Version 0.3

# Utilisation RUCHE

## Ressources

Documentation:  
<https://mesocentre.pages.centralesupelec.fr/user_doc/>

Version PDF : <https://mesocentre.pages.centralesupelec.fr/mesocenter_training/main.pdf>

## Login

ssh [ottf@ruche.mesocentre.universite-paris-saclay.fr (Juin2024!!)](http://ottf@ruche.mesocentre.universite-paris-saclay.fr)

ssh -X [ottf@ruche.mesocentre.universite-paris-saclay.fr](http://ottf@ruche.mesocentre.universite-paris-saclay.fr) (X11 forwarding)  
(advised but not crucial especially if using Visual Studio)

## Ne pas installer Miniforge (ou autre)

module load miniconda3/24.5.0/gcc-13.2.0  
module load openmpi/4.1.5/gcc-13.2.0  
source activate openmc-V15 (ne pas utiliser conda activate)

## Create openmc environment

Commencer par charger un module python

> module load miniconda3/24.5.0/gcc-13.2.0  
(le premier chiffre est la version de conda ; ca installe Python V3.12)

> conda config --add channels conda-forge  
> conda create –n openmc-V15  
> source activate openmc-V15  
> conda install openmc  
> conda list  
> conda env list

> module load miniconda3/24.5.0/gcc-13.2.0  
> module load openmpi/4.1.5/gcc-13.2.0  
(does not seem to be useful to get mpi support)  
**> source activate openmc-V15**

## Useful commands

> ruche-quota  
> module purge  
> squeue  
> seff

## Divers

OpenMC distirbutions : <https://anaconda.org/conda-forge/openmc>  
There are quite a few distributions with various options

To install a specififc version : > conda install /path/to/your/package.conda

Installation Neutron Materials Maker  
> conda install neutronics\_material\_maker -c conda-forge  
Ca semble marcher du premier coup sans installation additionnelle

> python baseline\_A\_V1.py

## Chargement des librairies

mkdir librairies  
wget <https://anl.box.com/shared/static/uhbxlrx7hvxqw27psymfbhi7bx7s6u6a.xz>  
tar -xJf uhbxlrx7hvxqw27psymfbhi7bx7s6u6a.xz

wget <https://anl.box.com/shared/static/nyezmyuofd4eqt6wzd626lqth7wvpprr.xml>  
mv nyezmyuofd4eqt6wzd626lqth7wvpprr.xml chain\_endfb8\_pwr.xml

## Visual Studio

Installer « Remote Explorer » dans VisualStudio (voir instruciton dans le fichier PDF) et se connecter sur la Ruche \*dans\* Visual Studio.   
C’est super pratique, ca permet d’éditer les fichiers.

## Lancer un job

#### Lancement job en interactif

srun --nodes=1 --time=00:10:00 -p cpu\_short python baseline\_A\_V1.py

**Ne lancer des jobs en interactif que sur \*1\* noeud sinon il fait N fois la même chose**

squeue pour voir les jobs dans le pipeline

seff xxx pour voir le résumé du job (ressources – temps)

#### Lancement via slurm

Création de slurm.sh

#!/bin/bash  
#SBATCH --job-name=Baseline\_A\_V1  
#SBATCH --time=00:20:00  
#SBATCH --ntasks=2  
#SBATCH --partition=cpu\_short  
  
module purge  
module load miniconda3/24.5.0/gcc-13.2.0  
module load openmpi/4.1.5/gcc-13.2.0  
source activate openmc-V15  
python baseline\_A\_V1.py

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Lancer avec > sbatch slurm.sh

## Other tools

emacs (bof), **plutôt utiliser Visual Studio pour éditer les fichiers**

#### scp: copying a file to Ruche —> use scp scp: copying a directory from Ruche —> use scp rsync: copying a large directory to Ruche (I)

Or use FileZilla or WinSCP

## Visualization interface

Utiliser NiceDCV

In a web browser, go to <https://ruche.mesocentre.universite-paris-saclay.fr:8443>

Ca permet de facilement visualiser les .png

**Pour utiliser openmc-plotter, il faut charger python et créer les variables d’environnement —> faire bash script <omp.sh>**

module load miniconda3/4.10.3/gcc-13.2.0  
source activate openmc-V15  
export OPENMC\_CROSS\_SECTIONS="/home/ottf/openmc/librairies/endfb-viii.0-hdf5/cross\_sections.xml"  
export OPENMC\_CHAIN\_FILE="/home/ottf/openmc/librairies/chain\_endfb8\_pwr.xml"

Ensuite openmc-plotter se lance sans problème

## Divers

source deactivate

module avail  
module load miniconda3/4.10.3/gcc-13.2.0  
conda create -n openmc-V15  
source activate openmc-V15  
conda install openmc -c conda-forge # OK avec miniconda et anaconda modules

Editer le fichier .py avec VisualStudio + Remote Explorer  
Définir le chemin des librairies dans le code.  
openmc.config['cross\_sections']='/gpfs/users/ottf/openmc/librairies/endfb80/cross\_sections.xml'  
openmc.config['chain\_file']='/gpfs/users/ottf/openmc/librairies/chain\_endfb80\_pwr.xml'

pip install neutronics-material-maker —> NON utiliser conda , ca marche  
pip install coolprop —> non plus nécessaire !

# ESSAI d’UTILISATION de MPI

Voir exemple

<https://gitlab-research.centralesupelec.fr/mesocentre-public/ruche_examples/-/tree/master/hello_mpi4py>

Quand je fais conda list, il n’y a pas mpich

>module load spack/....

spack install py-openmc +mpi ^openmpi ^openmc build\_type**=**Release

# Temps de calculs

#!/bin/bash

#SBATCH --job-name=Baseline\_A\_V1\_test1

#SBATCH --time=00:20:00

#SBATCH --ntasks=1

#SBATCH --cpus-per-task=40

#SBATCH --partition=cpu\_short

# as soon as I try using more than one node, every slows down

# keep on a single NUMA space ? --> mpiexec --bind-to numa --map-by numa ...

# how strictly do the above indications need to match the openmc parameters ?

module purge

module load miniconda3/24.5.0/gcc-13.2.0

module load openmpi/4.1.5/gcc-13.2.0

source activate openmc-V15

# echo of commands

set -x

# To compute in the submission directory

cd ${SLURM\_SUBMIT\_DIR}  # semble important pour qu'il reconnaisse l'environnement openmc-V15

# number of OpenMP threads

# export OMP\_NUM\_THREADS=${SLURM\_CPUS\_PER\_TASK}

# Binding OpenMP Threads of each MPI process on cores

#export OMP\_PLACES=cores

#echo $OMP\_NUM\_THREADS

#echo $OMP\_PLACES

python baseline\_A\_V1.py

On commence avec ntasks=1

Job ID: 8953937

Cluster: ruche

User/Group: ottf/llb

State: COMPLETED (exit code 0)

Nodes: 1

Cores per node: 40

CPU Utilized: 00:43:47

CPU Efficiency: 47.94% of 01:31:20 core-walltime

Job Wall-clock time: 00:02:17

Memory Utilized: 1.27 GB

Memory Efficiency: 0.80% of 160.00 GB

Avec ntasks = 2

Job ID: 8953938

Cluster: ruche

User/Group: ottf/llb

State: COMPLETED (exit code 0)

Nodes: 2

Cores per node: 40

CPU Utilized: 02:11:10

CPU Efficiency: 39.51% of 05:32:00 core-walltime

Job Wall-clock time: 00:04:09

Memory Utilized: 1.27 GB

Memory Efficiency: 0.40% of 320.00 GB

Job ID: 8953974

Cluster: ruche

User/Group: ottf/llb

State: COMPLETED (exit code 0)

Nodes: 4

Cores per node: 40

CPU Utilized: 00:44:54

CPU Efficiency: 13.36% of 05:36:00 core-walltime

Job Wall-clock time: 00:02:06

Memory Utilized: 1.28 GB

Memory Efficiency: 0.20% of 640.00 GB

Pas de corrélation entre le nombre de neouds et le temps de calcul ?!  
—> Il faut forcer l’exécution en mpi ? —> mpiexec ... ?

—> il y a des overheads importants ?