ColabFold 1.5.5 (local 版)

ウェブページ

https://github.com/sokrypton/ColabFold

導入手順概略

https://github.com/sokrypton/ColabFold/wiki/Running-ColabFold-in-Docker
の内容を参考に、データベースをローカルに持ち、インターネットへアクセスしない形で導入しています。
Docker/apptainer(singularity) についても今回は利用していません。

Python 環境構築

```
$ sh Miniforge3-Linux-x86_64.sh
$ cd /apl/colabfold/1.5.5
$ ./bin/conda shell.bash hook > conda_init.sh
$ ./bin/conda shell.tcsh hook > conda_init.csh
$ . /apl/colabfold/1.5.5/conda_init.sh
$ conda install cudatoolkit=11.8.0
$ CONDA_OVERRIDE_CUDA=11.8.0 conda install -c conda-forge -c bioconda colabfold=1.5.5=* jaxlib=*=*cuda* libabseil libgrpc python mmseqs2
vmtouch
```

データベースの取得

ColabFold 1.5.5 のリポジトリ中にある setup_databases.sh の内容を手動で実行。ただし、ここで使う mmseqs についてはリポジトリ中の MsaServer/setup-and-start-local.sh 中で指定されていたものを使用。またインデックスの作成(createindex)については省略。データベースは /apl/colabfold/1.5.5/MsaServer/databases 以下に作成。AlphaFold2 のパラメータについても /apl/colabfold/1.5.5/MsaServer/params に配置。(msa-server そのものは今回は利用しません。)

Lustre 上に配置した大きなファイルについては、複数の OST に置くようにしています。(lustre ファイルシステム限定の操作)

```
$ lfs migrate -c 2 colabfold_envdb_202108_db_h.index # 17GB
$ lfs migrate -c 2 colabfold_envdb_202108_db_seq.index # 18GB
$ lfs migrate -c 2 pdb100_foldseek_230517.tar.gz # 18GB
$ lfs migrate -c 3 colabfold_envdb_202108_db_h # 24GB
$ lfs migrate -c 3 colabfold_envdb_202108_db # 25GB
$ lfs migrate -c 3 colabfold_envdb_202108_db_aln # 27GB
$ lfs migrate -c 3 uniref30_2302_aln.tsv # 29GB
$ lfs migrate -c 3 colabfold_envdb_202108_h.tsv # 30GB
$ lfs migrate -c 4 colabfold_envdb_202108.tsv # 38GB
$ lfs migrate -c 5 uniref30_2302_db_h # 41GB
$ lfs migrate -c 5 uniref30_2302_h.tsv # 44GB
$ lfs migrate -c 5 colabfold_envdb_202108_aln.tsv # 52GB
$ lfs migrate -c 6 pdb100_a3m.ffdata # 60GB
$ lfs migrate -c 8 uniref30_2302_db_seq # 78GB
$ lfs migrate -c 10 colabfold_envdb_202108_db_seq # 87GB
$ lfs migrate -c 10 uniref30_2302.tar.gz # 96GB
$ lfs migrate -c 11 colabfold_envdb_202108.tar.gz # 110GB
$ lfs migrate -c 12 uniref30_2302_seq.tsv # 128GB
$ lfs migrate -c 12 colabfold_envdb_202108_seq.tsv # 128GB
```

PBS 用実行スクリプトサンプル

```
#!/bin/sh
#PBS -l select=1:ncpus=128:mpiprocs=1:ompthreads=128
#PBS -l walltime=24:00:00

if [ ! -z "${PBS_O_WORKDIR}" ]; then
 cd "${PBS_O_WORKDIR}"
fi

# input data and intermediate/work dirs
NUM_RELAX=1               # number of structures to be relaxed after inference
INPUTFASTA=./monomer.fasta # input sequence
```

```
MSADIR=./msas             # intermediate msa directory
OUTPUTDIR=./output        # output of colabfold_batch
MMSEQS_NUM_THREADS=$OMP_NUM_THREADS # number of threads; exporting this may work

# common params
COLABFOLDROOT=/apl/colabfold/1.5.5
. ${COLABFOLDROOT}/conda_init.sh

# search options
MMSEQS=${COLABFOLDROOT}/bin/mmseqs
DBDIR=${COLABFOLDROOT}/MsaServer/databases
colabfold_search --mmseqs ${MMSEQS} \
            --threads ${MMSEQS_NUM_THREADS} \
  ${INPUTFASTA} ${DBDIR} ${MSADIR}

# run prediction
AF2WEIGHTS=${COLABFOLDROOT}/MsaServer # af2 weight
colabfold_batch \
  --num-relax ${NUM_RELAX} \
  --data ${AF2WEIGHTS} \
  ${MSADIR} ${OUTPUTDIR}
```

メモ

- 計算ノードでローカルに msa-server を立てた場合、予測が最後まで完了させられなかった(template 関連のエラーが見える)
  - センターの運用上、事前にメモリ上にデータを置くことが困難であるため、速度の向上も望めない。そのため、サーバについては今回は使わない方針とした。
- 小規模な予測であれば、公式サイトの記述通り、colabfold_batch に直接予測すべき配列を与え、公式のサーバにアクセスした方が効率が良いと思われます。
- MsaServer/setup-and-start-local.sh で指定されているバージョンの mmseqs2 を colabfold_search で使う場合、complex に対する処理が正常に終了しない(pairing 関連のオプションについてエラー?)。conda で導入できる mmseqs2 ならば問題無し。
  - complex については配列間を：でつなげて入れる(AAAAAA:GGGGG)ことで指定できるとのことです。予測後の構造緩和まで問題なく動作しているように見えます。
- 公式サイトの記述通り、DB のインデックスを作成する場合はメモリにデータをあらかじめのせておくような操作を行わないと速度がかなり落ちてしまう。