## AlphaFold2 (2021/7/20)

## Main site

https://github.com/deepmind/alphafold

## References

- The paper (Nature): https://doi.org/10.1038/s41586-021-03819-2
- non-docker version: https://github.com/kalininalab/alphafold_non_docker
    - we emlpoyed this non-docker version.
- installation guide by @Ag_smith (Japanese): https://qiita.com/Ag_smith/items/7c76438906b3f665af38

## Installation

(The latest version of AlphaFold2 on Jul 20 (last commit: 2021/7/16; --preset=reduced_dbs is not available) was installed.)
The installation procedure is an analog to non-docker one above.

### miniconda

AlphaFold2-specific conda environment was prepared.

```
$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
$ sh Miniconda3-latest-Linux-x86_64.sh
```

The installation directory is /local/apl/lx/alphafold2-20210720/miniconda3.

```
[/home/users/***/miniconda3] >>> /local/apl/lx/alphafold2-20210720/miniconda3
```

### preparation of conda env

The conda environment is only for AlphaFold2; this env will be loaded only when you use AlphaFold2. We thus prepare initialization scripts (for csh/bash) in alphafold directory.

```
[user@ccfep5 ~]$ cd /local/apl/lx/alphafold2-20210720/
[user@ccfep5 alphafold2-20210720]$ ./miniconda3/bin/conda shell.bash hook > conda_init.sh
[user@ccfep5 alphafold2-20210720]$ ./miniconda3/bin/conda shell.csh hook > conda_init.csh
```

Now update and install packages.

```
[user@ccfep5 alphafold2-20210720]$ . /local/apl/lx/alphafold2-20210720/conda_init.sh
(base) [user@ccfep5 ~]$ conda install -y -c conda-forge cudatoolkit=11.0.3 openmm cudnn pdbfixer
...
(skipped)
...
  cudatoolkit conda-forge/linux-64::cudatoolkit-11.0.3-h15472ef_8
  cudnn conda-forge/linux-64::cudnn-8.2.1.32-h86fa8c9_0
  fftw conda-forge/linux-64::fftw-3.3.9-nompi_h74d3f13_101
  libblas conda-forge/linux-64::libblas-3.9.0-9_openblas
  libcblas conda-forge/linux-64::libcblas-3.9.0-9_openblas
  libgfortran-ng conda-forge/linux-64::libgfortran-ng-9.3.0-hff62375_19
  libgfortran5 conda-forge/linux-64::libgfortran5-9.3.0-hff62375_19
  liblapack conda-forge/linux-64::liblapack-3.9.0-9_openblas
  libopenblas conda-forge/linux-64::libopenblas-0.3.15-pthreads_h8fe5266_1
  numpy conda-forge/linux-64::numpy-1.21.1-py38h9894fe3_0
  ocl-icd conda-forge/linux-64::ocl-icd-2.3.0-h7f98852_0
  ocl-icd-system conda-forge/linux-64::ocl-icd-system-1.0.0-1
  openmm conda-forge/linux-64::openmm-7.5.1-py38h7850c2e_1
  pdbfixer conda-forge/noarch::pdbfixer-1.7-pyhd3deb0d_0
  python_abi conda-forge/linux-64::python_abi-3.8-2_cp38
...
(skipped)
...
(base) [user@ccfep5 ~]$ conda install -c bioconda hmmer hhsuite kalign2
...
```

```
(skipped)
...
  hhsuite bioconda/linux-64::hhsuite-3.3.0-py38pl5262hc37a69a_2
  hmmer bioconda/linux-64::hmmer-3.3.2-h1b792b2_1
  kalign2 bioconda/linux-64::kalign2-2.04-h779adbc_2
  perl pkgs/main/linux-64::perl-5.26.2-h14c3975_0
...
(skipped)
...
```

## installation of alphafold and pip packages

```
(base) [user@ccfep5 ~]$ cd /local/apl/lx/alphafold2-20210720/
(base) [user@ccfep5 alphafold2-20210720]$ git clone https://github.com/deepmind/alphafold.git
(base) [user@ccfep5 alphafold2-20210720]$ wget -q -P alphafold/alphafold/common/ https://git.scicore.unibas.ch/schwede/openstructure/-/raw/7102c63615b64735c4941278d92b554ec94415f8/modules/mol/alg/src/stereo_chemical_props.txt
(base) [user@ccfep5 alphafold2-20210720]$ conda install -c conda-forge pip
(base) [user@ccfep5 alphafold2-20210720]$ pip install absl-py==0.13.0 biopython==1.79 chex==0.0.7 dm-haiku==0.0.4 dm-tree==0.1.6 immutabledict==2.0.0 jax==0.2.14 ml-collections==0.1.0 numpy==1.19.5 scipy==1.7.0 tensorflow==2.5.0
...
Successfully installed PyYAML-5.4.1 absl-py-0.13.0 astunparse-1.6.3 biopython-1.79 cachetools-4.2.2 chex-0.0.7 contextlib2-21.6.0 dm-haiku-0.0.4 dm-tree-0.1.6 flatbuffers-1.12 gast-0.4.0 google-auth-1.33.0 google-auth-oauthlib-0.4.4 google-pasta-0.2.0 grpcio-1.34.1 h5py-3.1.0 immutabledict-2.0.0 jax-0.2.14 jaxlib-0.1.69 keras-nightly-2.5.0.dev2021032900 keras-preprocessing-1.1.2 markdown-3.3.4 ml-collections-0.1.0 numpy-1.19.5 oauthlib-3.1.1 opt-einsum-3.3.0 protobuf-3.17.3 pyasn1-0.4.8 pyasn1-modules-0.2.8 requests-oauthlib-1.3.0 rsa-4.7.2 scipy-1.7.0 six-1.15.0 tabulate-0.8.9 tensorboard-2.5.0 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.0 tensorflow-2.5.0 tensorflow-estimator-2.5.0 termcolor-1.1.0 toolz-0.11.1 typing-extensions-3.7.4.3 werkzeug-2.0.1 wrapt-1.12.1
...
(base) [user@ccfep5 alphafold2-20210720]$ pip install --upgrade jax jaxlib==0.1.69+cuda111 -f https://storage.googleapis.com/jax-releases/jax_releases.html
...
Successfully installed jax-0.2.17 jaxlib-0.1.69+cuda111
...
```

Patch for openmm is applied here.

```
(base) [user@ccfep5 site-packages]$ cd /local/apl/lx/alphafold2-20210720/miniconda3/lib/python3.8/site-packages
(base) [user@ccfep5 site-packages]$ patch -p0 < /local/apl/lx/alphafold2-20210720/alphafold/docker/openmm.patch
```

## Databases

aria2 must be installed beforehand.

```
(base) [user@ccfep5 scripts]$ mkdir /local/apl/lx/alphafold2-20210720/databases
(base) [user@ccfep5 scripts]$ cd /local/apl/lx/alphafold2-20210720/alphafold/scripts
(base) [user@ccfep5 scripts]$ ./download_all_data.sh /local/apl/lx/alphafold2-20210720/databases
```

(Some of databases were installed manually later, since ftp access was not allowed in RCCS...)

## To avoid I/O issues (lustre specific)

(not available for standard file system such as ext4/xfs.)
Some of database files are so large (> 10 GB). If many of users access those files at the same time, it will cause a severe I/O trouble. We thus distributed contents of large files over OSTs; the I/O load will also be distributed. We are not very sure whether this operation is effective, though.

```
[user@ccfep5 ~]$ cd /local/apl/lx/alphafold2-20210720/databases/bfd/
[user@ccfep5 bfd]$ lfs migrate -c -1 bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt_a3m.ffdata
[user@ccfep5 bfd]$ lfs migrate -c 4 bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt_cs219.ffdata
[user@ccfep5 bfd]$ lfs migrate -c 64 bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt_hhm.ffdata
```

```
[user@ccfep5 ~]$ cd /local/apl/lx/alphafold2-20210720/databases/mgnify/
[user@ccfep5 mgnify]$ lfs migrate -c 16 mgy_clusters.fa
```

```
[user@ccfep5 ~]$ cd /local/apl/lx/alphafold2-20210720/databases/pdb70/
[user@ccfep5 pdb70]$ lfs migrate -c 12 pdb70_a3m.ffdata
```

```
[user@ccfep3 ~]$ cd /local/apl/lx/alphafold2-20210720/databases/uniclust30/
[user@ccfep3 uniclust30_2018_08]$ lfs migrate -c 3 uniclust30_2018_08_hhm.ffdata
[user@ccfep3 uniclust30_2018_08]$ lfs migrate -c 2 uniclust30_2018_08_cs219.ffdata
[user@ccfep3 uniclust30_2018_08]$ lfs migrate -c 2 uniclust30_2018_08.cs219
[user@ccfep3 uniclust30_2018_08]$ lfs migrate -c 16 uniclust30_2018_08_a3m.ffdata
```

```
[user@ccfep3 ~]$ cd /local/apl/lx/alphafold2-20210720/databases/uniref90/
[user@ccfep3 uniref90]$ lfs migrate -c 12 uniref90.fasta
```

## Launch script preparation

We employ https://github.com/kalininalab/alphafold_non_docker/blob/main/run_alphafold.sh with some modifications described below. (Fix related to newline in the end of file is excluded.) Some default settings (database path, number of models) were changed.

```
--- run_alphafold.sh    2021-07-26 09:17:39.000000000 +0900
+++ run_alphafold_rccs_mod.sh    2021-07-26 09:23:07.000000000 +0900
@@ -1,21 +1,36 @@
 #!/bin/bash
 # Description: AlphaFold non-docker version
 # Author: Sanjay Kumar Srikakulam
+#
+#
+# RCCS notes:
+# This script was customized for RCCS by M. Kamiya (IMS).
+# original: https://github.com/kalininalab/alphafold_non_docker
+
+# RCCS default value
+af2root="/local/apl/lx/alphafold2-20210720/alphafold"
+data_dir="/local/apl/lx/alphafold2-20210720/databases"
+
+alphafold_script="$af2root/run_alphafold.py"
+max_template_date=None
+benchmark=false
+preset="full_dbs"
+model_names="model_1,model_2,model_3,model_4,model_5"

 usage() {
         echo ""
         echo "Usage: $0 <OPTIONS>"
         echo "Required Parameters:"
-        echo "-d <data_dir>     Path to directory of supporting data"
         echo "-o <output_dir>   Path to a directory that will store the results."
-        echo "-m <model_names>  Names of models to use (a comma separated list)"
         echo "-f <fasta_path>   Path to a FASTA file containing one sequence"
         echo "-t <max_template_date> Maximum template release date to consider (ISO-8601 format - i.e. YYYY-MM-DD). Important if folding historical test sets"
         echo "Optional Parameters:"
+        echo "-d <data_dir>     Path to directory of supporting data"
+        echo "-m <model_names>  Names of models to use (a comma separated list)"
         echo "-b <benchmark>    Run multiple JAX model evaluations to obtain a timing that excludes the compilation time, which should be more indicative of the time required for inferencing many
     proteins (default: 'False')"
-        echo "-g <use_gpu>      Enable NVIDIA runtime to run with GPUs (default: 'True')"
-        echo "-a <gpu_devices>  Comma separated list of devices to pass to 'CUDA_VISIBLE_DEVICES' (default: 'all')"
+        #echo "-g <use_gpu>      Enable NVIDIA runtime to run with GPUs (default: 'True')"
+        echo "-a <gpu_devices>  Comma separated list of devices to pass to 'CUDA_VISIBLE_DEVICES' (default: '')"
         echo "-p <preset>       Choose preset model configuration - no ensembling (full_dbs) or 8 model ensemblings (casp14) (default: 'full_dbs')"
         echo ""
         exit 1
@@ -42,7 +57,7 @@
```

```
                benchmark=true
        ;;
        g)
-               use_gpu=true
+                #use_gpu=true
        ;;
        a)
                gpu_devices=$OPTARG
@@ -58,47 +73,18 @@
     usage
 fi

-if [[ "$max_template_date" == "" ]] ; then
-    max_template_date=None
-fi
-
-if [[ "$benchmark" == "" ]] ; then
-    benchmark=false
-fi
-
-if [[ "$use_gpu" == "" ]] ; then
-    use_gpu=true
-fi
-
-if [[ "$gpu_devices" == "" ]] ; then
-    gpu_devices="all"
-fi
-
-if [[ "$preset" == "" ]] ; then
-    preset="full_dbs"
-fi
-
 if [[ "$preset" != "full_dbs" && "$preset" != "casp14" ]] ; then
     echo "Unknown preset! Using default ('full_dbs')"
     preset="full_dbs"
 fi

-# This bash script looks for the run_alphafold.py script in its current working directory, if it does not exist then exits
-current_working_dir=$(pwd)
-alphafold_script="$current_working_dir/run_alphafold.py"
-
 if [ ! -f "$alphafold_script" ]; then
     echo "Alphafold python script $alphafold_script does not exist."
     exit 1
 fi
-# Export ENVIRONMENT variables and set CUDA devices for use
-if [[ "$use_gpu" == true ]] ; then
-    export CUDA_VISIBLE_DEVICES=0
-
-    if [[ "$gpu_devices" ]] ; then
-        export CUDA_VISIBLE_DEVICES=$gpu_devices
-    fi
+if [[ "$gpu_devices" ]] ; then
+    export CUDA_VISIBLE_DEVICES=$gpu_devices
 fi

 export TF_FORCE_UNIFIED_MEMORY='1'
```

The file was named run_alphafold_rccs.sh and placed under /local/apl/lx/alphafold2-20210720.

(Aug 2, 2021) additional patch below was applied.

```
--- run_alphafold_rccs.sh.20210726    2021-07-26 09:23:07.000000000 +0900
+++ run_alphafold_rccs.sh.20210802    2021-08-02 09:15:17.000000000 +0900
@@ -27,6 +27,7 @@
```

```
        echo "Optional Parameters:"
        echo "-d <data_dir>     Path to directory of supporting data"
        echo "-m <model_names>  Names of models to use (a comma separated list)"
+       echo "-Q                show also pTM score etc. (modify default model names to model_1_ptm,...)"
        echo "-b <benchmark>    Run multiple JAX model evaluations to obtain a timing that excludes the compilation time, which should be more
indicative of the time required for inferencing many
    proteins (default: 'False')"
        #echo "-g <use_gpu>      Enable NVIDIA runtime to run with GPUs (default: 'True')"
@@ -36,7 +37,7 @@
        exit 1
 }

-while getopts ":d:o:m:f:t:a:p:bg" i; do
+while getopts ":d:o:m:f:t:a:p:bgQ" i; do
        case "${i}" in
        d)
                data_dir=$OPTARG
@@ -59,6 +60,9 @@
        g)
                #use_gpu=true
        ;;
+       Q)
+               model_names="model_1_ptm,model_2_ptm,model_3_ptm,model_4_ptm,model_5_ptm"
+       ;;
        a)
                gpu_devices=$OPTARG
        ;;
```

Model names will be modified to model_1_ptm,model_2_ptm,... when -Q is specified. pTM score and some additional data will be available in result_model_*_ptm.pkl files.
Ref: https://sbgrid.org/wiki/examples/alphafold2

## Fix textfile path

The path to a textfile in alphafold/common/residue_constants.py was modified. This is a small but important fix to use alphafold2 in user directory.

```
--- residue_constants.py.org    2021-07-20 13:47:44.000000000 +0900
+++ residue_constants.py        2021-07-26 09:24:53.000000000 +0900
@@ -14,6 +14,7 @@

 """Constants used in AlphaFold."""

+import os
 import collections
 import functools
 from typing import Mapping, List, Tuple
@@ -403,7 +404,8 @@
     residue_bond_angles: dict that maps resname --> list of BondAngle tuples
   """
   stereo_chemical_props_path = (
-      'alphafold/common/stereo_chemical_props.txt')
+       os.path.join( os.path.dirname( __file__ ),
+                     "stereo_chemical_props.txt" ))
   with open(stereo_chemical_props_path, 'rt') as f:
     stereo_chemical_props = f.read()
   lines_iter = iter(stereo_chemical_props.splitlines())
```

## Sample job script (for PBS)

run_alphafold_rccs.sh in this sample is the modified version of run_alphafold.sh (see above).
The amino acid seqeunce should be provided in FASTA format (query.fasta in the example).

```
#!/bin/sh
```

```
#PBS -l select=1:ncpus=12:mpiprocs=1:ompthreads=12:jobtype=core
#PBS -l walltime=72:00:00

# at least 8 cpu cores will be requested internally.
# in this sample, we employ 12 cores to use enough amount of memory.
# not sure how much is necessary/required, though.

# note about available memory:
# Available memory amount is proportional to ncpus value.
# If you need more memory, please increase ncpus in the header.

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

AF2ROOT=/local/apl/lx/alphafold2-20210720
RUNAF2=${AF2ROOT}/run_alphafold_rccs.sh

# load miniconda environment (where necessary binaries reside)
. ${AF2ROOT}/conda_init.sh

# Options required:
# -o outputdir
# -f sequence (FASTA)
# -t max template date
sh ${RUNAF2} \
    -o ./dummy_test/ \
    -f query.fasta \
    -t 2020-05-14

# note: please add -Q option to get pTM score and predicted aligned error
#       values. (default model names will be modified by this option)
```

(Aug 2, 2021 update) Sample script no longer depends on GPUs. Also, when -Q is specified, pTM score and some additional data will be available in result_model_*_ptm.pkl files. (-Q option is equivalent to "-m model_1_ptm,model_2_ptm,model_3_ptm,model_4_ptm,model_5_ptm")

```
sh ${RUNAF2} -Q \
    -o ./dummy_test/ \
    -f query.fasta \
    -t 2020-05-14
```

## Notes

- At least for short sequence cases, GPU can speedup the calculation, but not quite necessary.
  - GPU is surely employed in inference (tensorflow?) and relaxation of structure (openmm?) stages. But the most time-consuming step seems to be I/O limit, where GPU is not involved.
  - It will take 2-4 hours for structural prediction of short sequence.
  - We didn't try long sequences yet.
- (2021/8/2) update
  - According to the official repository, GPU is not quite necessary. We thus change the sample to non-GPU version.
    - (NOTE: AlphaFold still can use GPU.)
    - Ref1: https://github.com/deepmind/alphafold/commit/b3ed8603e8b5f085342c50259da0ba9fe485ef94
    - Ref2: https://twitter.com/Ag_smith/status/1421476812693991425 (japanese tweet)
  - run_alphafold*.sh script was modified to enable easier use of model_*_ptm (-Q option was added to run_alphafold*.sh)
    - Ref: https://sbgrid.org/wiki/examples/alphafold2