

The GPU-based cluster @IBiSCo-ReCaS-Bari for containerized scientific applications

Gioacchino Vino¹, M. Antonacci¹, G. Donvito¹, A. Italiano¹, M. Perniola²

¹ INFN Bari

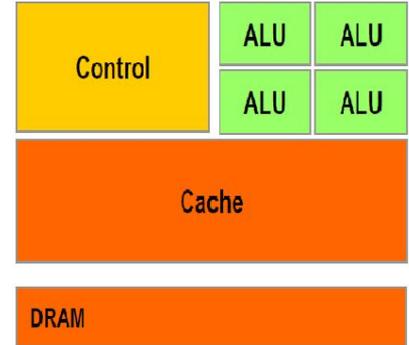
² Bari University

Workshop-IBiSCo / Napoli / 18-19 Apr 2024

Why GPU?

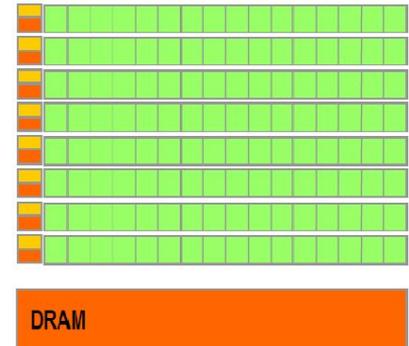
Control Processing Unit (CPU):

- Designed to handle complex tasks
- Low-level parallelism (<100 cores)



Graphical Processing Unit (GPU):

- Massively parallel hardware architecture (> 5000 cores)
- High performance of floating point arithmetic

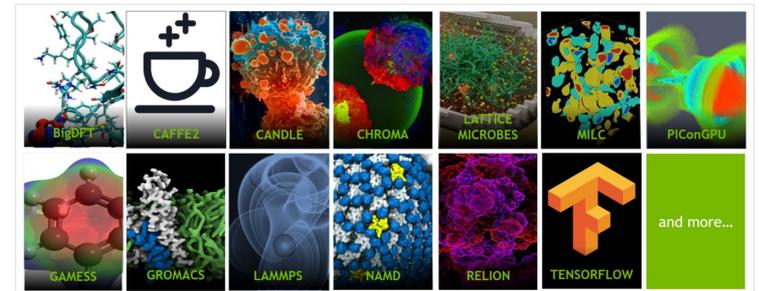
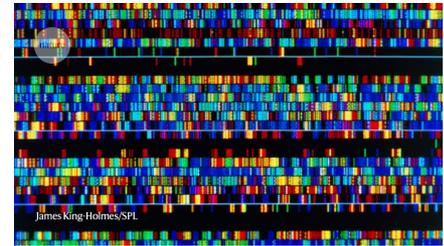
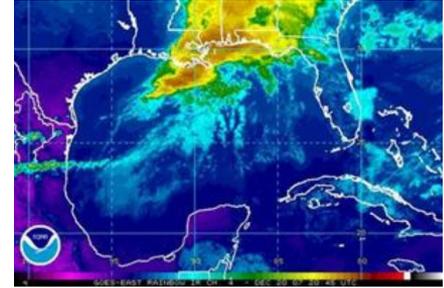


Make them suited for scientific workloads require a huge amount of floating point operations

Why GPU?

GPU main applications:

- Machine Learning (Deep Learning) algorithms
- Image processing applications
- Whole-genome sequencing
- Simulations of physical models
- Almost all problems that involve many floating point operations.



What we have: ReCaS GPU Cluster

Hardware Facility:

- Nodes: 10
- GPUs: 38 (V100 and A100 Nvidia GPU)
- Cores: 1755
- RAM: 13.7 TB
- Local Storage: 55 TB (SSD/HDD)
- Parallel File System: ReCaS storage based on IBM GPFS (12PB)
- Bandwidth between nodes: 10 Gbps



What we provide: GPU Cluster Services

- **Ready-to-use services:**

- Interactive remote GPU-based IDE services:

- Jupyter(Hub/Lab)

- “web service for interactive computing across all programming languages”

- Rstudio

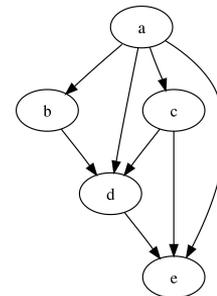
- “An integrated development environment for R”

- Job Scheduler and Orchestration:

- Support to GPU-based workflows represented as Directed Acyclic Graphs (DAG)

- **User-defined services**

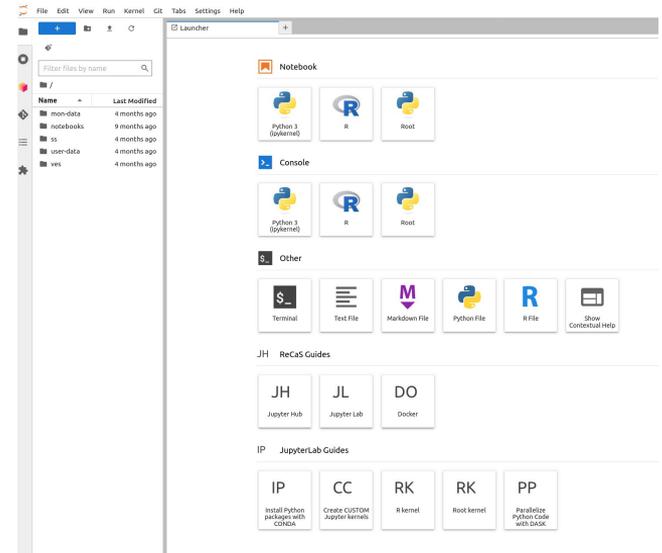
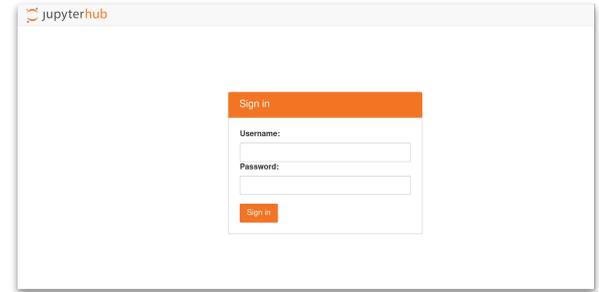
- Support and Knowledge sharing



What we provide: GPU Cluster Services

JupyterHub

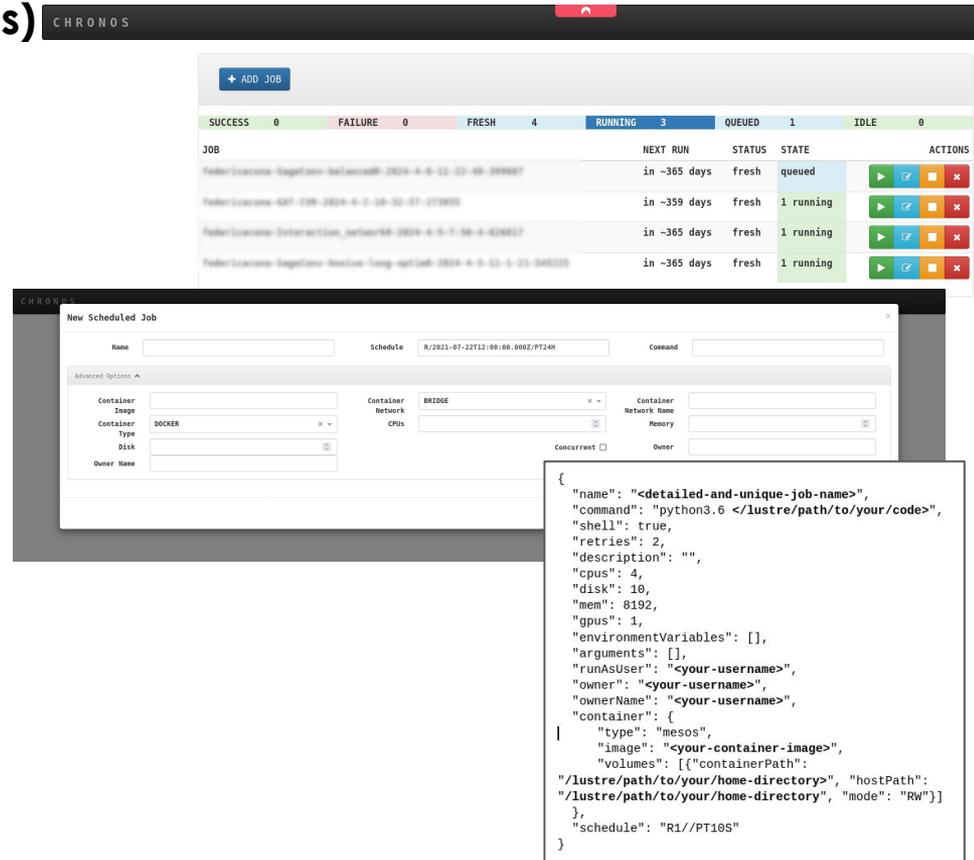
- Centralized authentication system combining IAM and LDAP
- Use computing resources (CPUs and GPUs) on demand without set them aside for a given user
- JupyterLab 4.0
- Jupyter extensions
- Python3, R and Root kernels available
- Use conda environments with Jupyter kernels
- Dask and Dask Jupyter extensions
- Execute notebook in batch with papermill
- Service and technologies guides in the Jupyter Launcher



What we provide: GPU Cluster Services

Job Scheduler and Orchestration (Chronos)

- Provides an intuitive and simple User Interface (UI) where to check job status
- New jobs can be submitted using UI or via command line using a JSON file describing the job
- Manages heterogeneous requests:
 - 2 GPU / 4 CPU / 20 GB RAM
 - 100 CPU / 8GB RAM
- A Python client has been implemented to simplify the submission of general purpose applications and Jupyter Notebooks (papermill)



The image displays the Chronos web interface. At the top, there is a header with the name 'CHRONOS' and a '+ ADD JOB' button. Below this is a summary bar showing job counts: SUCCESS 0, FAILURE 0, FRESH 4, RUNNING 3, QUEUED 1, and IDLE 0. A table below lists jobs with columns for JOB, NEXT RUN, STATUS, STATE, and ACTIONS. The table shows three jobs in various states (queued, running).

Below the table is a 'New Scheduled Job' form. It includes fields for Name, Schedule (R/2021-07-22T12:00:00.000Z/PT24H), and Command. There are also sections for 'Advanced Options' including Container Image, Type (set to DOCKER), Disk, Owner Name, Container Network (set to BRIDGE), CPU, Container Network Name, Memory, and Concurrent checkbox.

```
{
  "name": "<detailed-and-unique-job-name>",
  "command": "python3.6 </lustre/path/to/your/code>",
  "shell": true,
  "retries": 2,
  "description": "",
  "cpus": 4,
  "disk": 10,
  "mem": 8192,
  "gpus": 1,
  "environmentVariables": [],
  "arguments": [],
  "runAsUser": "<your-username>",
  "owner": "<your-username>",
  "ownerName": "<your-username>",
  "container": {
    "type": "mesos",
    "image": "<your-container-image>",
    "volumes": [{"containerPath":
"/lustre/path/to/your/home-directory", "hostPath":
"/lustre/path/to/your/home-directory", "mode": "RW"}]
  },
  "schedule": "R1/PT10S"
}
```

What's under ReCaS GPU Cluster

Apache Mesos:

- Abstracts all cluster resources in a single virtual entity
- Multi-users / High Availability / Manages many nodes



Marathon:

- Runs long running services on top of Apache Mesos
- High Availability / Load balancing



MARATHON

Chronos:

- Job scheduler for Apache Mesos
- Supports depending and periodic jobs

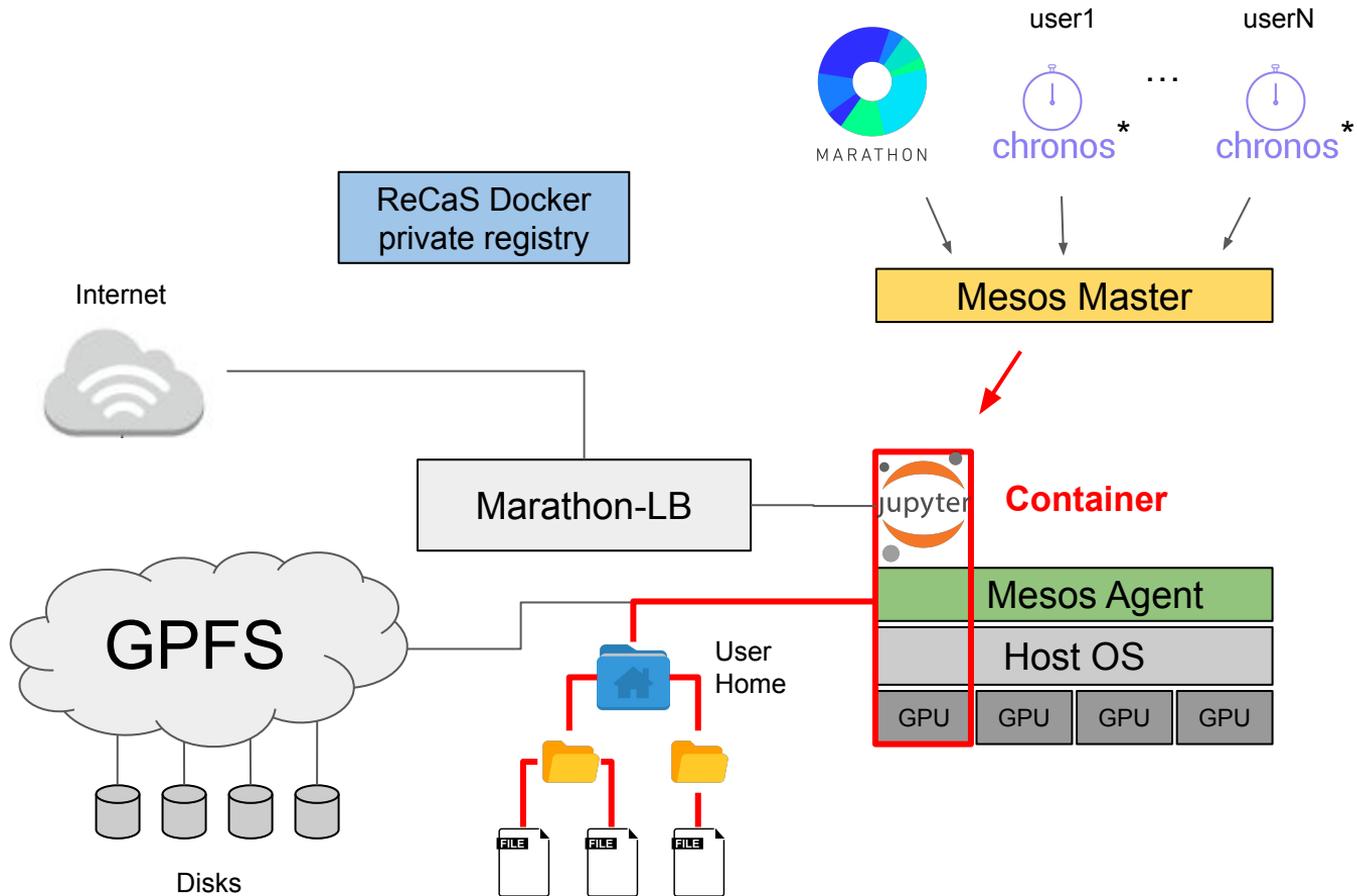


Docker

- Contains software, code, libraries and dependencies
- Isolates applications from the machine where it is executed
- Official images are available (Nvidia, TensorFlow, ...)



How ReCaS GPU Cluster works



What we learned

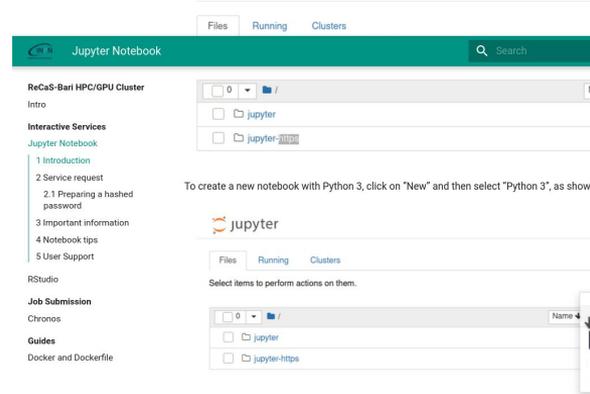
- +50 users
- Applications:
 - Artificial Intelligence
 - Whole-genome sequencing
 - Image processing
- Average speed-up (vs CPUs) x5
- Most of the users requested support in the building of their custom docker container images
- Most of users are not well trained to use parallel programming paradigm

What we do: Share knowledge

- User programming skills are not enough to take advantage from the provided computation resources
- There is a **gap** between the goal and the user programming skill
- **Guides** and **video tutorials** to support users at the beginning
- Provide support in docker container image building
- Provide support in writing efficient code

1 Introduction

The Jupyter Notebook is an open-source web application that allows you to create and share equations, visualisations and narrative text. Uses include: data cleaning and transformation, modelling, data visualisation, machine learning, and much more. Through Jupyter Notebook, directories stored in the ReCaS-Bari GPFS file system and browse graphically, as shown in the



The screenshot shows the Jupyter Notebook interface. On the left, there is a sidebar with a table of contents for the 'Introduction' section, including '1 Introduction', '2 Service request', '2.1 Preparing a hashed password', '3 Important information', '4 Notebook tips', and '5 User Support'. The main area displays the 'Files' tab of the file browser, showing a directory structure with 'jupyter' and 'jupyter-https' folders. Below the file browser, there is a section titled 'To create a new notebook with Python 3, click on "New" and then select "Python 3", as shown' with a 'jupyter' logo and a 'Files Running Clusters' header. The bottom part of the screenshot shows the Jupyter Notebook editor interface with the title 'jupyter Untitled (unsaved changes)' and a menu bar with options like File, Edit, View, Insert, Cell, Kernel, Widgets, and Help.

Finally, the Integrated Development Environment (IDE) is opened, as shown in the figure.

What we plan: Future Developments

- **Kubernetes** will replace Apache Mesos since it overcomes some known limitations
- Chronos will be replaced with a more complex workflow scheduler, like **Apache Airflow**
- Adding distributed computing tools to the service portfolio like **Apache Spark**
- Integrate the cluster with **INFN-DataCloud PaaS**
- Investigate the use of **Infiniband** to speed-up the internode connections



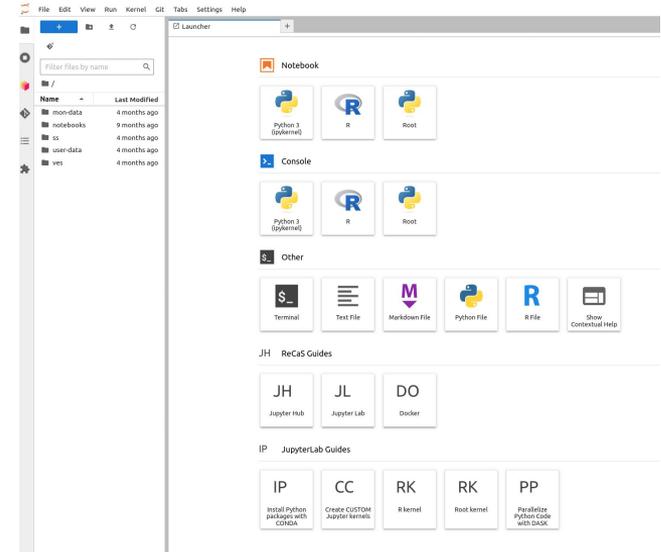
**THANKS
FOR YOUR
ATTENTION**

BACKUP

What we provide: GPU Cluster Services

JupyterLab

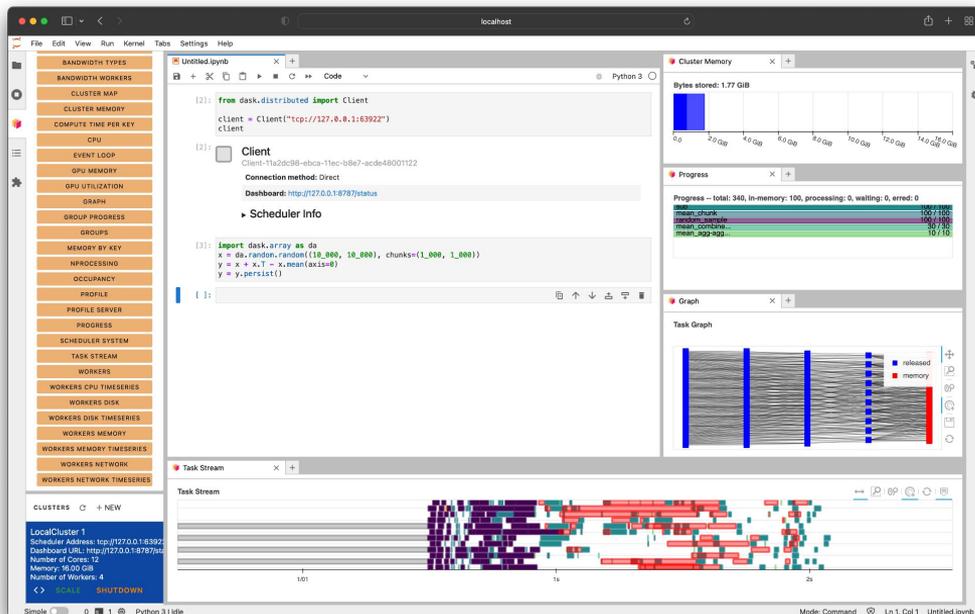
- Provided when user need to use a whole GPU
- Users have access to their home directory in the ReCaS distributed storage (GPFS)
- Users can immediately create a new Python3 script
- The Jupyter IDE (Integrated Development Environment) will be available and users can already write code and execute it
- Python modules can be installed directly within the code or using built-in terminal (pip or conda)



What we provide: GPU Cluster Services

JupyterLab: Dask and Dask Jupyter extension

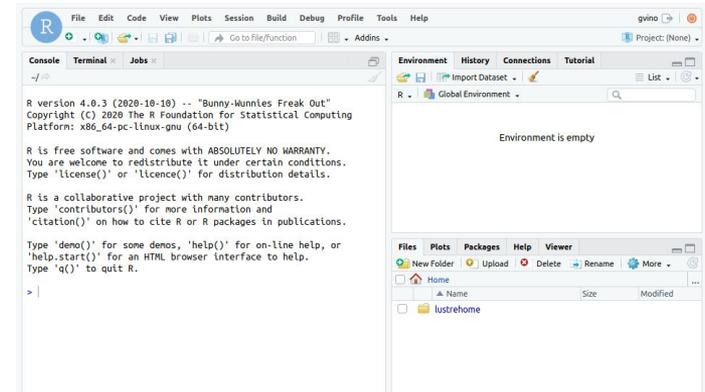
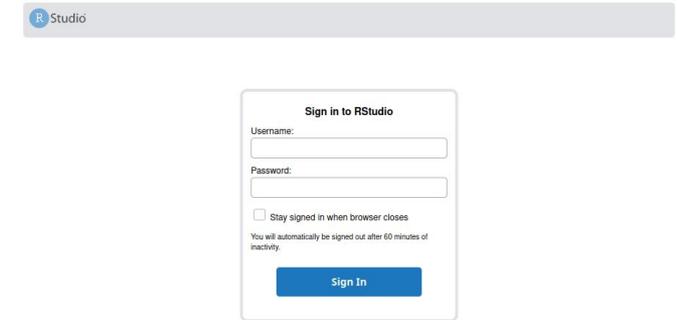
- Scale the Python libraries (like NumPy, pandas, and scikit-learn) on multiple cores and GPUs, using the same code
- Parallelize any Python code with Dask Futures
- Real-time resource usage Monitoring



What we provide: GPU Cluster Services

RStudio remote IDE

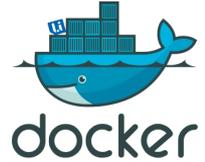
- After authentication, users have access to their home directory in the ReCaS distributed storage (GPFS)
- The Rstudio IDE (Integrated Development Environment) will be available and users can already write code and execute it
- R modules can be installed directly within the code



What's under ReCaS GPU Cluster

Docker container:

- Contains software, code, libraries and dependencies
- Isolates applications from the machine where it is executed
- Official images are available (Nvidia, TensorFlow, ...)



ReCaS GPU Cluster policies on Docker containers:

- Mandatory for security purpose
- Jupyter and RStudio containers have been developed in-house
- Not all users' containers can be developed in-house
- Users can build their own Docker image for their specific use-case using a dedicated machine with GPU in ReCaS-Bari
- ReCaS GPU Cluster Docker Registry is available