

Getting Started on Jean Zay

Research and Application of Artificial Intelligence



The IDRIS User Support Team - October 2024

Why this documentation?

- This presentation is intended to be a guide for new AI users of Jean Zay.
- The information here is designed as a synthesis to facilitate a rapid start on the supercomputer.
- Complete documentation is updated regularly by the IDRIS User Support Team on the IDRIS Web site. →
- A FAQ is available online. →

NB: In this presentation, the \rightarrow symbols represent hypertext links..



Syllabus

Principal characteristics of Jean Zay Administrative environment Access to computing resources Machine environment Computing environment Job submission Consumption of computing hours To go further



Principal characteristics of Jean Zay



The IDRIS User Support Team - October 2024

Principal characteristics of Jean Zay

- Jean Zay is a computer composed of two partitions:
 - ► A scalar or "CPU" partition containing 28800 processing cores
 - \blacktriangleright An accelerated or "GPU" partition containing 3704 GPUs with various architectures

Partitions	#nodes	RAM CPU	#cores/node	#GPU/node	RAM GPU
CPU	720	192 GB	40 cores [⊗]		
	126	192 GB	40 cores $^{\otimes}$	4 V100 [⊳]	16 GB
	270	192 GB	40 cores [⊗]	4 V100⊳	32 GB
CPU	20	384 GB	24 cores [⊖]	8 V100⊳	32 GB
GFU	11	768 GB	24 cores [⊖]	8 V100⊳	32 GB
	52	512 GB	64 cores [⊘]	8 A100►	80 GB
	364	512 Go	96 cœurs [⊕]	4 H100 ^{>}	80 Go

- (\otimes) 2 Intel Cascade Lake 6248 processors (2 \times 20 cores at 2,5 GHz)
- (\ominus) 2 Intel Cascade Lake 6226 processors (2×12 cores at 2,7 GHz)
- (\odot) 2 AMD Milan EPYC 7543 processors (2×32 cores at 2,8 GHz)
- (\oplus) 2 Intel Sapphire Rapids 8468 processors (2×48 cores at 2,1 GHz)

(⊳) Nvidia V100 SXM2 HBM2

- (►) Nvidia A100 SXM4 HBM2e
- (>) Nvidia H100 SXM5 HBM3



Principal characteristics of Jean Zay

- Cumulated peak performance: 126 Pflops/s
- OmniPath (100 Gb/s) and InfiniBand (400 Gb/s) interconnection networks
- Lustre parallel file system
- Three storage technologies:
 - ightarrow 7.6 PB on SSD Full Flash disks (1.5 TB/s in read and 1.1 TB/s in write)
 - $\rightarrow\,$ 39 PB on rotating disks (350 GB/s in read and 300 GB/s in write)
 - $\rightarrow~$ 50 PB on magnetic tapes
- 5 frontal nodes
- 4 pre- and post-processing nodes
- 5 visualisation nodes

A complete hardware description is available on line. ->



Administrative environment



The IDRIS User Support Team - October 2024

Administrative environment

NFORMATIOUE

SCIENTIFICIE



Access to computing resources



Access to computing resources

- To compute on Jean Zay, it is necessary to have:
 - ► A project, i.e., an allocation of computing hours (procedure managed by GENCI)
 - $\rightarrow~$ Description of the scientific project
 - $\rightarrow~$ Estimate of the number of computing hours necessary
 - $\rightarrow~$ Which computing libraries you will need
 - ightarrow etc
 - A computing account (procedure managed by IDRIS)
 - $\rightarrow~$ Request an account opening
 - $\rightarrow~$ Declare the connection IP addresses
 - $\rightarrow\,$ Declare the security manager of your laboratory
 - ightarrow etc
- These two procedures are done via the eDARI portal of GENCI. You must, therefore, first create a user account on this portal. →
- Detailed documentation of these procedures is available on the IDRIS Web site. \Rightarrow



Access to computing resources - Project

- Requests for computing hours are managed by GENCI (acces@genci.fr).
- The access procedures are explained on line (French only). →
- There are two types of access to resources, depending on the number of hours requested.

Regular Access (AR)	Dynamic Access (AD)
\geq 500k CPU hours	< 500k CPU hours
\geq 50k normalized GPU hours *	$<$ 50k normalized GPU hours *
Two allocation sessions per year:	Open throughout the year:
- In May (submit file before Feb.)	The validation of an AD
- In Nov. (submit file before Sept.)	takes several days.
Technical and scientific expert assessment	No expert assessment

(*) 50k V100 hours or 25k A100 hours or 12.5k H100 hours



Access to computing resources - Project

- It is possible to request CPU, V100 GPU, A100 GPU and H100 hours at the same time.
- An allocation is usable for **one year** after the project opening.
- A project can be **renewed** at the end of a year via the eDARI portal.
- Need complementary hours?
 - ► For Regular Access files, complementary hours can be requested at mid-point, during one of the annual allocation sessions.
 - Exceptional requests for supplementary hours as needed ("demandes au fil de l'eau") may be submitted throughout the year on the eDARI portal. These hours are attributed in function of the machine workload.



Access to computing resources - Computing account

- The opening of a computing account is managed by IDRIS (gestutil@idris.fr).
- The creation of a computing account takes a few days.
 - ▶ Important: IDRIS is a "Zone à Régime Restrictif" and some users may be submitted for a ministerial inquiry (duration 8 weeks maximum).
- A computing account can be attached to multiple projects.
- You can modify your computing account at any time (add an IP connection address, detach an account from a project,...) via the Administration Form for Login Accounts (FGC). →



Machine environment



Machine environment - Connection

• Connecting to Jean Zay is done via SSH:

\$ ssh login@jean-zay.idris.fr

- The connection must be initiated from an IP address which is fixed and institutional. This address must be declared and associated to your computing account during its opening or via the Administration Form for Login Accounts (FGC). →
- If you work under Windows, the connection can be done via an SSH client (PuTTY, MobaXterm, Ubuntu, ...).



Machine environment - Connection

• You connect to one of the 5 Jean Zay front-end connection nodes:

- These nodes are shared by all the users.
- They are dedicated to creating the computing environment (compilation, data transfers, ...).
- Limitations in execution time and memory are imposed on the scripts running on these nodes in order to avoid problems of overload.
- ▶ These nodes have an HTTP proxy (unlike the computing nodes). You can download data from remote servers (with *git* or *wget*, for example).
- ► They are not equipped with GPUs.
- Bash is the login shell supported at IDRIS. →



Machine environment - Disk spaces

- There are 4 major disk spaces on Jean Zay.
 - \rightarrow Their usages are defined in function of their storage capacities (in GBs and number of inodes or files) and their technical features (temporality, memory access, ...).

Space	Default capacity	Features	Usage
\$HOME	3 GB / 150k <i>inodes</i> per user	 Home directory at connection 	\cdot Storage of configuration files and small files
\$WORK	5 To* / 500k <i>inodes</i> per project	 Storage on rotating disks (350 GB/s in read and 300 GB/s in write) 	 Storage of sources and input/output data Execution in batch or interactive
\$SCRATCH	Very large security quotas 4.6 PB shared by all users	 SSD storage (1.5 TB/s in read and 1.1 TB/s in write) Lifespan of unused files (not read or modified): 30 days 	 Storage of large-sized input/output data Execution in batch or interactive Optimal performance for read/write operations
\$STORE	50 To* / 100k <i>inodes</i> * per project	 Backed up on magnetic tapes, with a cache on rotating disks 	• Long-term archive (lifespan of project)

(*) Quotas of the « project » spaces can be increased per request via the IDRIS extranet. \Rightarrow



Machine environment - Disk spaces

- To consult your disk guotas:
 - User view: \$ idr_quota_user
 - Project view:

\$ idr_quota_project

- By default, your disk spaces are partitioned: Only you have access rights to the files they contain.
- There are three common spaces to share files with members of your project:
 - ► In the \$WORK : \$ALL_CCFRWORK
 - ▶ In the \$SCRATCH : \$ALL_CCFRSCRATCH
 - ► In the \$STORE : \$ALL_CCFRSTORE
- If you use voluminous public data bases, IDRIS can install them for you in the **\$DSDIR** disk space. →
 - This space is accessible in read to all users.
 - It enables sharing resources and not saturating your disk spaces.



Computing environment



Computing environment - JupyterHub

 The IDRIS teams have installed JupyterHub, a tool enabling the usage of Jupyter Notebooks and other applications like VSCode, MLflow and Dask via a web interface, without having to initiate an SSH connection to the supercomputer beforehand. →





Computing environment - Modules

• IDRIS provides a tool catalogue (virtual environments, compiled libraries, ...) accessible via the **module** command.

• Other tools can be added to this catalogue per request (assist@idris.fr).

- **Warning**: to access the modules adapted to the A100 or the H100 partition, you have to load first:
 - ► For the A100 partition:
 - ► For the H100 partition:

module	load	arch/a100
module	load	arch/h100



Computing environment - Basic module commands

•	To display the complete catalogue: module avail
•	To search for a specific tool: module avail <pre>package></pre>
•	To obtain information about a module: module show <package></package>
•	To load a module: module load <package>/<version></version></package>
•	To unload a module:
•	To display the list of loaded modules:

• To restart from a virgin environment:

DEVELOPPEMENT ET DES

SCIENTIFICILE

module purge



Computing environment - Virtual environments

- Artificial Intelligence softwares are installed for Python 3 in **conda virtual environments**.
- Virtual environments are accessible with the **module** command.
 - ightarrow The environments are activated (conda activate) when loading the module.
 - ightarrow They are **not** deactivated (conda deactivate) when the module is unloaded.
- Each environment is based on one of these four major libraries: TensorFlow, PyTorch, MXNet or Caffe.
- To display all the available environments:

module avail tensorflow pytorch mxnet caffe

• Once the environment is activated, you can view all the Python packages contained in it via the pip list and conda list commands.

 $\rightarrow\,$ All the environments can be supplemented per request (assist@idris.fr).



Computing environment - Personal installations

- It is strongly advised to use the environments installed by our support team in order to obtain the best performance, share resources and avoid exhausting your quotas.
- For specific needs, you can make personal installations:
 - ► By enriching an existing environment:

\$ module load <env>
\$ pip install --user --no-cache-dir <paquet>

▶ By creating your own conda environment:

```
$ module load miniforge/24.9.0
$ conda create -n myenv
```

 The advantages and disadvantages of personal installations are explained in the on-line documentation. →



Job submission



Job submission - Slurm wait queue system

- Jobs run on Jean Zay compute nodes.
- To have access to one or more compute nodes, you must submit a request for resource allocations.
- The wait queue to access computing resources is managed by the **Slurm** workload manager for all users.
- A priority system is installed to guarantee the most equitable sharing of resources. →



Job submission - Slurm partitions

- When you reserve computing resources, you must specify to Slurm the **partition** you want (i.e., the type of nodes).
- The CPU partition is selected by default if you do not reserve GPUs.
 This partition is accessible if you have made a request for CPU hours.
- The **V100 quadri-GPU** accelerated partition is selected by default if you reserve GPUs.
 - ▶ This partition contains GPUs of both 16 GBs and 32 GBs of memory.
- Warning: accountings for V100, A100 and H100 GPU hours are distinct.
 To reserve V100 GPU ressources, you must specify --account=xyz@v100
 To reserve A100 GPU ressources, you must specify --account=xyz@a100
 To reserve H100 GPU ressources, you must specify --account=xyz@h100



Job submission - Slurm partitions

• To reserve a specific GPU partition, you must add the corresponding Slurm option during the reservation of resources:

Partition desired	Corresponding Slurm option	
Quadri-GPU V100		
Quadri-GPU V100 $+$ RAM GPU 16 GB	constraint v100-16g	
Quadri-GPU~V100+RAM~GPU~32~GB	constraint v100-32g	
Octo-GPU V100	partition=gpu_p2	
Octo-GPU V100 + RAM CPU 384 GB	partition=gpu_p2s	
Octo-GPU V100 + RAM CPU 768 GB	partition=gpu_p21	
Octo-GPU SXM4 80 GB A100	constraint=a100	
Quadri-GPU SXM5 80 GB H100	constraint=h100	



Job submission - Slurm partitions

Partition desired	Corresponding Slurm option	
Pre/post partition	partition=prepost	
Compil partition	partition=compil	
Archive partition	partition=archive	

- The prepost partition is dedicated to pre-/post-processing jobs.
 - ▶ It contains 4 large memory (3TB) nodes equipped with an NVIDIA V100 GPU.
- The compil partition is dedicated to heavy compilations.
- The prepost partition is dedicated to long data transfers.
- The hours used on the prepost, compil and archive partitions are not subtracted from your allocation.



Job submission - Slurm QoS

• When you submit a job, you must specify a **QoS** (Quality of Service) which calibrates your resource needs (number of GPUs, execution time, ...).

 Each of these partitions offers 2 or 3 QoS. Their limitations are detailed in the online documentation →

The QoS is an important factor in calculating the priority of your jobs.
 The priority will be higher on the -dev QoS and lower on the -t4 QoS.



Job submission - Batch script and interactive connection

• Your jobs can be submitted via batch script or run in interactive mode.

	Batch script	Interactive mode	
When?	During production	During development or debugging	
Why?	To launch large jobs asynchronously	For a rapid succession of short tests	
Advantages	 Submission of multiple jobs in parallel Possible to manage dependencies between the submitted jobs The execution continues if you are disconnected 	 Need to request only one resource allocation Fluid management of executions (interruption, correction, relaunch,) 	
Disadvantages	. Execution management is less fluid (interruption, correction, relaunch,)	. The execution is interrupted if you are disconnected	

- Job submission examples are presented later in this document. More details are available in the on-line documentation regarding:
 - ▶ Batch script submission → (Jean Zay index "Execution/control of a GPU code")
 - Execution in interactive mode ->



- A batch script contains:
 - ► A job configuration heading (name of job, requested resources, ...) in the form of a list of Slurm options preceded by the key word #SBATCH.
 - ▶ The command lines to execute (loading modules, launching the executable file, ...).

• In the submission script, launching the executable file is done by using the **srun** command. This command takes into account the batch configuration.



- Batch script example for an execution on the quadri-GPU V100 partition
 - Execution on one entire node containing 4 GPUs with 16 GB of RAM

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

```
#SBATCH --job-name=TravailGPU
#SBATCH --output=TravailGPU%j.out
#SBATCH --error=TravailGPU%j.err
#SBATCH --constraint=v100-16g
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=10
#SBATCH --time=01:00:00
#SBATCH --time=01:00:00
#SBATCH --hint=nomultithread
#SBATCH --hint=nomultithread
#SBATCH --account=xyz@v100
module purge
```

conda deactivate

module load pytorch-gpu/py3/2.3.0

set -x srun python script.py # name of iob # output file (%j = job ID) # error file (%j = job ID) # reserve GPUs with 16 GB of RAM # reserve 1 node # reserve 4 tasks (or processes) # reserve 4 GPUs # reserve 10 CPUs per task (and associated memory) # maximum allocation time "(HH:MM:SS)" # QoS # deactivate hyperthreading # V100 accounting # purge modules inherited by default # deactivate environments inherited by default # load modules # activate echo of launched commands # execute script



- Batch script example for an execution on the octo-GPU A100 partition
 - Execution on two entire nodes containing 8 GPUs each

#!/bin/bash

```
#SBATCH --job-name=TravailGPU
#SBATCH --output=TravailGPU%j.out
#SBATCH --orror=TravailGPU%j.err
#SBATCH --constraint=a100
#SBATCH --nodes=2
#SBATCH --ntasks=16
#SBATCH --gres=gpu:8
#SBATCH --gres=gpu:8
#SBATCH --cpus-per-task=8
#SBATCH --time=20:00:00
#SBATCH --hint=nomultithread
#SBATCH --account=xyz@a100
```

module purge conda deactivate

```
module load arch/a100
module load pytorch-gpu/py3/2.3.0
```

set -x srun python script.py # name of iob # output file (%j = job ID) # error file (%j = job ID) # reserve 80 GB A100 GPUs # reserve 2 nodes # reserve 16 tasks (or processes) # reserve 8 GPUs per node # reserve 8 CPUs per task (and associated memory) # maximum allocation time "(HH:MM:SS)" # deactivate hyperthreading # A100 accounting # purge modules inherited by default # deactivate environments inherited by default # select modules compiled for A100 # load modules # activate echo of launched commands # execute script



- Batch script example for an execution on the quadri-GPU H100 partition
 - Execution on two entire nodes containing 4 GPUs each

#!/bin/bash

#SBATCH --job-name=TravailGPU #SBATCH --output=TravailGPU%j.out #SBATCH --error=TravailGPU%j.err #SBATCH --constraint=h100 #SBATCH --nodes=2 #SBATCH --ntasks=8 #SBATCH --pres=gpu:4 #SBATCH --cpus-per-task=24 #SBATCH --time=20:00:00 #SBATCH --hint=nomultithread #SBATCH --account=xyz@h100

module purge conda deactivate

module load arch/h100 module load pytorch-gpu/py3/2.4.0

set -x srun python script.py # name of iob # output file (%j = job ID) # error file (%j = job ID) # reserve 80 GB H100 GPUs # reserve 2 nodes # reserve 8 tasks (or processes) # reserve 4 GPUs per node # reserve 24 CPUs per task (and associated memory) # maximum allocation time "(HH:MM:SS)" # deactivate hyperthreading # H100 accounting # purge modules inherited by default # deactivate environments inherited by default # select modules compiled for H100 # load modules # activate echo of launched commands # execute script





 You can connect in SSH to the compute nodes assigned to your jobs in order to monitor the execution of your calculations (top, htop, nvidia-smi,...) :

\$ ssh <node number>



Job submission - Interactive connection

- You can **open a terminal directly on a compute node** on which resources have been reserved for you.
- Example with reservation of one GPU in the default partition:

login@jean-zay3:~\$ srun --ntasks=1 --gres=gpu:1 --<add-options> --pty bash srun: job 123456 queued and waiting for resources srun: job 123456 has been allocated resources login@r13i0n8:~\$

• To disconnect:

```
login@r13i0n8:~$ exit
exit
login@jean-zay3:~$
```

• Important: MPI is not supported in this configuration.



Job submission - Interactive connection

- It is also possible to request resource allocations and **launch a series of** executions on these resources.
- Example with reservation of one GPU in the default partition:

```
login@jean-zay1:~$ salloc --ntasks=1 --gres=gpu:1 --<add-options>
salloc: Pending job allocation 654321
salloc: job 654321 queued and waiting for resources
salloc: job 654321 has been allocated resources
salloc: Granted job allocation 654321
login@jean-zay1:~$ srun script_0.py
...
login@jean-zay1:~$ srun script_1.py
...
```

• To free the resources:

```
login@jean-zay1:~$ exit
exit
login@jean-zay1:~$ salloc: Relinquishing job allocation 654321
```



Consumption of computing hours



Consumption of computing hours

• The number of computing hours h consumed by a job is determined as follows:

h = nb reserved GPUs \times elapsed time

• A node is exclusively reserved when more than one node is reserved, or if the Slurm option --exclusive is activated. In this case, the hours are counted as follows:

h = nb reserved nodes \times nb GPUs per node \times elapsed time

• To monitor the consumption of your projects:

\$ idracct

 For jobs requiring a large CPU RAM memory, the Slurm configuration of the job must be modified accordingly and the way the computing hours are counted is slightly different. →



To go further



To go further - Trainings, workshops and hackathons

- IDRIS provides various trainings destined for scientific HPC and AI computing users. →
 - ► MPI
 - OpenMP
 - MPI/OpenMP hybrid probramming
 - HPC debugging
 - SIMD vectorisation
 - PETSc
 - Introduction to OpenACC and GPU OpenMP
 - Hands-on Introduction to Deep Learning
 - Optimised Deep Learning on Jean Zay
 - LLM specialization
- IDRIS organises workshops about using the supercomputer and optimising your computing codes. →
- IDRIS organises hackathons in partnership with NVIDIA. →



To go further - Contact IDRIS

- For any questions or requests regarding machine access, deploying the software environment, debugging or code optimisation, ...
 - ▶ The IDRIS support team can be reached:

```
Monday through Thursday, 9:00 a.m. - 6:00 p.m.
Friday, 9:00 a.m. - 5:30 p.m.
By e-mail at assist@idris.fr
Or by telephone at +33 (0)1 69 35 85 55
```

 For any questions about the administration of your computing account (password, account opening, access authorisation, declaring IP addresses, ...),
 Contact gestutil@idris.fr.

