FENIX

RESEARCH INFRASTRUCTURE

Interactive computing services in Fenix

May 10th, 2023

Fabio Pitari (CINECA)
f.pitari@cineca.it

The ICEI project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No 800858.

Overview

- What's interactive computing (IAC)
- ICEI interactive computing implementation (Cineca)
- Additional interactive computing services in Fenix sites:
  - IAC at CSCS
  - IAC at JSC
  - IAC at BSC
  - IAC at CEA

Traditional approach to HPC resources

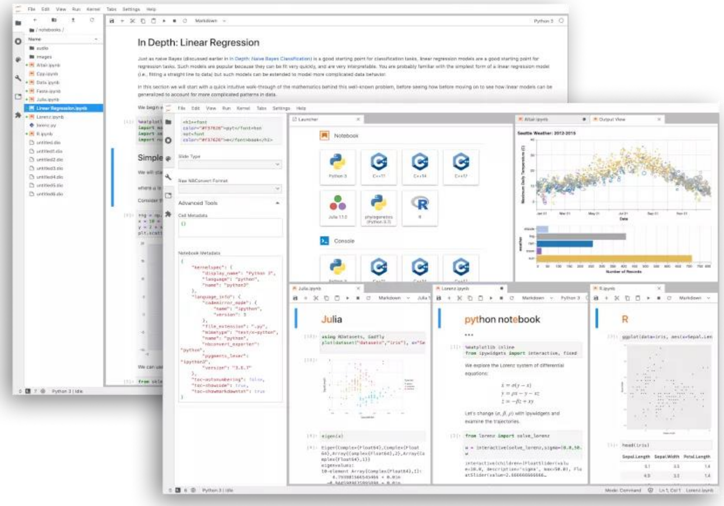
- Command line interface on remote machine
- Scheduler for jobs running the core of the computation
  - Queue-based system
  - Output is obtained at the end of the job
- Advantages:
  - Optimal sharing of resources among users
  - Large jobs can be scheduled
- Disadvantages:
  - Schedule time unkwnown, thus job results are usually checked at the end of the workflow
  - Visualization of results can't be obtained efficiently via command line
  - Command line is not user-friendly

Interactive computing approach

- Interactive Computing service is an alternative approach to the traditional access to HPC resources
- Main idea: interaction on the fly with the workflow during its execution
  - Resources are allocated to the user despite the code is running or not (near-immediate access in some implementations)
  - User can employ those resources on the fly (and not queueing them)
- Advantages:
  - User can easily check intermediate results and change the workflow accordingly
- Disadvantages:
  - Not the optimal efficiency for resources occupancy (idle resources might occur)

Jupyter

- The most common approach for interactive computing is via Jupyter (https://jupyter.org/) which grants additional features to the advantages already listed above:
  - Divide workflow into cells to be executed (=> execution can be driven on the fly)
  - Web interface (=> user-friendly, visualization tools)
  - Easily extensible (=> additional features)
- Other approaches can make use of ssh interactive sessions or remote desktop environments



Current IAC services in Fenix

- One service developed in the context of the ICEI project:
  - Currently up and running at CINECA
  - Develeloped for all the Fenix sites in case they want to implement the same solution (Ansible playbooks for deployment)
  - Not publicly released yet (public pre-production expected before end of May 2023)
  - Based on Jupyter + Slurm Spawner, on Galileo 100 cluster

- Four pre-existing services in Fenix sites:
  - JSC: Jupyter + Unicore spawning on multiple clusters/cloud solutions
  - BSC: Jupyter via ssh, on HSM cluster
  - CSCS: Jupyter + Slurm Spawner, on Piz Daint cluster
  - CEA: NiceDCV remote desktop session via ssh

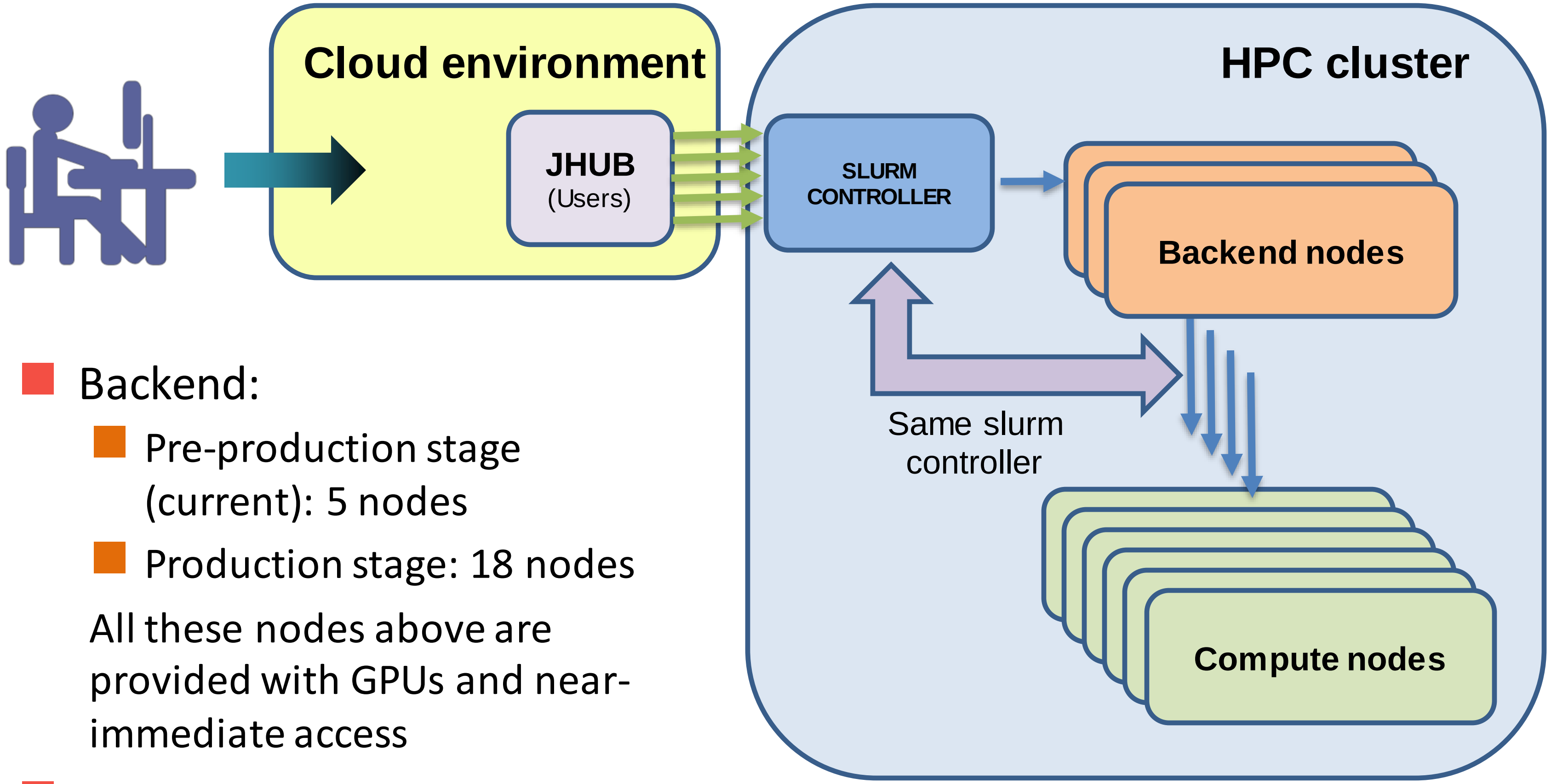
- **Please note: the only requirements for accessing to IAC services for each site is the membership in an active computing project at that site**

ICEI IAC IMPLEMENTATION

ICEI Interactive Computing implementation

- Current implementation at Cineca is deployed by E4 Computer Engineering (https://www.e4company.com/) thanks to ICEI project
- Deployment (via Ansible) is avaliable for all the Fenix sites
- The framework involves three main components:
  - a frontend layer, based on Jupyterlab
  - a backend layer, running on the HPC cluster
  - Interaction between these two layers happens behind the scenes using Slurm scheduler

Production infrastructure design



- Backend:
  - Pre-production stage (current): 5 nodes
  - Production stage: 18 nodes

  All these nodes above are provided with GPUs and near-immediate access
- Compute:
  - 460 nodes without GPUS, batch jobs

Frontend

- Jupyterlab FrontEnd
  - A (login) node with a public network interface (to expose the service)
  - Can be hosted on a bare-metal server or virtual resource
  - The latter has been chosen so far at Cineca, for several reason, e.g.:
    - Increased security
    - Being restricted, additional privileges can be given to E4 developers
    - Easy to export
    - Easy to expand
    - Different solutions can be tested, even during the production phase
  - Easy to switch to bare-metal server if needed (being delivered via ansible playbooks)
  - Interfaced with authentication services
  - Resources can be easily extended with high availability and load balancing

Frontend

- User access to a web service and is prompted with an authentication form
  - Note: this is compliant with cineca-hpc-idp (keycloak)
- Once authenticated, user is prompted to choose among a series of computational environment
  - Current Python-based envs: DASK, RAPIDS, Tensorflow, PyTorch, Life science packages
  - E4 solution integrates some other tools to the User interface
    - Visual Studio Code
    - Slurm batch interface for compute nodes
    - Remote desktop applications, I.e. VNC/Xpra (based on Jupyter remote desktop proxy (https://github.com/jupyterhub/jupyter-remote-desktop-proxy)

Requirements

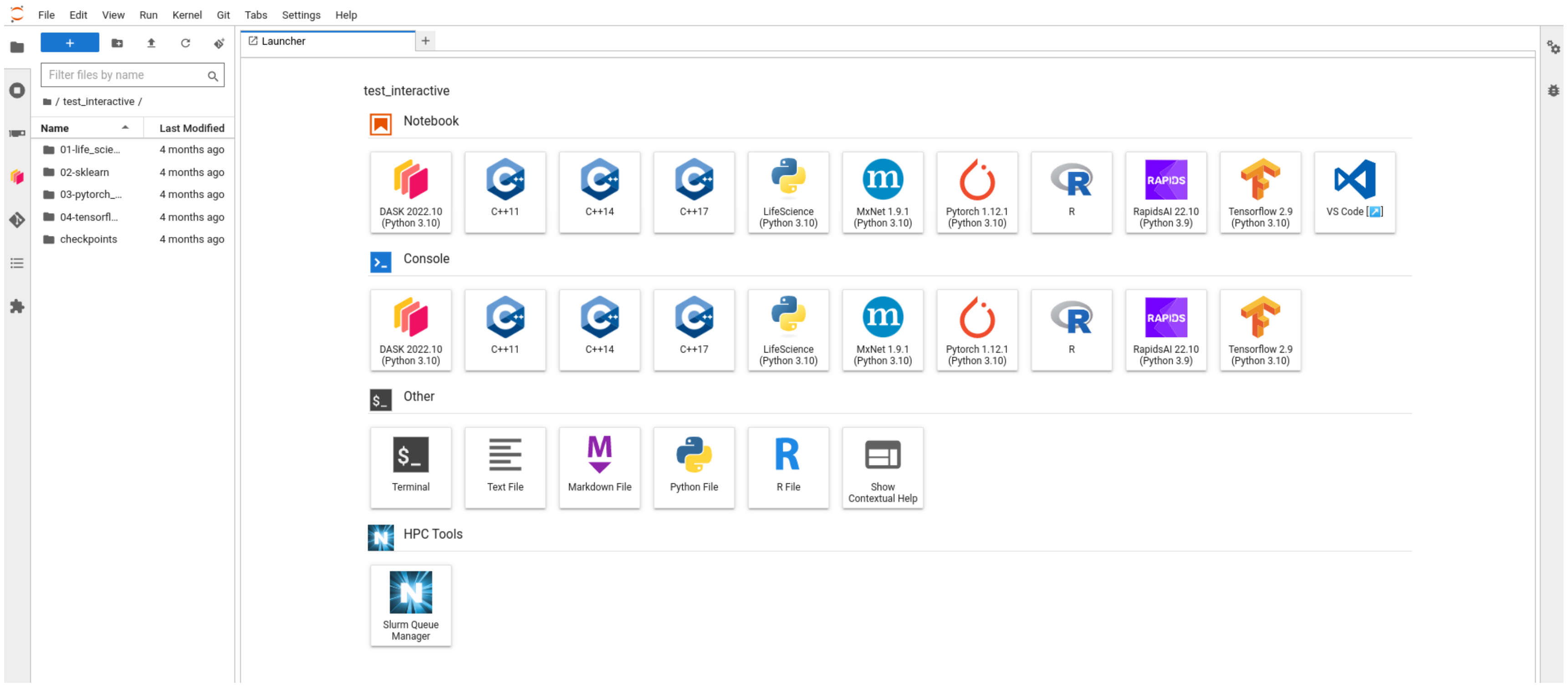
Login sessions for the users needs to be restored after the session is closed, i.e. session stays active as long as possible and can be resumed from the browser

This is achieved with a slurm partition with oversubscription:

It allows to minimize queues for users' sessions (expected only in case of huge overload) and exploits the backend nodes at most

It is defined on the dedicated partition

Frontend: layout



Name	Last Modified
01-life_scie...	4 months ago
02-sklearn	4 months ago
03-pytorch_...	4 months ago
04-tensorfl...	4 months ago
checkpoints	4 months ago

Backend

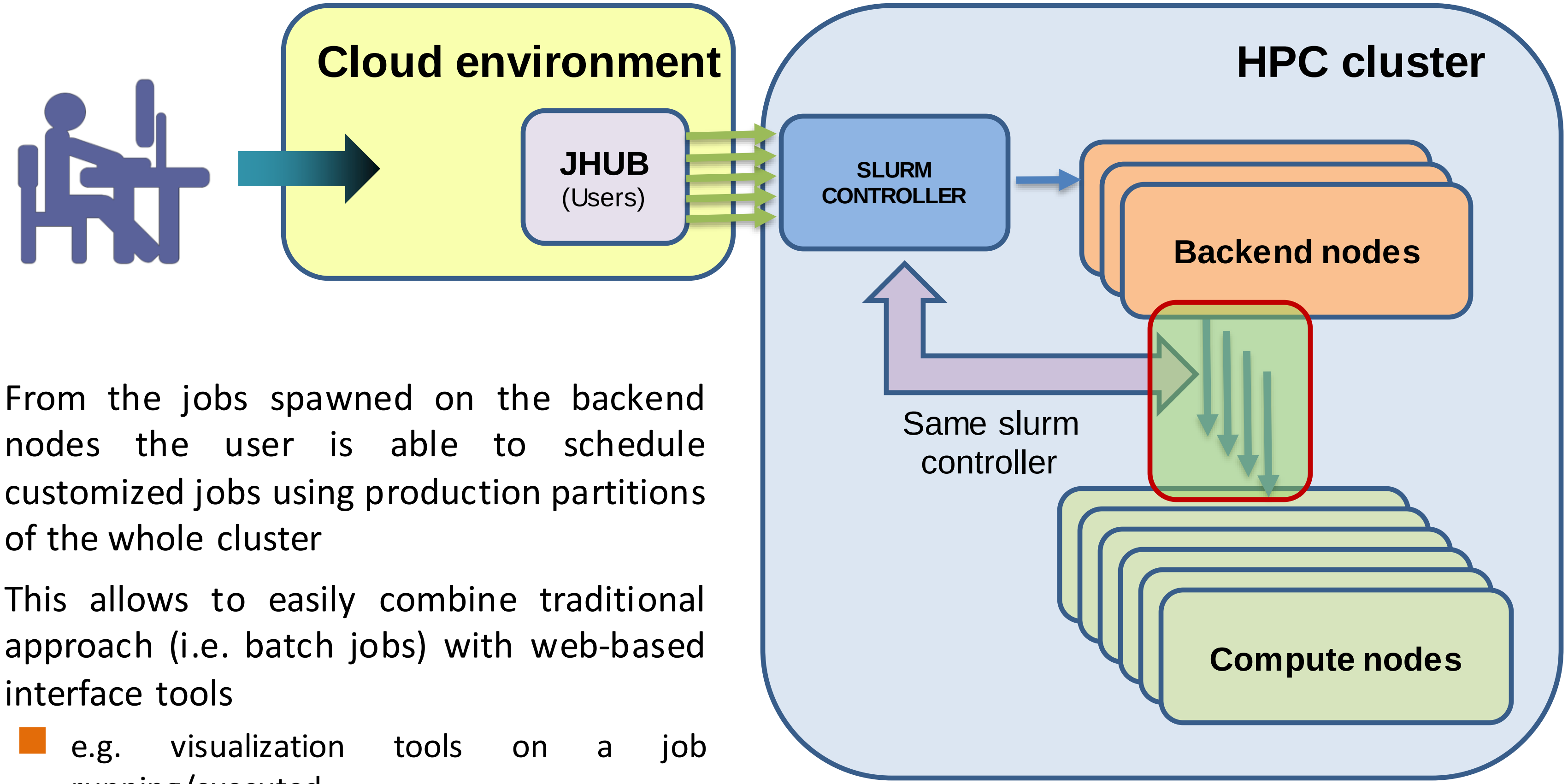
Backend nodes: actual computing nodes drained from the cluster

Comunication via frontend (VMs) and backend (bare metal) is achieved on a shared network

User sessions are handled on the backend nodes, whereas parallel tasks can be handled via compute nodes through a dedicated web interface

ssh and additional frontends for admin purpose can be given via different VMs

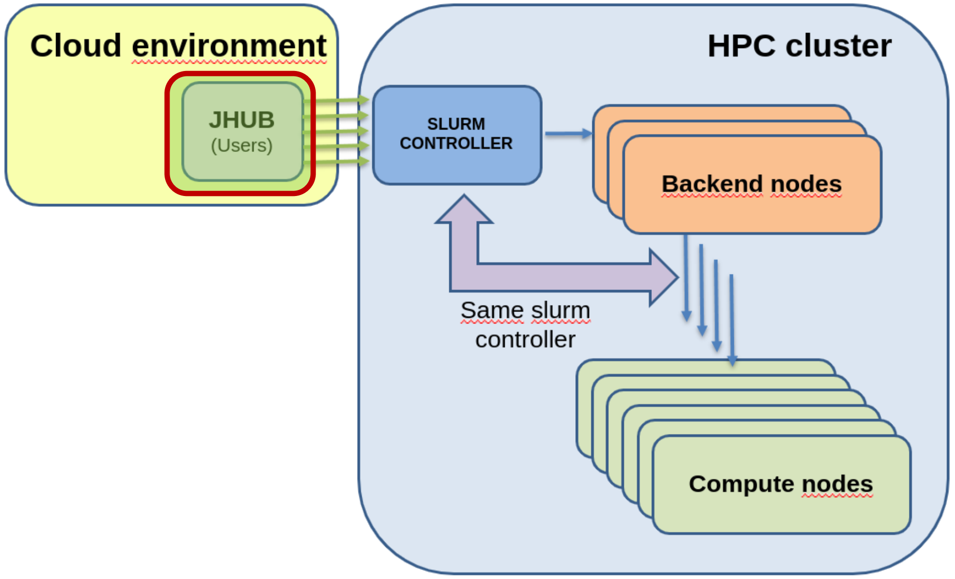
Interaction with compute nodes

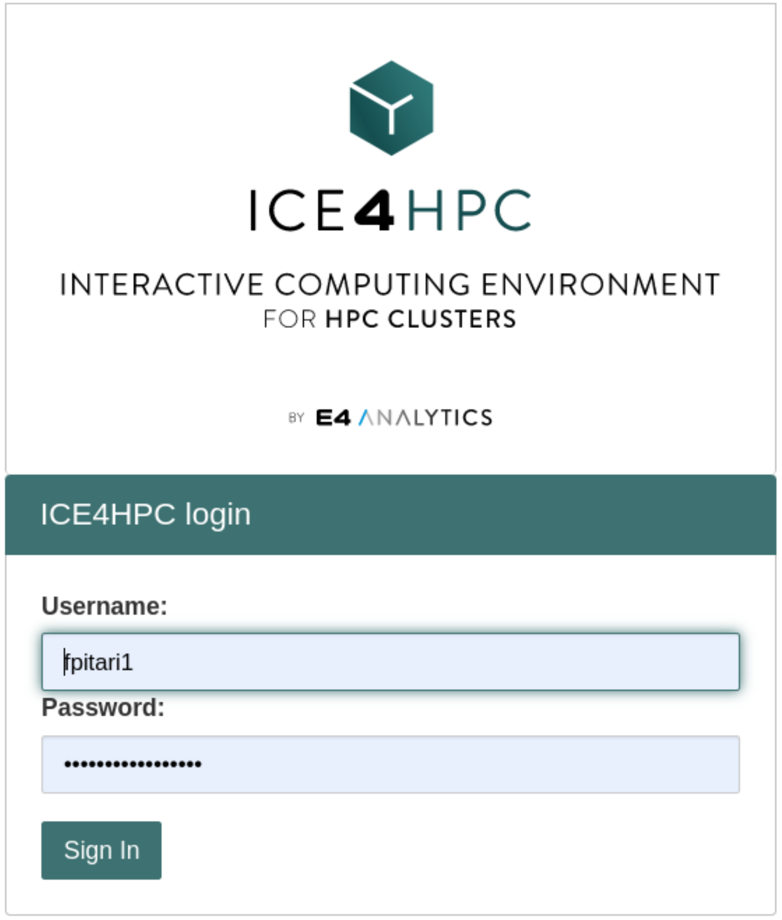


- From the jobs spawned on the backend nodes the user is able to schedule customized jobs using production partitions of the whole cluster
- This allows to easily combine traditional approach (i.e. batch jobs) with web-based interface tools
  - e.g. visualization tools on a job running/executed

Allocation sequence

- Login interface can be connected with idp authentication (e.g. keycloak)
- Allowed to every user of the cluster, or restricted to a specific subset




ICE4HPC

INTERACTIVE COMPUTING ENVIRONMENT FOR HPC CLUSTERS

BY E4 ANALYTICS

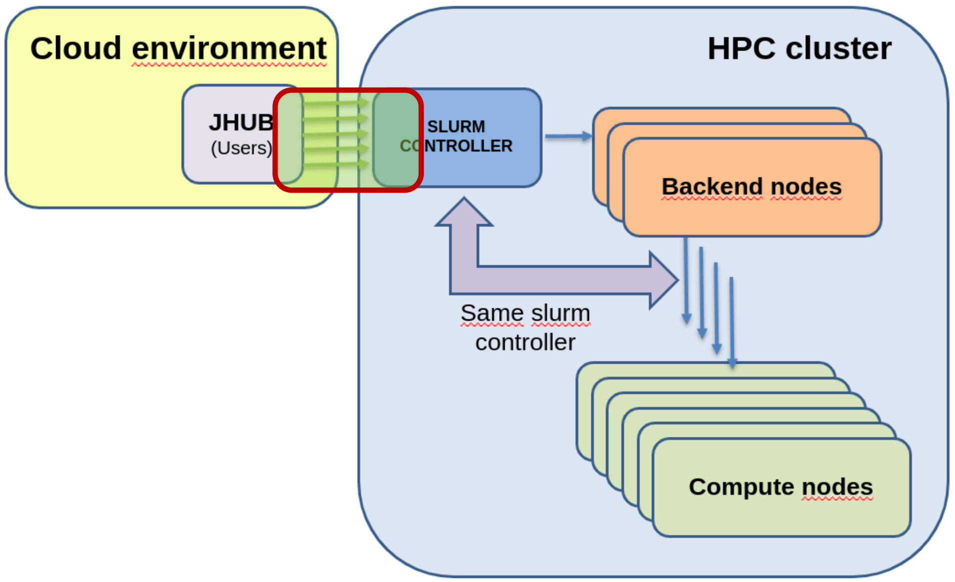
ICE4HPC login

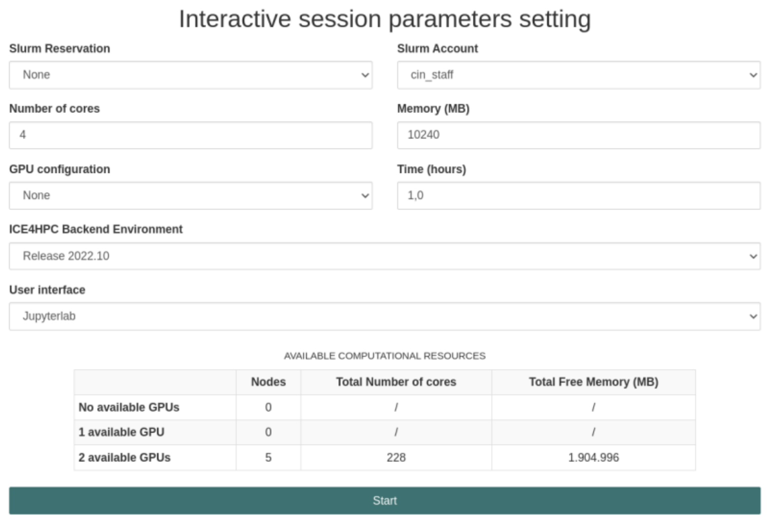
Username: fpitari1

Password: ••••••••••••••••••

Allocation sequence

- A form is displayed in order to let the user request resources
- A table is displayed parsing information about avaliable resources from the scheduler
- Session can be restored when the browser is closed, thanks to a job dispatched in oversubscription on the backend nodes by the slurm controller in the cloud environment




Interactive session parameters setting

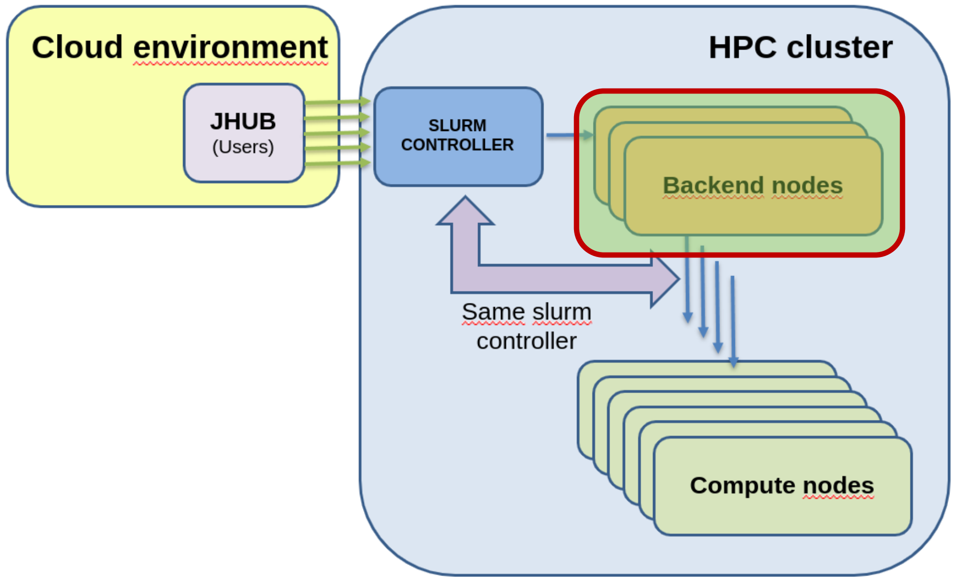
Slurm Reservation	None
Slurm Account	cin_staff
Number of cores	4
Memory (MB)	10240
GPU configuration	None
Time (hours)	1,0
ICE4HPC Backend Environment	Release 2022.10
User interface	Jupyterlab

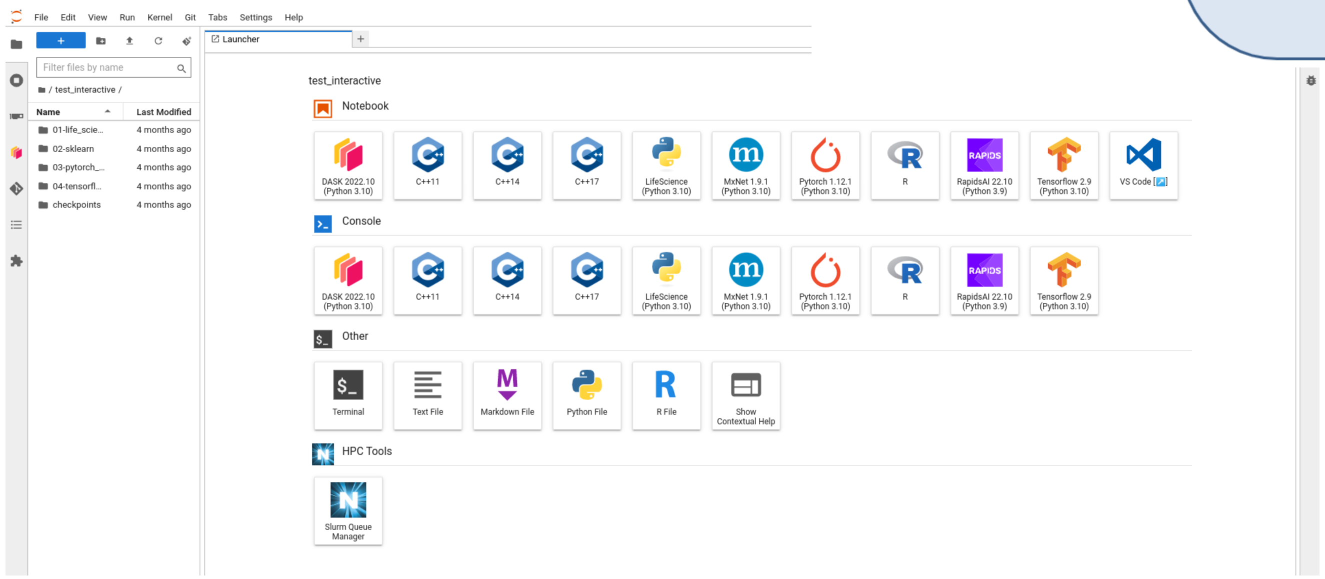
AVAILABLE COMPUTATIONAL RESOURCES

	Nodes	Total Number of cores	Total Free Memory (MB)
No available GPUs	0	/	/
1 available GPU	0	/	/
2 available GPUs	5	228	1.904.996

Allocation sequence

- Oversubsription is currently set to 5 jobs per core
- User session (i.e. backend job) expires currently in only 8 hours for the pre-production phase (to be increased to 36 hours in production)



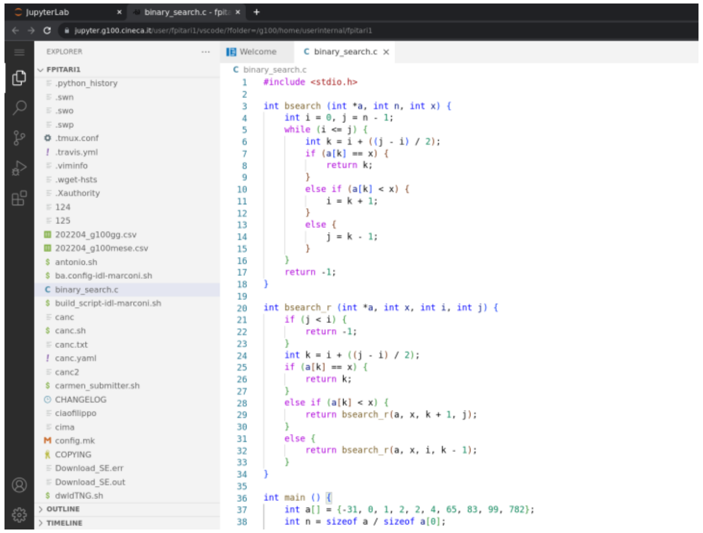

File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
Launcher
Filter files by name
test_interactive
Notebook
DASK 2022.10 (Python 3.10)
C++11
C++14
C++17
LifeScience (Python 3.10)
MxNet 1.9.1 (Python 3.10)
Pytorch 1.12.1 (Python 3.10)
R
RapidsAI 22.10 (Python 3.9)
Tensorflow 2.9 (Python 3.10)
VS Code
Console
DASK 2022.10 (Python 3.10)
C++11
C++14
C++17
LifeScience (Python 3.10)
MxNet 1.9.1 (Python 3.10)
Pytorch 1.12.1 (Python 3.10)
R
RapidsAI 22.10 (Python 3.9)
Tensorflow 2.9 (Python 3.10)
Other
Terminal
Text File
Markdown File
Python File
R File
Show Contextual Help
HPC Tools
Slurm Queue Manager

/ test_interactive /

Name	Last Modified
01-life_scie...	4 months ago
02-sklearn	4 months ago
03-pytorch_...	4 months ago
04-tensorfl...	4 months ago
checkpoints	4 months ago

Additional features

- Batch jobs can be submitted to the cluster via a dedicated interface
- Slurm queues and notifications can be also monitored
- Proxy for additional web interfaces from the launcher (currently just VSCode; R-Studio and VNC incoming)

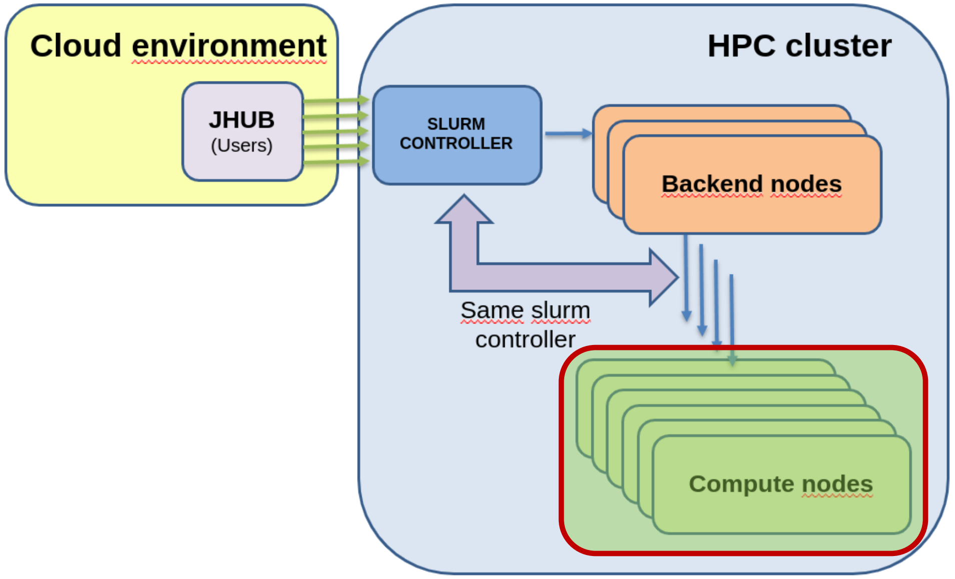


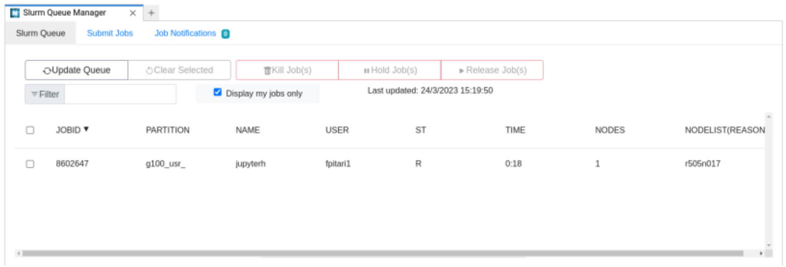
```c
#include <stdio.h>

int bsearch (int *a, int n, int x) {
    int i = 0, j = n - 1;
    while (i <= j) {
        int k = i + ((j - i) / 2);
        if (a[k] == x) {
            return k;
        }
        else if (a[k] < x) {
            i = k + 1;
        }
        else {
            j = k - 1;
        }
    }
    return -1;
}

int bsearch_r (int *a, int x, int i, int j) {
    if (j < i) {
        return -1;
    }
    int k = i + ((j - i) / 2);
    if (a[k] == x) {
        return k;
    }
    else if (a[k] < x) {
        return bsearch_r(a, x, k + 1, j);
    }
    else {
        return bsearch_r(a, x, i, k - 1);
    }
}

int main () {
    int a[] = {-31, 0, 1, 2, 2, 4, 65, 83, 99, 782};
    int n = sizeof a / sizeof a[0];
```

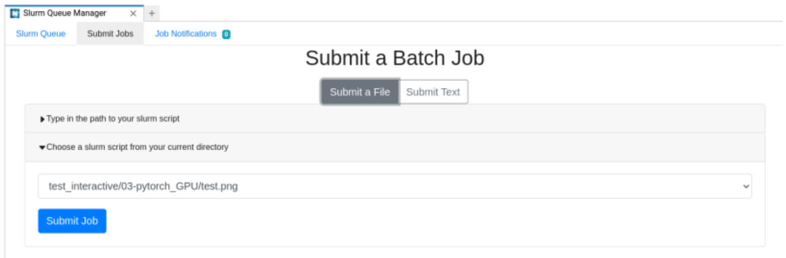




[✓] Display my jobs only

Last updated: 24/3/2023 15:19:50

	JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON
☐	8602647	g100_usr_	jupyterh	fpitari1	R	0:18	1	r505n017



Submit a Batch Job

▸ Type in the path to your slurm script

▾ Choose a slurm script from your current directory

test_interactive/03-pytorch_GPU/test.png

Current status

- Security assessment concluded, waiting for response
- Expected to be publicly avaliable before the end of the month (May 2023) at the following link:

  https://jupyter.g100.cineca.it/

- Membership in an active computing project at CINECA will be required
- Further extensions in the future (Julia, VNC, R-Studio, Octave...)

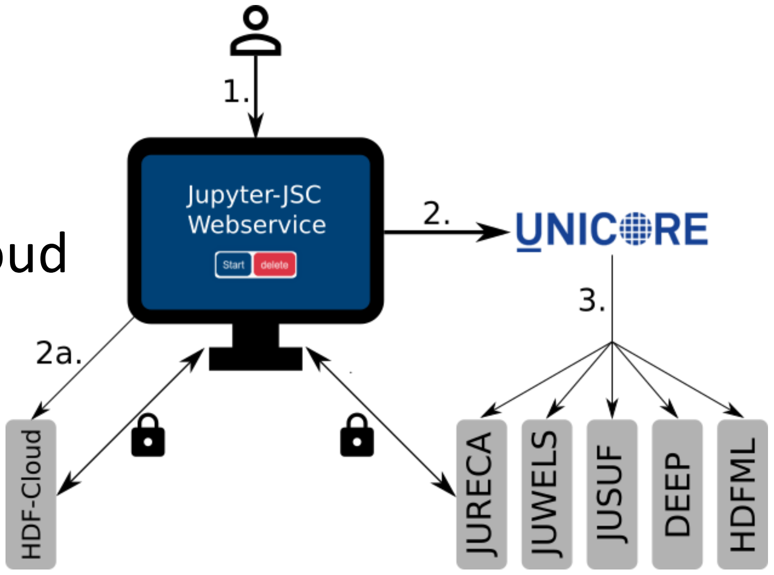
IAC IMPLEMENTATIONS AT FENIX SITES

Interactive computing at JSC

- Based on Jupyterlab
- Frontend runs on the cloud (HDF-cloud at JSC, deployed via Kubernetes)
- Main components:
  - Jupyterhub
  - Spawner:
    - via Unicore for multiple clusters
    - Via UserLab Manager for cloud-systems (starts a kubernetes pod)
  - Ad-hoc tunneling to compute nodes
- Access from the following link:

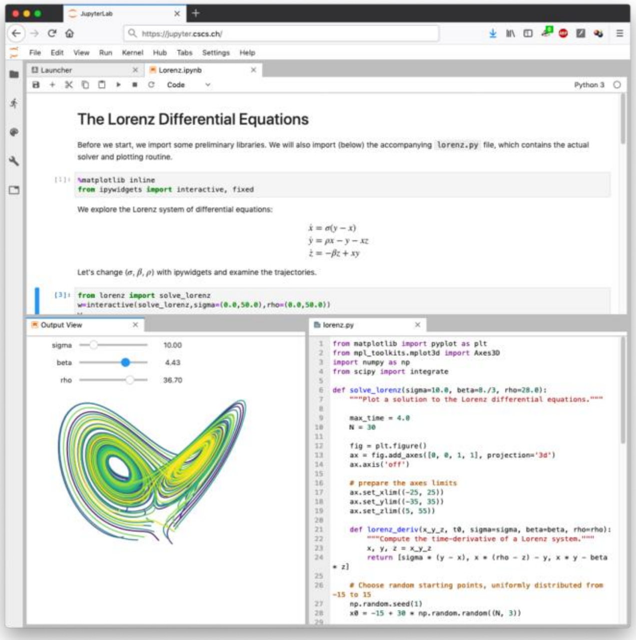
https://jupyter-jsc.fz-juelich.de/

Membership in an active computing project at JSC is required



Interactive computing at CSCS

- Similiar to Cineca implementation (Jupyterlab + Slurm spawner)
- Cluster used is Piz Daint
- Job is redirected to different queues depending on the amount of requested resources
- Cli tools to ease the customization of environments
- Access from the following link:



The Lorenz Differential Equations

Before we start, we import some preliminary libraries. We will also import (below) the accompanying `lorenz.py` file, which contains the actual solver and plotting routine.

```
[1]: %matplotlib inline
     from ipywidgets import interactive, fixed
```

We explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.

```
[3]: from lorenz import solve_lorenz
     w=interactive(solve_lorenz,sigma=(0.0,50.0),rho=(0.0,50.0))
```

```
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from scipy import integrate

def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""

    max_time = 4.0
    N = 30

    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))
```

https://jupyter.cscs.ch/

(Documentation here: https://user.cscs.ch/tools/interactive/jupyterlab/)
Membership in an active computing project at CSCS is required

Interactive computing at BSC

- Interactive sessions are driven via ssh
- Cluster employed is HSM
- Jupyterhub can be initialized via ssh on a compute node

- More information here:

  https://www.bsc.es/supportkc/docs/HSMCompute/intro/

  Membership in an active computing project at BSC is required.
  You need also to communicate a public ip to be whitelisted

Interactive computing at CEA

- Based on NiceDCV to access a Remote Desktop (Gnome environment)
- Running on Irene cluster
- Access in several steps
  - Access a TGCC machine via ssh
  - Initialize a NiceDCV session via specific command line script
  - An url is generated; the user can connect to the url from his/her browser (new authentication is required)
- More information here:

  https://www-hpc.cea.fr/tgcc-public/en/html/toc/fulldoc/Interactive_access.html

  Membership in an active computing project at CEA is required

FENIX

RESEARCH INFRASTRUCTURE

Thanks for listening!

Aknowledgements: Anne Nahm, Valentina Armuzza, Björn Hagemeier (JSC), Luciana Marques, Javier Bartolome (BSC), Alex Upton (CSCS), Thomas Leibovici (CEA), Mirko Cestari, Andrea Pieretti, Alessandro Federico (CINECA), Mario Rosati (E4)