Jak odpalać joby na klastrze w Cyfronecie?

1. Logowanie komendą **ssh <twoj login plgrid>@<nazwa klastra>.cyfronet.pl**
2. (Opcjonalne) Zabezpieczenie sesji **tmux**em (komenda **tmux**) lub **Screen**em. Dzięki temu będziemy mogli połączyć się do sesji ponownie, jeśli z dowolnego powodu nastąpi zerwanie połączenia (komenda **tmux a**)
3. Przejście do Scratcha, czyli katalogu przypisanego do twojego profilu plgrid (komenda **cd $SCRATCH**)
4. (Gdy wgrywamy dane) Przesłanie danych z lokalnej maszyny (np. za pomocą **windows scp** lub po prostu **scp** w Linuxie) lub pobranie ich z githuba (zamiast hasła podajemy github API key)
5. Uruchamiamy sesję na superkomputerze za pomocą komendy **srun -A plgwtdydoptym-cpu -p plgrid-gpu-v100 --gres=gpu:1 -C memfs -N 1 -n 4 --mem=64GB -t 24:00:00 --ntasks-per-node=4 --pty /bin/bash -l**

   dostępne są dwa tryby **srun** (interaktywny) i **sbatch** (batchowy). Argumenty można dostosowywać do swoich potrzeb, ale konieczność większych zasobów może powodować dłuższe oczekiwanie na ich przyznanie. Kolejne argumenty to:

   - **-A** – nazwa grantu, określana przez opiekuna (tutaj już jest wstawiona nazwa z grantu, który dostaliśmy od dr'a Turka),
   - **-p** – nazwa kolejki, dostęp do kart V100 na Aresie wymaga podania kolejki jw., (ta i kolejna z opcji wydaje się w naszym przypadku niepotrzebna)
   - **--gres=gpu:1** – zarezerwowanie dostępu do specjalnych zasobów, w tym wypadku jednej karty GPU,
   - **-N** – liczba węzłów, na których uruchamiane jest zadanie,
   - **-n** – liczba rdzeni (dla całego zadania), które zostaną udostępnione,
   - **--ntasks-per-node** – liczba rdzeni na pojedynczym węźle (w zasadzie może być pominięte),
   - **--mem** – ilość pamięci RAM dla każdego węzła obliczeniowego,
   - **-t** – maksymalny czas uruchomienia zadania (24h są maksymalną wartością dla tej kolejki),
   - **--pty** – powłoka, która zostanie uruchomiona po połączeniu się z węzłem.

6. Po rozpoczęciu sesji ładujemy potrzebne nam moduły. W naszym przypadku są to:

- **module load GCCcore/11.2.0**
- **module load OpenSSL/1.1**
- **module load Python/3.9.6** (prawdopodbnie dostępne są nowsze wersje, ja takiej używałem na Athenie)
- **module load epanet/2.2.0-gcc-13.2.0**

Raczej nic innego nie powinno być nam potrzebne, w razie czego opis jak sprawdzić listę dostępnych modułów i inne pomocne informacje są tutaj.

7. Następnie robimy to samo dla “zmiennych globalny” (?), u nas nie powinno być to konieczne, ale poniżej przykłady jak to zrobić, tak “w razie w”:

    - export WANDB_START_METHOD=fork
    - export TRANSFORMERS_CACHE="$SCRATCH/.cache"
    - export WANDB__SERVICE_WAIT=300

8. Następnie tworzymy (**virtualnenv [options -p = path]**) lub aktywujemy istniejące → **source <path to virtualenv>/bin/activate**

    - u mnie to było = **source /net/tscratch/people/<nazwa_w_plgrid>/<env-name>/bin/activate**

    I ważne jest, żeby to wirtualne środowisko było w twoim scratchu, czyli część **/net/tscratch/people/<plgrid_name>** jest obowiązkowa w pathie przy tworzeniu.

9. Jeśli nie zrobiliśmy tego wcześniej za pomocą **pip** pobieramy potrzebne bibilioteki, w naszym przypadku powinny wystarczyć **deap** i **scipy**. Trzeba oczywiście zadbać o odpowiednie ich wersje.

10. Potem możemy przemieszczać się po naszych folderach w scratchu w typowy linuxowy sposób i wykonywać dowolne komendy linuxowe, lub odpalać skrypty zapisane w Scratchu tj. parsing danych, trening, predykcja itd. W naszym przypadku może to być np. **python3 genetic.py**

    Trzeba pamiętać o pobraniu naszego repo (pkt 4) i przejścia do odpowiedniego katalogu.