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*

Integrazione e test di un sistema basato su Virtual Kubelet per l'offloading di workflow containerizzati

Workshop sul Calcolo nell'INFN, Palau (Sassari), 21/05/2024

INFN

1 Introduzione

Abilitazione di un modello di offloading per applicazioni cloud su backend non cloud*

2 Obiettivi

Matrice di test della soluzione identificata e obiettivi use case-driven

3 Stato delle attività

Stato dei tests e progressi verso i casi d'uso individuati

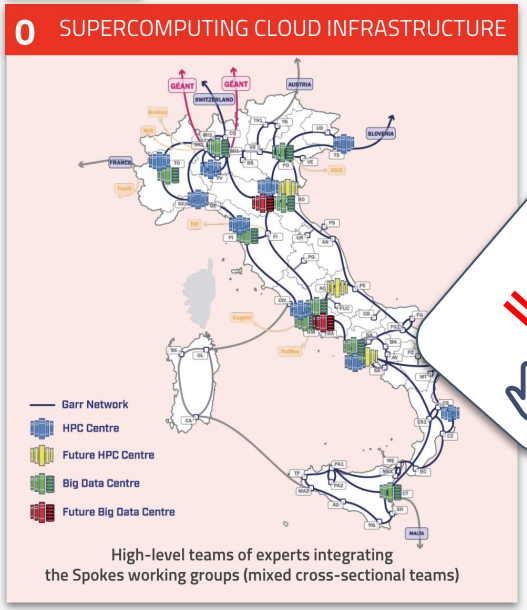
4 Conclusioni

Offloading = Processo attraverso il quale un container, progettato per funzionare in un ambiente cloud, viene delegato in modo trasparente ad un agente remoto (HPC, HTC, ecc.)*

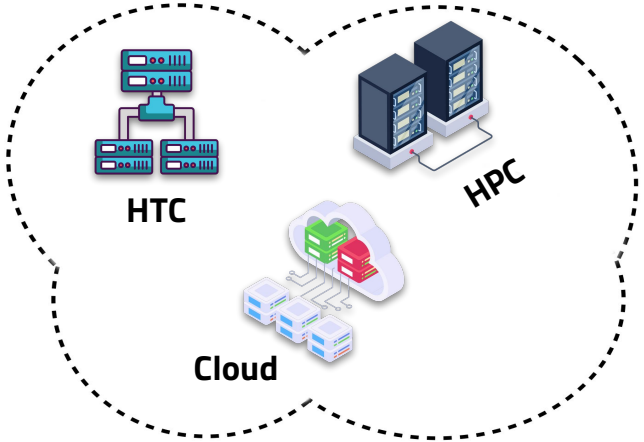
Introduzione

Un contesto altamente eterogeneo: HPC, HTC and Cloud

Infrastruttura all'avanguardia per il calcolo ad alte prestazioni e la gestione dei big data, che sfrutta le risorse esistenti e integra le tecnologie emergenti.



Il contesto lo conosciamo!



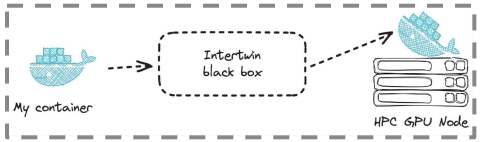
High-Performance Computing (**HPC**), High-Throughput Computing (**HTC**), and **Cloud computing** … diversi provider di risorse con backend differenti.

L'obiettivo: usare in modo trasparente risorse fornite con diversi modelli (backend) in particolare estendere applicazioni cloud native anche su backend NON cloud

Cosa vogliamo abilitare

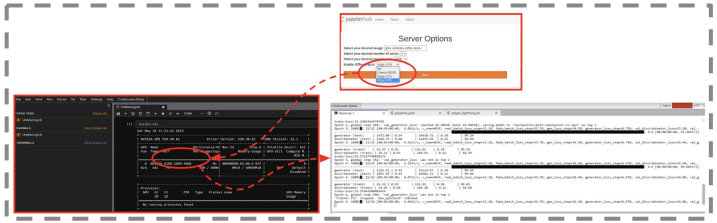
Esecuzione di un POD semplice

Per creare un semplice container e farlo eseguire da un batch slurm remoto in un HPC



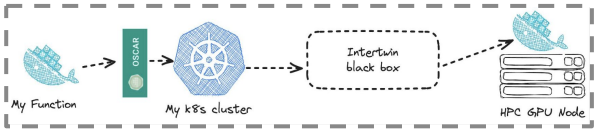
Sessioni interattive

Generare istanze JupyterLab on-demand su un HPC insieme ad altre istanze più "cloud-ish" su K8s.



Scale out workload

Per eseguire un payload in risposta ad un trigger esterno



- **Gestire backend / modelli di provisioning**
  - set unico di API per integrare risorse fornite da provider che utilizzano tecnologie e architetture diverse
- **Spostare i payload di un servizio cloud in base alle specifiche esigenze**
  - payload compute-intensive, memory-intensive, gpu-intensive, …
- **"nascondere" all'utente l'eterogeneità**
  - per l'utente finale e' tutto trasparente, il sistema di offloading si preoccupa di orchestrare l'esecuzione e decidere su quale backend (slurm, HTCondor, k8s, ….) eseguire i workload
- **Usare un sistema lightweight e facilmente mantenibile**
  - architettura modulare che permette di integrare diversi provider di backend tramite plugin.

Come: interLink

HOW?

Estendendo la soluzione **Virtual Kubelet** realizzando un primo draft di un generico API layer per delegare l'esecuzione di POD su **QUALSIASI** backend remoto.

VK core

Un pod che si maschera da nodo e prende in carico le richieste di POD dallo scheduler di K8S

InterLink Server

Si interpone tra il VK e il sidecar. Gestisce le richieste provenienti dal VK e le inoltre al sidecar

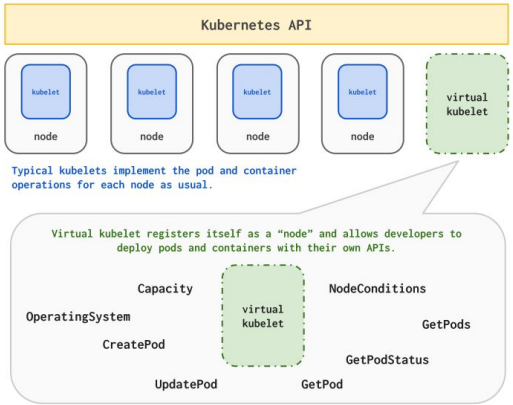
Sidecar

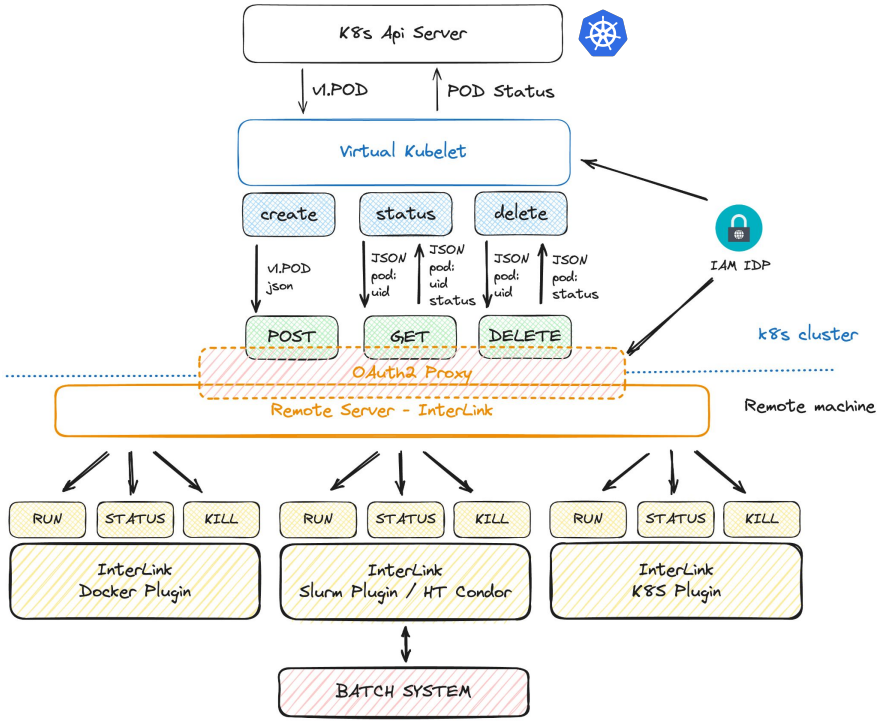
Esegue i container sull'infrastruttura e ritorna il risultato. Comunica con il server InterLink.

WHAT?

Virtual Kubelet

Estendere k8s senza imporre dipendenze specifiche di k8s





Verso i primi test ...

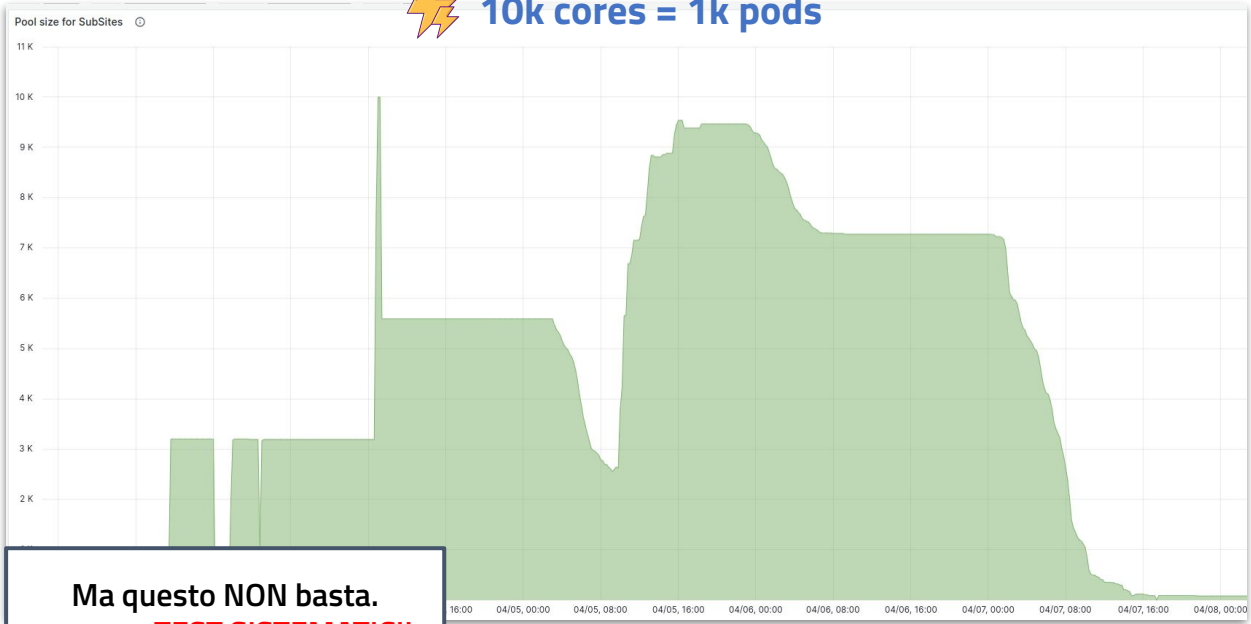
InterLink e' in fase di sviluppo ma alcuni test preliminari sono già stati eseguiti con successo!

Primo test di scala effettuato con il plugin slum!

I pod inviati richiedevano 10 core ciascuno. Abbiamo raggiunto il picco di **10k core su HPC VEGA (Maribor, Slovenia)**.

Tutto gestito da un **cluster Kubernetes comune** (senza hardware dedicato) su **cloud INFN**, senza segni di crisi.

10k cores = 1k pods



Ma questo NON basta.
servono TEST SISTEMATICI!

Test, obiettivi use-case driven e stato delle attività

Test e obiettivi use-case driven

Abbiamo individuato alcuni casi d'uso che potrebbero essere rilevanti per questo scenario

Testare il sistema di offloading

- sulle **funzionalità**, individuare quali features sono da implementare
  - multi container POD, empty dir, volumi condivisi, ...
- con **prove sistematiche** su diversi fronti
  - per verificare la resilienza, la robustezza, la scalabilità e la gestione delle risorse

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
...	...

- JupyterHub che crea istanze di JupyterLab "remote"
  - use case molto comune e generico
- Piattaforma AI_INFN Talk Rosa
  - sfrutta hw specializzato (GPU)
- High Rate Analysis platform
  - necessità di fare scale out (sinergia con use case Spoke2/3)

Stato dell'attività - test funzionalità

Il sistema di offloading è costituito da molteplici componenti. In questo contesto, i test sono essenziali per verificare quali funzionalità sono da implementare per i vari plugin, al fine di sviluppare e integrare il maggior numero possibile di funzionalità.

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

Matrice dei test

Affidabilità

Verificare resilienza agli imprevisti

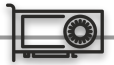
Multiutenza

Verificare la gestione di accessi concorrenziali

Scalabilità

Verificare scalabilità al variare delle richieste

Risorse



Verificare la gestione delle risorse (GPU).

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

interLink BETA

jupyterhub

jupyterLab jupyterLab ... jupyterLab

Ma un sistema non deve solo funzionare, ma deve essere guidato anche dalle necessità...

POD POD POD ...

interLink ?

POD POD ...

interLink ?

Stato dell'attività - JHUB verso AI_INFN

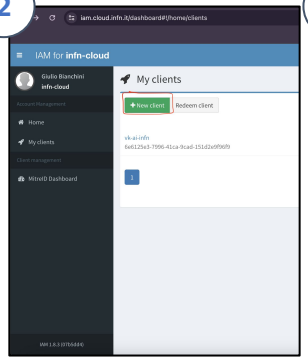
Esempio di deployment di un'applicazione cloud native (JHUB) in un cluster abilitato per l'offloading

Esempio documentato su confluence

1. Cluster K8S creato tramite dashboard di INFN cloud
2. Autenticazione tra VK e InterLink utilizzando token IAM INFN
3. Deployment del VK e di InterLink API Layer
4. Configurazione provider risorse con Docker Sidecar Plugin per eseguire le richieste
5. Installazione helm chart
6. Accesso a JupyterHub tramite autenticazione IAM INFN
7. GPU provisioning

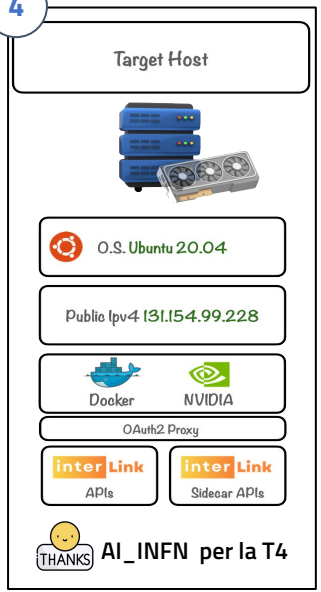
1

2

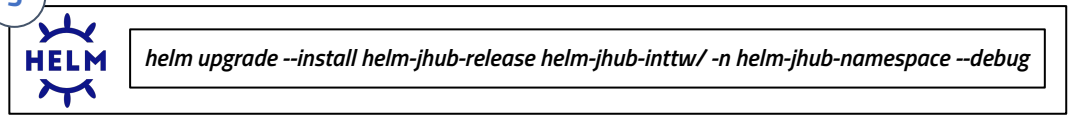


3

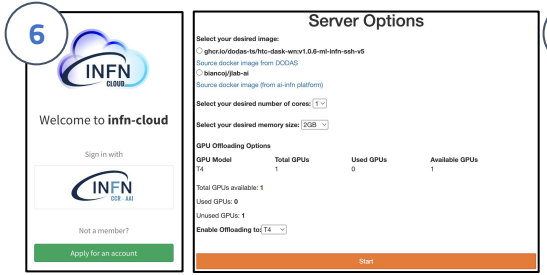
4



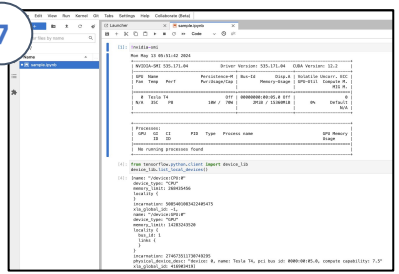
5



6



7



Il lavoro svolto per il deployment di un JHUB generico è stato fondamentale e preliminare per l'integrazione del sistema di offloading con la piattaforma AI_INFN.

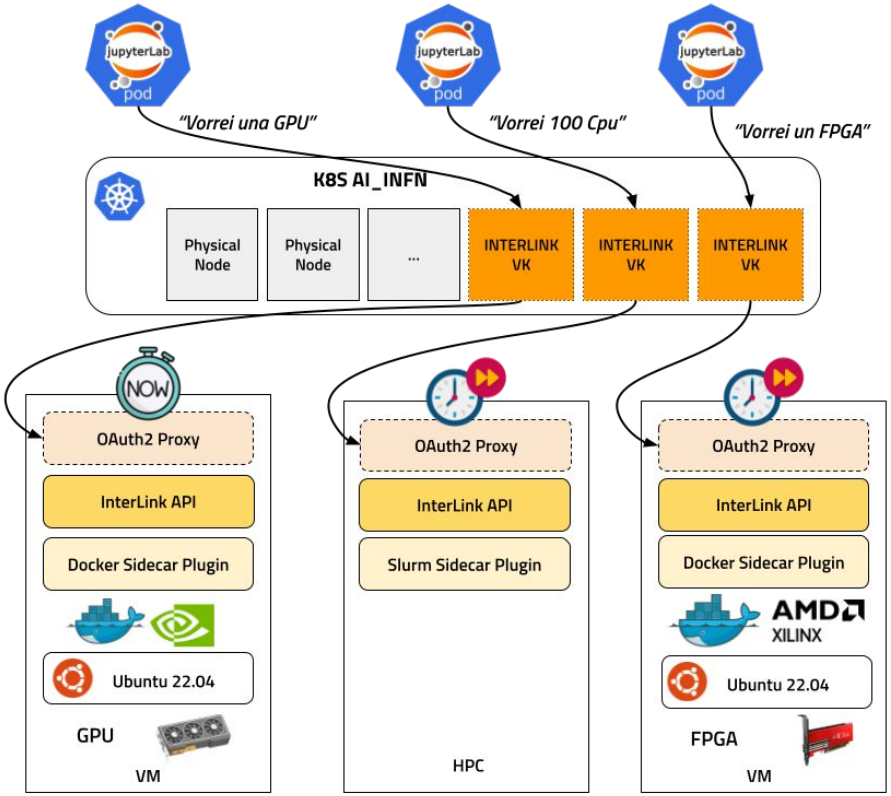
Stato dell'attività - Piattaforma AI_INFN

- La piattaforma **AI_INFN offre uno use-case cloud-native complesso** per mettere alla prova interLink con:
  - accesso interattivo tramite offloading
  - computing eterogeneo (CPU, GPU, FPGA...)

- Stiamo procedendo in parallelo con lo sviluppo di due plugin (**docker e kueue**) per dimostrare il disaccoppiamento dal backend.

- **Il plugin docker** è già stato validato per il provisioning di GPU e **può supportare analogamente il provisioning di FPGA**.

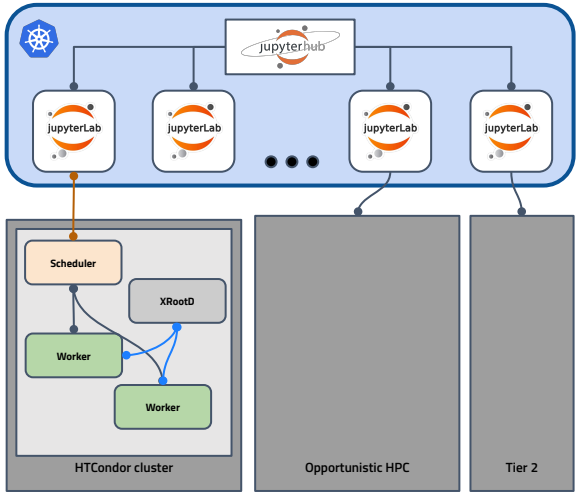
- In azioni concrete, la piattaforma AI_INFN potrebbe già sfruttare questo sistema di offloading per lo spawn di istanze JupyterLab con alcune limitazioni (NFS)



Use case - High Rate Platform

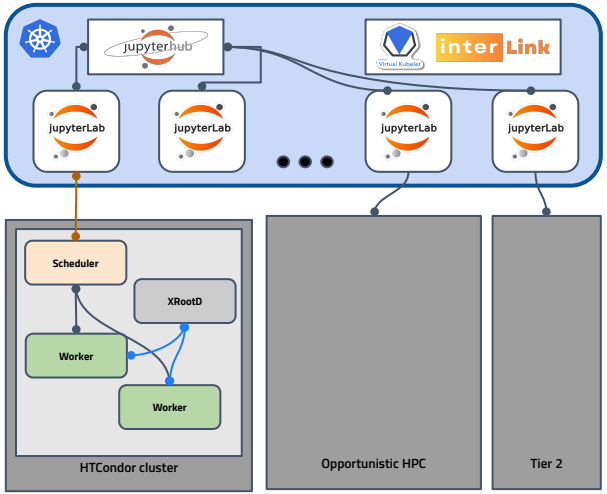
Analysis facility

- Accesso a un singolo HUB e autenticazione tramite token (INDIGO-IAM)
- Kernel Python personalizzabile
- Area di lavoro completamente containerizzabile
- Overlay basato su **HTCondor**
- Libreria DASK (Python) per il calcolo distribuito
  - scala l'esecuzione da 1 a N cores
- Possibile implementazione su risorse eterogenee
- Accesso ai dati configurabile con WLCG



high-end compute server

Inizialmente, i worker node del pool condor istanziati tramite Docker compose su risorse dedicate. L'obiettivo e' scalare andando oltre i nodi dedicati.



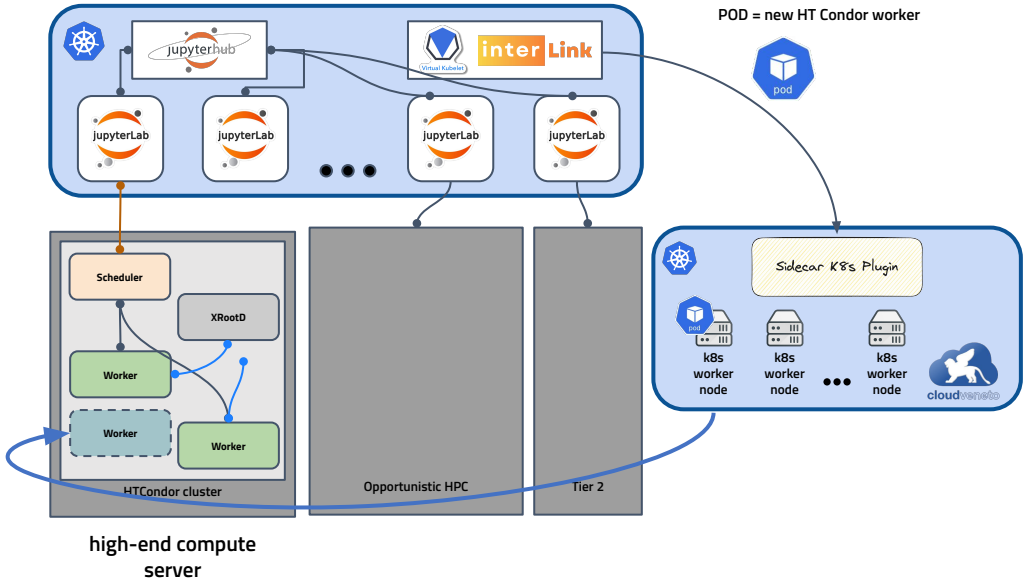
high-end compute server

InterLink per estendere il pool HT Condor e aumentare in modo opportunistico il numero di worker node a disposizione sfruttando back-end distribuiti parallelamente e geograficamente, ad esempio le risorse di CloudVeneto

Stato dell'attività - High Rate Platform

k8s plugin repository

E' stato sviluppato il **plugin sidecar kubernetes**. Il POD sottomesso al VK del cluster k8s della Analysis Facility diventa un worker node che si aggiunge al pool centrale HT Condor sfruttando le risorse di un cluster K8s che utilizza risorse di CloudVeneto.



Stato

- **Workflow completamente funzionante.**
  - nel pool condor vengono aggiunti dei nodi CloudVeneto che possono essere utilizzati per mandare job condor
  - Molteplici VK coesistono nello stesso k8s di AF e fanno offloading su diversi provider:
    - VK dedicato al tier2 di Legnaro, Bari e Pisa (HTCondor sidecar)
    - VK dedicato al tier2 di Roma (ARC sidecar)
- **Sidecar plugin k8s**
  - Supporta provisioning di CVMFS

Summary e prossimi passi

- L'attività di test del sistema di offloading è stata avviata. I primi use cases sono stati identificati. La selezione è stata fatta in base alle caratteristiche dei workflow.
  - questo ci permette di testare le funzionalità necessarie
  - i test saranno importanti

- Il sistema di offloading è in fase di integrazione nel **PoC di ICSC e TERABit** per l'integrazione delle risorse **Leonardo** (slurm)
  - dopo una validazione tecnica dovremo identificare use case per testare questo setup

- In sinergia con WP5 è stato definito il primo requirement per integrare l'offloading nei servizi di alto livello gestiti con **INDIGO PaaS Orchestrator**
  - l'offloading diventerà una "feature" gestita automaticamente dall'Orchestratore

Grazie per l'attenzione