The new GPU-based cluster @ReCaS-Bari

Gioacchino Vino, M. Antonacci, G. Donvito, D. Elia, A. Italiano

INFN Bari

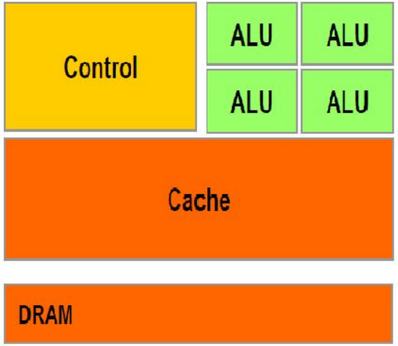
ACAT / Bari / 24-28 Oct 2022

Progetti di rafforzamento del capitale umano destinato alle Infrastrutture di ricerca beneficiarie dell'Avviso D.D. del 28 febbraio 2018, n. 424

Why GPU?

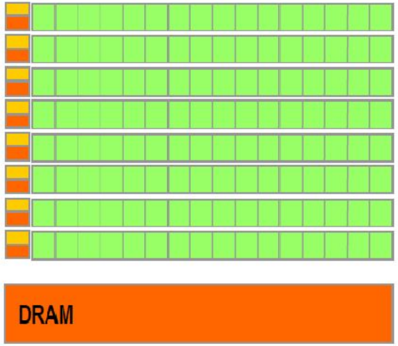
Control Processing Unit (CPU):

- 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

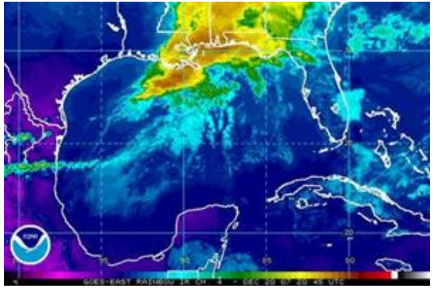


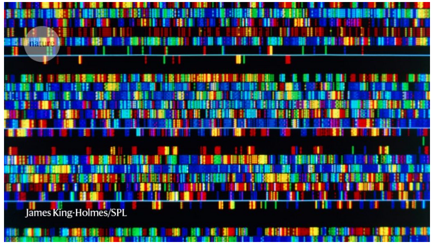
Make them suited for scientific workloads require a huge amount of floating point operations

Why GPU?

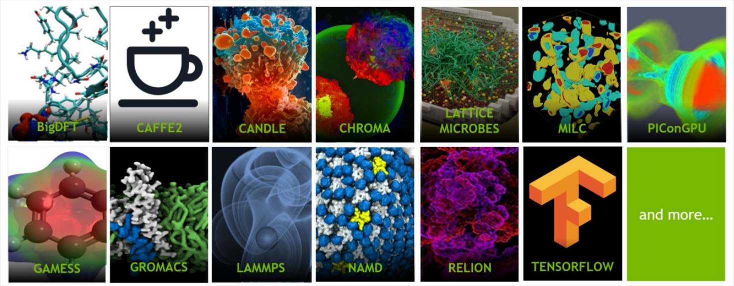
GPU main applications:

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





James King-Holmes/SPL



What users want

- Access to overall ReCaS-Bari storage
- Use the whole GPU cluster computing power
- Use GPUs efficiently without acquire new knowledge
- Build GPU custom applications in few minutes
- Use GPUs without worry about managing to underlying hardware and software
- Use the same code for 1 CPU or for all GPUs in the cluster

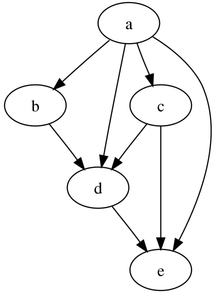
What we have: ReCaS GPU Cluster

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 (3800TB)
- Bandwidth between nodes: 10 Gbps

What we provide: GPU Cluster Services

- **Ready-to-use services:**
  - Interactive remote GPU-based IDE services:
    - JupyterLab + Dask
      - “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



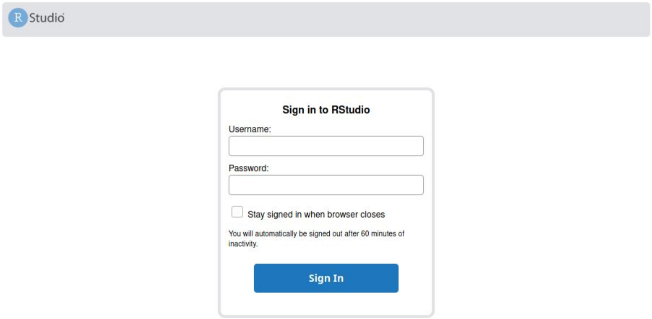
What we provide: GPU Cluster Services

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



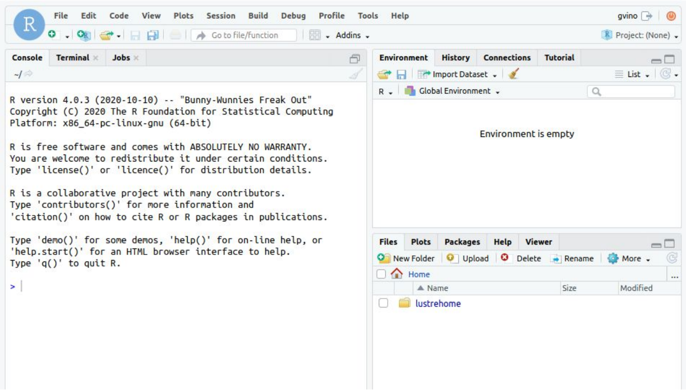
Sign in to RStudio

Username:

Password:

□ Stay signed in when browser closes

You will automatically be signed out after 60 minutes of inactivity.

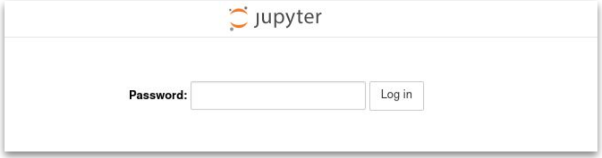

File Edit Code View Plots Session Build Debug Profile Tools Help
gvino
Go to file/function
Addins
Project: (None)
Console Terminal Jobs
~/
Environment History Connections Tutorial
Import Dataset
List
R
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder Upload Delete Rename More
Home
Name Size Modified
lustrehome

```
R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

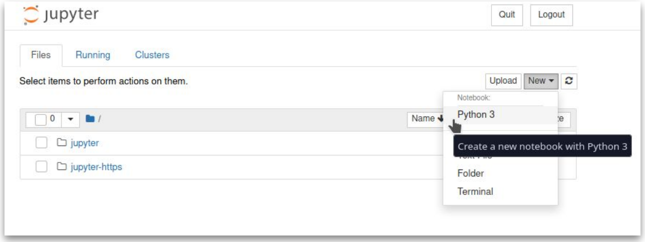
What we provide: GPU Cluster Services

JupyterLab + Dask remote IDE

- After authentication, users have access to their home directory in the ReCaS distributed storage (GPFS)



- Users can immediately create a new Python3 script

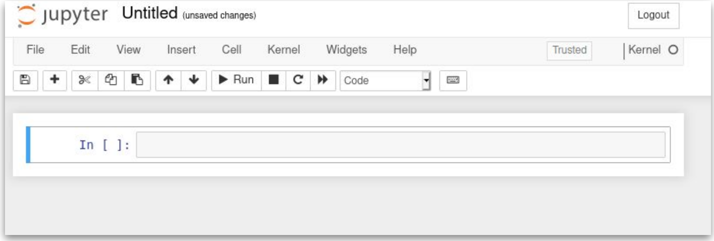


Select items to perform actions on them.

Create a new notebook with Python 3

- The Jupyter IDE (Integrated Development Environment) will be available and users can already write code and execute it

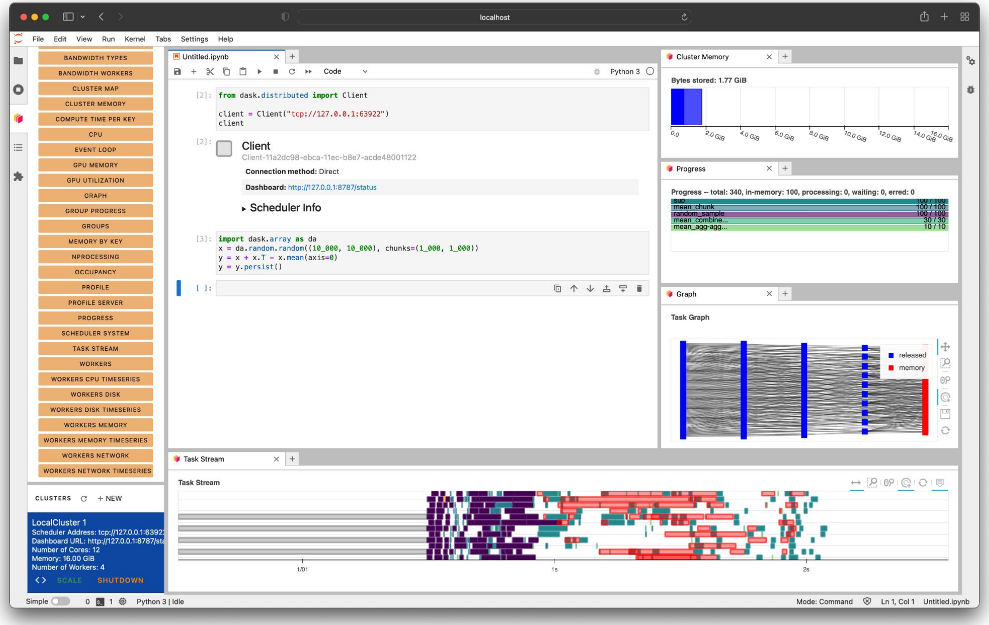
- Python modules can be installed directly within the code or using built-in terminal



What we provide: GPU Cluster Services

JupyterLab + Dask remote IDE

- 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



```
[2]: from dask.distributed import Client

     client = Client("tcp://127.0.0.1:63922")
     client
```

[2]: Client

Client-11a2dc98-ebca-11ec-b8e7-acde48001122

Connection method: Direct

Dashboard: http://127.0.0.1:8787/status

▸ Scheduler Info

```
[3]: import dask.array as da
     x = da.random.random((10_000, 10_000), chunks=(1_000, 1_000))
     y = x + x.T - x.mean(axis=0)
     y = y.persist()
```

[ ]:

LocalCluster 1
Scheduler Address: tcp://127.0.0.1:6392
Dashboard URL: http://127.0.0.1:8787/sta
Number of Cores: 12
Memory: 16.00 GiB
Number of Workers: 4

Bytes stored: 1.77 GiB

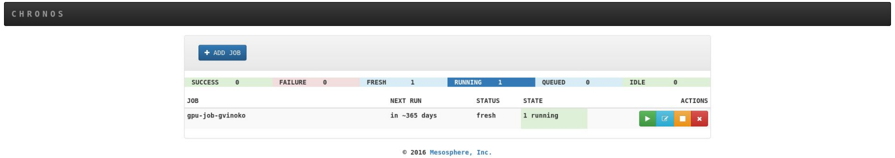
Progress -- total: 340, in-memory: 100, processing: 0, waiting: 0, erred: 0

```
sub                 100 / 100
mean_chunk          100 / 100
random_sample       100 / 100
mean_combine...       30 / 30
mean_agg-agg...       10 / 10
```

What we provide: GPU Cluster Services

Job Scheduler and Orchestration (Chronos)

- Provides an intuitive and simple User Interface (UI) where to check job status

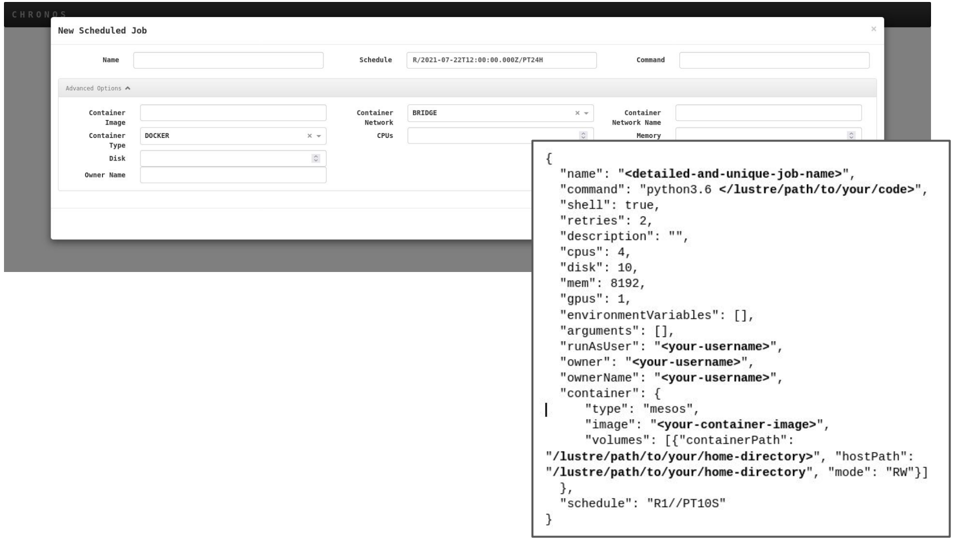


SUCCESS 0	FAILURE 0	FRESH 1	RUNNING 1	QUEUED 0	IDLE 0

JOB	NEXT RUN	STATUS	STATE	ACTIONS
gpu-job-gvinoko	in ~365 days	fresh	1 running	

© 2016 Mesosphere, Inc.

- New jobs can be submitted using UI or via command line using a JSON file describing the job



Name:
Schedule: R/2021-07-22T12:00:00.000Z/PT24H
Command:
Container Image:
Container Network: BRIDGE
Container Network Name:
Container Type: DOCKER
CPUs:
Memory:
Disk:
Owner Name:

```
{
  "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"
}
```

- Manages heterogeneous requests:
  - 2 GPU / 4 CPU / 20 GB RAM
  - 100 CPU / 8GB RAM

What’s under ReCaS GPU Cluster

Apache Mesos:

- Abstracts all cluster resources in a single virtual entity
- Multi-users
- High Availability
- Manages a lot number of nodes

Marathon:

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

Chronos:

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

What’s under ReCaS GPU Cluster

Docker container:

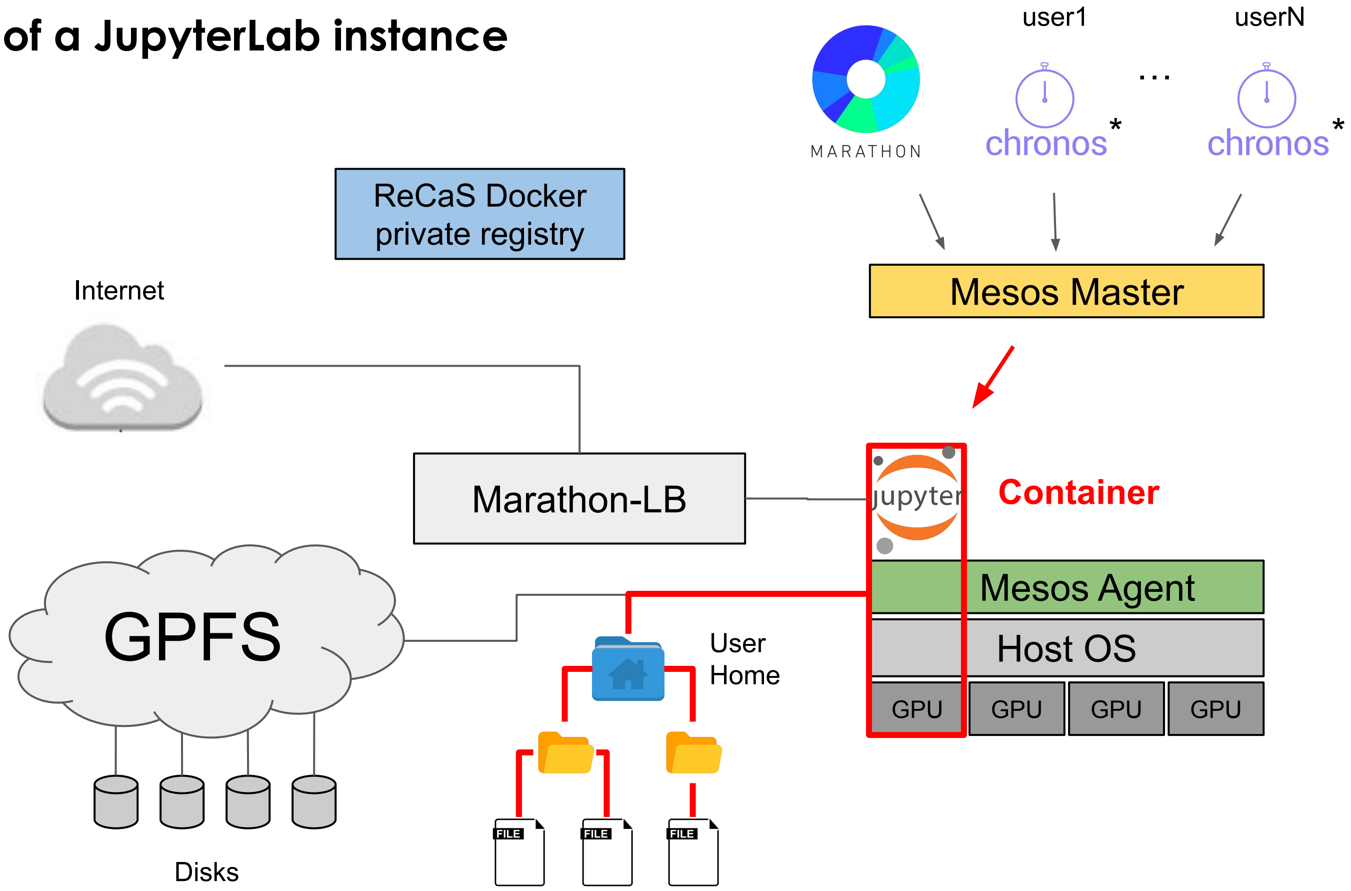
- Contains software, code, libraries and dependencies
- Isolates applications from the machine where it is executed
- Official images are available (Nvidia, TensorFlow, ...)

ReCaS GPU Cluster policies on Docker containers:

- Mandatory for security purpose
- Jupyter Notebook and RStudio containers have been developed in-house because the majority of the supported use cases needs them
- 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

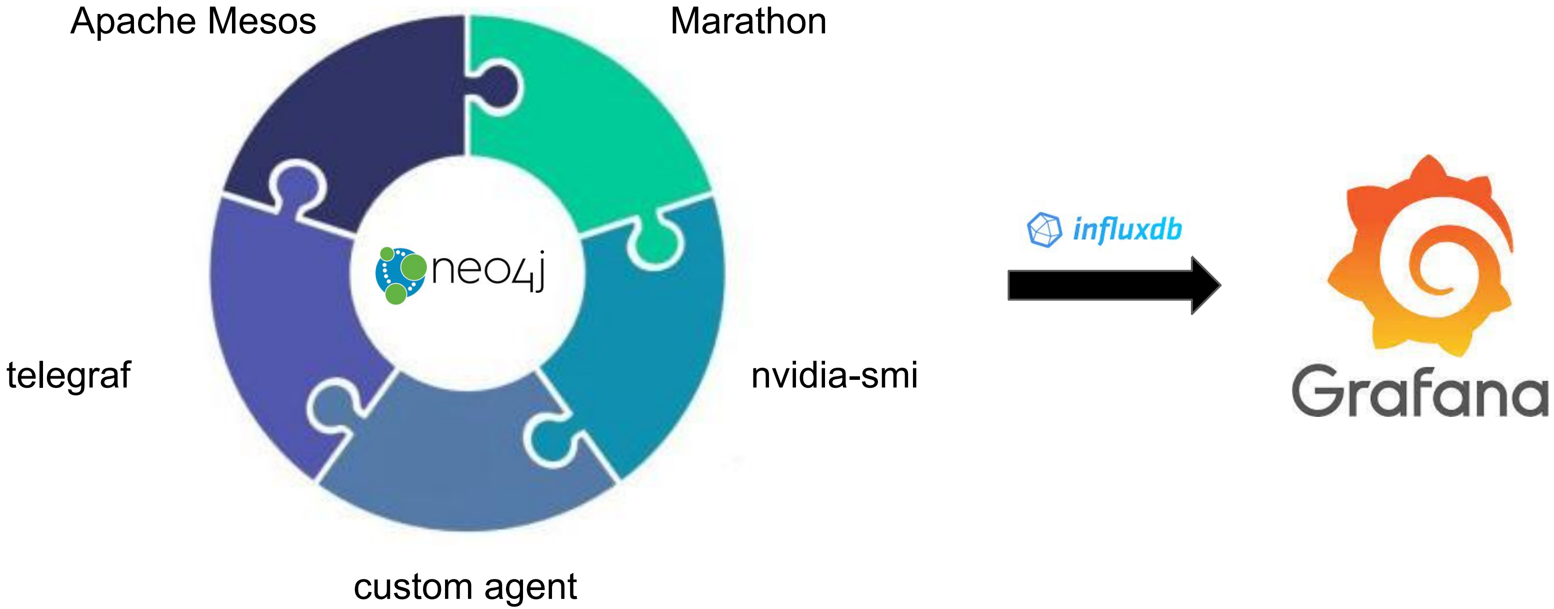
How ReCaS GPU Cluster works

Deployment of a JupyterLab instance



What’s under ReCaS GPU Cluster

Monitoring



Apache Mesos

Marathon

telegraf

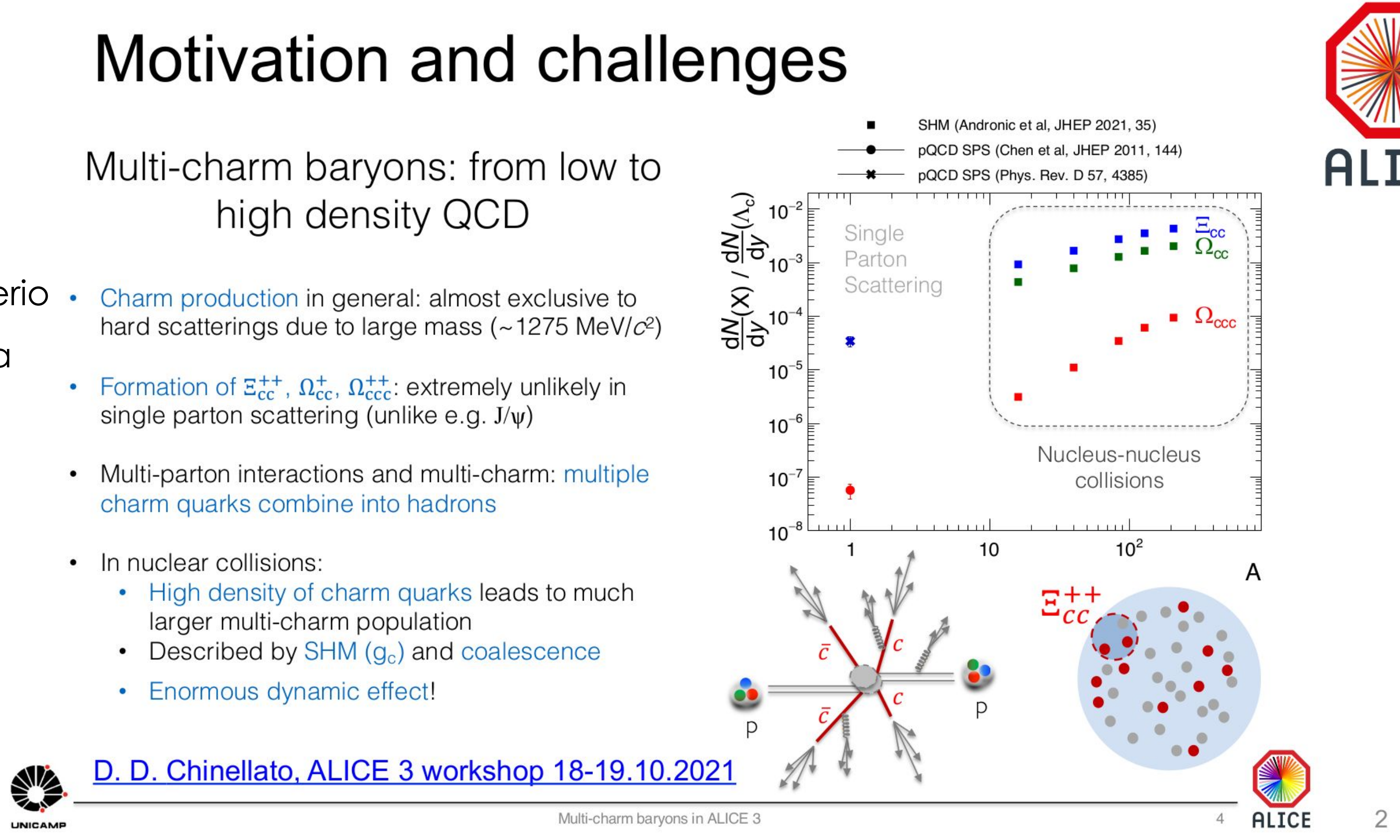
nvidia-smi

custom agent

Use case: Multi-charm reconstruction with ML in ALICE 3

Working Team:

- Domenico Elia
- Annalisa Mastroserio
- Domenico Colella
- Gioacchino Vino
- David Chinellato (CERN)



Motivation and challenges

Multi-charm baryons: from low to high density QCD

- Charm production in general: almost exclusive to hard scatterings due to large mass (~1275 MeV/c^2)
- Formation of Ξ_{cc}^{++}, Ω_{cc}^{+}, Ω_{ccc}^{++}: extremely unlikely in single parton scattering (unlike e.g. J/ψ)
- Multi-parton interactions and multi-charm: multiple charm quarks combine into hadrons
- In nuclear collisions:
  - High density of charm quarks leads to much larger multi-charm population
  - Described by SHM (g_c) and coalescence
  - Enormous dynamic effect!

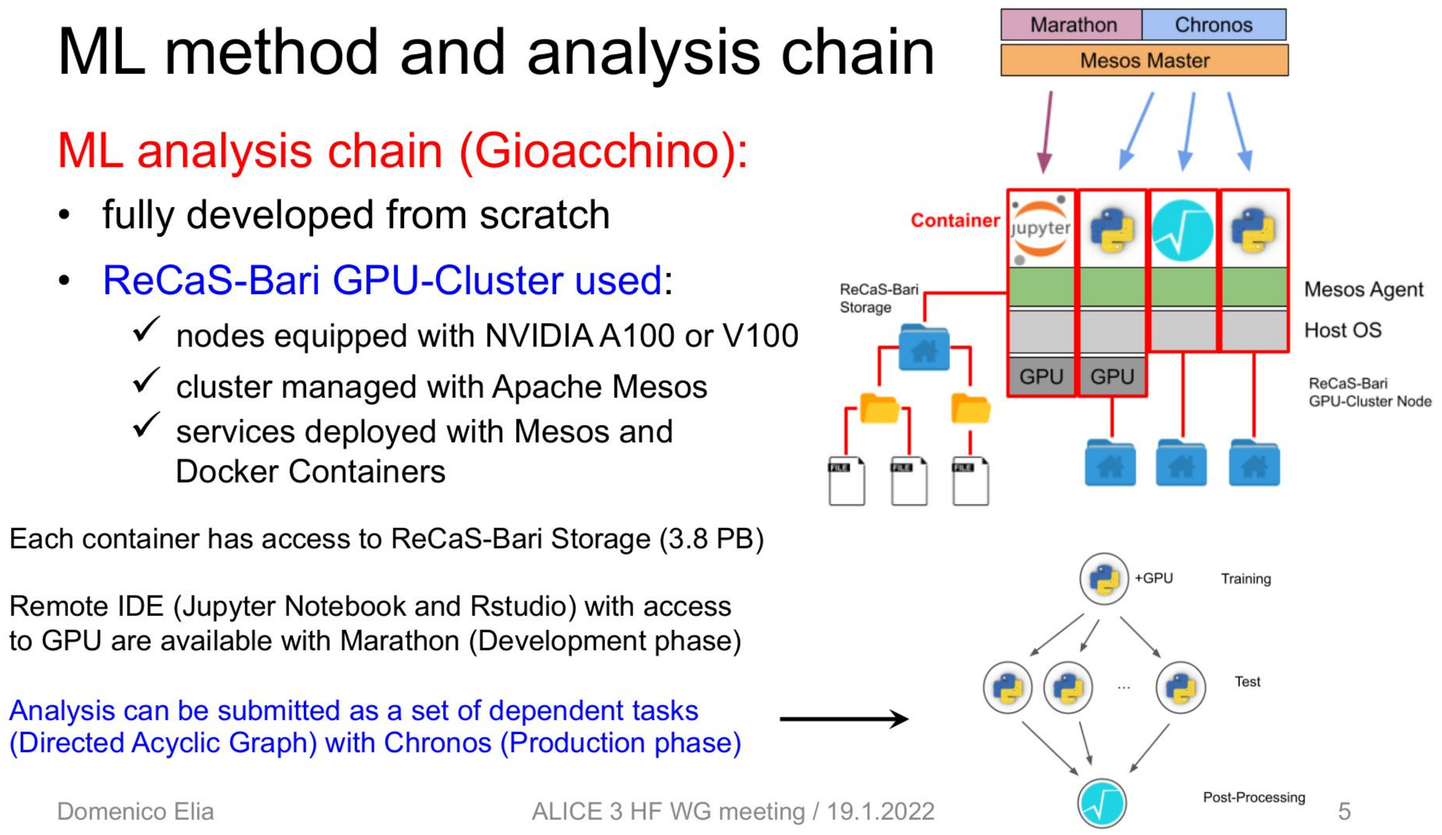
D. D. Chinellato, ALICE 3 workshop 18-19.10.2021

Multi-charm baryons in ALICE 3

4

2

Use case: Multi-charm reconstruction with ML in ALICE 3



ML method and analysis chain

ML analysis chain (Gioacchino):

- fully developed from scratch
- ReCaS-Bari GPU-Cluster used:
  - ✓ nodes equipped with NVIDIA A100 or V100
  - ✓ cluster managed with Apache Mesos
  - ✓ services deployed with Mesos and Docker Containers

Each container has access to ReCaS-Bari Storage (3.8 PB)

Remote IDE (Jupyter Notebook and Rstudio) with access to GPU are available with Marathon (Development phase)

Analysis can be submitted as a set of dependent tasks (Directed Acyclic Graph) with Chronos (Production phase)

Domenico Elia ALICE 3 HF WG meeting / 19.1.2022 5

Use case: Multi-charm reconstruction with ML in ALICE 3

Preliminary results:

- better than standard, especially at low p_T
- up to a factor of 4-5x improvement for $p_T < 2$ GeV/c

Impact on future measurements:

- ML-based selection has potential to allow measurement of multi-charm down to 0 p_T
- included in the **ALICE 3 Letter of Intent** recently published

Work ongoing:

- still room to improve ML-based selection performance, in particular exploiting p_T-dedicated training

ALICE 3 LoI:
https://cds.cern.ch/record/2803563

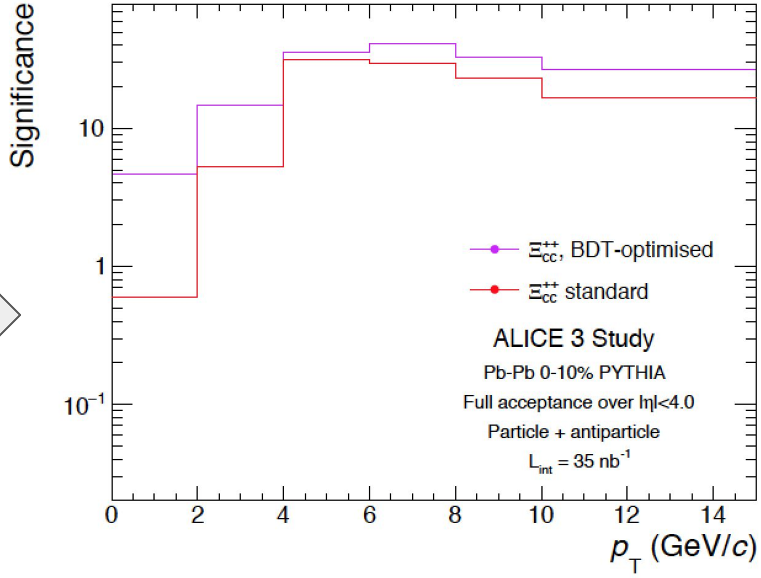


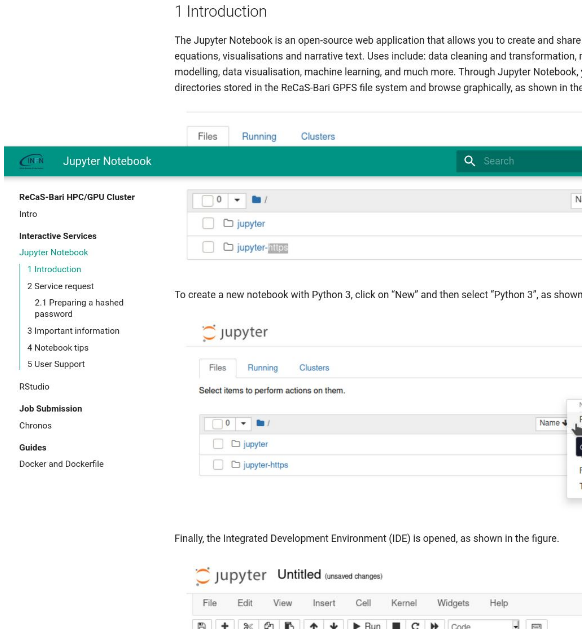
Figure 33: Ξ_{cc}^{++} significance in 0-10% central Pb-Pb collisions at $\sqrt{s_{NN}} = 5.52$ TeV as a function of p_T with a 2.0 T magnetic field using standard selections and using machine learning.

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 plan: Improve user learning curve

- It is no enough to provide performance tools: we would like users could use them **efficiently**
- There is a **gap** between the goal and the knowledge of users
- **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, r modelling, data visualisation, machine learning, and much more. Through Jupyter Notebook, y directories stored in the ReCaS-Bari GPFS file system and browse graphically, as shown in the

To create a new notebook with Python 3, click on “New” and then select “Python 3”, as shown

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

What we plan: Improve Jupyter experience

JupyterHub + Dask remote IDE

- Centralized authentication system combining IAM and LDAP

- Use computing resources (CPUs and GPUs) on demand without set them aside for a given user

What we plan: Future Developments

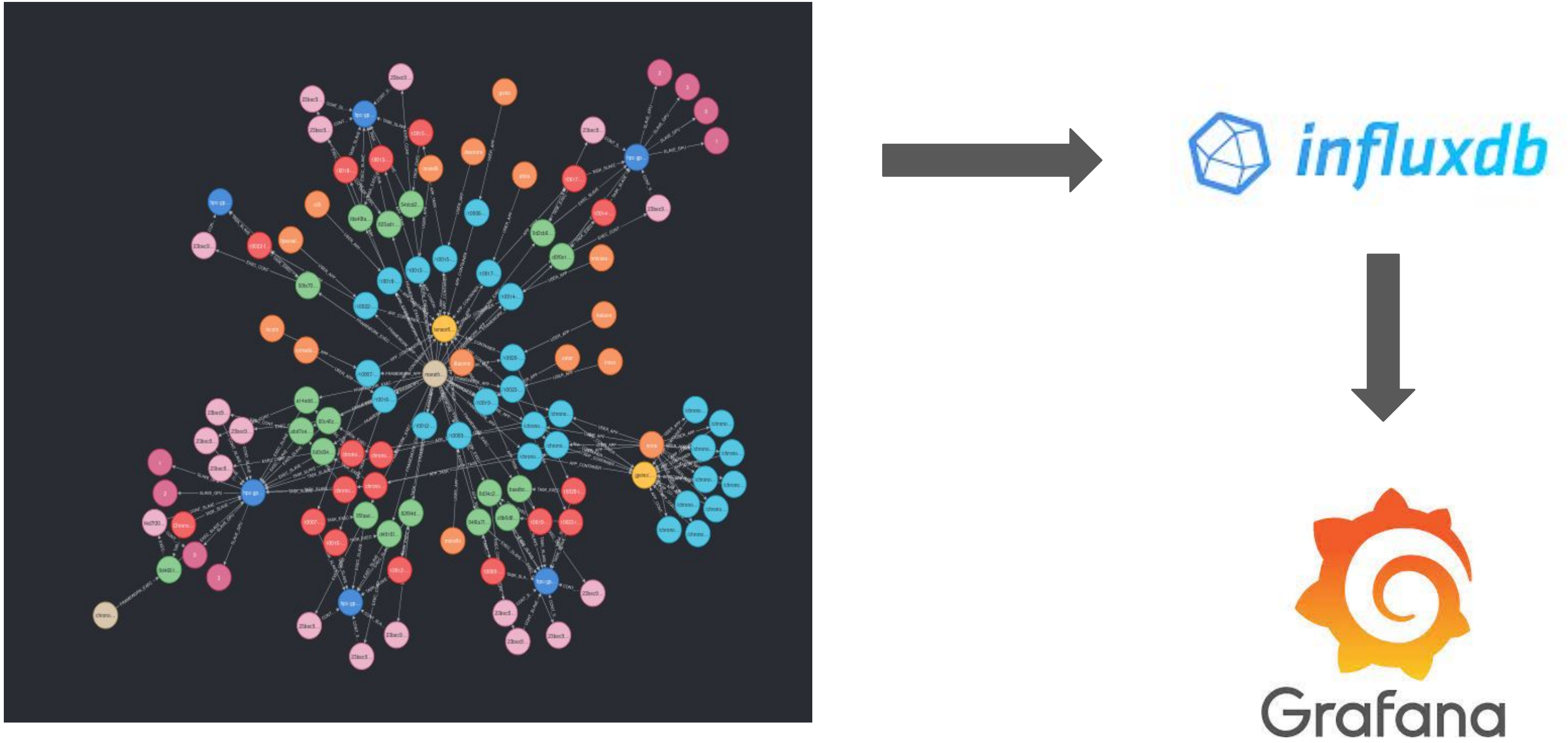
- **Kubernetes** will replace Apache Mesos since it overcomes some known limitations
- Chronos will be replaced with a more complex workflow scheduler, like **Apache Airflow**
- Adding distributed computing tools to the service portfolio like **Apache Spark**
- Integrate the cluster with **INFN-DataCloud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

THANKS

FOR YOUR

ATTENTION

BACKUP


influxdb
Grafana