HPC Introduction Handout

Accessing remote machines

ssh (secure shell)

ssh <login>@ares.cyfronet.pl	connects to the Ares supercomputer using plgrid account, login starts with "plg-"

tar (Tape archiver)

tar -zvcf TAR.GZ FILES*	compresses the files matching FILES* to archive TAR.GZ
tar -xvf TAR.GZ	unzips the archive TAR.GZ in current directory

scp (secure copy protocol)

scp -Cpv LOGIN@REMOTE.MACHINE:/path/on/the/remote/machine/FILE* /path/on/your/machine/. - copy files matching FILE* on remote to your machine
scp -Cpv /path/on/your/machine/FILE* LOGIN@REMOTE.MACHINE:/path/on/the/remote/machine/. - copy files matching FILE* on your machine to the remote

Environmental variables

export, unset, env, source

Regular variable is only callable from current terminal, while **environmental variable (env var)** propagates to any script run from the current terminal

export NAME=VALUE	sets a new or existing environmental variable NAME to VALUE (string, int or float)
unset NAME	unsets a regular/env variable, so it cannot be called with $NAME
NAME=VALUE	sets a new/existing regular variable NAME to VALUE (string, int or float)
echo $NAME	print the value of a regular/env. var
export PATH=$PATH:/path/to/dir	adds a path to the env. variable called PATH; allows to access any script from it as a command
source script.sh	runs the script as if you used **./script.sh** and ALL (regular and env) variables are added to the environment of your current terminal
env	prints all env vars in your current environment; grep it to look for specific one

useful env vars

$PATH	stores paths, where system looks for executable scripts; a script in $PATH is callable like a command
$HOME	stores your home directory
$SCRATCH	stores your scratch directory
$PLG_GROUPS_STORAGE	stores directory containing all project/plgrid team common directories; find directory of your team(s) using **hpc-fs**, it starts with "plgg-"
$LD_LIBRARY_PATH	stores paths, where system looks for libraries to dynamically link with software you are building; only use if you are about to install software
$USER	stores the current user's name
$PYTHONPATH	stores paths, where python looks for python packages
$MODULEPATH	stores paths, where lmod looks for files describing modules

the ~/.bashrc file

! <u>DO NOT CHANGE the .bashrc file on our clusters, unless you know what you are doing and have read precautions below</u>

Explanation: .bashrc is a hidden file in your home dir that gets sourced at the start of each of your sessions on the cluster, adding commands there will run them at the start of your session, adding variables will add them to your environment at the start of your session.

<u>Precautions:</u>
! Before you add anything to your .bashrc, store a clean copy of the one you had the first time you logged in somewhere safe. For example, use the command **cp ~/.bashrc ~/bashrc-backup**
! Please keep in mind that using this file is very likely to cause problems; people tend to forget about what they've put in the .bashrc or they do not know what effect it might have on their calculation. When you call commands, variables and modules inside .bashrc, they might interfere with commands, variables and modules called from the command line.
! When something breaks and you have modified your .bashrc at any point, before asking Helpdesk for help, try switching your modified .bashrc with the backup you made and see if it fixed the issues you are having. Commands to do it would be:
mv ~/.bashrc ~/bashrc-modified ; cp ~/bashrc-backup ~/.bashrc
And then retry running what did not work. Tell Helpdesk if you did that procedure when you report a problem.

!! In a mild case, you might pollute your environment and break a software so you could not use it until the env is cleared.

!!! In a bad case, <u>putting something that should not be there might break your account</u> and require Cyfronet intervention to fix it; e.g. if you put **exit** in your .bashrc file you won't be able to login to your account back, since you are just going to log out everytime you login, only our admins will be able to fix that

Files system and queueing system

basic Slurm commands

These commands come with Slurm. They are accessible on any slurm-based cluster, even outside plgrid.*
*Some conditions may apply, e.g. squeue might print all jobs of all users instead team-only.

squeue	shows all jobs your team(s) is(are) running currently (including you), this includes its status Pending/Running etc., jobID and time it is running
squeue --me	**squeue**, but only your current jobs; synonym: **squeue --user=$USER**
squeue -A GRANT	**squeue**, but only the specified grant
sbatch script.batch	sends the job specified in the file script.batch to the queue; more about it in the batch scripts section; remember to always use **sbatch** in the dir your script is; you cannot run **sbatch** from a running session
scancel JOBID	cancels the job with the jobID
scancel -u $USER	cancels all your jobs
scancel -n NAME	cancels jobs with the specified name
srun	allows to run interactive sessions and subprocesses
scontrol show jobid JOBID	shows more information about a running job with the jobID

Cyfronet-made commands

These commands were made by Cyfronet staff for your convenience. They are only usable on our clusters.

hpc-[Tab][Tab]	tabbing with "hpc-" will allow you to see all the Cyfronet commands
hpc-fs	shows you how much space was used on HOME, SCRATCH and the common directory of you team, additionally you see how many files you have and what teams you are in without looking on portal plgrid
hpc-grants	shows all current grants you have access to and information about them
hpc-scratch	a command used to switch to our new scratch introduced in 2024 (afscra) https://docs.cyfronet.pl/display/~plgpawlik/New+scratch+space+on+Ares
hpc-jobs	similar to **squeue --me**, but has more performance information
hpc-jobs-history	like **hpc-jobs**, but for completed jobs; newest at the bottom
hpc-grant-jobs GRANT	like **hpc-jobs**, but only for the specified GRANT
hpc-modules ARCH	prints the modules tree of selected architecture; as of 2024 this is a Helios-only command
ssh_slurm jobid node	attaches current terminal to a running session on a given node

Batch scripts (sbatch)

usage of common terms in HPC

job	a set of tasks that the *cluster* will do, it's specified by a script; unlike a script it also specifies resources to be allocated for the *job* and is sent to a queueing system rather than being run as is
sequential	(about software) means that each process is being done one at a time; some algorithms and types of data can only be calculated this way, forcing the software to be partially or fully *sequential*
parallelized	(about software) able to separate the input data into independent parts and combine the results when each has been processed
parallelism	means that multiple processes are being done at the same time; high *parallelism* is the main feature of *supercomputers*; to achieve *parallelism* the software must be *parallelized* too
core (CPU)	a fully functioning processor integrated with other *cores* to maximize performance; regular laptop in 2024 has four *cores*, each can run a different process at the same time (basic *parallelism*)
node	a collection of *cores* physically connected via high speed bus; calculations within one *node* are computed with highest efficiency
cluster	a collection of *nodes* connected via high speed network; it works like multiple computers connected to each other, allowing them to share the workload
supercomputer	a fancy and prestigious name for a *cluster*
task	a unit of computational work that can be run independently, each runs at least 1 *thread*; more in the sbatch options subsection
thread	the smallest unit of computational work that can be run independently; each *core* runs exactly 1 *thread* at a time*
GPU	an accelerator, that works best with highly *parallelized* software (requires to be compiled with *GPU* support)

*It is possible to run 2 *threads*/*cpu* using what's called *hyper-threading*.

job-specific environmental variables

$SLURM_SUBMIT_DIR	stores the path to the directory from which the **sbatch** was run
$SCRATCHDIR	stores the path to a directory on SCRATCH where the calculation should be run; $SCRATCH/slurm_jobdir/<jobid>/tmp.<nodename>
$SLURM_JOB_ID	stores the jobID of the current *job*
$SLURM_NTASKS	stores the number of *tasks* allocated for the *job*

sbatch options

bash vs batch - the difference is that a bash script is run wherever you run it, while batch script is sent to the queueing system using the **sbatch** command.

Writing a batch script requires adding lines specifying **sbatch** options in the script after the shebang and before other commands. Each line contains an option (e.g. -N) followed by its argument (e.g. 1) and preceded by the signifier #SBATCH, e.g. #SBATCH -N 1. One option per line.

Options below can be given in a batch script or to the command **sbatch** itself. The latter overwrites the former. The same options can also be given to **srun** when

-J NAME	sets the name of the *job* (visible using **squeue**); synonym: **--job-name**
-N VALUE	sets the number of *nodes* to allocate; synonym: **--nodes**
-A GRANT-ALLOC	sets which grant's resources to use in the *job* and what allocation to use (e.g. cpu, cpu-bigmem); synonym: **--account**
-p NAME	sets which *queue* to sent to *job* to (e.g. plgrid), synonym: **--partition**
-t D-HH:MM:SS	sets the time limit of the *job*, synonym: **--time**
--ntasks-per-node=VALUE	sets the number of *tasks* per *node* to allocate*; each *task* needs at least 1 *core*, so allocating *n tasks* automatically allocates *n cores* to the job; allocating *n nodes* requires allocating at least *n tasks*, so Slurm will automatically decrease the number of *nodes* to match the number of *tasks;*
--cpus-per-task=VALUE	sets the number of *cores* per *task* to allocate*
-n VALUE	sets the number of total tasks used in the *job*; synonym: **--ntasks**
--mem=VALUE[UNIT]	sets the total memory allocated for the *job*; default unit is MB, a very useful is GB, e.g. **--mem**=180GB
--mem-per-cpu= VALUE[UNIT]	sets the total memory allocated for each *core*; default unit is MB and you cannot use floats only integers, e.g. **--mem-per-cpu**=3750 (this is 3.75GB)
-o FILE	save the terminal outputs of the *job* to a file; default=slurm-<jobid>.out; if the application does not print anything to terminal, the output will be empty; synonym: **--output**
-e FILE	save the terminal error outputs of the *job* to a file; this will usually contain errors, system warnings and module loads; default=slurm-<jobid>.out; synonym: **--error**
--reservation=NAME	uses the given reservation, only if you have one (like on this training)
--gres=gpu:N	uses *N GPUs* for the *job*

*The product of *core*/*task* and *tasks*/*node* cannot be bigger than the maximum number of *cores* on a *node* (Ares: 48, Athena: 128, Helios: 192).

Scientific software management (modules)

Modules allow you to access specific software when you need it without cluttering your environment. Our modules manager is called lmod.

flat vs hierarchical (Ares vs Athena)

Flat module tree (used on Ares) allows you to load any module regardless of what it will do to your environment. You need to be aware of possible inconsistencies and prevent them, otherwise you might run your application with lower efficiency or break it. Additionally Ares changes names to lowercase.

Hierarchical module tree (used on Athena) allows you to load only the modules that are available by the compiler/toolchain modules you have loaded in your environment. This approach prevents inconsistencies and tells you exactly what tools the application was built with. The cost is that you need to load these additional modules first. Changing names to lowercase is impossible with hierarchical module trees.

<u>Helios actually uses two separate hierarchical module trees, each for one architecture (x86 (cpu) and ARM (gpu)).</u>

ml (module; lmod commands)

ml NAME	loads the <u>default version</u> of the module called NAME; synonyms: **module add** NAME, **module load** NAME, **ml load** NAME, **ml add** NAME
ml NAME/VERSION	loads the specific version of the module called NAME
ml purge	unloads all modules; use it to clean up before loading new modules
ml	lists currently loaded modules; synonyms: **ml list**, **module list**
ml avail NAME	prints the list of all versions of the module called NAME; on hierarchical it prints only the ones you can actually load with current modules loaded
ml avail	prints the list of ALL loadable software; on hierarchical it prints only the ones you can actually load with current modules loaded
ml spider NAME	like **ml avail** NAME, but shows the versions you cannot load right now, useful on hierarchical
ml spider NAME/VERSION	prints useful info about the specific version of the module called NAME, <u>including what dependency modules to load it on hierarchical</u>
ml use /path/to/dir	adds /path/to/dir to $MODULEPATH; used when adding custom modules

ml ML-bundle (machine learning kit for python; Helios-only)

ML-bundle is a special module made by Cyfronet staff to aid in building python machine learning applications. It is available only on Helios' ARM architecture (gpu).

Python virtual environment

python3 -m venv

Virtual environment is shortened to just venv.

ml python/3.11.5-gcccore-13.2.0	loads specific python version (on Ares)
ml GCCcore/13.2.0 Python/3.11.5	loads specific python version (on Helios)
python3 -m venv NAME	creates an empty virtual environment called NAME, it will be a directory with the same name containing amongst others bin/activate
source NAME/bin/activate	activates the virtual environment called NAME, the current directory must contain the venv or you would need to add the path to it;
deactivate	deactivates current venv
python3 -m pip --require-virtualenv freeze - prints what packages are in the venv	
python3 -m pip --no-cache-dir --require-virtualenv install NAME - installs the packages called NAME;	
export PYTHONPATH=$PYTHONPATH:/path/to/dir - adds /path/to/dir to the list of paths where python looks for packages	

ML-bundle is a special module made by Cyfronet staff to aid in building python machine learning app

creating and using virtual environment instructions

Here is the instructions:

1. Set up an interactive session
2. Choose python version
3. Create virtual environment
4. Install packages to the venv
5. Use the venv

1) Interactive session

On a login node, run the interactive session with:

```
srun -N 1 --ntasks-per-node=1 -t 1:00:00 -p plgrid-now -A <grantname>-cpu --pty /bin/bash
```

After a while the resources should be allocated. Correct output should look like this:

```
srun: job 12377732 queued and waiting for resources
srun: job 12377732 has been allocated resources
```

The jobID of this job is 12377732.

When the session starts, the prompt will change to e.g.:

```
[ares][plguser@ac0766 ~]$
```

2) Choosing python version

Available versions of python can be view with:

```
ml spider python
```

As long as the default version works, use that one. Only use the older version if you encounter compatibility issues. At the end there is a paragraph about reloading a python version.

Load the python module:

```
ml python/3.12.3-gcccore-13.3.0
```

You will need to load this module everytime BEFORE you use your virtual environment. Remember that a venv created with one python version would not work with another one.

3) Create the virtual environment

If you create the venvs for the first time it is advised to make a directory for them, otherwise skip the mkdir command. Start by making a directory in your team's storage directory:

```
cd $PLG_GROUPS_STORAGE/<groupname>
mkdir -p pythonenvs
cd pythonenvs
```

To create venv called myenv use:

```
python3 -m venv myenv
```

4) Installing packages to virtual environment

This key step should always be performed on a computing node. Therefore, the first step is to open the interactive session. If you have performed steps 2) and 3) without 1), you need to load the selected python module again after starting up the interactive session.

Regardless of that, use the next command to activate the virtual environment as long as your current directory is pythonenvs made before.

```
source myenv/bin/activate
```

The prompt should reflect the activation of the venv:

```
(myenv) [ares][plguser@ac0766 pythonenvs]$
```

You can begin installation of e.g. packages named *requests*:

```
python3 -m pip --no-cache-dir --require-virtualenv install requests
```

Python will automatically install all the dependencies and the package itself. Use the same command to install more packages and use the following command to view what is already installed in the venv:

```
python3 -m pip --require-virtualenv freeze
```

After successful installation, you can leave the interactive session with Ctrl+C and then Ctrl+D. The prompt should reflect annulation of the session:

```
[ares][plguser@login01 pythonenvs]$
```

as you are on the login node now.

5) Using the virtual environment

To use the venv you first load the correct python module and then activate the venv by sourcing either relative or absolute path to myenv/bin/activate.

```
ml python/3.12.3-gcccore-13.3.0
source $PLG_GROUPS_STORAGE/<groupname>/pythonenvs/myenv/bin/activate
```

Adjust to your case the groupname, pythonenvs and myenv to fit what you set up in previous points.

You can actually activate the venv on the login node, but DO NOT run calculations without a compute node.

Bonus) Reloading python version

To ensure that the old module is unloaded, please run:

```
ml purge
```

This command will clean the entire environment, unloading all the modules. Now load a different version:

```
ml python/3.10.8-gcccore-12.2.0
```

<u>Remember that a venv created with one python version would not work with another one.</u>

Modes of work with Slurm

srun (interactive sessions)

srun … --pty /bin/bash -l	only on login node: activates an interactive session, allowing you to do use a compute node from the terminal; in place of the … add any and all **sbatch** options you would normally use in a script; an example is given below:
srun -p plgrid-testing **-N** 1 **--ntasks-per-node**=1 **-n** 1 **-A** <grantname>-cpu **--pty /bin/bash -l**	

pro-viz (graphical user interface)

ml pro-viz	loads the pro-viz module
pro-viz start …	sends an undying session to the queue that will be able to run a graphical user interface; in place of the … add all the options you would add to **sbatch** and replace the ones featured below with their **pro-viz start** counterparts
pro-viz password JOBID	while the job is running: gives instruction how to use the GUI, paste the link into the browser and login with plgrid account
These are **pro-viz start** options that differ from **sbatch**	
-P	replaced **--ntasks-per-node** and **--cpus-per-task** with one option that specifies *cores* per *node*
-r	replaced **--reservation**
-g N	replaced **--gres=gpu:N**
-M	replaced **--mem**

localfs, memfs (temporary filesystems)

These are **sbatch** options	
-C localfs	adds temporary localfs space to your *job*, this filesystem decreases the amount of metadata necessary to manage the files and works better for software producing a lot of smaller files; the size is 1TB
-C memfs	allows you to use RAM as a temporary filesystem in your calculation, the size is the same as the total memory on the *job* and can be specified with **--mem** option; works best for calculations small enough to fit in the memory; not suitable for every calculation
These are environmental variables available only on *jobs* sent with options above, these temporary locations are removed at the end of the *job*, so any files generated on them need to be copied to a permanent storage like SCRATCH from within the batch script	
$LOCALFS	stores a path to the space granted by localfs
$MEMFS	stores a path to the space generated in-memory (RAM) by memfs

Helpdesk guide

BEFORE making a ticket

1. Make sure to not overwrite the outputs, in which the error messages appeared. If you must overwrite the directory, copy them to a safe dir and tell us where it is. Best way to ensure your ticket is going to be resolved as quickly as possible is to not move the script, inputs and outputs at all.
2. If you modified the ~/.bashrc file, undo the changes and try again. It is very easy to break things by adding commands to this file, see the ~/.bashrc section of this handout (p. 2).
3. Check for spelling mistakes in your batch script and input.
4. Read the error message.
   a. DUE TO TIME LIMIT → give the job more time, or more cores, or both and restart;
   b. invalid keyword→ check your input for typos and restart;
   c. command not found, module not found or permission denied → check your script for typos and restart;
   d. OOM (out-of-memory) → give your job more memory and restart;
   e. segmentation violation or other → usually best to make a ticket in Helpdesk.
5. Check the efficiency of the job with **hpc-jobs-history**.
6. If the job finishes correctly, but the problem is the efficiency, try doing a scalability tests for the software:
   a. downsize your problem to a manageable test, e.g. use smaller input, do not run long computation when doing scalability tests;
   b. try running a smaller input on 1, 2, 4, 8, 16, 48 cores and compare efficiency of each test;
   c. if the problem disappears during the tests, but returns in regular calculation, definitely make a ticket in Helpdesk, the issue might be with the software;
   d. when you are going to make the ticket about this issue, include the jobIDs of all your scalability tests.
7. If a severe issue has appeared once and then it did not happen again without changing anything, write a ticket to Helpdesk and call it an incident. Specify that you don't need help, just wanted to inform us - cases like this will greatly help us improve the overall health of our clusters.
   a. What's severe? The calculation failing without an error message, the
8. If you want to install software, we can do it for you - we will add it globally to our repertoire as a convenient module. Make a ticket to Helpdesk and ask,
   a. but if you want us to install a python package, please check if it is contained in the module Scipy-Bundle, like e.g. numpy, scipy;
   b. if it's machine learning related and on Helios, check out the ML-bundle module first;
   c. if not, try installing it to a virtual environment following the instructions given in this handout (p. 6-7).
9. If you want to install custom software on your account locally, tell us - make a ticket to Helpdesk and ask for guidance on how to do it on your own. We will provide the best tricks to make it convenient for you and we will help with arising problems.

WHEN making a ticket

1. In almost all cases, the entire problem can be explained by providing the jobID of the job that failed. ALWAYS provide the jobID to Helpdesk, it will allow us to see everything that happened, including what script you used to run it, where are the inputs and outputs stored and more. You rarely need to paste the script into HD for us, we will just check it on our end.
2. There might not be a jobID, when the problem touches programs running on interactive jobs, on pro-viz, or on the login node. In this case, describe what happened in detail and show the commands you used when the error appeared. First thing we are going to do is try to recreate the problem, so be meticulous in your description.
3. Write a clear title, we get a lot of "Problem with Ares" type tickets that are then revealed to be a specific-software-related issue. When a phrase like e.g. "segmentation violation in gaussian" is in the title, the appropriate expert can take the ticket immediately.
4. When sending a problem with python, use python3 -m pip freeze to generate the list of your packages with their versions. Save it as the requirements.txt file and attach to your ticket.

Best practices

1. If you have two issues, make two tickets.
    a. Unless they are analogous, e.g installing software is; if you need to install three unrelated pieces of software one ticket will suffice.
2. Respect that we work with a variety of work schedules. An expert in your issue might not be immediately available, please be patient.
3. When you have two tickets, try keeping the issues separate, especially when you talk to two different experts.