Enhancing CMS data analyses using a distributed high throughput platform

Tommaso Diotalevi
Carlo Battilana
Alessandra Fanfani
Daniele Bonacorsi

on behalf of the CMS Collaboration

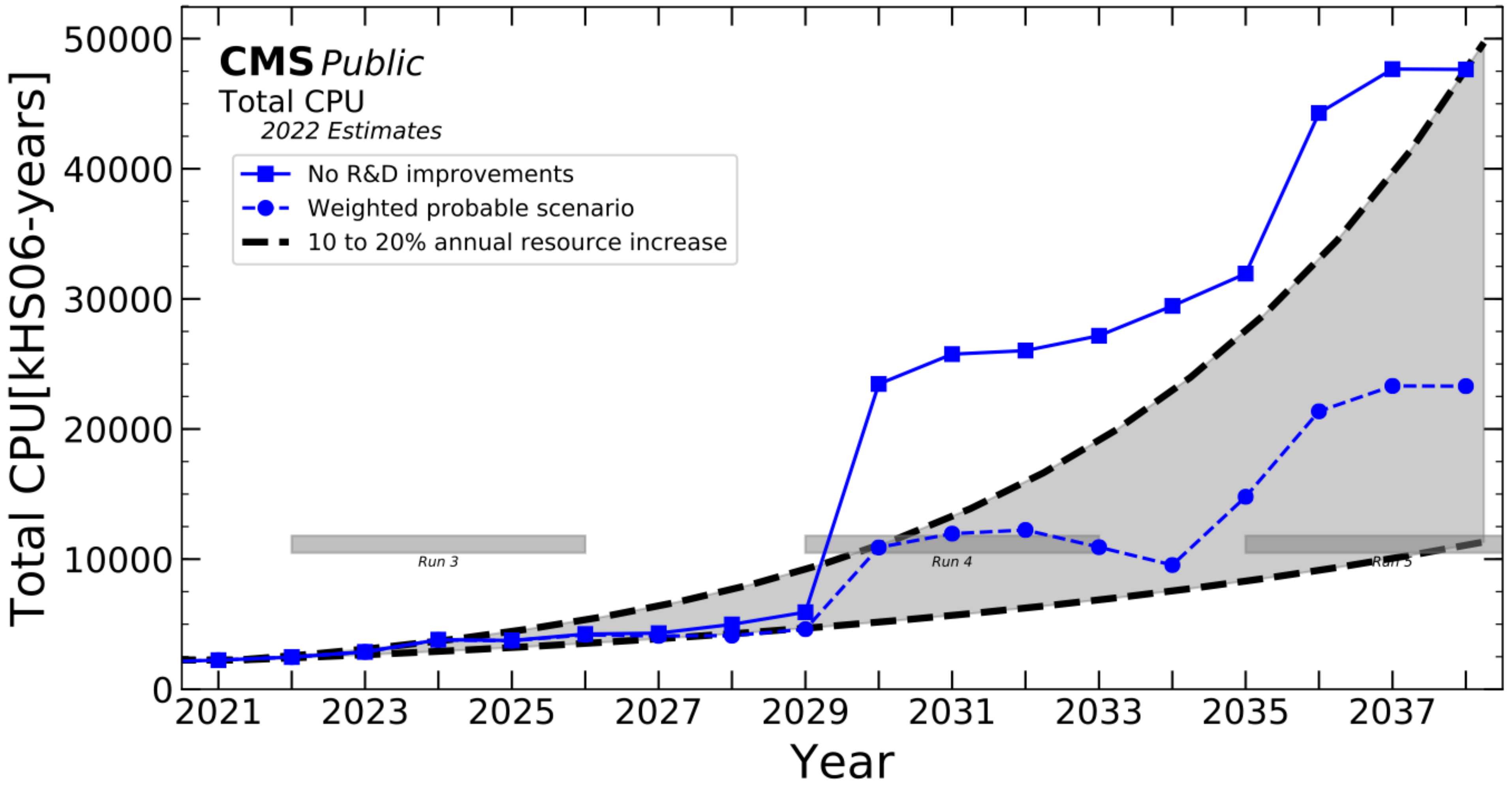
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
INFN BOLOGNA
CMS Compact Muon Solenoid

20th July 2024
Prague, Czechia

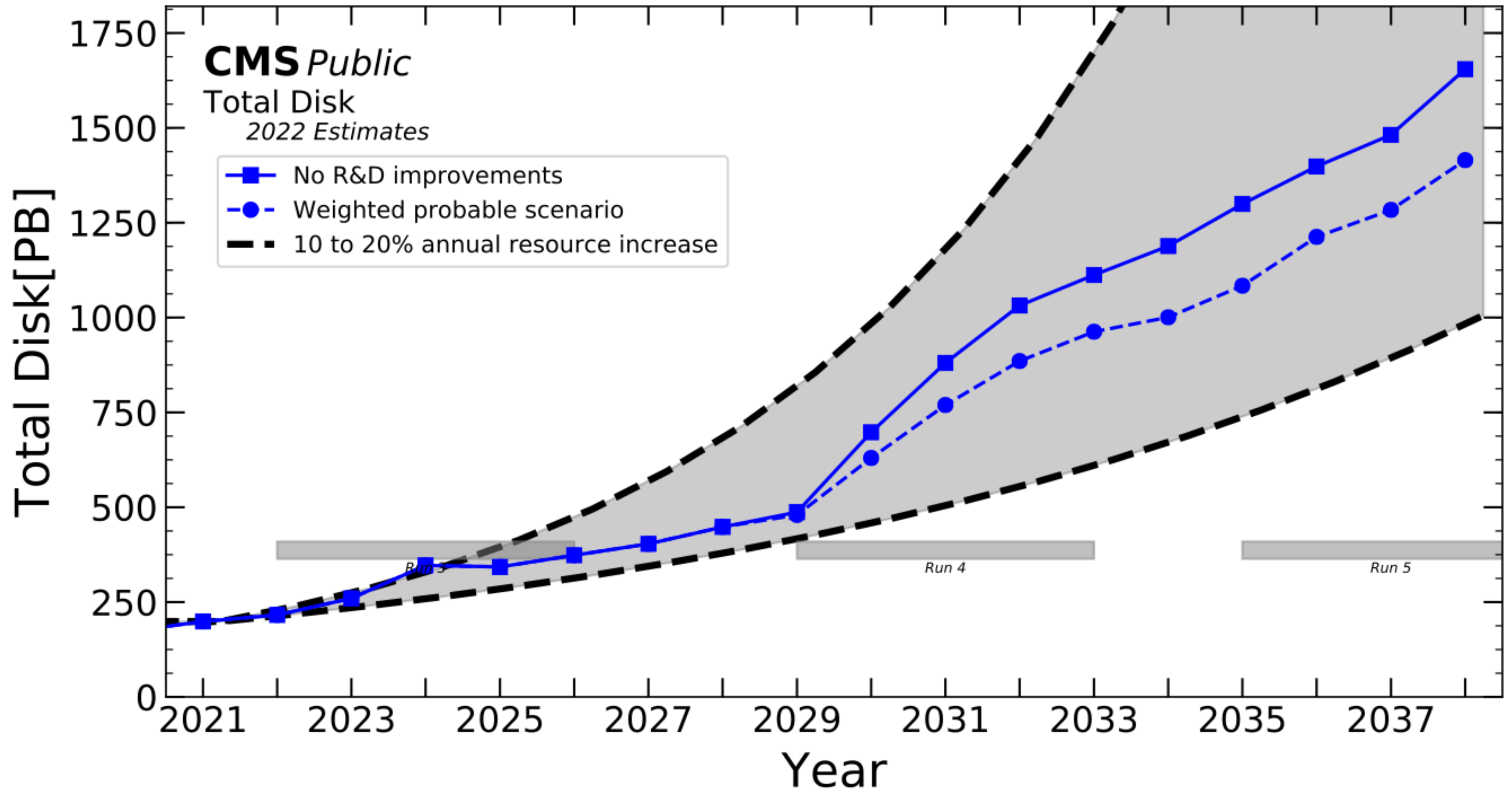
42nd International Conference on High Energy Physics - **ICHEP 2024**

Introduction

- The upcoming high-luminosity phase at the CERN Large Hadron Collider (LHC), will require an increasing amount of computing resources;



link



link

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

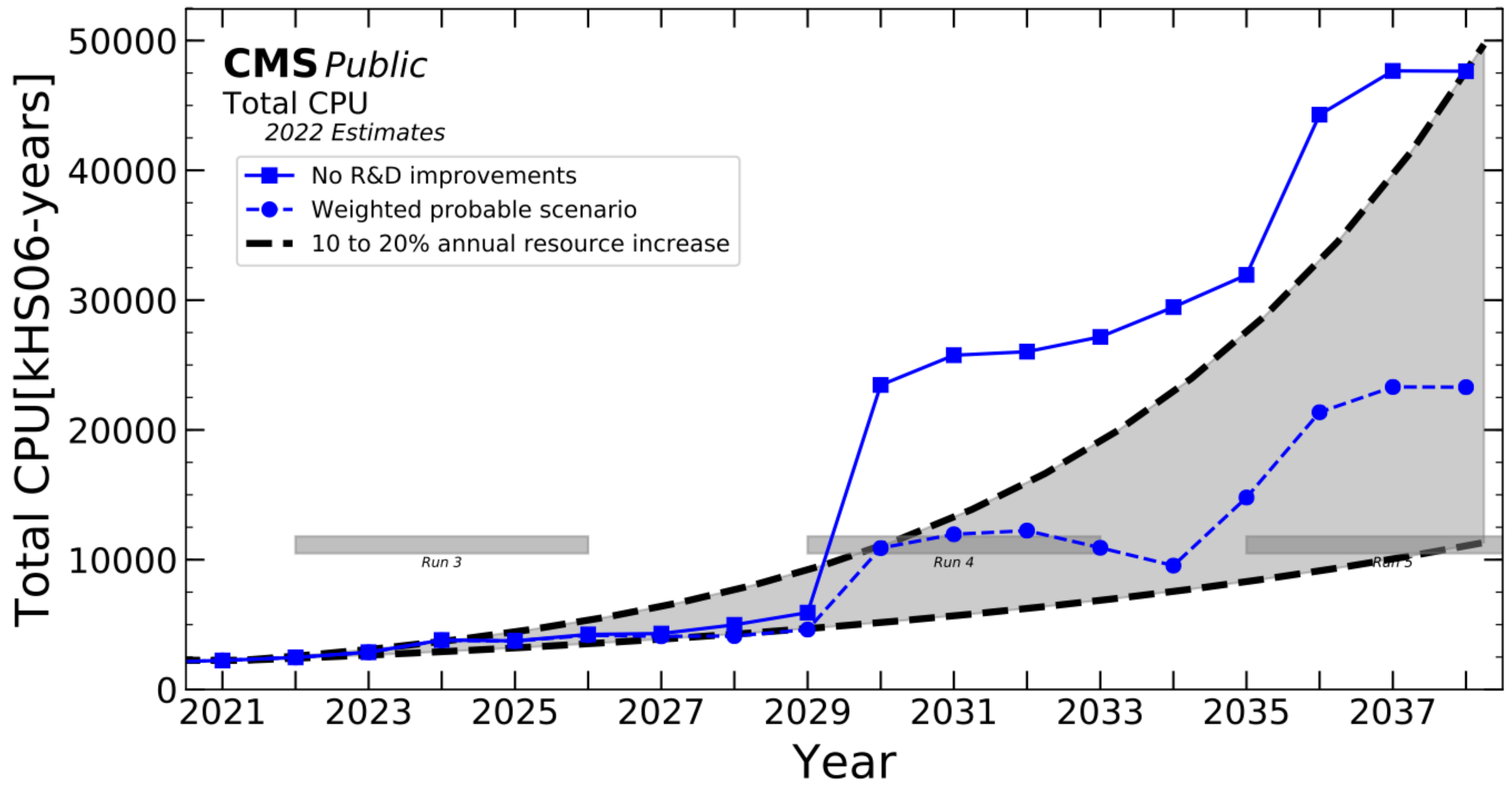
- To better analyse this increasing amount of Big Data:
  - Optimise the usage of CPU and storage;
  - Promote the usage of better data formats;
  - **Develop new analysis paradigms!**

→

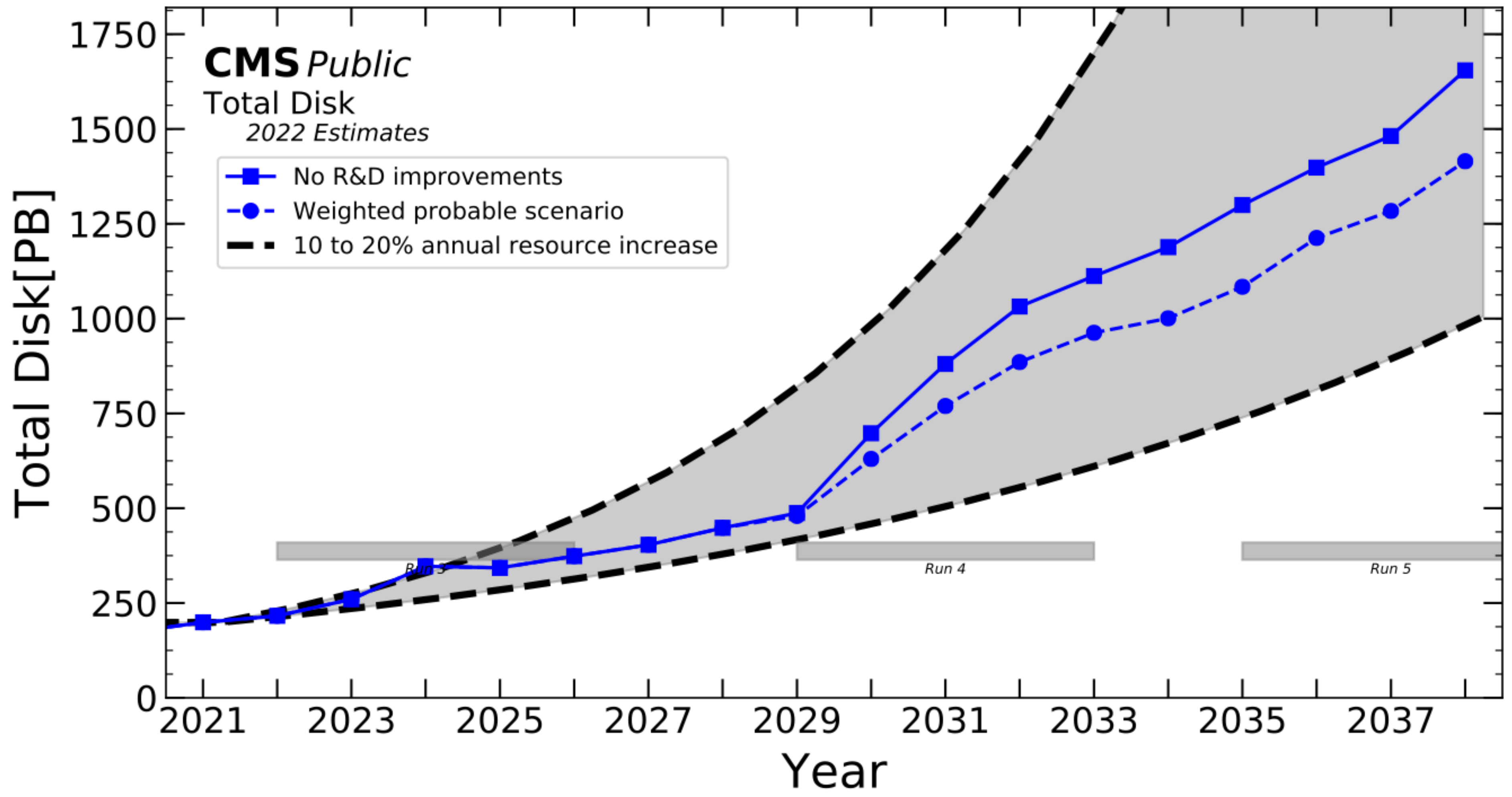
- New software based on declarative programming and interactive workflows;
- Distributed computing on geographically separated resources.

Introduction

- The upcoming high-luminosity phase at the CERN Large Hadron Collider (LHC), will require an increasing amount of computing resources;



link



link

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

- To better analyse this increasing amount of Big Data:

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

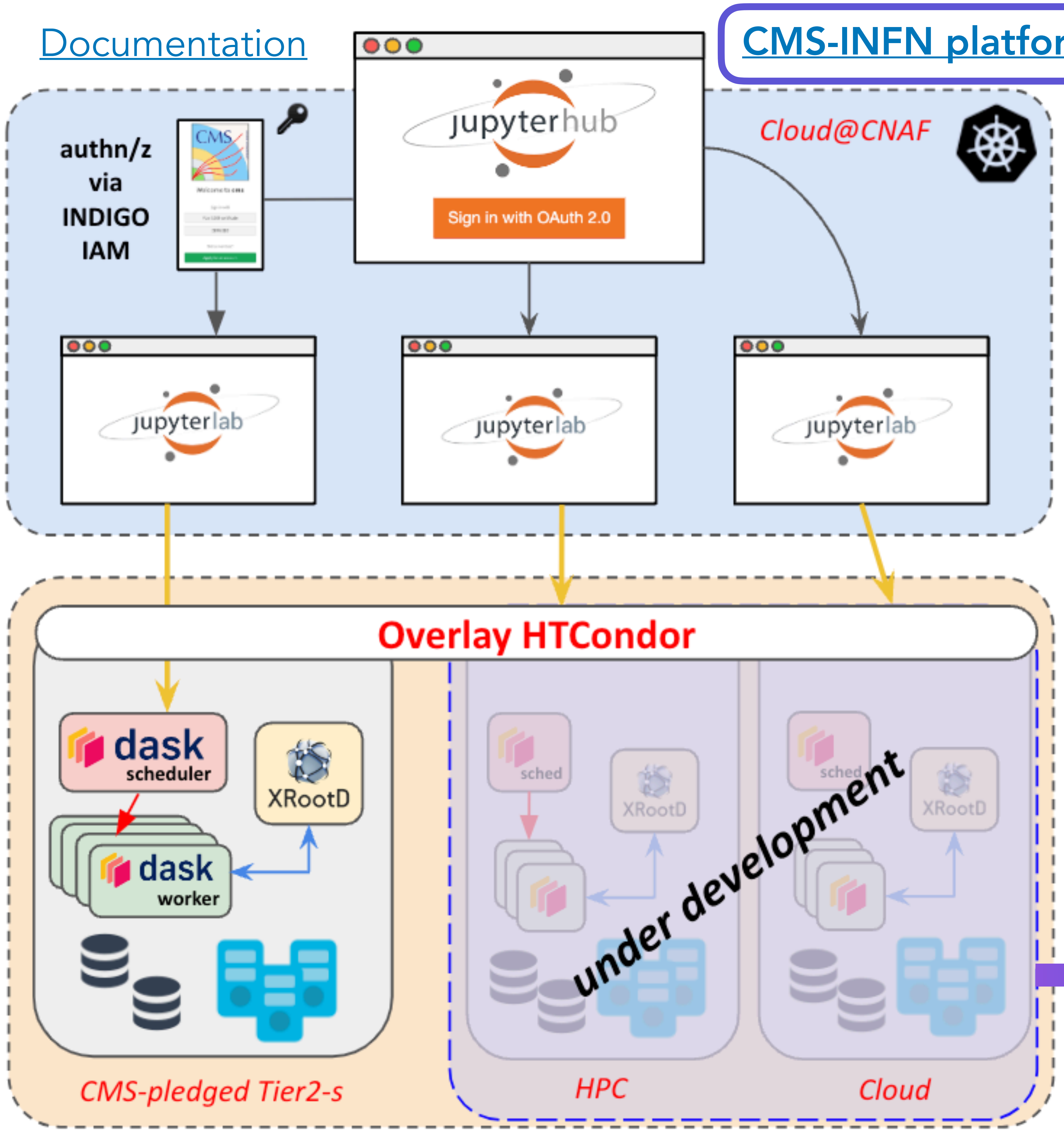
→

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

High-throughput platform!

What is a high throughput platform?

[Documentation](#) [CMS-INFN platform](#)



synergy with: Common Analysis Tools @ CMS

- Access to a single JupyterHub and authentication/authorisation token-based (Indigo-IAM);
- Based on industry standard technologies;
- Configurable kernel python (via containers), with specific working environment.

- HTCondor-based overlay (also available standalone);
- DASK library (python) for distributing the execution:
  - ✳ Scale from 1 to N cores (depending on resources availability)
- Interfaced with WLCG (using XRootD, WebDAV, …)

Offloading strategy:

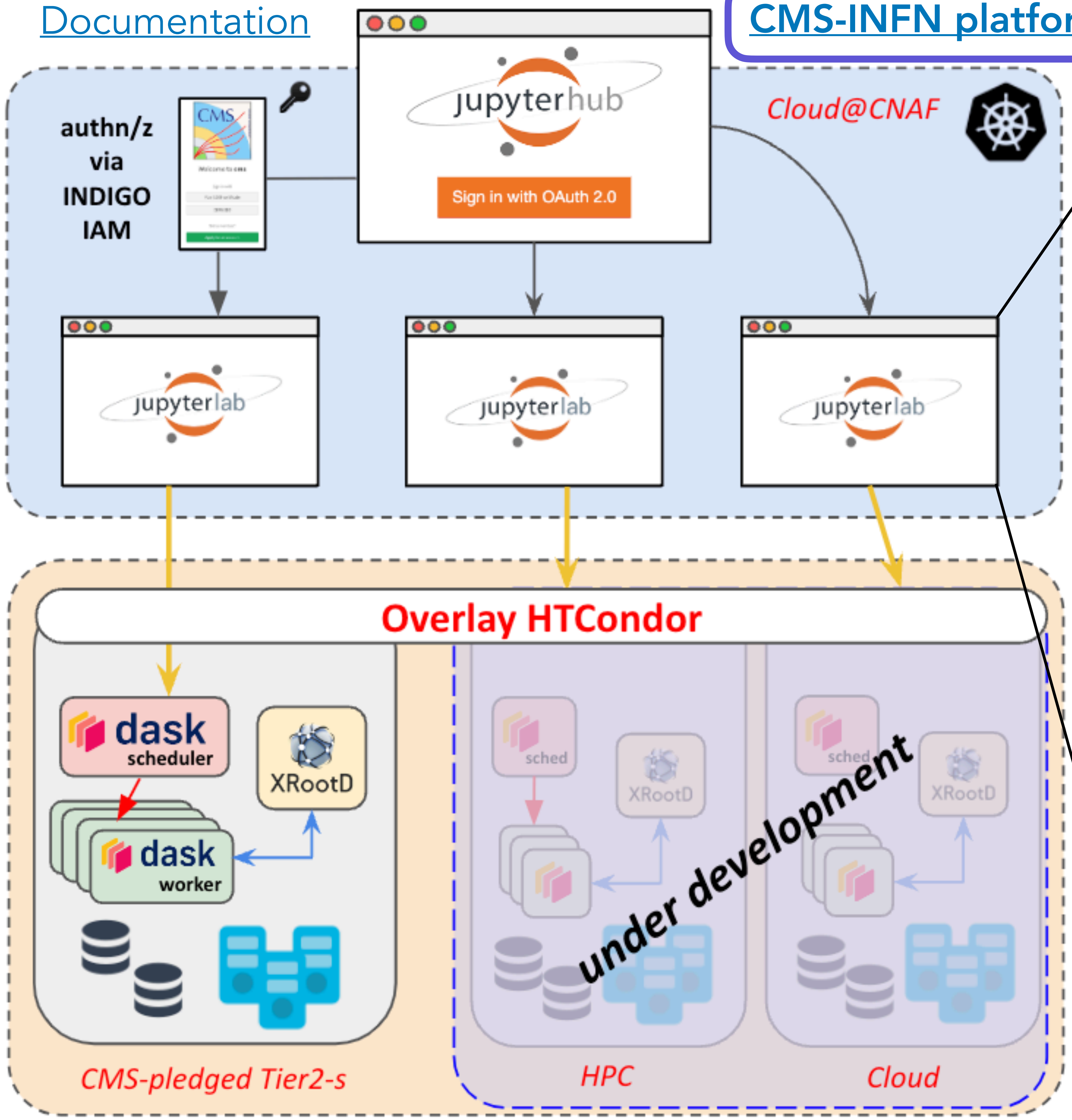
synergy with: interTwin

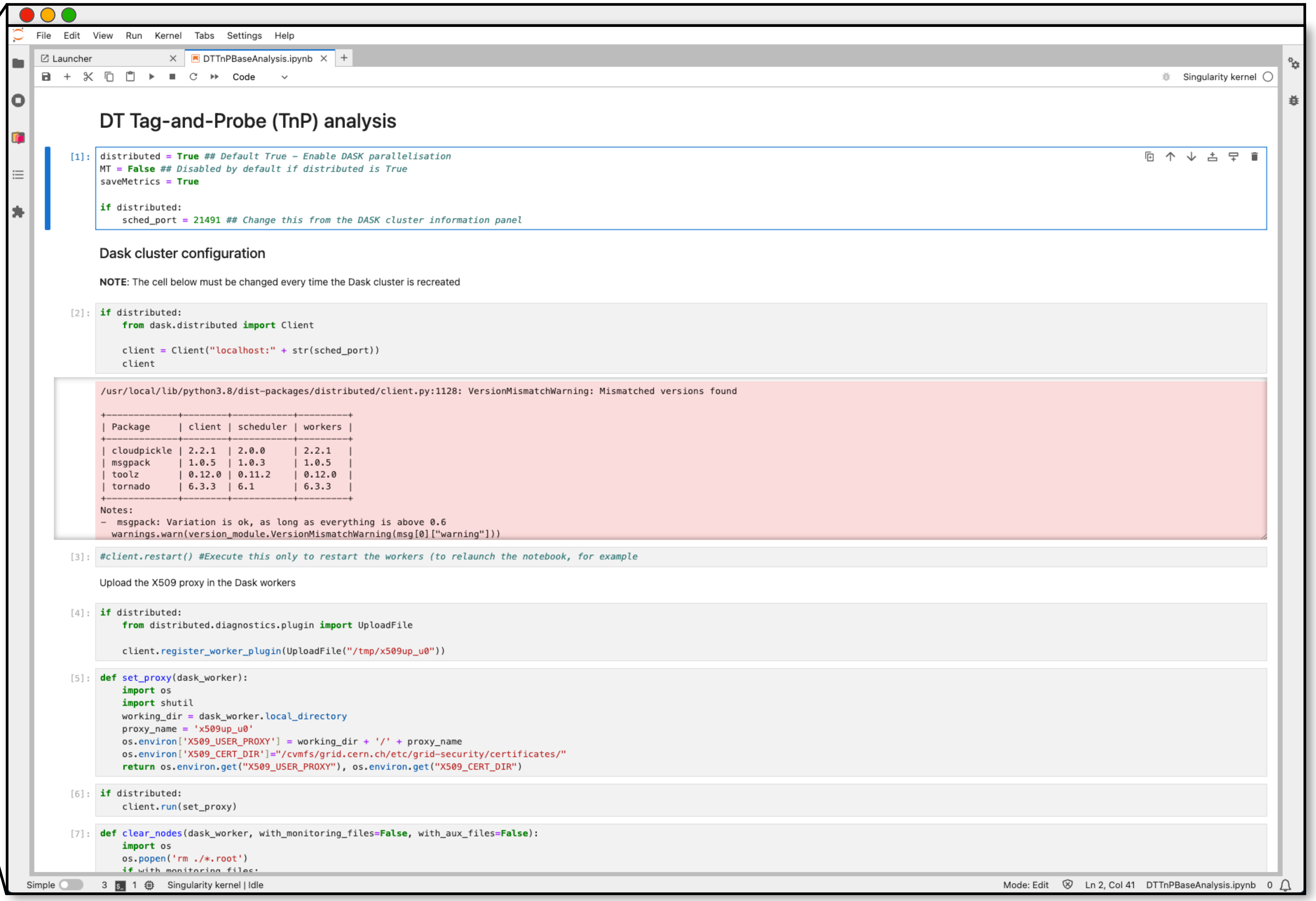
- Schedule worker processes spawning on multiple remote sites dynamically and transparently:
  - Implementation on heterogeneous resources (HTC/HPC/Cloud)

What is a high throughput platform?

Documentation

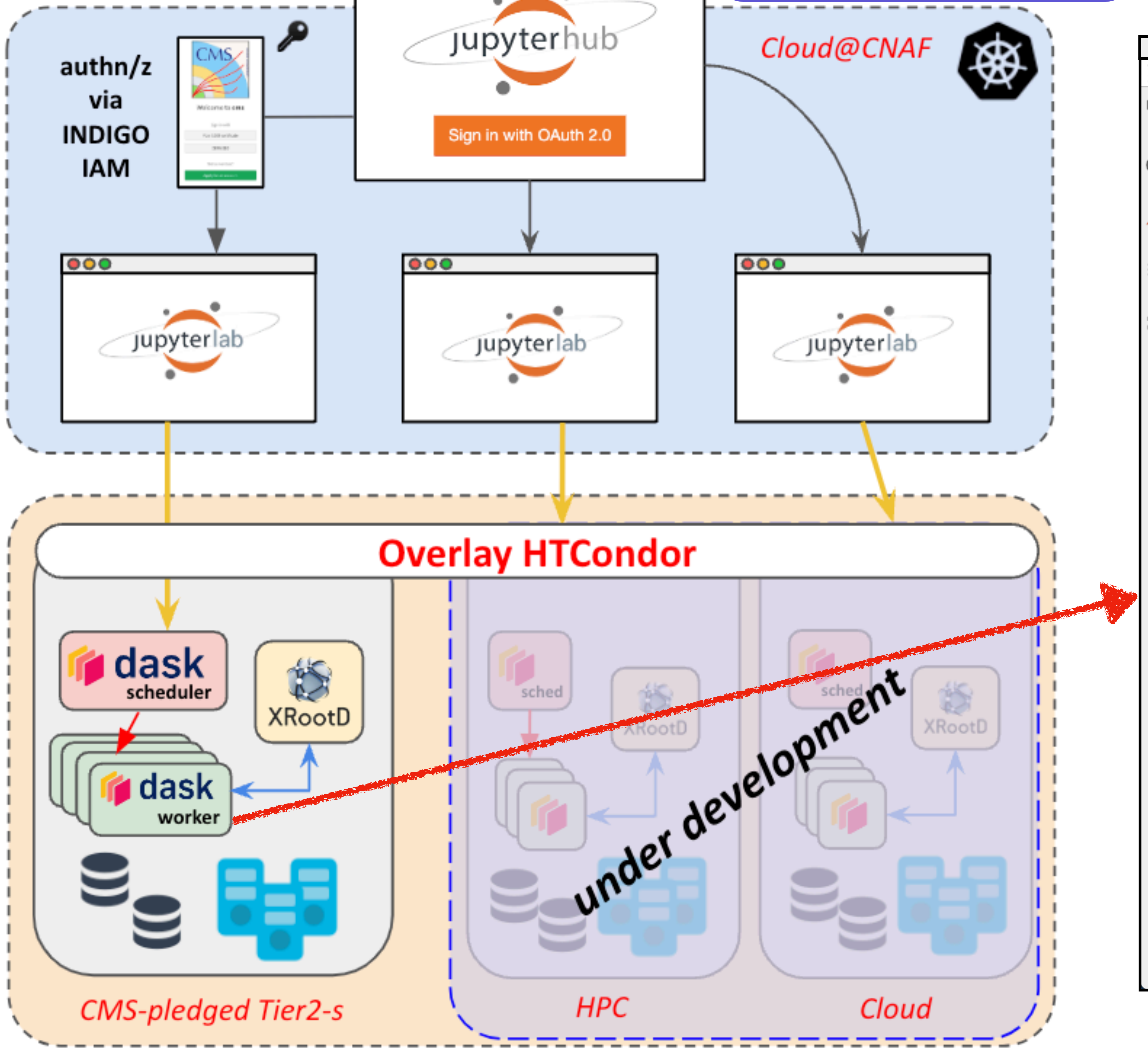
CMS-INFN platform

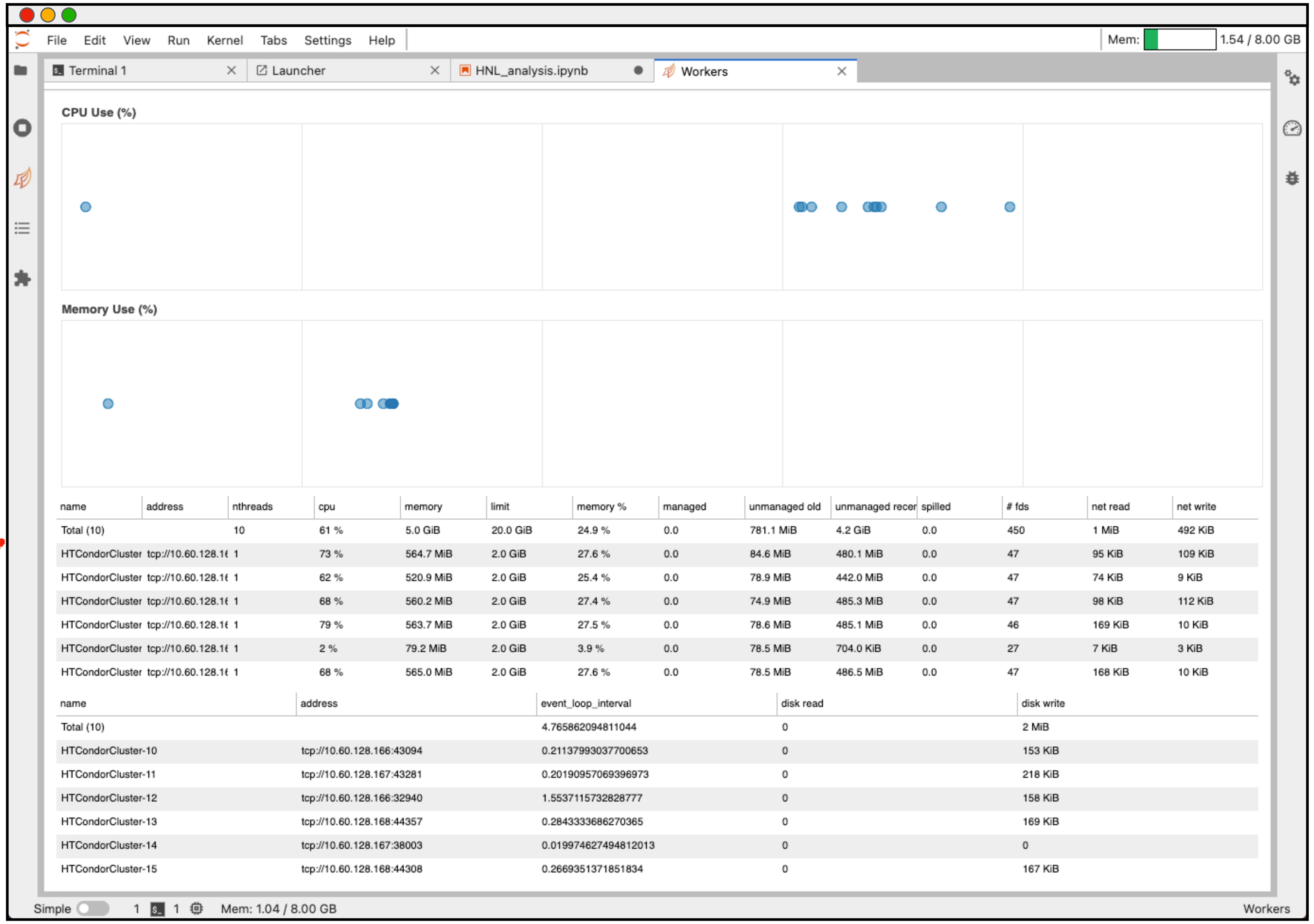




What is a high throughput platform?

[Documentation](#) [CMS-INFN platform](#)





name	address	nthreads	cpu	memory	limit	memory %	managed	unmanaged old	unmanaged recer	spilled	# fds	net read	net write
Total (10)		10	61 %	5.0 GiB	20.0 GiB	24.9 %	0.0	781.1 MiB	4.2 GiB	0.0	450	1 MiB	492 KiB
HTCondorCluster	tcp://10.60.128.1(	1	73 %	564.7 MiB	2.0 GiB	27.6 %	0.0	84.6 MiB	480.1 MiB	0.0	47	95 KiB	109 KiB
HTCondorCluster	tcp://10.60.128.1(	1	62 %	520.9 MiB	2.0 GiB	25.4 %	0.0	78.9 MiB	442.0 MiB	0.0	47	74 KiB	9 KiB
HTCondorCluster	tcp://10.60.128.1(	1	68 %	560.2 MiB	2.0 GiB	27.4 %	0.0	74.9 MiB	485.3 MiB	0.0	47	98 KiB	112 KiB
HTCondorCluster	tcp://10.60.128.1(	1	79 %	563.7 MiB	2.0 GiB	27.5 %	0.0	78.6 MiB	485.1 MiB	0.0	46	169 KiB	10 KiB
HTCondorCluster	tcp://10.60.128.1(	1	2 %	79.2 MiB	2.0 GiB	3.9 %	0.0	78.5 MiB	704.0 KiB	0.0	27	7 KiB	3 KiB
HTCondorCluster	tcp://10.60.128.1(	1	68 %	565.0 MiB	2.0 GiB	27.6 %	0.0	78.5 MiB	486.5 MiB	0.0	47	168 KiB	10 KiB

name	address	event_loop_interval	disk read	disk write
Total (10)		4.765862094811044	0	2 MiB
HTCondorCluster-10	tcp://10.60.128.166:43094	0.21137993037700653	0	153 KiB
HTCondorCluster-11	tcp://10.60.128.167:43281	0.20190957069396973	0	218 KiB
HTCondorCluster-12	tcp://10.60.128.166:32940	1.5537115732828777	0	158 KiB
HTCondorCluster-13	tcp://10.60.128.168:44357	0.2843333686270365	0	169 KiB
HTCondorCluster-14	tcp://10.60.128.167:38003	0.019974627494812013	0	0
HTCondorCluster-15	tcp://10.60.128.168:44308	0.2669351371851834	0	167 KiB

Use case: Detector Performance Analyses

Typically, **Detector Performance Group** (DPG) analyses are run on a reduced amount of data (e.g. one run or fill), but processing of large dataset, at once, might be needed:

- To assess/improve systematics of high precision analyses, when they are dominated by the response of a specific detector;
- To reprocess multiple year data, e.g. for detector stability studies (ageing).

Use case:

Porting of a well established Drift Tubes (DT) Tag-and-Probe analysis [CMS-DP-2023-049]

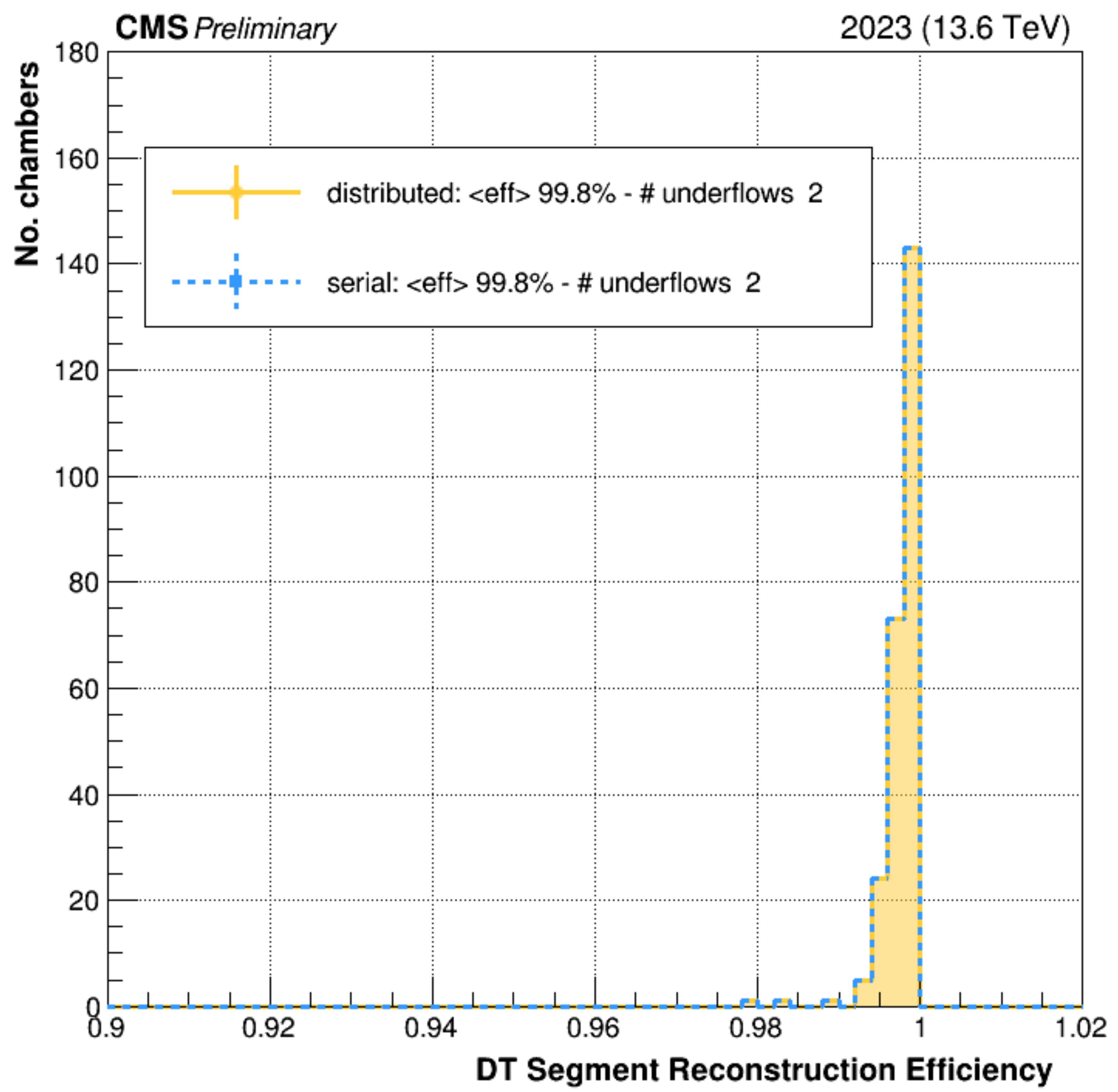
A **data sample** consisting in a skim of $Z \to \mu\mu$ **decay candidates** collected by CMS over **2023**, corresponding to ~27fb^{-1} was explored for the study. Size: 224GB

- The original code running mainly on C++, for the base histograms and computing the segment efficiencies.
    - The ported code is running on Jupyter notebook (in Python), using **ROOT RDataFrame**. The Tag-and-Probe libraries are stored in a dedicated header file.
    - The execution is off-loaded remotely (CMS Tier-2 - LNL) and the results are retrieved directly on the platform.

Use case: Detector Performance Analyses
Porting results

- Tag-and-Probe method [CMS-DP-2023-049]:
    - Two oppositely-charged well-reconstructed tracker muons;
    - Tag muon: $p_T > 27\text{GeV}$ passing HLT for isolated muons. TightID criteria in the muon detector reconstruction.
    - Probe muon: track with segment matching in at least a chamber other than the one under study. $p_T > 20\text{GeV}$.
- A DT chamber is efficient if reconstructed segment is near the extrapolated probe muon track.
- The **efficiency is computed in fiducial regions** (ignoring probes whose tracks falls near the chambers borders).
- The changes applied to the Tag-and-Probe program to run on the High Throughput Platform (distributed) do not affect performance, which is consistent with the original program (serial).

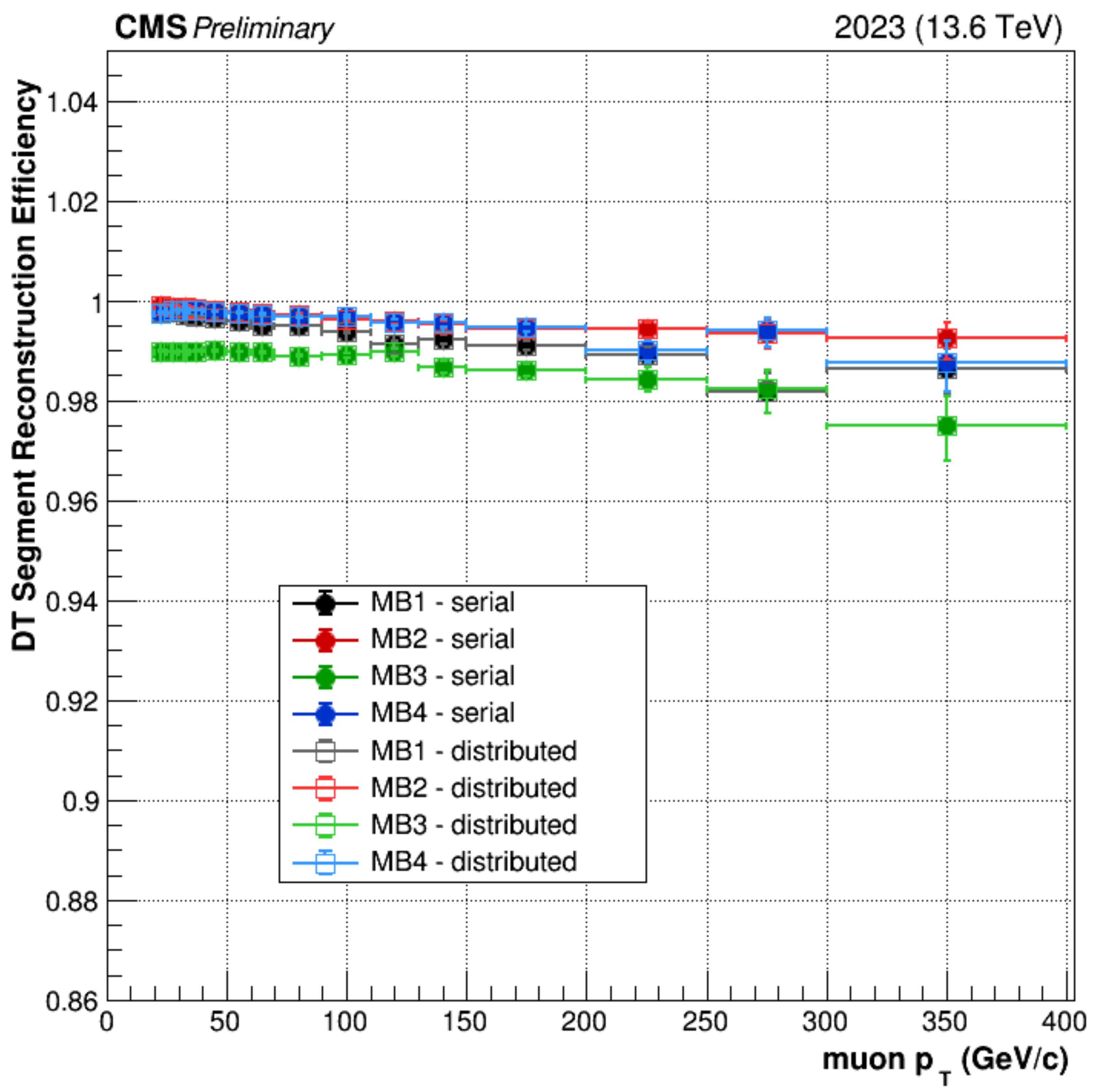
(one entry per chamber)



Use case: Detector Performance Analyses
Porting results

- Tag-and-Probe method [CMS-DP-2023-049]:
    - Two oppositely-charged well-reconstructed tracker muons;
    - Tag muon: $p_T > 27\text{GeV}$ passing HLT for isolated muons. TightID criteria in the muon detector reconstruction.
    - Probe muon: track with segment matching in at least a chamber other than the one under study. $p_T > 20\text{GeV}$.
- A DT chamber is efficient if reconstructed segment is near the extrapolated probe muon track.
- The **efficiency is computed in fiducial regions** (ignoring probes whose tracks falls near the chambers borders).
- The changes applied to the Tag-and-Probe program to run on the High Throughput Platform (distributed) do not affect performance, which is consistent with the original program (serial).
    - Including **high-energy muons**.

(measurement as function of muon p_T)



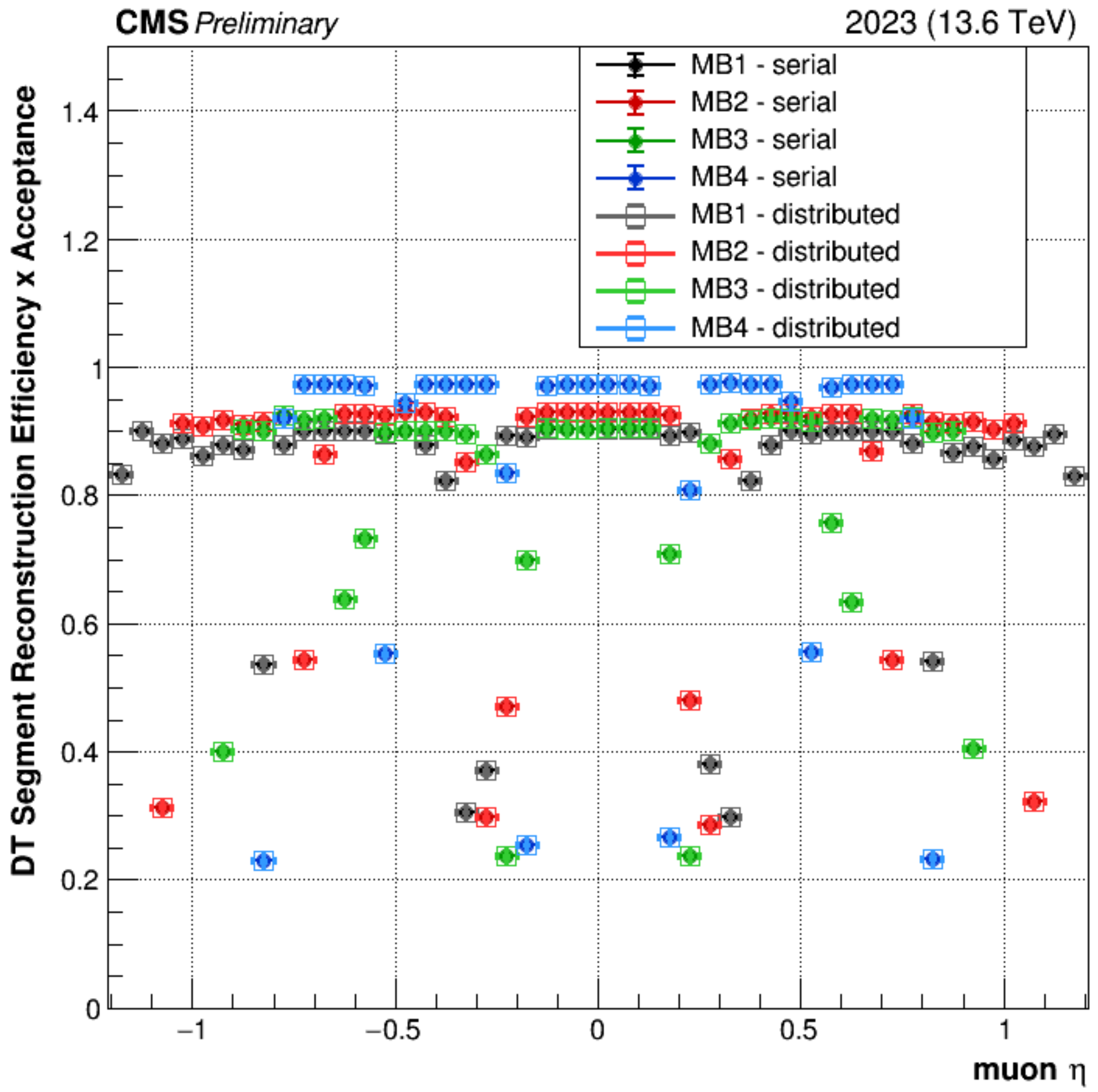
Use case: Detector Performance Analyses
Porting results

(efficiency x acceptance vs muon η)

- **The efficiency is computed**, this time, **without applying fiducial region selections** (convoluting reconstruction efficiency with detector acceptance) using the Tag-and-Probe method [CMS-DP-2023-049]

- The changes applied to the Tag-and-Probe program to run on the High Throughput Platform (distributed) do not affect performance, which is consistent with the original program (serial).

- Acceptance is the dominant effect observable in the plots:
    - Significant **efficiency drops** appearing in the boundaries between muon barrel wheels.
    - The impact of the cracks between barrel sectors varies among stations, explaining the differences in the regions where efficiency is maximal.



Use case: Detector Performance Analyses
Technical performance

- To evaluate the technical performance, the **available statistics has been processed 3 times**, mimicking an integrated luminosity of **~82fb^{-1}**, consisting of **~77M events** in total. Size: 224*3 = 672GB

- Serial processing (as a single job on HTCondor)

Wall time: ~120 minutes

1 CPU on a AMD EPYC 7302 16-Core Processor, with 2GB memory

- Distributed processing on the platform:

Wall time: ~6 minutes

Up to 92 CPUs (46 physical), on two AMD EPYC 7413 24-Core Processor, with 2GB memory per CPU. Resources hosted at T2_IT_LNL.

- The remote resources are monitored using in-site metrics, gathered and displayed using an **InfluxDB instance**.

- Quasi-interactivity is now reached:
  - Every time a re-execution of the analysis is needed (e.g. tweaking some thresholds or using different selection criteria), running a few Jupyter Notebook cells will do the trick (transparently accessing more resources)!!
  - This can result in a **great improvement** for any detector performance analysis application.

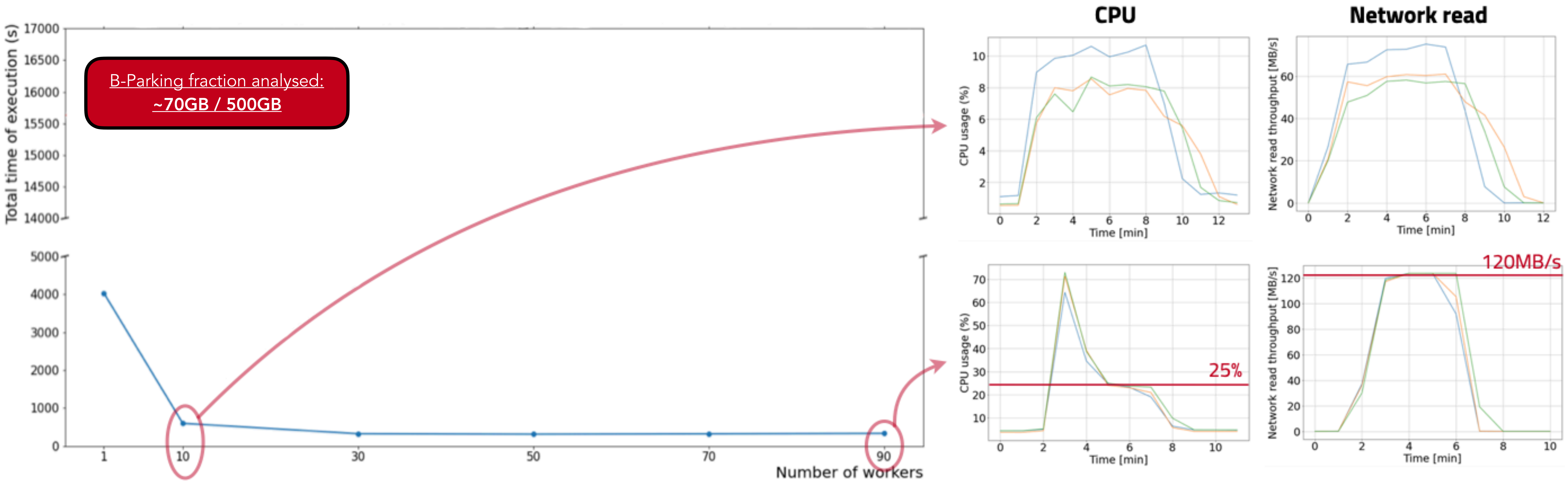
Use case: Physics data analysis
Technical performance

- The performances of the high-rate platform have been investigated also on a CMS physics data analysis.

Use case:

Analysis on a high amount of data, coming from the b-parking dataset gathered by CMS in 2018.

- The same analysis workflow, running on an increasing number of workers shows a decrease in execution time.



- As expected, low number of workers show a CPU usage saturation;
- For a high number of workers, network access becomes the main bottleneck (due to I/O access, via protocols like xRootd/WebDAV).

Moving towards a DataLake infrastructure

synergy with: ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing [1]

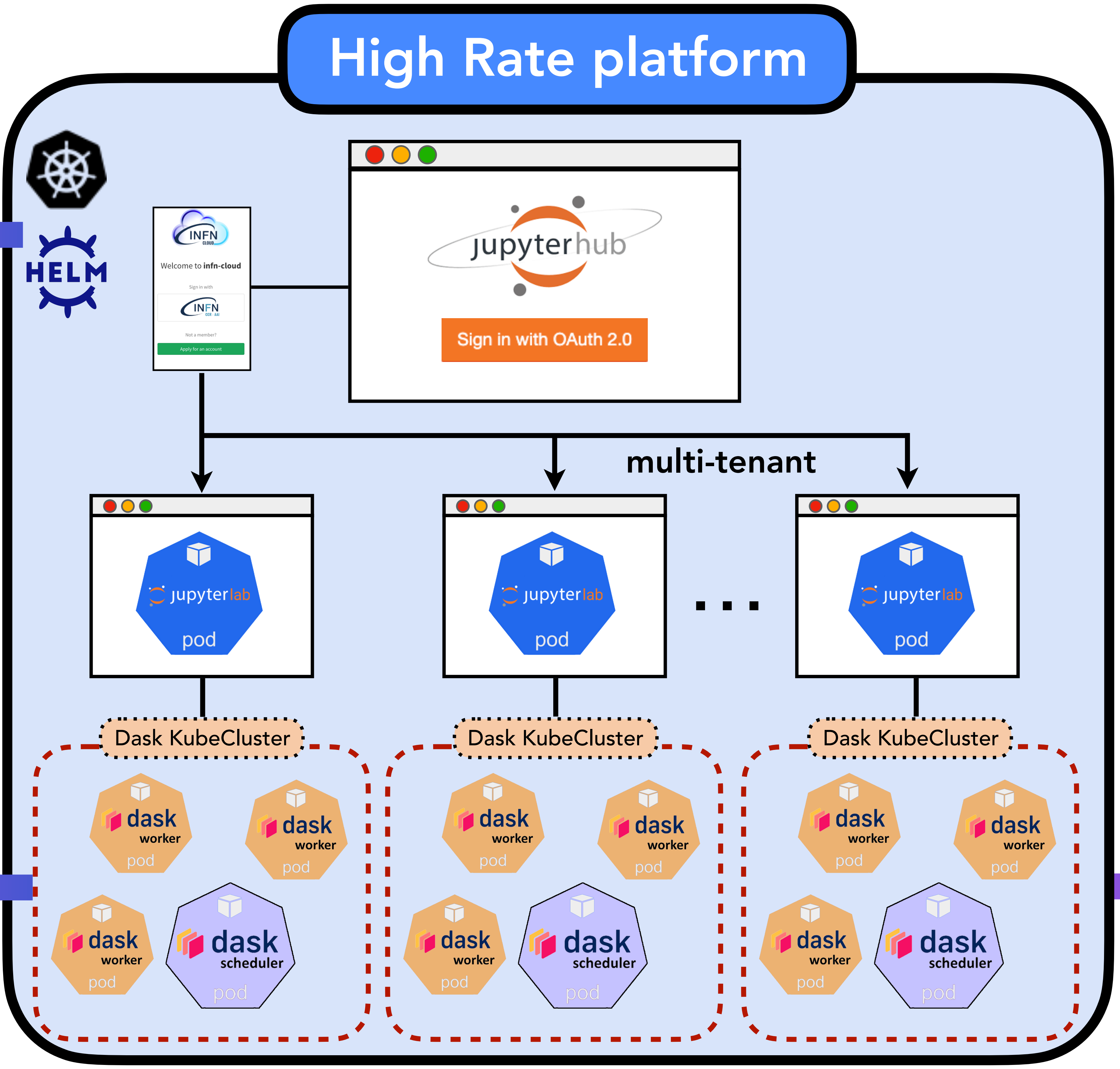
- Extending the entire infrastructure towards a multi-tenant platform, capable of intercepting the needs of data analysts from different scientific collaborations.

Deployment of the **Kubernetes** resources handled via **HELM charts**.

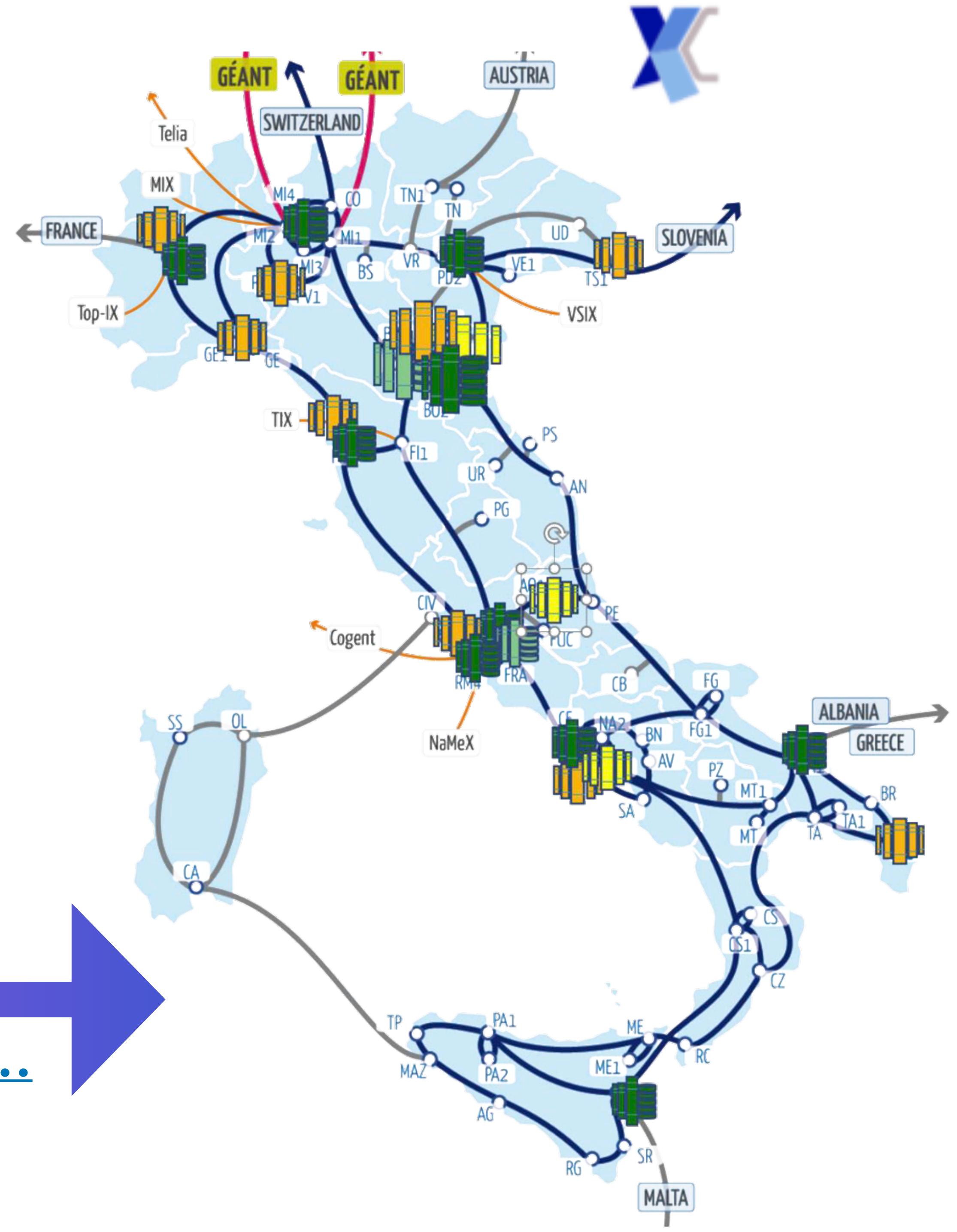
↓

Scalable deployment on the available resources

Also the **Dask cluster** is deployed on the K8s cluster, through the `KubeCluster` class, providing a Python API to manage the cluster.



offloading to...



[1] ICSC - Spoke 2 "Fundamental Research & Space Economy"

Conclusions

- In view of the R&D efforts for the next phases of LHC, new tools are required for making data analysis as efficient as possible;
- New **high-throughput platforms** have been developed:
    - Based on *interactive workflows* and *declarative programming* solutions;
    - Running on *distributed (and heterogeneous) resources*.
- Physics analysts already started the porting of their code, for testing and measuring the up-scaling performances:
- A **Detector Performance Group (DPG)** use case, coming from DT Tag-and-Probe analysis, has been successfully ported:
    - The changes applied to the source code are **not affecting performance**, and show an optimal consistency with the original analysis;
    - A **noticeable reduction in execution time** has been observed. In this way, analysts can re-run their applications multiple times (running on entire years of data-taking and/or performing multiple code changes).
- A performance speed-up of a physical data analysis has also been shown. For an increasing number of workers, the parallelisation of the workflow results in a faster execution. Some bottlenecks can be observed for massive I/O ops.

This work is partially supported by ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU.

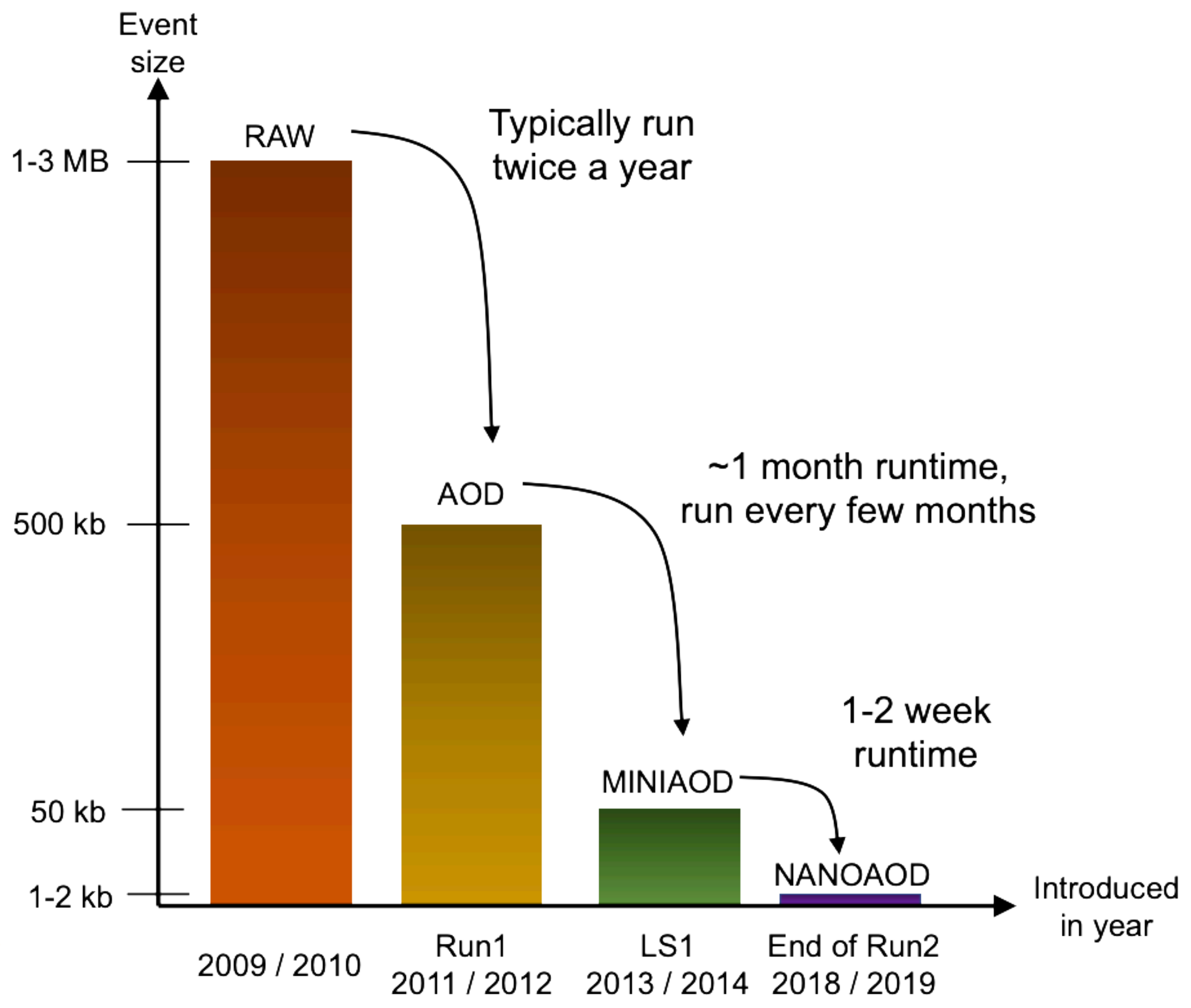
Finanziato dall'Unione europea NextGenerationEU | Ministero dell'Università e della Ricerca | Italiadomani | ICSC

BACKUP

Data processing in CMS

The entirety of CMS data, centrally produced, are saved in ROOT files, in different data formats:

- **RAW**: A collection of the detector electronics output. Event size: 1-3 MB
- **AOD**: Analysis Object Data. Transforming RAW data in analysis objects (used by analysts) like jets, muons, electrons, etc… Event size: 500kB
- **MiniAOD**: Reducing the size of AODs, making them more compact with the downside of losing some information (e.g. zeroing floating-point numbers bit). Event size: 50kB
- **NanoAOD**: Further reduction of MiniAODs, saved as a columnar ROOT file. This new format, using fundamental data types (int, float), exits from the CMS ecosystem and requires a small amount of dependencies to be analysed. Event size: 1-2 kB.



(image taken from: link to CHEP19 contribution)

DT Tag-and-Probe dataset:

Dataset used, based on muon DPG common NANO flavour: a NanoAOD-like dataset, with 410 physical variables, tailored for DPG-based analyses.

ROOT RDataFrame

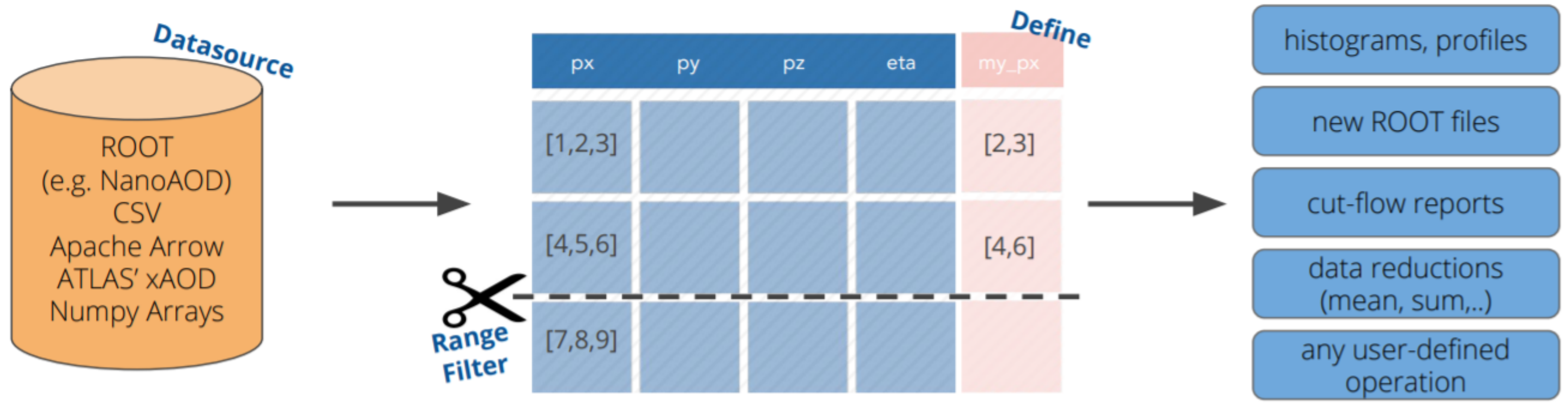
RDataFrame (RDF) is the high-level interface of ROOT for the data analysis saved in TTree, CSV and many other data formats. It is based on:

- multi-threading;
- low level optimisations (parallelism and caching).

Computations are expressed in terms of actions and transformations chain, constituting a computational graph.

The execution of such graph can be made also distributed, exploiting backends such as Dask and Spark.

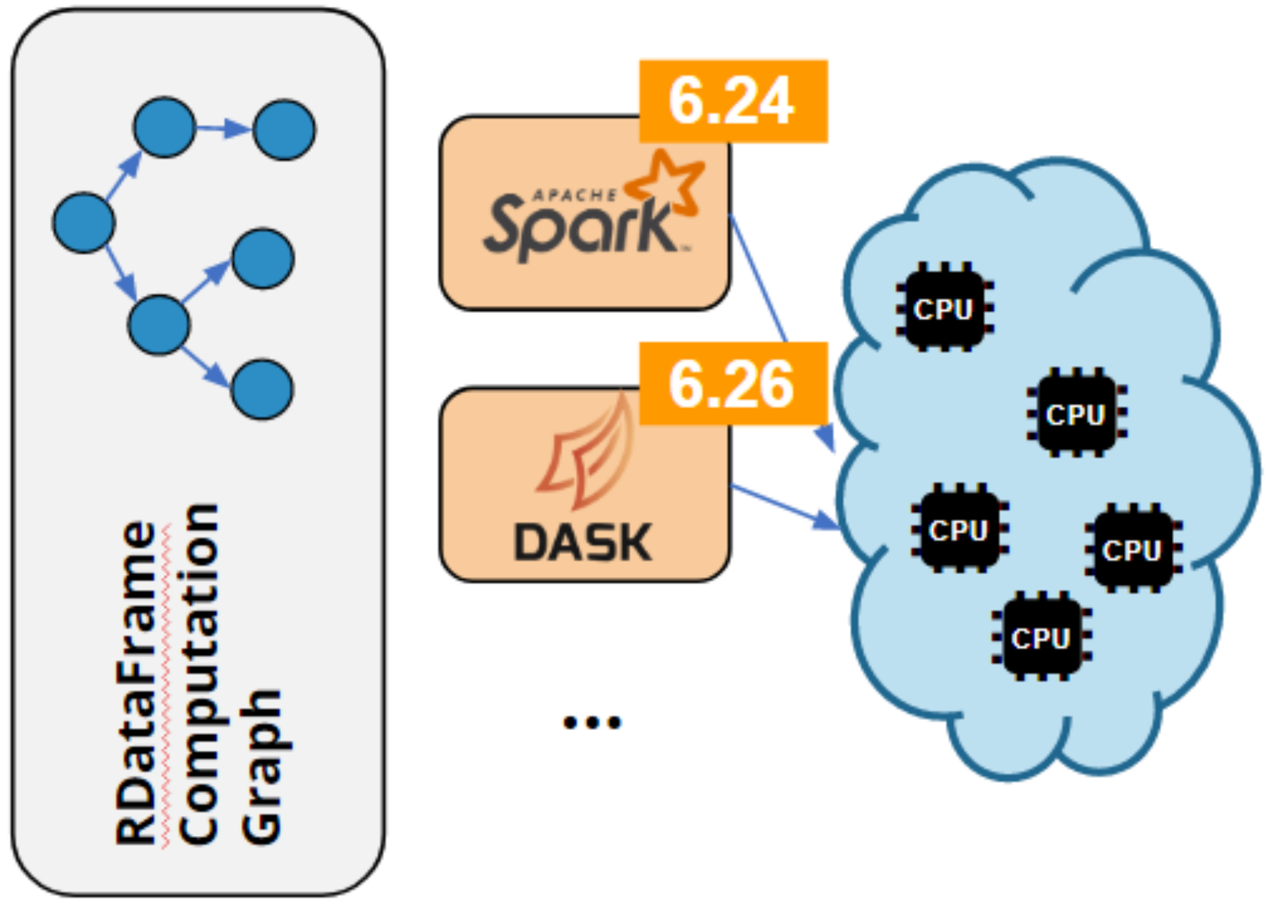
Thanks to the "distributed" extension of RDF, available experimentally.




```
# enable multi-threading
ROOT.EnableImplicitMT()
df = ROOT.RDataFrame(dataset)

df = df.Range(2)
       .Define("my_px", "px[eta > 0]")

# filled in a single loop
h1 = df.Histo1D("my_px", "w")
h2 = df.Histo1D("px", "w")
```

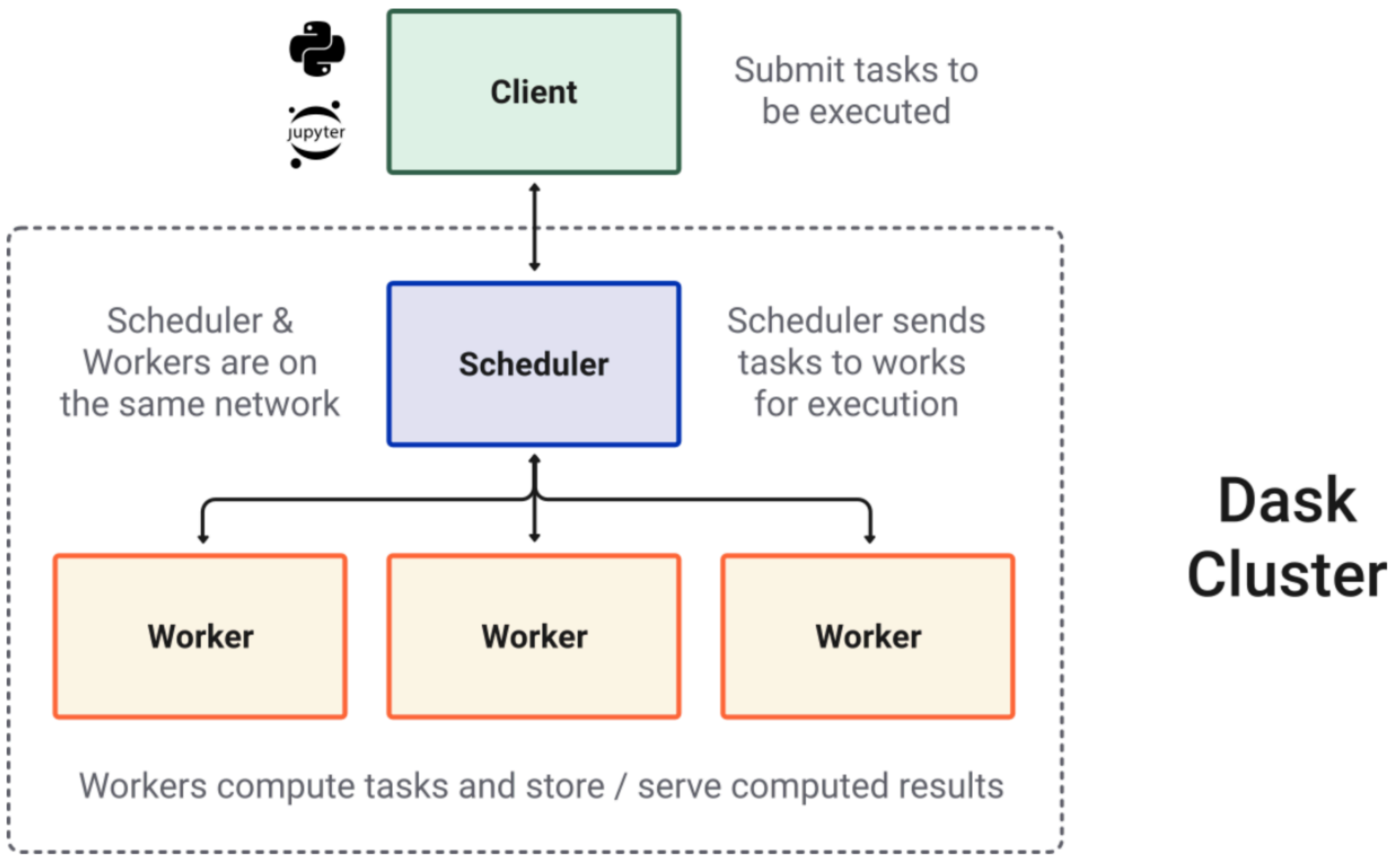


Images taken from: PyHEP 2021

Dask

There are many parts to the "Dask" cluster:

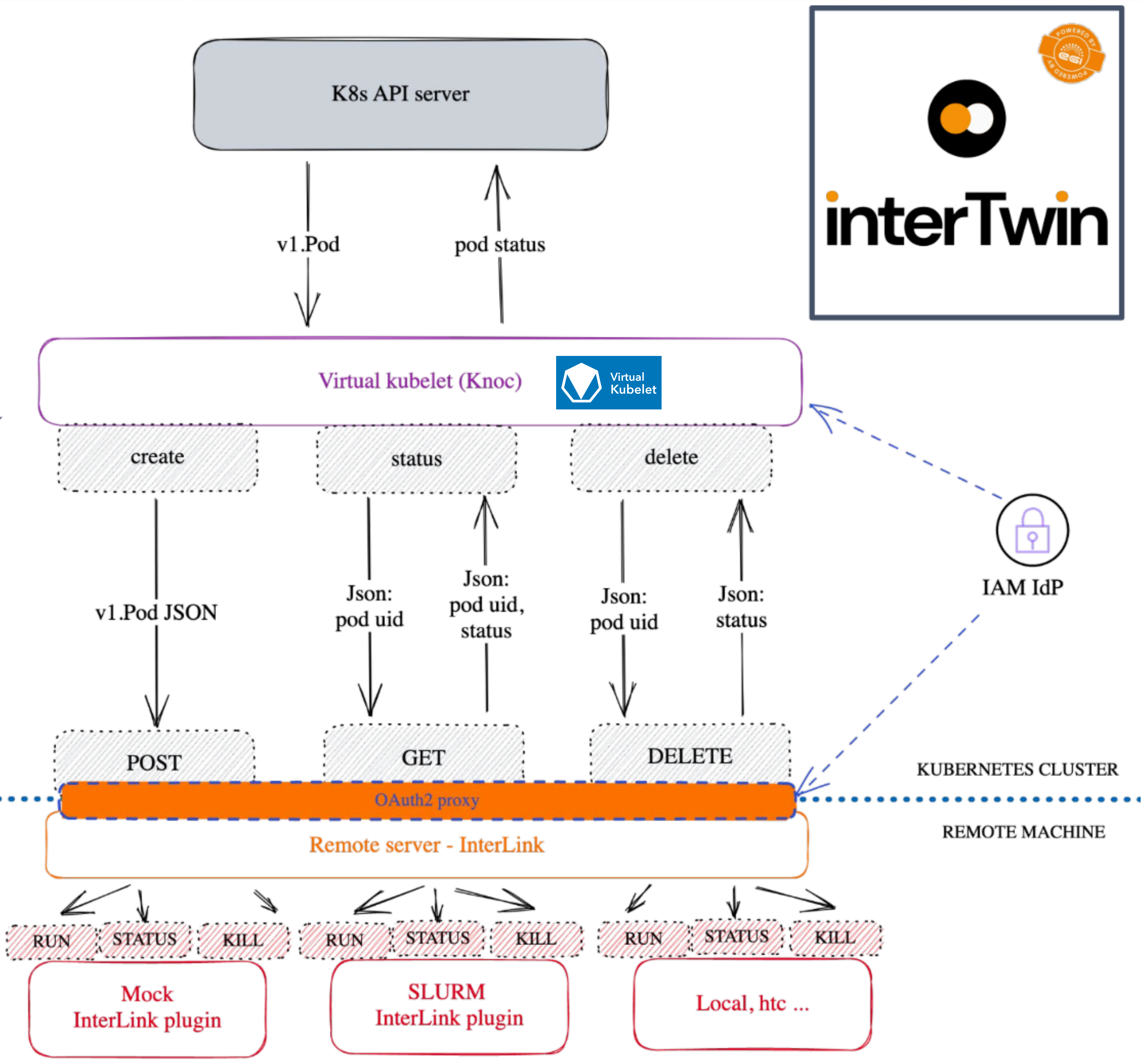
- Collections/API also known as "core-library".
- Distributed – to create clusters
- Integrations and broader ecosystem



The offloading strategy

Scheduling worker processes, spawning on multiple remote sites dynamically and transparently:

- Virtual Kubelet (open-source Kubernetes kubelet implementation that masquerades as a kubelet): registers as a virtual node and pulls work to run;
  - "It takes your pod and executes it wherever"
- InterLink + **HTCondor Sidecar (Plugin)**: pods are translated into HTCondor jobs:
  - Translating interlink create/status/delete calls interacting with the proper HTCondor schedd via CLI
    - `POST /create` call -> `condor_submit`
    - `GET /status` call -> `condor_q`
    - `POST /delete` call -> `condor_rm`
- In future, this strategy can be also applied to other job scheduling systems (e.g. Slurm/HPC)



DT local reconstruction efficiency

The DT efficiency to reconstruct a local track segment was defined and measured using a Tag & Probe method.

Events were selected to contain a pair of oppositely charged reconstructed muons.

Muon tracks were required to be well reconstructed in the tracker detector (≥ 6 hits in the strip detector and ≥ 1 hit in the pixel detector) and to be well isolated in η and ϕ from other tracks. Moreover the muon tracks were required to have a separation between each other $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2} > 0.3$.

- To ensure that they come from the same interaction vertex, their distance at the point of closest approach to the interaction point should be $\Delta z < 0.1$ cm.
- Their invariant mass should be within 10 GeV of the Z_0 mass.

The track used as tag is also required to be well reconstructed in the muon detector, by satisfying the Tight-ID criteria described in JINST 13 (2018) P06015. Furthermore, it is required to have a transverse momentum $p_T > 27$GeV and also to pass the High-Level Trigger selection of isolated muons with $p_T > 27$GeV.

The inner component of track used as probe is propagated inside-out to each station of the DT detector and must have segments matched in ≥ 2 muon stations different from the one under study. It also must have $p_T > 20$GeV.

A DT chamber crossed by a probe track is considered efficient if a reconstructed segment is found within 15 cm distance of the extrapolated track in the R-ϕ plane.

The DT Segment Reconstruction Efficiency can be computed:

- within the full solid angle, in this case it also includes detector acceptance
- within fiducial regions i.e. discarding probes that cross a chamber within 15 cm of its edges.