Avviso MUR DD3264 del 28/12/2021
Missione 4, Componente 2, Investimento 3.1
(Decreto Concessione n.415 del 27/10/2022)
Codice progetto MUR: IR0000034, CUP: C33C22000640006
STILES - Strengthening the Italian Leadership in ELT and SKA

ADHOC – Astrophysical Data HPC Operating Center User Guide

1 Introduction

ADHOC is a Tier 2 computing infrastructure integrated into a pre-existing computing center called DC1, with a full-capacity computing power of 2 Petaflops, equipped with a total storage system of 20 Petabytes (PB) "raw," divided into two systems: *(i)* 8 PB of long-term tape storage (two units, respectively, 6 PB located at the INAF data processing center in Bologna, and 2 PB at DC1); *(ii)* 12 PB of "hot" storage on disks, divided into 4 logical units of 3 PB each and managed by dedicated front-end servers; a computing system, divided into two subsystems, HPC (High Performance Computing) and HTC (High Throughput Computing), logically divided into three modular clusters of different sizes. The HPC system includes 34 dual processor servers, 22 of which are equipped with dual H100 GPU cards and 12 with dual L40 GPU cards, dedicated to high-performance parallel computing. The HTC system includes 10 dual-processor servers dedicated to multiprocessing (MPI) and, in part, to the management and interfacing of the various devices that make up the entire infrastructure. Three additional HTC/HPC servers are separated from the clusters and dedicated to specific projects and services.

A data network and monitoring system connected via a redundant 10 Gb/s line to the GARR node of the CSI (University Center for IT Services), located in

the same university complex. The data network architecture consists of various subnetworks, respectively, at 10/25 Gb/s for interfacing with the outside world and infrastructure management/monitoring, at 100 Gb/s for interfacing with the storage system and HTC, and at 200 Gb/s for Infiniband connections with the HPC system. From a functional point of view, the ADHOC data center consists of three clusters of computing and storage resources of varying sizes, intended for incremental and differentiated use. The three clusters are called *Newton*, *Fermi*, and *Einstein*, and refer, respectively, to the three small, medium and large clusters.

The clusters are equipped with advanced tools for resource management and optimization.

- User management and authentication with FreeIPA;
- SLURM job scheduler for efficient resource allocation;
- Parallel computing with OpenMPI;
- Acceleration on NVIDIA GPUs of different sizes and computational capacities;
- Shared file system based on different protocols;
- Support for Python and scientific libraries for data analysis and simulations.

2 Access to the ADHOC Cluster

Access Credentials

Each user receives personalized credentials to access the ADHOC-Newton Cluster:

Username: $< username >$
Initial Password: $\$\$cambiami$
Hostname: newton.stiles.unina.it

▲ On the first login, you will be asked to change your password. At the same time, a personal Conda virtual environment named $< username >$ will automatically be created.

VPN Connection

For security reasons, access to the cluster from external networks is possible only via VPN.

You will receive a personalized VPN configuration file named $ADHOC\_ < username > .ovpn$

For the installation of VPN Client, we suggest the following software:

- Windows/Linux: install and use OpenVPN `https://openvpn.net/client/`.
- macOS: install Tunnelblick, a free client compatible with .ovpn files `https://tunnelblick.net/downloads.html`.

Activation

Install the VPN client.
Import the file $ADHOC\_ < username > .ovpn$.
Start the VPN connection.
✓ No credentials will be required during VPN activation.

SSH Access

Once the VPN is active, connect to the cluster using SSH:

$$ssh < username > @newton.stiles.unina.it$$

Data Storage

Do not use your home directory to store data.
A dedicated folder has been created for you at:

$$/data1/ < username >$$

Please use this directory to store and write your data.

3 Launching jobs on the nodes

SLURM (Simple Linux Utility for Resource Management) is the workload manager used on the ADHOC-Newton cluster to allocate resources and run jobs on compute nodes. This section provides a practical introduction to submitting jobs using SLURM.

Interactive vs. Batch Jobs

There are two main ways to run tasks on the cluster:
Interactive jobs: allow the user to interact directly with a compute node. Useful for debugging or testing small scripts.
Batch jobs: the recommended method for production runs, where jobs are submitted via a script and executed by SLURM when resources become available.

Interactive Jobs

You can request an interactive session on a compute node using:

```
srun --partition=standard --nodelist=compute-node41 --ntasks
    =1 --cpus-per-task=4 --time=02:00:00 --pty bash
```

Explanation of the options:

- partition=standard → the queue/partition to use.
- nodelist=compute-node41 → specify the node (optional).
- ntasks=1 → number of tasks (MPI processes).
- cpus-per-task=4 → number of CPU cores per task.
- gres=gpu:1 → number of GPUs, this option has to be used only for nodes that have GPUs.
- time=02:00:00 → walltime limit (hh:mm:ss).
- pty bash → opens an interactive bash shell.

Once granted, you will be logged into the compute node and can run commands interactively.

Batch Jobs

For larger jobs, you should create a job script. This is a text file with SLURM directives and the commands you want to execute.

Listing 1: Example SLURM job script

```
#!/bin/bash
#SBATCH --job-name=test_job
#SBATCH --output=output_%j.txt
#SBATCH --error=error_%j.txt
#SBATCH --partition=normal
#SBATCH --nodelist=compute-node41
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --gres=gpu:1
#SBATCH --time=04:00:00

# Load required modules or environments
module load python/3.10

# Execute your program
python myscript.py
```

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*      up   infinite      1    mix compute-node45
normal*      up   infinite      4   idle compute-node[13-14,40-41]
gpusH100     up   infinite      2   idle compute-node[13-14]
gpusL40      up   infinite      2   idle compute-node[40-41]
gpusALL      up   infinite      4   idle compute-node[13-14,40-41]
```

Figure 1: Output of the command *sinfo*

Listing 2: Submit the job

```
sbatch job.slurm
```

SLURM will place the job in the queue and execute it when resources are available. The %j symbol in –output and –error will be replaced by the Job ID.

Monitoring Jobs

You can check the status of your jobs using the following:

```
squeue -u <username>
```

Other useful commands:

```
scancel <jobid> # cancel a running or pending job.
```

```
scontrol show job <jobid>  # detailed information about a
    job.
```

Resource Considerations

- Always request the minimum resources your job needs.
- Overestimating time or CPU usage may result in longer queue times.
- Jobs that exceed their allocated resources will be terminated by SLURM.

Before starting a job on a node, it is recommended to verify the most appropriate partition and check nodes currently available.

Listing 3: check the available nodes

```
sinfo
```

The expected output looks like Figure 1, where the first column lists the name of the partitions, the second column indicates if the partition is available, the third column shows the maximum wall-clock time a job can run in that partition (or queue), the node column shows the nodes that are included in the specific partition and state indicates if the node of the specific partition is available or not. Finally, the nodelist column lists the names of the nodes included in a partition that are in a given state.

Good Practices

- Use interactive jobs only for short tests.
- For reproducibility, prefer batch jobs with clearly defined scripts.
- Redirect all output to files (–output and –error) to avoid losing information.
- Store your data in the dedicated storage directory $/data1/ < username >$, not in your home directory.