User guide for PLEIADI @ IRA

This guide is aimed at users of our clusters to respond to frequently asked questions about HPC systems

Vs 1.2, September 2022

Contents

Introduction

The current guide illustrates the use of PLEIADI systems based on SLURM scheduler. The document specifically refers to the use of the PLEIADI cluster hosted at the Institute of Radioastronomy of Bologna [PLEIADI @ IRA] but the general information is general parts also applicable to other systems using SLURM.

Gentlemen's agreement

- Using our systems for research purposes, the user automatically authorizes the Staff of the pleiadi.inaf.it node used to publish their personal data (name, surname, research group) and the data associated with the search on the website (pleiadi.inaf.it) and in all other paper publications disseminated by PLEIADI (annual reports, presentations, etc.), as well as in any other media.

- By using our PLEIADI systems for research purposes, the user agrees to mention the PLEIADI services in all their scientific articles in papers, conferences, books or other types of media. We suggest the following quote: "The INAF PLEIADI computing resources (http://www.pleiadi.inaf.it) were used".

Rules for the use of the PLEIADI cluster

- Calculations, simulations, etc. should NEVER be run directly from the command line. The code will run on the login node which is NOT part of the PLEIADI system. In addition the login node will hang, making it unusable. ALWAYS go through the execution queues of the scheduler (even for compilation).

- Use of computing resources outside the training and / or research activities is NOT allowed.

- The Project PI is responsible for any activity carried out or attributable to the usage of the account and for the distribution of the login credentials to the project collaborators

The PLEIADI @ INAF project provides HPC computing resources and technical support for research and training activities.

The computing resources, hosted and managed at three INAF structures (O.A.Trieste, O.A.Catania, IRA Bologna) are administered in a coordinated manner by a board (board.pleiadi@inaf.it) which assigns the computing time and credentials for the use of resources.

The three centers provide diversified resources and it is the task of the board to direct users to the different centers according to specific requests and needs.

Each project group will be assigned an account that will allow access to the resources listed below. Additional accounts can be assigned to the single project and all accounts will be part of the same UNIX group, with an independent / home but sharing the working storage.

Account expiration and data storage policy

Account

The account is to be considered active from the moment you receive the confirmation email containing your login credentials. The account expires as indicated in the application form. To renew the account, the same form sent in the first request must be sent again, changing the fields with the updated information.

Data Storage

If the account is not reactivated within 6 months after the deadline indicated in the application form, two compressed archives are created containing all the files in the home directory and in the work directory. Compressed archives are placed in the same home directory for the home directory and in your own directory in work for the work directory. After a further 6 months period from the creation of the compressed archives (a total of one year from the expiration of the account), all the archives are permanently removed from the storage and the account is deleted.

Storage Quota

Storage quota available to users:

- `/homes/<user>` The quota of the HOME directory is 50 GBy
- `/iranet/home2/<user>` Each group can have up to 10 TBy of space

Users can request an increase in the group quota by sending us an email detailing and justifying the request. The request will then be evaluated by the local staff. Any extensions to the quotas relating to the space that can be used for user data must have an expiration and its duration has to be as short as possible. The extension cannot exceed the expiry date provided for the account.

For accounts that at the monthly check following the expiration of the account results to be over-quota, the data will be archived as soon as the normal grace time granted (7 days) has elapsed. If at the next monthly check the account is still over-quota, the archived files will be removed.

First access

The usernames provided to users are of the "pleiaXX" type (pleia01, pleia02 ... pleia99). Here “pleia0x” is used as an example. We strongly recommend changing the password on the first log-in using the following command:

```
$ passwd
```

It is strongly recommended to choose a password consisting of more than 8 characters, containing numbers and upper and lower case letters and special characters.

The users can access the servers: **scheduler.ira.inaf.it** and **gaia.ira.inaf.it,** having access to the corresponding working areas.

- **scheduler** it is the server that allows you to launch batches on the HPC cluster
- **gaia** allows the efficient transfer of your data between Pleiadi and your local computers, also provides a window manager via X2Go

Access to the cluster depends on the operating system used on your computer. If you have a Linux, Unix or OSX system, you can use the ssh client from any terminal using the command:

```
$ ssh pleia0x@scheduler.ira.inaf.it
```

If you have a Windows system, we recommend using the PuTTY application (available at http://www.putty.org) to be configured as shown in the figure:

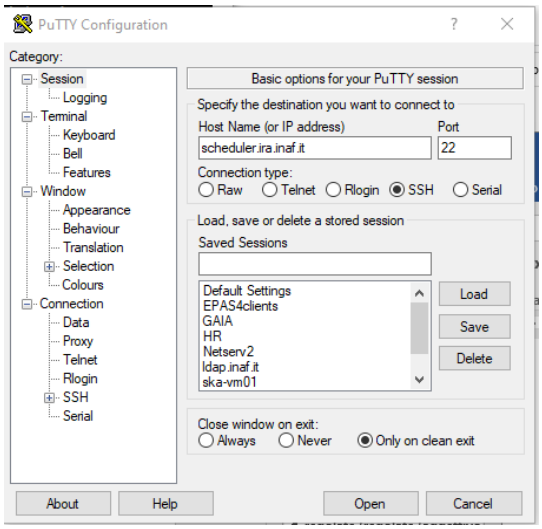

Figure 1: Putty - Configuration for access from Windows systems

For graphical access, you can use X2Go (https://wiki.x2go.org) for which a server is available on Gaia.

File transfer and storage

Data Transfer

At log in, the user is in her/his home directory:

/homes/*pleia0x*/

Access to this directory is allowed only to the owner user who can use up to 50 GBy of disk space. The user and other group members can request and share up to 10Tby of space in the /iranet/home2/xxx directory

To transfer files from your unix/linux host to your home directory on the Pleiadi cluster, you can use the scp command:

```
$scp -r /local/dir/file pleia0x@gaia.ira.inaf.it:/homes/pleia0x
```

Vice versa, to copy a file from Pleiadi to your host use the following command:

```
$scp -r pleia0x@gaia.ira.inaf.it:file /local/dir/
```

On PC windows or Mac you can use programs such as FileZilla or WinSCP

Storage on Pleiadi nodes

For each job it is possible to temporarily use the additional space present on the local disks of the computational nodes. This space is very limited:

/local/scratch 80 Gby

WARNING:

1. The filesystem is not accessible from the other nodes of the cluster
2. The available space depends on the use of the resource also by other jobs
3. It is compulsory to delete what is present in /local/scratch at the end of the job. Data left on this areas will be deleted with no notice or backup by the system administrators
4. If your program saves output in the /local/scratch folder, it must be transferred to the assigned storage areas.

Example sbatch

[esempio_script_con_tmpdir.sbatch]. . .

```
#!/bin/bash
#SBATCH --job-name=job_name
#SBATCH --mail-type=ALL
#SBATCH --mail-user=nome.cognome@inaf.it
#SBATCH  --partition=pleiadi
#SBATCH --time=hh:mm:s
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --output=job_name_%j.log
# Copy files from storage HOME to local storage
cp -r /homes/pleia0x/data /local/scratch/data

# Run the code
myexecutable -in /local/scratch/data -out /local/scratch/results

# Copy data back from the local storage
cp -r /local/scratch/results /home/myusername/results

# Clear local scratch area
rm –r /local/scratch/results
```

. . . . . .

Pleiadi@IRA Technical specifications

Support: pleiadi-support@ira.inaf.it

Available Software : https://wiki.ira.inaf.it/wiki/index.php/Irasoft

Computing Nodes	
Architecture	Cluster Linux Infiniband-DDR MIMD Distributed Shared-Memory
Nodes interconnection	Omnipath 100 Gb/s
Rete di Servizio	Ethernet 1 GB/s
Modello CPU	2x INTEL Xeon E5-2697 V4 (turbo 3.6 GHz) 18 cores
OS	Debian GNU/Linux 11
Scheduler	SLURM 20.11
Performance	**FP32 0.66 TFLOPS / FT64 0.33 TFLOPS**

Batch Codes	**Pleiadi**		
Number of Nodes	24		
Cores	864		
RAM memory Total/core	256 GBy / 7.0 GBy		
Locale disk /local/scratch	80 GBy		
Max. Time x job	3 days		
Max Core/hours per job	50.000		
USO	HPC		

Storage		
Home Storage	/**homes**/*user*	50 GBy per User
Working Storage	**/iranet/home2**/*user*	10 TBy per Group (Lustre Filesystem)
Scratch	**/local/scratch**	80 Gby
Remote access for tranfer file	Ethernet 10Gbit/s	

HPC batch frontend : **scheduler.ira.inaf.it**

Frontend for data transfers and graphic access via X2Go: **gaia.ira.inaf.it**

Using the PLEIADI cluster with SLURM

Batch processes have to be executed on the PLEIADI system: this means that they are not interactive and that their execution can be postponed over time. Each task or job is made up of one or more processes that cooperate together to achieve the targeted result.

A task is executed only after its scheduling; to allow scheduling, the job must be placed in a queue, managed by the cluster, where it waits for the necessary resources to be available. The PLEIADI system uses the SLURM scheduler.

Job Submission

`$ sbatch`

All jobs must be passed to the cluster's scheduler via submission. The submission of jobs takes place via the sbatch command which has as argument the complete name of a script containing all the necessary information.

The script file, also called sbatch script, is composed of a series of directives for the scheduler, followed by a number of commands as they should be typed on the command line.

An example of a sbatch script is the following, all lines starting with `#SBATCH` are options for the scheduler, and are not interpreted by the Linux shell.

……..[script.sbatch]……

```
#!/bin/bash
#SBATCH --job-name=job_name
#SBATCH --mail-type=ALL
#SBATCH --mail-user=nome.cognome@inaf.it
#SBATCH  --partition=pleiadi
#SBATCH --time=hh:mm:ss
#SBATCH --nodes=1
#SBATCH --ntasks-per node=32
#SBATCH --output=job_name_%j.log
#
example_task.bin
```

……..

Now you can submit the script with:

`$ sbatch script.sbatch`

To create the script.sbatch file you can use your PC and then transfer it to the cluster, or edit it directly on the cluster.

NOTES: text files created on DOS / Windows hosts have a different newline character than those created with Unix-like systems. DOS uses 'carriage-return' and 'line feed', while Unix simply uses 'line-feed'. During a file transfer between Windows and Unix hosts, care must therefore be taken to ensure that the end-of-lines are correctly translated.

Main Directives

Directive	Description
`--output`	Standard output is redirected to the specified file, by default both standard output and standard error are redirected to the same file.
`--error`	Standard error is redirected to the specified file
`--mail-user`	e-mail to send information on the task in progress.
`--mail-type`	Events triggering a notification via e-mail (eg ALL). Valid values are: NONE, BEGIN, END, FAIL, REQUEUE, ALL.
`--workdir={directory}`	Executes the task using the specified {directory} as the working directory (either an absolute or relative path can be specified).
`--ntasks-per-node`	Number of tasks per node, if used together with --ntasks this latter directive will take precedence.
`--cpus-per-task`	Required Number of CPU per task.
`--ntasks`	Total number of tasks per job
`--nodes`	Number of nodes to use
`--time`	Specifies the maximum run-time limit, which is the time it takes for the process to reach the end of the computation. This value must be less than 10 days (<240 hours) per partition task.
`--mem-per-cpu`	Specifies the minimum memory required per allocated CPU, expressed in megabytes (default = 1000)
`--mem`	Specifies the actual memory required per node, expressed in megabytes.
`--partition`	Indicates the partition (queue) on which the job is to be scheduled
`--exclude`	Explicitly excludes the specified nodes from the resource set
`--constraint`	Nodes have some features assigned, users can specify which features will be required by their job using this directive, for example --constraint = "gpu". The available features can be viewed in the Scheduling pool data section in the output of the sjstat command (Other Traits column), or via scontrol show node.
`--gres=gpu:{N_of_gpu}`	Generic resource required per node, used to specify GPU request per node (if GPUs are available)

For the full guide refer to: https://slurm.schedmd.com/sbatch.html.

Job Control (sinfo, squeue, scancel, sstat)

To get information on the status of the cluster nodes and on the partitions (queues) available

$ sinfo

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
pleiadi      up 1-00:00:00     12  down* r36c02s[01-12]
pleiadi      up 1-00:00:00     24   unk* r36c05s[01-12],r36c06s[01-12]
pleiadi      up 1-00:00:00     12   idle r36c01s[01-12]
```

To obtain information on the jobs scheduled with sbatch, such as job-ID, status, partition and the use of the slot number, you can use the following command:

$ squeue -u username

```
   JOBID PARTITION     NAME     USER   ST       TIME    NODES  NODELIST(REASON)
    600      pleiadi     test  username  R  2-23:29:19      1   compute-0-1
    601      pleiadi     test  username  R     3:53:04      3   compute-0-[2-4]
    602      pleiadi     test  username  R        7:55      1   compute-0-8
```

To get more information on the status of each specific job use:

$ scontrol show job 600

```
  JobId=600 JobName=test
   UserId=test(500) GroupId=test(500) MCS_label=N/A
   Priority=30937 Nice=0 Account=test QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=04:07:05 TimeLimit=8-08:00:00 TimeMin=N/A
   SubmitTime=2018-02-13T10:25:11 EligibleTime=2018-02-13T10:25:11
   StartTime=2018-02-13T10:25:12 EndTime=2018-02-21T18:25:12 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=pleiadi AllocNode:Sid=casperlogin:30786
   ReqNodeList=(null) ExcNodeList=compute-0-8 NodeList=compute-0-[2-4]
   BatchHost=compute-0-2
   NumNodes=3 NumCPUs=96 NumTasks=96 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=96,mem=240G,node=3
   Socks/Node=* NtasksPerN:B:S:C=32:0:*:* CoreSpec=*
   MinCPUsNode=32 MinMemoryNode=40G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   Gres=(null) Reservation=(null)
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/home/test/test.bin
   WorkDir=/home/test/
   StdErr=/home/test/job_600.log
   StdIn=/dev/null
   StdOut=/home/test/job_600.log
   Power=
```

To get information on the entire cluster and on all running jobs use:

$ squeue

```
JOBID PARTITION NAME   USER     ST      TIME  NODES NODELIST(REASON)
```

```
520   pleiadi Bbrune_1 pleia0x PD       0:00      1 (Resources)
508   pleiadi Bbrune_1 pleia0x  R       0:19      1 r36c01s01
509   pleiadi Bbrune_1 pleia0x  R       0:15      1 r36c01s02
510   pleiadi Bbrune_1 pleia0x  R       0:15      1 r36c01s03
511   pleiadi Bbrune_1 pleia0x  R       0:15      1 r36c01s04
512   pleiadi Bbrune_1 pleia0x  R       0:12      1 r36c01s05
513   pleiadi Bbrune_1 pleia0x  R       0:12      1 r36c01s06
514   pleiadi Bbrune_1 pleia0x  R       0:12      1 r36c01s07
515   pleiadi Bbrune_1 pleia0x  R       0:12      1 r36c01s08
516   pleiadi Bbrune_1 pleia0x  R       0:09      1 r36c01s09
517   pleiadi Bbrune_1 pleia0x  R       0:09      1 r36c01s10
518   pleiadi Bbrune_1 pleia0x  R       0:09      1 r36c01s11
519   pleiadi Bbrune_1 pleia0x  R       0:09      1 r36c01s12
```

To stop and remove a job from a partition use:

$ scancel jobID

To collect statistics on currently running jobs you can use:

$ sstat --format=JobID,MaxRSS,AveRSS,AveCPU,NTask -j $JOBID --allsteps

```
 JobID         MaxRSS   AveRSS      AveCPU     NTasks
-------------------------------------------------------
 JOBID.0        3248K    3248K      00:00.000     1
```

The above command only works for jobs run through srun interactive mode, for batch type jobs add .batch to the specified $ jobID, as in the following example:

$ sstat --format=JobID,MaxRSS,AveRSS,AveCPU,NTask -j ${JOBID}.batch --allsteps

```
JobID             MaxRSS    AveRSS    AveCPU    NTasks
……..
${JOBID}.batch   9488K      9488K  07:24.000     1
```

Interactive sessions on computing nodes

Whenever it is necessary to use an interactive session on a computing node, as, for instance, for compilation, the use of the srun command is useful.

From the login node, an interactive session can be started using the **srun** command with the following syntax:

$ srun --nodes=1 --tasks-per-node=1 --pty /bin/bash

The srun options corresponds to the directives of a sbatch script (see above)

After use, the node must be released with the following procedure:

$ sinfo

```
JOBID PARTITION NAME  USER     ST    TIME NODES NODELIST(REASON)
588  pleiadi interact pleiadi0  R    3:36:54      1 r36c01s01
```

$ scancel 588

Please note that within a srun session only one node can be used for MPI applications. Trying to use more than one node will cause mpirun to fail. In this case the salloc command can be used instead of srun. We refer to the complete SLURM reference manual: https://slurm.schedmd.com

Execution of MPI runs

SLURM provides a variety of options to configure how resources should be reserved for the job. The following table summarizes the main ones:

`--nodes`	Each multi-core CPU corresponds to a node and their number is specified through the directive --nodes
`--ntasks`	Set the number of MPI processes
`--ntasks-per-node`	It offers the possibility to control the number of tasks per single node
`--cpus-per-task`	Sets the number of OpenMP threads per MPI task

Example of script sbatch MPI

…… *MPI case 1 [mpi1.sbatch* ……

```
#!/bin/bash
#SBATCH --ntask=16
#SBATCH --cpus-per-task=1
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
```

……..

This example requires 16 tasks, corresponding to 16 MPI processes, to which one CPU is associated per task. All 16 tasks are bound to be executed on a single computation node. No multithreading is active.

…… MPI esempio 2 [mpi2.sbatch]… ……..…….

```
#!/bin/bash
#SBATCH --ntasks=32
#SBATCH --cpus-per-task=1
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=16
```

……...

In this example, 32 tasks are required split between 2 nodes, with 16 tasks for each node

……MPI esempio 3 [mpi3.sbatch……..

```
#!/bin/bash
```

```
#SBATCH --ntasks=32
```
……..

In this third example 32 tasks are required and the choice of their distribution within the cluster is left to the scheduler

……....Casi d'uso MPI e OpenMP [mpiOpenMP.sbatch]…….

```
#!/bin/bash
#SBATCH --ntasks=16
#SBATCH --cpus-per-task=4
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=4
```
……

In this case, an application is run which, for each MPI process (or rank), uses multiple cores (via multithreading). 16 tasks (--ntasks = 16) and 4 cores per task (--cpus-per-task = 4) are required, so 16 * 4 = 64 total cores. The 16 tasks are split between 4 nodes (--nodes = 4) with 4 tasks per node (--ntasks-per-node = 4)

Available applications and MPI compiler

The Pleiadi@IRA Cluster can use the programs and libraries available at the IRA computing center. The list of applications can be found at: https://wiki.ira.inaf.it/wiki/index.php/Irasoft.

It is also possible to have a short list using the **`$setup-help`** command.

Before using an application or library it is necessary to define the dependencies with the "$xxxx-setup" command, where xxxx is the name of the specific application. The setup allows you to choose the software version to use.

To compile MPI programs it is necessary to use the **`$openmpi-setup`** command which configures the environment for MPI. Then you find the classic commands: **`mpif90`**, **`mpicc`**, **`mpiCC`**, **`mpirun`**, **`mpiexec`**

APPENDIX A: generate submission scripts from csv files

For advanced tasks that require multiple execution of the same program, but with different parameters, we suggest the use of the following script capable of generating multiple sbatch files starting from a text file containing all the parameters.

.....[generate.sh].....

```
#!/bin/bash
#
list=$(cat ./parameters.csv)
#
for i in $list
 do
   parameter1=$(echo $i|cut -f 1 -d ',') parameter2=$(echo $i|cut -f 2
   -d ',') cat << EOF > ./test-$parameter1-$parameter2.sbatch
   #!/bin/bash
   #SBATCH --job-name=job_name
   #SBATCH --mail-type=ALL
   #SBATCH --mail-user=nome.cognome@inaf.it
   #SBATCH --partition=pleiadi
   #SBATCH --time=24:00:00
   #SBATCH --nodes=3
   #SBATCH --ntasks-per-node=32
   #SBATCH --output=job_name_%j.log
   #
   example.bin $parameter1 $parameter2
   EOF
done
```

.....[parameter.csv].....

```
A,1
A,2
B,1
B,2
```

.....

APPENDIX B: Singularity container (Apptainer)

To offer users maximum flexibility, Singularity containers are supported. The complete reference manual is available here: https://apptainer.org/docs/

Below is an example of creating and running a container with TensorFlow.

Step1: create a Singularity recipe file

There are several ways to create a container, one of them is starting from a Docker image. In this example, we start with an NVIDIA Docker image

Create a tensorflowCentos.recipe file on your PC with the following content:

```
BootStrap: docker
From: nvidia/cuda:9.0-cudnn7-devel-centos7 # This is a comment

%runscript
  echo "Hello from tensorflow
  container" whoami
%post
  echo "Hello from inside the container"
  yum -y update && yum install -y epel-
  release yum install -y python-pip
  python-devel
  pip2 install matplotlib h5py pillow tensorflow-gpu keras scikit-learn
```

Step2: build del container Singularity

There are different types of containers, in this case an immutable image will be created.

```
$ sudo singularity build tensorflowCentos.img tensorflowCentos.recipe
```

Step3: copy of the container to the cluster

```
$ scp tensorflowCentos.img {username}@{login-node}:/home/{login-
node}/tensorflowCentos.img
```

Step4: create the submission script

Log into the login node and create the following python file in your own home.

```
/home/{user name}/helloTensorflow.py
      import tensorflow as tf
      hello = tf.constant("Hello, TensorFlow!") sess = tf.Session()
      print sess.run(hello)
      exit()
```

Create a singularity.sbatch file based on the following example:

```
#!/bin/bash
#SBATCH ——time =00:10:00
#SBATCH ——ntasks=1
#SBATCH ——partition=cuda
#SBATCH ——gres=gpu :1
#SBATCH ——job—name=singularityTest
#SBATCH ——mail—type=ALL

module load singularity /2.4.5

singularity exec ——nv tensorflowCentos . img bash —c ’ python
helloTensorflow . py ’
```

Now you can submit the job with the sbatch command:

```
sbatch singularity.sbatch
```

[Adaptation to the PLEIADI@IRA system of the guide of Politecnico di Torino : https://www.hpc.polito.it/docs/guide-slurm-it.pdf]