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

Flash Simulation Flagship

Offloading Flash Simulation steps with InterLink

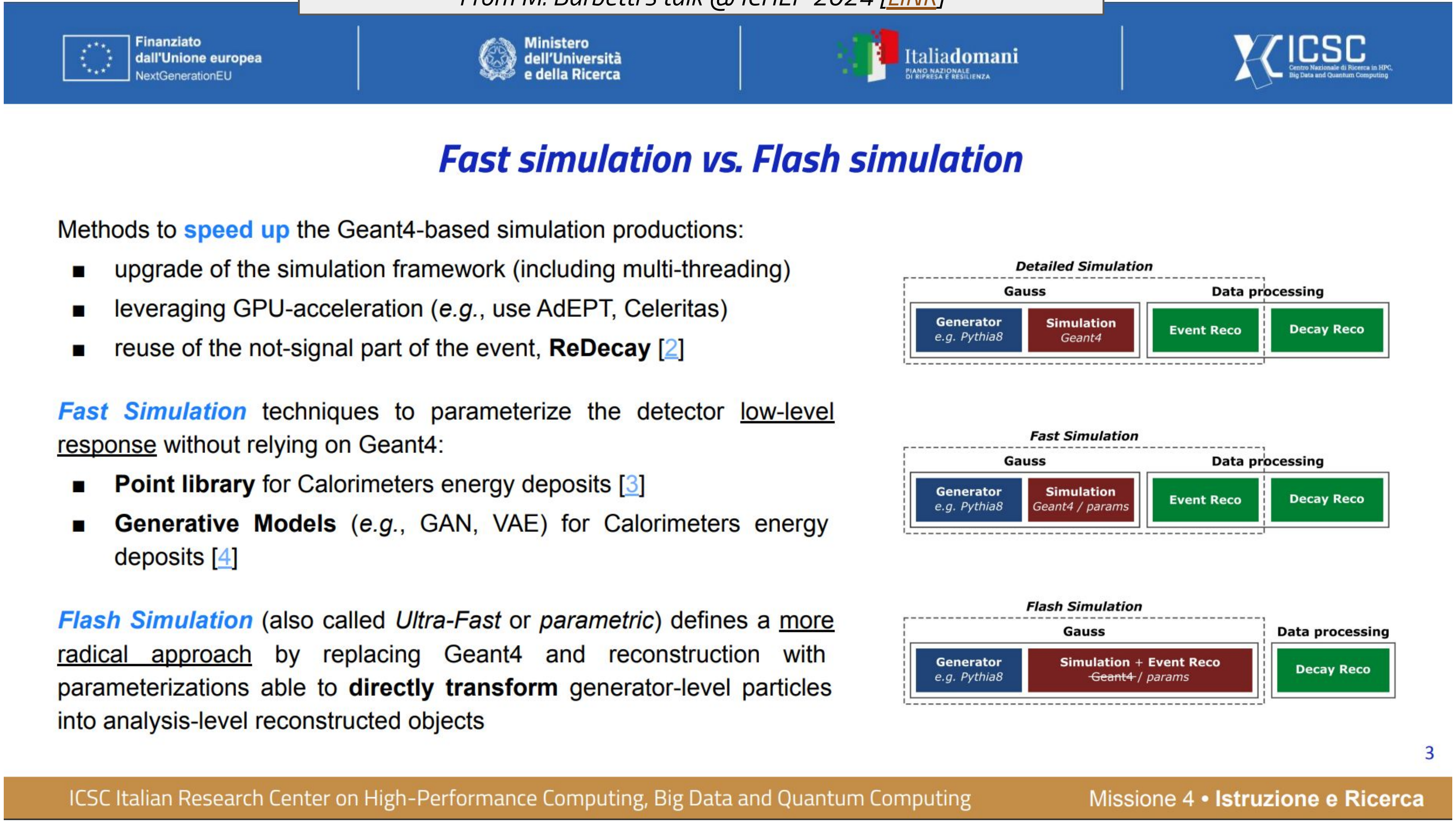
Lucio Anderlini

Istituto Nazionale di Fisica Nucleare, Sezione di Firenze

External Partner

Flash Simulation

From M. Barbetti's talk @ ICHEP 2024 [LINK]



Fast simulation vs. Flash simulation

Methods to **speed up** the Geant4-based simulation productions:

- upgrade of the simulation framework (including multi-threading)
- leveraging GPU-acceleration (*e.g.*, use AdEPT, Celeritas)
- reuse of the not-signal part of the event, **ReDecay** [2]

Fast Simulation techniques to parameterize the detector low-level response without relying on Geant4:

- **Point library** for Calorimeters energy deposits [3]
- **Generative Models** (*e.g.*, GAN, VAE) for Calorimeters energy deposits [4]

Flash Simulation (also called *Ultra-Fast* or *parametric*) defines a more radical approach by replacing Geant4 and reconstruction with parameterizations able to **directly transform** generator-level particles into analysis-level reconstructed objects

Detailed Simulation

Gauss: Generator *e.g. Pythia8* → Simulation *Geant4*

Data processing: Event Reco → Decay Reco

Fast Simulation

Gauss: Generator *e.g. Pythia8* → Simulation *Geant4 / params*

Data processing: Event Reco → Decay Reco

Flash Simulation

Gauss: Generator *e.g. Pythia8* → Simulation + Event Reco *~~Geant4~~ / params*

Data processing: Decay Reco

3

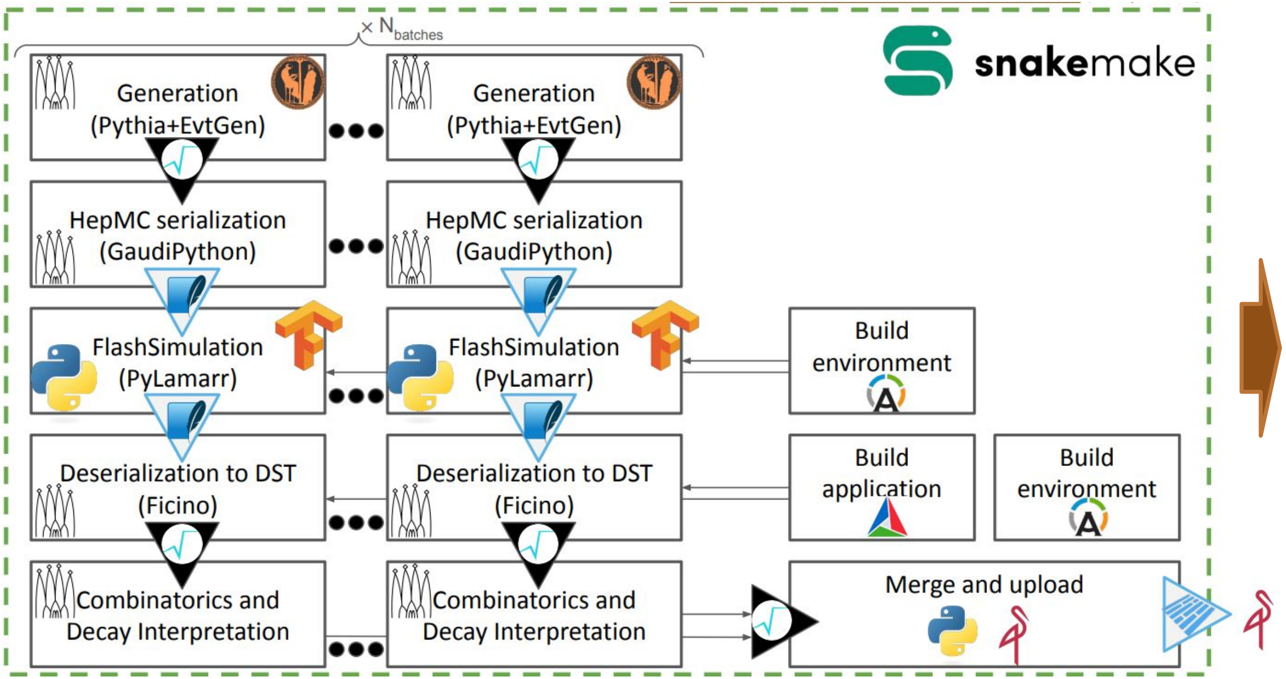
ICSC Italian Research Center on High-Performance Computing, Big Data and Quantum Computing — Missione 4 • **Istruzione e Ricerca**

Flash Simulation Workflow

Validation (mixed: HTC, HPC and Cloud)

Training

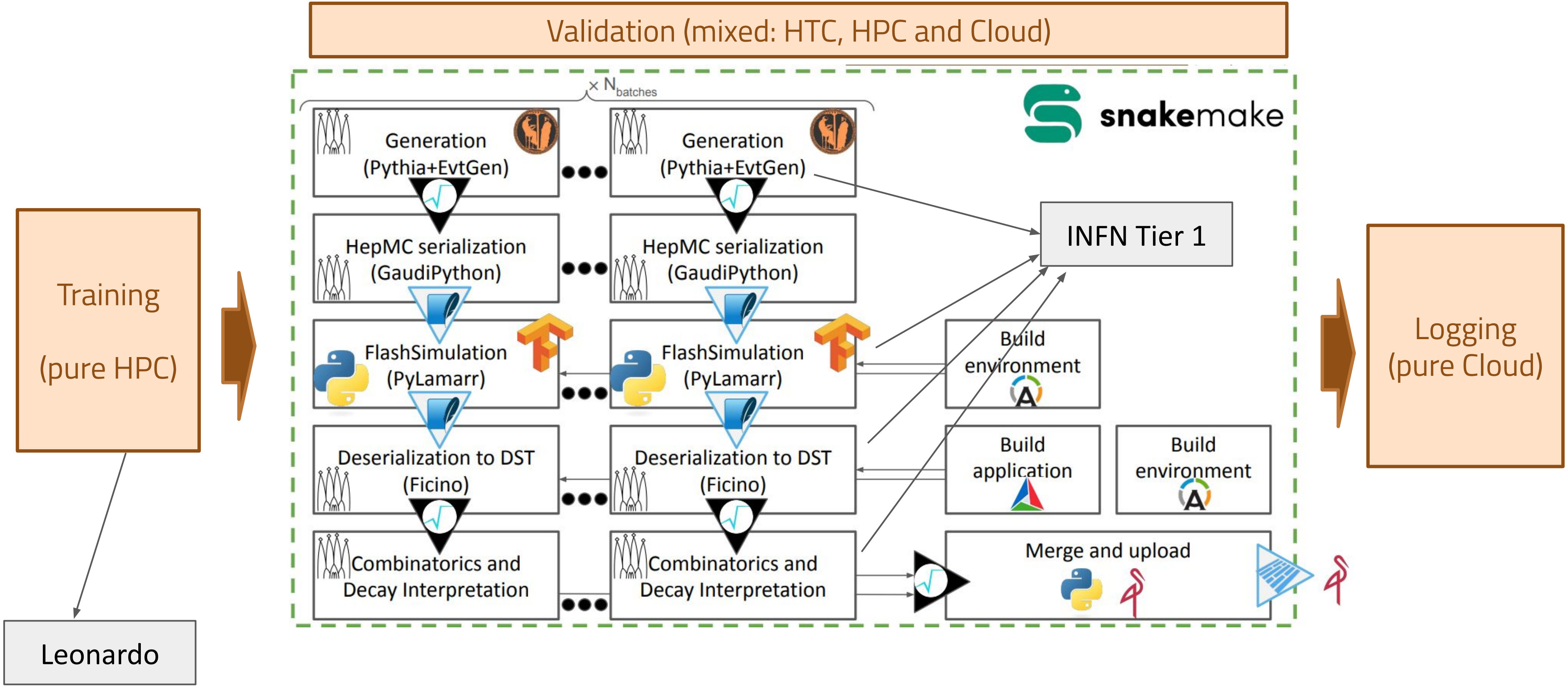
(pure HPC)


× $N_{batches}$
snakemake
Generation (Pythia+EvtGen)
Generation (Pythia+EvtGen)
HepMC serialization (GaudiPython)
HepMC serialization (GaudiPython)
FlashSimulation (PyLamarr)
FlashSimulation (PyLamarr)
Build environment
Deserialization to DST (Ficino)
Deserialization to DST (Ficino)
Build application
Build environment
Combinatorics and Decay Interpretation
Combinatorics and Decay Interpretation
Merge and upload

Logging (pure Cloud)

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

Flash Simulation Workflow – offloaded



Validation (mixed: HTC, HPC and Cloud)

Training

(pure HPC)

Leonardo

Logging
(pure Cloud)

× N_{batches}

snakemake

Generation (Pythia+EvtGen)

HepMC serialization (GaudiPython)

FlashSimulation (PyLamarr)

Deserialization to DST (Ficino)

Combinatorics and Decay Interpretation

INFN Tier 1

Build environment

Build application

Build environment

Merge and upload

Status of the integration of INFN-T1 resources

landerlini/interlink-condorce-plugin

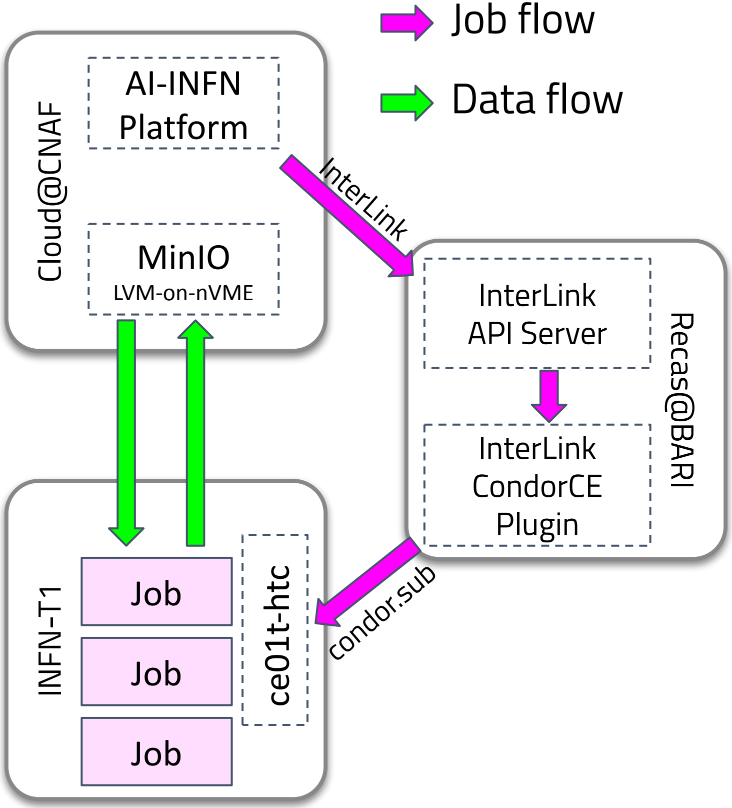
Developing the **HERD Computing Model**,
CNAF defined a CondorCE submitting jobs from remote locations through authentication.

Unfortunately, the CondorCE is not reachable from Cloud@CNAF for network policies, but it is, for example, from ReCaS@BARI.

We developed an **InterLink plugin** sitting in a VM in Bari, accepting InterLink submissions from Cloud@CNAF and forwarding them to CNAF Tier-1 test CE.

The plugin converts the **Kubernetes Pod** specifications into a (possibly rather long) shell script running **Apptainer** containers in multiple subprocesses.

Input and output data is managed through a self-managed **MinIO instance on LVM-on-nVME** hosted in **Cloud@CNAF**.

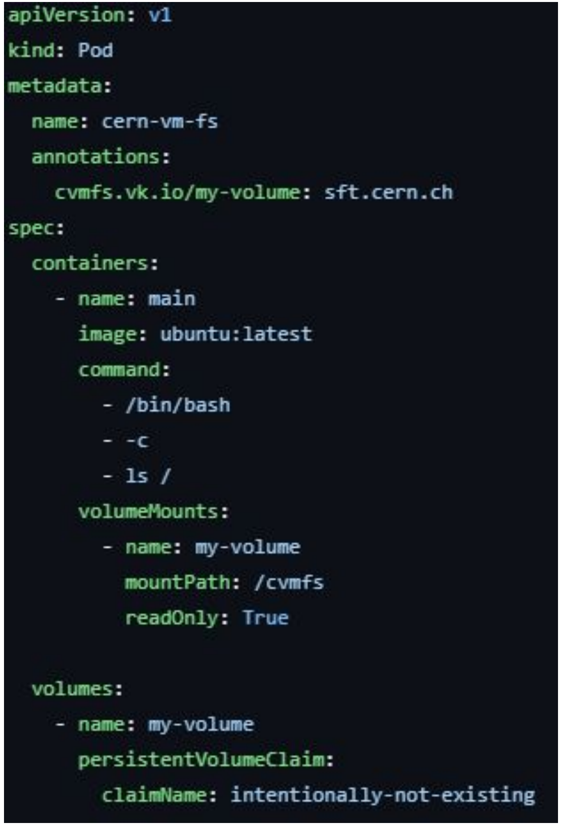


(re)Defining cvmfs and fuse volumes

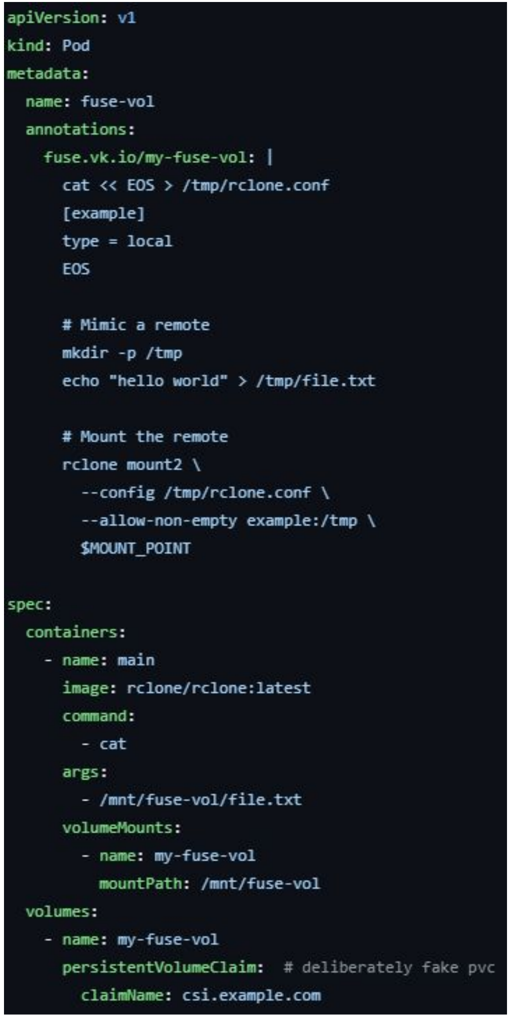
Converting Pod's requests to access cvmfs or fuse data should be responsibility of the plugin, as different compute backend may be subject to different rules.

In CondorCE plugin I use generic annotations to define volumes.

For Leonardo, this require hacking the singularity submission command (very verbose).



```
apiVersion: v1
kind: Pod
metadata:
  name: cern-vm-fs
  annotations:
    cvmfs.vk.io/my-volume: sft.cern.ch
spec:
  containers:
    - name: main
      image: ubuntu:latest
      command:
        - /bin/bash
        - -c
        - ls /
      volumeMounts:
        - name: my-volume
          mountPath: /cvmfs
          readOnly: True

  volumes:
    - name: my-volume
      persistentVolumeClaim:
        claimName: intentionally-not-existing
```



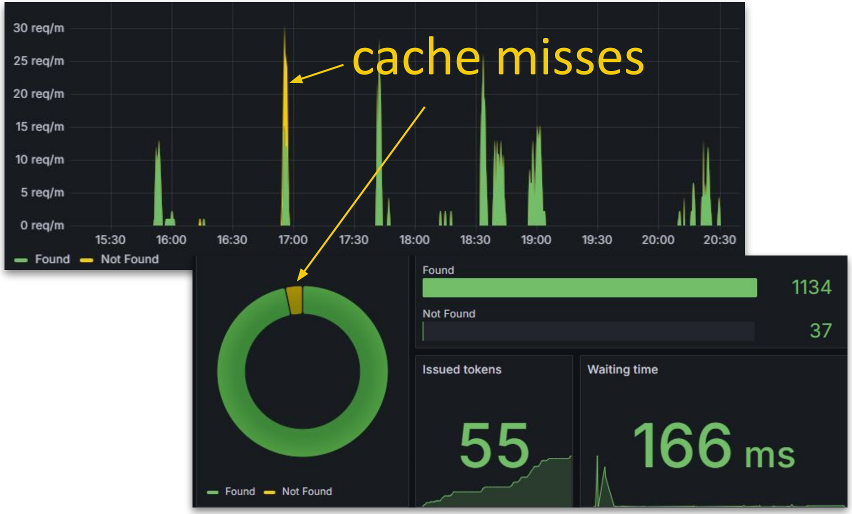
```
apiVersion: v1
kind: Pod
metadata:
  name: fuse-vol
  annotations:
    fuse.vk.io/my-fuse-vol: |
      cat << EOS > /tmp/rclone.conf
      [example]
      type = local
      EOS

      # Mimic a remote
      mkdir -p /tmp
      echo "hello world" > /tmp/file.txt

      # Mount the remote
      rclone mount2 \
        --config /tmp/rclone.conf \
        --allow-non-empty example:/tmp \
        $MOUNT_POINT

spec:
  containers:
    - name: main
      image: rclone/rclone:latest
      command:
        - cat
      args:
        - /mnt/fuse-vol/file.txt
      volumeMounts:
        - name: my-fuse-vol
          mountPath: /mnt/fuse-vol
  volumes:
    - name: my-fuse-vol
      persistentVolumeClaim:  # deliberately fake pvc
        claimName: csi.example.com
```

Solving the distributed cache problem

Focus on data flow

A **shared virtual file system** is mounted by the condor nodes with fuse using JuiceFS.

JuiceFS falls back on **MinIO** for the data and **Redis** (part of the AI-INFN platform) for the metadata

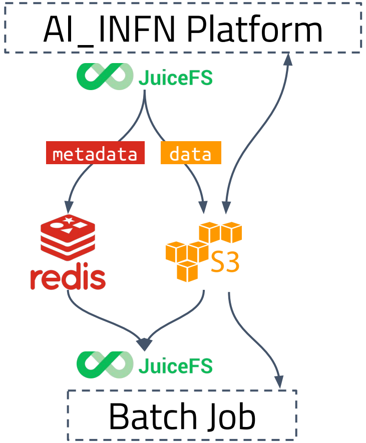


Found 1134

Not Found 37

Issued tokens 55

Waiting time 166 ms



AI_INFN Platform

JuiceFS

metadata

data

redis

S3

JuiceFS

Batch Job

[ShubProxy]

Downloading and building docker images into SIF for each jobs

- ➢ would cause a periodic bans of CNAF by DockerHub;
- ➢ cause large inefficiency in short jobs.

We deployed a simple web application defining a shared cache.

If the image is not available in S3, the web app schedule its build, otherwise it return the built artifact from cache.

Status of the integration with Leonardo

The slurm plugin in production in Leonardo, does not accept Pod requests from the Flash Simulation workflow.

All the building blocks were tested separately and we expect no fundamental reason for the plugin not to work.

Still, some polishing would be needed, probably in a joint debugging session.

Alternatively, we may try to use the CondorCE plugin submitting to slurm.