MPCDF psycl tutorial

Author: Vera N. Karlbauer

Contact: vera_karlbauer@psych.mpg.de

Date created: 19-06-2024

Purpose: Basic tutorial on how to use the MPCDF *psycl*

1. Logging into the cluster

1.1. Accounts you need

Getting an account in Garching/MPCDF (*psycl*):

- Fill out the online form: https://selfservice.mpcdf.mpg.de/index.php?r=registration
- Select Benno Puetz as mentor/supervisor
- Inform Benno per email (puetz@psych.mpg.de) that you've registered as a new user and which team/group you belong to (e.g. Binder Lab)
- Wait until you are activated (you will get an email) and then enable 2-factor authentication in the self-service portal: https://docs.mpcdf.mpg.de/faq/2fa.html

How to get an account at the molgen GitHub (optional, but nice to have):

- please write a ticket to the IT (via ticket@psych.mpg.de) and ask them to request a github/molgen account for your email adress
- once your request has been accepted and forwarded, you will get an email stating that an account for you has been created on a GitHub Enterprise installation. You will need to click on the activation the link in this email within 24 hours
- when you've clicked on the link and activated the account, add your full name to the description of your profile (click on the top right with your icon -> Settings -> Profile); this is important so that the IT finds you!
- reply to your ticket and specify the teams you want to be added to (in your case, this should be MPIP, Scientists, and binderlab)

1.2. Gateway system

- The cluster cannot be accessed directly, but via a **gateway machine**. This provides additional security.
- The gateway machine (gate1 and gate2) is only used to connect to the cluster. You have very limited memory here – DO NOT run any jobs or do anything on the gateway

Standard cluster login:

```
# logging onto the gateway (with agent- and x-forwarding)
ssh -A -X yourusername@gate1.mpcdf.mpg.de
# logging onto psycl
ssh -A -X yourusername@psycl01.bc.rzg.mpg.de
```

1.3. 2FA

- The gateway machines have 2-factor authentication (2FA). This means you need to enter your password AND an OTP (via an app on your phone) when connecting.
- If you only want to do this once per day and not every time you connect to the cluster, follow this guideline: https://docs.mpcdf.mpg.de/faq/tricks.html#how-can-i-avoid-having-to-type-my-password-repeatedly-how-can-i-tunnel-through-the-gateway-machines

2. Cluster structure

2.1. Folders

- /psycl/u hosts individual home directories. Your personal home directory is found under /u/yourusername . This is a good place to host individual projects that nobody else needs access to. All folders in here backed up.
- /psycl/g hosts group directories of different research groups. When you get your cluster account, you will have access to the folder of your group. This is a good place to host common resources that everyone in your group needs. In the long run, we should also install SPLS here. Talk to your research group members about how your folder is organized and what should and should not be put here.
- /ptmp hosts scratch space for any files that do not need to be backed up, e.g. intermediate files during preprocessing. You can put those files into /ptmp/yourusername

2.2. Nodes

- The *psycl* has 14 nodes (psycl01 – psycl14). Those are the parts of the cluster you run your jobs on
- The nodes and their specifications (available RAM, cores, type of CPU) are documented here: https://docs.mpcdf.mpg.de/doc/computing/clusters/systems/Psychiatry.html
- psycl01 is the login node. It is only used for logging in and you should not run any computationally expensive jobs on it. Jobs should be run on psycl02 – psycl14
- We will come back to this in section 4: Running a job

2.3. Pre-installed modules & software

- The MPCDF clusters use environment modules. These modules contain specific software that is pre-installed. To use this software, you need to load the module first.
- General introduction to environment modules: https://docs.mpcdf.mpg.de/doc/computing/software/environment-modules.html

- You can see all available modules on psycl with this command:

```
module avail
```

- Example: to see which MATLAB versions are installed, you can run:

```
module avail matlab
```

- Load a module (e.g. MATLAB R2021B):

```
module load matlab/R2021b
```

2.4. Installing your own software

- Any software or package that is not pre-installed and part of the modules needs to be installed yourself
- The problem: you do not have admin rights on the cluster (for good reason), so you need to install software into your own home directory
- This is also a problem if you install packages within a software (for example in R), because it will attempt to install into the software home directory, but you do not have the permissions to do this
- Tip: create a directory in your home directory where you install all of your software

```
mkdir /u/yourusername/mylibs
```

- Either permanently add directories to your library path by modifying the .bashrc file in your home directory (/u/yourusername)

```
# go to home directory
cd /u/yourusername
# edit the bashrc file with vim
vi  ~/.bashrc
# switch into edit mode by typing i
# add your directory to the library path
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/u/yourusername/mylibs
# exit edit mode via esc
# exit vim and save changes to file
:wq
```

- OR: specify the location temporarily in any script you use. In that case, the library path will only be changed for your current session.

Add to shell script or type into console:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/u/yourusername/mylibs
```

- You can also tell R/Python/MATLAB etc. where to look for and install packages. You might need to set specific paths or have specific configuration files for this (e.g. .Renviron) – check the respective documentation!

3. Data transfer

3.1. Via the command line

- Whenever we transfer data, we need to specify that we are not directly transferring from our local machine to the cluster, but we are jumping via the gateway
- The most fool-proof way to do this is via *scp*. This is NOT the fastest or most resource-efficient method, and it works best for small to medium data files

Example *scp* command for transferring data from local machine to cluster:

```
scp -r -J yourusername@gate1.mpcdf.mpg.de
/path/to/my/data/on/local/machine  yourusername@psycl01.bc.rzg.mpg.de:/u/yourusername/folder/where/you/want/to/put/your/data
```

Example *scp* command for transferring data from cluster to local machine:

```
scp -r -J yourusername@gate1.mpcdf.mpg.de
yourusername@psycl01.bc.rzg.mpg.de:/u/yourusername/path/to/my/data/on/cluster
/path/where/you/want/to/put/your/data/on/local/machine
```

- A better alternative is *rsync*. However, this will only work if you have properly set up your configuration files, see: https://github.molgen.mpg.de/mpip/IT-Wiki/blob/master/05%20MPCDF%20cluster/mpcdf_fileTransfer.md and https://docs.mpcdf.mpg.de/doc/data/data-transfer/data-transfer.html

3.2. Using a graphical interface

- If you are used to graphical interfaces and moving your files around via drag & drop, you might prefer a client that provides a graphical interface.
- Most clients are might for direct connection from a local machine to a cluster, so you have to setup the gateway machine (this might be a little tricky)
- FileZilla (client for mac): https://filezilla-project.org
- How to use FileZilla with a gateway: https://github.molgen.mpg.de/mpip/IT-Wiki/blob/master/05%20MPCDF%20cluster/mpcdf_fileTransfer.md or https://docs.mpcdf.mpg.de/doc/data/data-transfer/data-transfer.html
- How to use graphical interface clients on windows: https://docs.mpcdf.mpg.de/faq/tricks.html#how-can-i-access-the-clusters-using-a-windows-machine (WinSCP)

4. Running a job

4.1. SLURM

- Jobs on *psycl* are run and scheduled via the Slurm Workload Manager

- The most important Slurm commands:
  - *srun*: run an interactive shell
  - *sbatch*: run a .sh script on the cluster (you will mostly use this)
  - *squeue*: check which jobs are currently queued and running (use this to check on your jobs)
  - *scancel*: cancel a running or queued job
  - *sinfo*: info about psycl available nodes and their availability
  - For more commands and appropriate arguments, check the slurm handbook: https://slurm.schedmd.com
- More information about slurm here: https://github.molgen.mpg.de/mpip/IT-Wiki/blob/master/04%20MPIP%20cluster/slurm_introduction.md (this is about the local MPIP cluster, but the bits about slurm also apply for psycl) and here: https://docs.mpcdf.mpg.de/doc/computing/cobra-user-guide#slurm-batch-system (this mostly applies to psycl as well)

4.2. Shell scripts

- Non-interactive jobs on the cluster a usually run via shell scripts
- There, you specify how much memory you need, after how much time the job should be terminated, etc.
- Then, you specify what you actually want to do, e.g. run an R script, copy a file, or use any other tool available on the cluster

Example shell script for running an R script:

```
#! /usr/bin/bash
#SBATCH --job-name=myjob     # Name of the job showing up in the queue
#SBATCH --ntasks=1          # Nr of CPUs to run job on, e.g. 1 CPU
#SBATCH --mem=10g         # Job memory request, e.g. 10g
#SBATCH --time=1:00:00     # Time limit hrs:min:sec, this lets it run for 1h
#SBATCH --output=myjob_%j.log    # Standard output and error log
pwd; hostname; date

module load R/4.1.2         # load R module
Rscript myscript.R          # run R script
```

- Shell scripts are run using *sbatch*

4.3. Interactive sessions

- You can also run interactive sessions on the cluster (same as having the program open on your local machine)

- You can start many programs by loading the module and typing their name:

```
module load matlab/R2021b
matlab
```

- However, this will be on the login node and you will have no memory specifications
- If you want to make use of the cluster's RAM and work with large files, you can start the interactive session as a job via slurm
- Remember to terminate your interactive sessions by properly quitting the software or using scancel!

Example for starting an interactive R session with 10 GB of RAM:

```
module load R/4.1.2
srun --mem=10G R --no-save       # Run R without saving your workspace at end of
session
```

Example for starting an interactive MATLAB session with 10 GB of RAM:

```
module load matlab/R2021b
srun --mem=10G matlab
```

- In case you want a graphical interface for your sessions (e.g. in matlab), I recommend logging into the cluster with X-forwarding activated (ssh -X in your login command) and then starting the session on the login node:

```
module load matlab/R2021b
matlab
```

- I have not yet figured out how to make this work for the execution nodes, although this should theoretically be possible.

5. Easy cluster workflow with git

- If you have scripts to run on the cluster, you probably want to edit the scripts locally, try to run them on the cluster, change them locally if there's any errors, and go back to the cluster to run the new updated script. This can mean a lot of back-and-forth copying between the cluster and your local machine, which gets confusing really quickly
- To make sure that your local scripts and your scripts on the cluster are always up-to-data with each other, it's nice and easy to use GitHub. This also means you can track and document any changes you made and why. (Also, in general, it's a really good idea to have your analysis code on GitHub. This also makes it much easier to publish your code later on, or to share it with collaborators).
- I recommend having your analysis folder as a github repository, and then using git to synchronize any changes you either make locally or on the cluster. This means you need to get into the habit of first pulling from github before you do anything to your

analysis folder (locally and on the cluster), and committing your changes when you're done.

- **Disclaimer:** do this for scripts, but NOT FOR DATA! (not a good idea because of file size and data security)
- Generally, your data and your output should NOT be in the same folder as your scripts. The best way to move data to and from the cluster is explained in section 3: data transfer.
- Want a nice way to set up the folder structure of your project correctly? Check this out: https://github.com/jonas-hag/analysistemplates
- (This is an R package for setting up your project folders in a standardized way, and was developed by the MPI CodeClub team.)
- (Even if you do not use the R package, you can look at the folder structure and set it up the same way.)

Example git workflow when working on the cluster

- Step 1: create a github repository:
- Step 2: write/edit your scripts locally
- Step 3: Commit your scripts and push to github
- Step 4: Hop onto the cluster
- Step 5: Pull from github (if this is your first time, you need to clone the repository first)
- Step 6: Run the script
- Step 7: Commit and push any changes to github (if you made any, e.g. added logfiles)
- Step 8: Hop onto local machine
- Step 9: Pull changes from github
- Step 10: Rinse and repeat

6. More resources

- Psycl -specific documentation: https://docs.mpcdf.mpg.de/doc/computing/clusters/systems/Psychiatry.html
- General documentation for MPCDF clusters (basically all of this applies to psycl): https://docs.mpcdf.mpg.de
- MPCDF helpdesk (this is NOT run by our local IT, and has a good response time):
- IT-Wiki of the MPI of Psychiatry, managed by the CodeClub (you need a molgen github account to access this, see section 1.1): https://github.molgen.mpg.de/mpip/IT-Wiki

This has lots of useful tips and tricks on the cluster structure and on how to best run jobs with slurm