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.

Par défaut, deux modules sont dispos: anaconda/python2 et anaconda/python3, qui seront toujours les dernières versions, régulièrement mises à jour, de python 2 et 3. Dans les deux cas, il suffit d'appeler `python` dans le fichier job, une fois le module chargé, et la bonne version sera lancée.

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

Pour un python spécifique, par exemple python 2.4, vous pouvez installer un *environnement* qui sera dans votre home, et lié à l'installation système:

module load anaconda/python2
conda create --name monpythonamoi python=2.4

Vous aurez alors une installation minimale **sans les modules python du système**. Pour activer cet environnement:

conda activate monpythonamoi

et pour revenir au défaut:

conda deactivate

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

DOC - https://portail.emse.fr/dokuwiki/

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.

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

 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} cluster ssh -L
\${port}:localhost:\${port} -N \${computenode}
```

- 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

- $\circ\,$ 'srun -p intensive.q -time=00:30:00 -ntasks-per-node 1 -pty bash'
- $^\circ\,$ 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 cluster ssh -L 8146:localhost:8146 -N compute-0-5`
- connectez vous depuis votre navigateur `http://127.0.0.1:8146/?token=XXXX`
- \circ `File→Shut Down` quand vous avez finis
- rapatriez vos données du scratch avec un scp

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:

```
Last update:
23/11/2021 09:17 recherche:cluster:softwares https://portail.emse.fr/dokuwiki/doku.php?id=recherche:cluster:softwares&rev=1637655479
```

['/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 ------

From: https://portail.emse.fr/dokuwiki/ - DOC

Permanent link: https://portail.emse.fr/dokuwiki/doku.php?id=recherche:cluster:softwares&rev=1637655479

Last update: 23/11/2021 09:17

