

Usage of INAF resources

Pleiadi

Fabio Vitello - INAF OACT

USCVIII - General Assembly

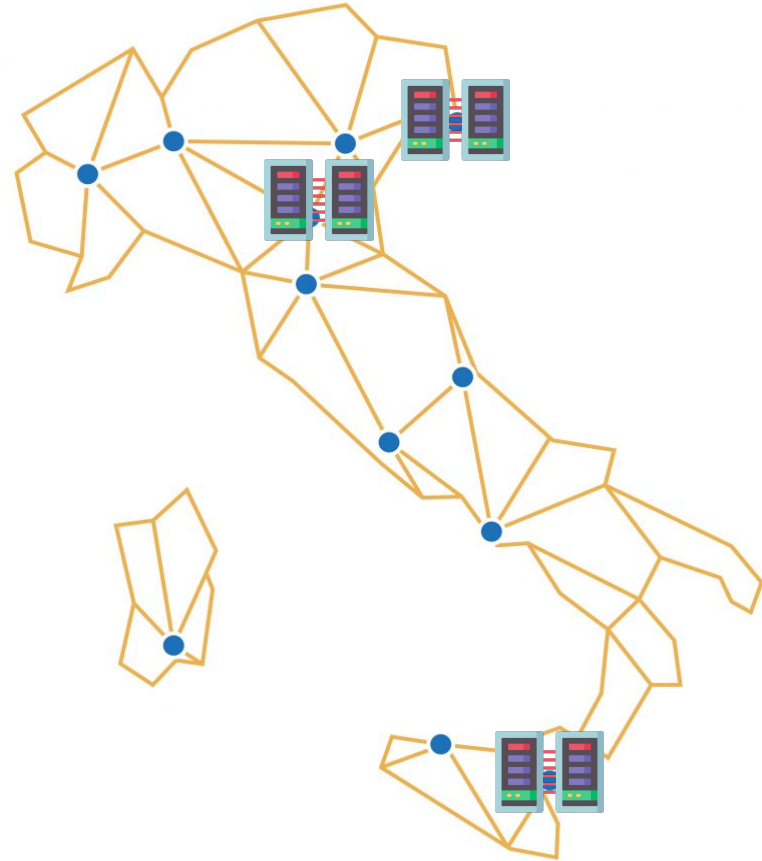
Architecture

PLEIADI is a project by **USC VIII-Computing** of INAF – National Institute for Astrophysics, **offering** high-performance computing (**HPC**) and high-throughput computing (**HTC**) resources.

Individual researchers and teams belonging to research projects, European projects, PRIN, INAF mainstream projects, scientific missions, etc. that require computing can apply requesting the resources.

The Pleiadi infrastructures is distributed on the following sites:

Bologna (IRA), Catania and Trieste (+ soon Palermo)



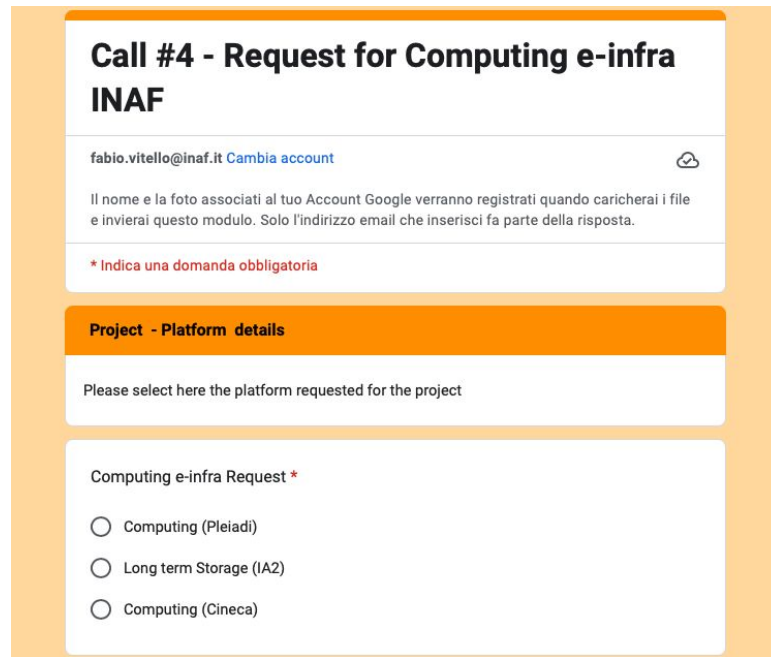
Call for requesting computing e-infra

Resources allocation

System in production since early 2022, resources allocated to **special projects**, available through **calls (semi-annual)** or **on-demand** assignment.

Resources offered:

- PLEIADI computing system
- Leonardo BOOSTER computing system
- Long-term preservation system of scientific products at IA2



The image shows a screenshot of a Google Form titled "Call #4 - Request for Computing e-infra INAF". The form is set against an orange background. At the top, the title is in bold black text. Below the title, the sender's email "fabio.vitello@inaf.it" is displayed with a "Cambia account" link and a cloud icon. A note in Italian explains that the user's name and photo will be registered upon file upload. A red asterisk indicates a mandatory field. The form has a section header "Project - Platform details" in orange. Below this, a text prompt asks the user to select a platform. A dropdown menu is open, showing three options: "Computing (Pleiadi)", "Long term Storage (IA2)", and "Computing (Cineca)".

Call #4 - Request for Computing e-infra INAF

fabio.vitello@inaf.it [Cambia account](#)

Il nome e la foto associati al tuo Account Google verranno registrati quando caricherai i file e invierai questo modulo. Solo l'indirizzo email che inserisci fa parte della risposta.

* Indica una domanda obbligatoria

Project - Platform details

Please select here the platform requested for the project

Computing e-infra Request *

Computing (Pleiadi)

Long term Storage (IA2)

Computing (Cineca)

Available resources - Computing (Cineca)

As part of the agreement with Cineca, INAF offers **1.5 Million Standard Hours** which will be usable from 01/08/2024.

Proposals for computing resources requiring a **minimum of 12,500 Standard Hours** up to a **maximum of 125,000 Standard Hours** will be evaluated. The assigned computing resources will be available for up to a **maximum of 6 months** from the aforementioned date of first use

Projects can also require **up to 4TB of work storage**. Beyond these limits, additional disk resources may be requested (subject to verification of actual availability by Cineca).

Available resources - Long Term Storage (IA2)

INAF provides **long-term data storage** spaces, even completely free from the need for computing resources. These long-term storage spaces will be made available starting from 01/08/2024.

The typical dimensions of the spaces required in each individual **proposal** are **up to 20TB**.

Their use may continue until the date indicated and justified by the applicant.

Since data preservation will take place on **Tape Library devices**, it is advisable to carefully read the description of the data sharing and preservation services offered by IA2 as well as the description of the Long-Term Preservation service for the preliminary measures necessary for preparing the data for the preservation and subsequent recovery of the data.

Available resources - Computing (Pleiadi)

The call makes a total of **15 Million CORE hours** available, which will be usable starting from 01/08/2024.

Proposals for the use of computing e-infra that require a **minimum of 100,000 CORE hours** up to a **maximum of 500,000 CORE** hours will be evaluated.

The assigned computing resources will be available for up to a **maximum of 6 months** from the aforementioned date of first use.

Proposal submission

In the application, the proposing research groups will be asked to specify in detail:

- the **scientific background**,
- the **technical characteristics** of the code,
- detailing **libraries**, computing environment, compilers, **paradigm and degree of parallelism**.
- any constraints on how resources are used (for example dedicated nodes, minimum number of nodes required per run, execution time of a single run, total memory for a single job,....etc).

Allocated time that is not used in the requested period cannot be recovered later.

Proposal submission

Proposers will also be asked to **specify the storage space** necessary for the execution of the code. This storage, not subject to backup, will be available for up to **6 months from the end of the project**, and is to be understood as functional for data production (i.e. it does not correspond to preservation storage).

If **long-term** saving of the data produced is necessary, it will be necessary to submit a further and separate request, again via the proposal submission form, for the saving space on **Tape Library IA2**.

Proposal submission

The application for **long-term** data preservation space must contain all the data necessary for the evaluation of the request:

- size of **preservation storage space** requested;
- **type, format** and **size of each scientific product**;
- description of the **structure of the collection**;
- description of the **expected frequency of access** as well as the data access policy;
- any future plans for publishing the data;
- all information deemed relevant for correct data preservation following the **FAIR principles**.

Additional info and Ex-post activities

Starting from 01/07/2024 (the next day after the closing of the call), it will also be possible to request computing resources on a **“first come, first serve basis”**, up to a **maximum of 100,000 CORE hours** per project, and **until the resources available** for this methodology, equal to **5,000,000 CORE hours**, are exhausted.

As regards **Cineca and Leonardo BOOSTER**, it will be possible to **request hours for “tests”** up to a **maximum of 10,000 Standard Hours** per project. The computing resources assigned in this mode will be available for up to a **maximum of 2 months** from the date of first use.

At the end of the project, the proponent will be asked for a **short report on the results** obtained and the **critical issues encountered** using the assigned resources. **Failure to send this report precludes participation in the subsequent call.**

Allocation on PLEIADI, Call 4

Posta in arrivo x

pleiadi/call4 x



Claudio Gheller <claudio.gheller@inaf.it>

a gaetano.scandariato, Andrea, me ▾

ven 26 lug, 14:53



Traduci in italiano



we are glad to inform you that the "Comitato di Allocazione di Tempo di calcolo e Spazio di archiviazione INAF (CAT&S)" has approved your request for computing resources submitted to the "Pleiadi Call 4", with the following feedback:

"The scientific goal is clear and compelling. The technical description is sound and the computing time request is properly motivated."

100000 core hours, available from AUGUST 1 2024 to JANUARY 31 2025, have been awarded to the project on the PLEIADI system in Catania. We invite you to contact Fabio Vitello (reading in CC) for the start-up of the project.

For information on the system, please refer to the documentation at <https://pleiadi.readthedocs.io>

For technical support use **ONLY** the ticketing system, sending an email to pleiadi-help@ced.inaf.it

We remind you also that at the end of the project, the proponent will be asked for a short report on the results obtained and the critical issues encountered using the assigned resources. Failure to send this report precludes participation in the subsequent call.

Regards.

Claudio Gheller (on behalf of CAT&S)

Pleiadi Resources

Bologna

- **48 compute nodes without GPUs**

Architecture	Cluster Linux x86_64
Nodes interconnection	Omni-Path HFI Silicon 100 Series, 100 Gbits interconnect
Service Network	Ethernet 1 Gbits
CPU Model	Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz
Number of Nodes	48
Operating System	Debian 11
Workload manager	SLURM 20.11.7
Storage volume	200 TB, Lustre parallel filesystem (quota is 10 TB per user)

Catania

- **72 compute nodes without GPUs** (12 with a RAM memory of 256 GB and 60 with a RAM memory of 128 GB)
- **6 compute nodes with 1 GPU each** (4 of Tesla K40m type, of 12 GB of memory each, and 2 of Tesla V100 PCIe type, of 16 GB of memory each), with a RAM memory of 128 GB

Architecture	Cluster Linux x86_64
Nodes interconnection	Omni-Path HFI Silicon 100 Series, 100 Gbits interconnect
Service Network	Ethernet 1 Gbits
CPU Model	Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz
Number of Nodes	78
Operating System	CentOS Linux release 7.9.2009
Workload manager	SLURM 21.08.5
Storage volume	174TB, BeeGFS parallel filesystem

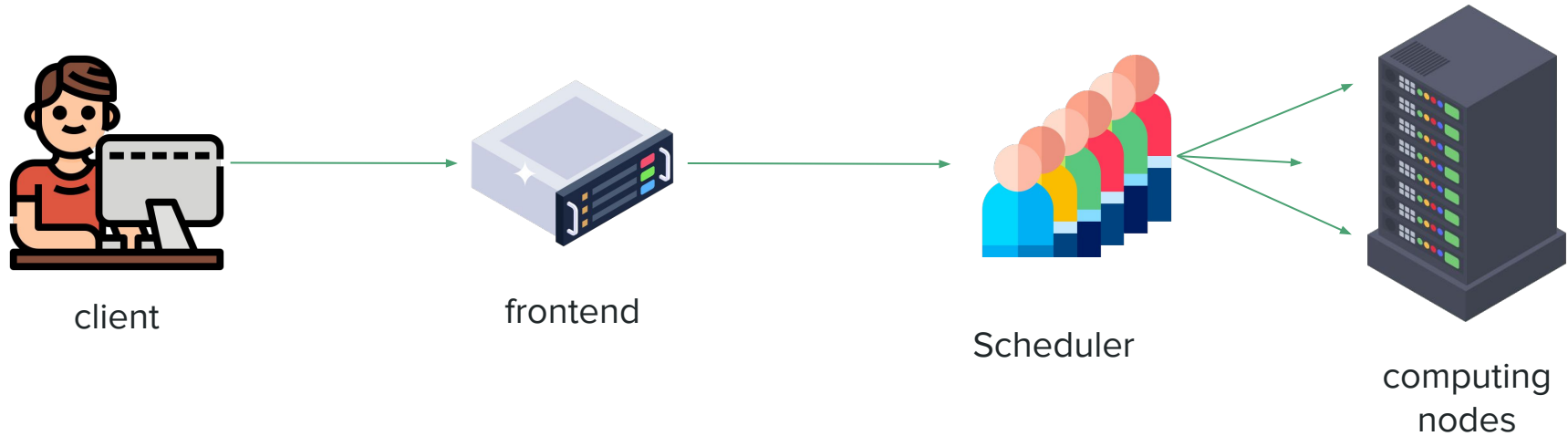
Trieste

- **60 compute nodes without GPUs** (all with 256 GB of RAM)
- **6 compute nodes with 1 GPU each** (Tesla K80 and 128 GB of RAM)

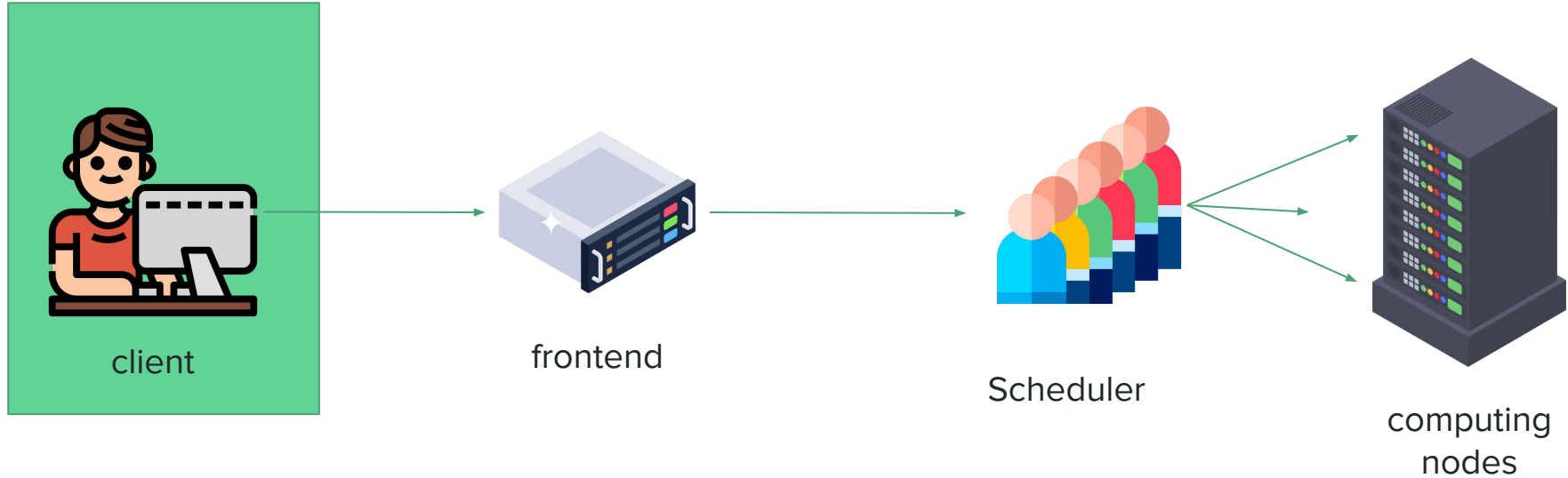
Architecture	Cluster Linux x86_64
Nodes interconnection	Omni-Path HFI Silicon 100 Series, 100 Gbits interconnect
Service Network	Ethernet 1 Gbits
CPU Model	Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz
Number of Nodes	66
Operating System	CentOS Linux release 7.9.2009
Workload manager	SLURM 19.05.50
Storage volume	480TB, BeeGFS parallel filesystem

Accessing Pleiadi

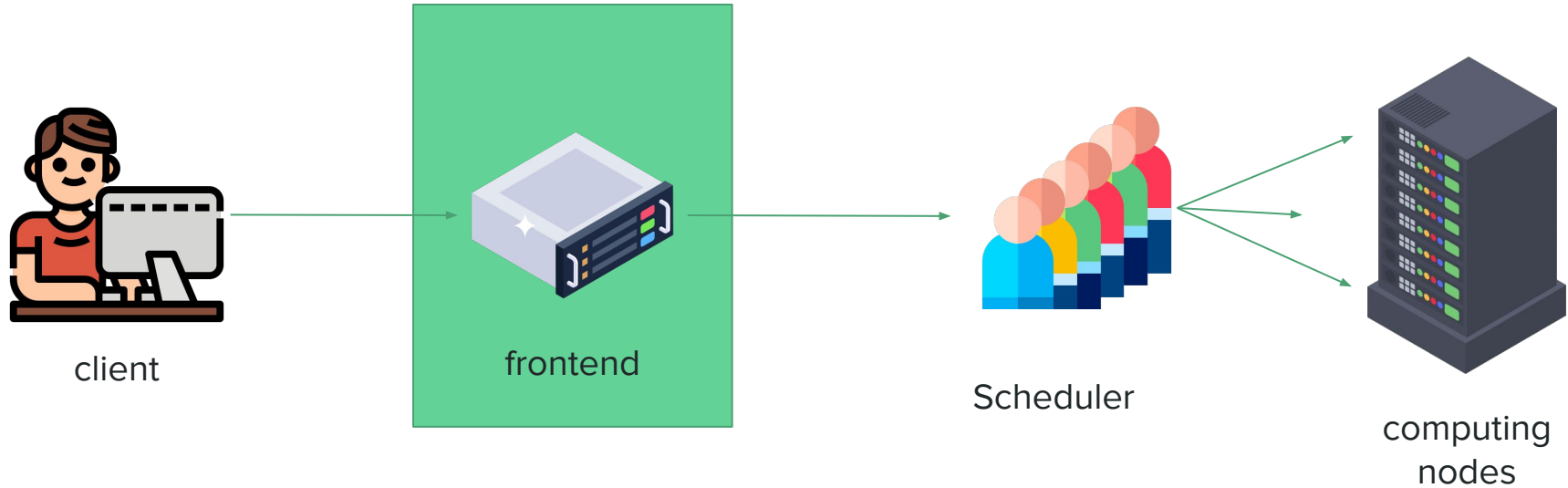
Accessing Pleiadi



Accessing Pleiadi



Accessing Pleiadi



Running jobs on Pleiadi

Cluster Frontend

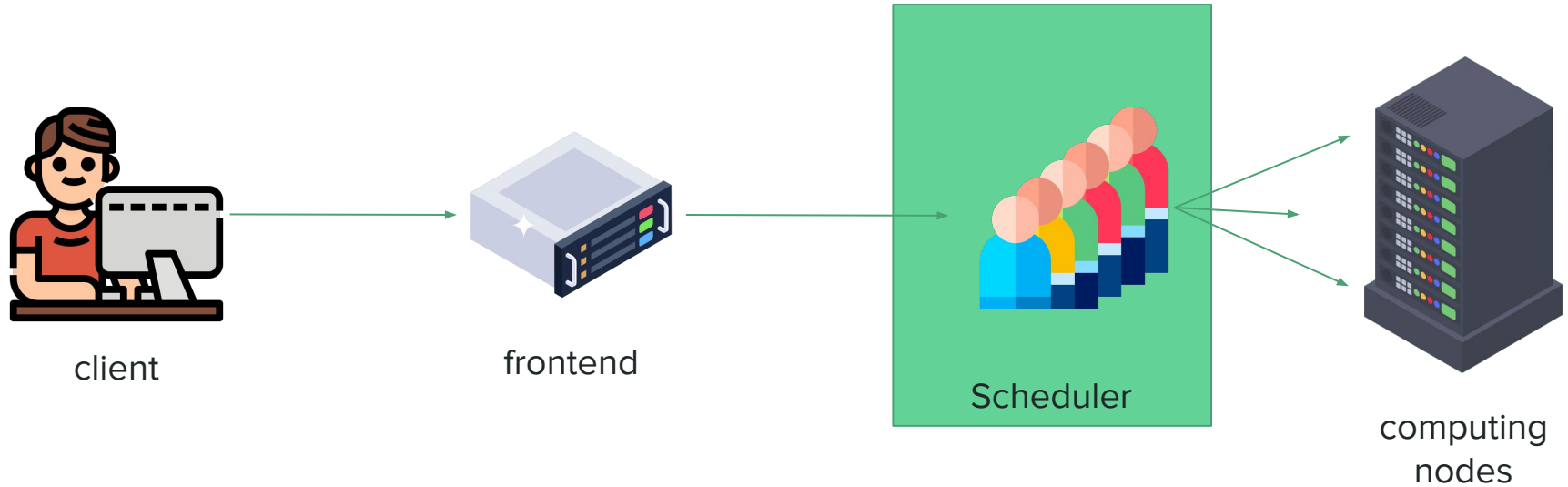
Once you have completed the previous step, you will be connected to what is referred to as a "login node" or "head node." These nodes link the cluster to the outside world and are intended for basic tasks such as:

- Managing files
- Submitting jobs to the compute nodes
- Uploading and downloading data
- Compiling software (*)

While small-scale interactive code and tests are allowed on the login nodes, it is important to remember that these nodes are shared by all users. Therefore, any resource-intensive code must be handled by the compute nodes.

Any resource-heavy jobs found running on login nodes may be terminated without warning.

Accessing Pleiadi



Slurm Job Scheduler

SLURM is the workload manager and job scheduler used on the Pleiadi cluster. It efficiently manages the allocation of computational resources and schedules jobs to run on the compute nodes.

Key points to keep in mind when working with SLURM:

- You must request resources (e.g., CPUs, memory, GPUs) through SLURM commands (**srun/salloc**) or job scripts (**sbatch**).
- SLURM manages job queues, allowing for efficient resource allocation based on priority and availability.
- It can handle both simple single-node jobs and more complex multi-node parallel jobs.

For detailed usage instructions, refer to the official SLURM documentation:

[SLURM Documentation](#)

sbatch

The typical way to submit a job to the compute nodes is by writing a job submission script. This script allows you to specify the resources needed and the commands to run. It is composed of three main parts, which must appear in the following order:

1. **Interpreter Declaration:**

The first line of the script specifies the interpreter that will be used to run the script (e.g., bash).

2. **#SBATCH Directives:**

These lines tell SLURM how to allocate resources for your job, such as the number of nodes, CPUs, memory, and the walltime

3. **Executable Code:**

After the directives, add the actual commands or the script you want to run. This is where you invoke your program or workflow.

sbatch directives - basic settings

#SBATCH lines typically look something like **#SBATCH** <parameter> <argument>

Parameter	Function
--job-name=<name>	job name to be displayed
--output=<path>	Path to the file where the job (error) output is written to
--mail-type=<type>	Turn on mail notification; type can be one of BEGIN, END, FAIL, REQUEUE or ALL
--mail-user=<email_address>	Email address to send notifications to

sbatch directives - Requesting resources

Parameter	Function
--time=<d-hh:mm:ss>	Time limit for job. Job will be killed by SLURM after time has run out.
--nodes=<num_nodes>	Number of nodes. Multiple nodes are only useful for jobs with distributed-memory (e.g. MPI).
--mem=<MB>	Memory (RAM) per node. Number followed by unit prefix, e.g. 16G
--mem-per-cpu=<MB>	Memory (RAM) per requested CPU core
--ntasks=<num_procs>	Number of processes. Useful for MPI jobs.
--ntasks-per-node=<num_procs>	Number of (MPI) processes per node. More than one useful only for MPI jobs.
--cpus-per-task=<num_threads>	CPU cores per task. For MPI use one. For parallelized applications benchmark this is the number of threads.
--exclusive	Job will not share nodes with other running jobs.

sbatch directives - Accounting & Advanced job control

Parameter	Function
--account=<name>	Project (not user) account the job should be charged to.
--partition=<name>	Partition/queue in which to run the job.

Parameter	Function
-dependency=<state:jobid>	Wait with the start of the job until specified dependencies have been satisfied. E.g. -dependency=afterok:123456
-ntasks-per-core=2	Enables hyperthreading. Only useful in special circumstances.

Serial Job

Many simple tools and scripts are not parallelized at all and therefore cannot profit from more than one CPU core.

Parameter	Function
<code>--nodes=1</code>	Start a serial job on only one node
<code>--ntasks-per-node=1</code>	Only one task is necessary
<code>--cpus-per-task=1</code>	Just one CPU core will be used.
<code>--mem=<MB></code>	Memory (RAM) for the job. Number followed by unit prefix, e.g. 16G

OpenMP Job

Parameter	Function
<code>-nodes=1</code>	Start the job on one node
<code>-ntasks-per-node=1</code>	For OpenMP, only one task is necessary
<code>-cpus-per-task=<num_threads></code>	Number of threads to use
<code>-mem=<MB></code>	Memory (RAM) for the job. Number followed by unit prefix, e.g. 16G

MPI Job

Parameter	Function
--nodes=<num_nodes>	Start a parallel job for a distributed memory system on several nodes
--ntasks-per-node=<num_procs>	Number of (MPI) processes per node. Maximum number depends nodes (36 on pleiadi)
--cpus-per-task=1	Use one CPU core per task.
--exclusive	Job will not share nodes with other running jobs. You don't need to specify memory as you will get all available on the node.

sbatch example

An example sbatch script may look something like this:

```
#!/bin/bash
#SBATCH --ntasks 1                #Request 1 tasks (cores)
#SBATCH --nodes 1                 #Request 1 node
#SBATCH --time 0-00:30           #Request runtime of 30 minutes
#SBATCH --partition 256g        #Run on sched_engaging_default partition
#SBATCH --mem-per-cpu=4000      #Request 4G of memory per CPU
#SBATCH --output output_%j.txt  #redirect output to output_JOBID.txt
#SBATCH --mail-type=BEGIN,END   #Mail when job starts and ends
#SBATCH --mail-user=test@test.com #email recipient

echo "Hello World"              #execute the echo command
```

sbatch example

An example sbatch script for an openMP job:

```
#!/bin/bash
#SBATCH --job-name=openmp_job           # Job name
#SBATCH --output=output_%j.txt         # Output file (%j is the job ID)
#SBATCH --ntasks=1                     # Run on a single node
#SBATCH --cpus-per-task=8               # Request 8 CPU cores for OpenMP threads
#SBATCH --time=00:30:00                 # Maximum run time of 30 minutes
#SBATCH --partition=256g                 # Use the "256g" partition

# Set the number of threads for OpenMP
export OMP_NUM_THREADS=8

# Run the OpenMP program
./my_openmp_program
```

sbatch example

An example sbatch script for an mpi job:

```
#!/bin/bash
#SBATCH --job-name=mpi_job           # Job name
#SBATCH --output=output_%j.txt      # Output file (%j is the job ID)
#SBATCH --ntasks=16                 # Request 16 MPI tasks
#SBATCH --nodes=4                   # Request 4 nodes (will distribute tasks
across nodes)
#SBATCH --ntasks-per-node=4         # 4 tasks per node
#SBATCH --time=01:00:00             # Maximum run time of 1 hour
#SBATCH --partition=256g            # Use the "256g" partition

# Run the MPI program
mpirun ./my_mpi_program
```

srun & salloc

Interactive jobs can be executed using either **srun** or **salloc**, allowing for flexible real-time control over allocated resources.

*Both of these commands take slurm directive as **command line arguments** rather than #SBATCH directives in a file.*

salloc is used to allocate resources (e.g., CPUs, memory) for an interactive session, giving you a shell prompt where you can manually launch tasks. This is useful when you want to experiment or run multiple commands interactively within the same session.

Example:

```
salloc --ntasks=4 --cpus-per-task=2 --mem=8G --time=01:00:00
```

srun & salloc

srun is more versatile and can be used in two ways:

Within a resource allocation (like salloc):

After using salloc, you can launch parallel tasks across the allocated resources with srun. This allows you to run your programs within the interactive session.

```
srun --ntasks=4 ./my_program
```

Directly allocate and run a program:

srun can both allocate resources and directly execute a program in one step, without needing a job script. This is useful for running quick, interactive jobs.

```
srun --ntasks=4 --cpus-per-task=2 --mem=4G ./my_program
```

Loading software modules

The user's environment, including available software, is managed through the **module** command. This allows you to load, unload, and switch between different software packages and libraries easily.

View available modules:

To see all the modules available on the system:

```
module avail
```

Load a module:

Once you've found the module you want, load it into your environment using the following commands:

```
module load <module_name>
```

Loading software modules

List loaded modules:

To check which modules are currently loaded in your environment, run:

```
module list
```

Remove a module:

To unload a specific module from your environment, use:

```
module rm <module_name>
```

Remove all modules:

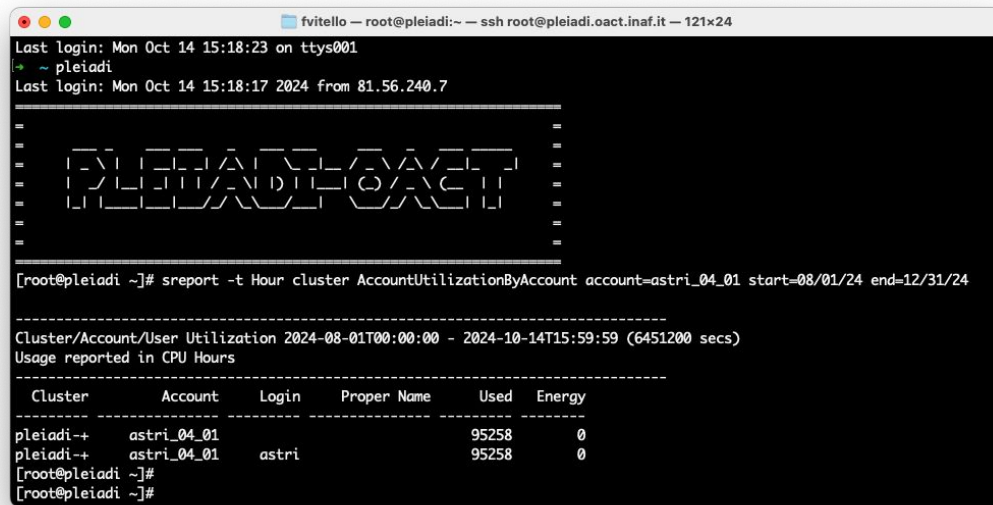
To reset your environment by removing all loaded modules:


```
module purge
```

Monitoring of the used CPU hours

To monitor the utilization of the CPU hours assigned when the account was activated by the board members, according to the user's request, you can use the SLURM report command `sreport`, for example with the following options:

```
$ sreport -t Hour cluster AccountUtilizationByAccount account=<username>  
start=M1/DD1/YY1end=M2/DD2/YY2
```



```
fvitello - root@pleiadi:~ - ssh root@pleiadi.oact.inaf.it - 121x24  
Last login: Mon Oct 14 15:18:23 on ttys001  
→ ~ pleiadi  
Last login: Mon Oct 14 15:18:17 2024 from 81.56.240.7  
=  
=  =  
=  
=  
=  
=  
=  
=  
=  
=  
=  
=  
[root@pleiadi ~]# sreport -t Hour cluster AccountUtilizationByAccount account=astri_04_01 start=08/01/24 end=12/31/24  
-----  
Cluster/Account/User Utilization 2024-08-01T00:00:00 - 2024-10-14T15:59:59 (6451200 secs)  
Usage reported in CPU Hours  
-----  
Cluster      Account    Login    Proper Name    Used    Energy  
-----  
pleiadi+    astri_04_01  
pleiadi+    astri_04_01    astri    astri          95258    0  
[root@pleiadi ~]#  
[root@pleiadi ~]#
```


Pleiadi computing documentation

The information provided in these slides is only a brief overview of how to work with the Pleiadi computing cluster. For comprehensive details on usage, configurations, and specific instructions, please refer to the full documentation for each site:

- **Catania:**

Full documentation is available online:

[Pleiadi @ Catania Documentation](#)

- **Bologna:**

Visit the IRA Wiki for detailed user guidelines:

[Pleiadi @ IRA User Guide](#)

- **Trieste:**

The documentation can be accessed directly from the frontend machine.

Pleiadi team & contacts

Bologna: Francesco Bedosti, Matteo Gandolfi

Catania: Fabio Vitello, Salvatore Scavo

Trieste: Giuliano Taffoni, Gianmarco Maggio

For technical assistance on the assigned resources, ticketing System:

pleiadi-help@ced.inaf.it

For general information:

info.pleiadi@inaf.it