Analysis Facility

D. Ciangottini, D. Spiga, T. Tedeschi
on behalf of CMS/INFN/Sites

Outline

- Problema ed obiettivi
- Analisi dichiarativa e distribuita, in sintesi
  - Un primo showcase con analisi CMS
- Infrastruttura abilitante per misure su analisi (quasi)interattiva
- Considerazioni e piani

Introduzione

L'idea dell'attività è dimostrare e verificare le principali funzionalità dei nuovi framework software per l'analisi dei dati **(quasi)interattiva e dichiarativa.**

R&D con obbiettivo **fase 2**, ma che speriamo di poter usare durante **Run3 come banco di prova.**

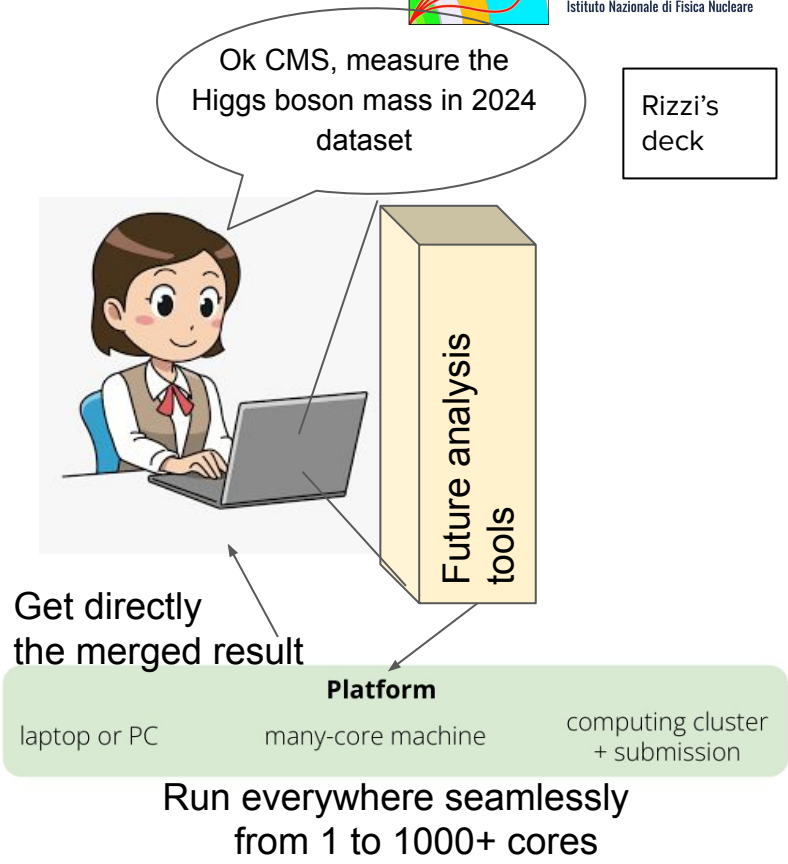
Stretta collaborazione con gli sviluppatori ROOT per la valutazione della soluzione **distributed RDataFrame (vedi dopo)**, allo stesso tempo e' in fase di studio l'utilizzo di framework alternativi (Coffea➜python data science pkgs)

Sebbene nel contesto CMS, come vedremo, e' un **insieme di tool del tutto generici.**

Utilizzeremo i risultati del demonstrator messo in piedi per un'analisi CMS solo come esempio di una analisi adattata dall'approccio a batch verso il dichiarativo/interattivo con RDataFrame

Perché lo facciamo?

- **Fare la fisica senza concentrarci sulle “technicalities”**
  - In questo senso abbandonare la programmazione ad event loop, in favore di **un approccio dichiarativo**
- **“Let the framework to optimize things”**
  - Delegare data splitting e gestione multi-threading al framework utilizzato
  - Esecuzione di codice ottimizzato “centralmente” invece che reinventato per ogni analisi
- **Condivisione e riutilizzo di “code for humans**
  - Facilitando il debug e la riproducibilita’



… e quindi:

- **Ottimizzare il “time to insight”**
  - Miliardi di eventi in O(1h)
  - Interattivita’ e user-friendly/standard UI
- **Fornire agli utenti un HUB per l'analisi**
  - Riducendo la complessita’ e scongiurando il tuning di soluzioni ad-hoc
- Promuovere la **portabilità attraverso l'uso di container**
  - Facilitando una transizione/adozione indolore
  - analysis framework agnostic
  - Propedeutico per sfruttare risorse eterogenee
- Migliorare il **throughput** (evts/s) rispetto all’attuale distribuzione “in batch”

Dal punto di vista infrastrutturale:

- **Un valore aggiunto deriva dalla possibilita’ di raccogliere e federare risorse eterogenee e distribuite in maniere efficiente**

Elementi caratterizzanti (high level)

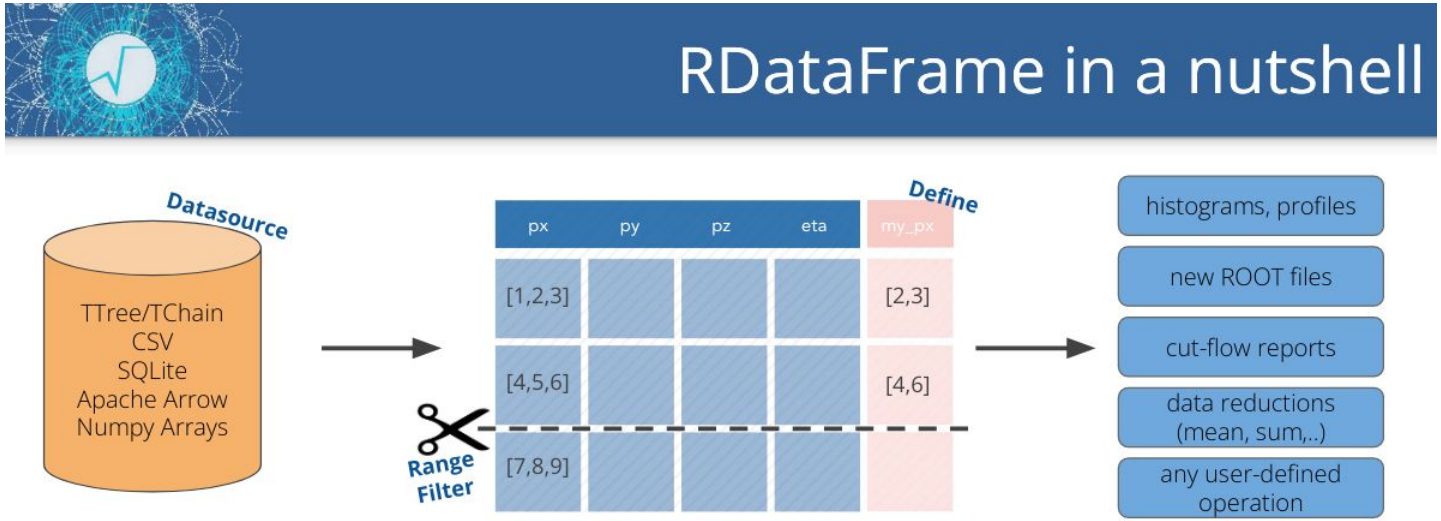
- Il **linguaggio principale** su cui ci concentriamo per analisi dati e' **Python**
  - alcune interfacce rimangono in C++ per i compiti più pesanti
- Focus su **formati tabulari** (**plain root files, csv**)
- Supporto per **l'offloading su risorse distribuite in modo tale che sia trasparente per l'utente**
  - multithreading
  - risorse distribuite

Anche dal punto di vista dei requirement non c'e' nulla di specifico per CMS, sono anzi richieste comuni del campo HEP ma anche data science.

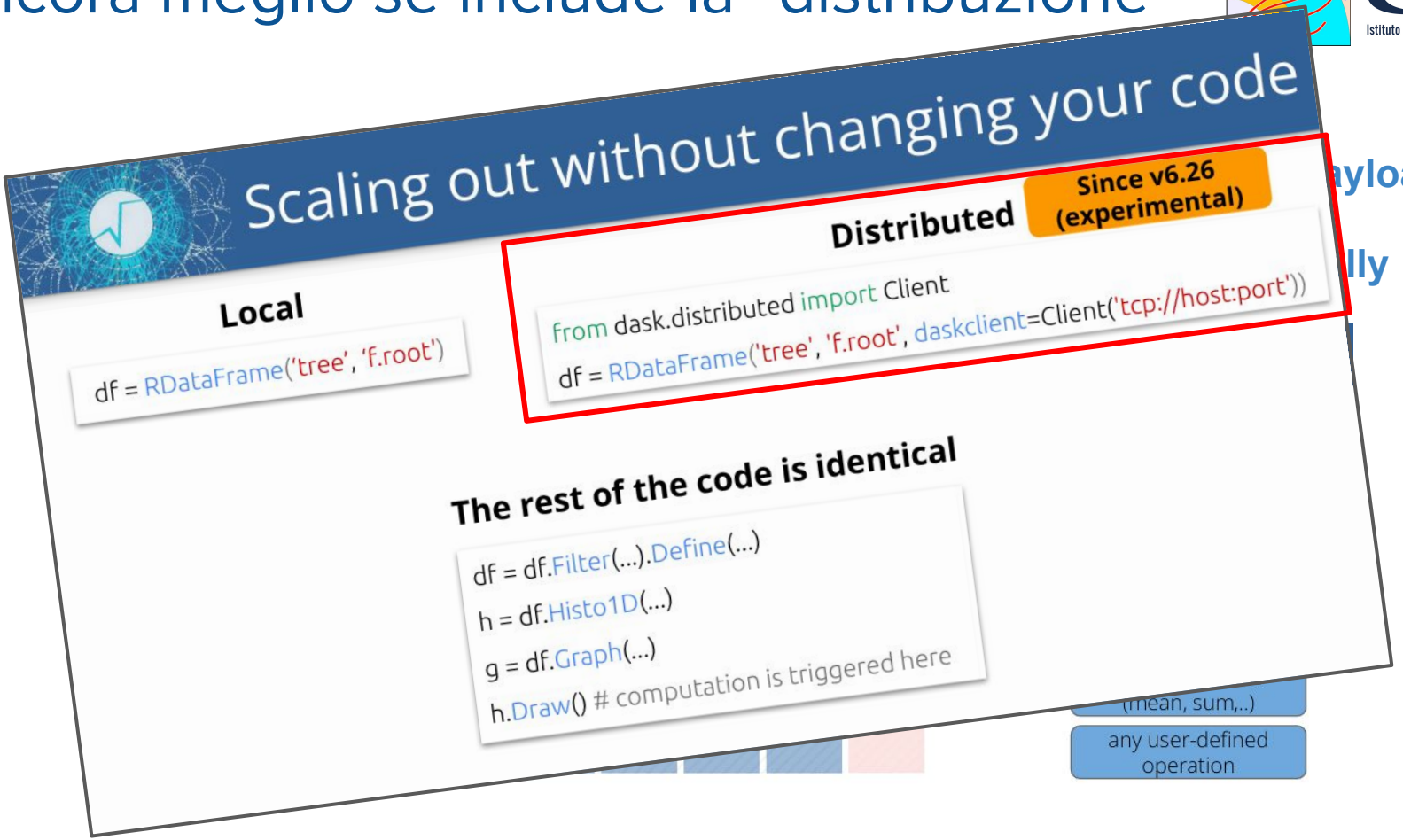
L’analisi dichiarativa

In contrapposizione alla definizione “imperativa” del codice di analisi:

- Il framework **si occupa di ottimizzare a livello piu’ basso le richieste dell’utente**
  - Ad esempio lo splitting dei payload e il bookkeeping/merging degli output
  - Con un impatto sia sulla curva di apprendimento che sulla efficienza

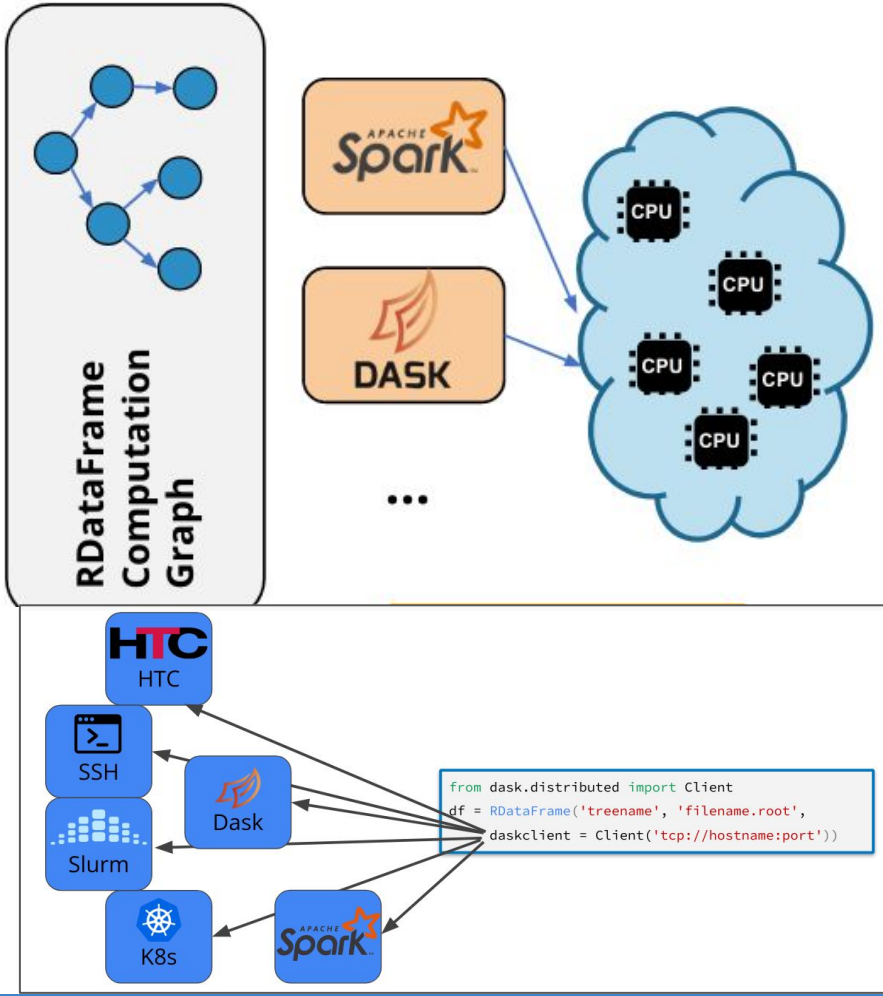


Ancora meglio se include la “distribuzione”



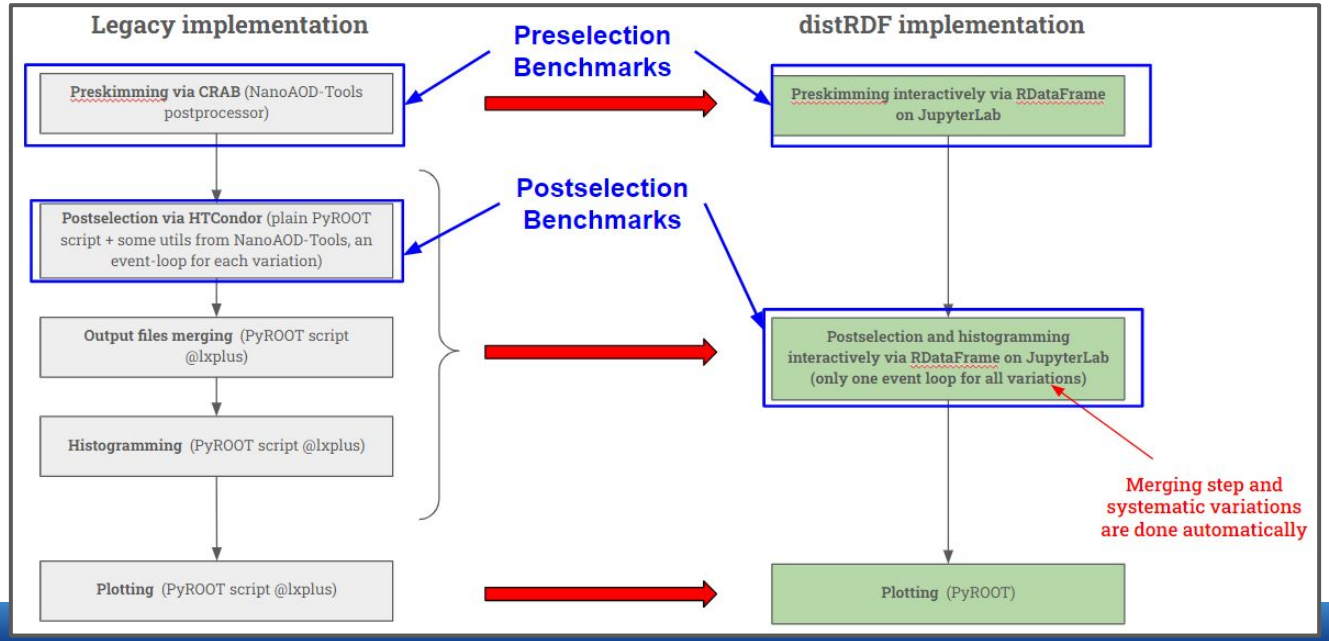
Fare offloading via DASK

- **DASK e' un framework per calcolo parallelo e distribuito**, principalmente basato sul paradigma map-reduce
- Offre una interfaccia **comune per gestire e distribuire task in python su una varieta' di "motori"**
  - HTCondor/Slurm queus
  - Machines via SSH
  - K8s
  - Spark
  - etc..
- E' questo il motivo per cui molti dei nuovi framework proposti sono in grado di interfacciarsi con DASK
  - E.g. **RDataFrame, Coffea** (see later)



Studio di prestazioni in CMS (1/2)

- Abbiamo completato la validazione e la comparazione di un'analisi CMS reale con o senza l'utilizzo di RDataFrame
  - VBS SSWW with a light lepton and an hadronic tau in final state
    - data to be processed for 2017 UL SM analysis: ca. 2TB (Data + MC)



Studio di prestazioni in CMS (2/2)

- Un **benchmark **preliminare** ha mostrato risultati incoraggianti** in termini di throughput
  - **gia' con una configurazione base**
  - Nonostante il laborioso tuning per il caso "legacy"

- Preselection:
  - Filters on trigger, correction computation
- Postselection:
  - Proper event reconstruction

Preselection		
	Legacy	**RDF (O2)**
Overall time	3h 40min	25min
Overall rate	693 Hz	7306 Hz
Event-loop rate	721 Hz	8473 Hz
Overall network read	488 GB	371 GB
Average RSS per-node	Ca. 13 GB	Ca. 17 GB

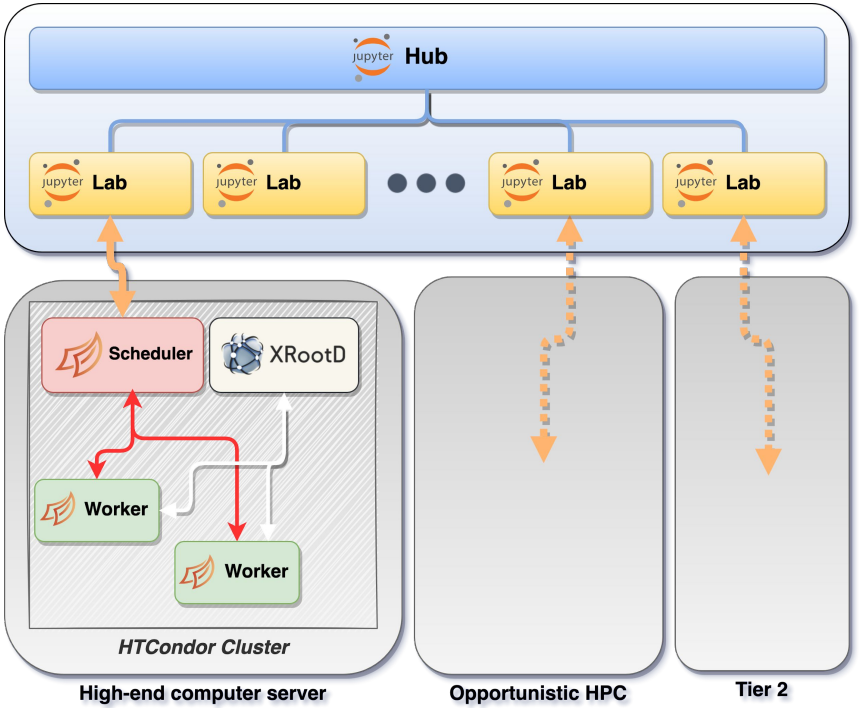
Postselection		
	Legacy	**RDF**
Overall time	0.25h	0.08h
Overall rate	306 Hz	855 Hz
Event-loop rate	412 Hz	1976 Hz
Overall network read	11 GB	10 GB
Average RSS per-node	Ca. 1 GB	Ca. 15 GB

Infrastruttura abilitante

WP5 sinergy

- Lo showcase mostrato ha sfruttato una infrastruttura abilitante che rappresenta il modello di facility che ci proponiamo di validare con use case di analisi
- Abbiamo istanziato un primo **testbed funzionante che puo' essere usato come playground**
  - **"state of the art/industry software toolsets"**
  - **"Develop locally than scale out seamlessly"**

Experiment "Data-lake"



La gestione dei dati: lo stato

WP5 sinergy

- **Data discovery**
  - The file list can be retrieved via RUCIO DM integration
    - Interaction with RUCIO server is done via OIDC authN/Z seamlessly once the user login
- **Input reading**
  - Currently data **can be accessed from the CMS storage federation via a legacy X509 user proxy authN/Z**
    - Some preliminary OIDC based access pattern has been validated
  - Software on **CVMFS mounted already on both UI and remote worker nodes**
    - **User containers can be also dynamically loaded from CVMFS**
- **Output registration**
  - The **output can be saved on a local (AF cluster) area**
  - In alternative, for the moment, you will need to upload a valid proxy and using it to copy files on your preferred storage endpoint

Visione per il futuro

WP5 + WP4 sinergy

- **Composable workflows**
  - Idealmente si puo pensare di **integrare tool industriali per la gestione di pipeline**
  - L'utente in questo deve solo **scegliere tra una serie di componenti pronti all'uso per ogni parte dell'analisi**
    - Solo in caso di necessita' veramente specifiche, il lavoro comprenderebbe scrivere del vero e proprio codice
- **Accesso ad hardware specializzato**
  - Nell'ambito dello spoke potremmo pensare alla validazione di **offloading su e.g. GPU and FPGA**
- **On-demand ML inference service**
  - Non solo l'analisi, ma anche l'inferenza puo' sfruttare tool moderni per essere delegata ad **hardware dedicato e servito su richiesta**
- Rimane centrale fare **"onboarding" di nuovi casi d'uso e idee da comunita' interessate**

Backup

Rimanere generici

Il mantra e' questo. Rimanere generici sia in termini di framework che di esperimento. In particolare sfruttando i containers e le soluzioni dedicate per la distribuzione del software. In altre parole l'approccio in sviluppo deve permettere:

- Il riutilizzo ed eventuale customizzazione da parte di diversi esperimenti
  - Al momento LHCb sta testando una istanza gemella
- Framework agnostic: supportare RDataframe in maniera NON esclusiva
  - E' gia in corso la validazione dell'infrastruttura con il framework Coffea
- Non solo HEP
  - con PLANET (un progetto "data-science" su covid) ha utilizzato una versione custom della stessa soluzione, per una prima validazione dell'utilizzo per attivita' generiche di data-science

Infrastruttura abilitante

WP5 sinergy

L'utente effettua il login via web ed accede ad un jupyterlab personale con un seed minimo di risorse per la fase di esplorazione dei dati

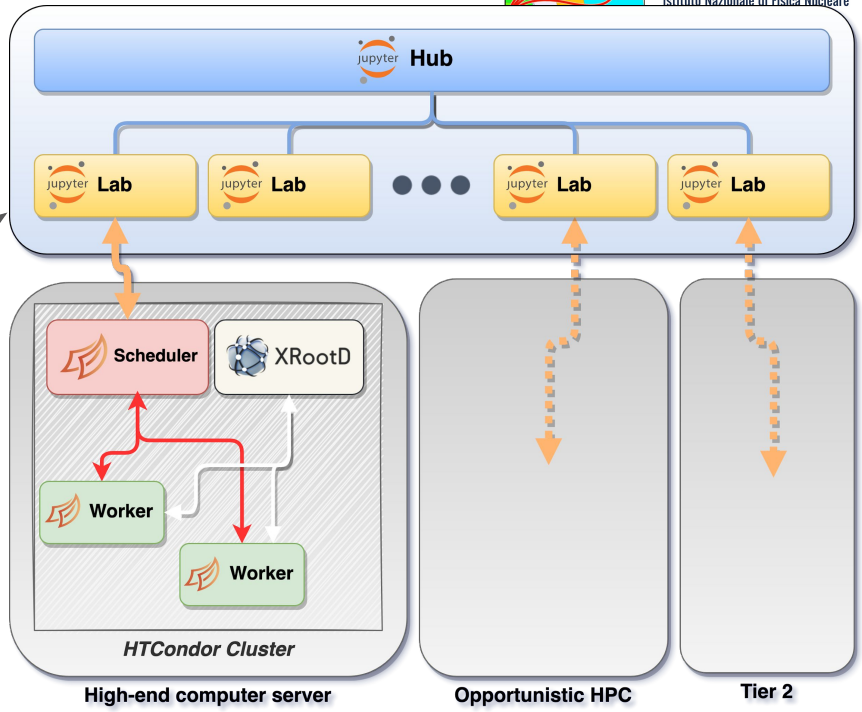
Attraverso l'integrazione nativa con Dask, attraverso la webUI l'utente puo creare un cluster Dask remoto su cui distribuire il suo worflow in maniera interattiva

ato una
esente
iamo di

tbed
sato

come playground

- "state of the art/industry software toolsets"
- "Develop locally than scale out seamlessly"



Experiment "Data-lake"

La gestione dei dati: lista dei desideri

- Output metadata and DM registration
  - Storing analysis output with metadata (versioning etc) on the very same “lake” where the input are stored
- Having access to a “shared home” experience
  - FAIR, easy and seamless access to all the data from anywhere (local machine, remote notebook, remote worker nodes)

Both can be implemented either with CMS current toolset, and (maybe more interestingly) this can be a target achievable within a CNS data-lake environment