# AlphaFold

The project home page where you can find the latest information is at
https://github.com/deepmind/alphafold

For details on how to run the model please see the Supplementary Information article

For some ideas on how to separate the CPU and GPU parts: https://github.com/Zuricho/ParallelFold.

Alternatively - check out what has already been calculated

## Note on GPU usage

Whilst Alphafold makes use of GPUs for the inference part of the modelling, depending on the use case, this can be a small part of the running time as shown by the `timings.json` file that is produced for every run:

For the T1024 test case:


```
{
    "features": 6510.152379751205,
    "process_features_model_1_pred_0": 3.555035352706909,
    "predict_and_compile_model_1_pred_0": 124.84101128578186,
    "relax_model_1_pred_0": 25.707252502441406,
    "process_features_model_2_pred_0": 2.0465400218963623,
    "predict_and_compile_model_2_pred_0": 104.1096305847168,
    "relax_model_2_pred_0": 14.539108514785767,
    "process_features_model_3_pred_0": 1.7761900424957275,
    "predict_and_compile_model_3_pred_0": 82.07982850074768,
    "relax_model_3_pred_0": 13.683411598205566,
    "process_features_model_4_pred_0": 1.8073537349700928,
    "predict_and_compile_model_4_pred_0": 82.5819890499115,
    "relax_model_4_pred_0": 15.835367441177368,
    "process_features_model_5_pred_0": 1.9143474102020264,
    "predict_and_compile_model_5_pred_0": 77.47663712501526,
    "relax_model_5_pred_0": 14.72615647315979
}
```

That means that out of the ~2 hour run time 1h48 is spend running "classical" code (mostly hhblits) and only ~10 minutes is spent on the GPU.

***As such do not request 2 GPUs as the potential speedup is negligible and this will block resources for other users***

For multimer modelling the GPU part can take longer and depending on what you need it might be worth turning off relaxation. Always check the **timings.json** file to see where time is being spent!

If we look at the overall efficiency of the job using seff we see:

```
Nodes: 1
Cores per node: 24
CPU Utilized: 03:28:24
CPU Efficiency: 7.33% of 1-23:21:36 core-walltime
Job Wall-clock time: 01:58:24
Memory Utilized: 81.94 GB
Memory Efficiency: 40.97% of 200.00 GB
```

# Reference databases

The reference databases needed for AlphaFold have been made available in `/reference/alphafold` so there is no need to download them - the directory name is the date on which the databases were downloaded.

```
$ ls /reference/alphafold/
20210719
20211104
20220414
20221206
```

New versions will be downloaded if required.

The versions correspond to:

- `20210719` - Initial Alphafold 2.0 release
- `20211104` - 2.1 release with multimer data
- `20220414` - Updated weights
- `20221206` - Updated weights

# Using containers

The Alphafold project recommend using Docker to run the code which works on cloud or personal resources but not when using shared HPC systems as administrative access (required for Docker) is obviously not permitted.

## Singularity container

We provide Singularity image which can be used on the DCSR clusters and these can be found in /dcsrsoft/singularity/containers/

The currently available image is:

- alphafold-032e2f2.sif

When running the image directly it is necessary to provide all the paths to databases which is error prone and tedious.

```
$ singularity run /dcsrsoft/singularity/containers/alphafold-032e2f2.sif --helpshort
Full AlphaFold protein structure prediction script.
flags:

/app/alphafold/run_alphafold.py:
  --[no]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')
  --bfd_database_path: Path to the BFD database for use by HHblits.
  --data_dir: Path to directory of supporting data.
  --db_preset: <full_dbs|reduced_dbs>: Choose preset MSA database configuration - smaller
genetic database config (reduced_dbs) or full genetic database config  (full_dbs)
    (default: 'full_dbs')
  --fasta_paths: Paths to FASTA files, each containing a prediction target that will be folded
one after another. If a FASTA file contains multiple sequences, then it will be folded as a
multimer. Paths should be separated by commas. All FASTA paths must have a unique basename as
the basename is used
    to name the output directories for each prediction.
    (a comma separated list)
  --hhblits_binary_path: Path to the HHblits executable.
    (default: '/opt/conda/bin/hhblits')
  --hhsearch_binary_path: Path to the HHsearch executable.
    (default: '/opt/conda/bin/hhsearch')
```

```
  --hmmbuild_binary_path: Path to the hmmbuild executable.
    (default: '/usr/bin/hmmbuild')
  --hmmsearch_binary_path: Path to the hmmsearch executable.
    (default: '/usr/bin/hmmsearch')
  --is_prokaryote_list: Optional for multimer system, not used by the single chain system.
This list should contain a boolean for each fasta specifying true where the target complex is
from a prokaryote, and false where it is not, or where the origin is unknown. These values
determine the pairing
    method for the MSA.
    (a comma separated list)
  --jackhmmer_binary_path: Path to the JackHMMER executable.
    (default: '/usr/bin/jackhmmer')
  --kalign_binary_path: Path to the Kalign executable.
    (default: '/usr/bin/kalign')
  --max_template_date: Maximum template release date to consider. Important if folding
historical test sets.
  --mgnify_database_path: Path to the MGnify database for use by JackHMMER.
  --model_preset: <monomer|monomer_casp14|monomer_ptm|multimer>: Choose preset model
configuration - the monomer model, the monomer model with extra ensembling, monomer model with
pTM head, or multimer model
    (default: 'monomer')
  --obsolete_pdbs_path: Path to file containing a mapping from obsolete PDB IDs to the PDB IDs
of their replacements.
  --output_dir: Path to a directory that will store the results.
  --pdb70_database_path: Path to the PDB70 database for use by HHsearch.
  --pdb_seqres_database_path: Path to the PDB seqres database for use by hmmsearch.
  --random_seed: The random seed for the data pipeline. By default, this is randomly
generated. Note that even if this is set, Alphafold may still not be deterministic, because
processes like GPU inference are nondeterministic.
    (an integer)
  --small_bfd_database_path: Path to the small version of BFD used with the "reduced_dbs"
preset.
  --template_mmcif_dir: Path to a directory with template mmCIF structures, each named
<pdb_id>.cif
  --uniclust30_database_path: Path to the Uniclust30 database for use by HHblits.
  --uniprot_database_path: Path to the Uniprot database for use by JackHMMer.
  --uniref90_database_path: Path to the Uniref90 database for use by JackHMMER.
  --[no]use_precomputed_msas: Whether to read MSAs that have been written to disk. WARNING:
This will not check if the sequence, database or configuration have changed.
    (default: 'false')
```

```
Try --helpfull to get a list of all flags.
```

To run the container - here we are using a GPU so the `--nv` flag must be used to make the GPU visible inside the container

```
module load singularity


singularity run --nv /dcsrsoft/singularity/containers/alphafold-032e2f2.sif <OPTIONS>
```

## Helper Scripts

In order to make life simpler there is a wrapper script: run_alphafold_032e2f2.sh - this can be found at:

/dcsrsoft/singularity/containers/run_alphafold_032e2f2.sh

Please copy it to your working directory

```
$ bash /dcsrsoft/singularity/containers/run_alphafold_032e2f2.sh --help


Please make sure all required parameters are given
Usage: /dcsrsoft/singularity/containers/run_alphafold_032e2f2.sh <OPTIONS>
Required Parameters:
-d <data_dir>         Path to directory of supporting data
-o <output_dir>       Path to a directory that will store the results.
-f <fasta_paths>      Path to FASTA files containing sequences. If a FASTA file contains
multiple sequences, then it will be folded as a multimer. To fold more sequences one after
another, write the files separated by a comma
-t <max_template_date> Maximum template release date to consider (ISO-8601 format - i.e. YYYY-
MM-DD). Important if folding historical test sets
Optional Parameters:
-g <use_gpu>          Enable NVIDIA runtime to run with GPUs (default: true)
-r <run_relax>        Whether to run the final relaxation step on the predicted models.
Turning relax off might result in predictions with distracting stereochemical violations but
might help in case you are having issues with the relaxation stage (default: true)
-e <enable_gpu_relax> Run relax on GPU if GPU is enabled (default: true)
-n <openmm_threads>   OpenMM threads (default: all available cores)
```

```
-a <gpu_devices>      Comma separated list of devices to pass to 'CUDA_VISIBLE_DEVICES'
(default: 0)
-m <model_preset>     Choose preset model configuration - the monomer model, the monomer model
with extra ensembling, monomer model with pTM head, or multimer model (default: 'monomer')
-c <db_preset>        Choose preset MSA database configuration - smaller genetic database
config (reduced_dbs) or full genetic database config (full_dbs) (default: 'full_dbs')
-p <use_precomputed_msas> Whether to read MSAs that have been written to disk. WARNING: This
will not check if the sequence, database or configuration have changed (default: 'false')
-l <num_multimer_predictions_per_model> How many predictions (each with a different random
seed) will be generated per model. E.g. if this is 2 and there are 5 models then there will be
10 predictions per input. Note: this FLAG only applies if model_preset=multimer (default: 5)
-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')
```

An example batch script using the helper script is:

```
#!/bin/bash

#SBATCH -c 24
#SBATCH -p gpu
#SBATCH --gres=gpu:1
#SBATCH --gres-flags=enforce-binding
#SBATCH --mem 200G
#SBATCH -t 6:00:00

module purge
module load singularityce

export SINGULARITY_BINDPATH="/scratch,/dcsrsoft,/users,/work,/reference"

bash /dcsrsoft/singularity/containers/run_alphafold_032e2f2.sh -d
/reference/alphafold/20221206 -t 2022-12-06 -n 24 -g true -f ./T1024.fasta -o
/scratch/ulambda/alphafold/runtest
```

# Alphafold without containers

Fans of Conda may also wish to check out https://github.com/kalininalab/alphafold_non_docker. Just make sure to `module load gcc miniconda3` rather than following the exact procedure!

---

Créé 2021-07-20 15:31:52 UTC par Ewan Roche
Mis à jour 2024-12-17 07:15:10 UTC par Emmanuel Jeanvoine