AlphaFold

Overview

AlphaFold is an AI system developed by DeepMind that predicts a protein’s 3D structure from its amino acid sequence. The source code for the inference pipeline can be found on their github page.

Both CPU and GPU compute versions have been installed on Viking, and are made available through the module system. **Since a few tweaks have been made to the installation, it is important to read through the following documentation before running any jobs with AlphaFold.**

Video Introduction to Alphafold and Viking

If you are new to Viking and Alphafold you can also watch this video (password:4.#L5jdx) which will take you through the basics on using Alphafold on Viking and an introduction on Alphafold.

A Viking's take on AlphaFold: Protein Structure Prediction at York
Host: Prof Tony Wilkinson
Speakers: Jon Agirre (YSBL) and Emma Barnes (IT Services)

The artificial intelligence system AlphaFold 2 has recently produced a much heralded solution to what is known as the Protein Folding Problem - one of Molecular
Biology's Grand Challenges of more than 50 years standing. In this seminar directed at researchers without specialist knowledge of computing, Jon Agirre (YSBL) will give a practical overview and assessment of AlphaFold 2. Together with Emma Barnes (IT Services), he will explain how AlphaFold can be accessed by Biological Scientists wishing to generate and interpret their own structural models.

Loading the AlphaFold software module

For each release of AlphaFold installed on Viking, two different modules may be provided - one optimised for CPU-based computing, and one for GPU computing.

CPU-only: use **one** of the following:

```
module load bio/AlphaFold/2.0.0-foss-2020b
```

GPU: use **one** of the following:

```
module load bio/AlphaFold/2.0.0-fosscuda-2020b
module load bio/AlphaFold/2.1.1-fosscuda-2020b
```

AlphaFold databases

AlphaFold currently requires various genetic databases to be available: UniRef90, MGnify, BFD, Uniclust30, PDB70, PDB.

To avoid needless duplication of large databases across the cluster, these have been made available in a central directory:

```
/mnt/bb/striped/alphafold_db/20210908
```

The name of the subdirectory (20210908) here indicates the date when the databases were downloaded. The files are hosted on the burst buffer (`` `/mnt/bb` ``) - a shared filesystem powered by fast SSDs - which is recommended for AlphaFold due to the random I/O access patterns (in test jobs, we have observed up to 2x slowdown when using the disk-based lustre filesystem `` `/mnt/lustre` `` instead of `` `/mnt/bb` ``).

Modifications to running AlphaFold

It is important to note that we have made a few enhancements to the installation to facilitate easier usage:

- The location to the AlphaFold data can be specified via the `` `$ALPHAFOLD_DATA_DIR` `` environment variable, so you should define this variable in your AlphaFold job script: **`` `export ALPHAFOLD_DATA_DIR=/mnt/bb/striped/alphafold-db/20210908` ``**
- A symbolic link named `` `alphafold` `` , which points to the `` `run_alphafold.py script` `` , is included. **This means you can just use `` `alphafold` ``** instead of `` `run_alphafold.py` `` or `` `python run_alphafold.py` `` .
- The `` `run_alphafold.py` `` script has been slightly modified such that **defining `` `$ALPHAFOLD_DATA_DIR` `` is sufficient** to pick up all the data provided in that location, meaning that you don't need to use options like `` `--data_dir` `` to specify the location of the data.
- Similarly, the `` `run_alphafold.py` `` script was tweaked such that the location to commands like `` `hhblits`/`hhsearch`/`jackhmmer`/`kalign` `` are already correctly set, and thus **options like `` `--hhblits_binary_path` `` are not required**.
- The Python script that are used to run `` `hhblits` `` and `` `jackhmmer` `` have been tweaked so you can control how many cores are used for these tools (rather than hard-coding this to 4 and 8 cores respectively).
  - If set, the `` `$ALPHAFOLD_HHBLITS_N_CPU` `` environment variable can be used to specify how many cores should be used for running `` `hhblits` ``. **The default of 4 cores will be used if `` `$ALPHAFOLD_HHBLITS_N_CPU` `` is not defined**
  - likewise for `` `jackhmmer` `` and `` `$ALPHAFOLD_JACKHMMER_N_CPU` `` .
  - Tweaking either of these may not be worth it however, since test jobs indicated that using more than 4/8 cores actually resulted in **worse** performance (although this may be workload dependent)

CPU- versus GPU performance

Using the T1050.fasta example mentioned in the AlphaFold README, we have seen the following runtimes (using `` `--preset=full_dbs` ``):

CPU cores	GPUs	Runtime (HH:MM:SS) on /mnt /bb	Runtime (HH:MM:SS) on /mnt /lustre
8	-	> 24:00:00	22:16:07
16	-	15:37:54	21:35:56
20	-	17:11:14	17:40:30
40	-	17:59:13	21:20:14
10	1	02:28:37	04:58:51
20	2	02:21:49	03:22:28

This highlights the importance of the resources requested when running AlphaFold. These tests suggest that:

- almost all jobs saw faster runtimes when using the AlphaFold database stored on the burst buffer, `` `/mnt/bb` ``
- GPU jobs are **significantly** faster than CPU (~6x as quick)
- Multi-GPU performance is not noticeably better than single-GPU
- Counter-intuitively, **using more CPUs can result in longer runtimes**!

Example job scripts

The following example scripts were used to run the CPU and GPU benchmarks above. To modify these for your own needs, ensure that you update:

- `--fasta_paths` to point to the fasta file(s) to be processed
- `--max_template_date`
- `--preset` to the preset to use (reduced_dbs, full_dbs or casp14)
- `--output_dir` to the path where output files should be written (`$PWD` will use the current directory)
- `--model_names`

The walltime and memory requests are based on the observed job efficiency data when using the T1050. fasta file with `` `--preset=full_dbs` ``. These will likely need to be increased for other fasta input files, or when running with a different preset.

Important note.

For later versions of Alphafold you may need to update the flags.

API changes between v2.0.0 and v2.1.0

We tried to keep the API as much backwards compatible as possible, but we had to change the following:

- The `RunModel.predict()` now needs a `random_seed` argument as MSA sampling happens inside the Multimer model.
- The `preset` flag in `run_alphafold.py` and `run_docker.py` was split into `db_preset` and `model_preset`.
- The models to use are not specified using `model_names` but rather using the `model_preset` flag. If you want to customize which models are used for each preset, you will have to modify the the `MODEL_PRESETS` dictionary in `alphafold/model/config.py`.
- Setting the `data_dir` flag is now needed when using `run_docker.py`.

CPU job for AlphaFold/2.0.0-foss-2020b

AlphaFold-CPU-example.job

```
#!/usr/bin/env bash

#SBATCH --job-name=alphafold-cpu-test
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=16
#SBATCH --mem=80G
#SBATCH --time=24:00:00
#SBATCH --output=%x-%j.log


# Load AlphaFold module
module load bio/AlphaFold/2.0.0-foss-2020b

# Path to genetic databases
export ALPHAFOLD_DATA_DIR=/mnt/bb/striped/alphafold_db/20210908/

# Optional: uncomment to change number of CPU cores to use for hhblits/jackhmmer
# export ALPHAFOLD_HHBLITS_N_CPU=8
# export ALPHAFOLD_JACKHMMER_N_CPU=8

# Run AlphaFold
alphafold --fasta_paths=T1050.fasta --max_template_date=2020-05-14 --preset=full_dbs --output_dir=$PWD --
model_names=model_1,model_2,model_3,model_4,model_5
```

GPU example job AlphaFold/2.0.0-foss-2020b

AlphaFold-CPU-example.job

```
#!/usr/bin/env bash

#SBATCH --job-name=alphafold-gpu-test
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=10
#SBATCH --gres=gpu:1
#SBATCH --partition=gpu
#SBATCH --time=4:00:00
#SBATCH --output=%x-%j.log

# Load AlphaFold module
module load bio/AlphaFold/2.0.0-fosscuda-2020b

# Path to genetic databases
export ALPHAFOLD_DATA_DIR=/mnt/bb/striped/alphafold_db/20210908/

# Optional: uncomment to change number of CPU cores to use for hhblits/jackhmmer
# export ALPHAFOLD_HHBLITS_N_CPU=8
# export ALPHAFOLD_JACKHMMER_N_CPU=8

# Run AlphaFold
alphafold --fasta_paths=T1050.fasta --max_template_date=2020-05-14 --preset=full_dbs --output_dir=$PWD --
model_names=model_1,model_2,model_3,model_4,model_5
```