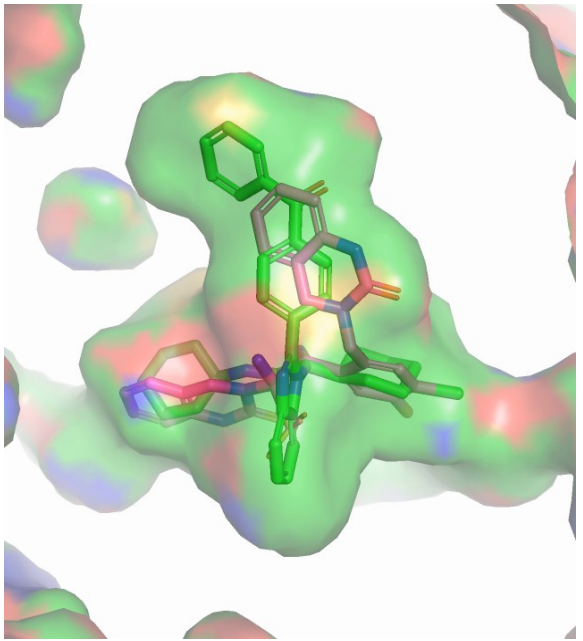
Méthodologie d’utilisation de Leto en python depuis un jupyter notebook

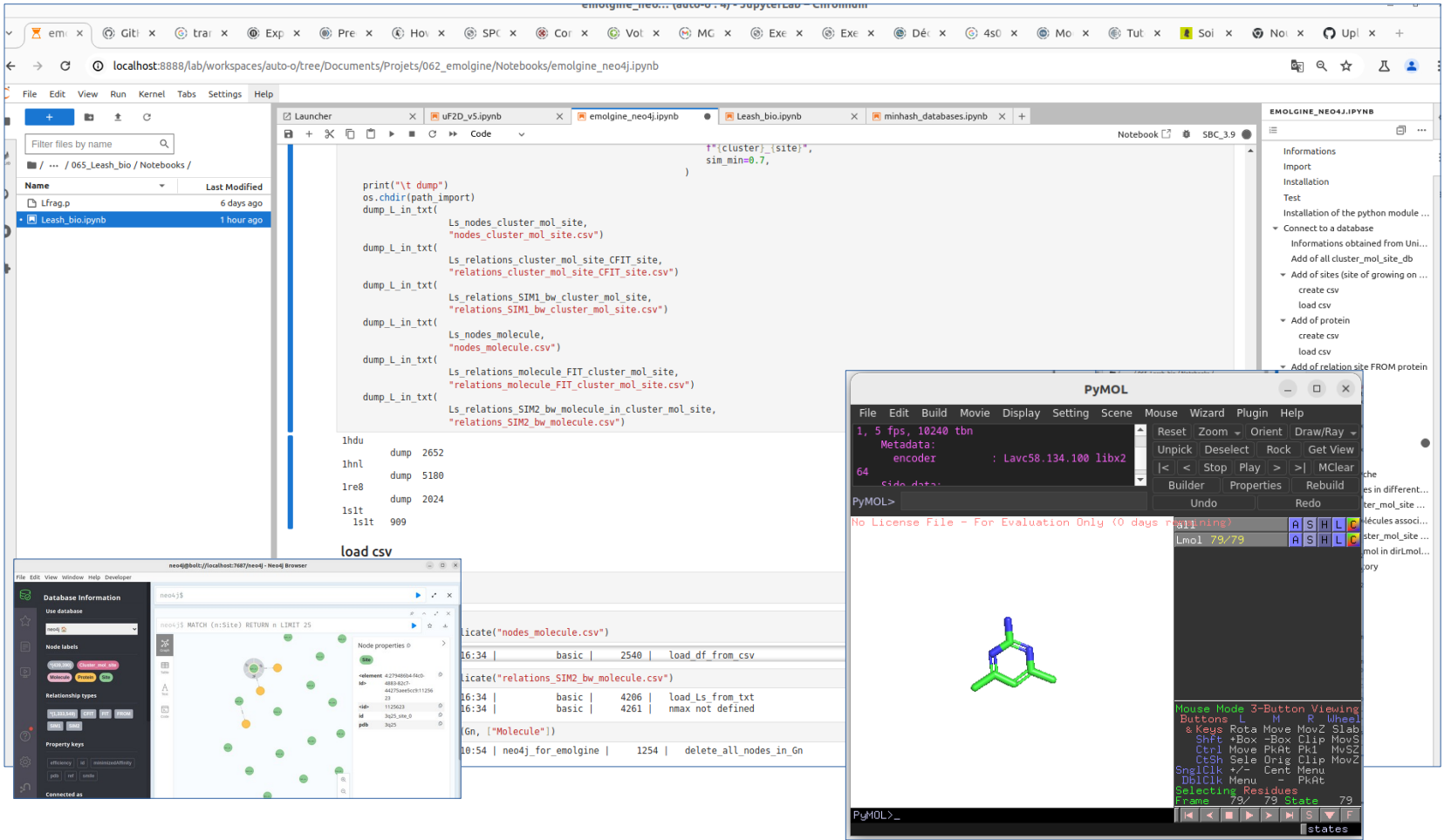
Pascal KREZEL, ICOA

Problématique scientifique à l’ICOA

Générer des molécules compatibles au sein d’un site biologique quelconque à partir de fragments de molécules repositionnés.



Mon environnement de travail



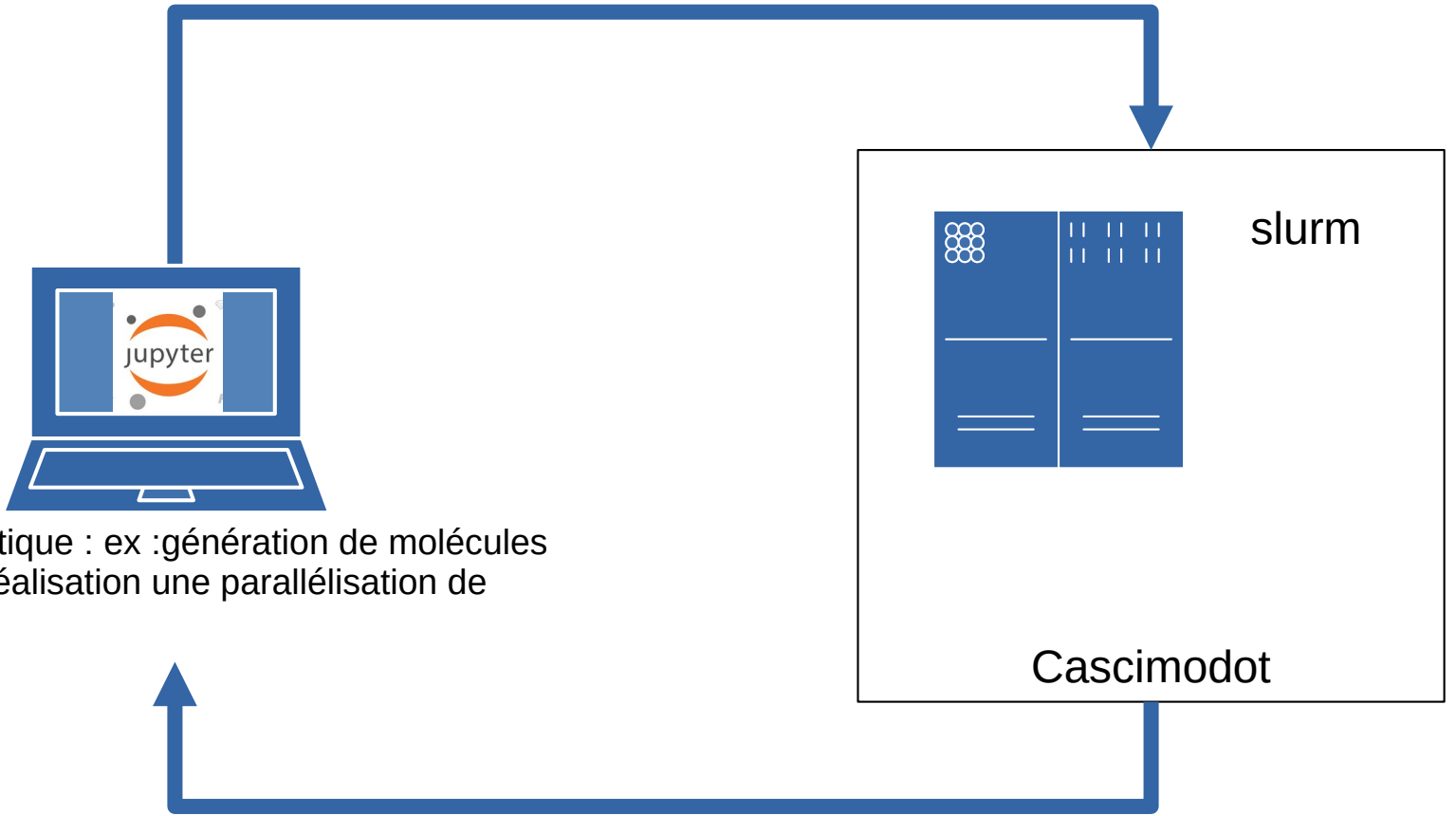
Ubuntu 22.04

+ Python 3.9

+ Jupyterlab

+ neo4j

Problématique de parallélisation sur cluster



Chemoinformatique : ex :génération de molécules
=> besoin de réalisation une parallélisation de données

Analyse des projets existants

De nombreux projets dédiés à ce sujet existent, **mais**
-Ne permettent pas forcément de gérer ma problématique,
-Peuvent être difficiles à mettre en oeuvre.

Finalement, en faisant un effort de programmation, j'ai pu faire ce que je voulais

Cluster Computing

Unlike SMP architectures and especially in contrast to thread-based concurrency, clus
alien to uninitiated developers. In this domain, some overlap with other distributed com

- batchlib - a distributed computation system with automatic selection of proce
- Celery - a distributed task queue based on distributed message passing
- Charm4py - General-purpose parallel/distributed computing framework for th
  of scaling applications to supercomputers. It is based on an efficient actor mode
  actor model. Supports *Linux, Windows, macOS*.
- Dask - Dask is a flexible library for parallel computing in Python. It offers
  - Dynamic task scheduling optimized for computation. This is similar to Airfl
  - "Big Data" collections like parallel arrays, dataframes, and lists that exten
    schedulers.
  - It extends Numpy/Pandas data structures allowing computing on many co
- Deap is a evolutionary algorithm library, which contains a parallelization mod
  compute something else than evolutionary algorithms -- and offers an interface
  communication and load balancing layers. It currently works over MPI, with mpi
  and so on).
- disco - an implementation of map-reduce. Developed by Nokia. Core writ
- dispy - Python module for distributing computations (functions or programs)
  arguments in SIMD style of parallel processing. The nodes can be shared by mu
  and scalability.
- DistributedPython - Very simple Python distributed computing framework, us
  in Python 2.6 and 3.
- exec_proxy - a system for executing arbitrary programs and transferring files
- execnet - asynchronous execution of client-provided code fragments (former
- IPython - the IPython shell supports interactive parallel computing across
- job_stream - An MPI/multiprocessing-based library for easy, distributed pipel
  distributed application. Can be used to realize map/reduce or more complicated
- jug - A task based parallel framework
- mpi4py - MPI-based solution
- NetWorkSpaces appears to be a rebranding and rebinding of Lindaspace

Description d’un exemple simple de calcul sur Cascimodot

Soit une liste de 1 145 molécules sous forme de SMILES :

```
]: Lsmile[:10]
```

```
]: ['NCc1cc(F)cc(F)c1',
    'Nc1nc[nH]n1',
    'O=C(O)C[C@H](Cc1c(F)c(F)c(F)c(F)c1F)NC(=O)OCC1c2ccccc2-c2ccccc21',
    'Nc1ncc([N+](=O)[O-])s1',
    'NCCNC(=O)c1cccnc1',
    'O=C(N[C@H](Cc1ccc(C(F)(F)F)cc1)C(=O)O)OCC1c2ccccc2-c2ccccc21',
    'CC(O)(CN)c1ccsc1',
    'Cc1cc(F)ccc1N',
    'Nc1ccc(-c2ncc[nH]2)cc1',
    'NCc1nc(-c2ccsc2)no1']
```

Je veux obtenir une liste de molécules 3D en appliquant la fonction **get_mol_3D_from_smile** pour cela, on définit la fonction suivante :

```
•[36]: @do_on_cluster(
           f0=get_mol_3D_from_smile,
           Dcluster=Dcluster,
           Lmodule_to_update=["tools_SBC", "F2D", "xScav"],
           )
       def get_Lmol_3D_from_Lsmile_on_cluster(
           Lsmile,
       ):
           pass
```

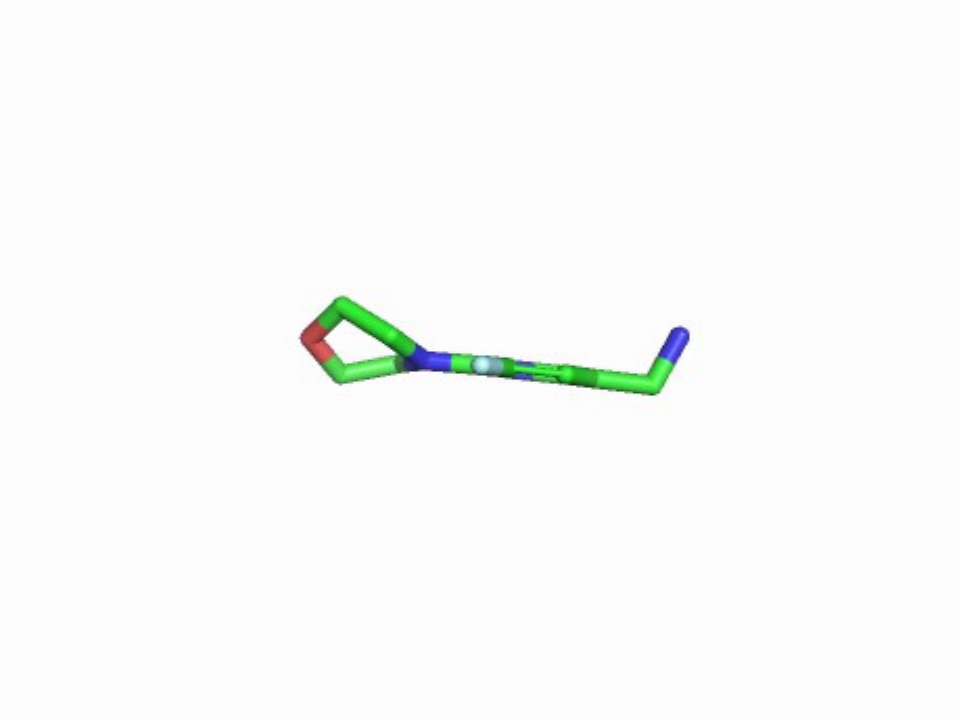
Description d’un exemple simple de calcul sur Cascimodot

```
[40]: %%time
Lmol = get_Lmol_3D_from_Lsmile_on_cluster(Lsmile)
len(Lmol)
```

```
    INFO | 2024-06-20 14:03 |           basic |     1136 |   rsync -e ssh -az --exclude-from=/home/krezel@ICOA-03.LOCAL/Documents/Programmes/Gitlab/tools_SBC/rsync_exclude.txt --delete-after /home/krezel@ICOA-03.LOCAL/Documents/Programmes/Gitlab/tools_SBC pkrezel@leto.cascimodot.datacentre-valdeloire.fr:/home/pkrezel/modules
    INFO | 2024-06-20 14:03 |           basic |     1136 |   rsync -e ssh -az --exclude-from=/home/krezel@ICOA-03.LOCAL/Documents/Programmes/Gitlab/tools_SBC/rsync_exclude.txt --delete-after /home/krezel@ICOA-03.LOCAL/Documents/Programmes/Gitlab/F2D pkrezel@leto.cascimodot.datacentre-valdeloire.fr:/home/pkrezel/modules
    INFO | 2024-06-20 14:03 |           basic |     1136 |   rsync -e ssh -az --exclude-from=/home/krezel@ICOA-03.LOCAL/Documents/Programmes/Gitlab/tools_SBC/rsync_exclude.txt --delete-after /home/krezel@ICOA-03.LOCAL/Documents/Programmes/Gitlab/xScav pkrezel@leto.cascimodot.datacentre-valdeloire.fr:/home/pkrezel/modules
    INFO | 2024-06-20 14:03 |         cluster |      221 |   * path_local=/mnt/49e06d45-a011-4e38-aa26-3981a55ace61/Projets/065_Leash_Bio/leash-BELKA
    INFO | 2024-06-20 14:03 |           basic |     1122 |   ssh pkrezel@leto.cascimodot.datacentre-valdeloire.fr 'scancel --user pkrezel--name get_mol_3D_from_smile'
    INFO | 2024-06-20 14:03 |         cluster |      231 |   Remove of directory get_mol_3D_from_smile on cluster ...
    INFO | 2024-06-20 14:03 |           basic |     1122 |   ssh pkrezel@leto.cascimodot.datacentre-valdeloire.fr 'rm -rf /home/pkrezel/get_mol_3D_from_smile'
    INFO | 2024-06-20 14:03 |         cluster |      239 |   Remove of directory of results in local
    INFO | 2024-06-20 14:03 |         cluster |      245 |   Creation of the directory to send
    INFO | 2024-06-20 14:03 |         cluster |      247 |   copy of variables
    INFO | 2024-06-20 14:03 |           basic |     4938 |   split_L_in_LL
    INFO | 2024-06-20 14:03 |           basic |     4945 |      with nL0=1145
    INFO | 2024-06-20 14:03 |           basic |     4960 |      with nL=200
    INFO | 2024-06-20 14:03 |           basic |     4972 |      with nLL_max=1000
    INFO | 2024-06-20 14:03 |           basic |     4975 |      with nL=200
    INFO | 2024-06-20 14:03 |           basic |     5009 |      len(LL)=6
    INFO | 2024-06-20 14:03 |           basic |     5025 |      len(LL)=6
    INFO | 2024-06-20 14:03 |           basic |     1680 |   dump_Lvar
    INFO | 2024-06-20 14:03 |         cluster |      352 |   Transfert  of get_mol_3D_from_smile on cluster ...
    INFO | 2024-06-20 14:03 |           basic |     1122 |   scp -C -r get_mol_3D_from_smile pkrezel@leto.cascimodot.datacentre-valdeloire.fr:/home/pkrezel
    INFO | 2024-06-20 14:03 |         cluster |      360 |   Launch of jobs ...
    INFO | 2024-06-20 14:03 |           basic |     1122 |   ssh pkrezel@leto.cascimodot.datacentre-valdeloire.fr 'cd /home/pkrezel/get_mol_3D_from_smile;bash job_all.sh'
    INFO | 2024-06-20 14:03 |         cluster |      368 |   on going ...
    INFO | 2024-06-20 14:03 |         cluster |      372 |   done in 0 s
    INFO | 2024-06-20 14:03 |         cluster |      376 |   loading of the results
    INFO | 2024-06-20 14:03 |           basic |     1122 |   scp -C -r pkrezel@leto.cascimodot.datacentre-valdeloire.fr:/home/pkrezel/get_mol_3D_from_smile/dir_Lres /mnt/49e06d45-a011-4e38-aa26-3981a55ace61/Projets/065_Leash_Bio/leash-BELKA/get_mol_3D_from_smile
    INFO | 2024-06-20 14:03 |           basic |     4364 |   load_Lvar in /mnt/49e06d45-a011-4e38-aa26-3981a55ace61/Projets/065_Leash_Bio/leash-BELKA/get_mol_3D_from_smile/dir_Lres
    INFO | 2024-06-20 14:03 |           basic |     4370 |      6 files
CPU times: user 45.9 ms, sys: 150 ms, total: 196 ms
Wall time: 14.9 s
```

```
[40]: 1145
```

Description d’un exemple simple de calcul sur Cascimodot



Présentation des différentes étapes permettant de lancer un calcul.

1-Mise à jour des modules

2-Suppression de répertoires et de jobs sur le cluster.

3-Création d’un répertoire globale

4-Copie du répertoire global sur le cluster

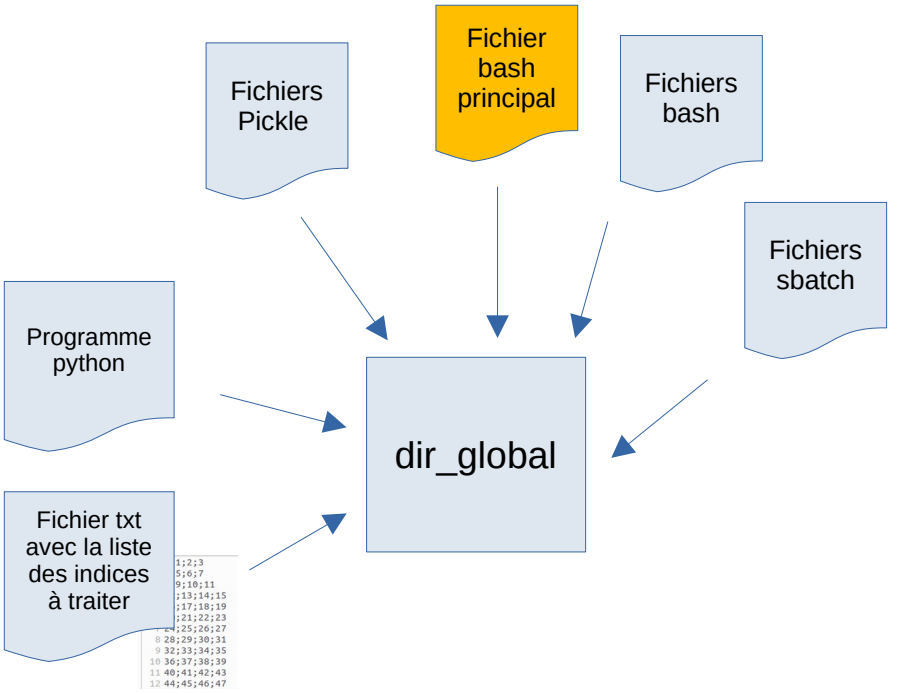
5-Lancement du fichier bash principal et attente de la fin des calculs

6-Rapitriement des résultats en local

Etape principale : Création d'un répertoire global

Tous les fichiers nécessaires au calcul sont rassemblés dans un répertoire global.

Après transfert sur le cluster, il suffit de lancer le fichier bash principal et d'attendre.



Conditions nécessaires pour que ça marche

Avoir un compte sur le cluster

Avoir supprimé la demande de mot de passe lié à ssh.

Description d’un exemple un peu plus compliqué

Soit une liste, Lmol, de 14 534 molécules (objets rdkit) générées au sein d’une cavité. Je veux les minimiser au sein de cette même cavité à l’aide de l’outil SMINA.

La conversion des objets rdkit en fichiers sdf attendus par SMINA prend un peu de temps, on souhaite donc la faire sur le cluster. De plus, des objets rdkit sont attendus en retour.

[206]:
```
@do_on_cluster(
    sl_py = f"""
import os, sys
# ========================================================================
from tools_SBC.logging_SBC import  (set_log, logger)
from tools_SBC.basic import (dump_var, get_line_n_in_file, move_file, touch,
            remove_file, remove_Lfile, ya, yapa)
# ========================================================================
from tools_SBC.rdkit_SBC import (create_sdf_from_Lmol, load_Lmol)
from tools_SBC.smina_SBC import minimize_sdf_with_smina_core
# ========================================================================
i_line = int(sys.argv[1])
file = get_line_n_in_file("Ls.txt", i_line)
Lmol = load_var(file)
sdf = file.replace(".p", ".sdf")
sdf_out = sdf.replace(".sdf", "_minimized.sdf")
sdf = create_sdf_from_Lmol(Lmol,sdf)
# ------------------------------------------------------------------------------
minimize_sdf_with_smina_core(
    sdf,
    sdf_out=sdf_out,
    dmax_from_Mp_site=4,
    protein="protein",
    cmd_smina="smina.static",
    )
Lmol_min = load_Lmol(sdf_out)
remove_Lfile([sdf, sdf_out, file])
dump_var(Lmol_min, f"dir_Lres/{{file}}")
""",
        Dcluster=Dcluster,
        with_GNU_parallel=True,
        ref="minimize_with_smina",
        level="INFO", # the logging level on the cluster
        nLmin_by_job=40, # Number min of cases to treat by job
        ncore_max=80, # number max of core to use
        njob_max=1000,
        )
def get_Lmol_minimized_with_smina_on_cluster(
        Lmol,
        LTvar_to_copy=[("protein")],
    ):
        logger.info("get_Lmol_minimized_with_smina_on_cluster")
```

```
INFO | 2024-06-20 16:22 |         cluster |      198 |   do_on_cluster
```

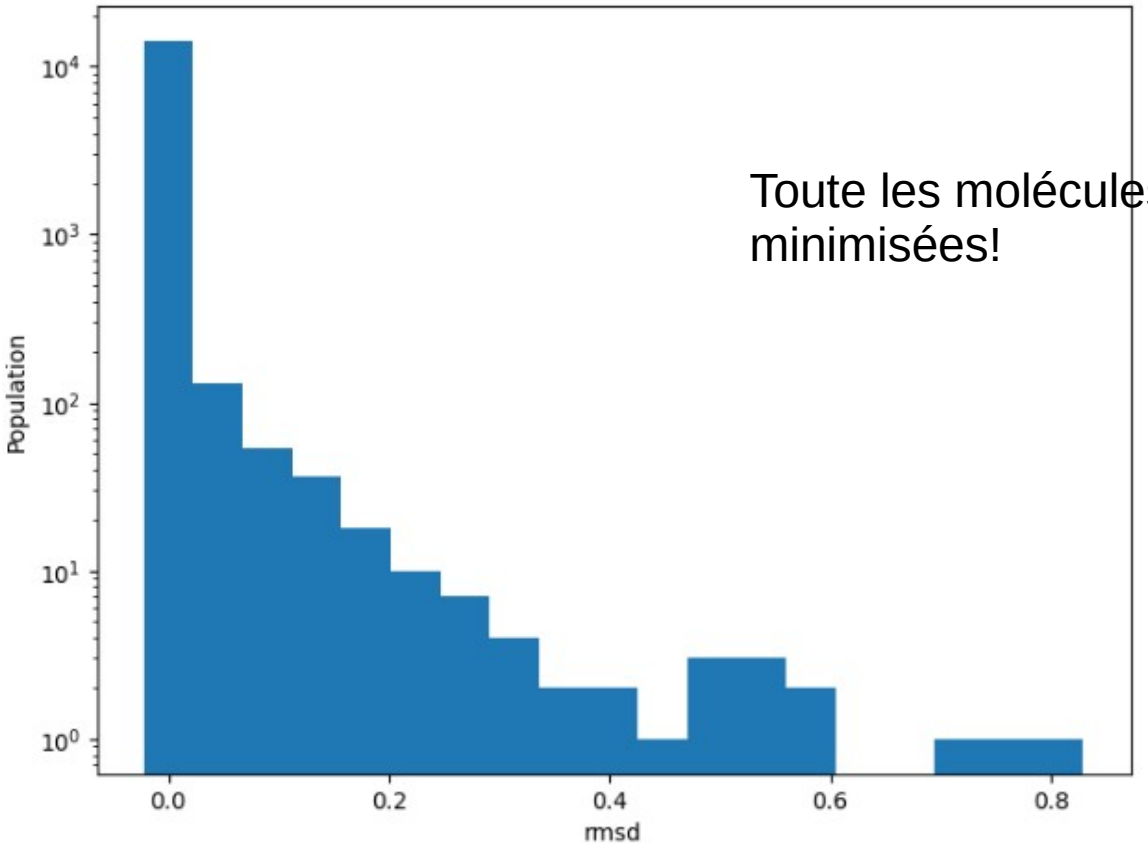
Description d’un exemple un peu plus compliqué de calcul sur Cascimodot

[207]:
```
%%time
os.chdir(f"{path_emolgine}/pdb_prepared/{pdb2}/site_0")
Lmol_min = get_Lmol_minimized_with_smina_on_cluster(
                                                    Lmol,
                                                    LTvar_to_copy=["protein.pdb"],
)
len(Lmol_min)
```

```
    INFO | 2024-06-20 16:22 |           basic |     1136 |   rsync -e ssh -az --exclude-from=/home/krezel@ICOA-03.LOCAL/Documents/Programmes/Gitlab/tools_SBC/rsync_ex
clude.txt --delete-after /home/krezel@ICOA-03.LOCAL/Documents/Programmes/Gitlab/None pkrezel@leto.cascimodot.datacentre-valdeloire.fr:/home/pkrezel/modules
    INFO | 2024-06-20 16:22 |      2146362920 |       42 |   get_Lmol_minimized_with_smina_on_cluster
    INFO | 2024-06-20 16:22 |         cluster |      221 |   * path_local=/mnt/49e06d45-a011-4e38-aa26-3981a55ace61/emolgine/pdb_prepared/5q0o/site_0
    INFO | 2024-06-20 16:22 |           basic |     1122 |   ssh pkrezel@leto.cascimodot.datacentre-valdeloire.fr 'scancel --user pkrezel--name minimize_with_smina'
    INFO | 2024-06-20 16:22 |         cluster |      231 |   Remove of directory minimize_with_smina on cluster ...
    INFO | 2024-06-20 16:22 |           basic |     1122 |   ssh pkrezel@leto.cascimodot.datacentre-valdeloire.fr 'rm -rf /home/pkrezel/minimize_with_smina'
    INFO | 2024-06-20 16:22 |         cluster |      239 |   Remove of directory of results in local
    INFO | 2024-06-20 16:22 |         cluster |      245 |   Creation of the directory to send
    INFO | 2024-06-20 16:22 |         cluster |      247 |   copy of variables
    INFO | 2024-06-20 16:22 |         cluster |      252 |       copy of protein.pdb
    INFO | 2024-06-20 16:22 |           basic |     4938 |   split_L_in_LL
    INFO | 2024-06-20 16:22 |           basic |     4945 |      with nL0=14534
    INFO | 2024-06-20 16:22 |           basic |     4960 |      with nL=40
    INFO | 2024-06-20 16:22 |           basic |     4972 |      with nLL_max=1000
    INFO | 2024-06-20 16:22 |           basic |     4975 |      with nL=40
    INFO | 2024-06-20 16:22 |           basic |     5009 |      len(LL)=364
    INFO | 2024-06-20 16:22 |           basic |     5025 |      len(LL)=364
    INFO | 2024-06-20 16:22 |           basic |     1680 |   dump_Lvar
    INFO | 2024-06-20 16:22 |         cluster |      352 |   Transfert  of minimize_with_smina on cluster ...
    INFO | 2024-06-20 16:22 |           basic |     1122 |   scp -C -r minimize_with_smina pkrezel@leto.cascimodot.datacentre-valdeloire.fr:/home/pkrezel
    INFO | 2024-06-20 16:22 |         cluster |      360 |   Launch of jobs ...
    INFO | 2024-06-20 16:22 |           basic |     1122 |   ssh pkrezel@leto.cascimodot.datacentre-valdeloire.fr 'cd /home/pkrezel/minimize_with_smina;bash job_all.s
h'
    INFO | 2024-06-20 16:24 |         cluster |      368 |   on going ...
    INFO | 2024-06-20 16:24 |         cluster |      372 |   done in 0 s
    INFO | 2024-06-20 16:24 |         cluster |      376 |   loading of the results
    INFO | 2024-06-20 16:24 |           basic |     1122 |   scp -C -r pkrezel@leto.cascimodot.datacentre-valdeloire.fr:/home/pkrezel/minimize_with_smina/dir_Lres /mn
t/49e06d45-a011-4e38-aa26-3981a55ace61/emolgine/pdb_prepared/5q0o/site_0/minimize_with_smina
    INFO | 2024-06-20 16:24 |           basic |     4364 |   load_Lvar in /mnt/49e06d45-a011-4e38-aa26-3981a55ace61/emolgine/pdb_prepared/5q0o/site_0/minimize_with_sm
ina/dir_Lres
    INFO | 2024-06-20 16:24 |           basic |     4370 |      364 files
CPU times: user 5.54 s, sys: 628 ms, total: 6.16 s
Wall time: 2min 1s
```

[207]:
```
14534
```

Description d’un exemple un peu plus compliqué de calcul sur Cascimodot



Toute les molécules étaient bien minimisées!

Conclusion

- En appliquant cette méthodologie, tout le monde peut lancer des calculs « simples » de parallélisation des données en python sur Cascimodot ou autre cluster utilisant slurm
- Nécessite un effort de programmation en écrivant des scripts python sous forme de string.
- Actuellement, les fonctions ne peuvent pas être définies dans un module : Il faut le faire dans le notebook.
- Toute personne connaissant les décorateurs en python peut contribuer à l’amélioration du code.

Précisions sur le code

```python
def do_on_cluster(
    f0=None,
    s1_py=None,
    user_local="krezel@icoa-03.local",
    IP_local="192.168.11.160",
    # ----------------------------
    Dcluster=None,
    user_cluster=None,
    IP_cluster=None,
    cmd_smina=None,
    conda_env=None,
    queue=None,
    # ----------------------------
    with_GNU_parallel=True,
    ref=None,
    level="INFO", # the logging level on the cluster
    nLmin_by_job=200, # Number min of cases to treat by job
    ncore_max=80, # number max of core to use
    njob_max=1000,
    dir_Lres="dir_Lres",
    with_return_Lres=True,
    with_fuse_LL=True, # to fuse the differents results of the parall
    Lmodule_to_update=None,
    ):
    # ----------------------------------------
    logger.info("do_on_cluster")
    if Dcluster is not None:
        user_cluster = Dcluster["user_cluster"]
        IP_cluster = Dcluster["IP_cluster"]
        conda_env = Dcluster["conda_env"]
        cmd_smina = Dcluster["cmd_smina"]
        queue = Dcluster["queue"]
    if ref is None:
        ref = f0.__name__
    def decorator(func):
        def wrapped(*args,s1_py = s1_py, **kwargs):
            # ------------------------------------------------------
            update_module_on_cluster(
                Lmodule_to_update,
                IP_cluster,
                user_cluster,
            )
            func(*args, **kwargs)
            # ------------------------------------------------------
            Lkw = list(kwargs.keys())
            # ------------------------------------------------------
            path_cluster = f"/home/{user_cluster}"
            path_local = get_path()
            logger.info(f"* path_local={path_local}")
            # to be sure that the same job is not still running
            # ------------------------------------------------------
            cmd = f"scancel --user {user_cluster}--name {ref}"
            bash_on_remote(
                            cmd,
                            user_remote=user_cluster,
                            IP_remote=IP_cluster,
                        )
            # ------------------------------------------------------
            logger.info(f"Remove of directory {ref} on cluster ...")
            Lcmd = [f"rm -rf {path_cluster}/{ref}"]
            bash_on_remote(
                            Lcmd,
                            user_cluster,
                            IP_cluster,
                        )
```

```python
            # ======================================================================
            # Creation of a new directory to gather all the informations
            # usefull to launch the script on the cluster.
            # ----------------------------------------------------------------------
            logger.info(f"Creation of the directory to send")
            make_dir(ref)
            logger.info("copy of variables")
            if "LTvar_to_copy"  in Lkw:
                LTvar_to_copy = kwargs["LTvar_to_copy"]
                for T in LTvar_to_copy:
                    if type(T) is str:
                        logger.info(f"\t copy of {T}")
                        copy_file_in_dir(T, ref)
                    else:
                        logger.info(f"copy of {T[0]}")
                        dump_var(T[1], f"{ref}/{T[0]}")
            os.chdir(ref)
            make_dir(dir_Lres)
            path_local_ref = get_path()
            # ----------------------------------------------------------------------
            LL = split_L_in_LL(
                                args[0],
                                nL=nLmin_by_job,
                                nLL_max=njob_max,
                                )

            if len(LL) < ncore_max:
                n_core = len(LL)
            else:
                n_core = ncore_max
            Ls = dump_Lvar(LL, "L")
            create_txt_from_L(Ls, "Ls.txt")
            nLs = len(Ls)
            create_txt_from_s(str(nLs), "nLs.txt")
            # ----------------------------------------------------------------------
            # Dump of informations that can't be pass directly
            # ----------------------------------------------------------------------

            # ======================================================================
            # Creation of job_all.sh to launch of the different jobs
            # can be sbatch or bash
            # ----------------------------------------------------------------------
            Ls = f"""
#!/bin/bash
sbatch --wait job1.sh
wait
    """.split("\n")
            create_txt_from_L(Ls[1:], "job_all.sh")
            # ======================================================================
            # Creation of job1.sh slurm file
            # ----------------------------------------------------------------------
            create_slurm_job(
                                "do1.py",
                                with_GNU_parallel=with_GNU_parallel,
                                ref=ref,
                                queue=queue,
                                fname="job1.sh",
                                n_core=n_core,
                                user_cluster=user_cluster,
                                conda_env=conda_env,
                                cmd_to_insert_in_0=f"cd /home/{user_cluster}/{ref}",
                                s_parallel="parallel1",
                                )
```

```python
            # ----------------------------------------------------------------------
            # do1.py file
            # ----------------------------------------------------------------------
            # ----------------------------------------------------------------------
            if f0:
                module = f0.__module__
                sf0 = f0.__name__
                s1_py = ""
                s_py = f"""
import sys
import os
import traceback
sys.path.insert(0, "/home/{user_cluster}/modules")
# ======================================================================
from tools_SBC.logging_SBC import  (set_log, logger)
from tools_SBC.basic import (load_var, dump_var, get_line_n_in_file, move_file, tou
                             remove_file, remove_Lfile)
# ======================================================================
from {module} import {sf0}
# ----------------------------------------------------------------------
i_line = int(sys.argv[1])
file = get_line_n_in_file("Ls.txt", i_line)
Lx = load_var(file)
exec("Lx2 = [{sf0}(x) for x in Lx]")
dump_var(Lx2, f"{dir_Lres}/{{file}}")
remove_file(file)
"""
            elif s1_py :
                s_py = f"""
import sys
import os
import traceback
sys.path.insert(0, "/home/{user_cluster}/modules")
# ======================================================================
from tools_SBC.basic import (load_var)
"""
            Ls2 = []
            if "LTvar_to_copy"  in Lkw:
                for T in LTvar_to_copy:
                    if "." not in T[0]:
                        Ls2.append(f'{T[0]} = load_var("{T[0]}")')
            # ----------------------------------------------------------------------
            Ls = s_py.split("\n")[1:] + Ls2 + s1_py.split("\n")[1:]
            create_txt_from_L(Ls, "do1.py")
            # ======================================================================
            # Export of the directory on the cluster and launch of the job
            # ----------------------------------------------------------------------
            os.chdir("..")
            logger.info(f"Transfert  of {ref} on cluster ...")
            bash(
                        f"scp -C -r {ref} {user_cluster}@{IP_cluster}:{path_cluster
                        info=True,
                )
            # ----------------------------------------------------------------------
            # Launch of the jobs
            # ----------------------------------------------------------------------
            logger.info(f"Launch of jobs ...")
            Lcmd = [f"cd {path_cluster}/{ref}",
                    "bash job_all.sh"]
            bash_on_remote(
                        Lcmd,
                        user_cluster,
                        IP_cluster,
                    )
            logger.info(f"on going ...")
            logger.debug(f"wait for results")
            start_time = time.time()
            # ----------------------------------------------------------------------
            logger.info(f"done in {round(time.time() - start_time)} s")
            # ----------------------------------------------------------------------
```

```python
            # ----------------------------------------------------------------------
            # Loading of results
            # ----------------------------------------------------------------------
            logger.info(f"loading of the results")
            os.chdir(path_local_ref)
            remove_dir(dir_Lres)
            bash(
                        f"scp -C -r {user_cluster}@{IP_cluster}:{path_cluster}/{ref}/{dir_
                        info=True,
                    )
            # ----------------------------------------------------------------------

            Lcmd = [f"rm -rf {path_cluster}/{ref}"]
            #bash_on_remote(Lcmd, user_cluster, IP_cluster)
            if with_return_Lres:
                os.chdir(dir_Lres)
                LLres = load_Lvar("L*")
                os.chdir(path_local)
                if with_fuse_LL:
                    return fuse_LL(LLres)
                else:
                    return LLres
            os.chdir(path_local)
        return wrapped
    return decorator
```