# Le attività del T2 di Napoli con Kubernetes

Adelina D'Onofrio, Gianluca Sabella, Bernardino Spisso

# The Cloud infrastructure

## Cluster

- Kubernetes v1.28.3 (clients) - v1.23.7 (server)· 6 nodes (1 master + 5 worker)
- Runtime: Docker 26.1.3
- Networking: Calico [1] CNI (veth_mtu=1440), NGINX Ingress.
- Storage: OpenEBS LocalPV

## Master (x1)

- OS: AlmaLinux 9.4 · Kernel 5.14.0-427
- Allocatable[2]: ~8 vCPU · ~15 GiB RAM
- Disk: ~50 GB XFS
- CPU: Intel Xeon (CascadeLake)

## Workers (x5)

- OS: AlmaLinux 9.4 · Kernel 5.14.0-427
- Typical Allocatable: ~32 vCPU · ~62 GiB RAM
- Typical disk: ~100 GB XFS

[1] Calico (CNI): Pod-to-Pod Networking + NetworkPolicy; Transport: BGP or overlay (IP-in-IP/VXLAN); In the active overlay cluster, Pod MTU ≈ 1440.

[2] Allocatable = Capacity − reserves for kubelet/OS/eviction/pod overhead. This is the value used by the scheduler.

[3]

# The Bare Metal infrastructure

## Cluster

- Kubernetes v1.31.0 (clients) - v1.32.6 (server)· 6 nodes (1 master + 5 worker)
- Runtime: Docker 26.1.3
- Networking: Calico CNI (veth_mtu=1440), NGINX Ingress.
- Storage: OpenEBS LocalPV

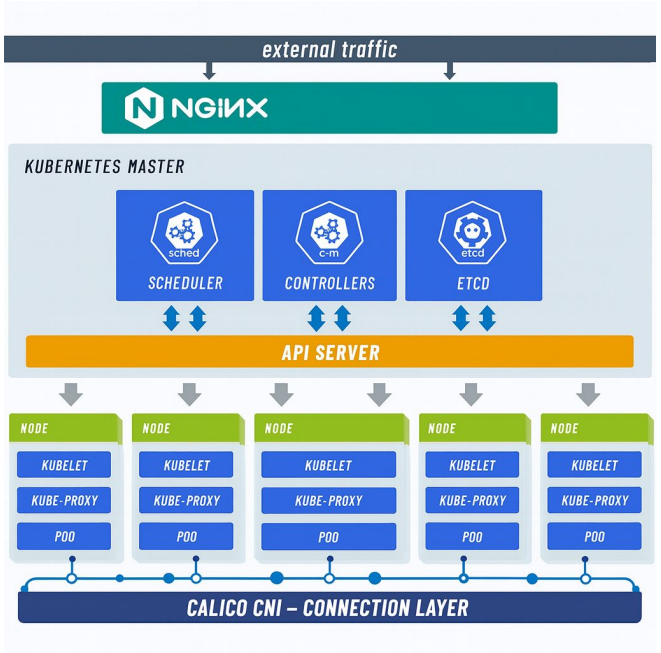
## Master (x1)

- OS: AlmaLinux 9.7 (Moss Jungle Cat) · Kernel 5.14.0-611.5.1.el9_7.x86_64
- Allocatable: ~64 vCPU · ~219 GiB RAM
- Disk: ~1.1 TB XFS
- CPU: Intel Xeon E5-4620 @ 2.20GHz (8 core/socket · 4 socket)

## Workers (x6)

- AlmaLinux 9.7 (5.14.0-611), Docker 24.0.9
- Allocatable: 32 vCPU (2×16)
- Disk: 251 GiB RAM · RAID1 460-894 GiB ·
- CPU: AMD EPYC 7351

# Infrastructure Implementation & Tooling

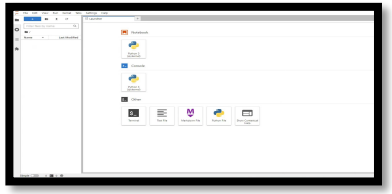


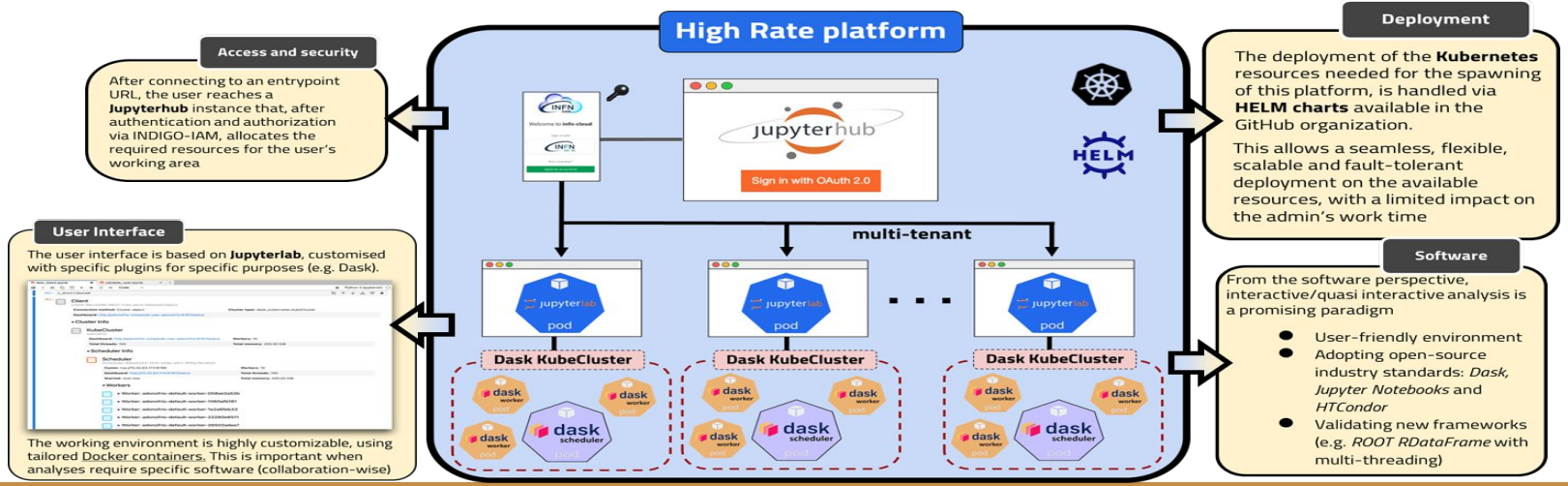

- **Orchestration**: Kubernetes via RKE (*deterministic cluster deployment)*
- **Configuration Management**: Helm (*version control, application reproducibility*)
- **Networking**: Calico CNI (*tunable MTU for fair overlay/physical comparison*)
- **Ingress**: Nginx (*external traffic routing*)

**Why these choices: Reproducibility, automation, fair infrastructure comparison**

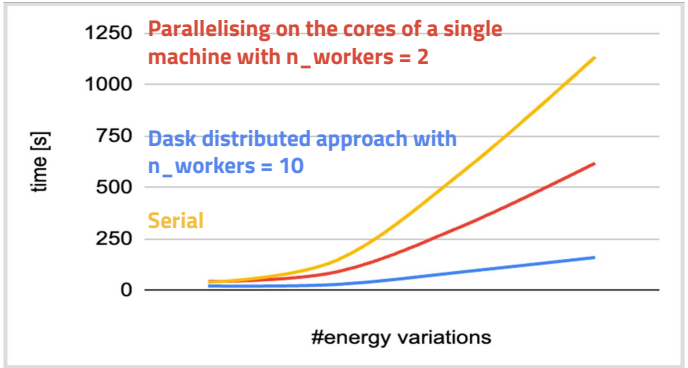
# Unified Analytics Platform

- Custom JupyterHub platform (by ICSC SPOKE 2)
- Integrated with Dask (distributed execution), HTCondor (autoscaling), ROOT RDataFrame
- Supports interactive analysis and large-scale batch jobs



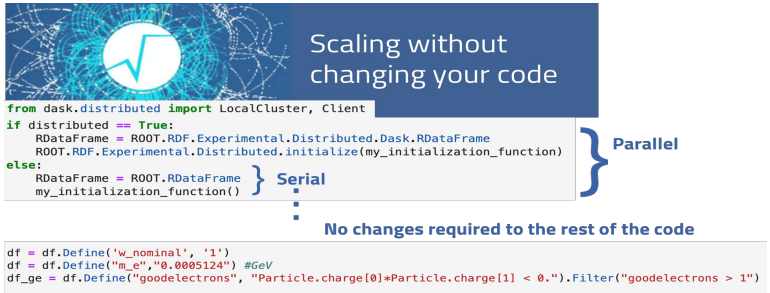


# Use Case 1: $e^+e^-$ Simulation & Event Processing - Cloud




| Defined Metric | |
|---|---|
| **Overall execution time** | Time elapsed from the start of the execution (execution triggered) to the end of execution |

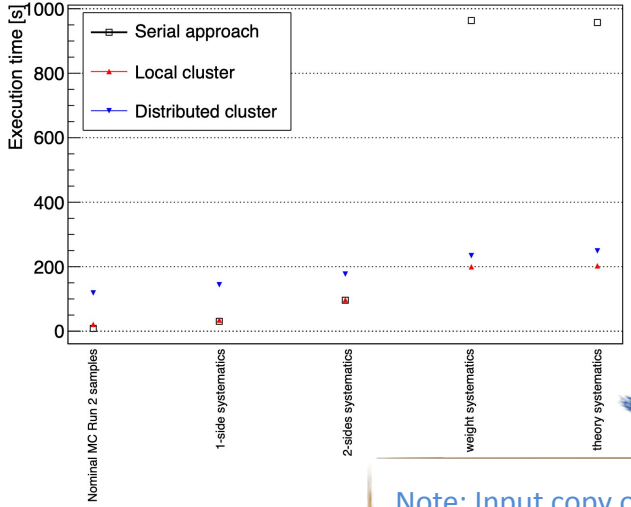
| # iterations | Serial approach | Local client Dask | Distributed Dask |
|---|---|---|---|
| 50 | 590 s | 320 s | 75 s |
| 100 | 1135 s | 618 s | 138 s |




- ✔ Exploiting the local client approach, the execution time improves *wrt* the standard/serial approach if we iterate over a significative number of energy variations ( > 10)
- ✔ Moving to a distributed Dask model and **scaling resources, the performance improves**

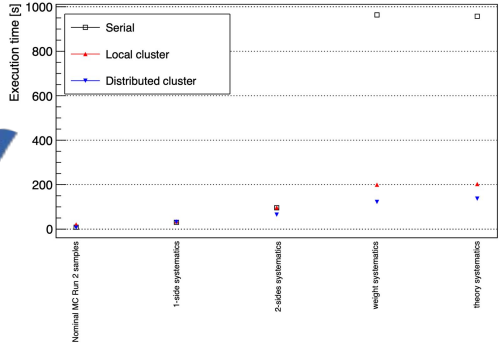
# Use Case 2: High-Rate Distributed Analysis (ATLAS) - Cloud

**Execution time including spent to copy the inputs to the worker nodes (~120 s)**




- **ATLAS-derived analysis workflow**
- Five-fold speedup at high systematic complexity
- Distributed approach amortizes overhead for realistic workloads

**Execution time not including spent to copy the inputs to the worker nodes (~120 s)**




Note: Input copy overhead (~120s) is a limiting factor for distributed approach. Optimization ongoing.

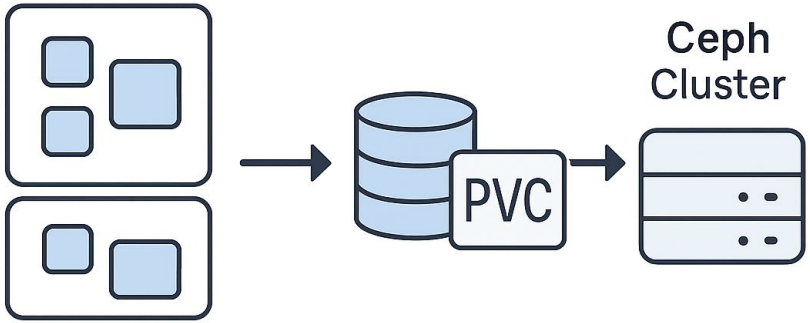
kubernetes + ceph

## Kubernetes e Storage Persistente

- Kubernetes gestisce applicazioni containerizzate in modo scalabile.
- I Pod sono **effimeri** → serve storage persistente per dati.
- Soluzione: **Persistent Volumes (PV)** e **Persistent Volume Claims (PVC)**.
- Necessità di un backend affidabile per storage distribuito.

# Perché usare Ceph con Kubernetes

- Ceph fornisce:
  - **Block Storage (RBD)** per PVC
  - **File System (CephFS)** per condivisione tra Pod
  - **Object Storage (RGW)** per servizi cloud-native
- Integrazione tramite **CSI Driver** → provisioning dinamico.
- Benefici:
  - Scalabilità orizzontale
  - Replica e resilienza
  - Multi-tenant e sicurezza



# Architettura di Integrazione

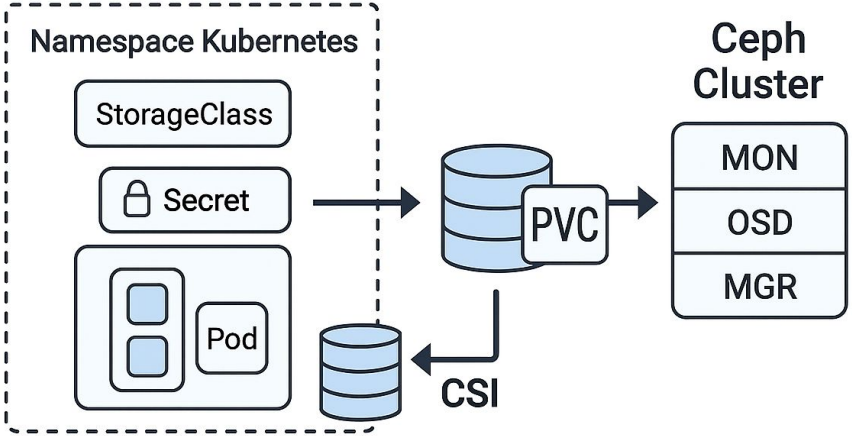
- **Ceph Cluster**: MON, OSD, MGR.
- **Kubernetes**:
  - StorageClass definisce il tipo di volume.
  - PVC richiesto dall'app → CSI crea volume Ceph.
- Comunicazione sicura tramite **Secret** per credenziali Ceph.
- Deployment tipico: **Ceph-Operator** o Helm chart.

## Testbed attuale

- **ceph-dev-mds01**
  - 2x Xeon E5-2650
- **ceph-dev-mds02**
  - 2x Xeon E5-2650L
- **ceph-dev-monmgr01**
  - 2x AMD Opteron 6376
- **ceph-dev-monmgr02**
  - 2x AMD EPYC 7351
- **ceph-dev-monmgr03**
  - 2x AMD EPYC 7351
- **ceph-dev-osd[0-4]**
  - 2x Xeon E5-2640 v4
  - **12x 8TB SAS HDD**

# Architettura di Integrazione

- **Namespace dedicato** per Ceph.
- **RBAC** per controllare accessi.
- **Monitoraggio**: Dashboard Ceph + metriche Kubernetes.



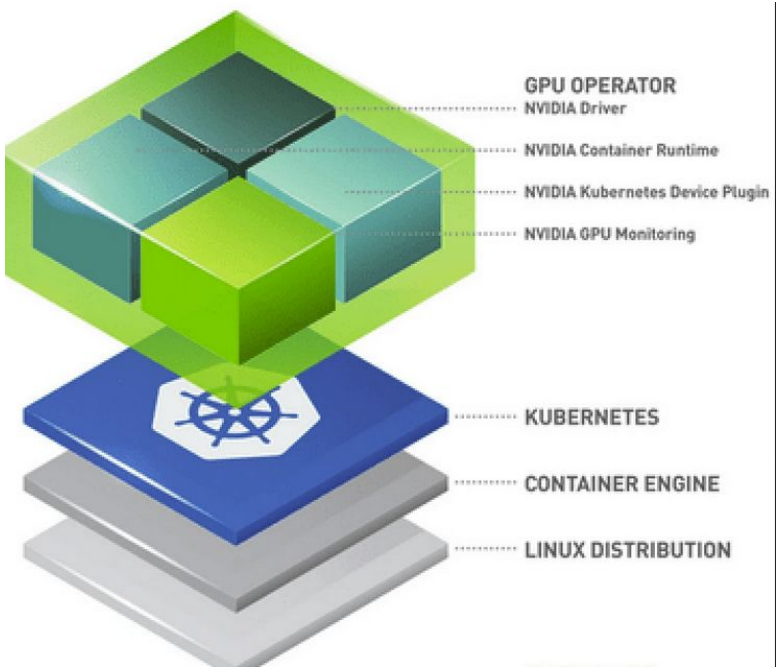

PVC → CSI → Volume Ceph

kubernetes + NVIDIA®

- Le GPU sono essenziali per accelerare AI, deep learning e rendering, migliorando le prestazioni dei carichi di lavoro complessi.
- Kubernetes non gestisce nativamente le GPU, limitando l'efficienza per applicazioni che richiedono accelerazione hardware.
- Integrare GPU in Kubernetes permette scalabilità, automazione e gestione efficiente delle risorse per workload AI.
- L'uso di GPU in Kubernetes consente la gestione flessibile delle risorse nei container

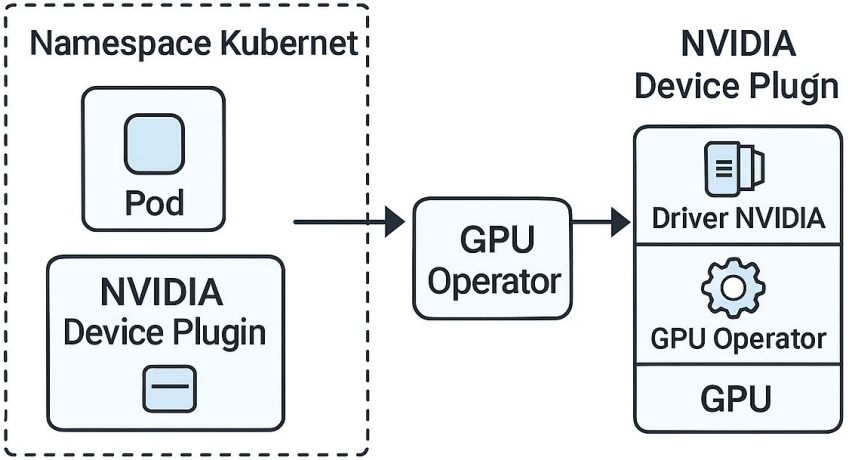
# NVIDIA GPU Operator

- **Cos'è il GPU Operator?**
  - Un Kubernetes Operator che automatizza la gestione delle GPU NVIDIA.
- **Funzioni principali:**
  - Deploy del Device Plugin per esporre GPU come risorse
  - Configurazione del runtime (containerd, CRI-O)
  - Semplifica l'integrazione delle GPU in cluster Kubernetes
  - Supporta AI/ML, HPC e workload accelerati



# NVIDIA GPU Operator

- Il GPU Operator, il Device Plugin NVIDIA e il runtime container permettono l'utilizzo della GPU  da parte di uno specifico  pod
- Il Pod richiede GPU specifiche; lo scheduler assegna il Pod a un nodo con GPU disponibili per l'esecuzione.
- Il GPU Operator garantisce il monitoraggio continuo delle GPU nel cluster.




Pod → Scheduler → GPU Node

# Conclusioni

- 2x cluster Kubernetes di testbed fisico e virtuale su cui  tramite  il deploy “Analysis Facility ” stiamo testando use case di ATLAS
- Abbiamo un testbed Ceph Funzionate che ci apprestiamo ad intyderate in Kubernetes
- Stiamo approntando un nodo  Kubernetes provvisto di GPU per sperimentare l'integrazione con i pod

# Grazie