Multi site Analysis Facility model

CMS experience at INFN

D. Spiga, on behalf of the
INFN R&D team

spiga@pg.infn.it

Outline

- Model objectives
- Implementation
- Supported use cases
- Resource provisioning point of view
- Current focuses and priorities

Contributors

- Diego Ciangottini
- Daniele Spiga
- Mirco Tracolli
- Tommaso Boccali
- Massimo Biasotto
- Massimo Sgaravatto
- Stefano Nicotri
- Francesco Failla
-

Also thanks to ROOT/SWAN developers

- Enrico Guiraud
- Enric Tejedor
- Vincenzo Padulano

Recap: intro, background and main motivations

1. To develop solutions for **a new analysis model.** From event looping to declarative analysis, interactive execution…
2. offering handles for a possible (smooth) transition/ adoption
3. Reducing analysis “time to insight” … I’d like to process **billion of events in O(hours)**
4. Increasing the system delivered **throughput (evts/s)**, which optimises resource usage
5. No to reinvent the wheel! On the one hand **use (all) available resources** and on the other hand to reuse and integrate (industry std) solutions
6. Do measurements (**benchmarking**) to quantify where is the gain, if any, to understand what to do where (and when)
7. Analysis **framework agnostic**

TODAY: The focus of this presentation is mostly on showing the computing model and resource exploitation under study

a. many technical details underneath, but not the purpose of this talk!

Pillars

The scenario we have in mind:

- Provide a distributed system compatible with **Python ecosystem**
  - Exploitable via DASK
- **Single entrypoint** exposed to the user
  - **Aside note**: this is a example of where we benefit of **INFN-Cloud** already today
- Federating under the hood an **heterogeneous set of resources**
  - Current Tier2s
  - Dedicated fat node
  - HPC/opportunistic
- Support **interactive + quasi interactive + batch workflows**

… meaning…

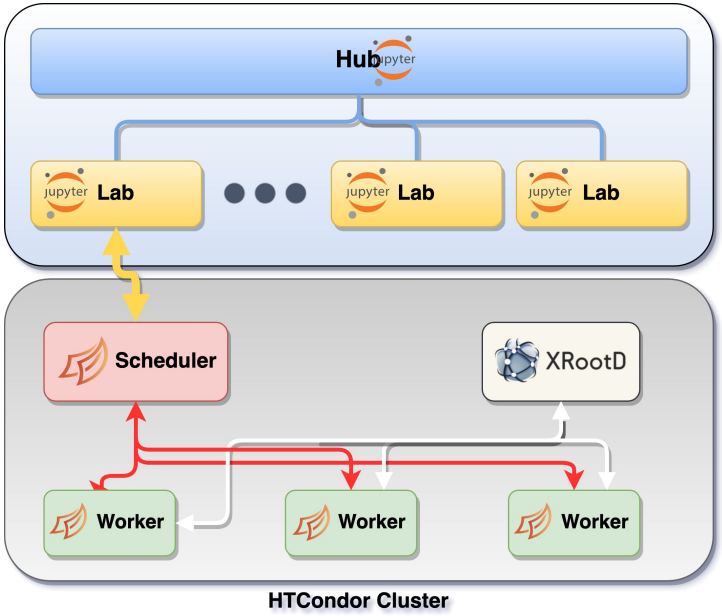
- Keep the **requirements for resource provider very low!**
  - No open ports/public ips
  - Auto healing (as much as possible)
- User friendly **interfaces**
  - Not only jupyterlab, e.g. integration with DM tools and with Dask
    - Both from CLI and webUI
- A **federation layer capable of dynamically stretch** over new resources.
  - Cloud - HTC - HPC …
  - NOTE: we are looking forward for optimizing the resources.. Work is atomically allocated

The overall idea, from where we started

Integration work of well established technologies

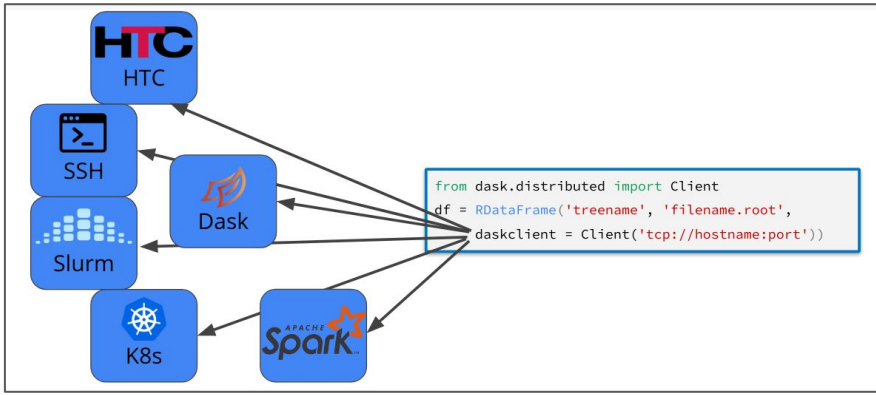
- **JupyterHub** (JHub) and **JupyterLab** (JLab) to manage the **user-facing part of the infrastructure**
- **DASK** to introduce the **scaling over a batch system**
- **XRootD as data access protocol** toward AAA:
  - Here we foresee the usage of caching layers (see later)

So far we opted for scaling over HTCondor:
⇒ User prioritization and in general configuration tuning is under study



What's the main point?

- We have **high level frameworks capable of leverage distributed computing engine**: RDataFrame and Coffea
- We can **scale/offload it over distributed resources through DASK + batch system** (e.g. HTCondor)



Can we use WLCG infrastructure in this kind of approach? At least to offload the most intensive computation?

- The **mantra**: Use and reuse what we have already today

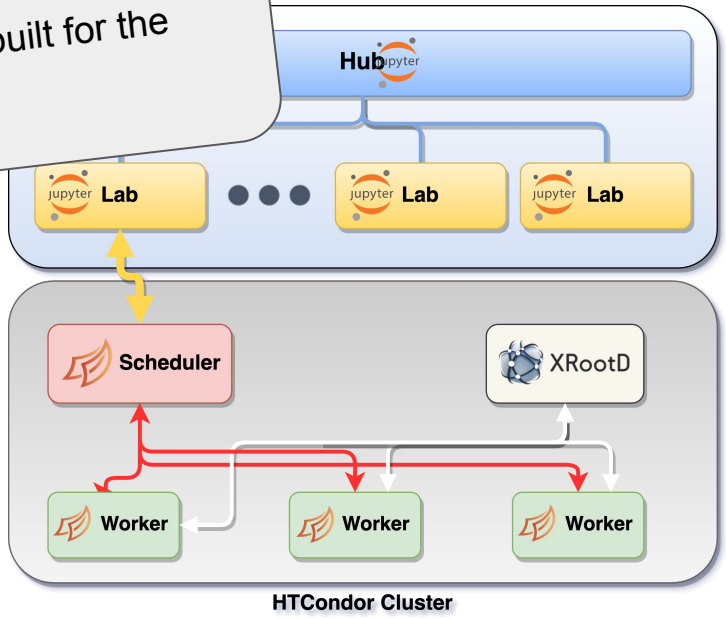
The overall idea

Integrati…
te…

- …
- **DASK** to introduce the **scaling over a batch system**
- **XRootD as data access protocol** toward AAA:
  - Here we foresee the usage of caching layers (see later)

A question we made to ourselves:
"Can we leverage the HTCondor experience we have built for the GRID and build something integrated with Dask?"

So far we opted for scaling over HTCondor:
⇒ User prioritization and in general configuration tuning is under study



The overall idea

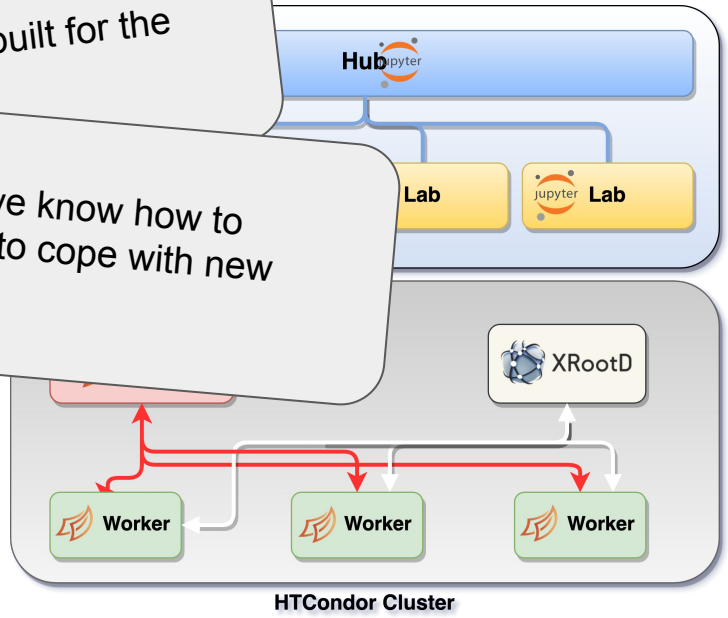
Integrati
te

- **batch syst**
- **XRootD as data access protocol**
  toward AAA:
  - Here we foresee the usage of caching layers (see later)

A que
"Can
GRID

...de to ourselves:
...er experience we have built for the
...Pack?"

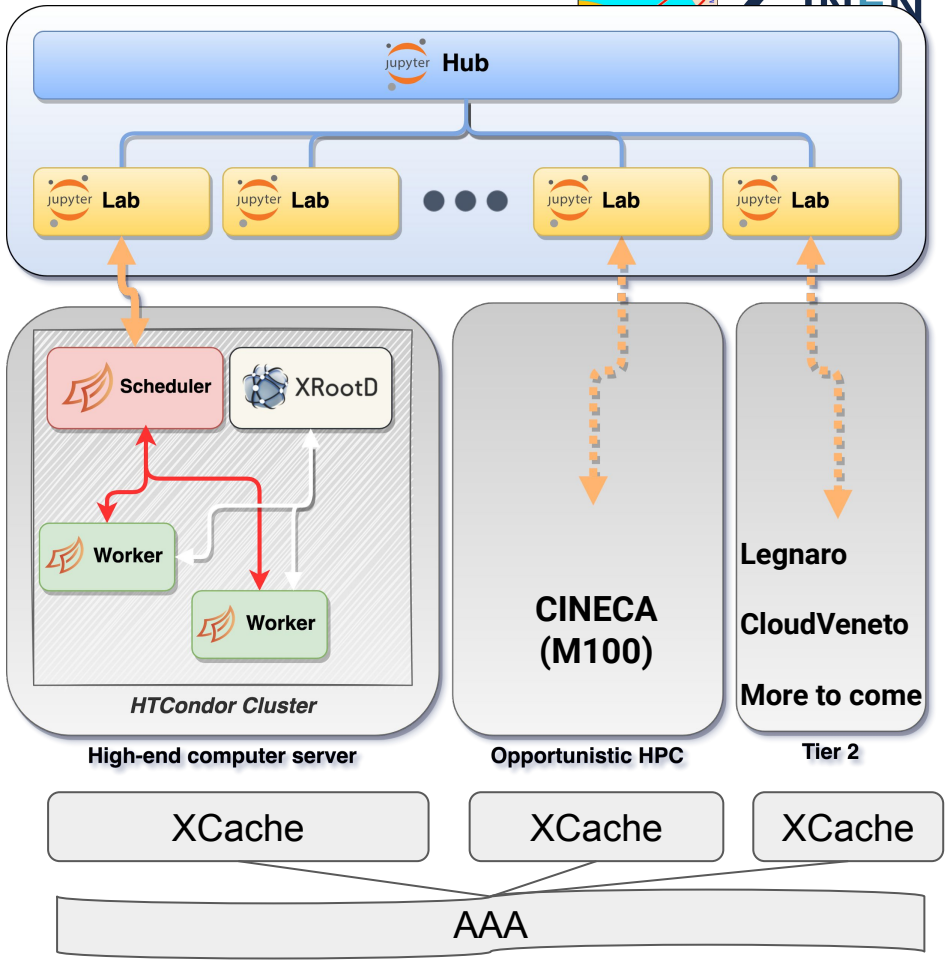
That will provide us with an infrastructure that we know how to manage and extend, but at the same time able to cope with new analysis paradigm!



So far we opted for scaling over HTCondor:
⇒ User prioritization and in general configuration tuning is under study

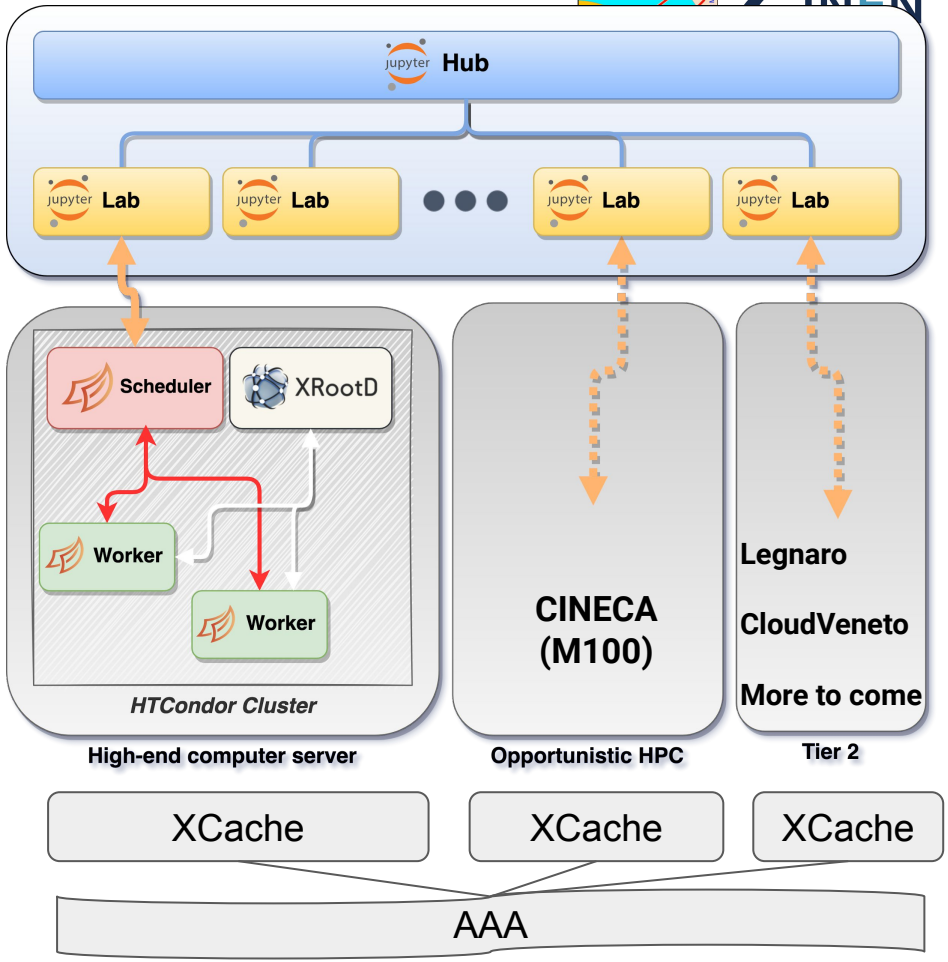
INFN testbed for future analyses at CMS

- a **testbed setup to provide a playground** for the design of a future analysis infrastructure
  - Leveraging **state of the art software toolsets**
  - Develop locally than scale out and make **use of already-available/spare resources**
- **Already demonstrated the functionality via a real CMS analyses workflow "ported" into RDataFrame**



INFN testbed for future analyses

- User will login via web and directed to a personal notebook area with a minimum amount of resource for local exploration
  - Develop locally than scale out and make **use of already-available/spare resources**

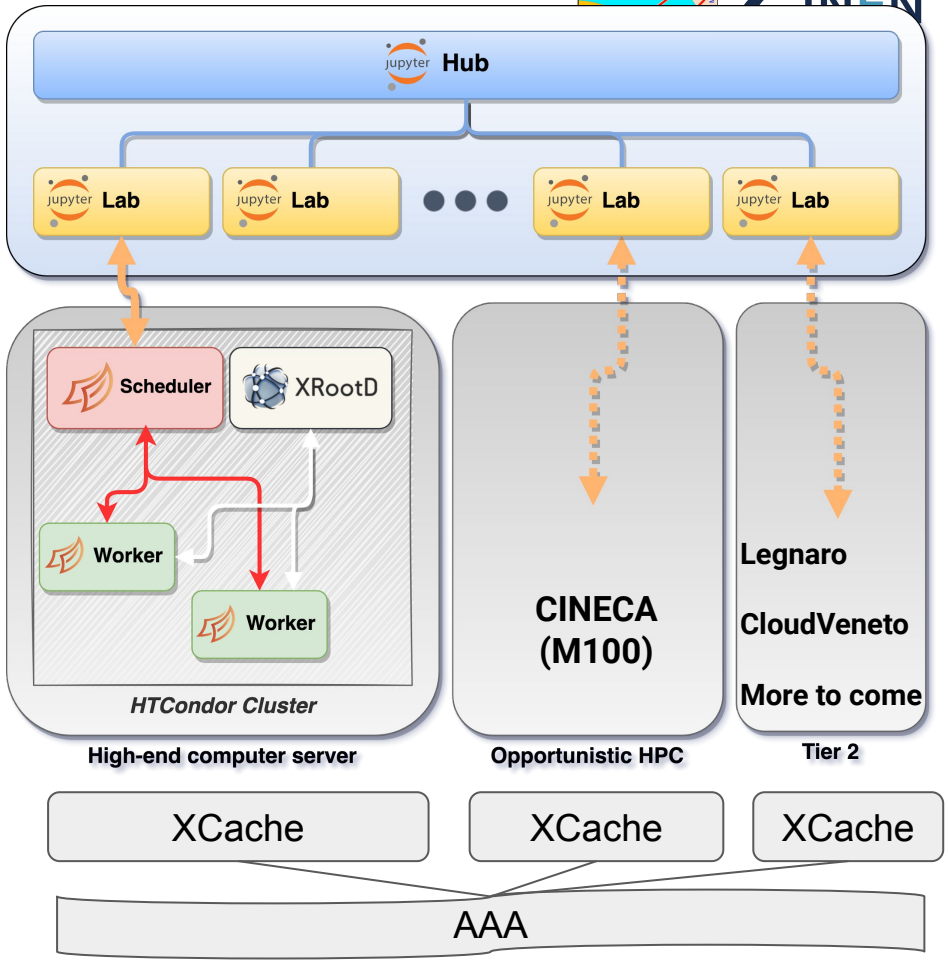


INFN testbed for future analyses

More resources are made available via a dedicate HTCondor pool where anyone can provide something with a bare minimum of requirements

(containerized environment and step-by-step instructions available)

So the idea here to use HTCondor a federator of resources we have around



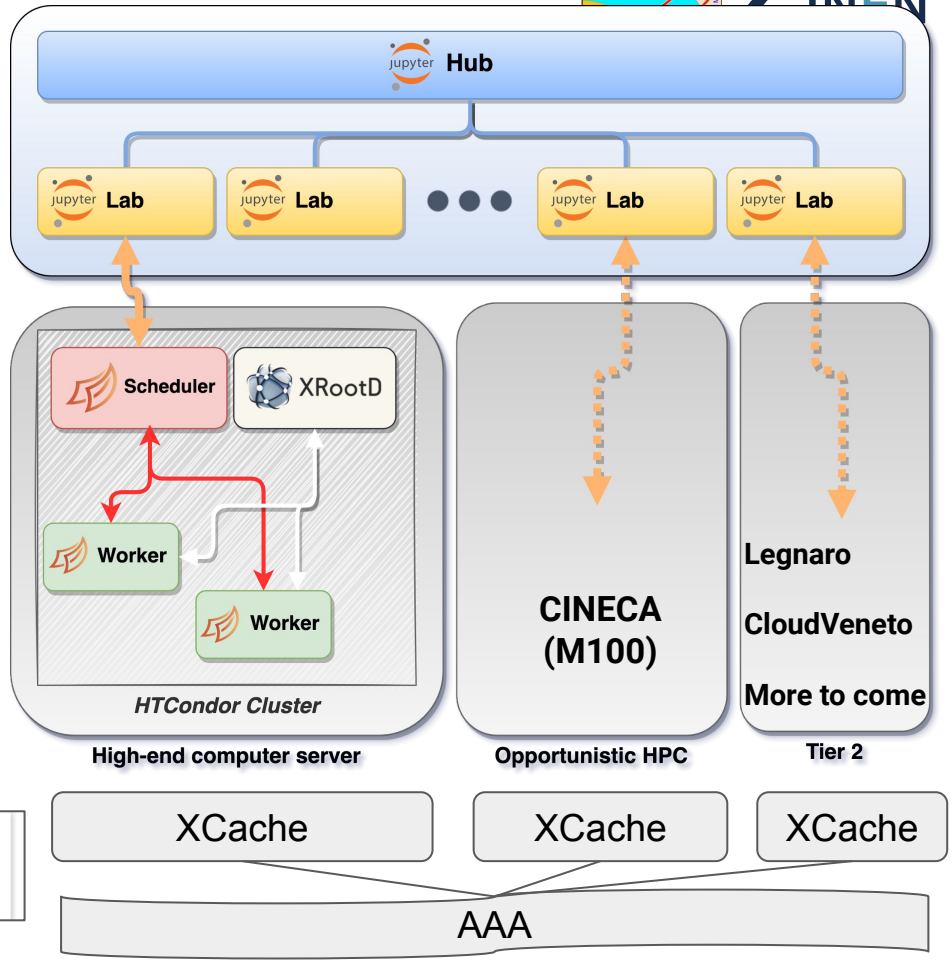
INFN testbed for future analyses

- From its notebook the user can ask for extension toward more resources directly via WebUI.

  Everything else will happen under the hood, allowing the code to be executed remotely on available resources.

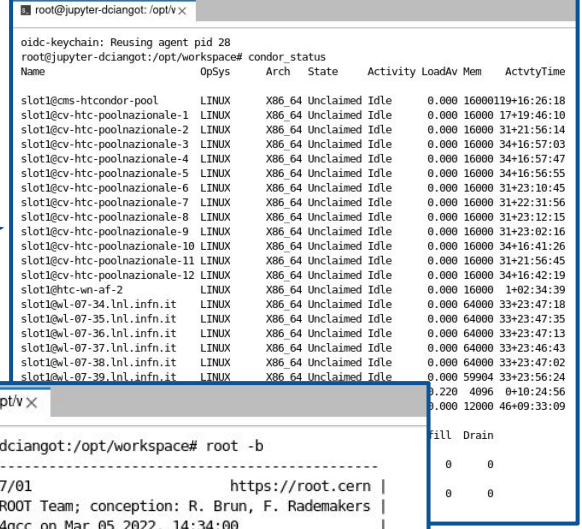
  Once ready the user will be prompted with the DASK client to be put in here:

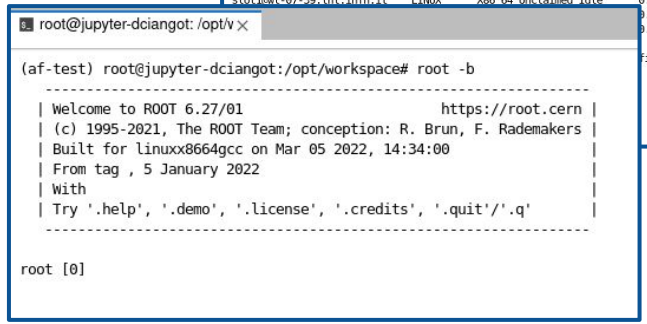
```
from dask.distributed import Client
df = RDataFrame('tree', 'f.root', daskclient=Client('tcp://host:port'))
```

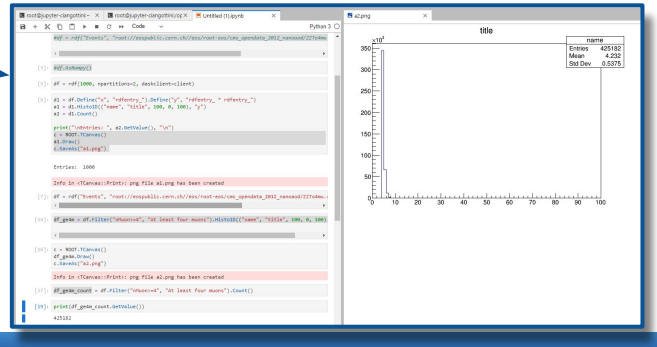


What can a user do in there?

- Batch/Legacy
  - Run your analysis on a batch system with resources collected over sites
  - Run your analysis on a batch system targeting specifically HPC resources
- Quasi-interactive python scripting
  - On-demand leverage big-notebooks on big machines (hpc node, dedicate hw) and run locally with a portable and ready-to-use environment
- Interactive python notebooks
  - Effortlessly scale local code with distributed mode RDataFrame workflows over a T2 site or over dedicated/specialized resources
  - Edit an reproduce plot interactively







From resource provider perspective

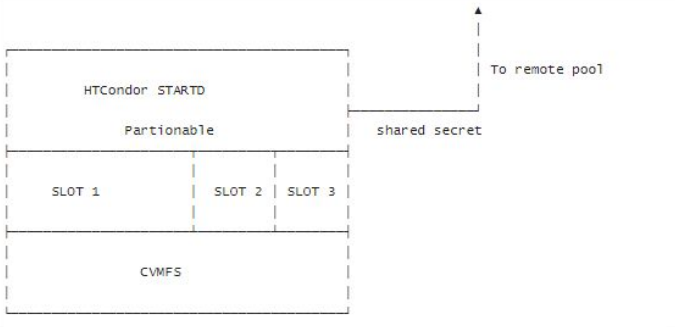
A **simple and battle tested recipe** is available for anyone wants to include its resources in the pool.

Minimum disk space requirement and that's all.

Container-based and managed via docker-compose

What will be installed?

- A HTCondor WN configured to register via shared secret to a remote pool
  - it will consists in a single partitionable slot with the whole node resources
- A cvmfs container mounting all the needed repositories
- An auto heal daemon to restart unhealthy containers



Requirements

- docker
  - most commonly you will only need the following:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

- docker-compose
  - on linux, usually this is done via:

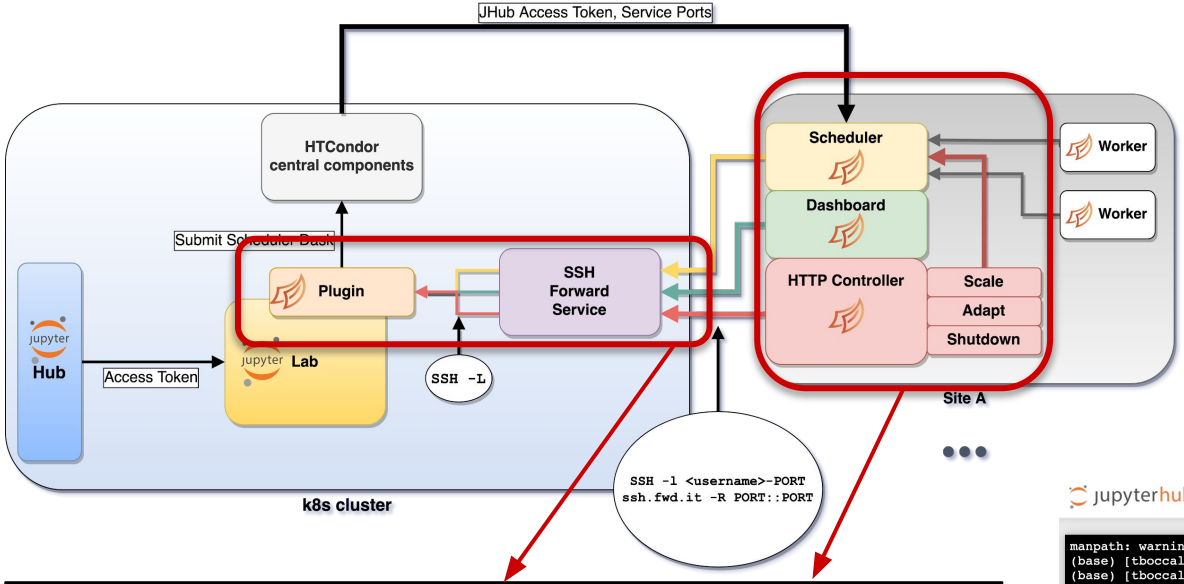
```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/dock
sudo chmod +x /usr/local/bin/docker-compose
```

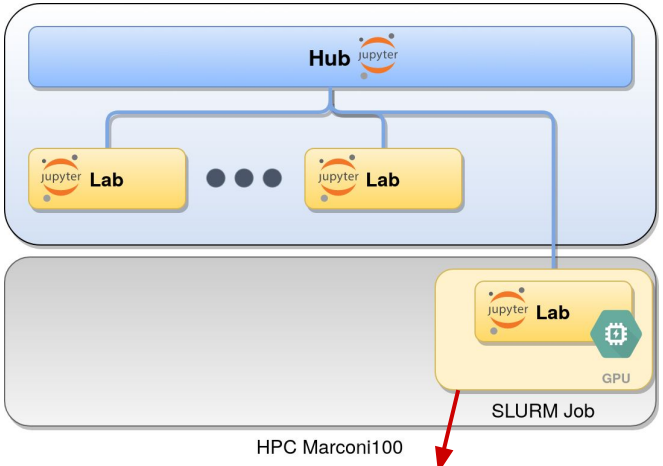
We now focus on those priorities

- **Benchmark event throughput and validate** of real analyses with:
  - Different data access patterns
  - Different code bases ➔ Dask task distribution/configuration
  - **In collaboration with RDataFrame developers**
- **Validating the compatibility with Coffea users** ⇦ **New Entry!!**
- **Scale tests (multiple users, multiple tasks)**
  - Dedicated high-performance machine
  - Scale over T2 site resources
  - But then also to try to scale over HPC CINECA/opportunistic resources
- **Demonstrating that we are able to satisfy a multi-experiment scenario** would be, of course, a great added value

BACKUP

Built in flexibility





Also, spawning notebooks directly on HPC (**M100**) resources via custom JHUB spawners has been demonstrated to be an option!

jupyterhub

```
manpath: warning: $MANPATH set, ignoring /etc/man_db.conf
(base) [tboccali@r256n06 tomboc1973]$
(base) [tboccali@r256n06 tomboc1973]$
(base) [tboccali@r256n06 tomboc1973]$ !ps
ps -ef|grep slurm|grep  tbocc
tboccali  87927  87919  0 11:12 ?        00:00:00 /bin/bash /var/spool/slurmd/job3440237/slurm_script
tboccali  88997  88660  0 11:13 pts/0    00:00:00 grep --color=auto slurm
(base) [tboccali@r256n06 tomboc1973]$
```

Creation of **custom components to add the capability to spawn DASK clusters remotely** w.r.t. to a JupyterLAB

SSH on demand via JHUB

Requirements

- go to https://cms-it-hub.cloud.cnaf.infn.it/hub/token and get a token. Take note of it.
- use the token as password to connect to your on-demand UI via:

```
ssh <username>@cms-it-hub.cloud.cnaf.infn.it -p 32022
```