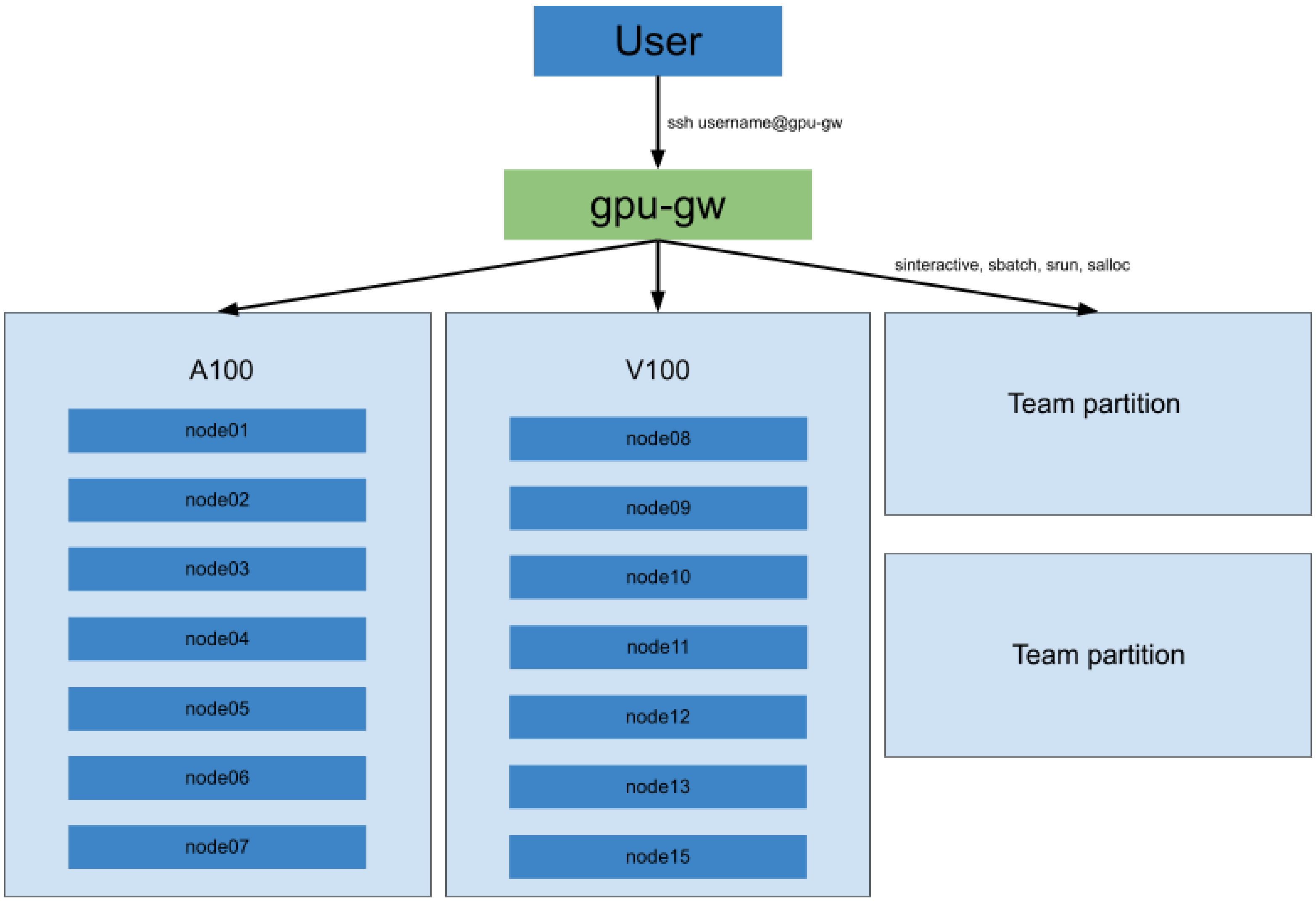
Getting started :

Cluster presentation :

The cluster is managed by a SLURM scheduler. Users cannot directly connect to a calculating node. They have to access through the SLURM scheduler.

https://slurm.schedmd.com



Important note: ***If you have never used the ids cluster (tsi-cluster), please connect once on it before accessing the new cluster, it will create a storage space for you on ids storage server. Your home from the new cluster will be on it.***
https://computingdoc.telecom-paristech.fr/IdsServers

Accessing the cluster:

Connecting to the cluster is done via ssh with your school credentials (you need to pass by ssh.enst.fr if you are on an external network):

```
ssh -Y login@gpu-gw.enst.fr
```

Note: the option -Y activates the confiance connection, which is needed to visualize the images on your screen.

If you work under windows, the connection can be done via an SSH client such as MobaXterm, PuTTY…

With this command, you access the SLURM scheduler. It acts as a gateway to the computation nodes which you can't access directly. You can use the scheduler to create your computing environment (compilation, data transfer, …) but you can't use it to execute code, its resources are limited.

Partitions

The cluster is divided into partitions:

- A100: is composed of 7 nodes:
    - 3 Nvidia Ampère A100 40G GPU
    - 2 AMD EPYC 73022 16 core CPU with 2 threads per core
    - 378G of RAM
- V100: is composed of 4 nodes:
    - 2 Nvidia Tesla V100
    - 2 Intel(R) Xeon(R) Silver 10 cores CPU with 2 threads per core
    - 188G of RAM
- Team partition on witch the owner team has priority (with preemptive right on the jobs running on the partition)

The Partition by default is the V100 one, if you want to specify a different partition, you can add to your SBATH submission script :

```
#SBATCH --partition=<partition name>
```

Or just the `--partition=<partition name>` in your sinteractive or srun command.

To access a specific node, on a team partition for example, you need to specify both, the partition name and the node name:

```
--partition=<partition name>  -w=<node name>
```

Disk spaces

On the SLURM scheduler you have access to the following storage space:

- */home/infres/<login>/   (infres   cluster storage)*
- */home/ids/<login>   (ids cluster storage)*

Those storage can be accessed by the computation nodes.

Your folder /home/ids/<login> is the same as /tsi/clusterhome/<login> on the old cluster. It is the same storage server so all your data from the old cluster can be found on the new one.

//TODO speed of each and rules of storage life.;

The nodes are also equipped with local storage where you can access the following dataset:

- /data/datasets/inet
- /data/datasets/GTA5
- /data/datasets/RAND_CITYSCAPES

You can ask for for new dataset by adding them to this list:

List of datasets

Virtual environments

For the moment, you can install anaconda in your home and create a Conda virtual environment.

https://docs.anaconda.com/anaconda/install/linux/

Note: to download anaconda installer:

```
wget https://repo.anaconda.com/archive/Anaconda3-2022.10-Linux-x86_64.sh
```

You might need to add the anaconda directory to your PATH variables :

```
export PATH="path_to_anaconda/bin:$PATH"
```

Also, in your job script, before activating your environment, for conda to work, you should add this line:

```
eval “$(conda shell.bash hook)”
```

Job submission

The SLURM scheduler gives you access to an interactive mode and a wait queue mode.

Interactive mode

The interactive mode is for development and debugging only, therefore its resources are limited.

This mode gives you access to a console running on a computation node. You can launch a notebook like Jupyter for easy interruption, correction and relaunch of your execution.

If you are disconnected, your execution is interrupted, so using a screen console is recommended.

The command to access the interactive mode is :

```
sinteractive [--OPTION]
```

By default, it give you access to the following configuration:

- execution time : 1 hour
- GPU : 1 Tesla P100 16GB
- CPU : 10 CPUs

You have the following option to change the configuration:

--time (JJ-HH:MM:SS)

--gpus (number of GPU, 20 CPU per GPU)

--partition (A100 | V100 | P100 | mm)

ex: sinteractive --gpus 2 --time 00-10:00:00 --partition V100

To disconnect from the interactive console, you can use the following command (or Ctrl-d):

```
exit
```

Wait queue mode (or batch mode)

Once your code is working, you can submit it to the job queue and access high performance resources.

You can only submit a bash script to the queue.

//add default parameters

This script must contains:

- A job configuration heading (name of job, requested resources, ….) in the form of a list of SLURM options preceded by the word $SBATCH.

- The command lines to execute (loading env, 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

```
#!/bin/bash


#SBATCH --output=TravailGPU%j.out          # fichier de sortie (%j = job ID)
#SBATCH --error=TravailGPU%j.err           # fichier d’erreur (%j = job ID)
#SBATCH --time=00:10:00                    # temps maximal d’allocation "(HH:MM:SS)"
#SBATCH --nodes=1                          # reserver 1 nœud
#SBATCH --gpus:8                           # reserver 8 GPU
#SBATCH --cpus-per-task=3                  # reserver 3 CPU par tache (et memoire associee)


set -x                                     # activer l’echo des commandes
srun python -u script.py                   # executer son script
```

You can find the list of all the configuration options available here:

https://slurm.schedmd.com/sbatch.html

To submit this script to the queue, you have to use the following command:

```
sbatch <scriptname>
```

To follow the state of your job, you can use the following command:

```
squeue -u $USER
```

To print all the submission parameter of a submitted job, you can use:

```
scontrol show job <jobid>
```

To cancel a job, you can use:

```
scancel <jobid>
```

Using a debugger

You can connect IDE to the cluster to use the integrated debugger directly on the nodes.

For this, you need to ask for an interactive console on one node. This will allow you to connect to this node via ssh. You can’t access a node if you don’t have a job running on it.

For Pycharm, here is the official documentation:

https://www.jetbrains.com/help/pycharm/configuring-remote-interpreters-via-ssh.html#ssh

For VSCode, here is the official documentation:

https://devblogs.microsoft.com/python/remote-python-development-in-visual-studio-code/#remote-ssh-workspaces

Tips:

You can create a ssh tunnel between one of your local ports and the node that you want to work on. This will avoid changing your IDE configuration each time you are assigned to a different node.

For example, if your interactive console is on node01, you can type on your local computer:

```
ssh -f <USERNAME>@node01 -L 10000:localhost:22 -N
```

This command will create a ssh tunnel between your local port 10000 and the ssh port (22) on node01.

After this, you need to use localhost and port 10000 as the remote host in your IDE configuration.

From now on, every time you arrive on a different node, you just need to execute the ssh binding command with the correct node name. Your IDE will automatically find the distant node.

Visualisation

Tensorboard

First, you need to connect to an interactive shell. You can use the sinteractive command and specify with the --time=(JJ-HH:MM:SS) option, the duration of your work.

You need to activate a conda environment and install Tensorflow on it:

```
(base) user@node03:~$ conda create –name tensorboardenv
```

```
(base) user@node03:~$ conda activate tensorboardenv
```

```
(tensorboardenv) user@node03:~$ conda install -c conda-forge tensorflow
```

Note: I've been assigned to the node03 in this example, it may vary in your case, note with one you are working on, you will need to know later on.

You can now launch your tensorboard server using the command (you need to adapt the path of the logdir):

```
(tensorboardenv) user@node03:~$ tensorboard --logdir=PATH_TO_logdir --port=8889
```

This command will start a tensorboard server on the node.

You are now ready to connect to tensorboard via ssh.

You need to create a ssh tunnel between the port 8889 on your personal computer and the port 8889 on the node. In a new console, on your personal computer, you can use this command:

```
ssh -t user@node03 -L 8889:localhost:8889
```

You need to replace "user" by your user name and replace node03 by the name of the node your notebook is running on.

you can now open your browser and access to the jupyter notebook via the url:

http://localhost:8889/

Note: Please kill your job on the cluster using the scancel JOBID command to let others access the resource. Leaving your job running will have a negative impact on your priority.

Note: you can also execute Tensorboard in a sbatch script to monitor your training.

Weights & biases

First, you need to install wandb in one of your conda env:

```
(wandb) user@node03:~$ conda install -c conda-forge wandb
```

Next, you need to log to your wandb account using:

```
wandb login
```

You can now use wandb as you would normally do on your computer.

Jupyter notebook

First, you need to connect to an interactive shell. You can use the sinteractive command and specify with the --time=(JJ-HH:MM:SS) option, the duration of your work.

You need to activate a conda environment and install jupyter on it:

```
(base) user@node03:~$ conda create –name jypterenv
```

```
(base) user@node03:~$ conda activate jupyterenv
```

```
(jupyter) user@node03:~$ conda install -c anaconda jupyter
```

Note: I've been assigned to the node03 in this example, it may vary in your case, note with one you are working on, you will need to know later on.

You can now launch a jupyter server with the following command line:

```
(jupyter) user@node03:~$ jupyter notebook --no-browser --port=8888
```

This command will start a jupyter notebook on the node.

Note: A token is generated at the creation of the server. It is displayed now, you will need it later to connect to the server on your local browser.

You are now ready to connect to the jupyter notebook via ssh.

You need to create a ssh tunnel between the port 8888 on your personal computer and the port 8888 on the node. In a new console, on your personal computer, you can use this command:

```
ssh -t user@node03 -L 8888:localhost:8888
```

You need to replace "user" by your user name and replace node03 by the name of the node your notebook is running on.

you can now open your browser and access to the jupyter notebook via the url:

http://localhost:8888

You need to enter the token generated at the creation of the jupyter server.

Note: Please kill your job on the cluster using the scancel JOBID command to let others access the resource. Leaving your job running will have a negative impact on your priority.

Note: A more in-depth tutorial: https://alexanderlabwhoi.github.io/post/2019-03-08_jpn-slurm/

Singularity

Singularity is installed on the cluster, you can find its documentation here: singularity documentation.

Singularity provides an image converter to be able to use Docker images.
The --fakeroot option is not available on the cluster.

Changing Cuda version

There are 3 different version of Cuda available on every node:

- Cuda 10.0
- Cuda 11.6
- Cuda 12.0

By default, cuda 12.0 is used.
To change the version used, you can add to your .bashrc (adapt this to the cuda version you want):

```
export PATH=$PATH:/usr/local/cuda-10.0/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-10.0/lib64
```

Installed package

Image visualization

feh is installed on the cluster, you can use it like so:

```
feh <IMAGE_NAME>
```

Or just feh in a folder of images.

Note: you should have used the -Y option in your ssh command to connect to the cluster.