



Finanziato  
dall'Unione europea  
NextGenerationEU



Ministero  
dell'Università  
e della Ricerca



Italiadomani  
PIANO NAZIONALE  
DI RIPRESA E RESILIENZA



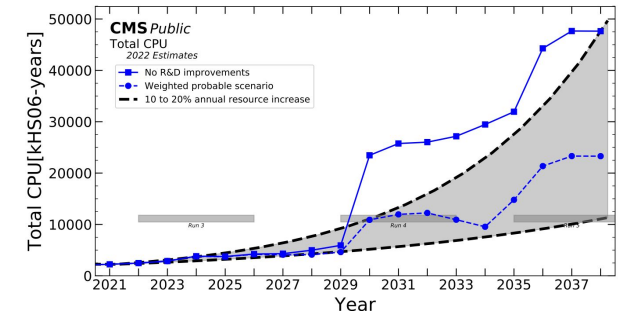
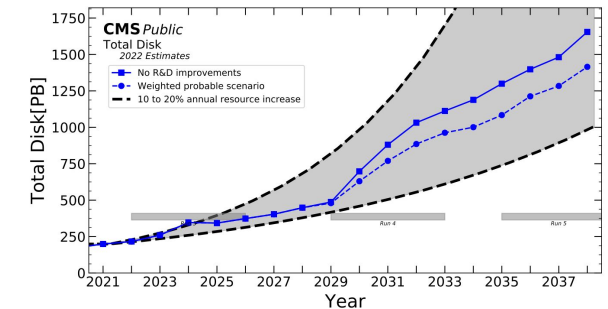
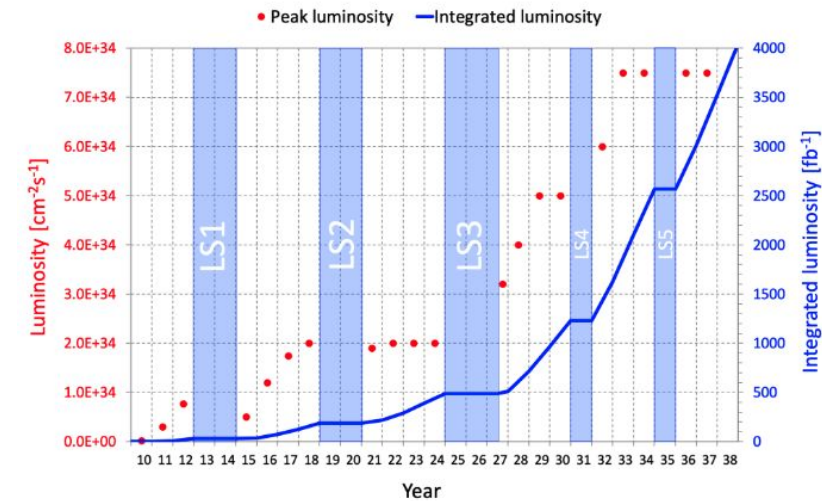
Centro Nazionale di Ricerca in HPC,  
Big Data and Quantum Computing

## Evolving High Rate Analysis infrastructure Tommaso Tedeschi (INFN Perugia)

Spoke2 INFN Meeting - 12 September 2024 - ZOOM

# The context

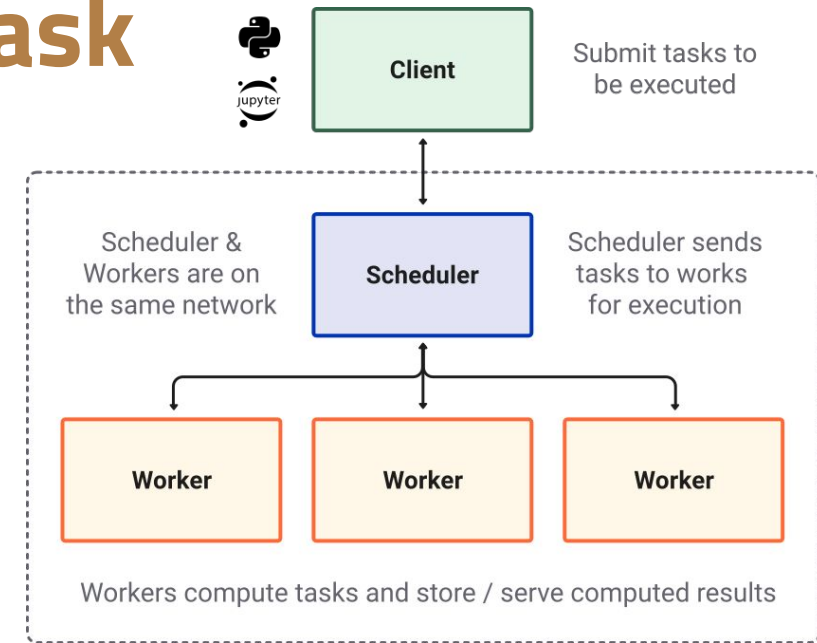
- Pushed by the enormous increase in HEP experiments computing resources requests from 2029/30 onwards (HL-LHC above all), a new analysis paradigm is arising:
  - High-rate, declarative, interactive or quasi-interactive data analysis approach**
    - enabled by the usage of slimmed (flat) data formats
    - based cutting-edge analysis tools (ROOT's RDataFrame, Coffea, ...) which scale up thanks to industry-standard data science backends (Dask)
- The development of infrastructural solutions (high rate platform) to implement such a new model is done inside WP2 and WP5:**
  - the infrastructure is developed using a use case-driven approach and tested with real-world analyses
  - and synergically with Spoke0:**
    - adopting/proposing infrastructural solutions



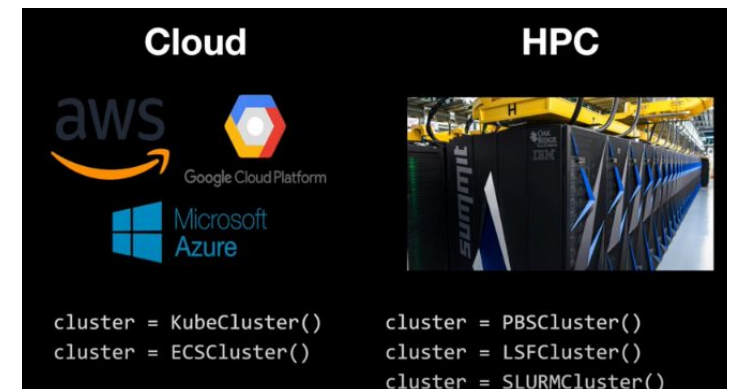


# Key technology enabler: Distributed Dask

- **Dask.distributed** is a centrally managed, distributed, dynamic task scheduler:
  - The central dask scheduler process coordinates
    - actions of several Dask worker processes on multiple machines
    - the concurrent requests of several clients
- Users interact by connecting a local Python session to the scheduler and submitting work
- Best to use a **cluster manager** utility class:
  - It deploys a scheduler and the necessary workers as determined by communicating with the resource manager
  - **Dask-jobqueue** is a set of cluster managers for job queueing systems
    - Supports PBS, Slurm, LSF, HTCondor, ...



**This obviously fits very well with the python analysis ecosystem, but also ROOT's RDataFrame can use Dask as backend: Dask therefore enables analysis on very different resource providers.. provided you get access to your data!**



# Spoke2 High Rate Platform

As Spoke2 WP5 we are ready, using ICSC resources as per RAC allocation

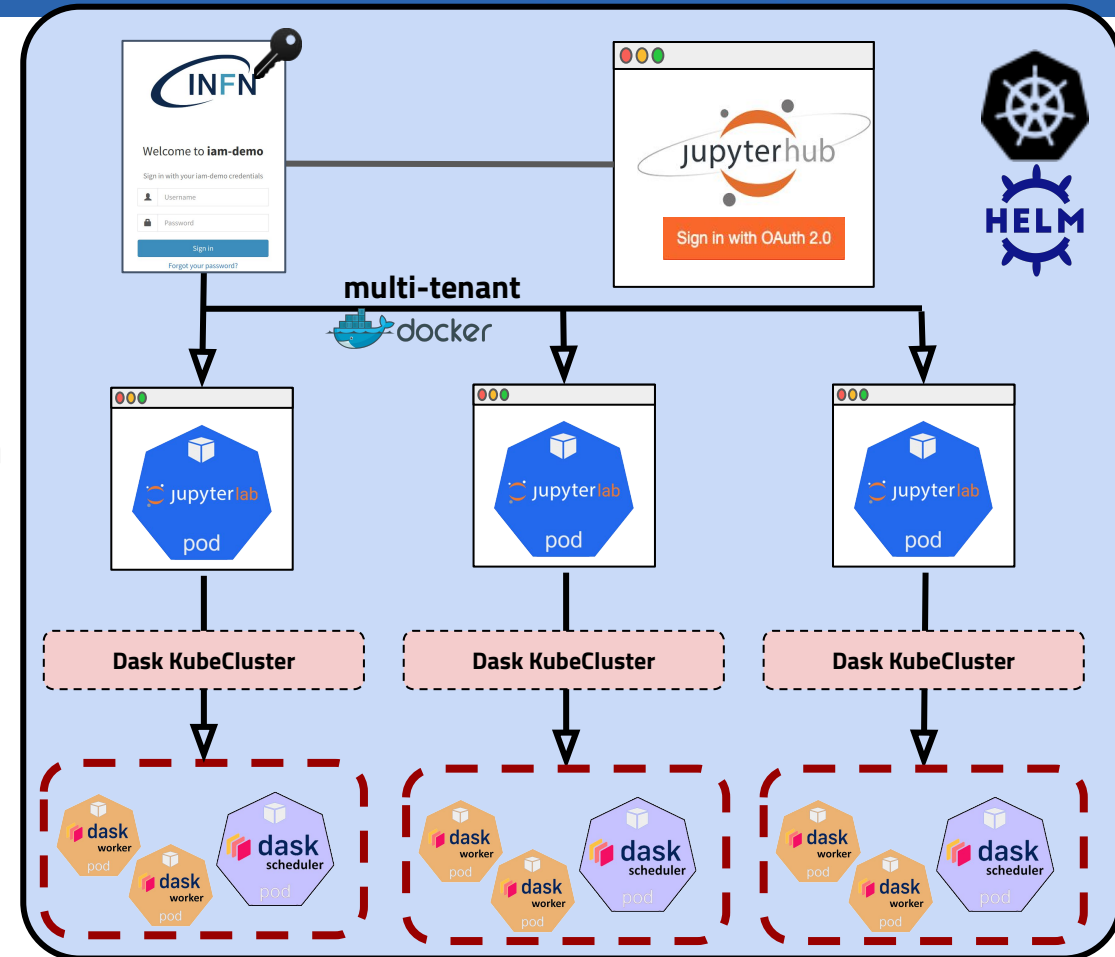
A jupyterhub is deployed on a k8s cluster (128 vCPUs and 258 GB) via the official [Spoke2 JHub Helm repo](#)

- endpoint is [here](#) and Indigo-IAM is used for authentication

Users choose JupyterLab image to deploy and after deployment completion, they get access to a full IDE (persistent storage, terminal, notebooks, editors)

Users can then scale up their python-based computation, using Dask library, **within the cluster that possibly scale within the provider:**

- this can be done via the Dask Jupyterlab extension which uses Dask KubeCluster under the hood
- Interface with Worldwide LHC Computing Grid storage sites (xrootd, WebDAV, ...)

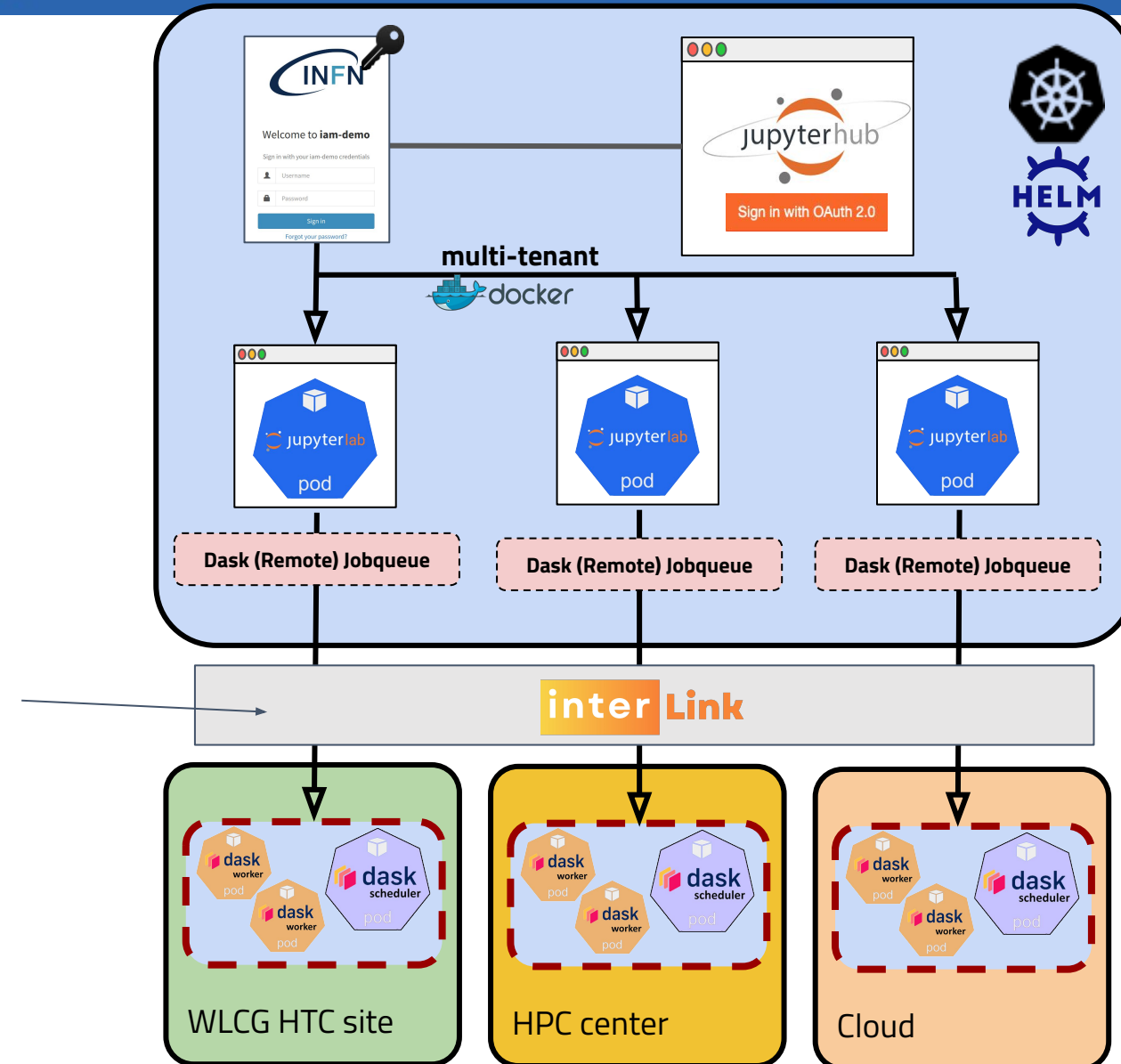


Potentially, a huge amount of users with diverse use cases may join.. how to scale up to external resources?

# Extending the Platform

Aim: enable the platform to **dynamically exploit all kinds of resources (HTC, HPC, Cloud)** transparently for the user

- looking for a synergy with active developments in this context, to delegate container execution on remote resources while keeping the very same user interface
- Possible solution: [InterLink](#), which provides execution of a Kubernetes pod on almost any remote resource
  - Resources visible to the user thanks to an HTCondor overlay



# Data are important: Rucio integration

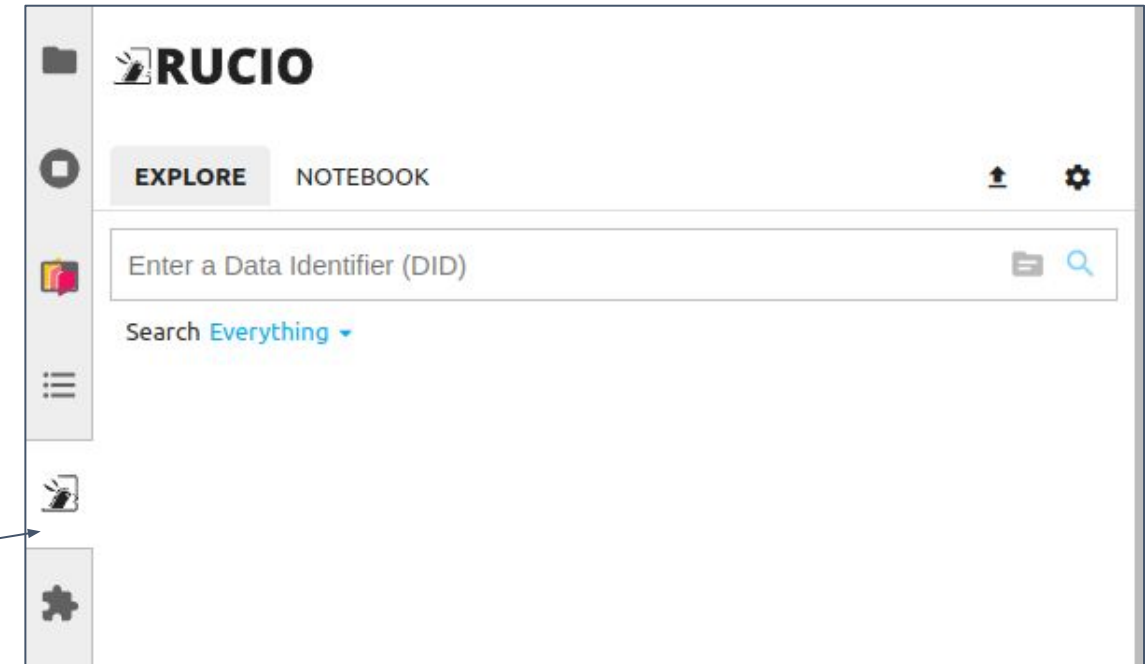
## High rate platform also integrates with Rucio:

- a project that provides services and associated libraries for allowing scientific collaborations to **manage large volumes of data spread across facilities** at multiple institutions and organisations.

Rucio client enables users to interact with the system and access the distributed data:

- **The client can upload, download, manage and delete** everything from single files up to Petabyte sized datasets.

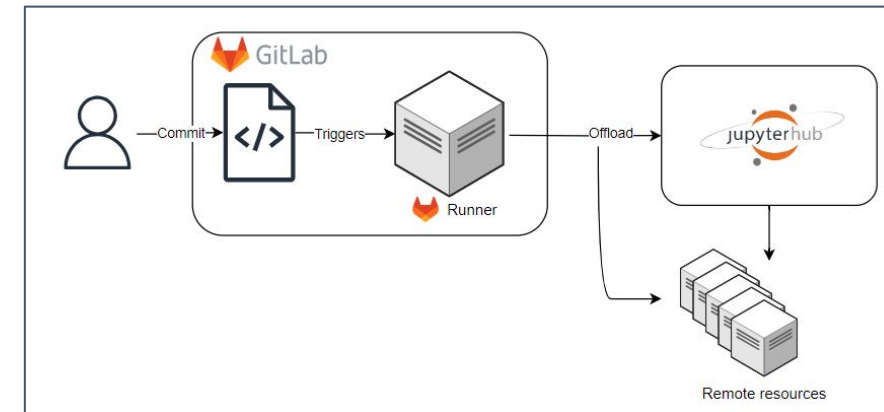
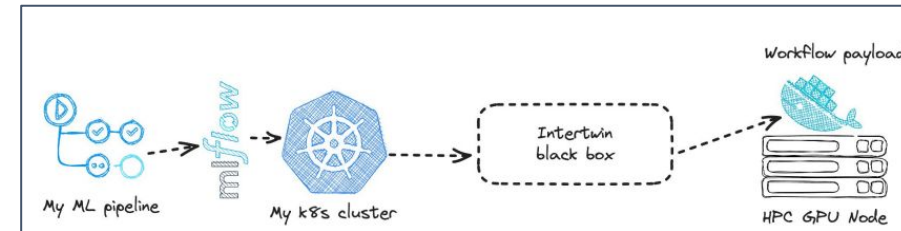
A **JupyterLab extension** integrates with Rucio to allow users to access some of Rucio's capabilities directly from the JupyterLab interface



# Enabling use cases

These are some use cases we have in mind as a scientific community:

- **Unlock full power of cutting-edge analysis tools**
  - Speed-up of factor  $O(10-100)$  for HEP analysis workflows
- **Easy GPU access:**
  - seamless access to HPC centers
  - ML training triggered via workflow automation, e.g. ML pipelining tools (Kubeflow, MLflow, ...)
  - many GPUs at once == more/faster hyperparameter optimization
- Enable **CI/CD as a trigger for analysis execution**
- ...

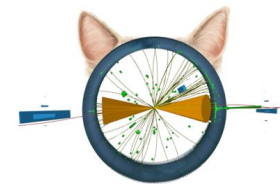
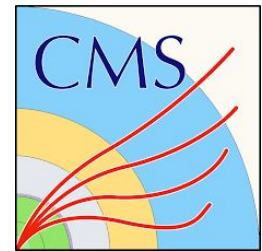




# High rate analysis in CMS



**Tools for achieving high throughput data analysis in the CMS experiment are discussed, developed and supported in the Common Analysis Tools (CAT) group**



- it was established in 2022 in order to deal with a multitude of diverging tools, providing support and documentation, while moving towards efficiency, interactivity, and reusability of analyses
- It is charged with two main tasks
  - Take ownership of the development, maintenance and documentation of analysis tools of common interest
  - provide a forum to discuss developments of new analysis tools, offering guidance




# High rate analysis in CMS

CAT through its DPROC and WFLOWS subgroups is progressively taking ownership of already-existing **analysis frameworks** which are based on

- cutting edge tools:
  - **ROOT's RDataFrame (RDF)** - modern, high-level interface for analysis of data stored in TTree, CSV and other data formats, in C++ or Python
  - **HSF's Coffea** (or bare awkward-array) - makes use of uproot and awkward-array to provide an array-based syntax for manipulating HEP event data in an efficient and numpythonic way
- **NanoAOD** data format, most reduced CMS data format

This ensures declarativeness, efficiency and (quasi-)interactivity of analysis, reducing time-to-insight

In collaboration with O&C, CAT represents the software needs of physics analyses in the context of the development of **analysis facilities** and other innovative computing infrastructures



|                              |
|------------------------------|
| bamboo                       |
| CMSJMECalculators            |
| CROWN                        |
| mkShapesRDF (a.k.a. Latinos) |
| PocketCoffea                 |
| columnflow                   |
| DasAnalysisSystem            |
| HiggsDNA                     |



Taken  
from CAT  
official  
docs

In addition to CERN interactive logon services such as the [LXPLUS service \(Linux Public Logon User Service\)](#) and the [SWAN \(Service for Web based Analysis\)](#) platform, other CMS institutions also provide access to computing resources by providing so-called Analysis Facilities to users with a CERN account that is associated with CMS. These are summarised here.

- [Coffea Casa](#)
- [Purdue Analysis Facility](#)
- [INFN Analysis Facility](#)

# 1st benchmark: 2017 SSWW VBS with tau with RDF

Further Performance gain when  
offloading to production grid site

The feasibility of the interactive approach has been **tested and benchmarked on the CMS INFN Analysis Facility**

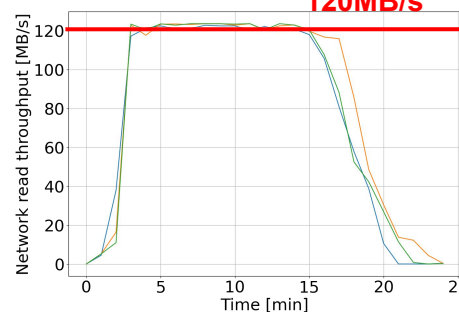
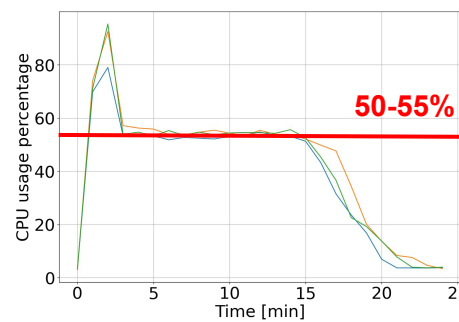
- The analysis with CMS detector of the scattering (VBS) of two same-sign W bosons decaying to a hadronic tau and a light lepton was taken as benchmark to test this approach:

$$q\bar{q} \rightarrow W^{\pm}W^{\pm}q\bar{q} \rightarrow \tau_h^{\pm}\nu_{\tau}\ell^{\pm}\nu_{\ell}q\bar{q}$$

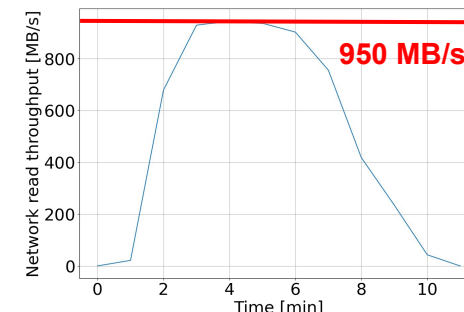
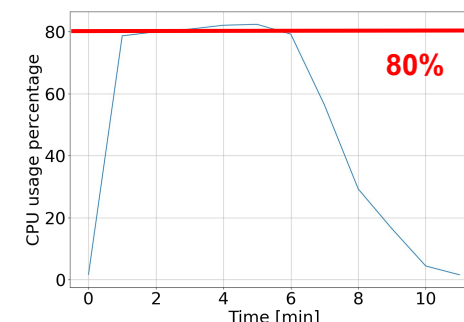
The physics analysis was converted from a legacy iterative approach to the modern declarative approach offered

**A gain of one order of magnitude** in terms of time with respect to a batch-like approach

Three T2\_LNL\_PD non-production  
nodes with 1 Gbit/s connection  
Total 96 cpus (48 physical)



One T2\_LNL\_PD production node with  
10 Gbit/s connection  
Total 96 cpus (48 physical)



<https://www.sciencedirect.com/science/article/pii/S0010465523003107>

<https://cds.cern.ch/record/2867989>

# More tests: 2018 Z- $\mu\mu$ with PocketCoffea

First tests with a Coffea-based analysis:

- 2018 Z- $\mu\mu$  using PocketCoffea configuration layer
- Very preliminary** performance results:
  - CPU usage around 80/90% at Production Tier2 Legnaro
  - Network throughput at ca. 500MB/s
  - UNDER INVESTIGATION**
- Next steps:
  - test a more computation-intensive benchmark
  - replicate on ICSC Spoke2 high rate platform:
    - exploit an HPC bubble with nvme storage



Joint effort with Davide Valsecchi (thanks!)

| Category    | Events     | Throughput (events/s) |
|-------------|------------|-----------------------|
| Total       | 1180932962 | 3335597.84            |
| Skipped     | 822423975  | 2322973.21            |
| Preselected | 69283566   | 195694.52             |

<https://github.com/PocketCoffea/AnalysisConfigs/tree/main/configs/zmumu>



# What do I plan to do in addition on the ICSC High Rate Platform?



# Run3 polarization in VBS with mkShapesRDF

The measurements of the **longitudinally polarized scattering** of the W and Z bosons provide complementary information to direct measurements of the Higgs boson couplings to gauge bosons:

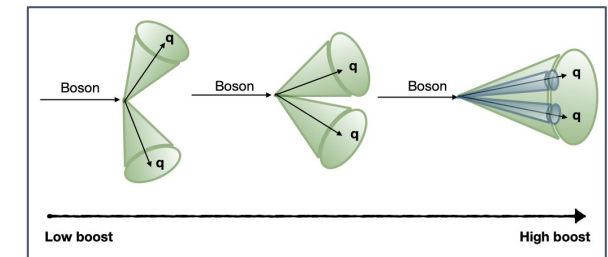
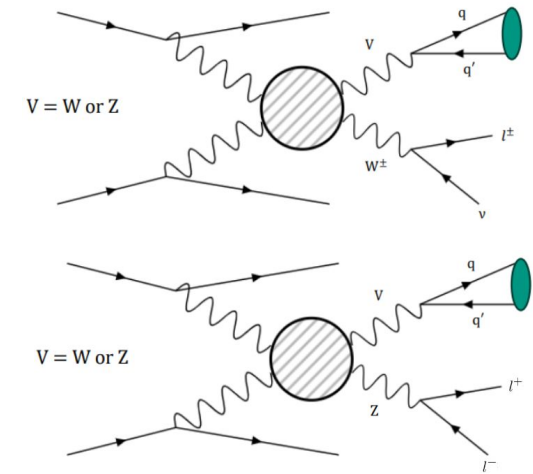
- Boson polarization can be measured as **angular distributions** of particles produced in the decay process

Plan to run the analysis in WZ and ZV semileptonic decays (and fully-hadronic if possible/needed) using the mkShapesRDF framework (based on RDataFrame)

- Benefit from the **large hadronic branching fraction** of W or Z boson

## ParticleNet for fatjets

- flavour/boson tagging:
  - ParticleNet ~2x bkg rejection wrt prev. tagger (DeepAK8)
- Jet charge tagging (Distinguishing between boosted  $W^+$ ,  $W^-$ , Z):
- even develop own algorithm to boost sensitivity to specific polarization states



# Conclusions and next steps

High throughput analysis is becoming a must due to the forthcoming HEP experiments resource needs:

- New paradigm based on declarativeness and efficiency: quasi-interactivity
- Python (ROOT's RDataFrame and Coffea) and Dask as the main players (but not the only ones)

ICSC Spoke2 High rate analysis platform is up and running

- Capability to offload to heterogeneous resources will be introduced soon

CMS is pushing towards interactive analysis:

- INFN Analysis Facility already tested with RDataFrame and Coffea-based analyses, more tests to come

More VBS polarization studies to come to be run on the ICSC2 High rate analysis platform

# Backup



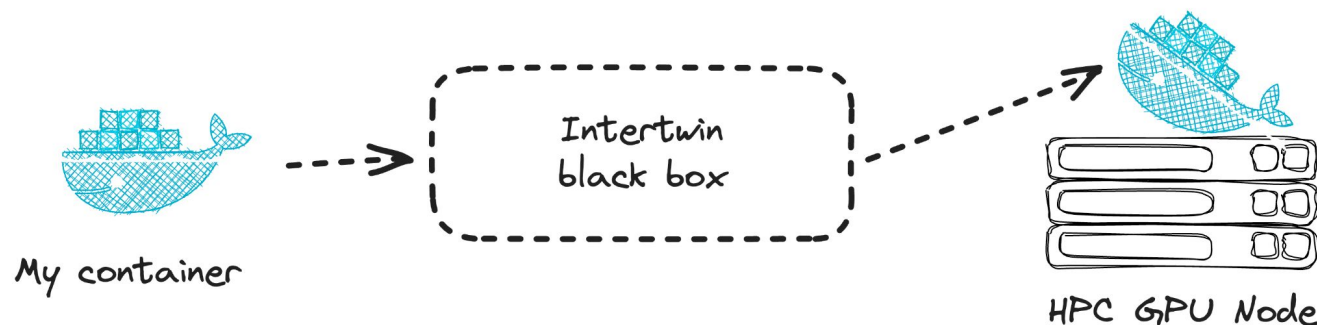
# High rate platform details

- **Access and security:** After connecting to an endpoint URL, the user reaches a Jupyterhub [2] instance that, after authentication and authorization via INDIGO-IAM [3], 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)
- **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 [7] with multi-threading)
- **Deployment:** The deployment of the Kubernetes [4] resources needed for the spawning of this platform, is handled via HELM [5] charts available in the GitHub organization [6]. This allows a seamless, flexible, scalable and fault-tolerant deployment on the available resources, with a limited impact on the admin's work time



# What is offloading?

- Delegate the execution of a container/workflow on remote resources while keeping the user interface unchanged.
- Example:
  - "I have my own ML training container, and I want to run it on a node with 4 A100s"



# How can we implement offloading?

We want a NATIVE integration with the Kubernetes primitives, acting underneath as a virtual node.

N.B. We aim to use Kubernetes as the workhorse for the “offloading”, NOT as the user interface though



Kelsey Hightower  
@kelseyhightower

The problem is we asked developers to do all that. Kubernetes is not a tool for developers. They can use it, but we have to be honest, Kubernetes is low level infrastructure and works best when people don't know it's there.

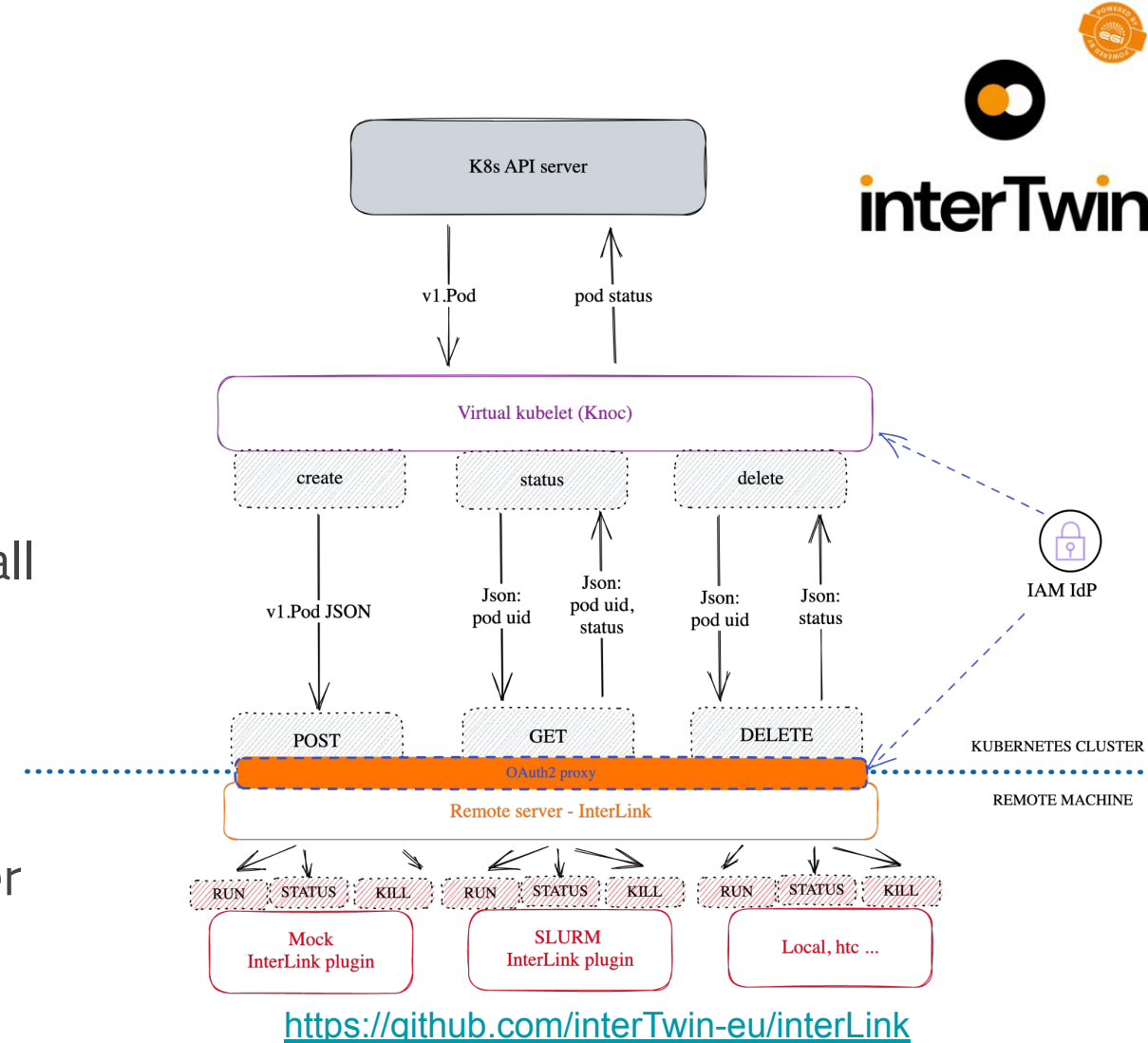
Offloading should be transparent for the users

# A possible solution: InterLink

**InterLink** aims to provide an abstraction for the execution of a Kubernetes pod on any remote resource capable of managing a container execution lifecycle.

The project consists of two main components:

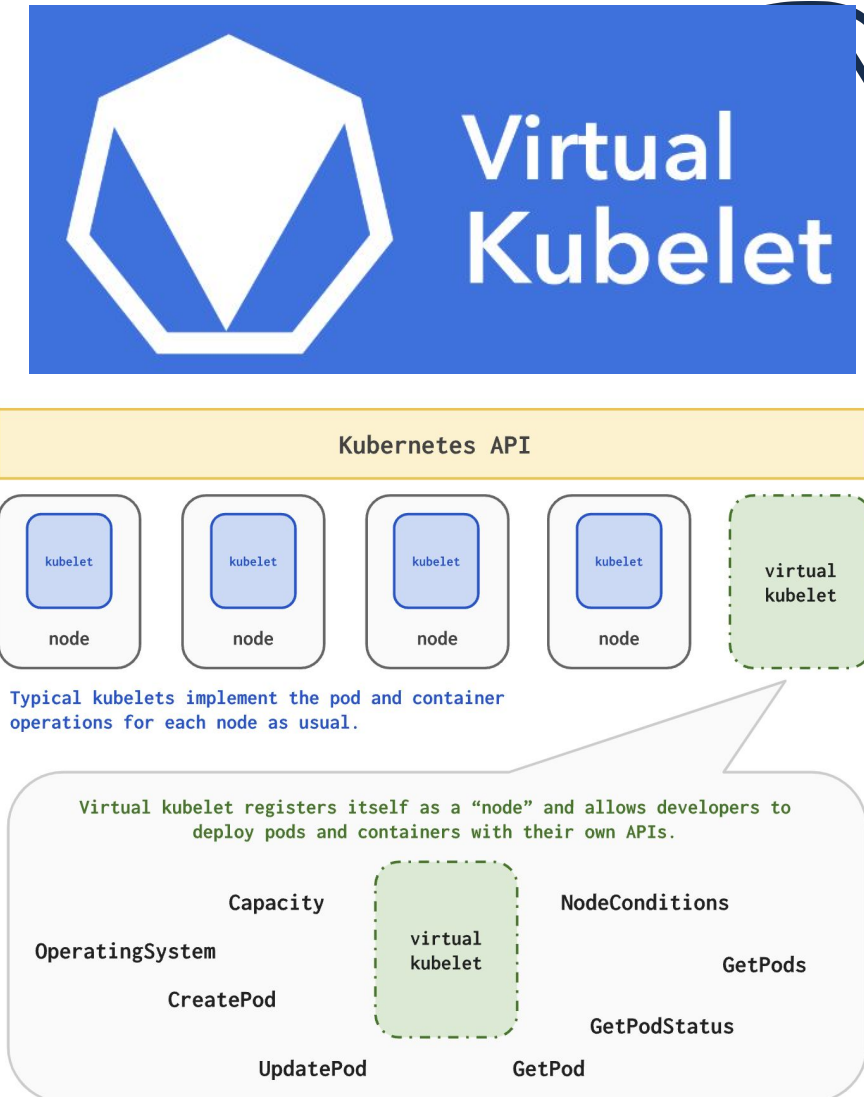
- **A Kubernetes Virtual Node:** based on the VirtualKubelet technology. Translating request for a kubernetes pod execution into a remote call to the interLink API server.
- **The interLink API server:** a modular and pluggable REST server where you can create your own container manager plugin (called sidecar), or use the existing ones: remote docker execution on a remote host, singularity Container on a remote SLURM or **HTCondor batch system**, etc...



# Components: VK

<https://virtual-kubelet.io/>

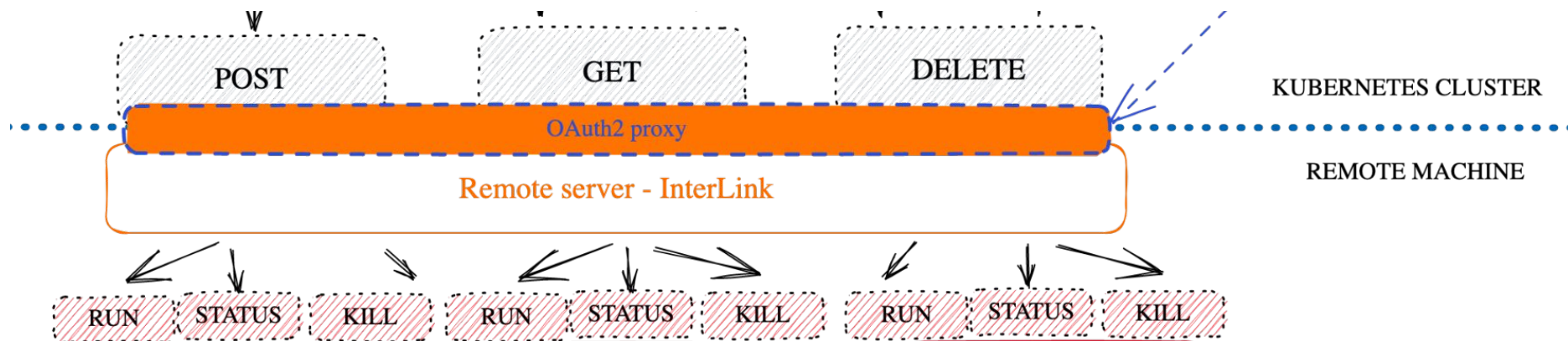
- **Virtual kubelet (VK):**
  - “Open-source Kubernetes kubelet implementation that masquerades as a kubelet. This allows Kubernetes nodes to be backed by Virtual Kubelet providers”
- Can be imagined as a translation layer:
  - “I take your pod and run your container wherever I want”
- Registers virtual node and pulls work to run
- The pod lifecycle is managed via interlink rest calls
- OAuth2 via service token kept “refreshed”





# Components: Interlink + OAuth2 proxy

- OAuth2 proxy: authN with IAM and authZ configurable on aud and groups
- "Digests" and manipulates calls from VK to the sidecar
- Self contained binary, distributable on all OS without dependencies



# Components: Sidecar/Plugin

- Agent that must expose a REST with defined specs, but which can be implemented in the language and with the methods you prefer:
  - creation of the pod: run local docker or submit a job on htc, slurm etc
  - collect the execution states
  - collect and forward logs upon request
  - kill
- Existing plugins: local Docker (Go), Slurm (Go), HTCondor (python), ARC (python), Kubernetes (python), Kueue (python)



# InterLink: development context and ICSC related activities

The technical solution (interLink) has been initially prototyped by INFN in the context of the interTwin EU Funded project and is now enhanced within the ICSC development/research programme.

In particular

- **It is part of the Spoke0** infrastructural toolkits. As such it is under consolidation, testing and improvement
- **It is part of the Spoke2 - WP5 work plan**
  - in this respect there is a ongoing integration effort to extend the High rate analysis platform over HTC/HPC computing resources
- **Also part of the Spoke3 integration plan**
  - idea is to benefit of the interLink capabilities to offer highly dynamic access to specialized HW (i.e. over Leonardo)
  - integrating offloading with data retrieval from the data-lake prototype

Many fruitful synergies should lead toward a generic technical solution, versatile and extendible based on specific needs.

# CAT group substructure

- CAT organization includes three subgroups:
  - **Data Processing Tools (DPROC)**
    - support, management, and development of tools running directly on the CMS centrally-produced datasets
  - **Workflow Orchestration and Analysis Preservation (WFLOWS)**
    - support, management, and development of tools for the orchestration of physics analysis workflows, promoting tools that ease the long-term reproducibility of analyses
  - **Statistical Interpretation Tools (STATS)**
    - support, management and development of statistical interpretation tools (most importantly of Combine, the RooStats / RooFit - based software tool used for statistical analysis within CMS)



# CMS INFN AF analysis

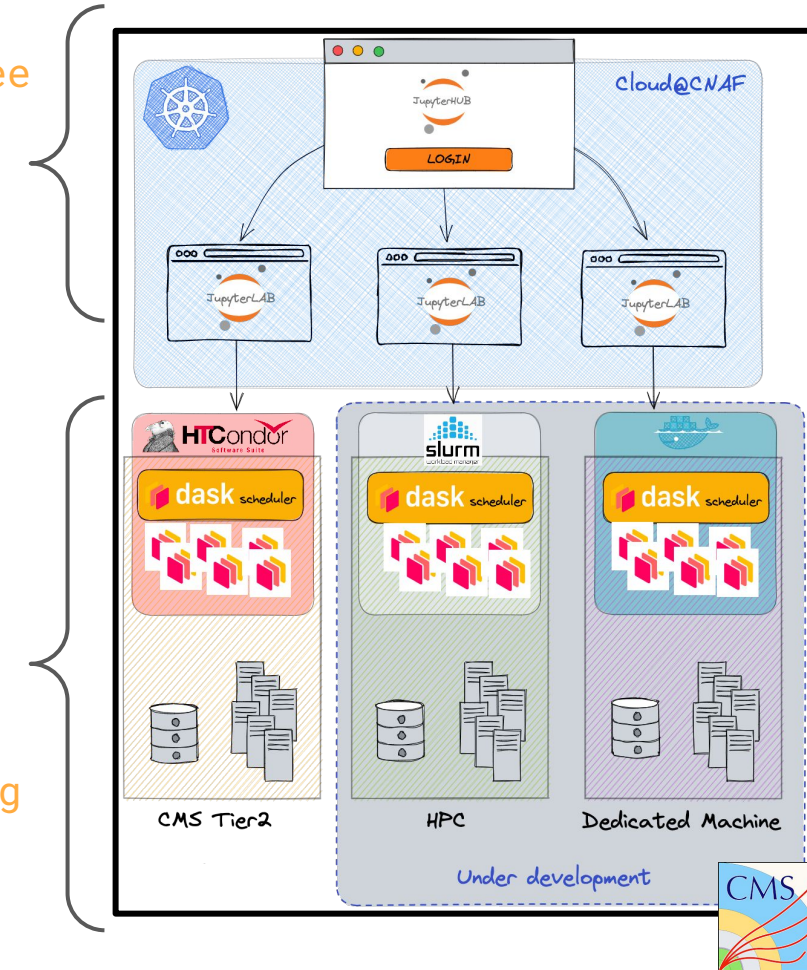


A growing project (activity started in 2020):

- Now 28 registered users: at least 4/5 of them can be considered "very active" users, authenticated via CMS IAM
- Containerized Jupyterlab environment:
  - Both "a-la-batch" and interactive processing allowed
- Integration of heterogeneous resources under the same pool:
  - Existing WLCG infrastructure and batch-systems for interactive use used for both legacy and interactive processing:
  - Using an HTCondor overlay and Dask in HTCondor mode
- offload on all Italian Tier2 sites via Interlink mechanism:
  - Deployment of Dask clusters on remote resources via RemoteHTCondor (Dask-jobqueue plugin)

What  
users see

What the  
offloading  
hides to  
the user



# Distributing the workload

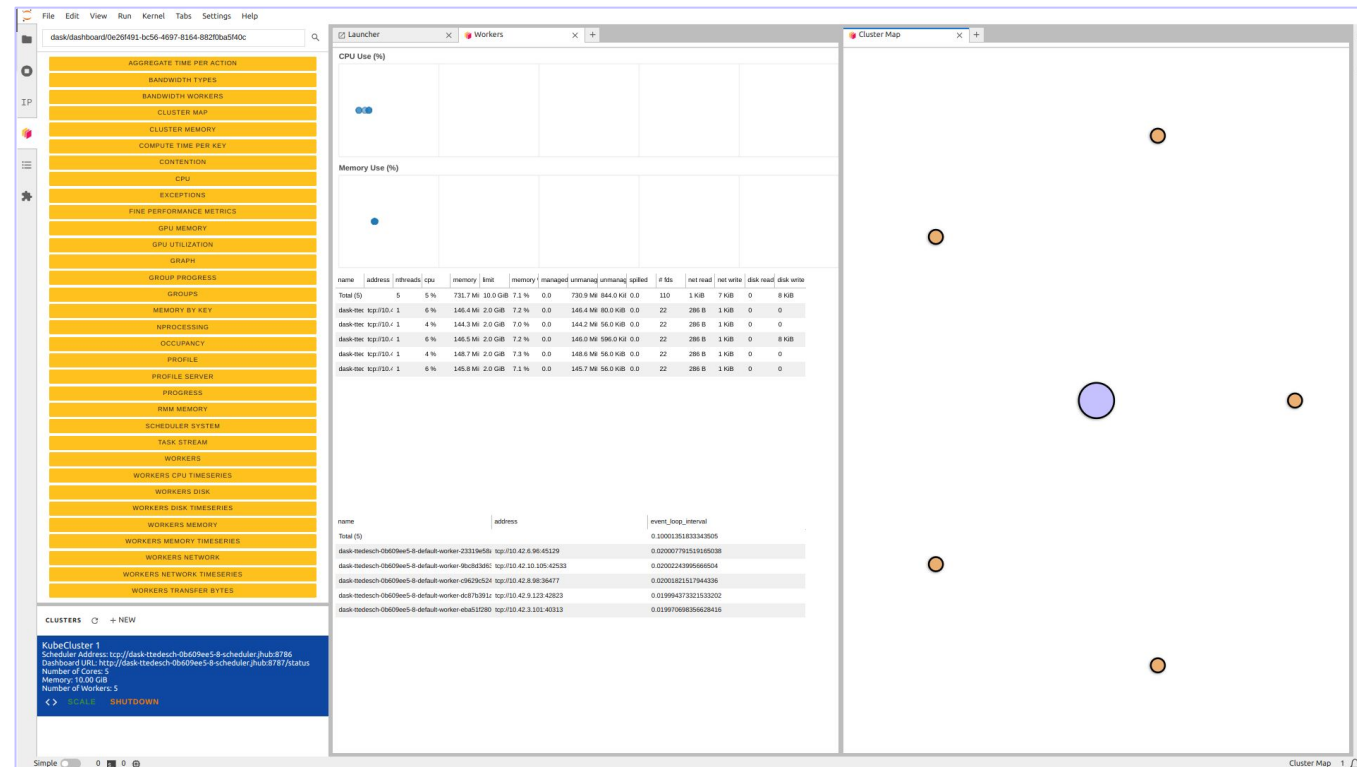
The Dask cluster is deployed on the Kubernetes (K8s) cluster using the Dask Operator (a service that runs on your K8s cluster and allows to create and manage Dask clusters as K8s resources) through `dask_kubernetes.operator.KubeCluster` class, which provides a simple Python API to manage the cluster and allowing maximum flexibility for the end-user.

The deployment of such cluster can be done:

either via the Dask Labextension (which implements a convenient GUI to create, scale and delete Dask clusters)

or via CLI/notebook cell (this allows to better customize your cluster, choosing images, scheduler and workers resource requests, etc...).

In both cases, the user needs to instantiate a `dask.distributed Client` object to interact with the scheduler and start the computation.



The **Dask Labextension** plugin allows to interact with the Dask dashboard directly in the Jupyterlab session, getting access to useful monitoring panels.

# Pocketcoffea run

The screenshot displays the Pocketcoffea application interface. On the left is a sidebar with a file explorer and a list of system metrics. The main area is divided into a terminal window showing command output and a progress/worker status panel on the right.

**File Explorer (Left Sidebar):**

- CONTENTION
- CPU
- EXCEPTIONS
- FINE PERFORMANCE METRICS
- GPU MEMORY
- GPU UTILIZATION
- GRAPH
- GROUP PROGRESS
- GROUPS
- LOGS
- MEMORY BY KEY
- NPROCESSING
- OCCUPANCY
- PROFILE
- PROFILE SERVER
- PROGRESS
- RMM MEMORY
- SCHEDULER SYSTEM
- TASK STREAM
- WORKERS
- WORKERS CPU TIMESERIES
- WORKERS DISK
- WORKERS DISK TIMESERIES

**Terminal Window (Main Area):**

```

- N. datasets: 5
-- Dataset: DATA_SingleMuon_2018_EraA, Sample: DATA_SingleMuon, N. file
s: 92, N. events: 241608232
-- Dataset: DATA_SingleMuon_2018_EraB, Sample: DATA_SingleMuon, N. file
s: 51, N. events: 119918017
-- Dataset: DATA_SingleMuon_2018_EraC, Sample: DATA_SingleMuon, N. file
s: 56, N. events: 109986009
-- Dataset: DATA_SingleMuon_2018_EraD, Sample: DATA_SingleMuon, N. file
s: 194, N. events: 513909894
-- Dataset: DYJetsToLL_M-50_2018, Sample: DYJetsToLL, N. files: 204, N.
events: 195510810
- Subsamples:
-- Sample DATA_SingleMuon: StandardSelection ['DATA_SingleMuon'], (1 cat
egories)
-- Sample DYJetsToLL: StandardSelection ['DYJetsToLL'], (1 categories)
- Skim: ['nPVGgood_1', 'event_flags', 'golden_json_lumi', 'nMuon_min1_pt18
.0', 'HLT_trigger_SingleMuon']
- Preselection: ['dilepton']
- Categories: StandardSelection ['baseline'], (1 categories)
- Variables: ['MuonGood_eta_1', 'MuonGood_pt_1', 'MuonGood_phi_1', 'nEle
ctronGood',
'nMuonGood', 'nJets', 'nBJets', 'JetGood_eta_1', 'JetGood_pt_1',
'JetGood_phi_1', 'JetGood_btagDeepFlavB_1', 'JetGood_btagDeepFlavCvL_1',
'JetGood_btagDeepFlavCvB_1', 'JetGood_eta_2', 'JetGood_pt_2', 'JetGood_phi
_2',
'JetGood_btagDeepFlavB_2', 'JetGood_btagDeepFlavCvL_2',
'JetGood_btagDeepFlavCvB_2', 'mll']
- Columns: {'DATA_SingleMuon': {'baseline': []}, 'DYJetsToLL': {'baseline
': []}}
- available weights variations: {'DATA_SingleMuon': ['nominal'],
'DYJetsToLL': ['pileup', 'sf_mu_id', 'sf_mu_iso', 'nominal']}
- available shape variations: {'DATA_SingleMuon': [], 'DYJetsToLL': []}
Running with executor dask infn-af
[INFO ] Working on samples: ['DATA_SingleMuon_2018_EraA', 'DATA_SingleMu
on_2018_EraB', 'DATA_SingleMuon_2018_EraC', 'DATA_SingleMuon_2018_EraD', 'D
YJetsToLL_M-50_2018']
[### ] | 8% Completed | 3min 26.9s
  
```

**Progress Panel (Right):**

Progress -- total: 8314, waiting: 385, queued: 7041, processing: 180, in-memory: 148, no

| Task            | Progress   |
|-----------------|------------|
| ZmumuBasePro... | 679 / 7897 |
| reduce          | 28 / 416   |
| lambda          | 1 / 1      |

**Workers Panel (Right):**

CPU Use (%)

Memory Use (%)

| name    | address | nthreads | cpu    | memd limit | memc mana | unma  | unma   | spilled | # fds | net re | net w | disk r | disk w |
|---------|---------|----------|--------|------------|-----------|-------|--------|---------|-------|--------|-------|--------|--------|
| Total ( | 90      | 33 %     | 63.9 C | 180.0      | 35.5 %    | 185.8 | 57.7 C | 6.0 G   | 0.0   | 9080   | 16 Gi | 171 M  | 0      |

name address event\_loop\_interval

Total (90) 2.37074370573973

Workers 1