Liste des logiciels disponibles sur Centaure

- La plupart des logiciels sont installés en tant que modules activables en tapant `module avail` dans le terminal centaure. Certaines choses sont installées “de base”.
- Certaines personnes ont besoin de versions spécifiques de codes. Elles doivent être faites dans le home, afin de ne pas polluer l'espace commun. Un exemple est donné ci dessous pour Gpaw. Si un code peut être utile à suffisamment d'utilisateurs, vous pouvez envoyer un mail à `centaure-admins@listes.emse.fr` pour l'installer en tant que module.

Python

Référent: `aurelien.villani@emse.fr`

La distribution Python du cluster est Anaconda depuis juin 2020. La doc se trouve ici, et le principe de la distribution va être récapitulée ci dessous

Les avantages par rapport à une distribution système de base:

- plusieurs versions parallèles facilement gérables
- pas besoin d'installer pythyon et ses modules sur tous les noeuds, tout est dans /export/apps
- on peut personnaliser l'installation sur son home pour un coût d'espace disque modique.

Un module va etre dispobile disponible: `anaconda2022/python` qui remplacera les `anaconda3/XXX` Au chargement, le module indique qu'il faudra ajouter une fonction dans votre shellrc. Pour bash, dans `~/.bashrc` :

```
XXX
```

Pour zsh, dans `~/.zshrc`:

```
XXX
```

Ensuite, vous pouvez afficher les environnement disponibles avec `conda env list`, et activer celui qui vous intéresse avec `conda activate celui_que_je_veux`

Env perso

Il se peut que vous ayez besoin d'une version spécifique, ou d'un paquet qui n'est pas installé sur le cluster.

Vous pouvez installer un *environnement* qui sera dans votre home, et lié à l'installation système:

```
module load anaconda/python
conda create --name monpythonamoi python=3.6 numpy tensorflow ETC
```

Pour activer cet environnement:

```
conda activate monpythonamoi
```

et pour revenir au défaut:

```
conda deactivate
```

Vous pouvez installer une version spécifique d'un package aussi:

```
conda install scipy=0.15.0
```

Jupyter lab

Référent: `aurelien.villani@emse.fr` New!

On peut avoir un JupyterLab fonctionnel, qui tourne proprement sur un noeud du cluster avec une réservation.

Avant toute chose ! Python est séquentiel, sauf si vous lui dites autrement, et sauf librairies très spécifiques. Autrement dit, dans 99% des cas, ne réservez qu'un cœur pour votre notebook (ou renseignez vous sur les librairies que vous utilisez)

```
module load tools/cluster-bin
cluster-create-slurm-script-01.sh -jupyter-lab
```

Çà utilise le module anaconda/python3.

Note: si vous avez un alias pour le cluster dans ~/.ssh/config, remplacez l'adresse de centaure (193.49.173.181) par votre alias !

Faites vos résas comme d'habitude et soumettez. Si le job démarre alors que vous êtes connecté sur centaure, un message pop pour vous dire de regarder le fichier slurm-XXX.out. Procédez ensuite comme suit (en anglais parce voilà, comme dans le job):

You now have to connect to your notebook. Open slurm-XXX.out

1. Open a terminal from your desktop (NOT centaure) and create a tunnel to the compute where you notebook has been assigned, with the command below. It will have the correct values in the slurm-XXX.out. If you want the tunnel to stay in the background, remove use 'ssh -f ...' Otherwise, don't forget to kill the tunnel once done.

```
   ssh -L \${port}:localhost:\${port} YOUR_cluster_login@193.49.173.181
ssh -L \${port}:localhost:\${port} -N \${computenode}
```

2. Now look for the connection link to paste in your browser in the bottom of the file. It looks like `http://localhost:YYYY/?token=XXXXX`
3. Happy notebook use !
4. **VERY IMPORTANT**: when you are finished and have saved your notebook, you **must**, in jupyterlab, click on the menu: `File→Shut Down` . This way, the slurm job will finish properly, and your data will be copied back on your home folder (as usual in a subfolder 'youname-$SLURM_JOB_ID'. If you do a scancel, you will have to copy your data from the compute scratch yourself and risk data loss if not done in the next days.

Notes:

- Attention à la durée de réservation, dans un sens comme dans l'autre. Si vous demandez une heure, au bout d'une heure, vous êtes éjecté, comme un job normal !
- Si vous voulez lancer avec un autre module python, modifiez le fichier job. Après `module load anaconda/python3`, ajoutez une ligne `conda activate mon_env`
- **Non recommandé, sauf si vous savez ce que vous faites.** Vous pouvez aussi lancer une résa en interactif et tout faire en manuel, par exemple
  - 'srun -p intensive.q –time=00:30:00 –ntasks-per-node 1 –pty bash'
  - ça vous mets sur, par exemple, compute-0-5
  - 'mkdir /scratch/villani_monjupyter && cd scratch/villani_monjupyter'
  - `module load anaconda/python3` (ou 2, ou autre, puis charger un de vos environnements locaux avec `conda activate mon_env`)
  - trouvez un port libre: `comm -23 <(seq 7777 8888 | sort) <(ss -Htan | awk '{print $4}' | cut -d':' -f2 | sort -u)| shuf | head -n 1` ça vous sors par exemple 8146
  - `jupyter-lab –no-browser –port=8146`
  - faites le tunnel `ssh -L 8146:localhost:8146 YOUR_cluster_login@193.49.173.181 ssh -L 8146:localhost:8146 -N compute-0-5`
  - connectez vous depuis votre navigateur `http://127.0.0.1:8146/?token=XXXX`
  - `File→Shut Down` quand vous avez finis
  - rapatriez vos données du scratch avec un scp

MTEX

Utiliser MTEX sur le cluster

Référent: `julien.favre@emse.fr`

Mtex est un package utilisable avec Matlab pour traiter les cartes EBSD. La documentation est disponible sur : https://mtex-toolbox.github.io/

Le traitement pouvant être lourd, cela devient intéressant d'utiliser le cluster pour ça. Par contre vous devez installer Mtex sur votre home, et ça demande quelques manips.

Plusieurs étapes pour l'installation:

1) Télécharger et décompresser Mtex Télecharger avec la commande (pensez à télécharger la dernière version bien sur, le lien peut changer):

```
wget
https://github.com/mtex-toolbox/mtex/releases/download/mtex-5.9.0/mtex-5.9.0
.zip
```

Puis décompresser l'archive avec

```
unzip mtex-5.9.0.zip
```

Ces deux commandes vont créer un dossier mtex-5.9.0 sur votre home (bien sur le nom des dossiers et des archives va varier avec la version que vous aurez téléchargé).

2) Dans votre dossier home, vous allez devoir installer et compiler en local une librairie. Suivez le guide... Importer le module GCC9 avec la commande module load gcc/9.3.0 Importer matlab aussi, sinon ça va planter: module load matlab/R2020b

Télécharger la librairie NFFT avec (pensez à télécharger la dernière version bien sur, le lien peut changer):

```
wget https://www-user.tu-chemnitz.de/~potts/nfft/download/nfft-3.5.3.tar.gz
```

Puis décompresser l'archive avec:

```
tar -xf nfft-3.5.3.tar.gz
```

On navigue dans le dossier décompressé:

```
cd nfft-3.5.3/
```

Puis exécuter les commandes suivantes pour faire la compilation. Faites attention au chemin d'installation de Matlab qui peut différer selon la version installée:

```
./bootstrap.sh
./configure --with-matlab=/export/apps/MATLAB/R2020b --enable-nfsoft --
enable-nfsft --enable-openmp --enable-portable-binary
make
```

Normalement la compilation doit se dérouler sans problème. Sinon, c'est dommage pour vous.

Maintenant copiez certains fichiers obtenus de la compilation vers le répertoire de mtex que vous avez décompressé:

```
cp ~/nfft-3.5.3/matlab/nfsoft/nfsoftmex.mex* ~/mtex-5.9.0/extern/nfft_openMP
cp ~/nfft-3.5.3/matlab/nfsft/nfsftmex.mex* ~/mtex-5.9.0/extern/nfft_openMP
cp ~/nfft-3.5.3/matlab/nfft/nfftmex.mex* ~/mtex-5.9.0/extern/nfft_openMP
```

3) Installation à proprement dit de Mtex. Naviguez dans le répertoire d'installation de Mtex avec la

commande cd ~/mtex-5.9.0/ Vérifiez bien que vous avez importé GCC9 et Matlab avec “module load gcc/9.3.0” et “module load matlab/R2020b” Lancer la commande “matlab -nodisplay -nojvm -nodesktop -nosplash -r install_mtex” (c'est mieux de lancer ça d'un noeud de calcul ou bien du noeud compute-build) Normalement Mtex va s'installer…

4) Vous pensez avoir fini… ben non… En fait quand on lance Matlab, Mtex ne se lance pas spontanément. Donc pour lancer un job avec Mtex il faut quelques astuces. Déja, commencez par générer un fichier job pour votre calcul avec :

```
module load tools/cluster-bin
cluster-create-slurm-script-01.sh -matlab
```

Pensez bien à modifier la version importée de matlab en mettant la commande module load matlab/R2020b Pour lancer un job matlab il faut bien utiliser la commande “matlab -nodisplay -nojvm -nodesktop -nosplash -r run_mtex” avec “run_mtex” le nom de votre fichier de script (sans marquer l'extension “.m”), vérifiez que c'est bien cette commande que vous avez dans le fichier job. Gardez le nom de script “run_mtex”, car on va l'utiliser juste après…

Une fois que vous avez fait ça, dans votre script il faut d'abord lancer Mtex, puis après seulement lancer votre script de post-traitement. Je vous propose donc de faire un script matlab de lancement; faites un fichier “run_mtex.m” contenant :

```
cd ~/mtex-5.9.0
startup_mtex
cd ~/mtex
monscriptmtexici
```

avec “monscriptmtexici” à remplacer par votre nom de fichier de script Mtex (sans marquer l'extension “.m”). On suppose aussi ici que votre répertoire de travail est /export/home/tartempion/mtex. Donc pensez bien à ajuster ce nom de répertoire dans le script de lancement.

Pour résumer, quand vous lancez un traitement depuis un dossier de travail, mettez le fichier job dedans, celui-ci va lancer run_mtex.m, qui démarre Mtex et lance votre script. A priori ça devrait marcher. Sinon, n'hésitez pas à compléter ce tuto…

Compilations perso

Gpaw

Referent: `julien.favre@emse.fr`

Calcul DFT https://wiki.fysik.dtu.dk/gpaw/index.html

Gpaw est un peu galère à installer avec les support de FFTW et mpi.

Récupérer le soft avec git (vérifier dernière version stable sur la doc https://wiki.fysik.dtu.dk/gpaw/index.html):

```
git clone -b 20.1.0 https://gitlab.com/gpaw/gpaw.git
cd gpaw
cp siteconfig_example.py ~/.gpaw/siteconfig.py
```

Modifiez le fichier 'siteconfig.py' aux alentours de la ligne 47 avec le bloc suivant:

```
fftw = True
if fftw:
    libraries += ['fftw3']

# ScaLAPACK (version 2.0.1+ required):
scalapack = True
if scalapack:
    libraries += ['mkl_avx2', 'mkl_intel_lp64' ,'mkl_sequential'
,'mkl_core',
            'mkl_scalapack_lp64', 'mkl_blacs_intelmpi_lp64',
            'pthread'
            ]
    library_dirs +=
['/export/apps/intel/parallel_studio_xe_2018/mkl/lib/intel64'
            ]
```

Puis compilez dans votre home:

```
module load intel/parallel_studio_xe_2018
python3 setup.py install --user
```

Un fichier job de test est donné en example:

```
#!/bin/bash
#SBATCH --job-name=test_gpaw
##SBATCH --mail-type=ALL
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4
#SBATCH --time=500:00:00
#SBATCH --partition=intensive.q

ulimit -l unlimited

unset SLURM_GTIDS

echo ------------------------------------------------------
echo SLURM_NNODES: $SLURM_NNODES
echo SLURM_JOB_NODELIST: $SLURM_JOB_NODELIST
```

```
echo SLURM_SUBMIT_DIR: $SLURM_SUBMIT_DIR
echo SLURM_SUBMIT_HOST: $SLURM_SUBMIT_HOST
echo SLURM_JOB_ID: $SLURM_JOB_ID
echo SLURM_JOB_NAME: $SLURM_JOB_NAME
echo SLURM_JOB_PARTITION: $SLURM_JOB_PARTITION
echo SLURM_NTASKS: $SLURM_NTASKS
echo SLURM_TASKS_PER_NODE: $SLURM_TASKS_PER_NODE
echo SLURM_NTASKS_PER_NODE: $SLURM_NTASKS_PER_NODE

echo ----------------------------------------------------
echo Generating hostname list...
COMPUTEHOSTLIST=$( scontrol show hostnames $SLURM_JOB_NODELIST |paste -d, -s
)
echo ----------------------------------------------------

echo Creating SCRATCH directories on nodes $SLURM_JOB_NODELIST...
SCRATCH=/scratch/$USER-$SLURM_JOB_ID
srun -n$SLURM_NNODES mkdir -m 770 -p $SCRATCH  || exit $?
echo ----------------------------------------------------
echo Transferring files from frontend to compute nodes $SLURM_JOB_NODELIST
srun -n$SLURM_NNODES cp -rf $SLURM_SUBMIT_DIR/* $SCRATCH  || exit $?
echo ----------------------------------------------------

echo Running gpaw...
module purge
module load intel/parallel_studio_xe_2018

cd $SCRATCH

gpaw info
mpiexec -np $SLURM_NTASKS_PER_NODE -mca btl openib,self,vader -host
$COMPUTEHOSTLIST python3 -c "import gpaw.mpi as mpi; print(mpi.rank)"

# tests si besoin
# mpiexec -np $SLURM_NTASKS_PER_NODE -mca btl openib,self,vader -host
$COMPUTEHOSTLIST python3 -m gpaw test



echo ----------------------------------------------------

echo Transferring result files from compute nodes to frontend
srun -n$SLURM_NNODES cp -rf $SCRATCH  $SLURM_SUBMIT_DIR 2> /dev/null
echo ----------------------------------------------------
echo Deleting scratch...
srun -n$SLURM_NNODES rm -rf $SCRATCH
echo ----------------------------------------------------
```