AlphaFold3

1. Introduction

This package provides an implementation of the inference pipeline of AlphaFold 3. See below for how to access the model parameters. You may only use AlphaFold 3 model parameters if received directly from Google. Use is subject to these terms of use.

Any publication that discloses findings arising from using this source code, the model parameters or outputs produced by those should cite the Accurate structure prediction of biomolecular interactions with AlphaFold 3 paper.

Please also refer to the Supplementary Information for a detailed description of the method.

AlphaFold 3 is also available at alphafoldserver.com for non-commercial use, though with a more limited set of ligands and covalent modifications.

If you have any questions, please contact the AlphaFold team at alphafold@google.com.

1.1 Local Installation

The **CNCA** team prepared a local installation of AlphaPhold3 using a container based on **singularity or (appatainer)** including the *Genetic Database*.

> The *Model Parameters* is not included, users must request access filling a form and comply with the Google DeepMind terms of use at all times.

The local installation provide the **AlphaFold3** version **3.0.1** over a container based on **Ubuntu 24.04** distribution with **cuda-12.6**. The container is already prepared and is used throughtout a wrapper that accept all *run_alphafold.py* options; the *Genetic Database* location is already configured, users should provide the json path, model and output directories location.

The main resource target of **AlphaFold3** is the **GPU** but the application can execute the data stage only on the **CPU** although the performance is substantially worst. The model inference stage requires a GPU, if you submit a job to a partition without a **GPU**, such as the **hpc** or **fct** partitions, then the option ***--norun-inference*** is added automatically to the *run_alphafold.py* script.

The user is free to prepare is own container as explained on https://github.com/google-deepmind/alphafold3/tree/main.

1.2 Environment

The environment is activate with command

```
$ module load alphaphold3/3.0.1
```

this will activate automatically a virtual environment ready to start the **AlphaFold3** container throught the python script *run_alphafold.py*.

The command *run_alphafold.py* is a wrapper to the real script in the container and accept the same options. For example, you can get the *run_alphafold.py* help with command

```
$ run_alphafold.py --helpshort
```

1.3 Data Base Location

The **Genetic Database** is installed only on local disk of workernodes bellow the directory

```
/local/alphafold3
```

The environment variable **ALPHAFOLD3_DATABASE** points to this location.

> The **Genectic Database** is not available on the user interfaces.

2. How to Run

2.1 Choose a working directory

```
$ mkdir ~/alphafold3_test
$ cd ~/alphafold3_test
```

2.2 Obtain the Model Parameters

The model parameters access is subject to Google DeepMind terms of use, please fill the form https://forms.gle/svvpY4u2jsHEwWYS6 and follow the instructions. Once you have access to the model parameters you will be responsible to keep the data private.

2.3 Create a submission script

Choose your favority editor and create a file, for example *alphafold3.sh*, with the following content:

```
$ emacs alphafold3.sh
#!/bin/bash
#SBATCH --partition=gpu
#SBATCH --gres=gpu
#SBATCH --nodes=1
#SBATCH --ntasks=8
#SBATCH --mem=64G

module purge
module load alphafold3
run_alphafold.py \
    --json_path=fold_input.json \
    --model_dir=./models \
    --output_dir=./af_output
```

Create the *models* local directory and copy the model parameters into it:

```
$ mkdir models
```

You can place the model parameters on any other directory you just have to insert the appropriate directory location on the *--model_dir* option.

It is recommended to request at least 64GB of RAM but you can increase if necessary.

The output directory is created automatically if it does not exist.

2.4 Create the input JSON

You can use the example of section **"Installation and Running Your First Prediction"** of page https://github.com/google-deepmind/alphafold3?tab=readme-ov-file.

```
$ emacs fold_input.json
{
"name": "2PV7",
"sequences": [
  {
    "protein": {
      "id": ["A", "B"],
         "sequence"
"GMRESYANENQFGFKTINSDIHKIVIVGGYGKLGGLFARYLRASGYPISILDREDWAVAESILANADVVIVSVPINLTLETIERLKPYL
TENMLLADLTSVKREPLAKMLEVHTGAVLGLHPMFGADIASMAKQVVVRCDGRFPERYEWLLEQIQIWGAKIYQTNATEHDHNM
TYIQALRHFSTFANGLHLSKQPINLANLLALSSPIYRLELAMIGRLFAQDAELYADIIMDKSENLAVIETLKQTYDEALTFFENNDRQG
FIDAFHKVRDWFGDYSEQFLKESRQLLQQANDLKQG"
    }
  }
],
"modelSeeds": [1],
"dialect": "alphafold3",
"version": 1
}
```

2.5 Submit the job and check status

```
$ sbatch alphafold3.sh
Submitted batch job 22719987

$ squeue
JOBID    PARTITION NAME      USER     ST TIME NODES CPUS TRES_PER_NODE NODELIST
```

```
22719987 gpu       alphafold3 martinsj PD 0:00 1     8    gres/gpu
```

The *squeue* command list shows that the job is pending, or waiting for resources.

2.6 Check the running job

Eventually the job will run and the *squeue* command will show something like:

```
$ squeue
JOBID    PARTITION NAME       USER     ST TIME NODES CPUS TRES_PER_NODE NODELIST
22719987 gpu       alphafold3 martinsj R  1:25 1     8    gres/gpu      hpc060
```

You can get more details about the job with command *pestat*:

```
$ pestat -j 22719987
Select only nodes with jobs in joblist=22719987
Hostname Partition  Node Num_CPU CPUload Memsize Freemem  GRES/       Joblist
                State Use/Tot           (MB)    (MB)  node        JobId User GRES/job ...
hpc060   gpu        mix   8  96   2.05*  512000   28466* gpu:t4:3(S:0-1) 22719987 martinsj
```

2.7 Check the finished job

On completion the job vanish from *squeue* list and you have the output on subdirectory "af_output" and the job standard output will be on file *slurm-2271987.out*:

```
$ ls -l
drwxr-x---+ 2 martinsj csys  4096 out  4 14:08 af_input
drwxr-x---+ 2 martinsj csys  4096 out  4 14:08 models
drwxrwx---+ 4 martinsj csys  4096 out  4 16:10 af_output
-rw-rw----+ 1 martinsj csys   424 out  4 14:29 alphafold3.sh
-rw-rw----+ 1 martinsj csys 46591 out  4 16:11 slurm-22719987.out

$ ls -l af_output/2PV7/
-rw-rw----+ 1 martinsj csys 3118163 Oct  4 16:11 2PV7_confidences.json
-rw-rw----+ 1 martinsj csys 7301275 Oct  4 16:11 2PV7_data.json
-rw-rw----+ 1 martinsj csys  409731 Oct  4 16:11 2PV7_model.cif
-rw-rw----+ 1 martinsj csys     147 Oct  4 16:11 2PV7_ranking_scores.csv
-rw-rw----+ 1 martinsj csys     332 Oct  4 16:11 2PV7_summary_confidences.json
drwxrwx---+ 2 martinsj csys    4096 Oct  4 16:11 seed-1_sample-0
drwxrwx---+ 2 martinsj csys    4096 Oct  4 16:11 seed-1_sample-1
drwxrwx---+ 2 martinsj csys    4096 Oct  4 16:11 seed-1_sample-2
```

```
drwxrwx---+ 2 martinsj csys    4096 Oct  4 16:11 seed-1_sample-3
drwxrwx---+ 2 martinsj csys    4096 Oct  4 16:11 seed-1_sample-4
-rw-rw----+ 1 martinsj csys   13036 Oct  4 16:11 TERMS_OF_USE.md
```

2.8 Get some job statistics

You can get some job statistics with commands:

```
$ seff 22719987
Job ID: 22719987
Cluster: production
User/Group: martinsj/csys
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 8
CPU Utilized: 01:03:58
CPU Efficiency: 44.01% of 02:25:20 core-walltime
Job Wall-clock time: 00:18:10
Memory Utilized: 6.20 GB
Memory Efficiency: 6.46% of 96.00 GB
```

and

```
$ sacct -j 22719987
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
22719987     alphafold+        gpu       csys         8  COMPLETED      0:0
22719987.ba+      batch                  csys         8  COMPLETED      0:0
22719987.ex+     extern                  csys         8  COMPLETED      0:0
```

You can obtain more details with command

```
$ macct -j 22719987
```

3. Permance

It is recommended to request a *NVIDIA A100 80GB* GPU or a GPU with Compute Capatibily of *8.0* at least. The container is built with the *NVIDIA A100 80GB* in mind.

The **CIRRUS** provide three GPU types:

- NVIDIA A100 80GB
- NVIDIA Tesla V100s 32GB
- NVIDIA Tesla T4 16GB

The best performance will be obtained with the *NVIDIA A100 80GB* controller but this is the most requested resource and users may have to wait days to obtain one. The other controllers are less performant but more available and it may compensate to use the *Tesla* GPU's. As a reference, the example above completation times for each GPU was the following:

Controller	Completion Time (hh:mm:ss)
NVIDIA A100	18:10
NVIDIA Tesla V100s	2:19:31
NVIDIA Tesla T4	3:02:37

Users may request a specific controller modifying the option **SBATCH --gres=gpu** on the script to:

- #SBATCH --gres=gpu:a100
- #SBATCH --gres=gpu:v100s
- #SBATCH --gres=gpu:t4

to request the *A100*, or the *Tesla V100s*, or the *Tesla T4*, respectively. The *Tesla* GPU's controllers Compute Capabilities are less than *8.0* but it is possible to run *alphafold3*. The *run_alphafold.py* wrapper insert the right options on the container *run_alphafold.py* script in order to execute, although with reduced performance, on the *Tesla* GPU's.

4. References

- AlphaFold3
- Installation and Running Your First Prediction
- Request to access model parameters for AlphFold3

Revision #29
Created 9 October 2025 09:45:01 by João Paulo Martins
Updated 9 October 2025 14:56:34 by João Paulo Martins