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

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

*Lucio Anderlini, **Giulio Bianchini**, Diego Ciangottini, Federica Fanzago, Rosa Petrini, Massimo Sgaravatto, Daniele Spiga, Tommaso Tedeschi, Antonino Troja, Lisa Zangrando*

Integration and testing of a system based on Virtual Kubelet for the offloading of containerized workflows

04/06/2024

INFN

1 Introduction

Enabling an offloading model for cloud applications on non-cloud backends*

2 Goals

Test of the identified solution and use case-driven objectives

3 Status of activities

Status of tests and progress towards the identified use cases

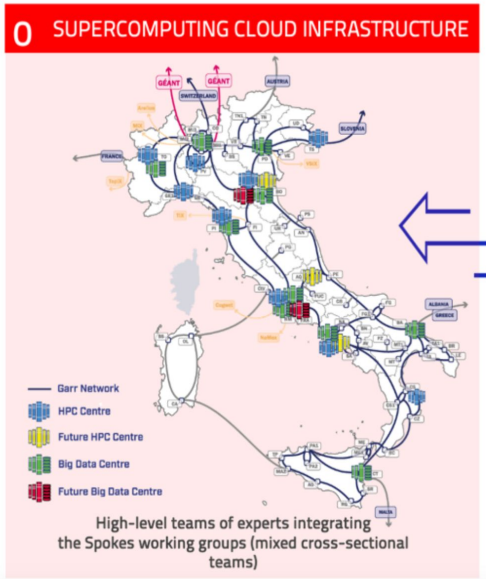
4 Conclusions

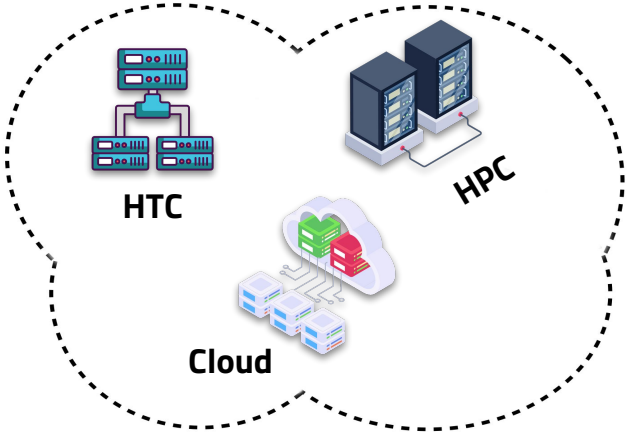
Offloading = Process through which a container, designed to run in a cloud environment, is transparently delegated to a remote agent (HPC, HTC, etc.)*

Introduction

A highly heterogeneous context: HPC, HTC, and Cloud

State-of-the-art infrastructure for high-performance computing and big data management, leveraging existing resources and integrating emerging technologies.





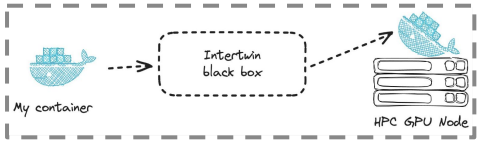
High-Performance Computing (**HPC**), High-Throughput Computing (**HTC**), and **Cloud computing** ... various resource providers with different backends.

Goal: Transparently utilize resources provided with different models (backends), particularly extending cloud-native applications even to non-cloud backends.

What we want to enable

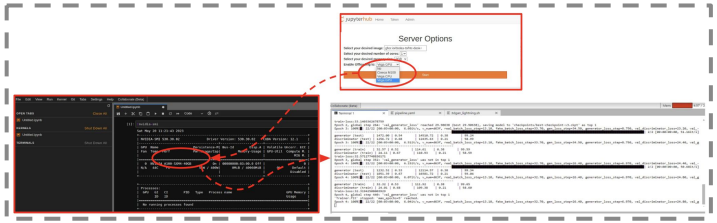
Execution of a simple POD

To create a simple container and have it executed by a remote Slurm batch on an HPC



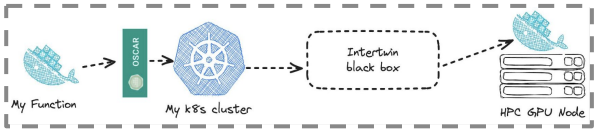
Interactive sessions

Generate on-demand JupyterLab instances on an HPC along with other more "cloud-ish" instances on K8s.



Scale out workload

To execute a payload in response to an external trigger



Managing backends / provisioning models

- Single set of APIs to integrate resources provided by providers using diverse technologies and architectures

Moving cloud service payloads according to specific needs

- payload compute-intensive, memory-intensive, gpu-intensive, ...

To "hide" heterogeneity from the user

- For the end user, everything is transparent; the offloading system takes care of orchestrating the execution and deciding on which backend (Slurm, HTCondor, Kubernetes, etc.) to run the workloads.

Using a lightweight and easily maintainable system

- A modular architecture that allows for the integration of different backend providers through plugins.
- Ease of maintenance

How: interLink

HOW?

Extending the **Virtual Kubelet** solution by creating a first draft of a generic API layer to delegate the execution of PODs to **ANY** remote backend.

VK core
A pod that masquerades as a node and handles POD requests from the K8S scheduler

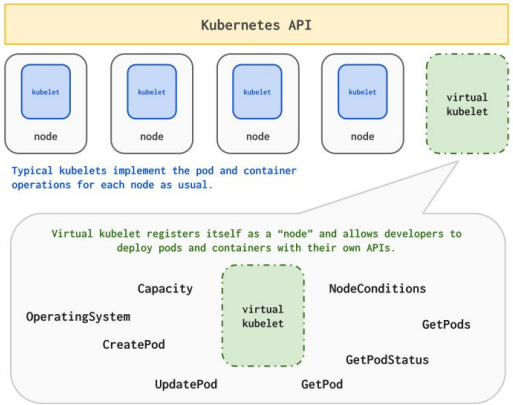
InterLink Server
It sits between the VK and the sidecar. It manages requests coming from the VK and forwards them to the sidecar.

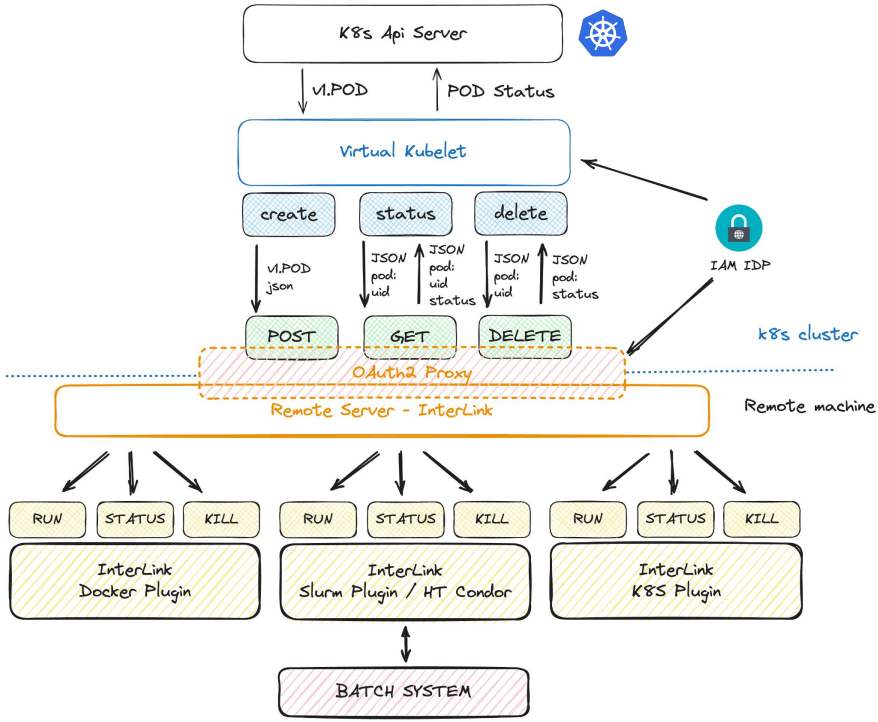
Sidecar
It runs the containers on the infrastructure and returns the result. It communicates with the InterLink server.

WHAT?

Virtual Kubelet

Extending k8s without imposing specific k8s dependencies





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. Further details will be discussed on Friday
- **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 sinergies should lead toward a generic technical solution, versatile and extendible based on specific needs.

Toward the first tests...

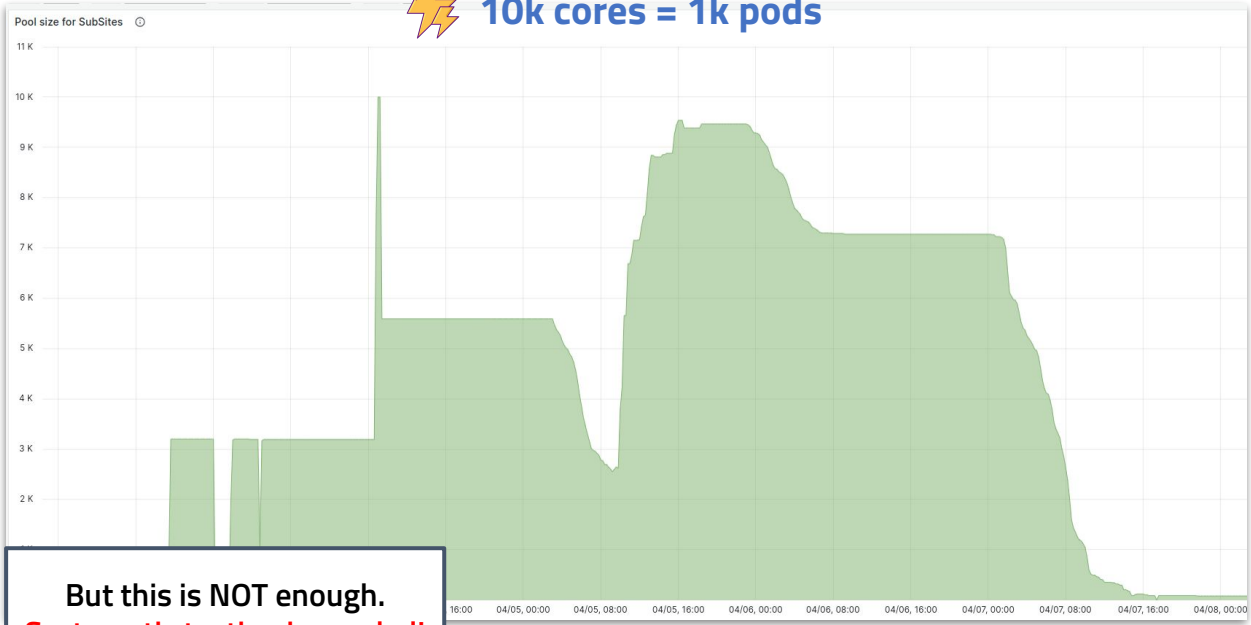
InterLink is under development, but some preliminary tests have already been successfully conducted!

First scalability test conducted with the Slurm plugin!

The pods submitted required 10 cores each. **We reached a peak of 10k cores on the VEGA HPC** (Maribor, Slovenia).

All managed by a **common Kubernetes cluster** (without dedicated hardware) on the **INFN cloud**, with no signs of crisis.

10k cores = 1k pods



But this is NOT enough.
Systematic testing is needed!

Testing, use-case driven objectives, and status of activities

Test e objectives use-case driven

We have identified some use cases that could be relevant for this scenario

Testing the offloading system

- on the **functionalities**, identify which features need to be implemented
  - multi container POD, empty dir, shared volumes, ...
- with **systematic** tests on various fronts
  - to verify resilience, robustness, scalability, and resource management

- **JupyterHub that spawn "remote" JupyterLab instances**
  - very common and generic use case
- **AI_INFN Platform**
  - exploit specialized hardware (GPU)
- **High Rate Analysis platform**
  - need for scaling out (synergy with Spoke2/3 use cases)

Test	Feature tested
000-hello-world	submit
	log retrival
010-simple-python	unicode formatting
020-python-env	env vars by value
	env vars by configmap
	env vars by secret
030-shared-volume	multi-container pod
	emptyDir volume shared
	read-only mount points
040-config-volumes	configmap volumes
	secret volumes
050-limits	job gets killed
	status set to OOM Killed
060-init-container	initContainer treated properly
	egress towards github
070-rclone-bind	inter-container networking
	fuse mount point
	synchronization of containers
...	...

Activity status - functionality testing

The offloading system consists of multiple components. In this context, tests are essential to verify which functionalities need to be implemented for the various plugins, in order to develop and integrate as many features as possible.

Test	Feature tested	Interlink support	docker-plugin support	kueue-plugin support	k8s-plugin support
000-hello-world	submit	TRUE	TRUE	TRUE	TRUE
	log retrival	TRUE	TRUE	TRUE	TRUE
010-simple-python	unicode formatting	TRUE	TRUE	TRUE	TRUE
020-python-env	env vars by value	TRUE	TRUE	TRUE	TRUE
	env vars by configmap	FALSE			
	env vars by secret	FALSE			
030-shared-volume	multi-container pod	TRUE	TRUE	TRUE	TRUE
	emptyDir volume shared	TRUE	TRUE	TRUE	TRUE
	read-only mount points	TRUE	TRUE	TRUE	TRUE
040-config-volumes	configmap volumes	TRUE	TRUE	TRUE	TRUE
	secret volumes	TRUE	TRUE	TRUE	TRUE
050-limits	job gets killed	TRUE	TRUE	TRUE	FALSE
	status set to OOM Killed	TRUE	TRUE	TRUE	FALSE
060-init-container	initContainer treated properly	TRUE	TRUE	TRUE	TRUE
	egress towards github	TRUE	TRUE	TRUE	TRUE
070-rclone-bind	inter-container networking	TRUE	FALSE	TRUE	TRUE
	fuse mount point	TRUE	FALSE	TRUE	TRUE
	synchronization of containers	TRUE	FALSE	TRUE	TRUE

More tests...

Reliability

Verify resilience to unexpected events

Multi-tenancy

Verify management of concurrent accesses

Scalability

Verify scalability with varying demands

Resources

Verify resource management (GPU).

Virtual Kubelet	InterLink	Sidecar
↑	↑	↑
↑	↑	↓
↑	↓	↑
↑	↓	↓
↓	↑	↑
↓	↑	↓
↓	↓	↑
↓	↓	↓

interLink BETA

jupyterhub

jupyterLab jupyterLab ... jupyterLab

But a system must not only work, it must also be guided by the needs...

POD POD POD ...

interLink ?

POD POD ...

interLink ?

Activity status - JHUB towards AI_INFN

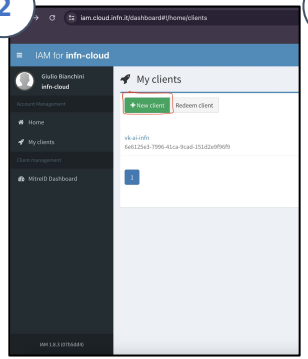
Example of deploying a cloud-native application (JHUB) in an offloading-enabled cluster

Documented example on confluence

1. K8S cluster created through INFN cloud dashboard
2. Authentication between VK and InterLink using IAM INFN token
3. Deployment of VK and InterLink API Layer
4. Resource provider configuration with Docker Sidecar Plugin to execute requests
5. Deploy with helm chart
6. JHUB access through IAM INFN authentication
7. GPU provisioning

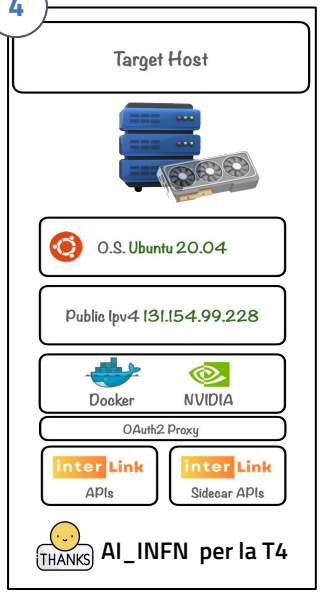
1

2



3

4

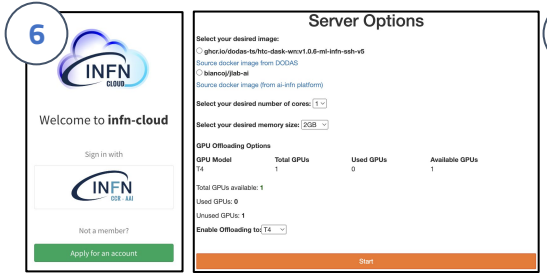


5

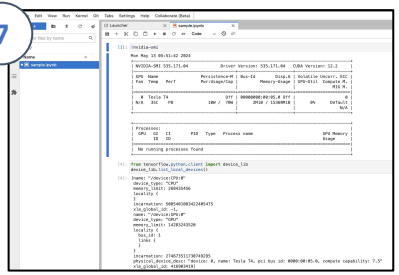
HELM

```
helm upgrade --install helm-jhub-release helm-jhub-inttw/ -n helm-jhub-namespace --debug
```

6



7



The work done for the deployment of a generic JHUB was fundamental and preliminary for integrating the offloading system with the AI_INFN platform.

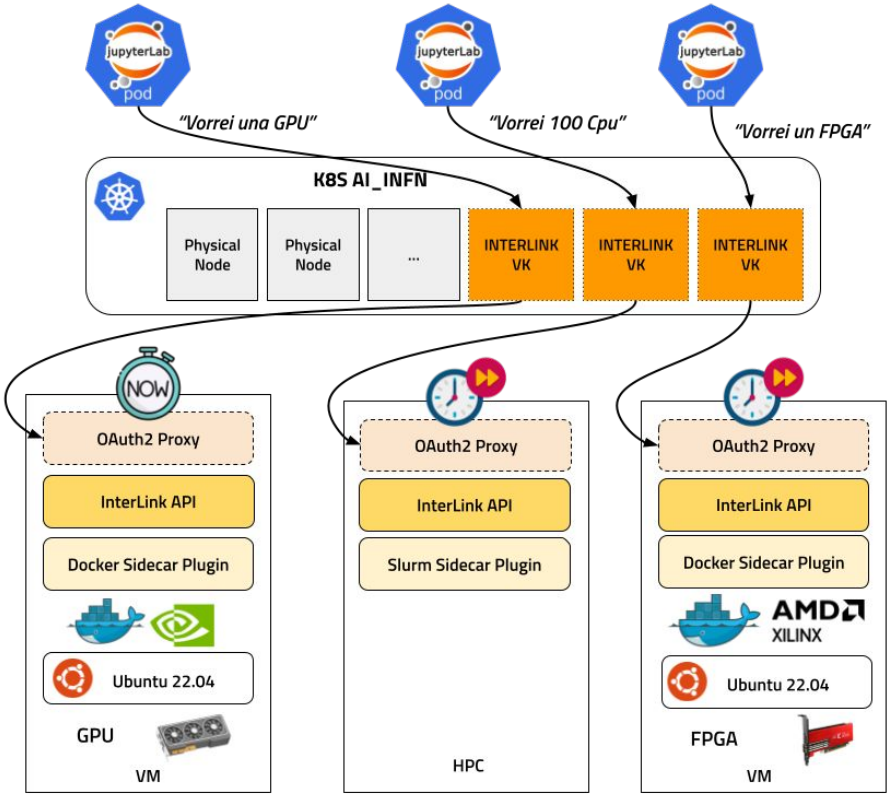
Activity status - AI_INFN Platform

- The **AI_INFN platform provides a complex cloud-native use case** to test InterLink with:
  - Interactive access via offloading
  - Heterogeneous computing (CPU, GPU, FPGA...)

- We are proceeding in parallel with the development of two plugins (**docker and kueue**) to demonstrate decoupling from the backend.

- **The docker plugin** has already been validated for GPU provisioning and **can similarly support FPGA provisioning**.

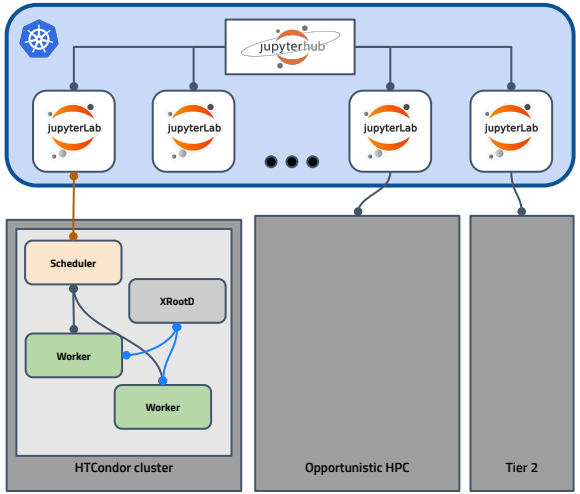
- In concrete actions, the AI_INFN platform could already leverage this offloading system for spawning JupyterLab instances with some limitations (NFS).



Use case - High Rate Platform

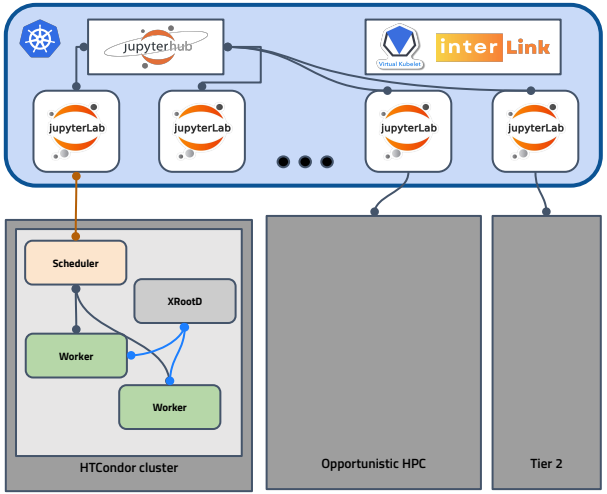
Analysis facility

- Access to a single HUB and authentication via token (INDIGO-IAM)
- Customizable Python kernel
- Fully containerizable workspace
- Overlay based on **HTCondor**
- DASK library (Python) for distributed computing
  - Scales execution from 1 to N cores
- Possible implementation on heterogeneous resources
- Configurable data access with WLCG



high-end compute server

Initially, the worker nodes of the Condor pool are instantiated via Docker compose on dedicated resources. The goal is to scale beyond dedicated nodes.



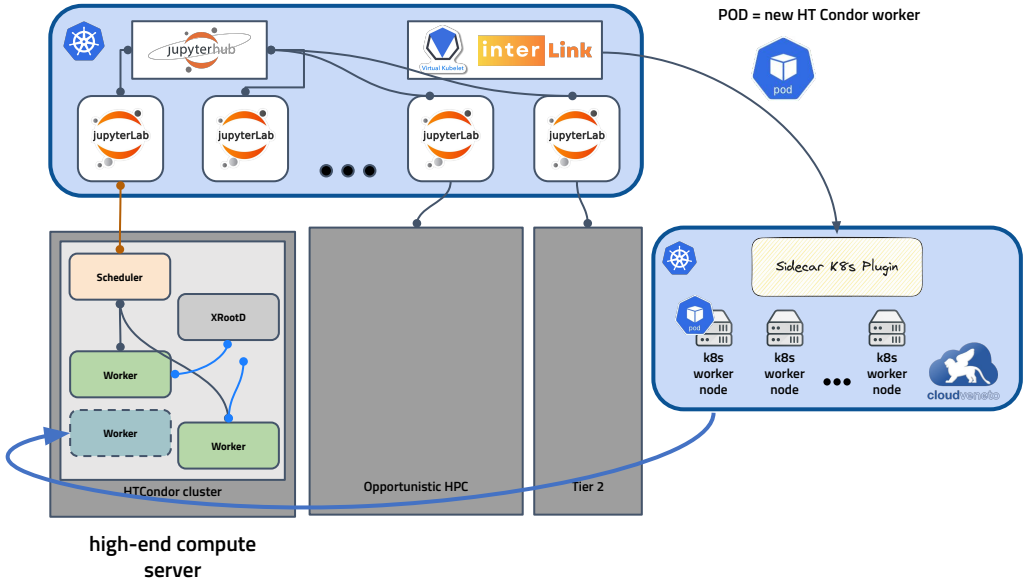
high-end compute server

InterLink to extend the HT Condor pool and opportunistically increase the number of available worker nodes by leveraging distributed back-ends in parallel and geographically, such as CloudVeneto resources.

Activity status - High Rate Platform

k8s plugin repository

The **Kubernetes sidecar plugin** has been developed. The POD submitted to the VK of the Analysis Facility's k8s cluster becomes a worker node that joins the central HT Condor pool by leveraging the resources of a K8s cluster that utilizes CloudVeneto resources.



Status

- **Fully functional workflow.**
  - In the Condor pool, CloudVeneto nodes are added which can be used to submit Condor jobs.
  - Multiple VKs coexist in the same AF k8s and offload onto different providers:
    - VK dedicated to the Legnaro, Bari, and Pisa tier2 (HTCondor sidecar)
    - VK dedicated to the Rome tier2 (ARC sidecar)
- **Sidecar plugin k8s**
  - Supports CVMFS provisioning

Thank you for your attention!