AlphaFold

1. Introduction
    1. Environment
    2. Data Base Location
    3. run_udocker.py
2. How to Run
    1. Example on Partition "gpu"
    2. Example on Partition "fct"
    3. Example on Partition "hpc"
    4. sbatch Options
3. Benchmarks
4. References

1. Introduction

The **CNCA** team prepared a local installation of AlphaPhold using a container based on **UDOCKER** (instead of **DOCKER**) and includes the *Genetic Database*.

The local installation provide the **AlphaFold** version **2.1.1** over a container based on **Ubuntu 18.04** distribution with **cuda-11.0** and **cudnn-8**.

The main resource target of **AlphaFold** is the **GPU** but the application also execute only on the **CPU** although the performance is substantially worst, see the Benchmarks section bellow.

1.1 Environment

The environment is activate with command

```
$ module load udocker/alphaphold/2.1.1
```

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

1.2 Data Base Location

The **Genetic Database** is installed bellow the filesystem directory

```
/users3/data/alphafold
```

on read-only mode, upgrades may be requested using the support: helpdesk (at) acnca.pt

1.3 run_udocker.py Script

The **run_udocker.py** script was adapted from the **run_docker.py** script normally used by **AlphaFold** with the **docker** container technology.

The **run_udocker.py** accept the same options as the **run_docker.py** script with a few minor changes that we hope it will facilitate user interaction. The user may change the script behavour throught environment variables or command line options, we can see only the changes bellow:

Optional environment variables:

Variable Name	Default Value	Comment
DOWNLOAD_DIR	none	Genetic database location (absolute path)
OUPTPUT_DIR	none	Output results directory (absolute path)

Command line options:

Command Option	Mandatory	Default Value	Comment
--data_dir	no	**/local/alphafold** or **/users3/data/alphafold**	Genetic database location, takes precedence over DOWNLOAD_DIR when both are selected
--output_dir	no	<working_dir>/output	Absolute path to the results directory, takes precedence over OUTPUT_DIR when both are selected

> The option **--data_dir** is required on the standard AlphaFold **run_docker.py** script, we choose to select automatically the location of the **genetic database** but the user may change this path throught the environment variable **DOWNLOAD_DIR** or the command line option **--data_dir**. When possible, we provide a local copy to the workernodes of the database directory in order to improve job performance.

> The AlphaFold standard output results directory location is **/tmp/alphafold** by default, please note that we change this location to the local working directory, the user can select a different path throught the environment variable **OUTPUT_DIR** or the command line option **--output_dir**.

2. How to Run

We only need a protein and a submition script, if we analyze multiple proteins on parallel it is advise to submit then from different directory in order to avoid interference between runs.

2.1 Example on Partition "gpu"

Lets analyze the https://www.uniprot.org/uniprot/P19113 protein, for example.

Create a working directory and get the protein:

```
[user@cirrus ~]$ mkdir run_P19113
[user@cirrus ~]$ cd run_P19113
[user@cirrus run_P19113]$ wget -q https://www.uniprot.org/uniprot/P19113.fasta
```

Use your favority editor the create the submition script **submit.sh***:

```
[user@cirrus run_P19113]$ emacs submit.sh
#!/bin/bash
# ------------------------------------------------------------------------------
#SBATCH --job-name=P19113
#SBATCH --partition=gpu
#SBATCH --mem=50G
#SBATCH --ntasks=4
#SBATCH --gres=gpu
# ------------------------------------------------------------------------------
module purge
module load udocker/alphafold/2.1.1
run_udocker.py --fasta_paths=P19113.fasta --max_template_date=2020-05-14
```

Finally, submit your job, check if it is running and wait for it:

```
[user@cirrus run_P19113]$ sbatch submit.sh
[user@cirrus run_P19113]$ squeue
```

When finish the local directory **./output** will have the analyze results.

2.2 Example on Partition "fct"

```
[user@cirrus run_P19113]$ emacs submit.sh
#!/bin/bash
# ------------------------------------------------------------------------------
#SBATCH --job-name=P19113
#SBATCH --partition=fct
#SBATCH --qos=<qos>
#SBATCH --account=<account>  # optional on most cases
#SBATCH --mem=50G
#SBATCH --ntasks=4
#SBATCH --gres=gpu
# ------------------------------------------------------------------------------
module purge
module load udocker/alphafold/2.1.1
run_udocker.py --fasta_paths=P19113.fasta --max_template_date=2020-05-14
```

2.3 Example on Partition "hpc"

```
[user@cirrus run_P19113]$ emacs submit.sh
#!/bin/bash
# ------------------------------------------------------------------------------
#SBATCH --job-name=P19113
#SBATCH --partition=hpc
#SBATCH --mem=50G
#SBATCH --ntasks=4
# ------------------------------------------------------------------------------
module purge
module load udocker/alphafold/2.1.1
run_udocker.py --fasta_paths=P19113.fasta --max_template_date=2020-05-14
```

2.4 sbatch Options

--partition=XX

The best job performance is achivied on the **gpu** or **fct** partitions, the later is restricted to users with a valid **QOS**.

The **alphafold** and also run on the **hpc** partition but is this case it will use only a slower **CPU** and there is no **GPU** available, the total run time is roughly eight times greather when compared to jobs executed on the **gpu** or **fct** partitions.

--mem=50G

The default job memory allocation per cpu depends on the used partition but it may be insuficient, we recommend you to request **50GB** of memory, the benchmarks sugest this value should be enough on all cases.

--ntasks=4

Apparentelly this is the maximum number of tasks needed by the application, we didn't get any noticible improvement when rising this parameter.

--gres=gpu

The partitions **gpu** and **fct** provide up to eight **GPUs**. The application was built for compute using **GPU**, there is no point is requesting more than one **GPU**, we didn't notice any improvement on the total run time. We also notice that the total compute time for both types of available **GPUs** is similar.

The **alphafold** also run only on **CPU** but the total run time increase substantial, as seen on benchmarks results bellow.

4. Benchmarks

We made some benchmarks with the protein *P19113* in order to help users organizing their work.

The results bellow sugest that the best choice would be use four **CPU** tasks, one **GPU** and let the system select the local copy of the *genetic data base* on the workernodes.

Since a **GPU** run takes roughly two hours and half then users may run up to thirty five protein analyzes in one submit job, as long they are executed in sequence.

Partition	CPU	#CPU	GPU	#GPU	#JOBS	DOWNLOAD_DIR	ELAPSED_TIME
gpu/fct	EPYC_7552	4	Tesla_T4	1	1	/local/alphafold	02:22:19
gpu/fct	EPYC_7552	4	Tesla_V100S	1	1	/local/alphafold	02:38:21
gpu/fct	EPYC_7552	4	Tesla_T4	2	1	/local/alphafold	02:22:25

gpu/fct	EPYC_7552	4	Tesla_T4	1	1	/users3/data/alphafold	15:59:50
gpu/fct	EPYC_7552	4	Tesla_V100S	1	1	/users3/data/alphafold	11:40:04
gpu/fct	EPYC_7552	4	Tesla_T4	2	1	/users3/data/alphafold	14:58:52
gpu/fct	EPYC_7552	4		0	1	/local/alphafold	16:17:32
gpu/fct	EPYC_7552	4		0	1	/users3/data/alphafold	18:22:07
gpu/fct	EPYC_7552	4		0	4	/local/alphafold	17:53:25
hpc	EPYC_7501	4		0	3	/local/alphafold	21:44:35
hpc	EPYC_7501	32		0	1	/local/alphafold	16:35:59
hpc	EPYC_7501	4		0	1	/users3/data/alphafold	1-02:28:33
hpc	EPYC_7501	16		0	1	/users3/data/alphafold	1-03:42:23
hpc	EPYC_7501	32		0	1	/users3/data/alphafold	1-03:15:19

5. References

- https://github.com/deepmind/alphafold
- https://github.com/indigo-dc/udocker
- https://www.docker.com
- https://www.uniprot.org/uniprot
- https://www.uniprot.org/uniprot/P19113

Revision #48
Created 2021-11-29 14:16:40 UTC by João Paulo Martins
Updated 2026-07-10 15:53:51 UTC by Jorge Gomes