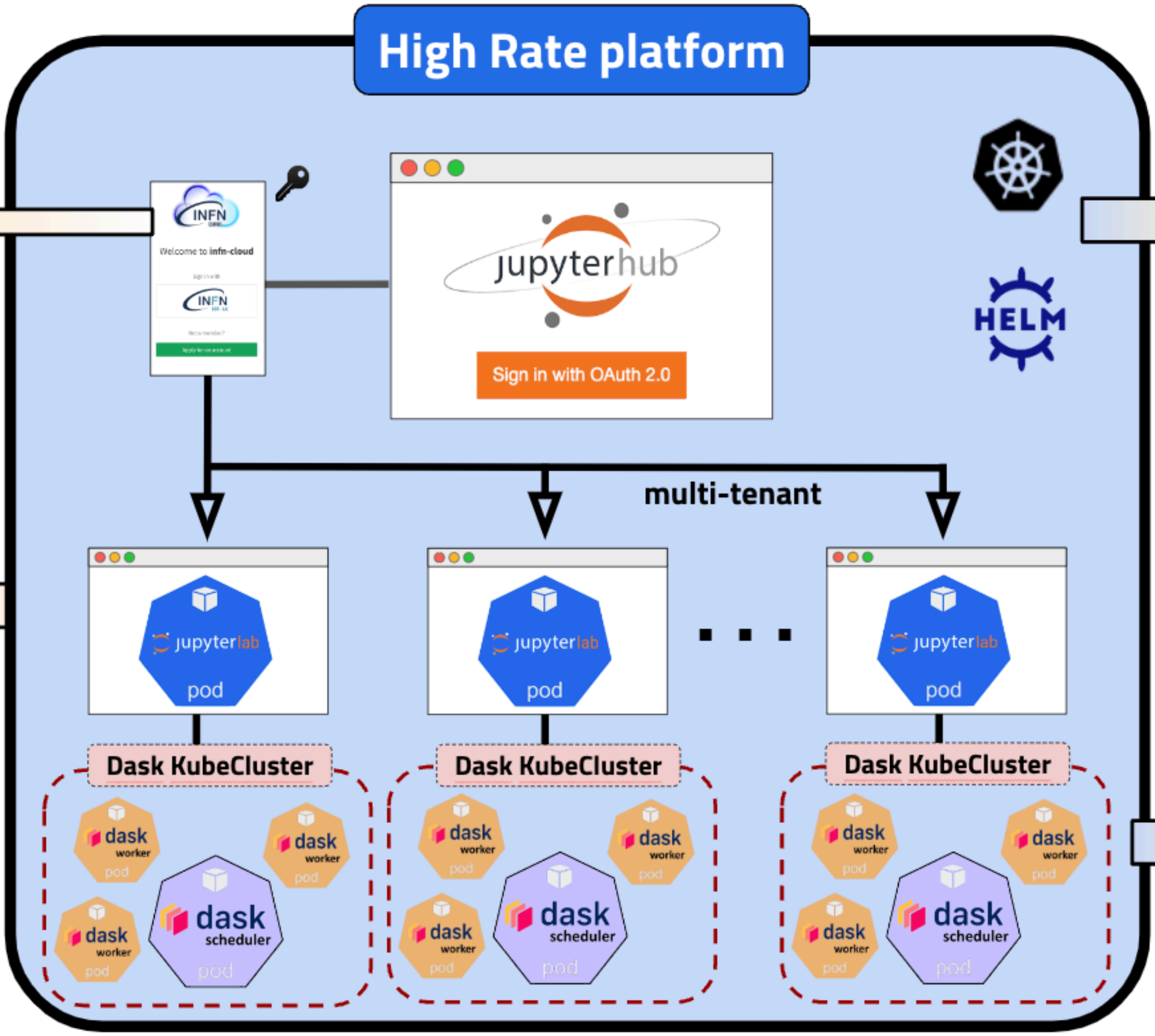
After connecting to an entrypoint URL, the user reaches a **Jupyterhub** instance that, after authentication and authorization via INDIGO-IAM, allocates the required resources for the user's working area

## User Interface

The user interface is based on **Jupyterlab**, customised with specific plugins for specific purposes (e.g. Dask).




The working environment is highly customizable, using tailored Docker containers. This is important when analyses require specific software (collaboration-wise)




## Deployment

The deployment of the **Kubernetes** resources needed for the spawning of this platform, is handled via **HELM charts** available in the GitHub organization.

*Check the docs!*

This allows a seamless, flexible, scalable and fault-tolerant deployment on the available resources, with a limited impact on the admin's work time

## Software

From the software perspective, interactive/quasi interactive analysis is a promising paradigm

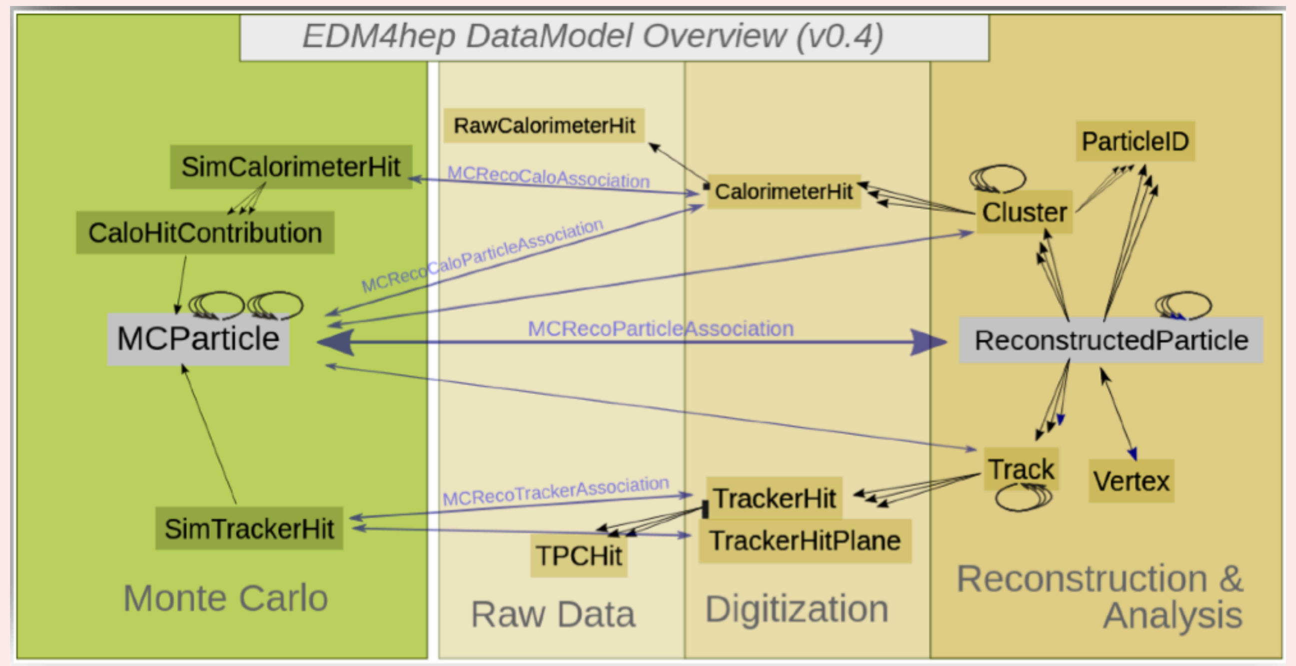
- User-friendly environment
- Adopting open-source industry standards: *Dask, Jupyter Notebooks* and *HTCondor*
- Validating new frameworks (e.g. *ROOT RDataFrame* with multi-threading)

- The feasibility studies shown in the following slides were initially tested on the INFN Napoli facility (details in back-up) and then migrated to the high throuput platform

# Benchmark interactive analyses
# Use-cases

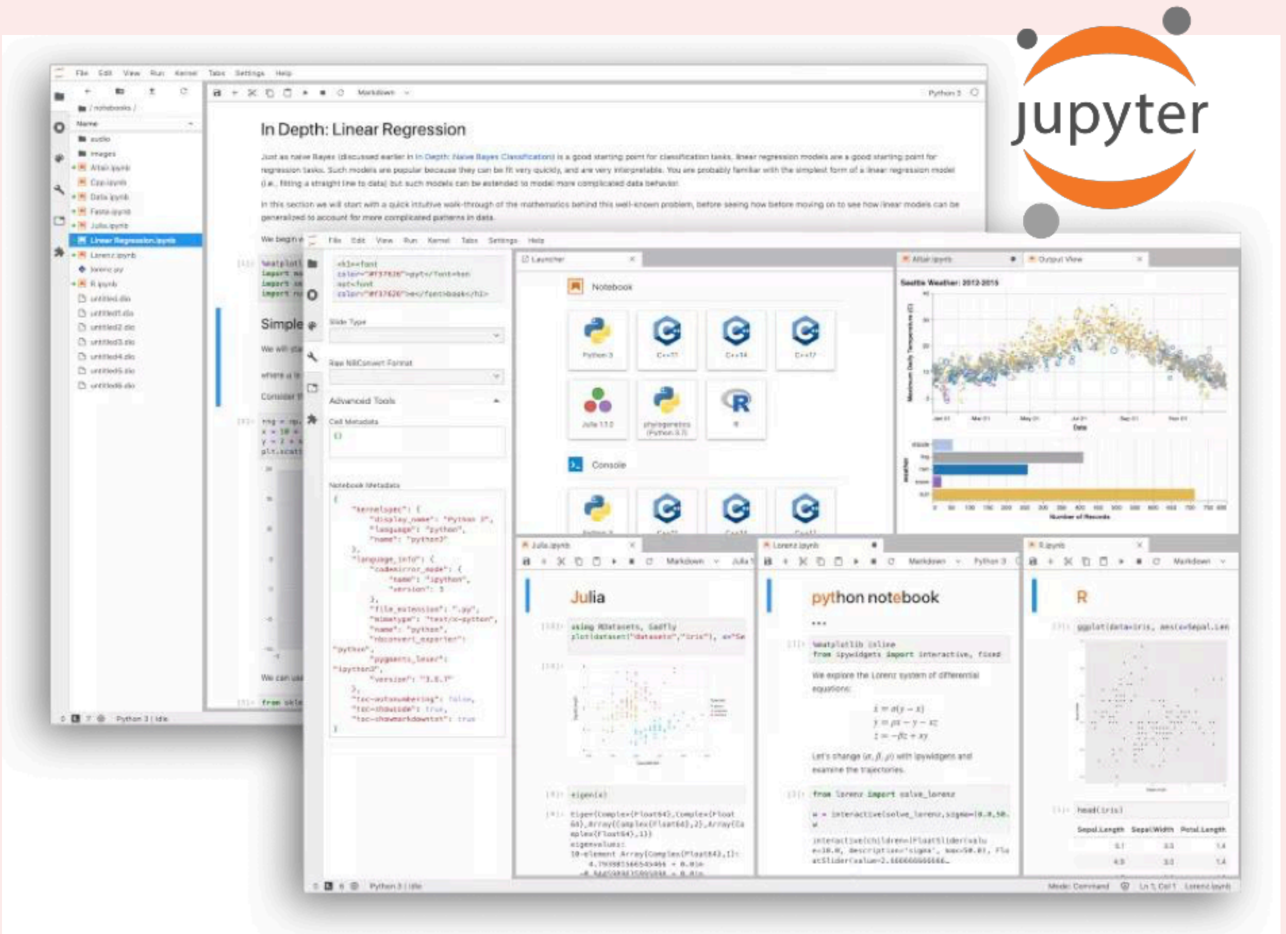
# FCCee use-case

## EDM4hep input data format



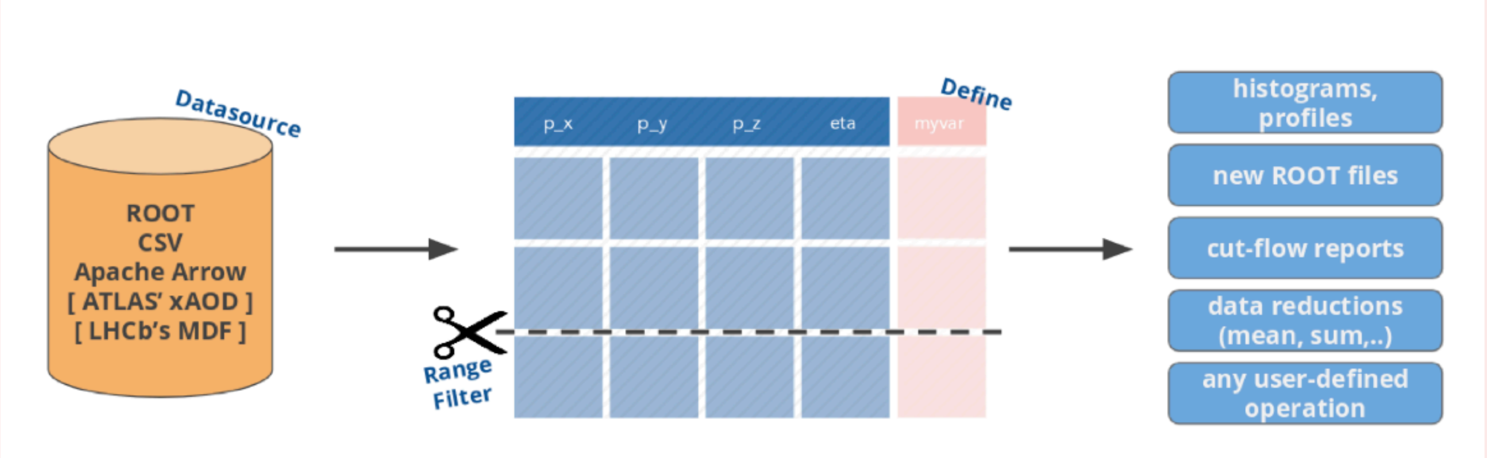

flat input ntuples

## New approach to data analysis



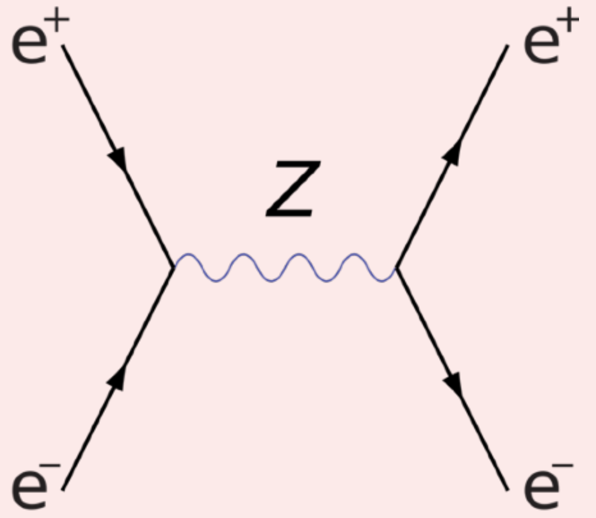

## Selection and histogramming interactively via RDataFrame on JupyterHub

RDataFrame

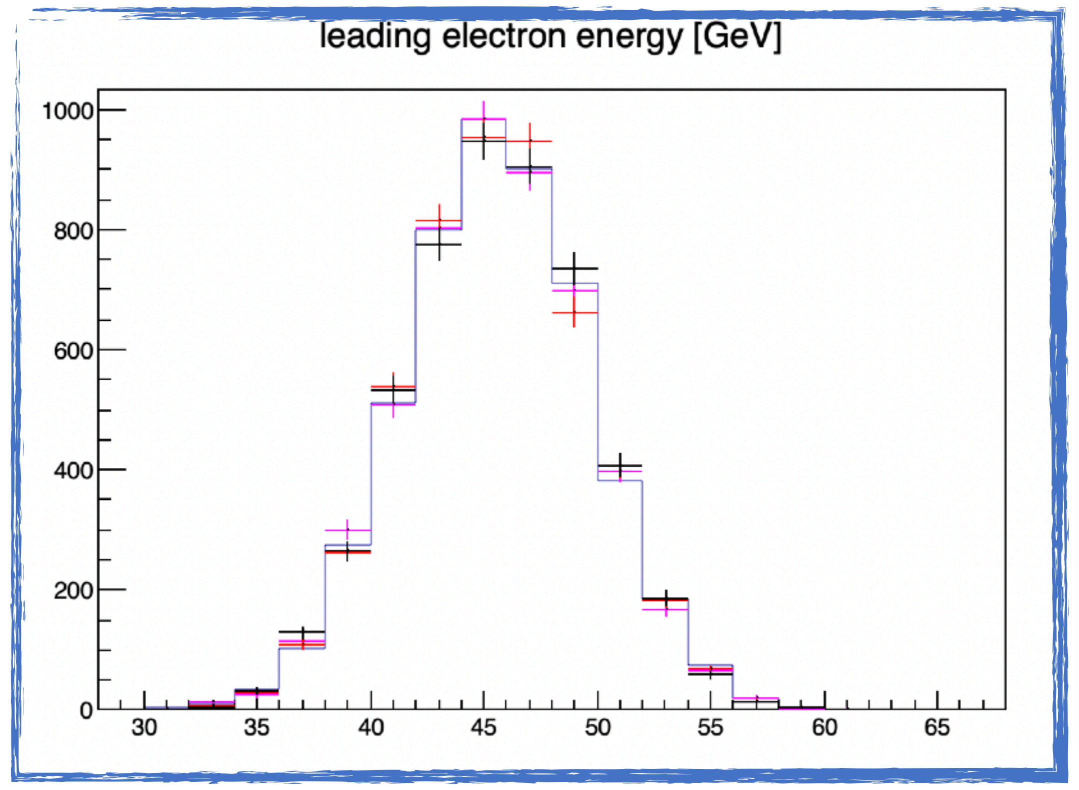



+ dask used as backend

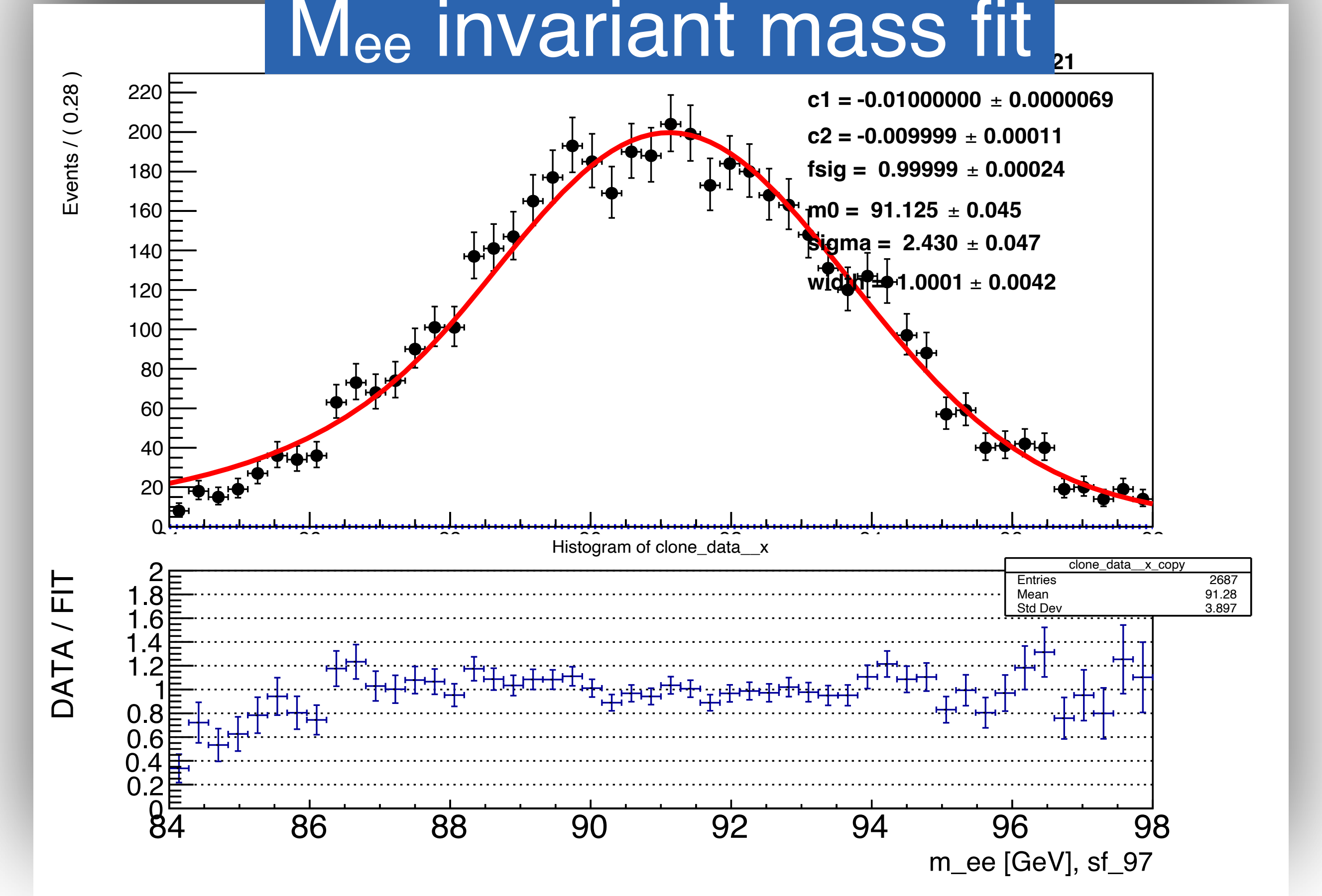
## Feasibility study




:




$M_{ee}$ invariant mass fit




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

ECFA presentation link

github link to the code

# Preliminary results: local client

Scaling without changing your code

```
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()
```

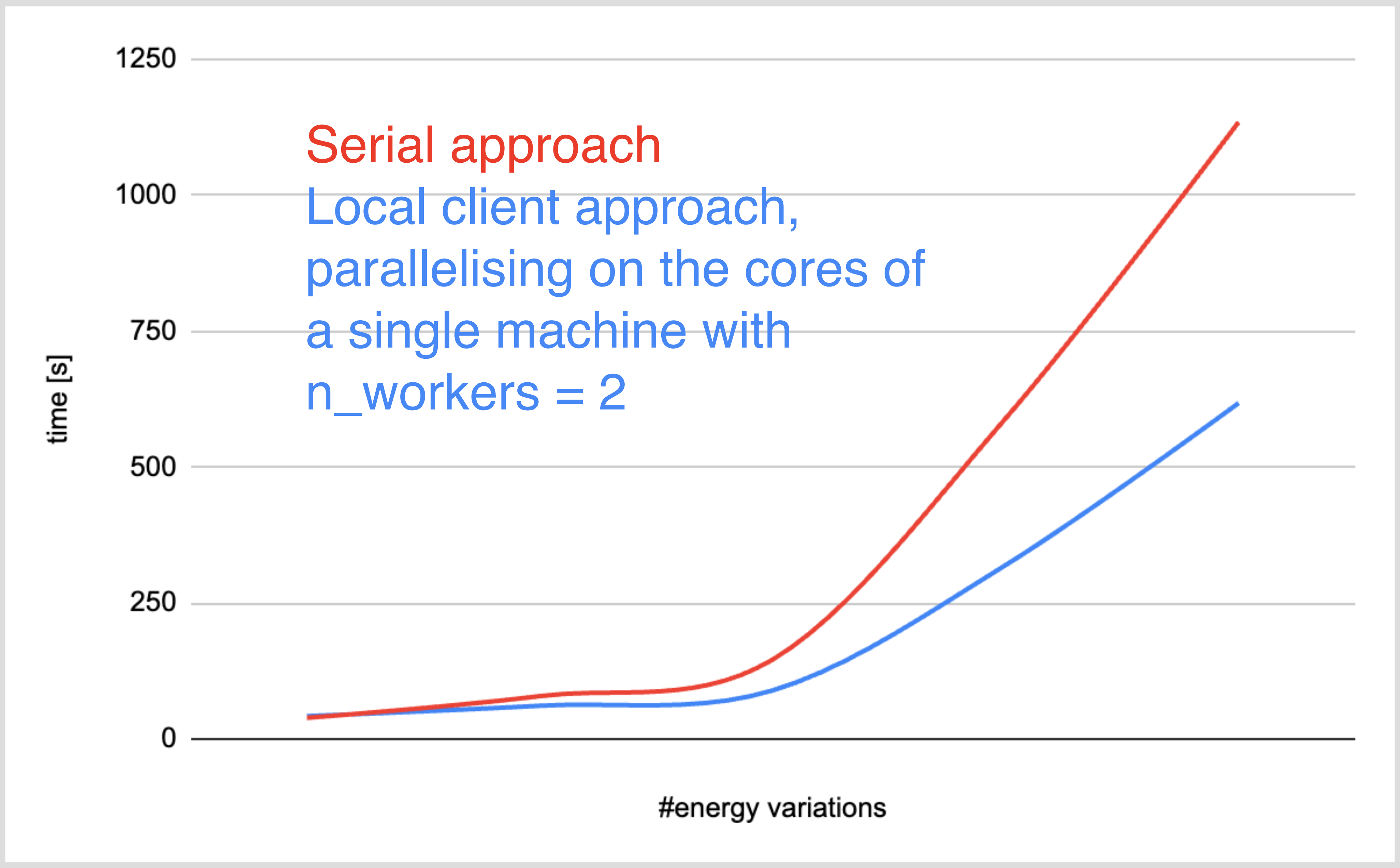
} **Parallel** (the `if distributed == True:` branch)

} **Serial** (the `else:` branch)

⋮

**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")
```



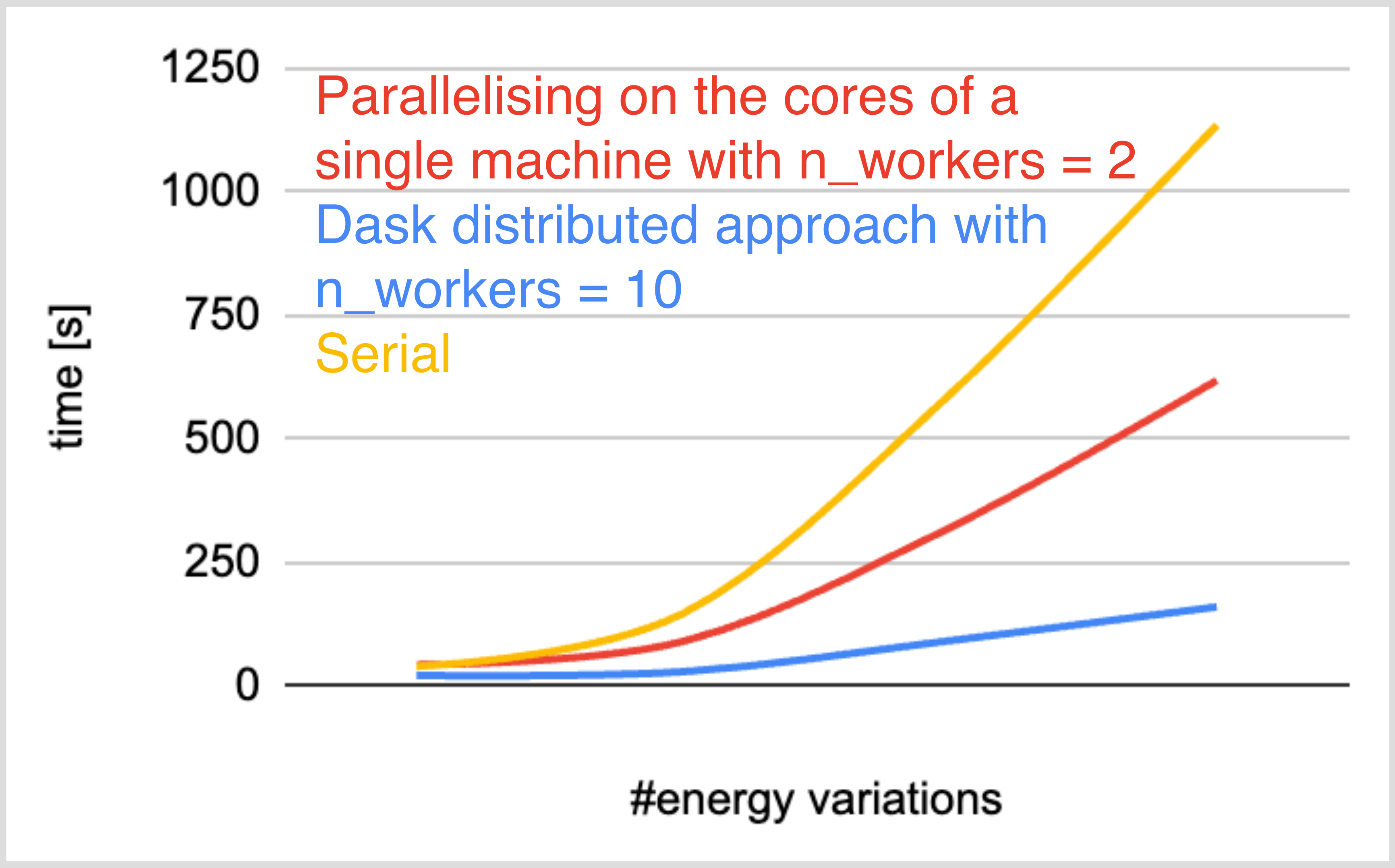

## How to compare the performance?

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

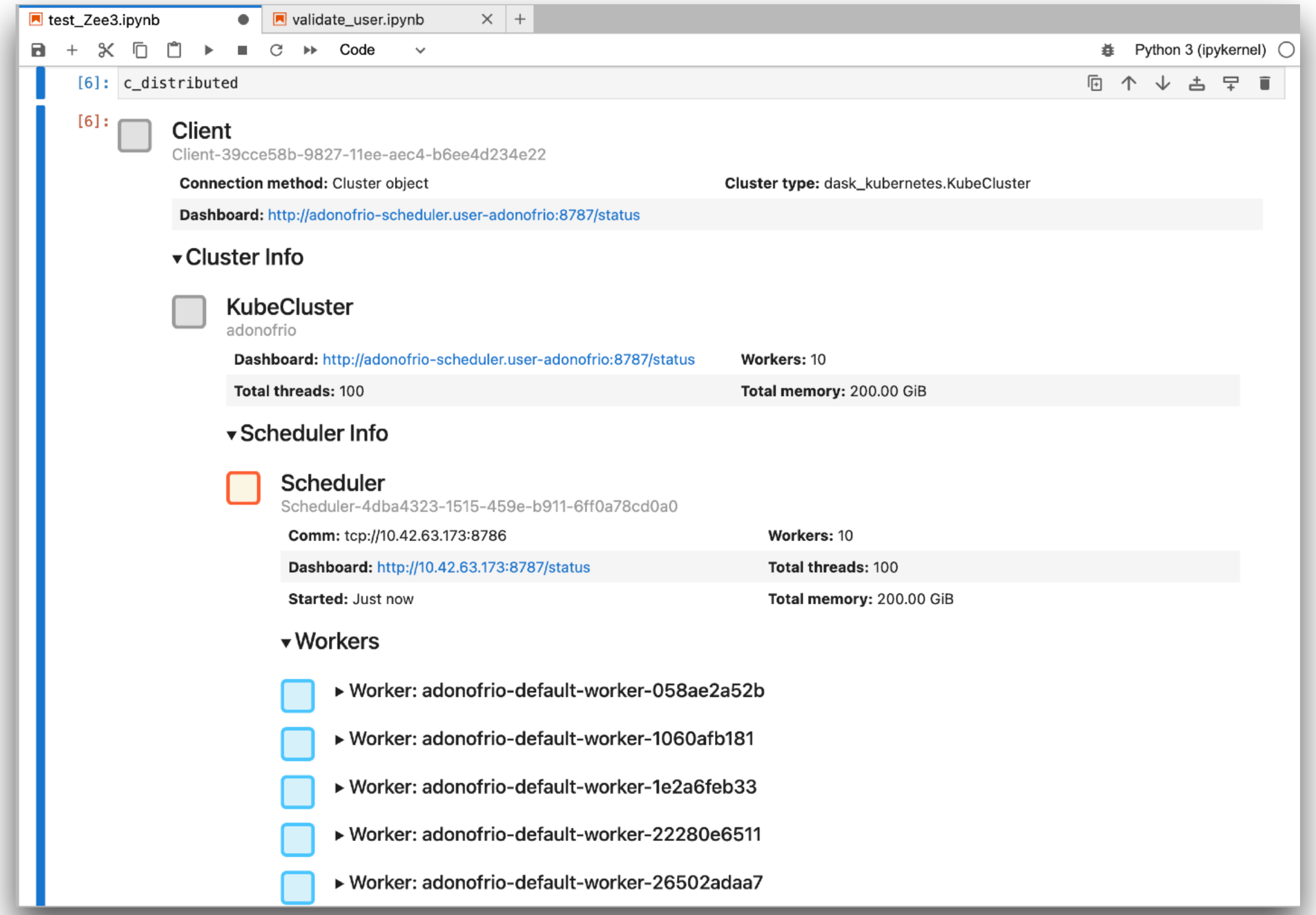
- Exploiting the local client approach, the execution time improves *wrt* the standard/serial approach if we iterate over a significative number of energy variations ( > 10)

# Preliminary results: distributed cluster

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



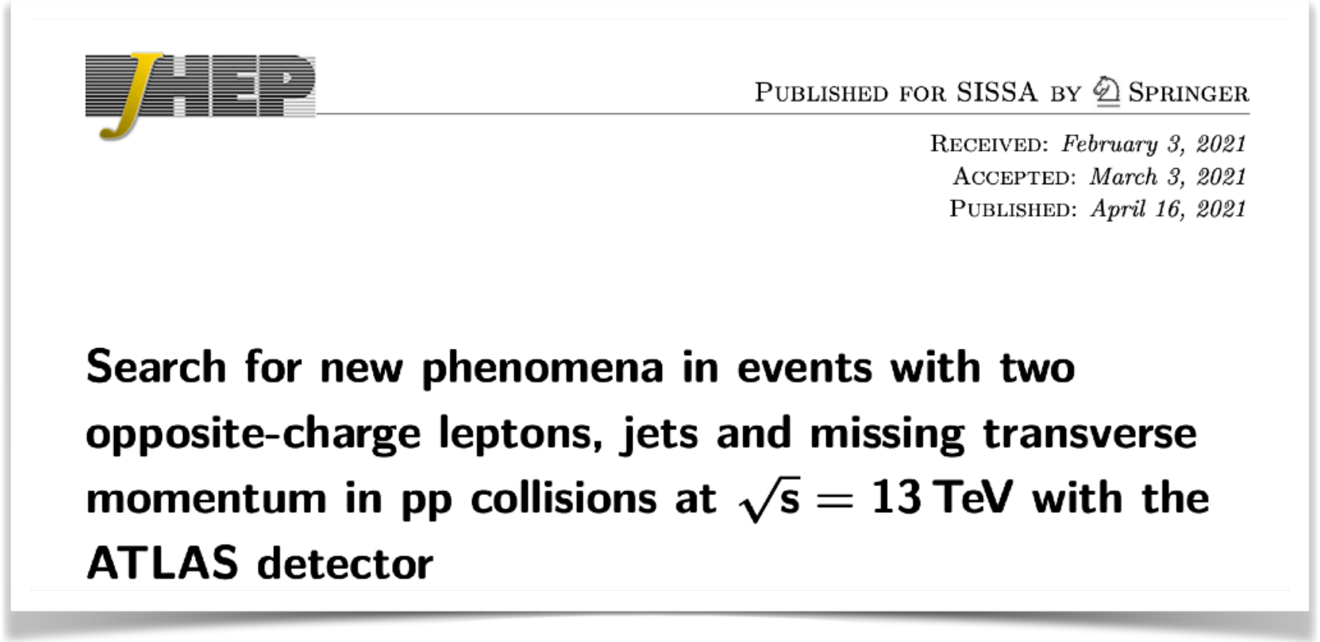

| # iterations | Serial approach | Local client Dask | Distributed Dask |
|---|---|---|---|
| 50 | 590 s | 320 s | 75 s |
| 100 | 1135 s | 618 s | 138 s |



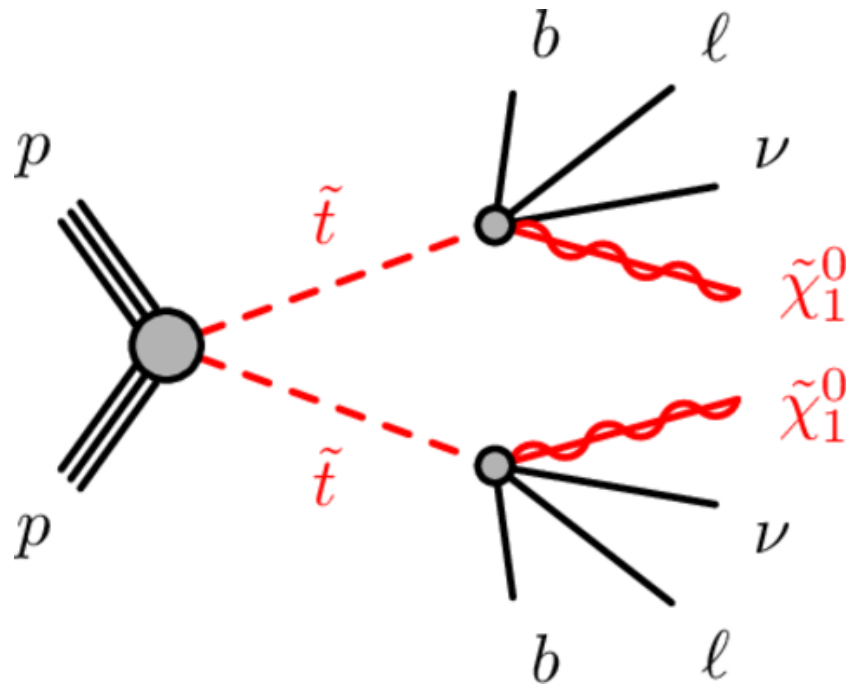

- Moving to a distributed Dask model and **scaling resources, the performance improves**
- Advantage: use this use case as simple test for who wants to benefit from the **WP5** infrastructure

# ATLAS use-case

## SUperSYmmetry: Beyond Standard Model theory


JHEP
Published for SISSA by Springer
Received: February 3, 2021
Accepted: March 3, 2021
Published: April 16, 2021

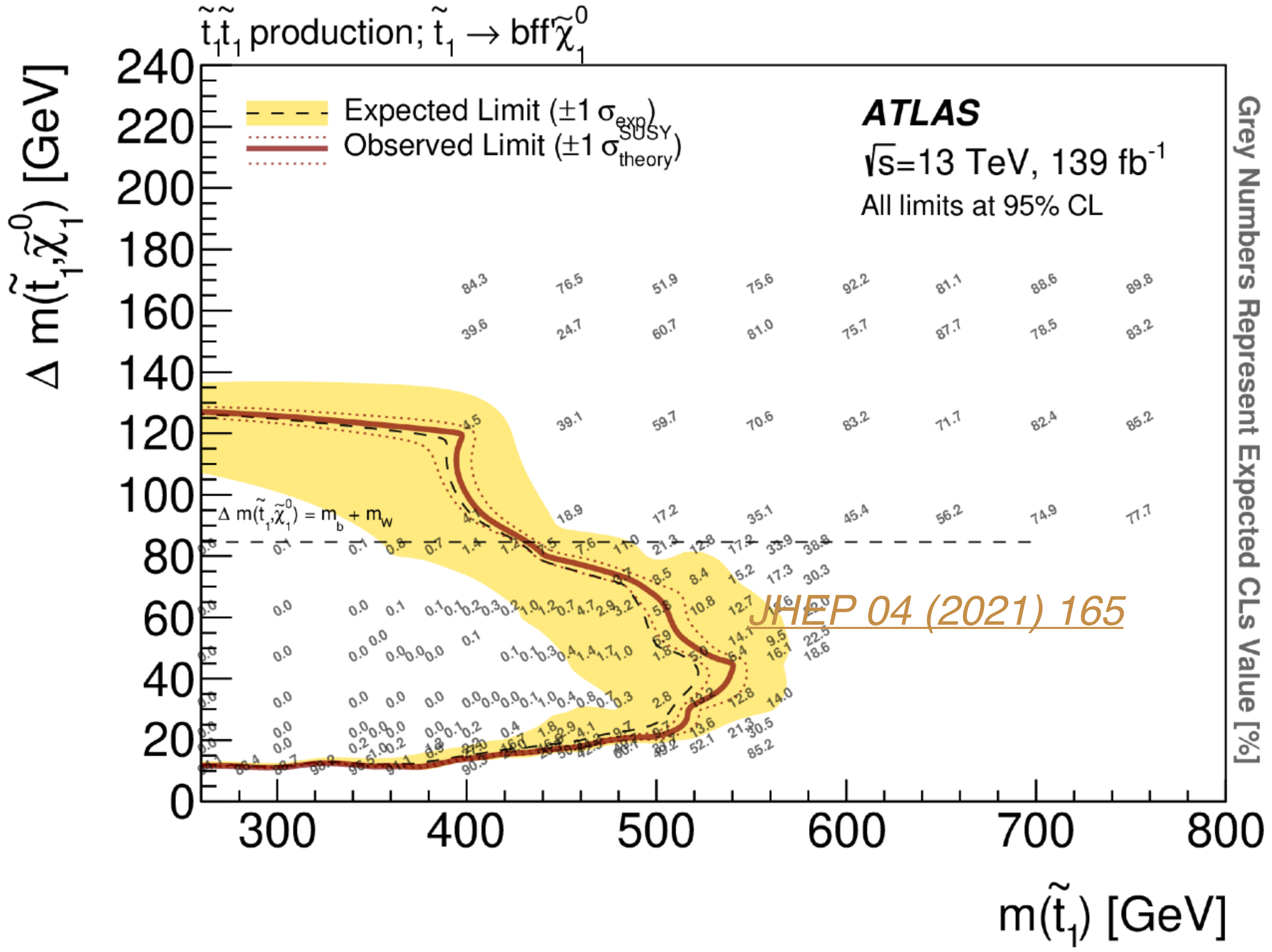
**Search for new phenomena in events with two opposite-charge leptons, jets and missing transverse momentum in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector**




Soft leptons coming from a virtual W* boson decay

Compressed mass spectra:
$\Delta m < m_W + m_b$

- Three different analysis in the ***Run 2 paper***, already published, according to mass splitting between *stop* ( $\tilde{t}_1$ ) and *neutralino* ($\tilde{\chi}^0_1$), allowing different decay modes:
  - 2 body → $\Delta m > m_t$
  - 3 body → $m_W + m_b < \Delta m < m_t$
  - 4 body, the one picked up → $\Delta m < m_W + m_b$
- Common final state signature: 2 OS leptons (electrons/muons), jets and missing transverse energy
- Cut & Count based approach




JHEP 04 (2021) 165

# 4-body search workflow

*Analysis Facility*

**Skimming**
- Provided by the Collaboration
- Offline reconstruction
- $\mathcal{O}$(PB) for data and MC

DAOD reduction

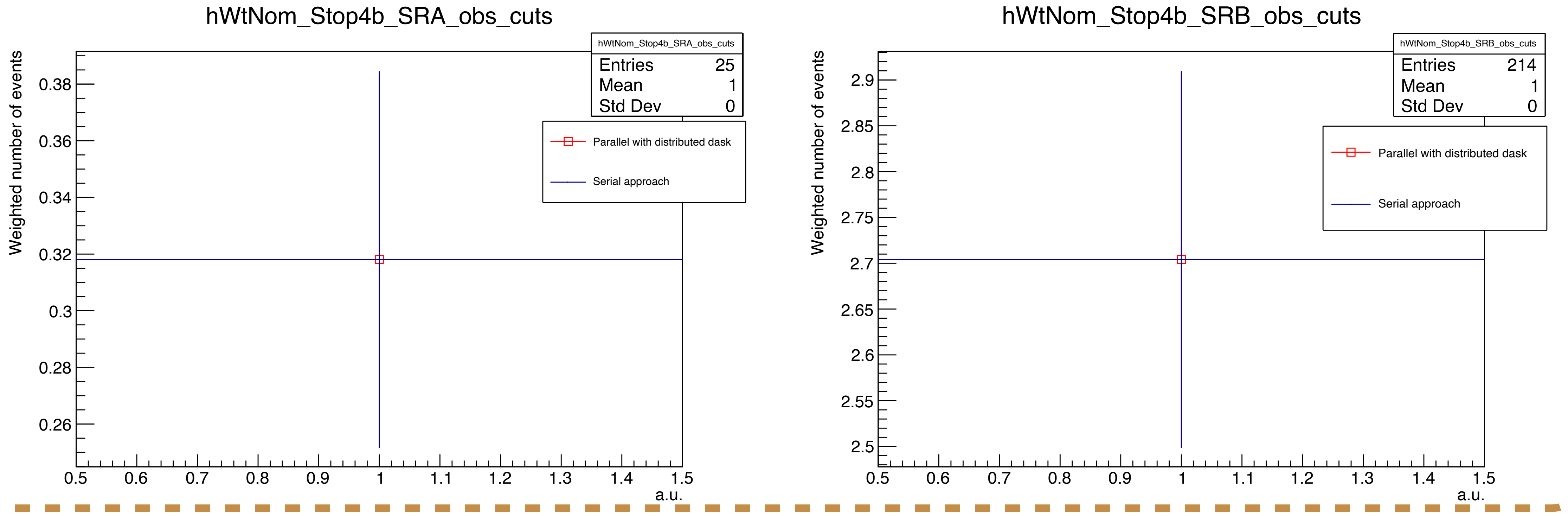
**Thinning**
- Removal of collections
- Baseline objects and trigger
- Scale Factors retrieval
- $\mathcal{O}$(TB) for data and MC

**Slimming**
- Removal of object quantities
- New variable definitions
- Weights application
- $\mathcal{O}(10^2$ GB) for data and MC

**Event Selection**
- Region definitions
- Nominal yields
- Systematic variations
- $\mathcal{O}$(MB), inputs to fitting tool(s)

## Sanity check

Weighted number of events in the Wt background sample, after the event selection cuts in signal regions A and B, nominal case




**Slimming**

ATLAS slimming code already in RDataFrame, but entirely written and compiled in C++ —> NO dask distributed approach

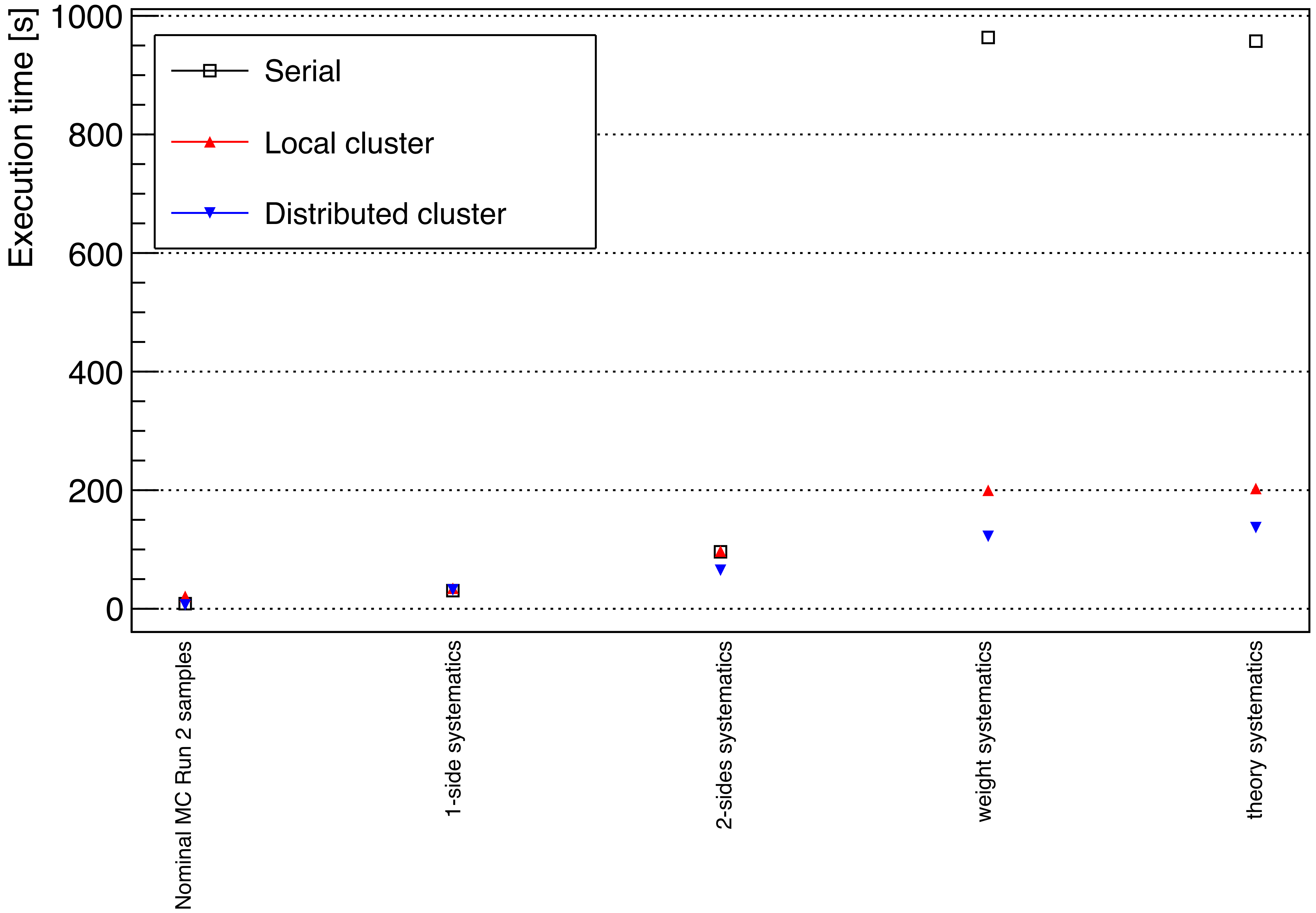
**Event Selection**

- Event selection for fitting tools
- RDataFrame + Dask applied to Wt background sample
  ~ 1.8 GB copied to the INFN workspace
- Tested nominal case and playing with syst. variations
- Code ready to play with other backgrounds

# Preliminary results

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

- Exploiting the distributed approach, the execution time improves *wrt* the standard/serial approach if we iterate over a significative number of systematic variations (each step in the x-axis includes previous contributions)



# Scheduler and Working Nodes Reports

Distributed approach

## Dask Performance Report

*Select different tabs on the top for additional information*

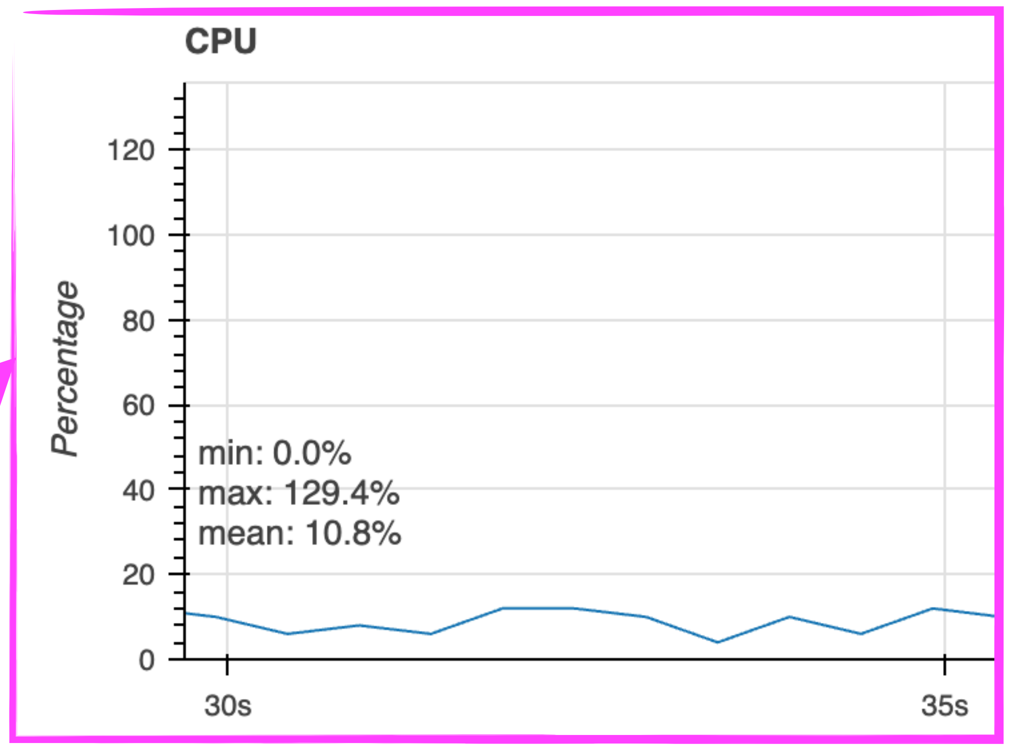
**Duration: 252.87 s**

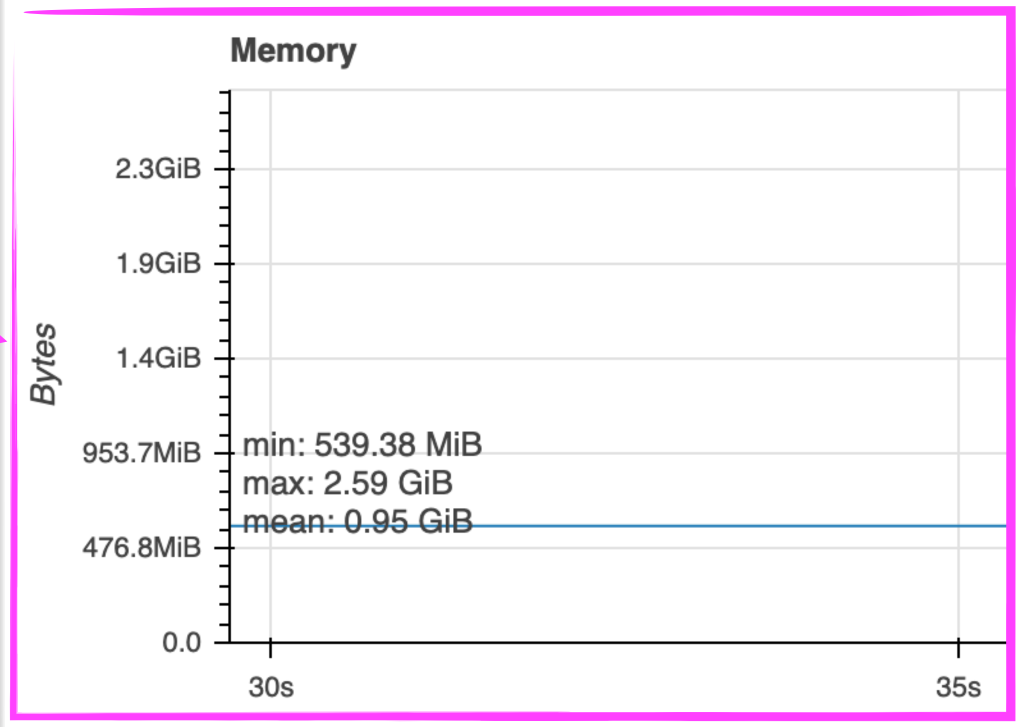
**Tasks Information**

- number of tasks: 621
- compute time: 118.06 s
- deserialize time: 2.39 s

**Scheduler Information**

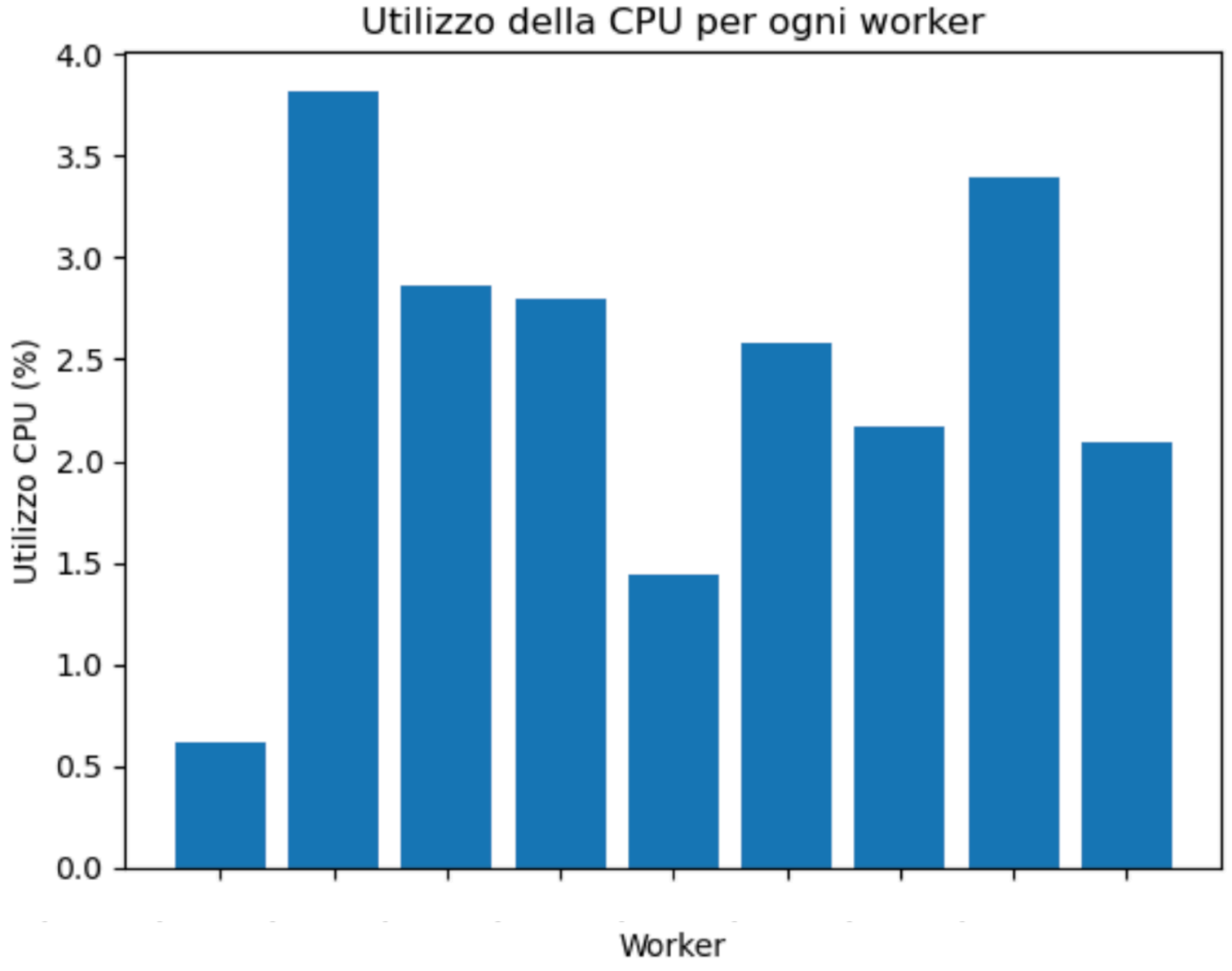
- Address: tcp://127.0.0.1:43821
- Workers: 2
- Threads: 2
- Memory: 4.39 GiB
- Dask Version: 2022.11.0
- Dask.Distributed Version: 2022.11.0




### Connecting to working nodes

- Out of 9 worker nodes, we get up to 4% CPU occupancy on each worker node
- Limited CPU consumption due to the easy cut&count operations

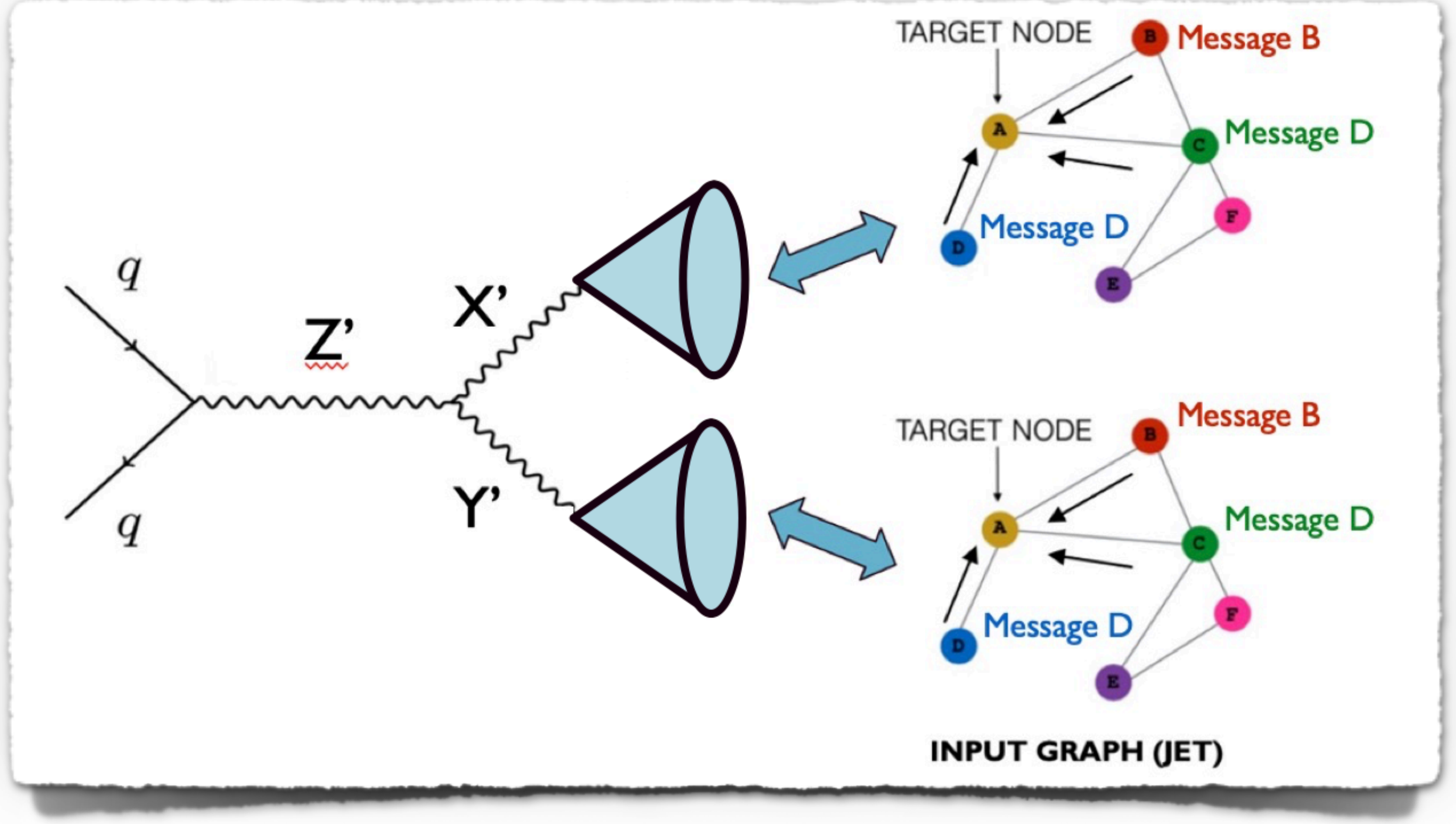


# ATLAS use-case

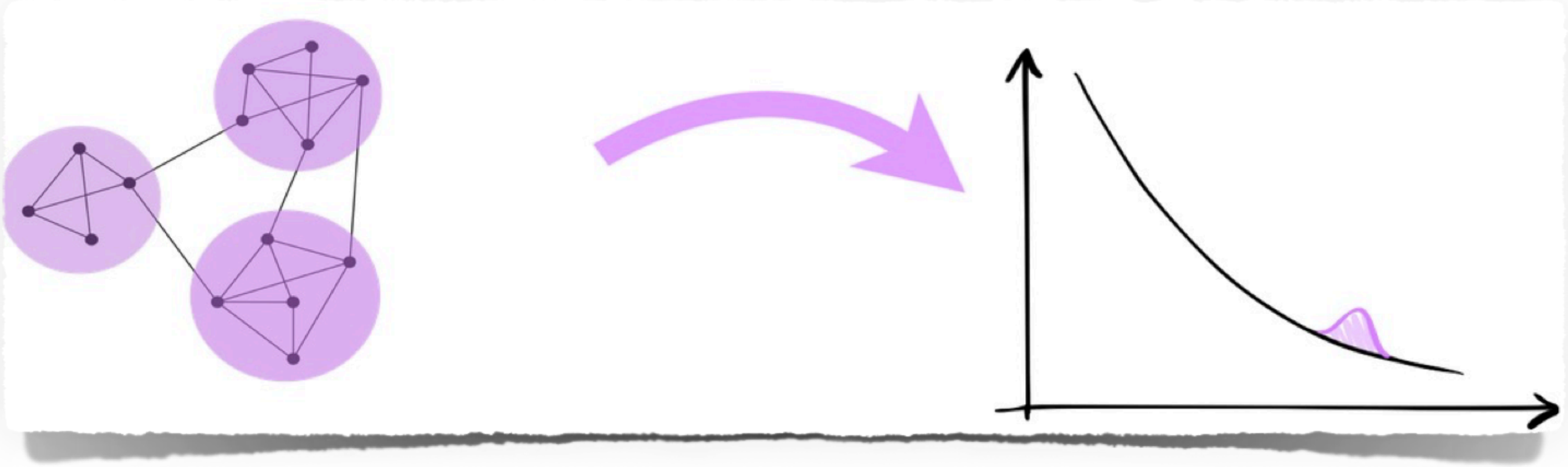
- Anomaly Detection in fully hadronic events with message passing based Graph Neural Netwoks (GNNs)
- Final goal: **LHC Run 3** fully hadronic search
  - Completely model agnostic, 2 large-R jets per event
  - Signal region based on Anomaly Score cut
- Graphs representing the final states jets, with 2 pT leading jets per event, built from transformed constituents
- Analysis performed by the Napoli ATLAS group in collaboration with Rome "La Sapienza" ATLAS group

- My personal contributions:
  - Data pre-processing with parallel approach, crucial to reduce the computational time
  - Performance evaluated on IBISCO cluster: *https://ibisco-ui.na.infn.it/*

## Anomaly detection in di-boson searches with fully hadronic final state



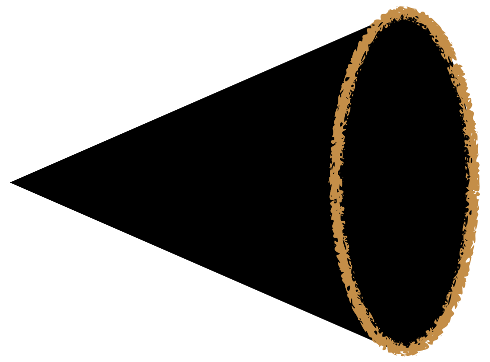

## Analysis strategy

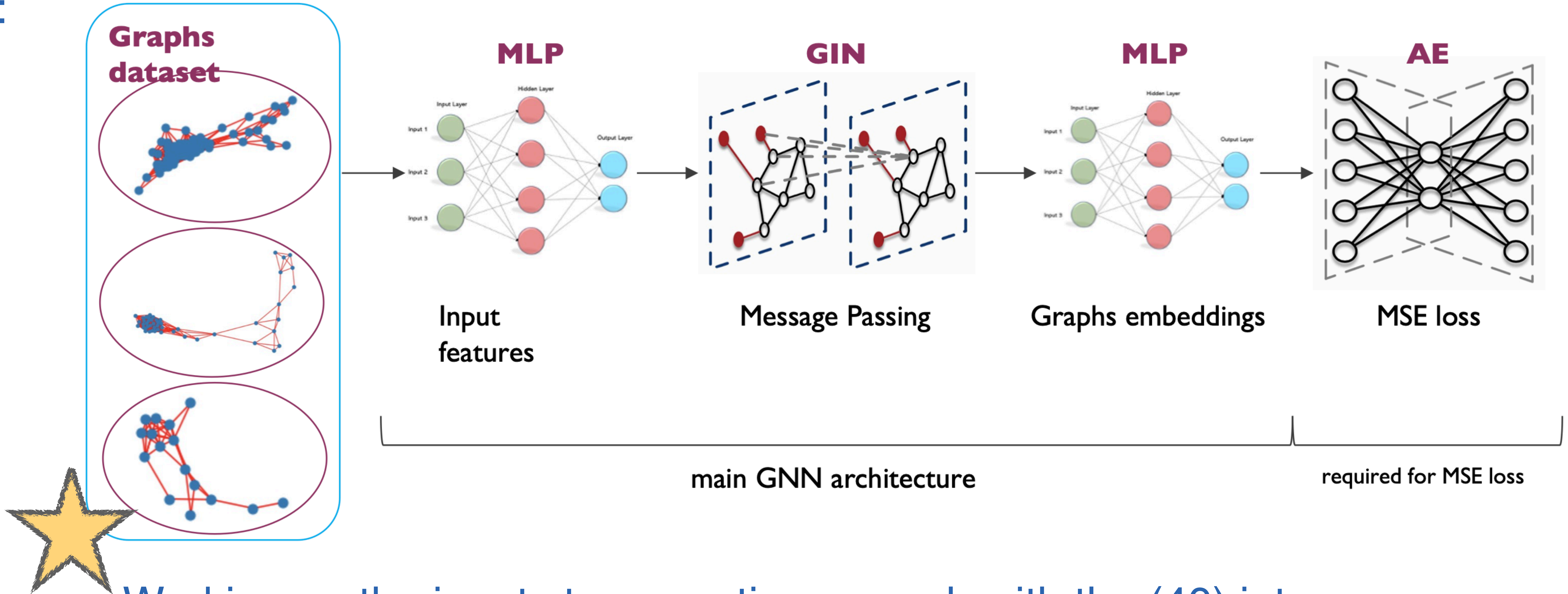


# Analysis workflow

- Inputs:
  - ATLAS Run 3 simulations for signal and background (already skimmed samples ~ hundreds of MB)

Physics objects: calorimetric jets






Working on the input step: creating a graph with the (40) jets constituents: $p_T$, $\eta$, $\phi$ & evaluate isomorphism between graphs

# Create Graph Dataset

- Initial issue: graph creation was time consuming and computationally expensive ~ 20 minutes for a 17k events dataset
- Task: parallelise the graph creation step to reduce the execution time
- Performed on CPUs (max 128 nodes available on IBISCO, both ibisco-gpu02 & ibisco-ui exploited)
  - pandas & RDataFrame used
  - from joblib import Parallel, delayed
  - results = Parallel(n_jobs=self.num_cores, backend="multiprocessing")(delayed(self.createGraph)(chunk)for chunk in chunks)

To do: test the parallel approach on kubecluster and compare performance of IBISCO *vs* virtual machines

**Input: signal sample (~17k events)**

| # nodes | #chunks | execution time |
|---|---|---|
| 60 | 10 | 5.8 minutes |
| 60 | 100 | 2.5 minutes |
| 60 | 1000 | 2  minutes |
| 120 | 1000 | 1 minute |

**Input: background sample (~434k events)**

| # nodes | #chunks | execution time |
|---|---|---|
| 60 | 1000 | 40 minutes |
| 120 | 1000 | 20 minutes |

# Isomorphism between Graphs

- Analogous issue, task, and setup as in the previous slide
- Issue: initial execution time for isomorphism evaluation ~ 10 minutes for a input dataset with 500 entries

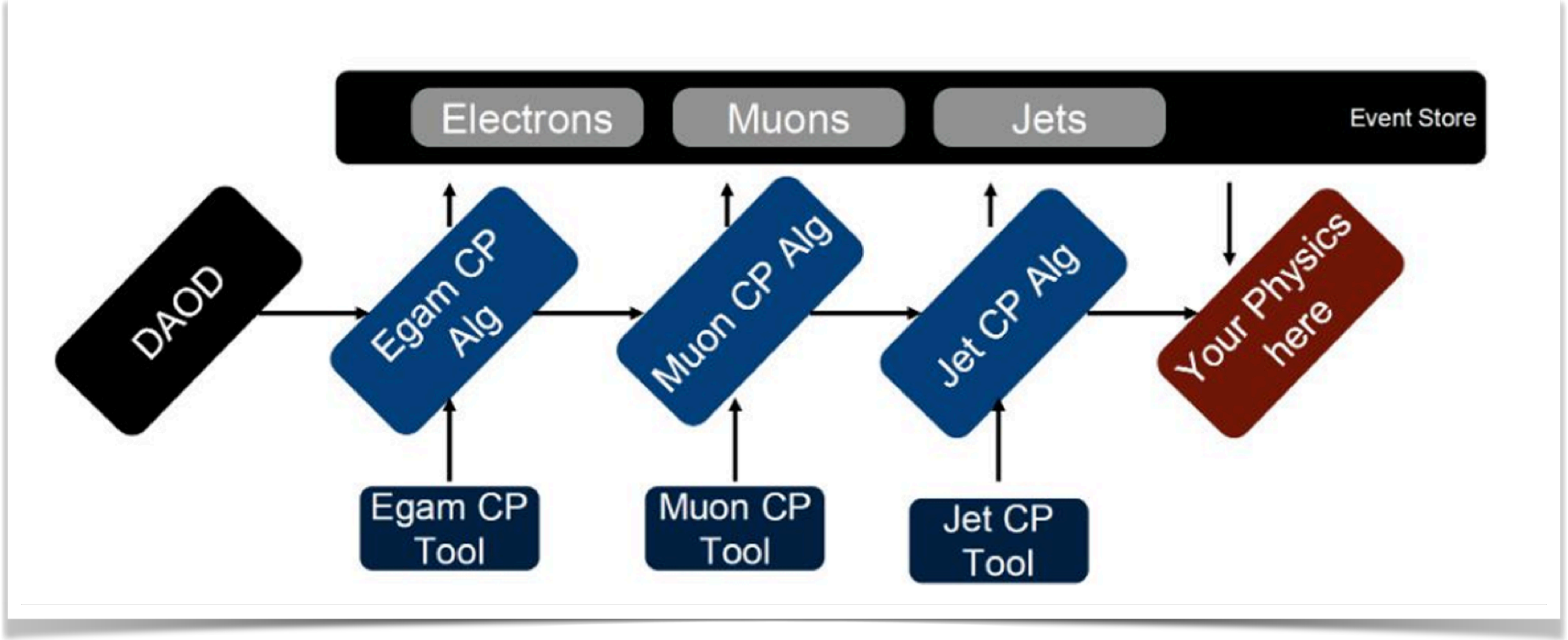
- A graph kernel is a symmetric, positive semidefinite function on the set of graphs $\boldsymbol{G}$.
  - $k: \boldsymbol{G} \times \boldsymbol{G} \rightarrow R$ $\quad \phi: \boldsymbol{G} \rightarrow H$ $\quad k(G_i, G_j) = \langle \phi(G_i), \phi(G_j) \rangle_H$ $\quad \langle\, , \rangle_H$ is the inner product in the Hilbert space

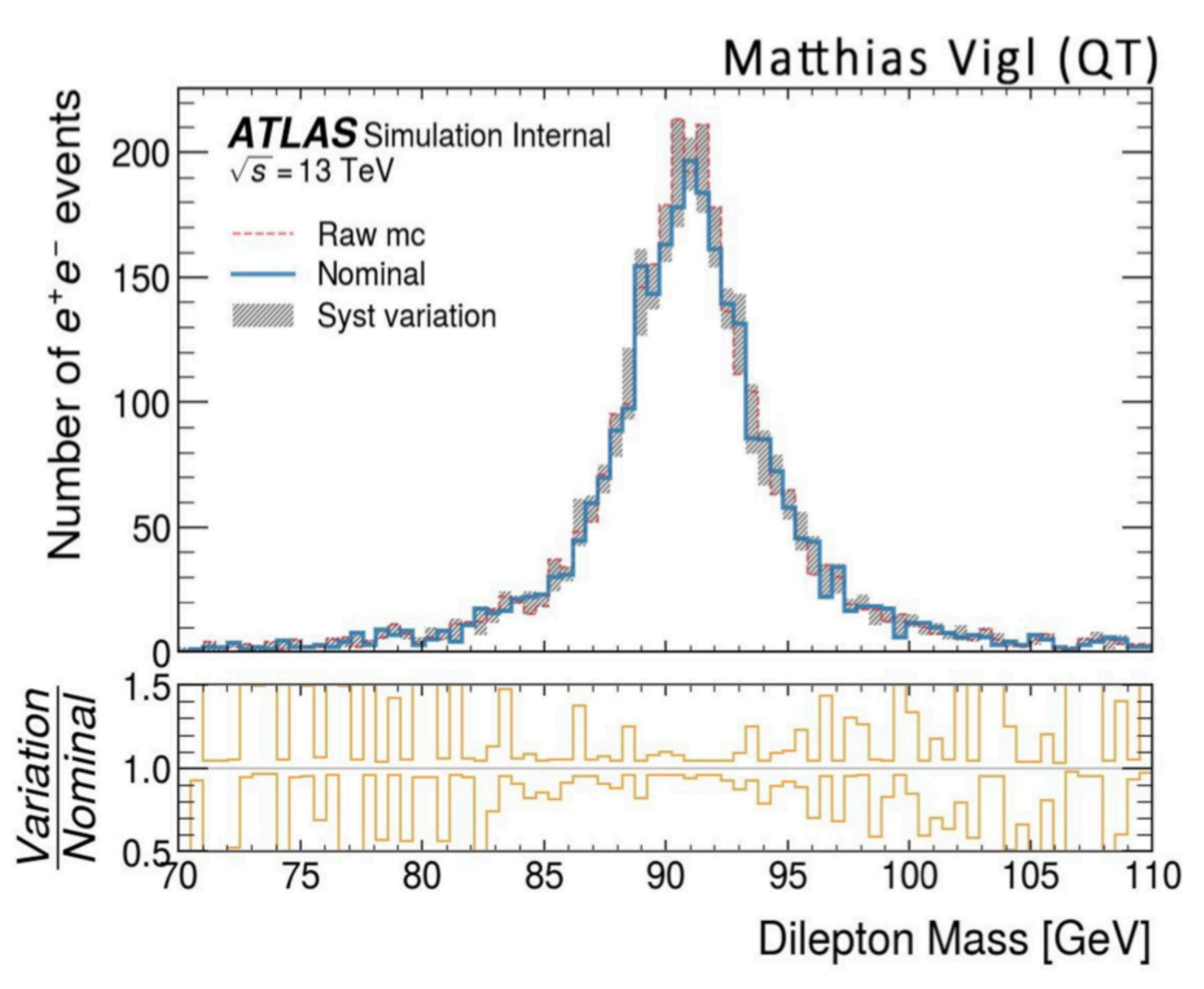
| # nodes | #chunks | execution time |
|---|---|---|
| 120 | 1000 | 1.12 minutes |

# ATLAS use-case III

- Effort just started
- My personal contribution: mainly coordinating the inclusion of the columnar analysis in the EGamma Calibration software
- Goal: evaluate computing performance on INFN clusters

## Columnar analysis implementation in CP tools




**Example to implement and improve: Zee demonstrator**

# Miscellanea

# International and Regional Conferences

- ECFA 2023 talk —> delivered, *link*
- ICHEP 2024 poster —> delivered, *link*
- CHEP 2024 —> abstract submitted, accepted as a talk *link*
- SIF 2024 —> abstract submitted, accepted as a talk *link*

# Presentations in Spoke 2 and WP2/5 Meetings

- Spoke 2 annual meeting talk: *link*
- Talks at WP2: *link*, *link*,
- Talks at WP5: *link*, *link*

# INFN - ICSC schools and courses attended

- I attended two INFN trainings for newly hired personnel at Frascati INFN laboratories, focussed on the INFN organisation and computing infrastructure (May 2024).
- I attended the INFN Introductory course to HLS (High-Level Synthesis) FPGA programming, promoted in the framework of the ICSC project (Nov. 2023).
- I attended and I successfully completed the individual project of the school SOSC 2023 Fifth International School on Open Science Cloud, focussing on Computing Models for Scientific Experiments (Oct. 2023).
- I attended the INFN First course about the porting on GPUs of code and algorithms, promoted by the ICSC project (June 2023).

# Public Engagement

- Ansa ICSC *link*
- Futuro Remoto @ città della scienza, HEPSCAPE room

# Conclusions & Next Steps

- Interactive analyses feasibility studies on the local testbed infrastructure & on INFN cloud 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

# Thank you!

# Back-up

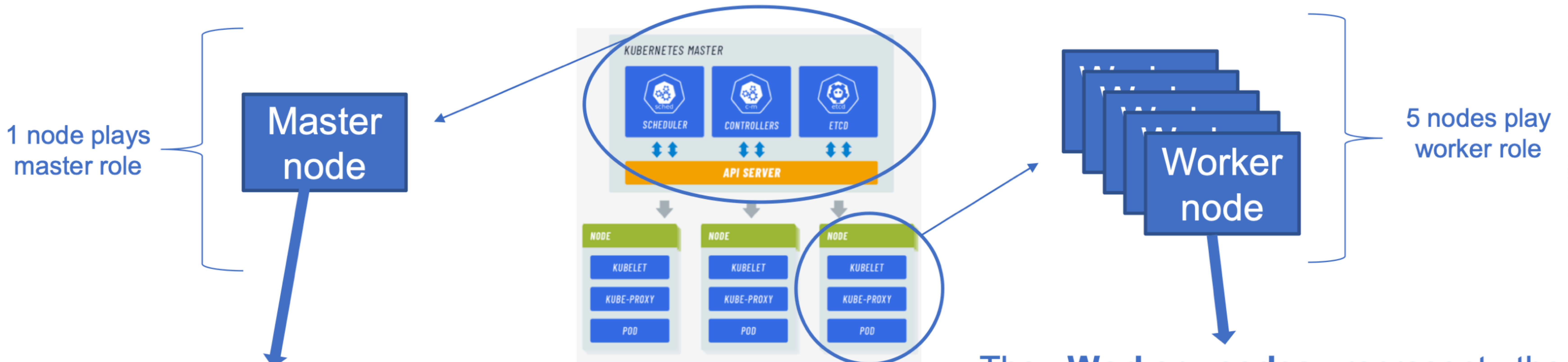
# Playground infrastructure at Naples (INFN)

- 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

# High throughput data analysis platform

- **Goal:** provide the users with an infrastructure that represents a tradeoff between deployment speed-flexibility, resource efficiency and service performance
- **Solution being tested:** the use of container technology (via Docker 20.10) that runs the applications and the Kubernetes tool for orchestration

Local testbed infrastructure provides 6 nodes, orchestrated via **Kubernetes (1.26.3)**:




The **Master node** coordinates and manages the entire Kubernetes infrastructure. It includes the API Server, the Controller Manager, the Scheduler, and etcd.

The **Worker nodes** represent the nodes of the cluster where containers are executed. Each one hosts the Kubelet, Kube-proxy and the containers, providing the actual computational capacity of the cluster.

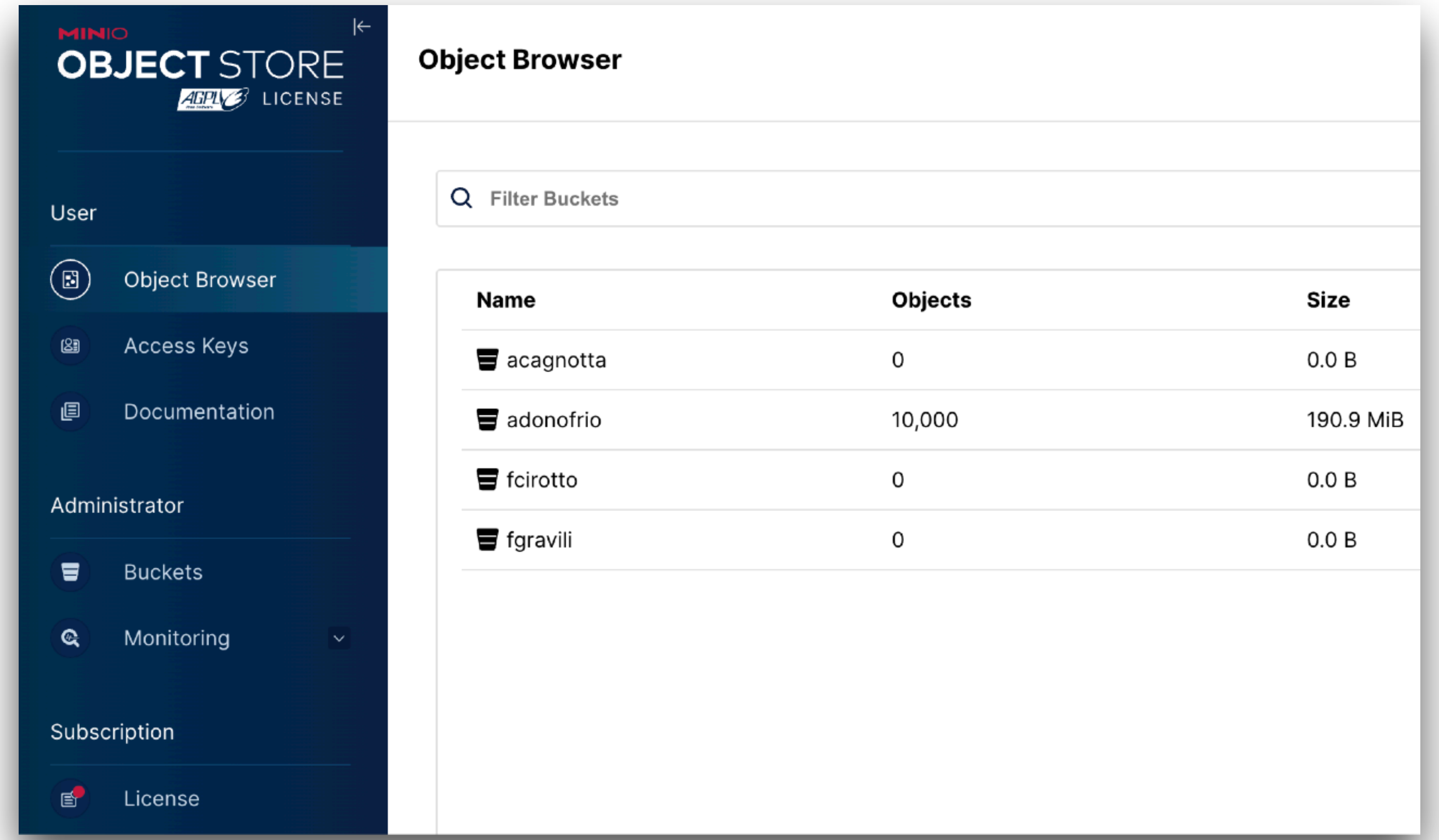
*Gianluca's talk*

# Efficient & user friendly infrastructure

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

## MinIO

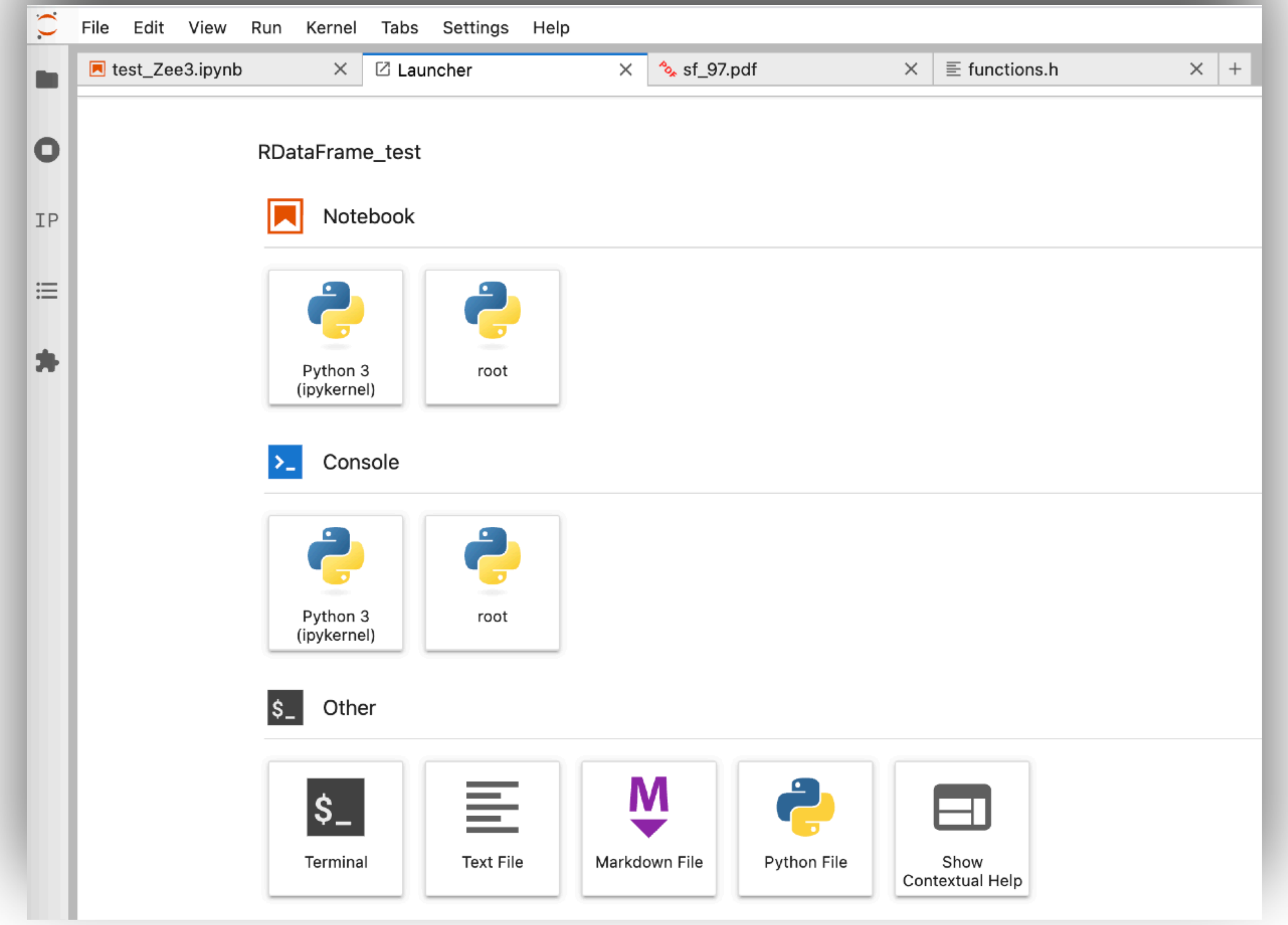
An object storage instance where users can store data




Gianluca's presentation *link*

## Jupyter

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



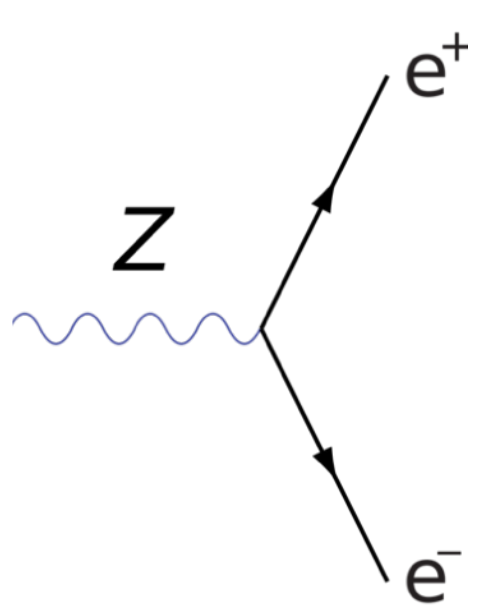

## Dask

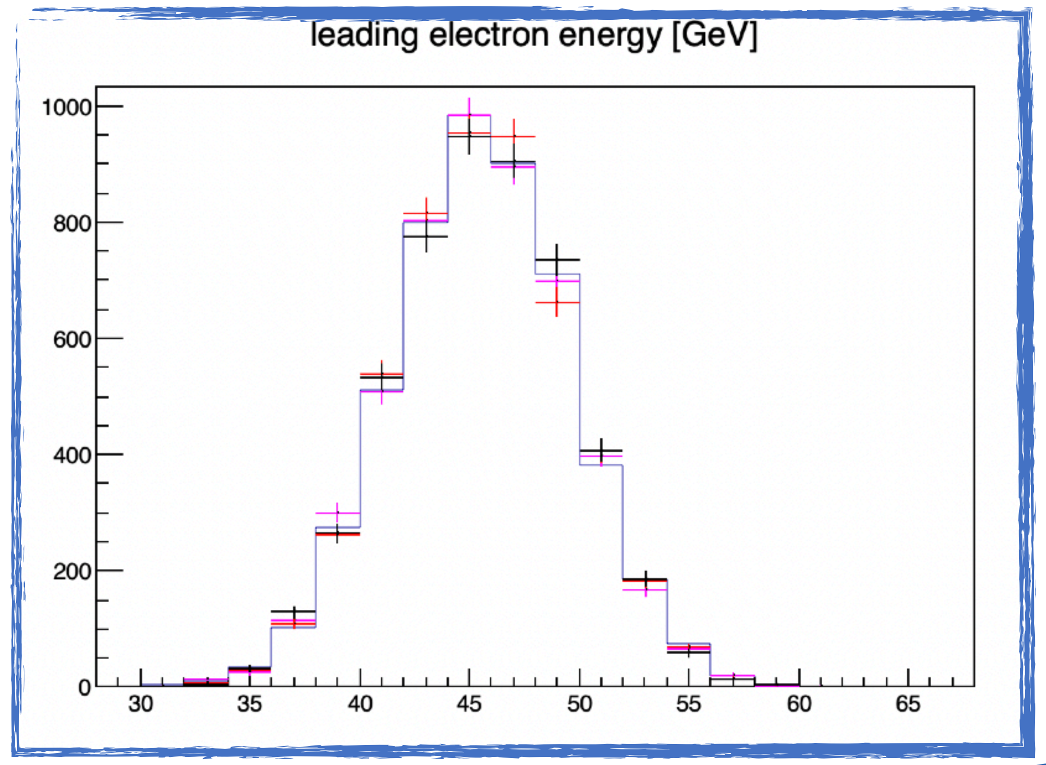
A python library to scale python code from multi-core local machines to large distributed clusters in the cloud

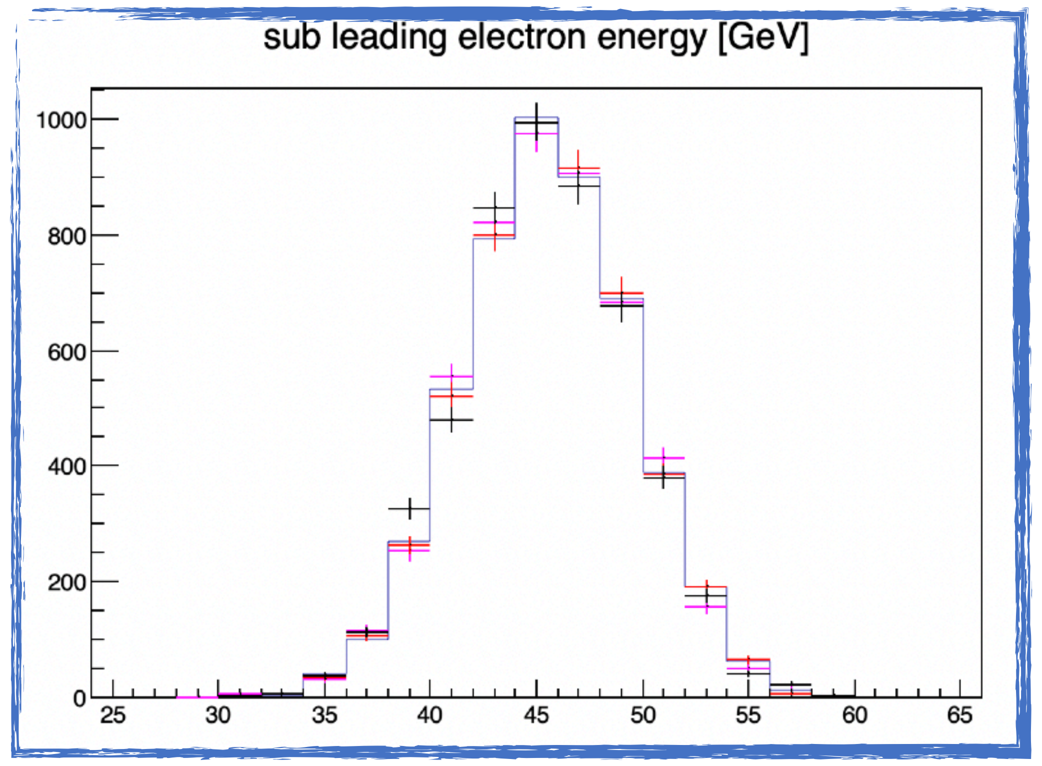
- Jupyter interface includes:
  - Terminal
  - Notebook implementation
    - Completely exportable and replicable

# Simple test

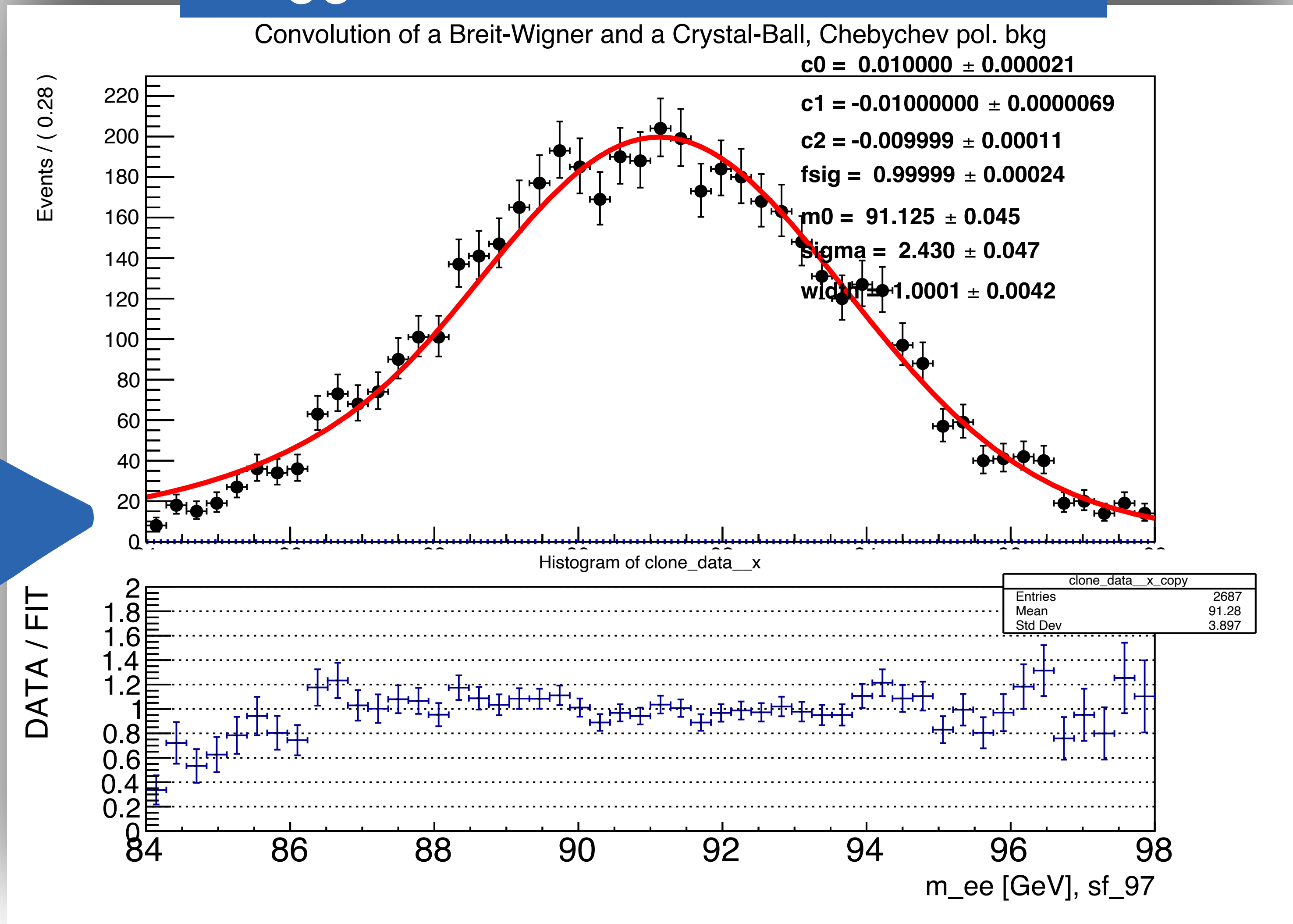
- Simulation exploited:
  - 5k events, scaled to 1M events replicating the available dataset
  - Idea: mimic systematic variations, gaussian smearing the electrons energy to compute $M_{ee}$ resolution



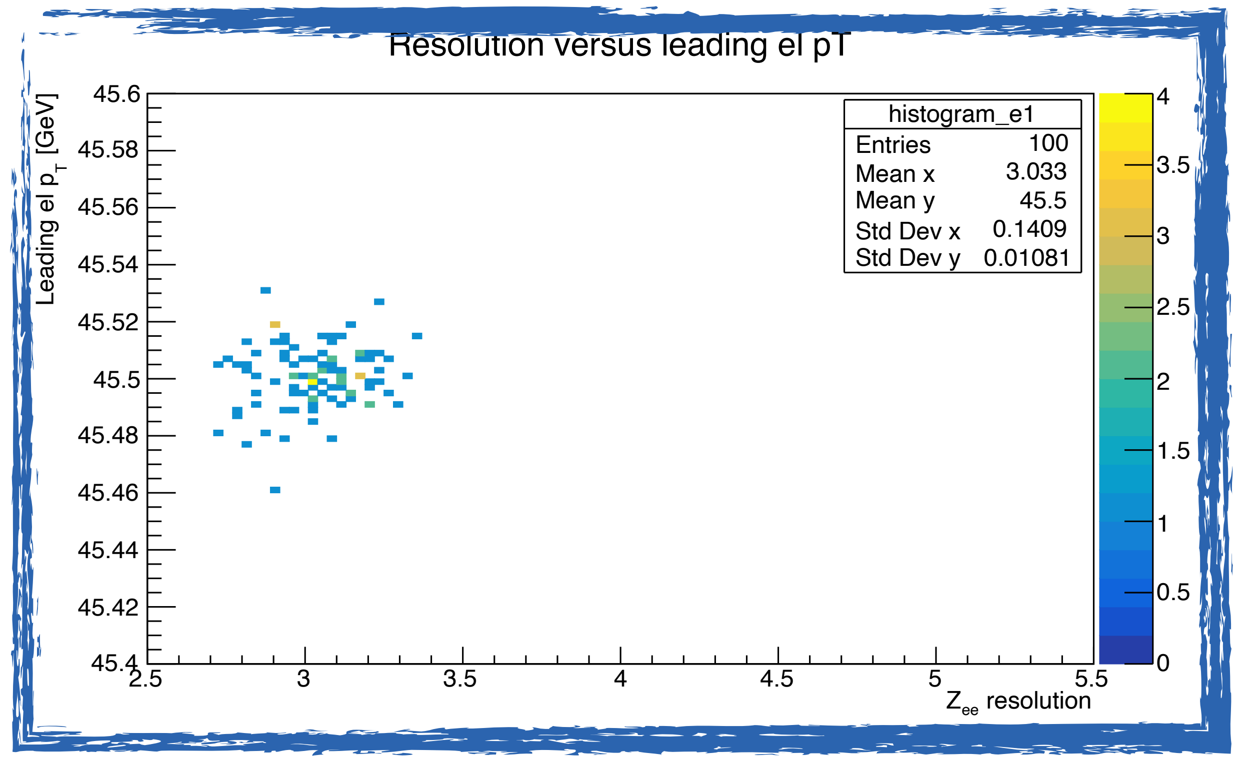



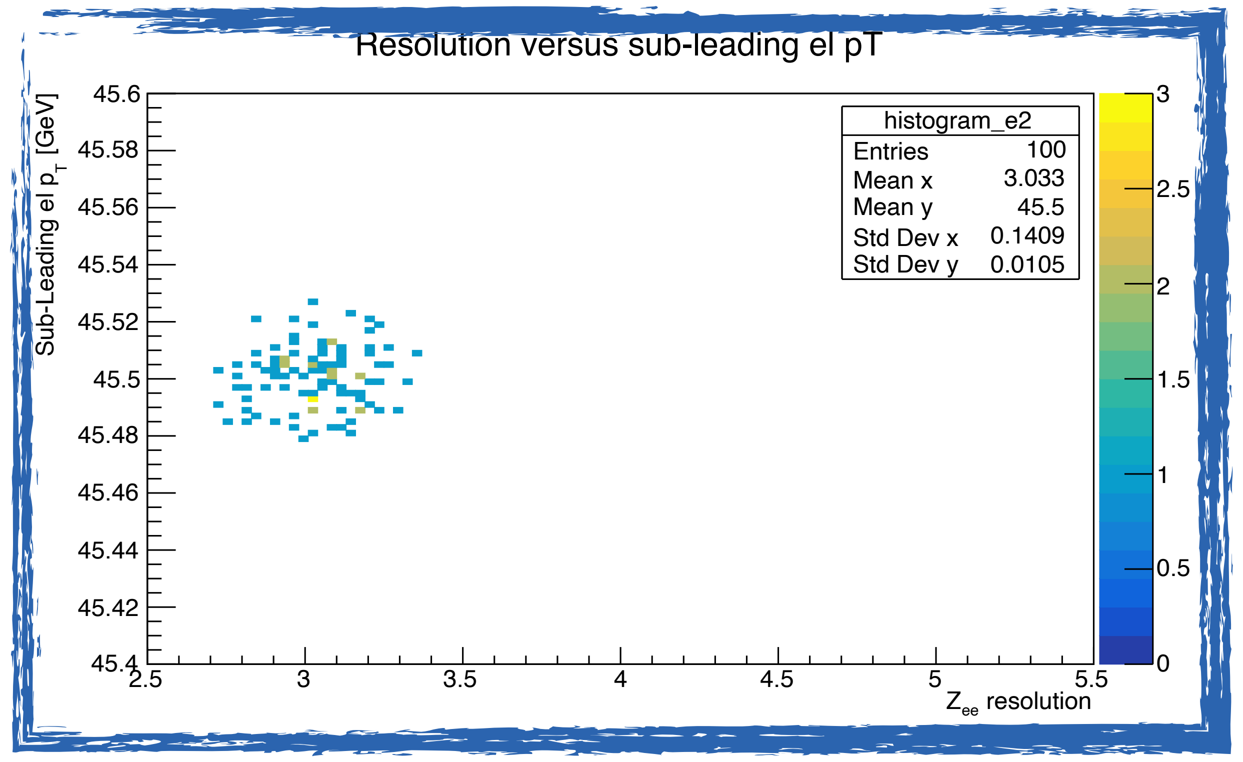

**$M_{ee}$ invariant mass fit**



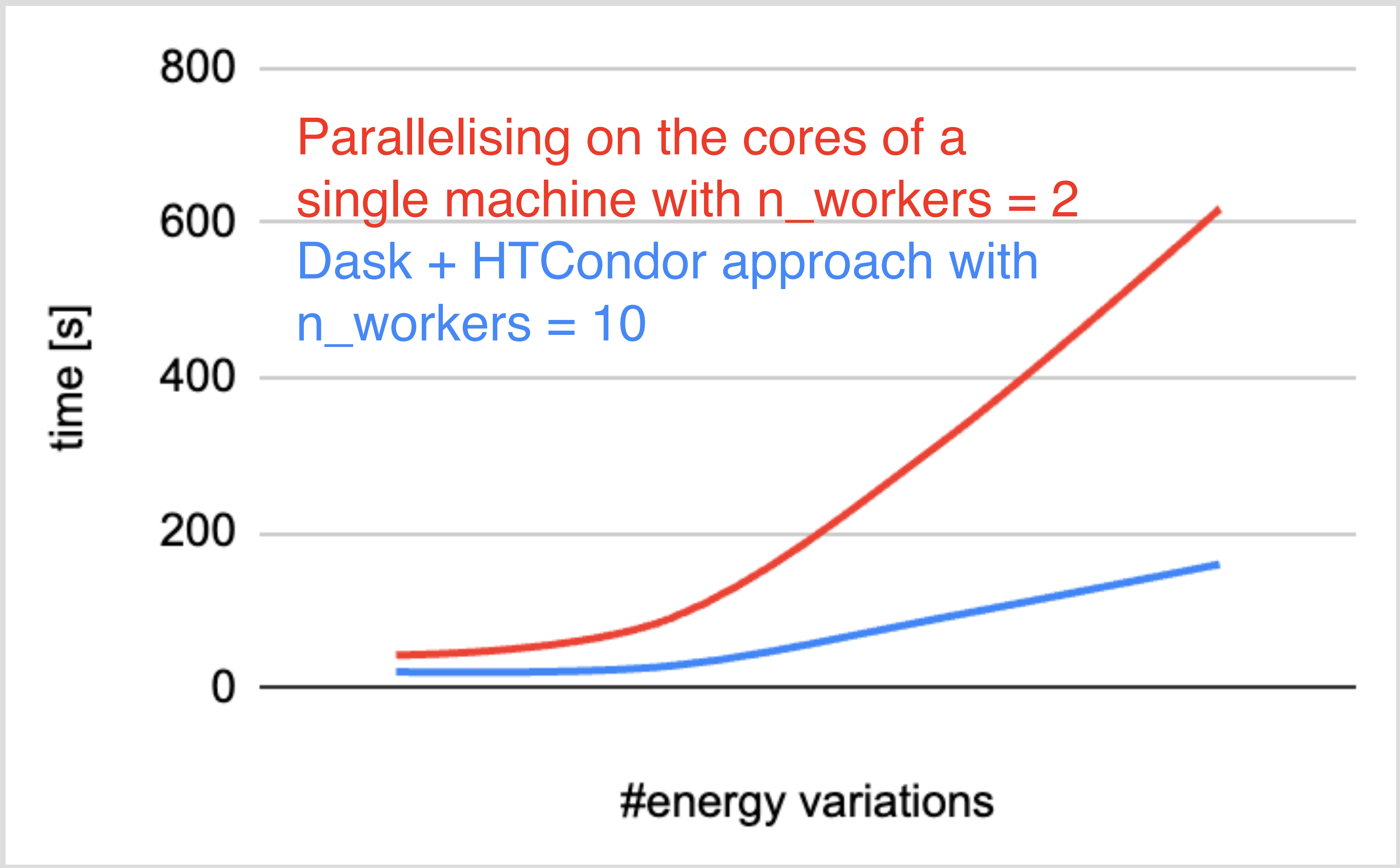

**Selection and histogramming interactively via RDataFrame on JupyterHub**

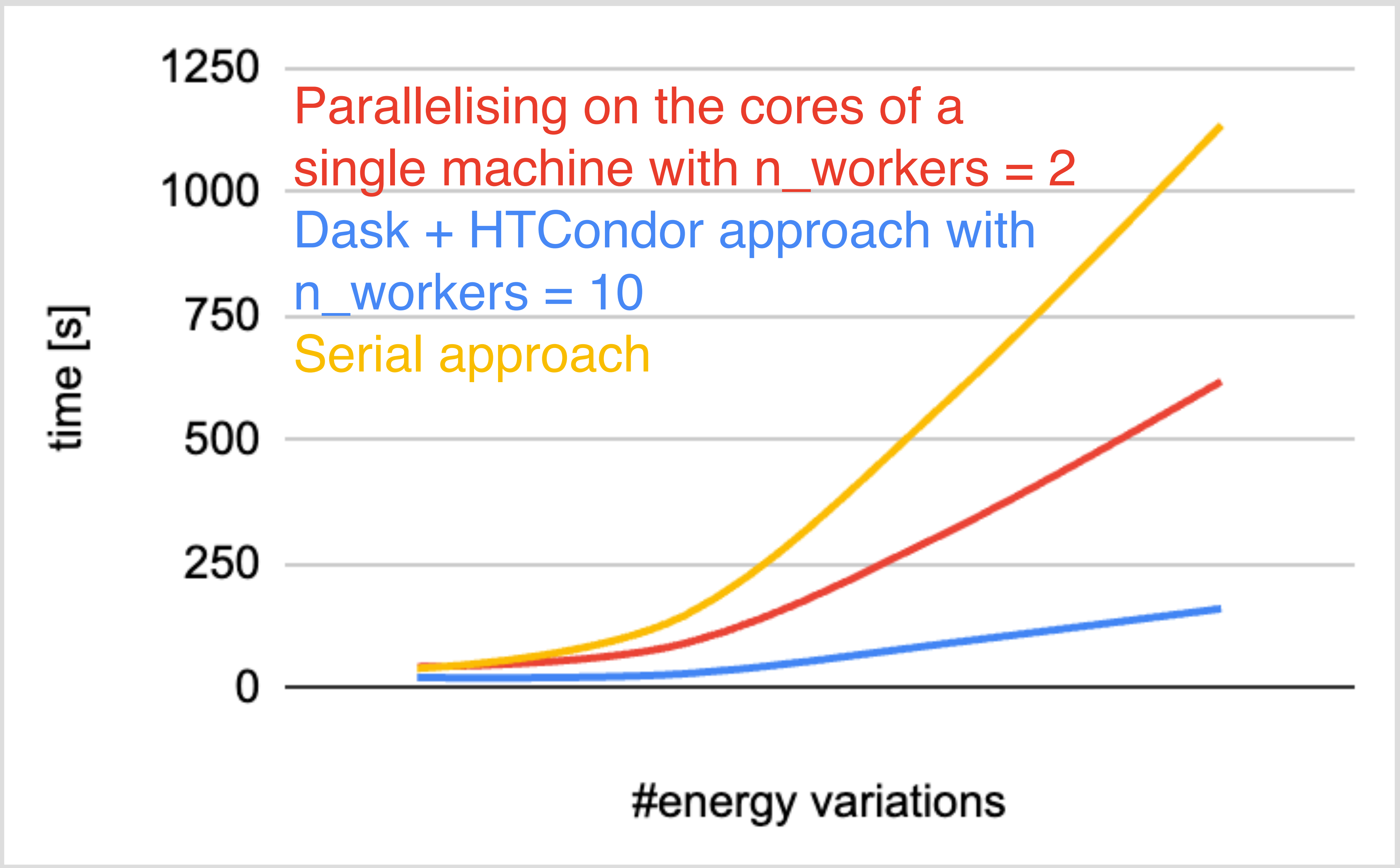
*github link to the code*



# 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