Boada cluster: abbreviated guide

Contingut

Description of the Boada teaching cluster

Boada's cluster is composed of several nodes. Currently there's only one with GPU capabilities:

- Node boada-10
    - 2 Intel(R) Xeon(R) Silver 4314 at 2.40 GHz
        - 16 cores per processor
        - 32 threads per processor
    - 128 GB of RAM
    - GIGABYTE MG62-G41-00 motherboard

    - 4 NVIDIA GeForce RTX 3080 GPU accelerators
    - 1 NVIDIA GeForce RTX 4090 GPU accelerator

- All nodes have a local disk, where all users have access to `/scratch/1/{username}` (replace `{username}` with your username)
- Additionally, all nodes can see the NAS disk at `/scratch/nas/1/{username}`

□ To manage the various jobs that can be executed, the **slurm** queue system is used:

- The queue system is responsible for running the jobs submitted by users based on the cluster's usage and the jobs requested by each user.
- In this way, if one user launches 1000 jobs and another user subsequently launches a couple of jobs, the latter will not have to wait for the 1000 jobs launched by the first user to finish.
- An order is established based on the usage requested by each user and the available resources.

Logging in

To use this cluster, you need to establish an SSH connection and activate X11 forwarding (you must replace `{username}` with your username):

```
ssh -X {username}@boada.ac.upc.edu
```

⚠ **Operating system requirements:**

- Unix/Linux operating systems usually already have the SSH client.
- On Windows operating systems, you need to install Cygwin or similar (during installation, you must select the openssh and openssl packages).
- On Mac operating systems, you need to install XQuartz.

User groups (accounts)

Each user belongs to at least one account.

Account	Observacions
default	All users belong to at least this account
execution2	Only authorized users belong to this account
cuda	Only authorized users belong to this account

cudabig	Only authorized users belong to this account

Coordinators

We can define a user to act as a coordinator of an *account*, which will allow the user, for example, to cancel other users' jobs.

QOS (Quality of Service)

Time and/or resource limits have been defined for Slurm jobs.

QOS	Limits	Observations
cuda3080	max time: 15 minutes, min gpu: 1, max gpu: 1	
cuda4090	max time: 15 minutes, min gpu: 1, max gpu: 1	
cudabig3080	max time: 8 hours, min gpu: 1, max gpu: 1	
cudabig4090	max time: 8 hours, min gpu: 1, max gpu: 1	

Partitions

The cluster nodes are distributed in different partitions. The different accounts can access the partitions assigned to them:

Partition	**Account**	**Nodes**	**Default QOS**	**Generic Resources (GRES)**	**Observations**
`cuda`	`cuda`	boada-[10]	`cuda3080`	gpu [gpu:rtx4090:1] [gpu:rtx3080:4]	Partition restricted to authorised users
`cuda`	`cudabig`	boada-[10]	`cudabig3080`	gpu [gpu:rtx4090:1] [gpu:rtx3080:4]	Partition restricted to authorised users

Executing jobs

Jobs must be sent to the cluster's execution queues (in the case of Boada, equivalent to partitions):

Queue	Description	Observations
cuda3080	A special queue that uses the RTX3080 GPUs of the *boada-10* node	This queue has restricted access
cuda4090	A special queue that uses the RTX4090 GPUs of the *boada-10* node	This queue has restricted access
cudabig3080	A special queue that uses the RTX3080 GPUs of the *boada-10* node	This queue has restricted access
cudabig4090	A special queue that uses the RTX4090 GPUs of the *boada-10* node	This queue has restricted access

- A job is nothing more than a script that tells the cluster what actions to perform. At the beginning of the job, you put the directives (which start with #SBATCH) followed by the commands you want to execute. For instance:

```
#!/bin/bash
#SBATCH --chdir=/scratch/nas/1/{username}
hostname
```

⚠ Replace `{username}` with your username.

Jobs can be executed in two ways:

- Interactive: to execute immediately, intended for testing before queuing the job. Since we are already in an interactive session, we can execute the script directly:

```
./run-xxx.sh
```

- Queue system: to queue the job for execution when resources become available. We will use the sbatch command, specifying the name of the file to execute:

```
sbatch [-p partition] ./submit-xxx.sh
```

□ If the partition is not specified, it will be the `execution` queue

Executing commands in another partition

Interactively

To open an interactive session on a node of a special partition, we will do:

```
salloc -A {account} -p {partition} srun --pty /bin/bash
```

where `{account}` corresponds to the account name (e.g., **cuda9**) and `{partition}` corresponds to the partition name (e.g., **cuda9**).

Batch

To send a job to a queue other than execution, we will have to define the job file:

```
#SBATCH -A {account}
#SBATCH -p {partition}
....
```

where `{account}` corresponds to the account name (e.g., **cuda9**) and `{partition}` corresponds to the partition name (e.g., **cuda9**).

Execute interactive commands with X11

If we want to access interactively and be able to execute commands that use X11, we will do:

```
/usr/local/bin/srun.x11 -A {account} -p {partition} [additional
params]
```

For example, to access the cuda9 partition, we will do:

```
/usr/local/bin/srun.x11 -A cuda9 -p cuda9 --gres=gpu:1
```

Redirecting standard and error output

By default, the standard output and standard error go to the home directory we have defined with the names slurm-$ID.out and slurm-$ID.err, where $ID is the job number.

If we want these files to go to a specific file, we can use the --output and --error options of the sbatch command, or put it as directives:

```
#!/bin/bash

#SBATCH --chdir=/scratch/nas/1/{username}
```

```
#SBATCH --output=/scratch/nas/1/{username}/output-%j.out
#SBATCH --error=/scratch/nas/1/{username}/error-%j.out

hostname
```

⚠ Replace `{username}` with your username.

 In the names, you can put directives to indicate slurm parameters, in the example %j will indicate the JobID. Other directives:

- %%: The % character
- %a: Job array ID
- %j: JobID
- %N: Hostname where the script is executed

 You can see more directives in the filename pattern section of the sbatch command manual.

Using multiple nodes

We can indicate that our job must be executed on multiple nodes with the **--nodes** parameter:

```
sbatch --nodes=4 testmulti.sh
```

 The more nodes we request, the lower the priority of our job will be and the longer it will take to start executing.

When a job requesting multiple nodes is executed, the execution starts on one of the nodes assigned to the job, and we have the rest of the nodes reserved. To access the different nodes from the initial node, the **srun** command is used.
Let's see an example:

```
#!/bin/bash

#SBATCH --chdir=/scratch/nas/1/{username}
#SBATCH --job-name="testMultiNode"
#SBATCH --wait-all-nodes=1

DIR=/scratch/nas/1/{username}

JOBMASTER="$DIR/master.sh"
```

```
JOBSLAVE="$DIR/slave.sh"

for node in `scontrol show hostnames $SLURM_JOB_NODELIST`; do
        if [ "$HOSTNAME" = "$node" ] ; then
                $JOBMASTER &
        else
                srun --nodes=1 --nodelist $node --ntasks=1
$JOBSLAVE &
        fi
done

wait
```

⚠ Replace `{username}` with your username.

At the beginning of the example, there is a set of directives:

- The **chdir** directive indicates the path where the script should be executed.
- The **job-name** directive indicates the name of the job. It helps us to identify it.
- The **wait-all-nodes** directive indicates that the job will not start until all nodes are available.

Next, a set of variables is prepared (DIR, JOBMASTER, JOBSLAVE).
Then, a for loop is made based on the value of the **SLURM_JOB_NODELIST** variable. Whenever we request a job with multiple nodes (parameter --nodes), the nodes assigned to us can be consulted in the job through this variable.

> □ Since the format in which the variable is indicated is not a list of nodes, the command **scontrol show hostnames $SLURM_JOB_NODELIST** is used, which returns a list of the indicated nodes separated by spaces.

In the example, inside the loop, we check if it is the initial node (which we call master) or one of the other nodes (slaves).

- If it is the master, we execute the process that is considered to be executed on this node (we do it in batch by putting the & symbol, otherwise the script will not continue until this process finishes).
- If it is one of the slaves, using the **srun** command we execute the process on the indicated node. Let's look at the parameters:
    - --nodes=1 indicates that we want to execute the $JOBSLAVE process on one of the nodes.
    - --nodelist $node indicates the node on which the execution is to be done (remember that we have these nodes reserved).
    - --ntasks=1 indicates that only one task should be executed on the node.
    - $JOBSLAVE is the script to be executed on the slave node.
    - Again, it is important to see that we execute it in batch using the **&** symbol.

When the loop finishes, we do a wait to wait for all processes to finish.

GPU jobs

Example source code

hello_cuda.cu

```
#include <stdio.h>

__global__ void helloFromGPU (void)
{
  printf("Hello World from GPU!\n");
}

int main(void)
{
  // hello from cpu
  printf("Hello World from CPU!\n");
  helloFromGPU <<<1, 5>>>();
  cudaDeviceReset();
  return 0;
}
```

Building

```
nvcc -ccbin gcc-4.8 -o hello_cuda hello_cuda.cu
```

Interactive execution

```
boada-1$ salloc -A {account} -p {partition} --gres=gpu:1 srun
--pty /bin/bash
salloc: Granted job allocation 1137
boada-9$ echo $CUDA_VISIBLE_DEVICES
1
boada-9$ ./hello_cuda
Hello World from CPU!
Hello World from GPU!
Hello World from GPU!
Hello World from GPU!
Hello World from GPU!
Hello World from GPU!
boada-9$ exit
```

⚠ where {account} corresponds to the account name (e.g., cuda9) and {partition} corresponds to the partition name (e.g., cuda9).

 With the --gres parameter we indicate the GPU resources we want to reserve with the syntax --gres=gpu:[type]:quantity. In this example, we have reserved one (1) GPU without specifying what type. We can see the different types of GPU in the partitions table.

 The CUDA_VISIBLE_DEVICES variable tells us the ID of the GPUs assigned to us.

Batch execution

hello_cuda.sbatch

```
#SBATCH --chdir=/scratch/nas/1/{username}
#SBATCH --output=/scratch/nas/1/{username}/sortida-%j.out
#SBATCH --error=/scratch/nas/1/{username}/error-%j.out
#SBATCH --job-name="cuda"
#SBATCH -A cuda9
#SBATCH -p cuda9
#SBATCH --gres=gpu:1
echo "CUDA_VISIBLE_DEVICES=$CUDA_VISIBLE_DEVICES"
./hello_cuda
```

```
$ sbatch hello_cuda.sbatch
Submitted batch job 1196
```

We will get the output in the file */scratch/nas/1/{username}/sortida-1196.out*:

```
$ cat /scratch/nas/1/$USER/sortida-1196.out
CUDA_VISIBLE_DEVICES=1
Hello World from CPU!
Hello World from GPU!
Hello World from GPU!
Hello World from GPU!
Hello World from GPU!
Hello World from GPU!
```

Slurm commands

To see the list of jobs waiting to be executed:

```
squeue -t PENDING
```

To see the list of running jobs:

```
squeue -t RUNNING
```

To see the detailed status of a running job:

```
scontrol show jobid -dd {jobid}
```

To list only our own jobs:

```
squeue -u $USER
```

To cancel a job:

```
scancel {jobid}
```

To see the general status of the cluster:

```
sinfo
```

To see the status of a specific node:

```
scontrol show node boada-2
```

To see the priority of the jobs:

```
sprio -l
```

Interactive sessions and node limits

To ensure that no one makes indiscriminate use of the nodes, all are limited to a maximum of 5 CPU minutes per session if accessed via ssh. This should be enough time for editing, small compilations, and consulting files.

Working with GPUs

If you have authorization to work with GPUs, we can enter the corresponding node indicating the number of available GPUs:

```
salloc -A cuda -p cuda --qos=cuda3080 --gres=gpu:rtx3080:2 srun
--pty /bin/bash
```

□ With the parameter --gres=gpu:rtx3080:2 we indicate the number of GPUs we want available for our job within the limits allowed by the queue where we execute.

Since the nodes can have various types of GPUs, you can indicate the type of GPU you want by specifying it in the same parameter:

```
salloc -A cuda -p cuda --qos=cuda4090 --gres=gpu:rtx4090:1 srun
--pty /bin/bash
```

Or if we want several GPUs of a specific model:

```
salloc -A cuda -p cuda --qos=cuda3080 --gres=gpu:rtx3080:2 srun
--pty /bin/bash
```

If you want to access the long-duration queues, you must specify the specific account:

```
salloc -A cudabig -p cuda --qos=cudabig3080 --gres=gpu:rtx3080:4
srun --pty /bin/bash
```

Copying files between Boada and a personal computer

Files can be copied from a personal computer using the scp command, indicating the username:

• Copy a file to Boada:

```
scp file user@boada.ac.upc.edu:
```

• Copy a file from Boada:

```
scp user@boada.ac.upc.edu:file .
```

□ If we are using a very modern version of SSH on the personal computer, it is necessary to add the -O option because the current version of SCP on Boada does not support the internal SFTP protocol by default:

```
scp -O file user@boada.ac.upc.edu
```

Changing password

To change the password, you must execute exactly:

```
ssh -t boada.ac.upc.edu passwd
```

⚠ The word "passwd" is part of the command, not the password.

⚠ The -t option is necessary because the entry server is not one of the cluster nodes.

Additional software

Intel oneAPI

⚠ It is important to remember to connect to Boada with X11 forwarding enabled to have the graphical environment of the suite available.

1. Initialize the environment:

```
% source /Soft/intel_oneapi/2023_02/setvars.sh
```

2. Now we have all the utilities available. For instance:

```
% icc -V
Intel(R) C Intel(R) 64 Compiler Classic for applications running
on Intel(R) 64, Version 2021.10.0 Build 20230609_000000
Copyright (C) 1985-2023 Intel Corporation. All rights reserved.
% ifort -V
Intel(R) Fortran Intel(R) 64 Compiler Classic for applications
running on Intel(R) 64, Version 2021.10.0 Build 20230609_000000
Copyright (C) 1985-2023 Intel Corporation. All rights reserved.
```

Control of sessions and executed jobs

There is a mechanism to view connection and job data in the course account queues.

Since the IP address is personal data, the IP data of the accounts cannot be indicated if students have not been previously notified that this transfer and processing will be done. For this reason, the IP data has been "hidden" with a hash value, so that the same IP will give the same hash value. In addition, to facilitate the interpretation of the data, it has been indicated for each hash value which type of IP it belongs to.
Therefore we will have:

```
INFO: Encrypted IP address typology:
"(IP1) Public UPC IP (e.g. computer labs) [147.83.xxx.xxx]"
"(IP2) Private UPC IP (e.g. eduroam, NAT networks, etc.)
[10.xxx.xxx.xxx]"
"(IP3) External UPC IP (can also be behind an external NAT, e.g.
libraries, etc.)"
```

The data report that can be consulted for an account will include data for the entire current semester. For the fall semester, we will have data from September to January of the following year. And for the spring semester, we will have data from February to August. Therefore, on September 1st and February 1st is when this data is cleared. This data will be sorted by session start date in ascending order.

The execution of the query script will be as follows:

```
boada-1:~$ ./LOG-{course}/showUsage.sh
usage: ./showUsage.sh [[-u usuari ] [-i ] [-j] | [-h]]
```

where `{course}` corresponds to the course acronyms.

E.g.

```
boada-1:~/LOG-par$ ./showUsage.sh -u par4206

IMPORTANT: les dades son actualitzades cada 5 minuts! Les
connexions inferiors a 5 minuts respecte l'hora actual poden no
apareixer encara!

Llistat de sessions de l'usuari par4206 des de Feb-2021 fins
Feb-2021:

par4206  pts/9   (IP1)f0c6c40d2f39dded0eec0cdc2e894c81  -
Wed Feb 17 17:18:16 2021 - Wed Feb 17 17:20:20 2021 (00:02)
par4206  pts/13  (IP1)7a54ad6cb280f0184c0a15bc66a8ffe0  -
Wed Feb 17 17:19:14 2021 - Wed Feb 17 17:19:14 2021 (00:00)
par4206  pts/9   (IP1)f0c6c40d2f39dded0eec0cdc2e894c81  -
Wed Feb 17 17:20:32 2021 - Wed Feb 17 17:21:29 2021 (00:00)
par4206  pts/9   (IP1)f0c6c40d2f39dded0eec0cdc2e894c81  -
Wed Feb 17 17:21:48 2021 - Wed Feb 17 17:21:48 2021 (00:00)
par4206  pts/10  (IP1)f0c6c40d2f39dded0eec0cdc2e894c81  -
Wed Feb 17 17:22:04 2021 - Wed Feb 17 17:22:04 2021 (00:00)
par4206  pts/10  (IP1)f0c6c40d2f39dded0eec0cdc2e894c81  -
Wed Feb 17 17:23:41 2021 - Wed Feb 17 17:23:52 2021 (00:00)
par4206  pts/10  (IP1)f0c6c40d2f39dded0eec0cdc2e894c81  -
Wed Feb 17 17:24:08 2021 - Wed Feb 17 17:24:27 2021 (00:00)
par4206  pts/10  (IP1)f0c6c40d2f39dded0eec0cdc2e894c81  -
Wed Feb 17 17:24:35 2021 - Wed Feb 17 18:54:59 2021 (01:30)

INFO: Encrypted IP address typology:
"(IP1) Public UPC IP (e.g. computer labs) [147.83.xxx.xxx]"
"(IP2) Private UPC IP (e.g. eduroam, NAT networks, etc.)
[10.xxx.xxx.xxx]"
"(IP3) External UPC IP (can also be behind an external NAT, e.g.
libraries, etc.)"
```

```
Detall treballs executats a nodes de calcul (no es compta
boada-1):
       JobID      State        NodeList               Start
End      User ExitCode    CPUTime
60950         COMPLETED        boada-2 2021-02-17T18:14:04
2021-02-17T18:14:08   par4206      0:0   00:01:36
60963         COMPLETED        boada-2 2021-02-17T18:19:58
2021-02-17T18:19:59   par4206      0:0   00:00:24
60993         COMPLETED        boada-3 2021-02-17T18:30:53
2021-02-17T18:30:57   par4206      0:0   00:01:36
60995         COMPLETED        boada-3 2021-02-17T18:30:57
2021-02-17T18:31:02   par4206      0:0   00:02:00
60996         COMPLETED        boada-3 2021-02-17T18:31:02
2021-02-17T18:31:06   par4206      0:0   00:01:36
60997         COMPLETED        boada-3 2021-02-17T18:31:06
2021-02-17T18:31:10   par4206      0:0   00:01:36
61015         COMPLETED        boada-2 2021-02-17T18:35:15
2021-02-17T18:35:19   par4206      0:0   00:01:36
61024         COMPLETED        boada-3 2021-02-17T18:40:31
2021-02-17T18:40:33   par4206      0:0   00:00:48
61026         COMPLETED        boada-2 2021-02-17T18:41:25
2021-02-17T18:41:26   par4206      0:0   00:00:24
61029         COMPLETED        boada-2 2021-02-17T18:41:38
2021-02-17T18:41:39   par4206      0:0   00:00:24
```

Here we will see the session start and end data. Regarding the IP, we will see a hash value (in the example f0c6c40d2f39dded0eec0cdc2e894c81 and 7a54ad6cb280f0184c0a15bc66a8ffe0) which, if they match, means that the connection IP address was the same for each corresponding session. In the example, we also see that they are identified with the type IP1, therefore corresponding to public UPC IPs. We will also see the jobs executed in the queues (boada-1 is not taken into account, as it is the node where the interactive job is executed).

If you want to summarize only the IP data (we will use the -i parameter):

```
boada-1:~/LOG-par$ ./showUsage.sh -u par4206 -i

IMPORTANT: les dades son actualitzades cada 5 minuts! Les
connexions inferiors a 5 minuts respecte l'hora actual poden no
apareixer encara!

Llistat de IP encriptades des d'on s'ha connectat l'usuari
par4206 des de Feb-2021 fins Feb-2021:

(IP1)7a54ad6cb280f0184c0a15bc66a8ffe0
```

```
(IP1)f0c6c40d2f39dded0eec0cdc2e894c81
(IP1)f0c6c40d2f39dded0eec0cdc2e894c81
```

where we will only see the IP data filtered for duplicates.

We can also display a summary of the executed jobs:

```
boada-1:~/LOG-par$ ./showUsage.sh -u par4206 -j

...

Resum nombre de treballs executats a nodes de calcul (no es
compta boada-1): 10 treballs completats      0 treballs en altres
estats
```

□ We can also use the *-i* and *-j* parameters at the same time to see the connections and jobs in a summarized way.

⚠ **It is important to keep in mind that connection IP addresses can be behind external and internal NATs, so that the same public IP may be serving different private IPs, and therefore this encrypted IP would not identify a single user. We leave the interpretation of the data to your discretion.**

In order to set up the system, we will need to create a directory in your account, where the connection and executed job data for the semester will be saved. It will be important that no one else has access to this directory to avoid showing this data to student accounts. The directory will be /scratch/nas/1/PDI/{user}/LOG-{course}. Additionally, you will find the query script (showUsage.sh) in this directory, which will allow you to filter the data by user.

□ If you are interested, you must notify the LCAC.