HIGH PERFORMANCE RESEARCH COMPUTING

Introduction to AlphaFold for 3D Protein Structure Prediction on Grace

High Performance
Research Computing
DIVISION OF RESEARCH

Spring 2023

AlphaFold for 3D Protein Structure Prediction on Grace

- Resources and Limitations
- Database Files
- Running AlphaFold
  - Google Colab
  - ChimeraX + Google Colab
  - Grace GPU or non-GPU nodes
- Visualization of Results
  - job resource usage
  - view predictions in ChimeraX
  - plotting pLDDT values
- Alternative Workflows

https://hprc.tamu.edu/wiki/SW:AlphaFold

Resource Limitations

- AlphaFold
  - currently AlphaFold can only utilize one GPU
  - about 90% of processing is done on CPU when using DeepMind's workflow
- AlphaFold on HPRC Grace
  - sometimes GPU not detected on certain nodes
- AlphaFold in Google Colab (web browser or ChimeraX app)
  - no guarantee of available resources in Colab
  - runs as a Jupyter notebook on Google Colab cloud servers
    - 12GB RAM max
    - a notebook can run for up to 12 hours per day
      - 24 hours per day with Colab Pro ($9.99/month)
    - not suitable for large predictions

AlphaFold Databases on Grace

/scratch/data/bio/alphafold/2.2.0

```
├── bfd
│   ├── [1.4T]  bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt_a3m.ffdata
│   ├── [1.7G]  bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt_a3m.ffindex
│   ├── [ 16G]  bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt_cs219.ffdata
│   ├── [1.6G]  bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt_cs219.ffindex
│   ├── [304G]  bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt_hhm.ffdata
│   └── [124M]  bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt_hhm.ffindex
├── mgnify
│   └── [ 64G]  mgy_clusters_2018_12.fa
├── mgnify_2019_05
│   └── [271G]  mgy_proteins.fa
├── params
│   ├── [ 18K]  LICENSE
│   ├── [356M]  params_model_1_multimer_v2.npz
│   ├── [356M]  params_model_1.npz
│   ├── [356M]  params_model_1_ptm.npz
│   ├── [356M]  params_model_2_multimer_v2.npz
│   ├── [356M]  params_model_2.npz
│   ├── [356M]  params_model_2_ptm.npz
│   ├── [356M]  params_model_3_multimer_v2.npz
│   ├── [354M]  params_model_3.npz
│   ├── [355M]  params_model_3_ptm.npz
│   ├── [356M]  params_model_4_multimer_v2.npz
│   ├── [354M]  params_model_4.npz
│   ├── [355M]  params_model_4_ptm.npz
│   ├── [356M]  params_model_5_multimer_v2.npz
│   ├── [354M]  params_model_5.npz
│   └── [355M]  params_model_5_ptm.npz
├── pdb70
│   ├── [ 410]  md5sum
│   ├── [ 53G]  pdb70_a3m.ffdata
│   ├── [2.0M]  pdb70_a3m.ffindex
│   ├── [6.6M]  pdb70_clu.tsv
│   ├── [ 21M]  pdb70_cs219.ffdata
│   ├── [1.5M]  pdb70_cs219.ffindex
│   ├── [3.2G]  pdb70_hhm.ffdata
│   ├── [1.8M]  pdb70_hhm.ffindex
│   └── [ 19M]  pdb_filter.dat
├── pdb_mmcif
│   ├── [9.4M]  mmcif_files
│   └── [141K]  obsolete.dat
├── pdb_seqres
│   └── [218M]  pdb_seqres.txt
├── small_bfd
│   └── [ 17G]  bfd-first_non_consensus_sequences.fasta
├── uniclust30
│   ├── [  87G]  uniclust30_2018_08
│   └── [ 206G]  uniclust30_2021_03
├── uniprot
│   └── [104G]  uniprot.fasta
└── uniref90
    └── [ 62G]  uniref90.fasta
```

total size: 2.6TB
number of files: 183,000+

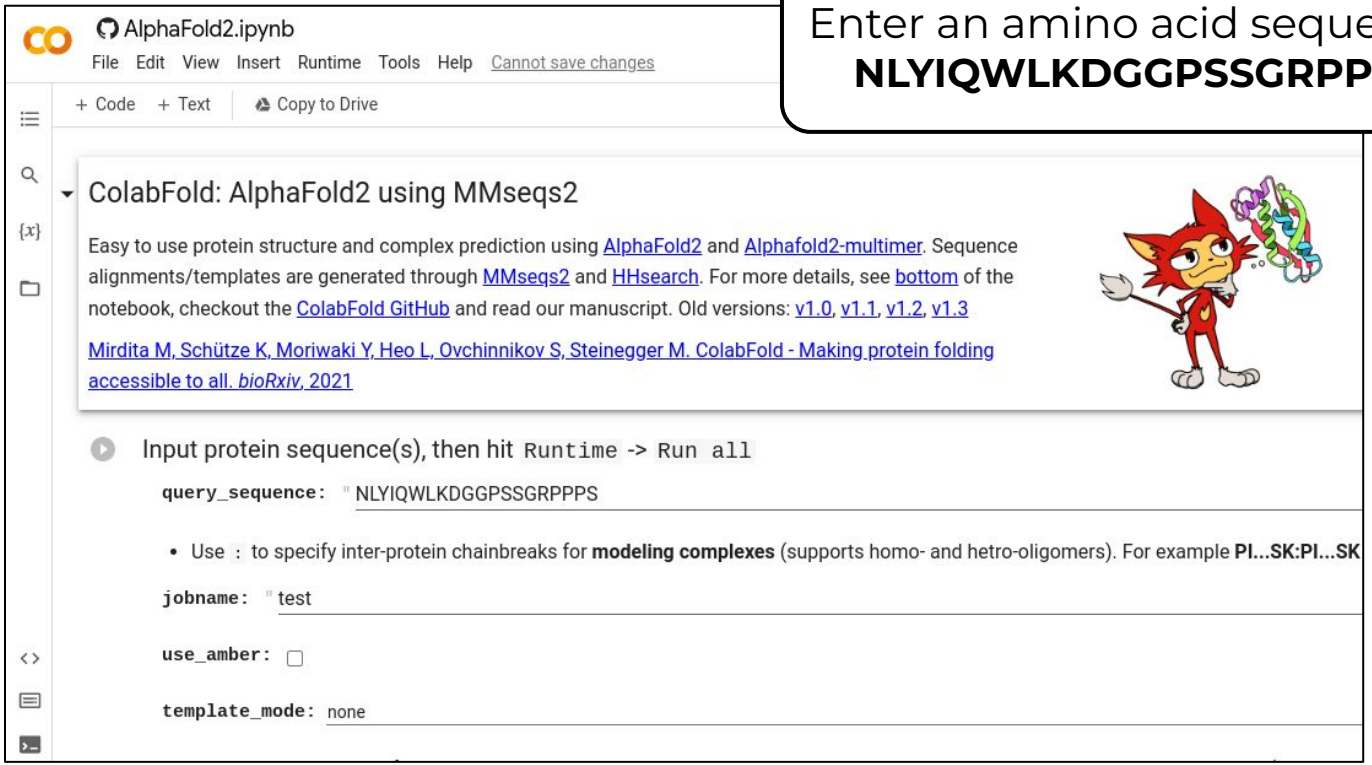
Resources for Running AlphaFold

- Run as a Jupyter Notebook on Google Colab in web browser
- Run as a Jupyter Notebook on Google Colab in ChimeraX
  - ChimeraX Interactive App on HPRC Grace Portal
    - https://portal-grace.hprc.tamu.edu
- Run as a Slurm job script on Grace

https://hprc.tamu.edu/wiki/SW:AlphaFold

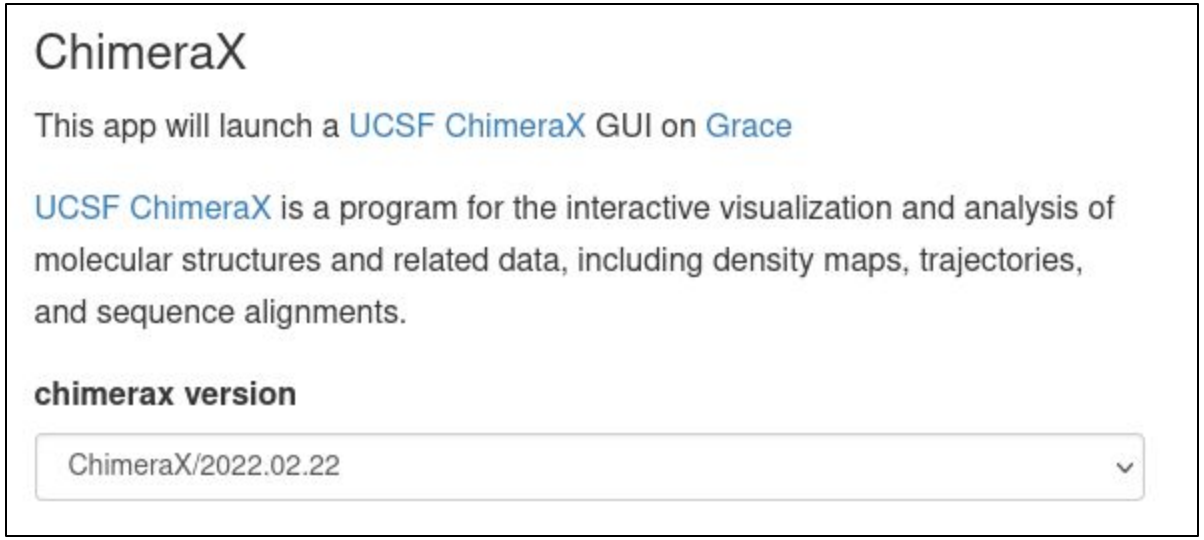
ColabFold AlphaFold2 Jupyter Notebook

Enter an amino acid sequence
NLYIQWLKDGGPSSGRPPPS

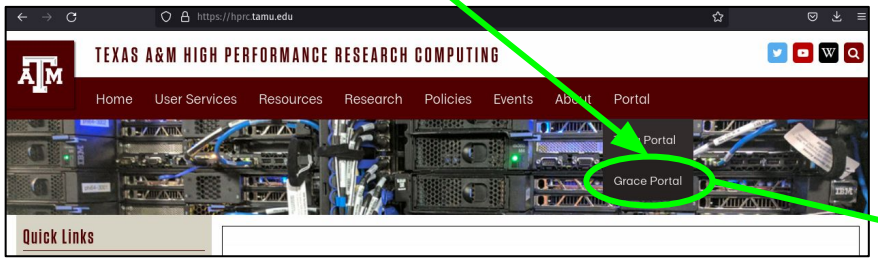


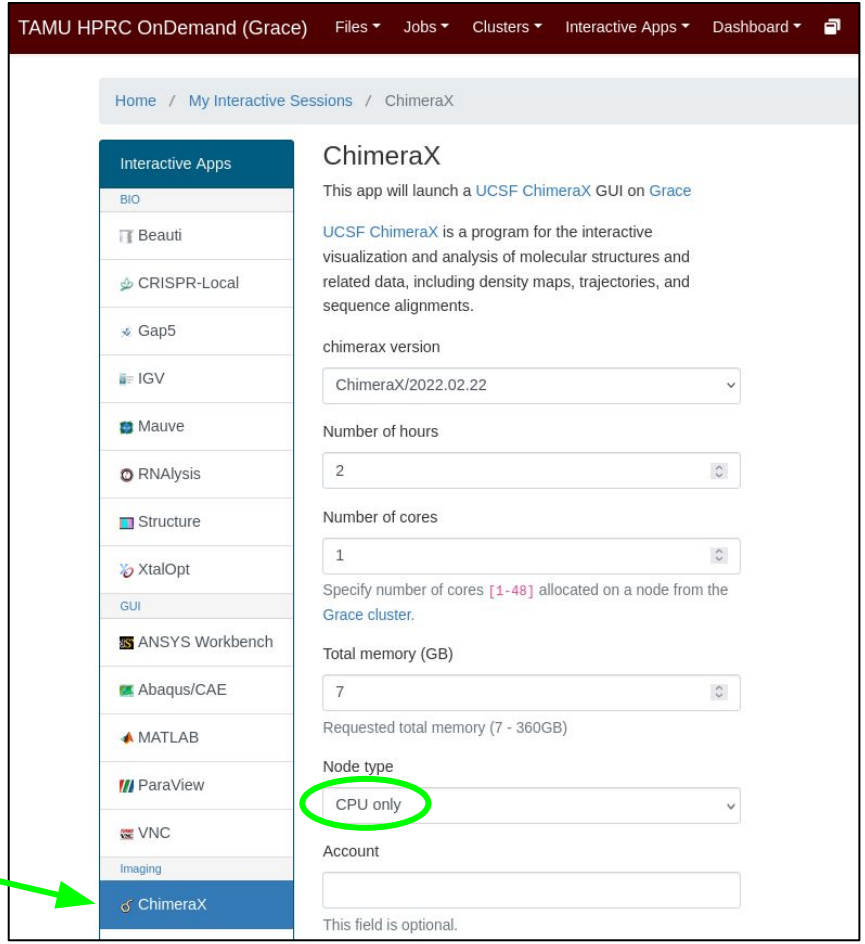
ChimeraX

- Can be used to visualize protein structures
- Can be launched using the Grace portal
  - portal-grace.hprc.tamu.edu
- Can be used to run AlphaFold using the daily build version (2022.02.22+)
  - uses Google Colab with limited resources



- Launch a ChimeraX job on the Grace portal portal-grace.hprc.tamu.edu
  - select Node Type: CPU only
  - AlphaFold runs on Google Colab GPUs so we can use a non-GPU for running ChimeraX
- ChimeraX will be used later for the following
  - run AlphaFold in Google Colab
  - visualize results from an AlphaFold job on Grace





Running AlphaFold on Grace

- Can be run as a job script requesting one GPU
- Shared databases are available: 2.6TB total size

benchmark	monomer_ptm 98 aa	multimer 98 & 73 aa
A100	1 hour 26 minutes	4 hours 49 minutes
RTX 6000	2 hours 39 minutes	4 hours 44 minutes
T4	2 hours 35 minutes	4 hours 45 minutes
CPU only	2 hours 50 minutes	6 hours 11 minutes

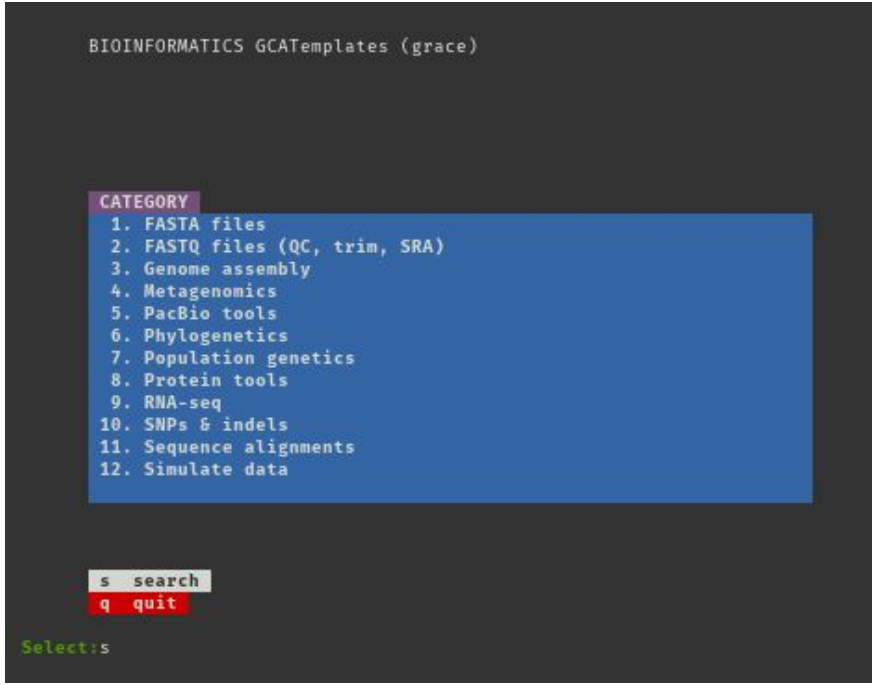
- Can be run in ChimeraX from the Grace portal
  - using the ChimeraX AlphaFold option
  - all processing done on Google cloud servers
    - monomer_ptm (98 aa) on GPU = 1 hour 14 minutes

Finding AlphaFold template job scripts using GCATemplates on Grace

- Genomic Computational Analysis Templates have example input data so you can run the script for demo purposes

```
mkdir $SCRATCH/af2demo
cd $SCRATCH/af2demo
gcatemplates
```

- Type **s** for search then enter **alphafold** to search for the alphafold 2.2.0 template script and select the **reduced_dbs** script
- Review the script and submit the job script which takes about 30 minutes to complete



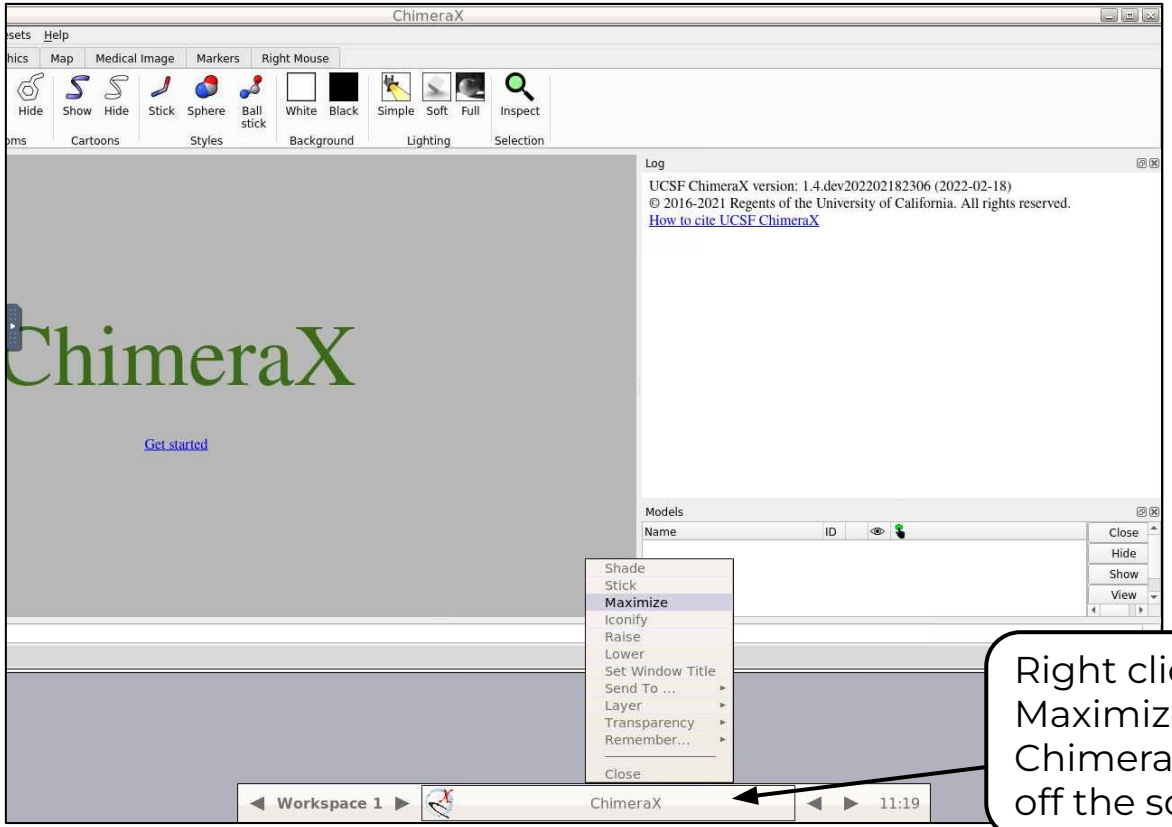

```
sbatch run_alphafold_2.2.0_reduced_dbs_monomer_ptm_grace.sh
```

Monitoring AlphaFold GPUs Found

- Check to make sure that AlphaFold can detect GPUs
  - wait a few minutes after the job starts
  - search for the text "No GPU/TPU found, falling back to CPU."
    - `grep CPU stderr*`
- If the job did not detect GPUs
  - find the compute node name in the NodeList column
    - `sacct -j jobID`
  - cancel the job
    - `scancel jobID`
  - add a line in your job script to ignore the compute node
    - `#SBATCH --exclude=g016`
  - submit your updated job script
  - send an email to the HPRC helpdesk with the node name
    - `help@hprc.tamu.edu`

AlphaFold with ChimeraX + Google Colab

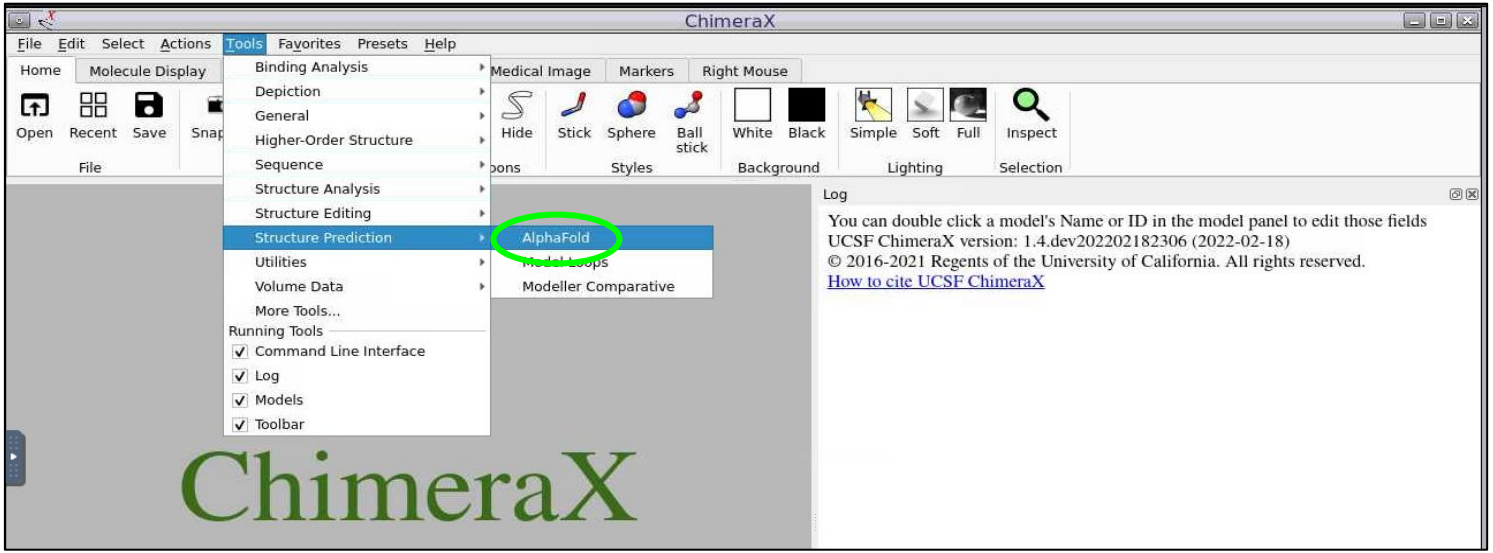
Maximize ChimeraX Window



Right click and select Maximize if the ChimeraX window is off the screen

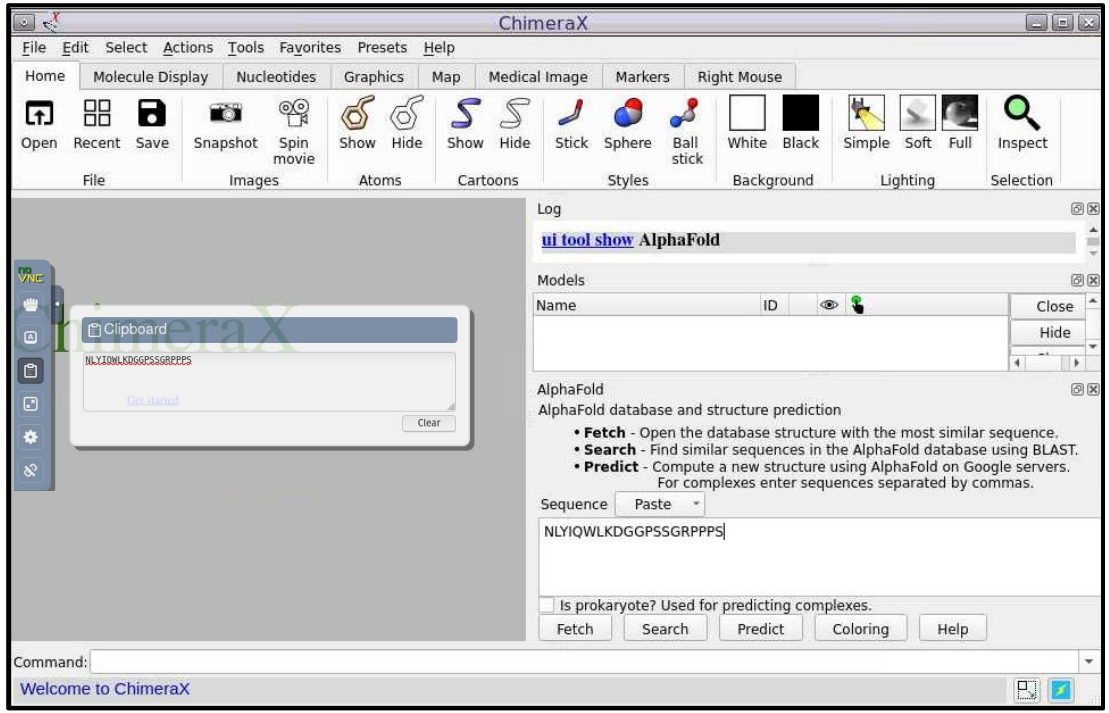
AlphaFold with ChimeraX

- Launch ChimeraX using the HPRC Grace portal
- Select the AlphaFold Structure Prediction option



AlphaFold with ChimeraX

- Enter an amino acid sequence **NLYIQWLKDGGPSSGRPPPS**
  - or paste in Clipboard first then paste in Sequence field
- Click Predict
- A Google Colab page will start and prompt you for your Google login
- Login to your Google account to begin processing
- Prediction completes in about 1 hour
- Not ideal for large prediction jobs



AlphaFold Grace Job Scripts

Example AlphaFold Job Script

- **multimer**
  - dbs in red are required for multimer
- AlphaFold can only use one GPU so reserve half the CPU and memory resources so another job can use the other GPU
  - Grace compute nodes have 360GB of available memory and 48 cores

```
#!/bin/bash
#SBATCH --job-name=alphafold        # job name
#SBATCH --time=2-00:00:00           # max job run time dd-hh:mm:ss
#SBATCH --ntasks-per-node=1         # tasks (commands) per compute node
#SBATCH --cpus-per-task=24          # CPUs (threads) per command
#SBATCH --mem=180G                  # total memory per node
#SBATCH --gres=gpu:a100:1           # request 1 A100 GPU
#SBATCH --output=stdout.%x.%j       # save stdout to file
#SBATCH --error=stderr.%x.%j        # save stderr to file

module purge

export SINGULARITYENV_TF_FORCE_UNIFIED_MEMORY=1
export SINGULARITYENV_XLA_PYTHON_CLIENT_MEM_FRACTION=4.0

DOWNLOAD_DIR=/scratch/data/bio/alphafold/2.2.0

# run jobstats in the background (&) to monitor cpu and gpu usage
jobstats &

singularity exec --nv /sw/hprc/sw/containers/alphafold/alphafold_2.2.0.sif \
 python /app/alphafold/run_alphafold.py \
 --use_gpu_relax \
 --data_dir=$DOWNLOAD_DIR \
 --uniref90_database_path=$DOWNLOAD_DIR/uniref90/uniref90.fasta \
 --mgnify_database_path=$DOWNLOAD_DIR/mgnify/mgy_clusters_2018_12.fa \
 --uniclust30_database_path=$DOWNLOAD_DIR/uniclust30/uniclust30_2021_03/UniRef30_2021_03 \
 --bfd_database_path=$DOWNLOAD_DIR/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt \
 --model_preset=multimer \
 --pdb_seqres_database_path=$DOWNLOAD_DIR/pdb_seqres/pdb_seqres.txt \
 --uniprot_database_path=$DOWNLOAD_DIR/uniprot/uniprot.fasta \
 --template_mmcif_dir=$DOWNLOAD_DIR/pdb_mmcif/mmcif_files \
 --obsolete_pdbs_path=$DOWNLOAD_DIR/pdb_mmcif/obsolete.dat \
 --max_template_date=2022-1-1 \
 --db_preset=full_dbs \
 --output_dir=out_alphafold \
 --fasta_paths=$DOWNLOAD_DIR/example_data/T1083_T1084_multimer.fasta

# run jobstats to create a graph of cpu and gpu usage for this job
jobstats
```

Example AlphaFold Job Script

- **monomer**
  - dbs in red required for monomer
- **monomer_ptm**
  - will produce pTM scores that can be graphed using AlphaPickle
- AlphaFold can only use one GPU so reserve half the CPU and memory resources so another job can use the other GPU

```
#!/bin/bash
#SBATCH --job-name=alphafold        # job name
#SBATCH --time=2-00:00:00           # max job run time dd-hh:mm:ss
#SBATCH --ntasks-per-node=1         # tasks (commands) per compute node
#SBATCH --cpus-per-task=24          # CPUs (threads) per command
#SBATCH --mem=180G                  # total memory per node
#SBATCH --gres=gpu:a100:1           # request 1 A100 GPU
#SBATCH --output=stdout.%x.%j       # save stdout to file
#SBATCH --error=stderr.%x.%j        # save stderr to file

module purge

export SINGULARITYENV_TF_FORCE_UNIFIED_MEMORY=1
export SINGULARITYENV_XLA_PYTHON_CLIENT_MEM_FRACTION=4.0

DOWNLOAD_DIR=/scratch/data/bio/alphafold/2.2.0

# run jobstats in the background (&) to monitor cpu and gpu usage
jobstats &

singularity exec --nv /sw/hprc/sw/containers/alphafold/alphafold_2.2.0.sif \
 python /app/alphafold/run_alphafold.py \
 --use_gpu_relax \
 --data_dir=$DOWNLOAD_DIR \
 --uniref90_database_path=$DOWNLOAD_DIR/uniref90/uniref90.fasta \
 --mgnify_database_path=$DOWNLOAD_DIR/mgnify/mgy_clusters_2018_12.fa \
 --uniclust30_database_path=$DOWNLOAD_DIR/uniclust30/uniclust30_2021_03/UniRef30_2021_03 \
 --bfd_database_path=$DOWNLOAD_DIR/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt \
 --model_preset=monomer \
 --pdb70_database_path=$DOWNLOAD_DIR/pdb70/pdb70 \
 --template_mmcif_dir=$DOWNLOAD_DIR/pdb_mmcif/mmcif_files \
 --obsolete_pdbs_path=$DOWNLOAD_DIR/pdb_mmcif/obsolete.dat \
 --max_template_date=2022-1-1 \
 --db_preset=full_dbs \
 --output_dir=out_alphafold \
 --fasta_paths=$DOWNLOAD_DIR/example_data/T1083.fasta

# run jobstats to create a graph of cpu and gpu usage for this job
jobstats
```

Example AlphaFold Job Script

- **monomer + reduced_dbs**
- dbs in red required for monomer + reduced_dbs
- small_bfd_database is a subset of BFD and is generated by taking the first non-consensus sequence from every cluster in BFD
- AlphaFold can only use one GPU so reserve half the CPU and memory resources so another job can use the other GPU

```
#!/bin/bash
#SBATCH --job-name=alphafold        # job name
#SBATCH --time=2-00:00:00           # max job run time dd-hh:mm:ss
#SBATCH --ntasks-per-node=1         # tasks (commands) per compute node
#SBATCH --cpus-per-task=24          # CPUs (threads) per command
#SBATCH --mem=180G                  # total memory per node
#SBATCH --gres=gpu:a100:1           # request 1 A100 GPU
#SBATCH --output=stdout.%x.%j       # save stdout to file
#SBATCH --error=stderr.%x.%j        # save stderr to file

module purge

export SINGULARITYENV_TF_FORCE_UNIFIED_MEMORY=1
export SINGULARITYENV_XLA_PYTHON_CLIENT_MEM_FRACTION=4.0

DOWNLOAD_DIR=/scratch/data/bio/alphafold/2.2.0

# run jobstats in the background (&) to monitor cpu and gpu usage
jobstats &

singularity exec --nv /sw/hprc/sw/containers/alphafold/alphafold_2.2.0.sif \
 python /app/alphafold/run_alphafold.py \
 --use_gpu_relax \
 --data_dir=$DOWNLOAD_DIR \
 --uniref90_database_path=$DOWNLOAD_DIR/uniref90/uniref90.fasta \
 --mgnify_database_path=$DOWNLOAD_DIR/mgnify/mgy_clusters_2018_12.fa \
 --small_bfd_database_path=$DOWNLOAD_DIR/small_bfd/bfd-first_non_consensus_sequences.fasta \
 --model_preset=monomer \
 --pdb70_database_path=$DOWNLOAD_DIR/pdb70/pdb70 \
 --template_mmcif_dir=$DOWNLOAD_DIR/pdb_mmcif/mmcif_files \
 --obsolete_pdbs_path=$DOWNLOAD_DIR/pdb_mmcif/obsolete.dat \
 --max_template_date=2022-1-1 \
 --db_preset=reduced_dbs \
 --output_dir=out_alphafold \
 --fasta_paths=$DOWNLOAD_DIR/example_data/T1083.fasta

# run jobstats to create a graph of cpu and gpu usage for this job
jobstats
```

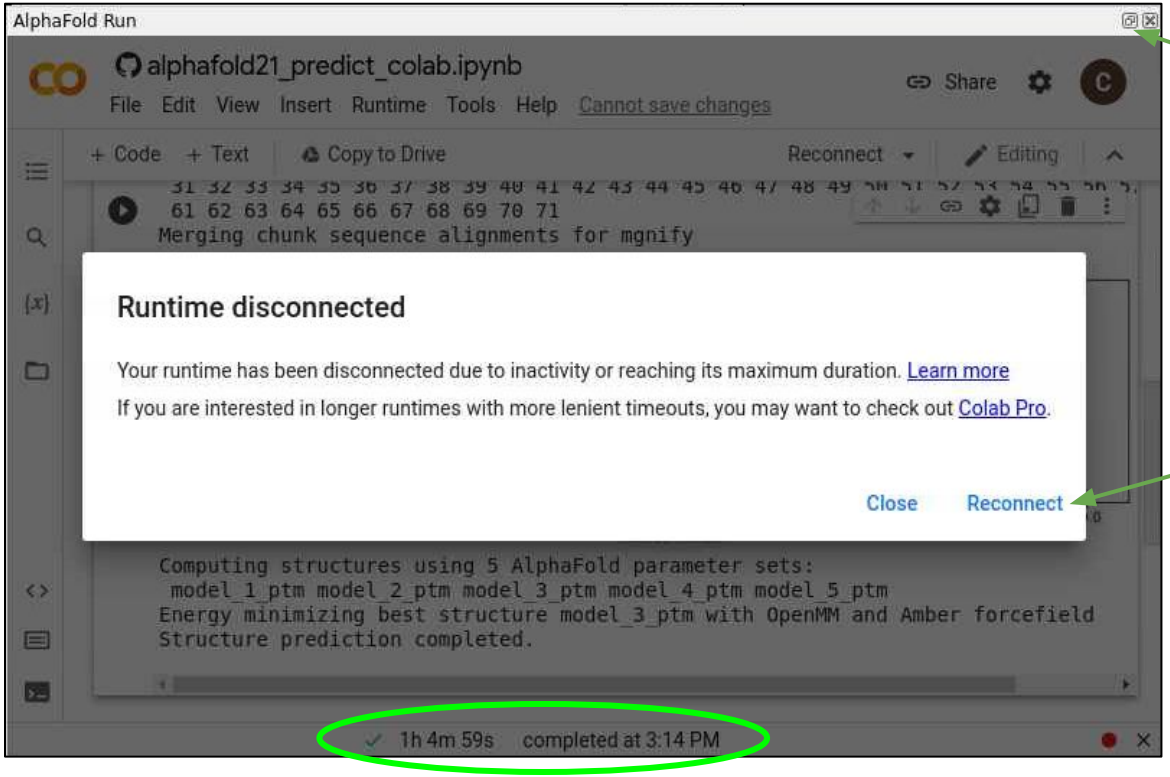
Unified Memory

```
#!/bin/bash
#SBATCH --job-name=alphafold          # job name
#SBATCH --time=2-00:00:00             # max job run time dd-hh:mm:ss
#SBATCH --ntasks-per-node=1           # tasks (commands) per compute node
#SBATCH --cpus-per-task=24            # CPUs (threads) per command
#SBATCH --mem=180G                    # total memory per node
#SBATCH --gres=gpu:a100:1             # request 1 A100 GPU
#SBATCH --output=stdout.%x.%j         # save stdout to file
#SBATCH --error=stderr.%x.%j          # save stderr to file

export SINGULARITYENV_TF_FORCE_UNIFIED_MEMORY=1
export SINGULARITYENV_XLA_PYTHON_CLIENT_MEM_FRACTION=4.0
```

- unified memory can be used to request more than just the total GPU memory for the JAX step in AlphaFold
  - A100 GPU has 40GB memory
  - GPU total memory (40) * `XLA_PYTHON_CLIENT_MEM_FRACTION` (4.0)
  - `XLA_PYTHON_CLIENT_MEM_FRACTION` default = 0.9
- this example script has 160 GB of unified memory
  - 40 GB from A100 GPU + 120 GB DDR from motherboard

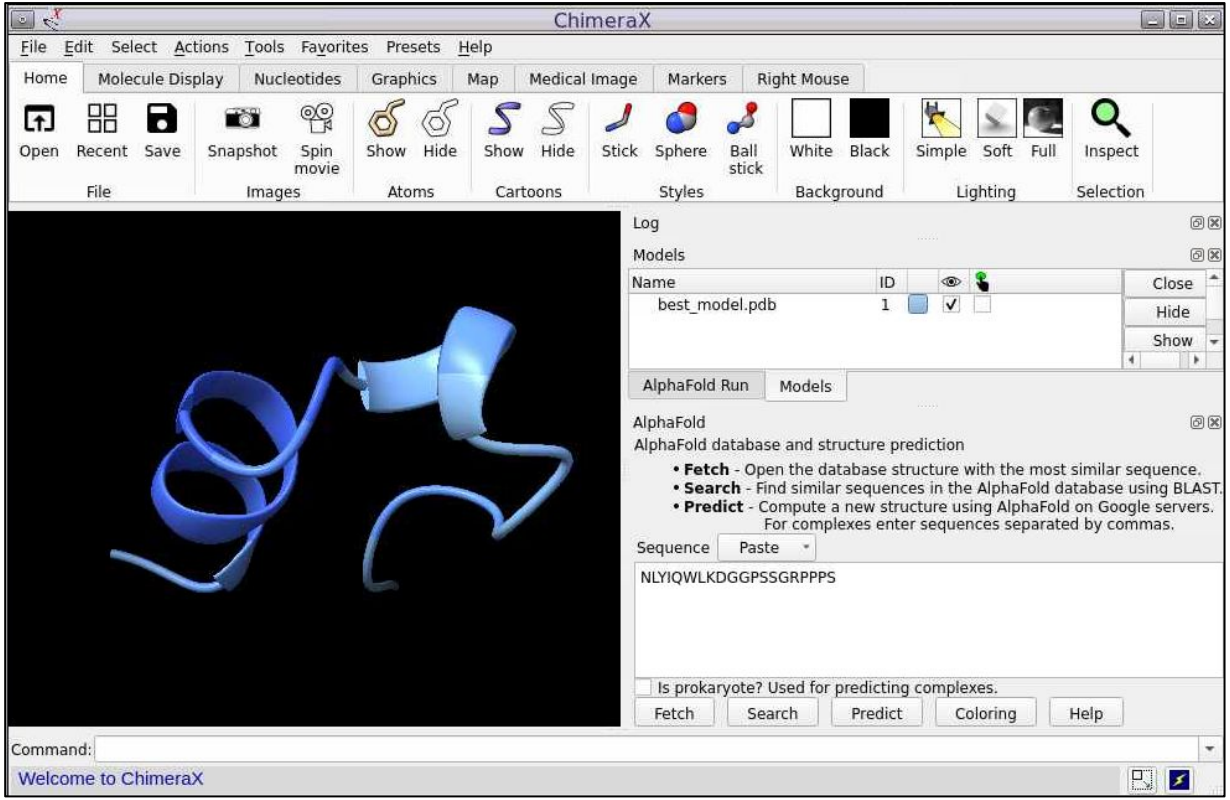
AlphaFold Results Visualization

Visualize Results with ChimeraX

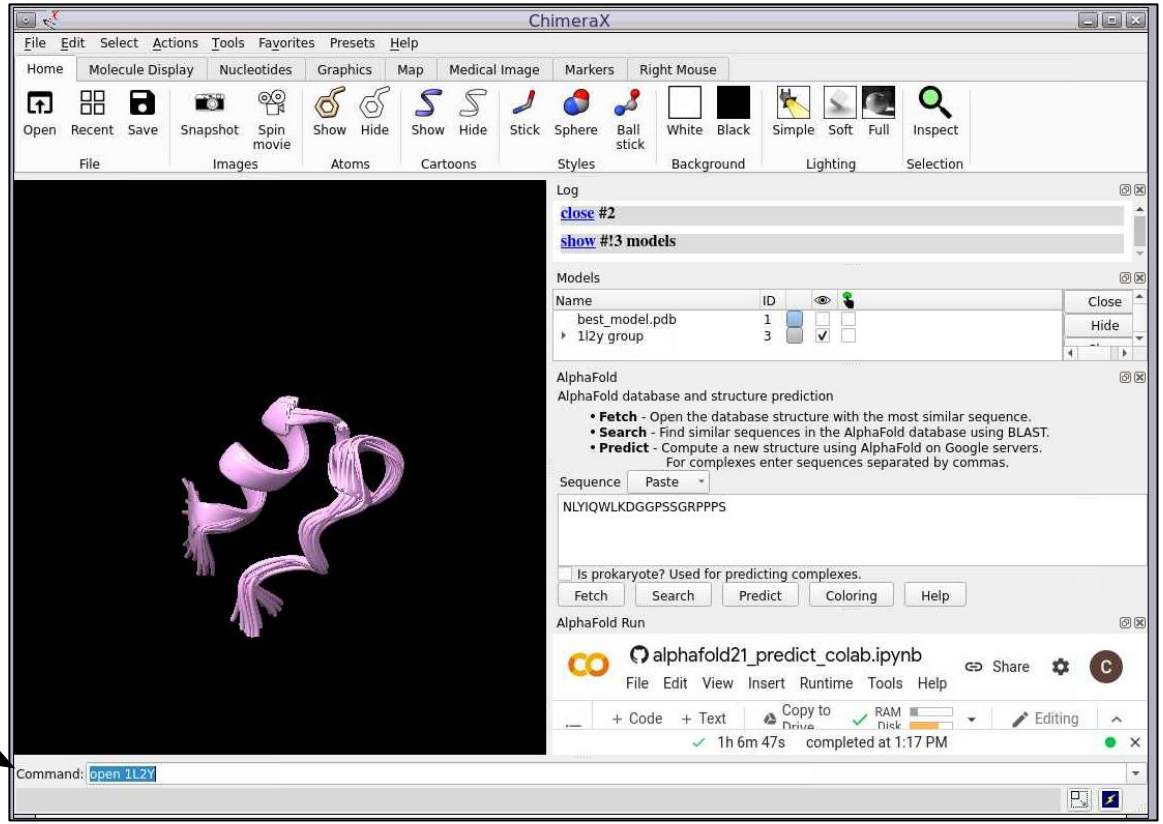


Click the minimize button to return to ChimeraX or click Reconnect then minimize

Visualize AlphaFold Google Colab Results with ChimeraX

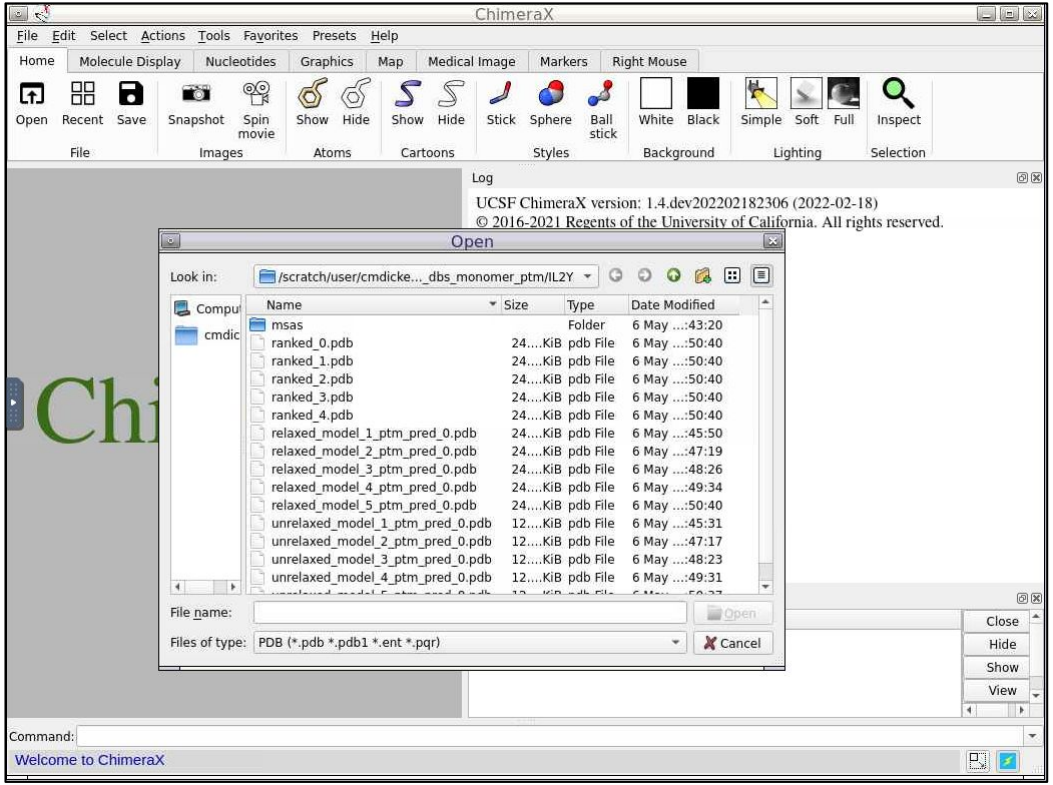


View PDB Structure if Available



type ***open*** and the protein name ***1L2Y***

Visualize AlphaFold Grace Results with ChimeraX



AlphaFold Confidence Metrics

AlphaPickle for Visualization of Confidence Scores

AlphaPickle can be used to create graphs for pLDDT and PAE scores

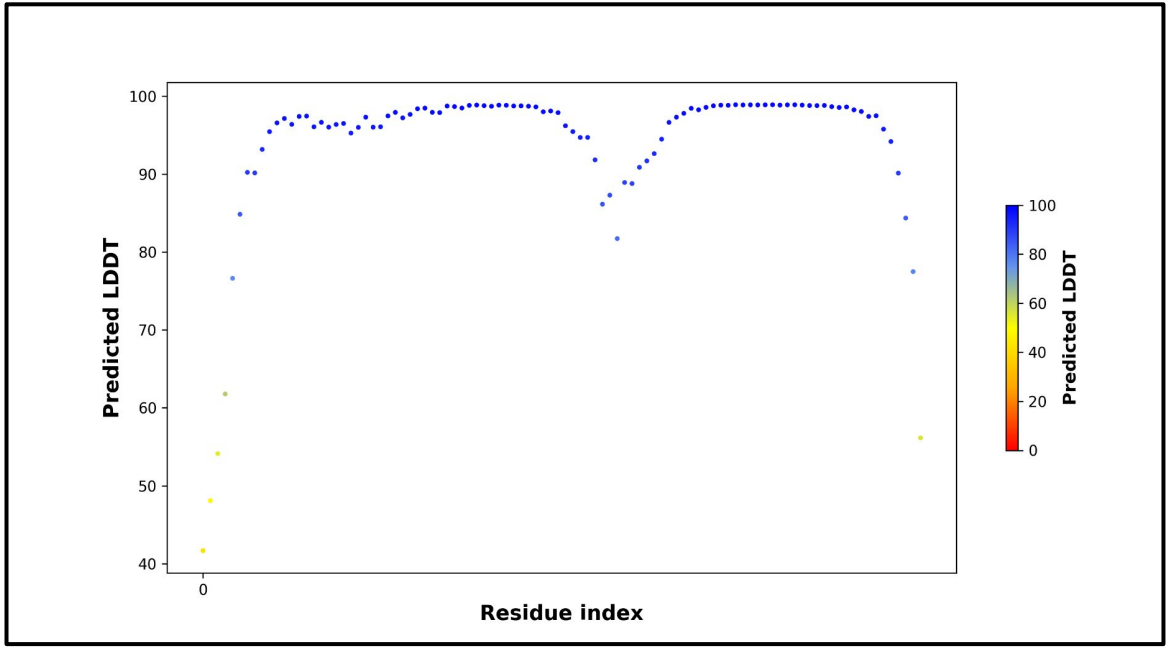
- graphing PAE scores is only available for the **monomer_ptm** and **multimer** model presets
- load the AlphaPickle module at the beginning of the job script
- run AlphaPickle at the end specifying the output directory used in the run_alphafold.py command

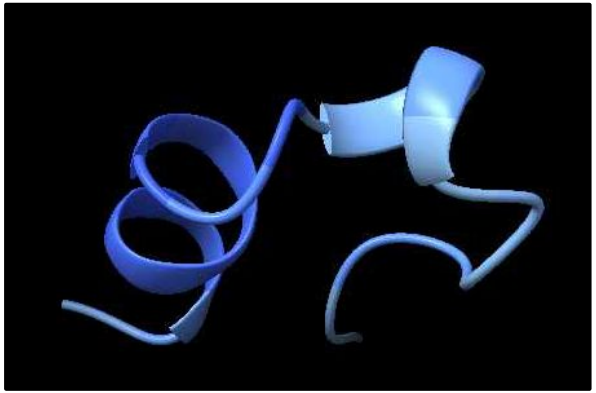
- pLDDT: scale from 0 - 100 of per-residue estimate of prediction confidence
- PAE: Predicted Alignment Error

https://github.com/mattarnoldbio/alphapickle

```
#!/bin/bash
#SBATCH --job-name=alphafold        # job name
#SBATCH --time=2-00:00:00           # max job run time dd-hh:mm:ss
#SBATCH --ntasks-per-node=1         # tasks (commands) per compute node
#SBATCH --cpus-per-task=24          # CPUs (threads) per command
#SBATCH --mem=180G                  # total memory per node
#SBATCH --gres=gpu:a100:1           # request 1 A100 GPU
#SBATCH --output=stdout.%x.%j       # save stdout to file
#SBATCH --error=stderr.%x.%j        # save stderr to file

module purge
module load GCC/10.2.0  CUDA/11.1.1  OpenMPI/4.0.5  AlphaPickle/1.4.1

export SINGULARITYENV_TF_FORCE_UNIFIED_MEMORY=1
export SINGULARITYENV_XLA_PYTHON_CLIENT_MEM_FRACTION=4.0
DOWNLOAD_DIR=/scratch/data/bio/alphafold/2.2.0
# run jobstats in the background (&) to monitor cpu and gpu usage
jobstats &

singularity exec --nv /sw/hprc/sw/containers/alphafold/alphafold_2.2.0.sif \
 python /app/alphafold/run_alphafold.py \
 --use_gpu_relax \
 --data_dir=$DOWNLOAD_DIR \
 --uniref90_database_path=$DOWNLOAD_DIR/uniref90/uniref90.fasta \
 --mgnify_database_path=$DOWNLOAD_DIR/mgnify/mgy_clusters_2018_12.fa \
 --small_bfd_database_path=$DOWNLOAD_DIR/small_bfd/bfd-first_non_consensus_sequences.fasta \
 --model_preset=monomer_ptm \
 --pdb70_database_path=$DOWNLOAD_DIR/pdb70/pdb70 \
 --template_mmcif_dir=$DOWNLOAD_DIR/pdb_mmcif/mmcif_files \
 --obsolete_pdbs_path=$DOWNLOAD_DIR/pdb_mmcif/obsolete.dat \
 --max_template_date=2022-1-1 \
 --db_preset=reduced_dbs \
 --output_dir=out_alphafold \
 --fasta_paths=$DOWNLOAD_DIR/example_data/T1083.fasta

# graph pLDDT and PAE .pkl files
run_AlphaPickle.py -od out_alphafold/pickle_out_dir

# run jobstats to create a graph of cpu and gpu usage for this job
jobstats
```

Visualize AlphaFold pLDDT Scores



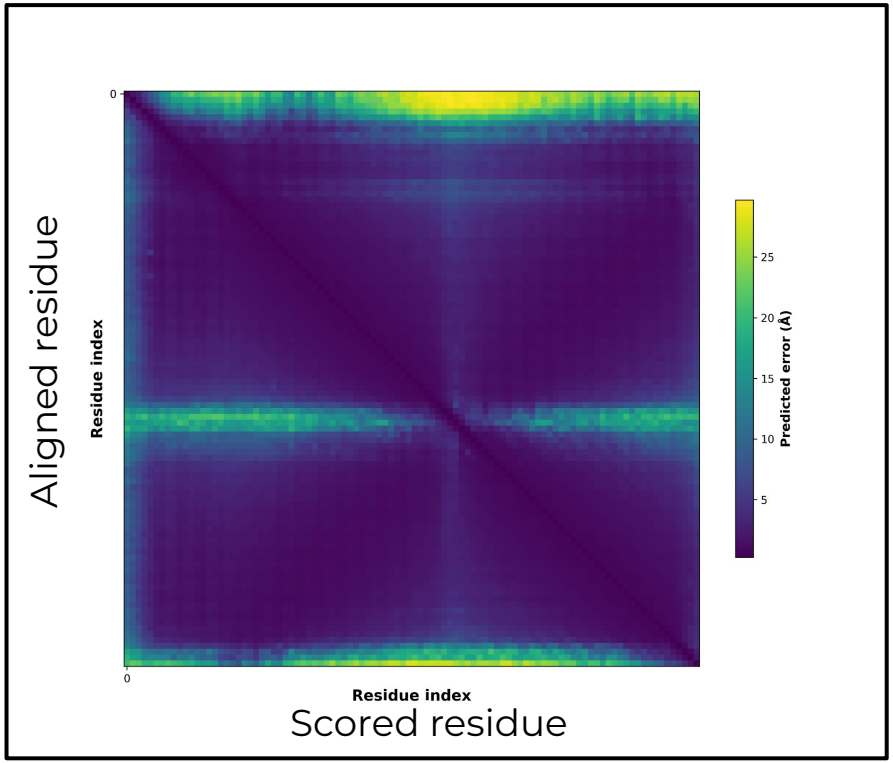


> 90 = Very high
70 - 90 = Confident
50 - 70 = Low
< 50 = Very low

```
eog out_1L2Y_reduced_dbs_monomer_ptm/1L2Y/ranked_0_pLDDT.png
```

you may get different results compared to the image above when using reduced_dbs

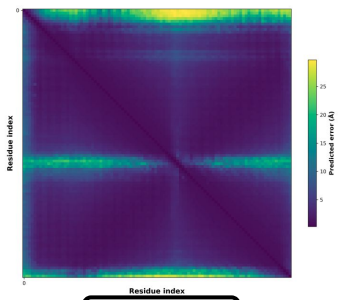
Visualize AlphaFold PAE Results (monomer_ptm)



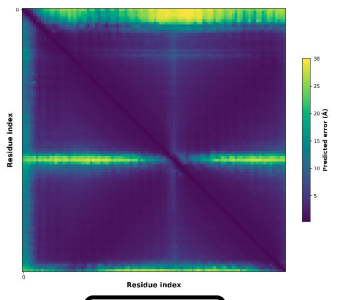
- Low Predicted Aligned Error (PAE) value has higher confidence of accuracy
- Must use monomer_ptm or multimer as model_preset to create PAE image
- The colour at position (x, y) indicates AlphaFold's expected position error at residue x, when the predicted and true structures are aligned on residue y.

```
eog out_1L2Y_reduced_dbs_monomer_ptm/1L2Y/ranked_0_PAE.png
```

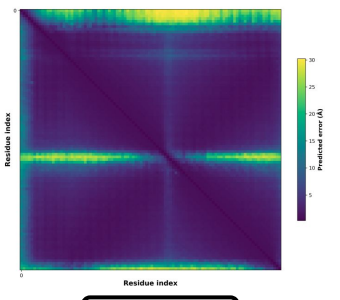
Evaluating Models



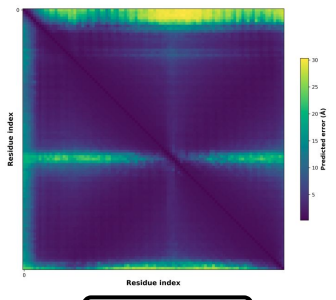
ranked_0



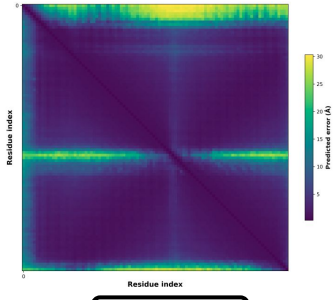
ranked_1



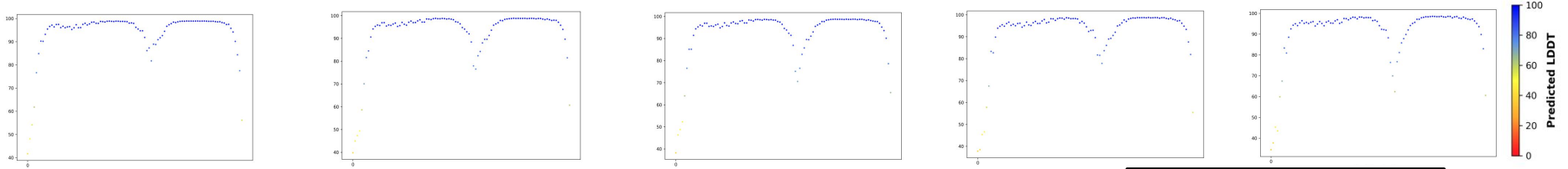
ranked_2



ranked_3



ranked_4



see which model has the top rank based on pLDDT score

```
cat out_IL2Y_reduced_dbs_monomer_ptm/IL2Y/ranking_debug.json
```


```
"plddts": {
  "model_1_ptm_pred_0": 94.16585478746399,
  "model_2_ptm_pred_0": 94.64120852328334,
  "model_3_ptm_pred_0": 89.94980057627299,
  "model_4_ptm_pred_0": 77.53515668415058,
  "model_5_ptm_pred_0": 88.40610380463586
},
"order": [
  "model_2_ptm_pred_0",
  "model_1_ptm_pred_0",
  "model_3_ptm_pred_0",
  "model_5_ptm_pred_0",
  "model_4_ptm_pred_0"
]
```


ranked_0

ranked_4

AlphaFold Job Resource Monitoring

Review CPU usage for a Job

The **seff** command displays CPU and memory resource usage and efficiency

```
seff 7355535
```

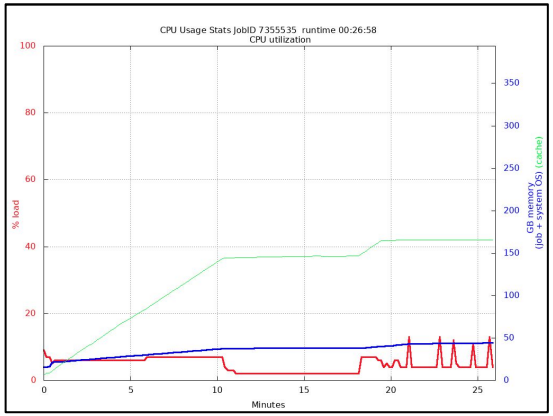
```
Job ID: 7355535
Cluster: grace
User/Group: netid/netid
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 24
CPU Utilized: 00:34:12
CPU Efficiency: 5.12% of 11:08:00 core-walltime
Job Wall-clock time: 00:27:50
Memory Utilized: 9.27 GB
Memory Efficiency: 5.15% of 180.00 GB
```

usage stats are not accurate until the job is complete

Review GPU and CPU usage for a Job

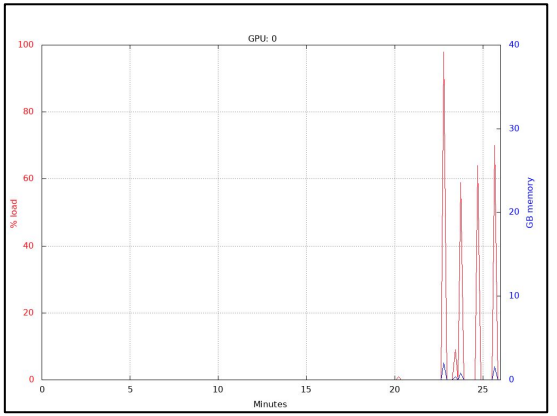
The **jobstats** command monitors GPU and CPU resource usage and create graphs

```
#!/bin/bash
#SBATCH --job-name=my_gpu_job
#SBATCH --time=1-00:00:00
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=24
#SBATCH --mem=180G
#SBATCH --gres=gpu:a100:1
#SBATCH --output=stdout.%x.%j
#SBATCH --error=stderr.%x.%j

module purge

# run jobstats in the background (&)
# to monitor resource usage
jobstats &

my_alphafold_command

# run jobstats to create a graph
# of cpu and gpu usage for this job
jobstats
```



eog stats_cpu.7355535.png

only works for jobs using just one compute node



eog stats_gpu.7355535.png

.pdf file is created when more than 4 GPUs used in a job

When the job is complete, login with ssh -X option and view graphs of GPU and CPU usage stats using **eog** for .png files and **evince** for .pdf files

AlphaFold Workflow Alternatives

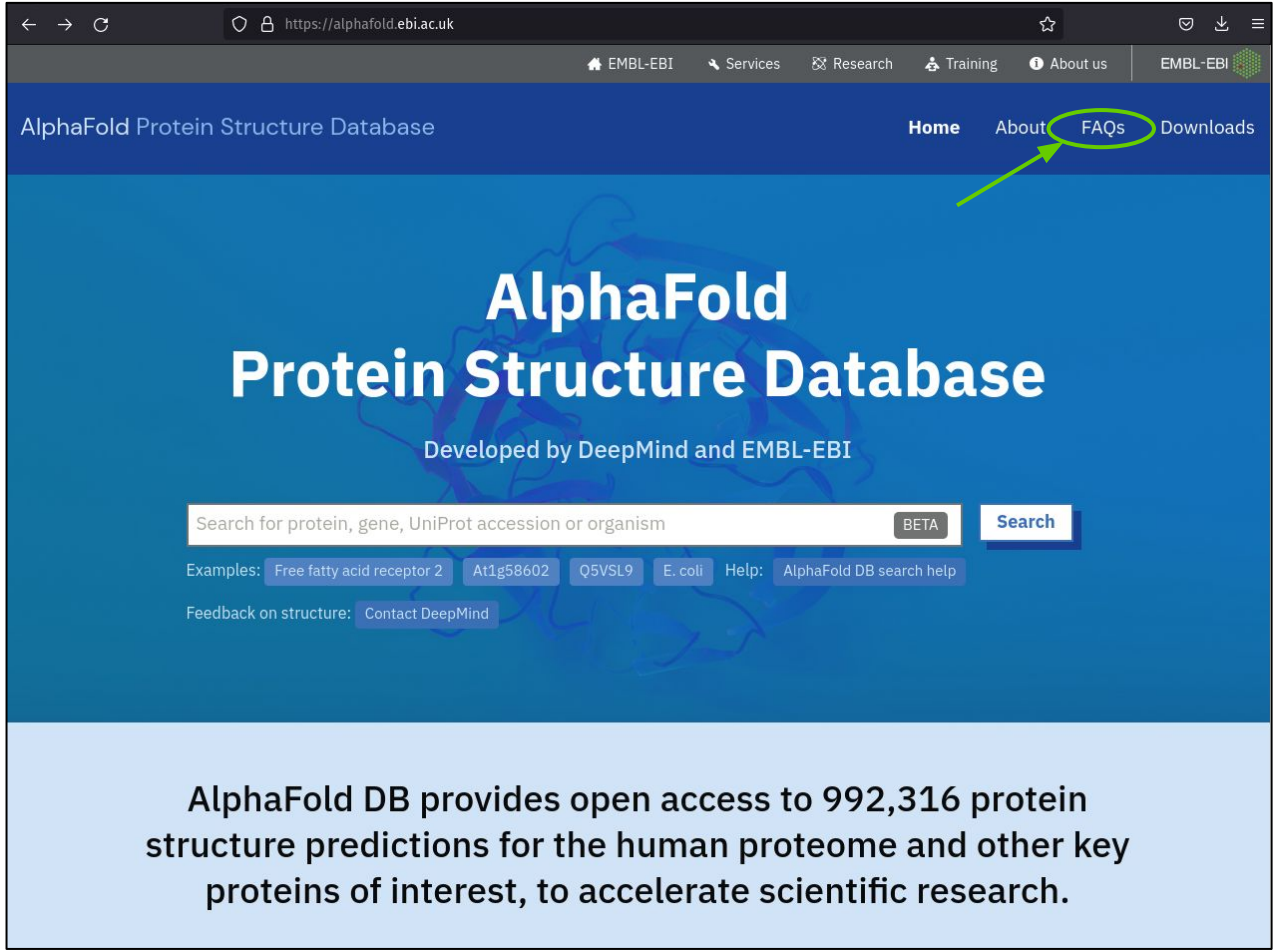
ParallelFold

- ParallelFold (ParaFold) breaks the AlphaFold workflow into two steps
  - processing of the three CPU steps in parallel
  - processing of the GPU step
- The parallel portion for CPU steps is not implemented yet resulting in similar or longer runtimes than the DeepMind approach
  - the first three CPU steps, jackhammer, jackhammer and HHblits are supposed to run as three separate processes in parallel but currently this parallelization step is not implemented yet
  - when the CPU processing is parallelized, it will require 21 cores
    - 8 cores for each of the two jackhammer steps
    - 5 cores for the HHblits step.
- These steps may be implemented in parallel in AlphaFold soon

https://github.com/Zuricho/ParallelFold

Databases and References

DeepMind and EMBL's European Bioinformatics Institute (EMBL-EBI) have partnered to create AlphaFold DB to make these predictions freely available to the scientific community.



References

Article | Open Access | Published: 15 July 2021

Highly accurate protein structure prediction with AlphaFold

John Jumper, Richard Evans, ... Demis Hassabis + Show authors

Nature **596**, 583–589 (2021) | Cite this article

Article | Open Access | Published: 22 July 2021

Highly accurate protein structure prediction for the human proteome

Kathryn Tunyasuvunakool, Jonas Adler, ... Demis Hassabis + Show authors

Nature **596**, 590–596 (2021) | Cite this article

Arnold, M. J. (2021) AlphaPickle doi.org/10.5281/zenodo.5708709