HIGH PERFORMANCE RESEARCH COMPUTING

Introduction to AlphaFold for 3D Protein Structure Prediction on Grace

High Performance
Research Computing
DIVISION OF RESEARCH

Fall 2024

AlphaFold for 3D Protein Structure Prediction on Grace

- HPRC Cluster Utilities
- Running AlphaFold
  - Grace ParaFold AlphaFold Workflow
  - ChimeraX + Google Colab
- Visualization of Results
  - job resource usage
  - view predictions in ChimeraX
  - plotting pLDDT values
- AlphaFold History
- AlphaFold3 Server

HPRC Cluster Utilities

HPRC Cluster Utilities

Many cluster utilities are available to help you query resources at the command line such as available nodes, GPUs, cores, memory, template job scripts and shared conda and Python environments.

myjob	**gpuavail**
maxconfig	**cpuavail**
gcatemplates	**envsavail**
jobstats	**venvavail**
toolchains	**maintenance**

use the **-h** or **--help** flag with any utility to see available options

Show Your Job Details using myjob

- The **`myjob`** command can be used to see detailed information related to your job.
  - Status (PENDING, RUNNING, COMPLETED, FAILED, …)
  - Node List
  - Submit time, Start time, End time, Total runtime
  - CPU Efficiency
  - Memory Utilized, Memory Efficiency
- will advise you if your job is PENDING due to a scheduled maintenance.
- will advise you if your job FAILED due to CRLF characters in the job script and provide a link to the HPRC documentation on how to resolve this issue.
- will advise you if your job FAILED due to file or disk quota being reached.
  - will show you the directory in your $HOME directory that has the most files when $HOME file quota is reached.

https://hprc.tamu.edu/kb/Software/useful-tools/myjob

Show Your Job Details using myjob

```
[userid@grace ~]$ myjob 11760808

              Job ID: 11760808
             Cluster: grace
          User/Group: userid/userid
               State: COMPLETED (exit code 0)
           Partition: cpu
          Node Count: 1
            NodeList: c098
      Cores per node: 24
        CPU Utilized: 01:01:54
      CPU Efficiency: 0.57% of 7-12:11:12 core-walltime
         Submit time: 2024-10-21 15:20:05
          Start time: 2024-10-21 15:20:26
            End time: 2024-10-21 22:50:54
 Job Wall-clock time: 07:30:28
     Memory Utilized: 37.08 GB
   Memory Efficiency: 20.60% of 180.00 GB
            Job Name: parafold-cpu
Job Submit Directory: /scratch/user/userid/af_demo
         Submit Line: sbatch run_parafold_2.0_alphafold_2.3.2_a100_monomer_ptm_grace.sh
```

use the -h flag to view usage

```
myjob -h
```

PENDING Job due to a Scheduled Maintenance

```
[userid@grace ~]$ myjob 1320633

               Job ID: 1320633
              Cluster: faster
           User/Group: userid/userid
              Account: 123456789101
                State: PENDING
               Reason: ReqNodeNotAvail, Reserved for maintenance
          Submit time: 2024-10-14 10:09:59
            Partition: cpu
           Node Count: 1
             NodeList: None assigned
                Cores: 1
                 Note: Efficiency not available for jobs in the PENDING state.
             Job Name: picard
Job Submit Directory: /scratch/user/userid/myproject/picard
          Submit Line: sbatch run_bwa_samtools_pilon_faster.sh
                 Note: job is PENDING due to runtime overlapping with maintenance window.
                 Note: maintenance will begin in  22 hours, and 49 minutes.
```

Viewing Maximum Available Resources

The **maxconfig** command will show the recommended Slurm parameters for the maximum available resources (cores, memory, time) per node for a specified accelerator or partition (default Grace partition: long) and show SUs charged.

```
[userid@grace ~]$ maxconfig

  Grace partitions:   short  medium  long  xlong  vnc  gpu  bigmem  special  gpu-a40
  Grace GPUs in gpu partition:  a100:2  a40:3  rtx:2  t4:4

  Showing max parameters (cores, mem, time) for partition long

  CPU-billing * hours * nodes =   SUs
           48 *   168 *     1 = 8,064

#!/bin/bash
#SBATCH --job-name=my_job
#SBATCH --time=7-00:00:00
#SBATCH --nodes=1           # max 64 nodes for partition long
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=48
#SBATCH --mem=360G
#SBATCH --output=stdout.%x.%j
#SBATCH --error=stderr.%x.%j
```

Viewing Maximum Available Resources

See the recommended Slurm parameters for requesting 1 x A100 GPU with ½ the total CPUs and memory since there are 2 x A100s per node.

```
[userid@grace ~]$ maxconfig -g a100 -G 1

  Grace partitions:   short  medium  long  xlong  vnc  gpu  bigmem  special  gpu-a40
  Grace GPUs in gpu partition:  a100:2  a40:3  rtx:2  t4:4

  Showing 1/2 of total cores and memory for using 1 x a100 GPU

  (CPU-billing + (GPU-billing * GPU-count)) * hours * nodes = SUs
  (         25 + (         72 *         1)) *    96 *     1 = 9,312

#!/bin/bash
#SBATCH --job-name=my_job
#SBATCH --time=4-00:00:00
#SBATCH --nodes=1              # max 32 nodes for partition gpu
#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
```

Grace SUs Charged: GPU vs CPU-only Jobs

show SU rate for 1 day CPU-only

```
maxconfig -d 1
```

```
CPU-billing * hours * nodes =  SUs
         48 *    24 *     1 =  1,152
```

show SU rate for 1 x A100 GPU for 1 day

```
maxconfig -g a100 -G 1 -d 1
```

```
(CPU-billing + (GPU-billing * GPU-count)) * hours * nodes = SUs
(         25 + (         72 *         1)) *    24 *     1 = 2,328
```

show SU charge rate

```
maxconfig -h
```

```
SU rate per GPU:
     GPU   SUs per hour
     ----  ------------
     a100        72
     a40         48
     rtx         48
     t4          24
```

Checking GPU Configuration & Availability on Grace

- Use the command line (shell) to see the current GPU configuration and availability.
- If there are no GPU nodes in the AVAILABILITY output, it means that the GPUs are busy with other jobs and a GPU job that you submit may take a while to start.

```
[userid@grace ~]$ gpuavail

    CONFIGURATION
NODE           NODE
TYPE           COUNT
--------------------
gpu:a100:2     100
gpu:a40:3      15
gpu:rtx:2      9
gpu:t4:4       8


               AVAILABILITY
NODE   GPU     GPU    GPUs     CPUs   GB MEM
NAME   TYPE    COUNT  AVAIL    AVAIL  AVAIL
-----------------------------------------------
g003   a100      2      1       47    296
g004   a100      2      2       48    360
g005   a100      2      2       48    360
g006   a100      2      2       48    360
g008   a100      2      1       47    356
```

https://hprc.tamu.edu/kb/Software/useful-tools/gpuavail

Check non-GPU node Availability

Use the **`cpuavail`** command to see non-GPU nodes readily available for jobs. non-GPU nodes not in the AVAILABILITY table are busy with other jobs and will be available after jobs have completed.

```
[userid@grace ~]$ cpuavail

    CONFIGURATION                          AVAILABILITY
  NODE          NODE                  NODE        CPUs        GB MEM
  TYPE          COUNT                 NAME        AVAIL       AVAIL
  ---------------------               ----------------------------
  CPU-only       808                  c003          2          20
  GPU            132                  c028         12          84
                                      c044         48         360
                                      c045         32         352
                                      c047         36         352
                                      c048         48         360
                                      c050          2         197
                                      c052         48         360
                                      c053         48         360
                                      c056         43         252
```

https://hprc.tamu.edu/kb/Software/useful-tools/cpuavail

Running AlphaFold on Grace using ParaFold

AlphaFold Databases on Grace

`/scratch/data/bio/alphafold/2.3.2`

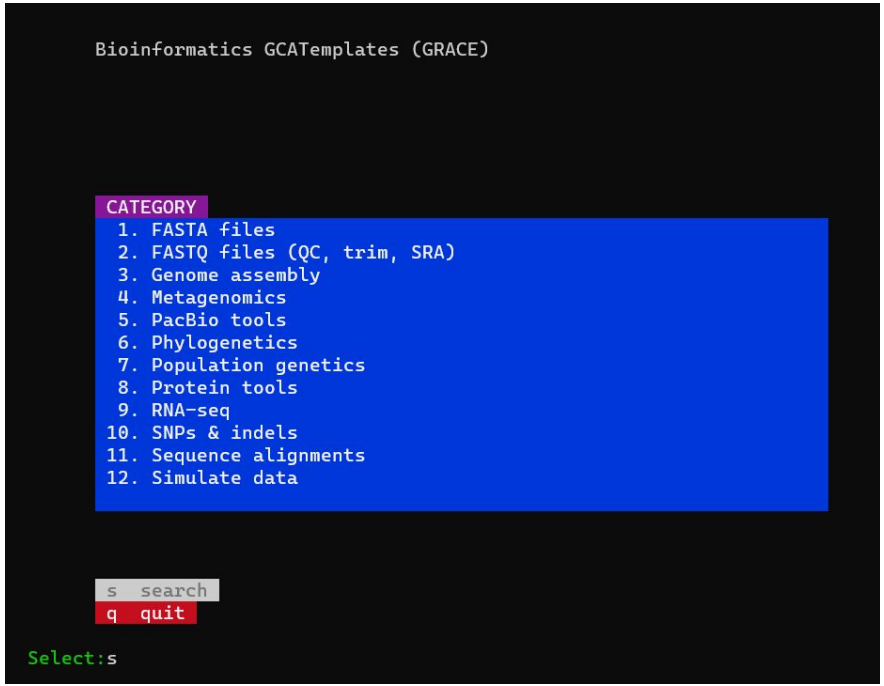
Database	Size	File Count	monomer	multimer
bfd	1.8T	7	✔	✔
mgnify	120G	2	✔	✔
params	5.3G	17	✔	✔
pdb70	56G	10	✔	-
pbd_mmcif	264G	211,106	✔	✔
pdb_seqres	257M	2	-	✔
uniprot	114G	2	-	✔
uniref30	467G	15	✔	✔
uniref90	77G	2	✔	✔
small_bfd	17G	2	✔	✔
example_data	6K	5	✔	✔
TOTAL	**2.9T**	**211,170**		

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/af_demo
cd $SCRATCH/af_demo
gcatemplates
```

- Type **s** for search then enter **parafold** to search for the **parafold_2.0+alphafold 2.3.2** template script and select the **monomer_ptm** script option
- Review and submit the job script, which takes about 3 hours to complete.



Submit and Monitor the Job

- Run the cpuavail utility to see cluster usage status.

```
cpuavail
```

- Edit your job script to use 10 cores and 100 GB memory for CPU job.
- Submit the job script to the Slurm scheduler.
  - completes in about 7.5 hours so we will review a completed job

```
sbatch run_parafold_2.0_alphafold_2.3.2_a100_monomer_ptm_grace.sh

Submitted batch job 11760808
```

- Monitor the job status.

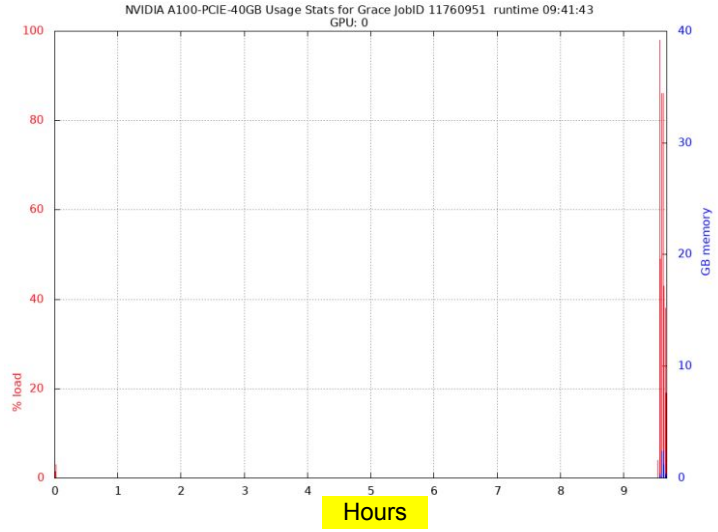
```
squeue --me

JOBID     NAME           USER      PARTITION  NODES  CPUS  STATE   TIME   TIME_LEFT   START_TIME          REASON  NODELIST
11760808  parafold-cpu   userid    cpu        1      24    RUNNING 6.59   6-23:53:01  2024-10-21T15:20    None    c398
```

Comparison of DeepMind vs ParaFold Workflows

- AlphaFold DeepMind's one job workflow (1 GPU job) vs ParaFold's two job workflow (1 CPU-only job + 1 GPU job) for the same monomer_ptm full_dbs analysis
- *The ParaFold workflow significantly reduces GPU idle time*

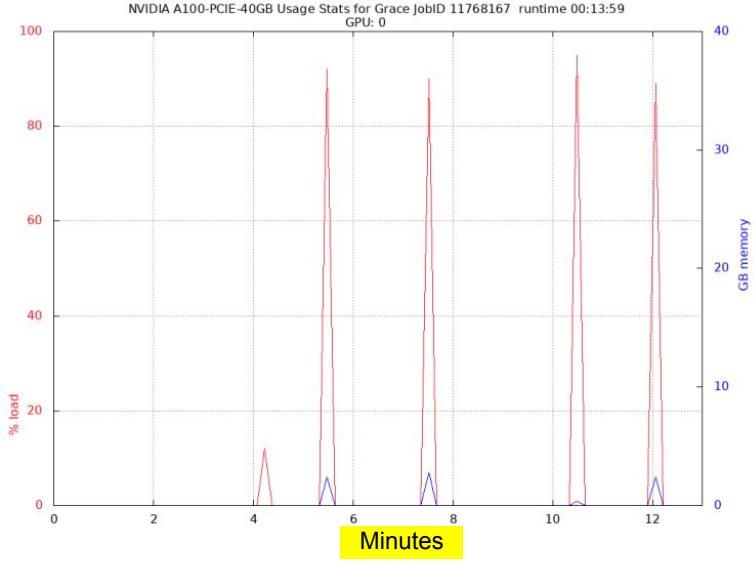
DeepMind Workflow

AlphaFold 2.3.1



ParaFold Workflow (GPU job)

AlphaFold 2.3.2



The first job of the ParaFold workflow (CPU-only) completed in 7.5 hours

ParaFold Workflow

- The ParaFold module uses the official AlphaFold code.
- ParaFold divides the AlphaFold workflow into two steps which can be run as two separate jobs:
  - CPU-only: processing the CPU steps to generate multiple sequence alignments
  - GPU: processing the GPU steps to generate predictions
- Test run of multimer (T1083_T1084_multimer.fasta) with full_dbs on ACES
- Runtimes for the same job script varied +- 1 hour; TM-scores also vary

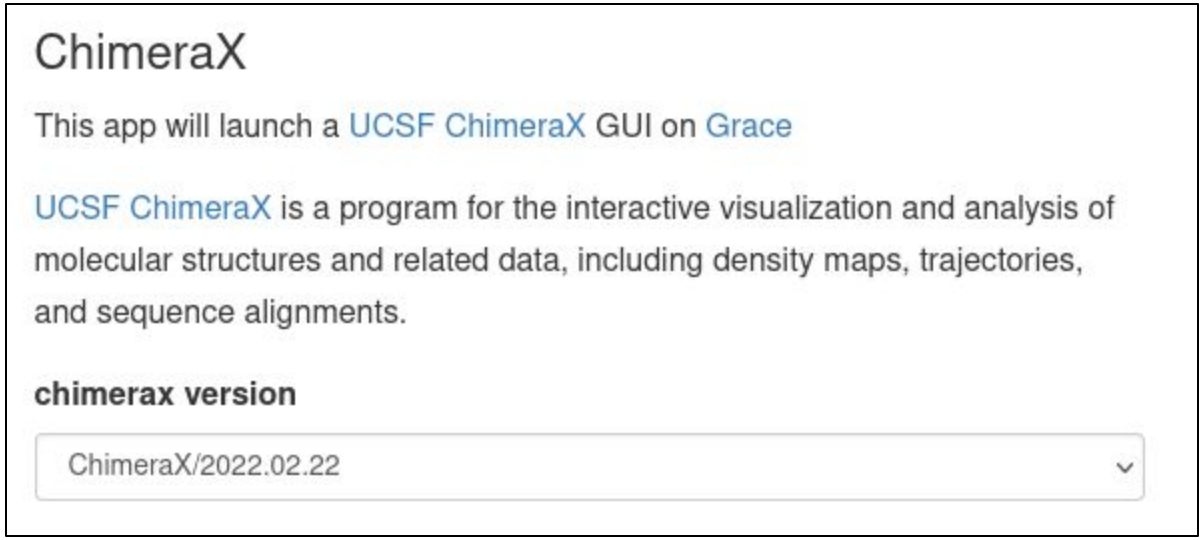
AlphaFold 2.3.2	Runtime	Highest Scoring Model	TM-score**
ParaFold	3 hrs 10 min*	model_1_multimer_v3_pred_4	0.892
DeepMind	2 hrs 45 min	model_1_multimer_v3_pred_1	0.883

***** combined time for the separate CPU 3 hour job and GPU 10 min job
****** measure of similarity between two protein structures

https://github.com/Zuricho/ParallelFold

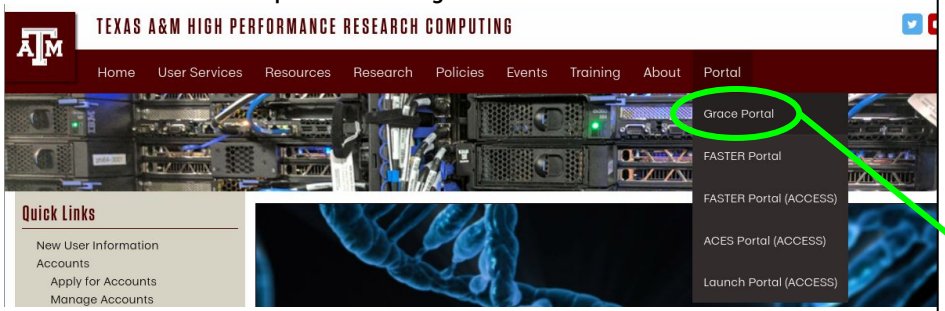
ChimeraX

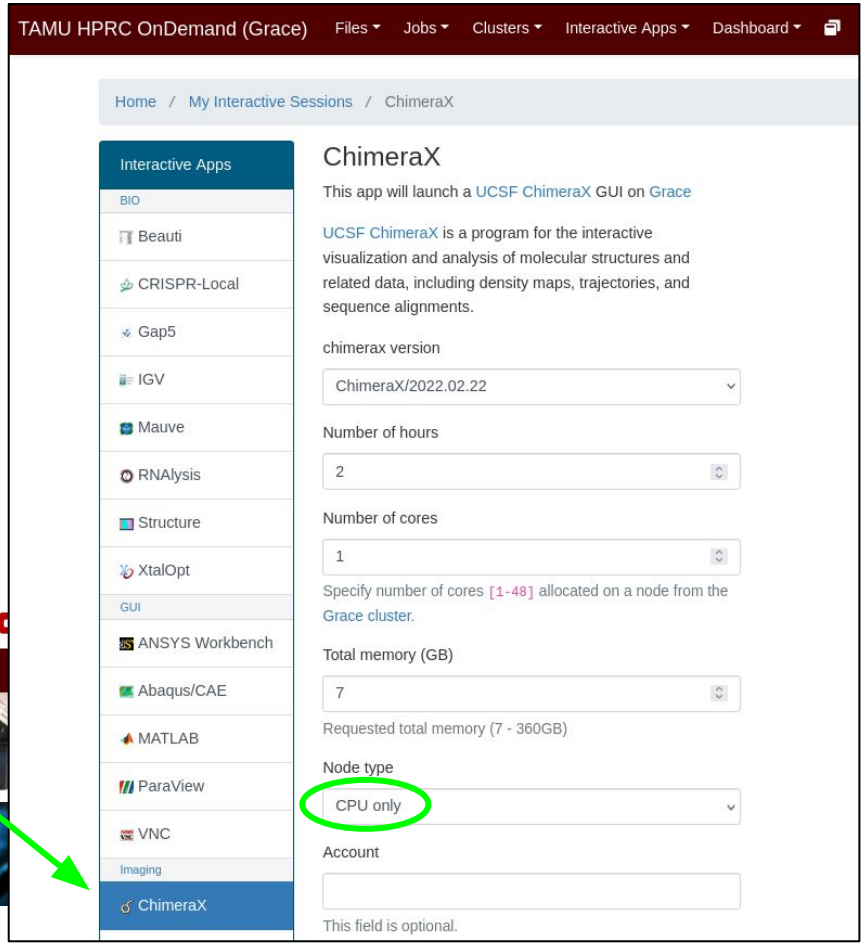
- Can be used to visualize protein structures
- Can be launched using the Grace portal
  - portal-grace.hprc.tamu.edu
- Can be used to run AlphaFold using the daily build version (2022.02.22+)
  - uses Google Colab with limited resources



ChimeraX

- 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





AlphaFold Runtimes on Grace

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

AlphaFold 2.2.3	monomer_ptm 20 aa	multimer 98 & 73 aa
A100	36 minutes	4 hours 49 minutes
A40	35 minutes	-
RTX 6000	33 minutes	4 hours 44 minutes
T4	33 minutes	4 hours 45 minutes
CPU only	40 minutes	6 hours 11 minutes

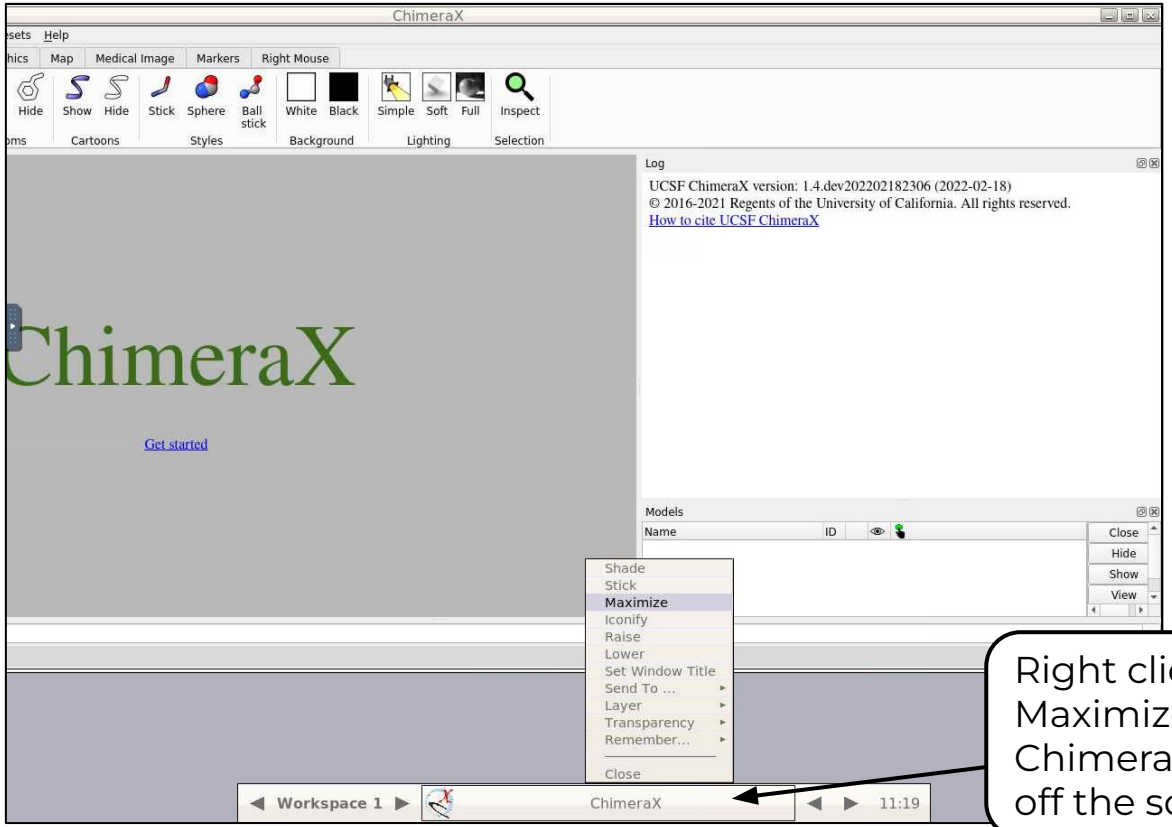
- Can be run in ChimeraX from the Grace portal
  - using the ChimeraX AlphaFold option
  - all processing done on Google cloud servers

Resource Limitations

- AlphaFold
  - currently AlphaFold can only utilize one GPU
  - about 90% of processing is done on CPU when using DeepMind's workflow
- 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 with ChimeraX + Google Colab

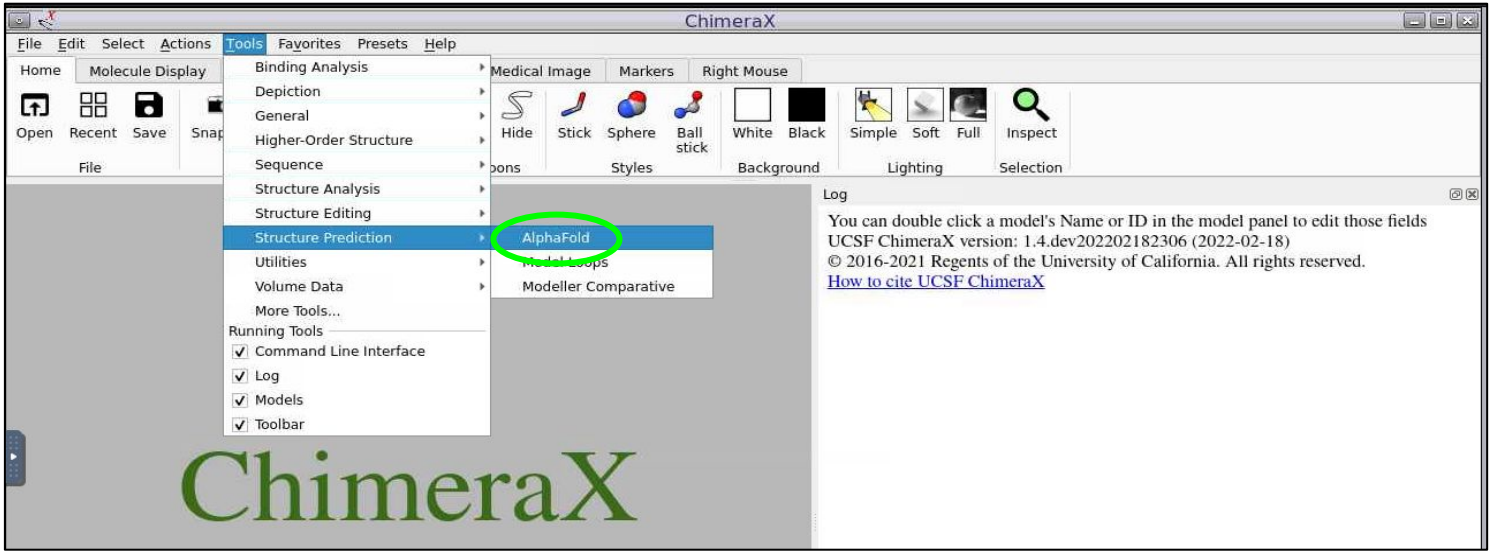
Maximize ChimeraX Window



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

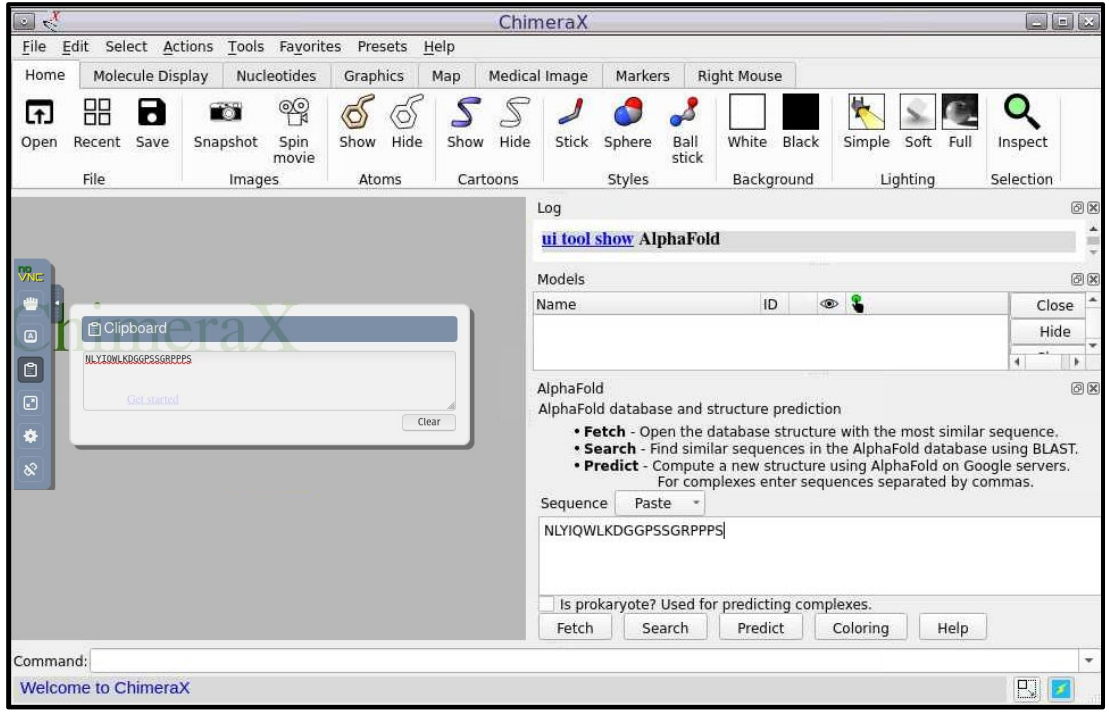
AlphaFold with ChimeraX

- 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 DeepMind Workflow

- **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
module load GCC/11.3.0  OpenMPI/4.1.4 AlphaFold/2.3.1-CUDA-11.7.0

ALPHAFOLD_DATA_DIR=/scratch/data/bio/alphafold/2.3.2

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

run_alphafold.py \
 --use_gpu_relax \
 --data_dir=$ALPHAFOLD_DATA_DIR \
 --uniref90_database_path=$ALPHAFOLD_DATA_DIR/uniref90/uniref90.fasta \
 --uniref30_database_path=$ALPHAFOLD_DATA_DIR/uniref30/UniRef30_2023_02  \
 --mgnify_database_path=$ALPHAFOLD_DATA_DIR/mgnify/mgy_clusters_2025_05.fa \
 --bfd_database_path=$ALPHAFOLD_DATA_DIR/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt \
 --model_preset=multimer \
 --pdb_seqres_database_path=$ALPHAFOLD_DATA_DIR/pdb_seqres/pdb_seqres.txt \
 --uniprot_database_path=$ALPHAFOLD_DATA_DIR/uniprot/uniprot.fasta \
 --template_mmcif_dir=$ALPHAFOLD_DATA_DIR/pdb_mmcif/mmcif_files \
 --obsolete_pdbs_path=$ALPHAFOLD_DATA_DIR/pdb_mmcif/obsolete.dat \
 --max_template_date=2024-1-1 \
 --db_preset=full_dbs \
 --output_dir=out_alphafold \
 --fasta_paths=/scratch/data/bio/alphafold/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 DeepMind Workflow

- **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
module load GCC/11.3.0  OpenMPI/4.1.4 AlphaFold/2.3.1-CUDA-11.7.0

ALPHAFOLD_DATA_DIR=/scratch/data/bio/alphafold/2.3.2

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

run_alphafold.py \
 --use_gpu_relax \
 --data_dir=$ALPHAFOLD_DATA_DIR \
 --uniref90_database_path=$ALPHAFOLD_DATA_DIR/uniref90/uniref90.fasta \
 --uniref30_database_path=$ALPHAFOLD_DATA_DIR/uniref30/UniRef30_2023_02  \
 --mgnify_database_path=$ALPHAFOLD_DATA_DIR/mgnify/mgy_clusters_2022_05.fa \
 --bfd_database_path=$ALPHAFOLD_DATA_DIR/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt \
 --template_mmcif_dir=$ALPHAFOLD_DATA_DIR/pdb_mmcif/mmcif_files \
 --obsolete_pdbs_path=$ALPHAFOLD_DATA_DIR/pdb_mmcif/obsolete.dat \
 --pdb70_database_path=$ALPHAFOLD_DATA_DIR/pdb70/pdb70 \
 --model_preset=monomer \
 --max_template_date=2024-1-1 \
 --db_preset=full_dbs \
 --output_dir=out_alphafold \
 --fasta_paths=/scratch/data/bio/alphafold/example_data/T1083.fasta

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

Example AlphaFold Job Script DeepMind Workflow

- **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
module load GCC/11.3.0  OpenMPI/4.1.4 AlphaFold/2.3.1-CUDA-11.7.0

ALPHAFOLD_DATA_DIR=/scratch/data/bio/alphafold/2.3.2

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

run_alphafold.py \
 --use_gpu_relax \
 --data_dir=$ALPHAFOLD_DATA_DIR \
 --uniref90_database_path=$ALPHAFOLD_DATA_DIR/uniref90/uniref90.fasta \
 --mgnify_database_path=$ALPHAFOLD_DATA_DIR/mgnify/mgy_clusters_2022_05.fa \
 --small_bfd_database_path=$ALPHAFOLD_DATA_DIR/small_bfd/bfd-first_non_consensus_sequences.fasta \
 --model_preset=monomer \
 --pdb70_database_path=$ALPHAFOLD_DATA_DIR/pdb70/pdb70 \
 --template_mmcif_dir=$ALPHAFOLD_DATA_DIR/pdb_mmcif/mmcif_files \
 --obsolete_pdbs_path=$ALPHAFOLD_DATA_DIR/pdb_mmcif/obsolete.dat \
 --max_template_date=2024-1-1 \
 --db_preset=reduced_dbs \
 --output_dir=out_alphafold \
 --fasta_paths=/scratch/data/bio/alphafold/example_data/1L2Y.fasta

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

Unified Memory

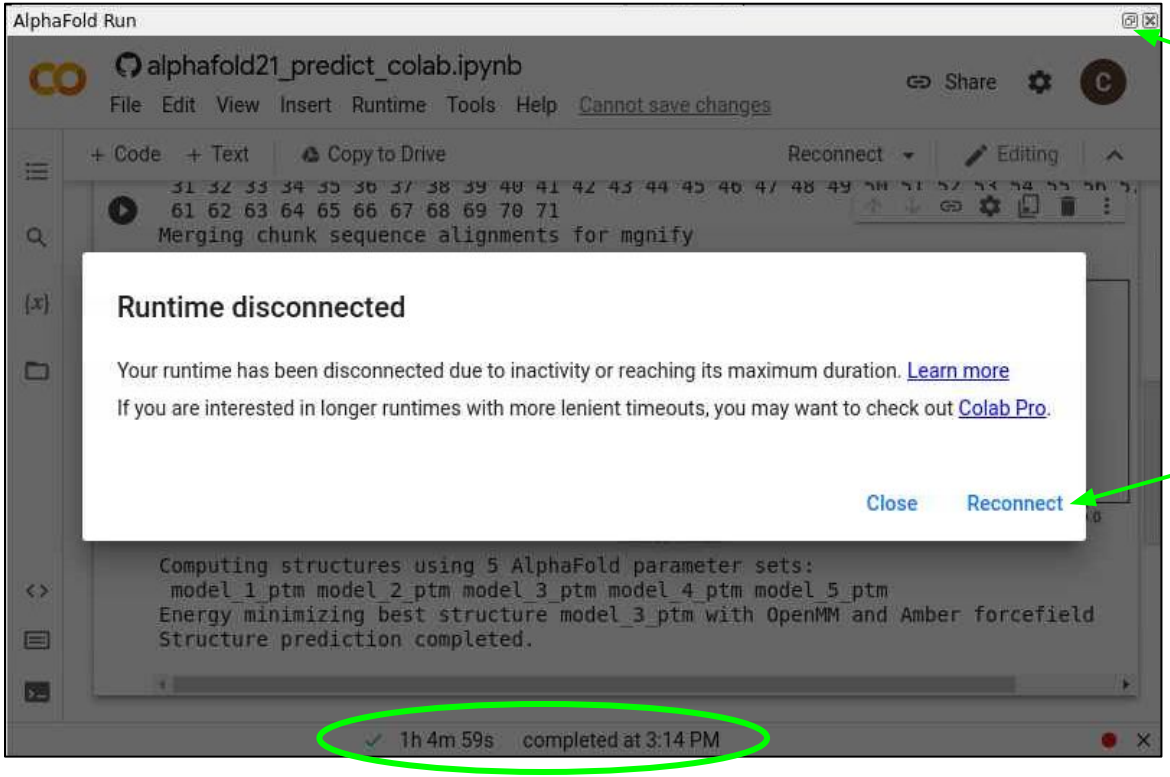
```
#!/bin/bash
#SBATCH --job-name=alphafold        # job name
#SBATCH --time=2-00:00:00           # max job run time dd-hh:mm:ss
#SBATCH --ntasks-per-node=1         # tasks (commands) per compute node
#SBATCH --cpus-per-task=24          # CPUs (threads) per command
#SBATCH --mem=180G                  # 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

# version 2.3.1 has the unified memory variable already configured
module purge
module load GCC/11.3.0  OpenMPI/4.1.4 AlphaFold/2.3.1-CUDA-11.7.0
```

- 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.
  - `XLA_PYTHON_CLIENT_MEM_FRACTION` (configured to 3.0 in the AlphaFold 2.3.1 module)
    - ParaFold is configured to 4.0
- this example script has 120 GB of unified preallocated memory:
  - 40 GB from A100 GPU + 80 GB DDR from motherboard.

  https://jax.readthedocs.io/en/latest/gpu_memory_allocation.html

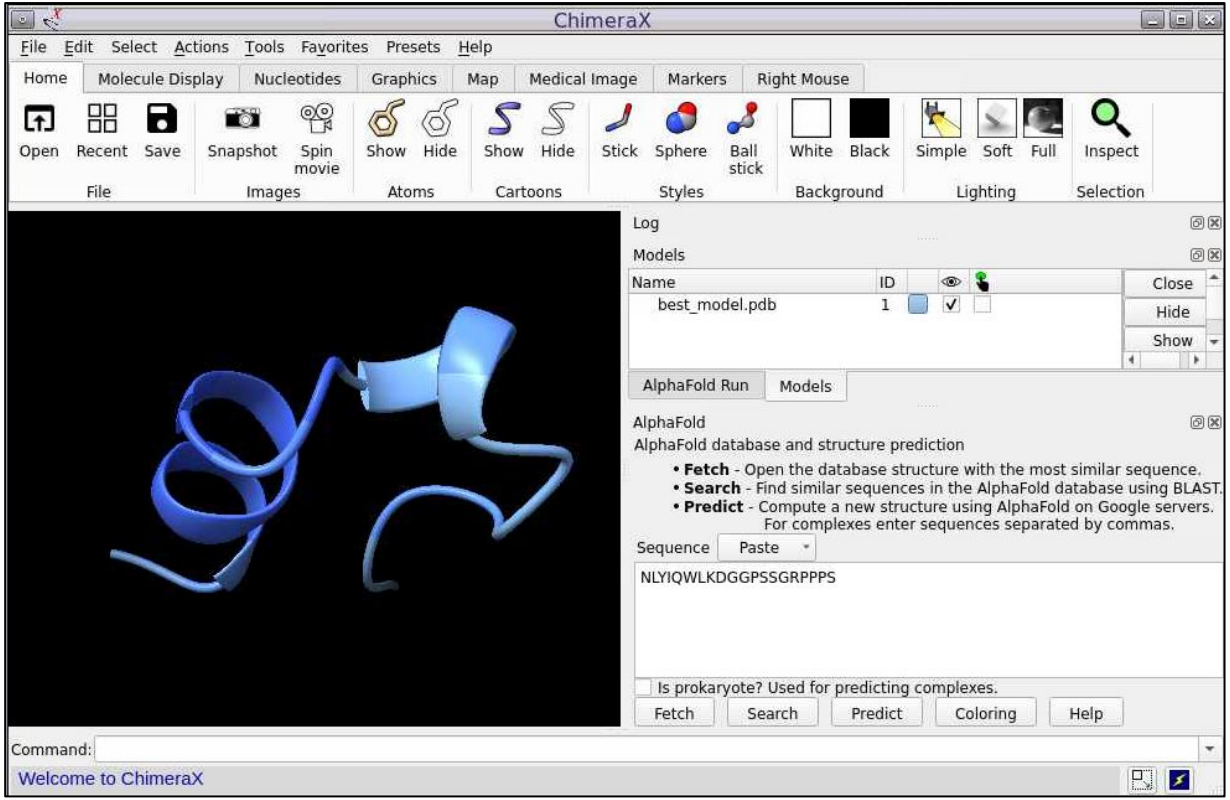
AlphaFold Results Visualization

Visualize Results with ChimeraX



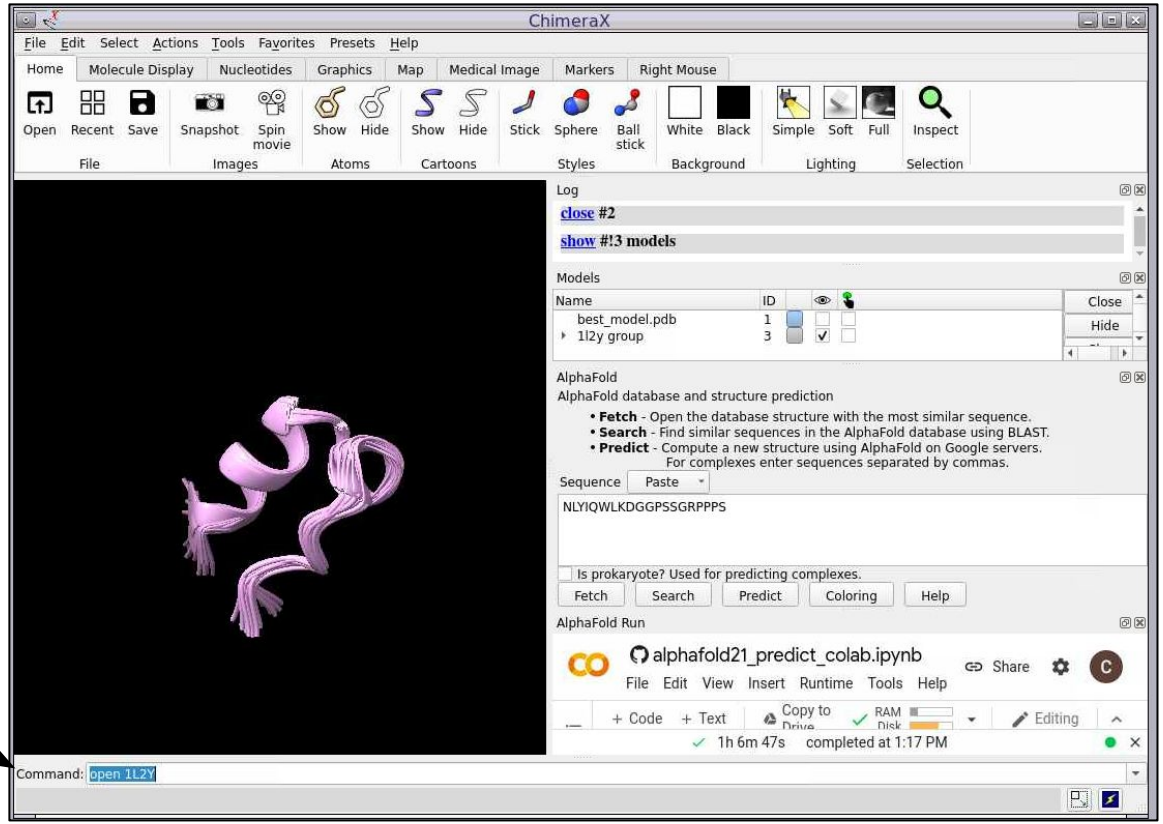
Click the minimize button to return to ChimeraX or click Reconnect then minimize

Visualize AlphaFold Google Colab Results with ChimeraX

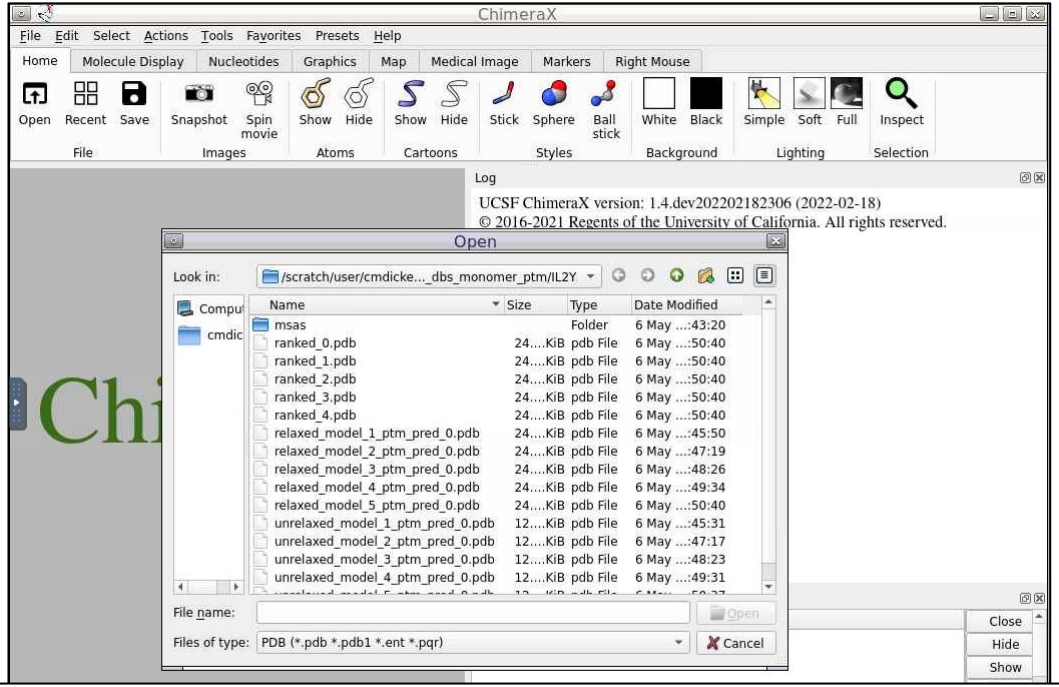


View PDB Structure if Available

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



Visualize AlphaFold Grace Results with ChimeraX



`/scratch/training/parafold/monomer_ptm/out_parafold_1L2Y_monomer_ptm`

AlphaFold Confidence Metrics

AlphaPickle for Visualization of Confidence Scores

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

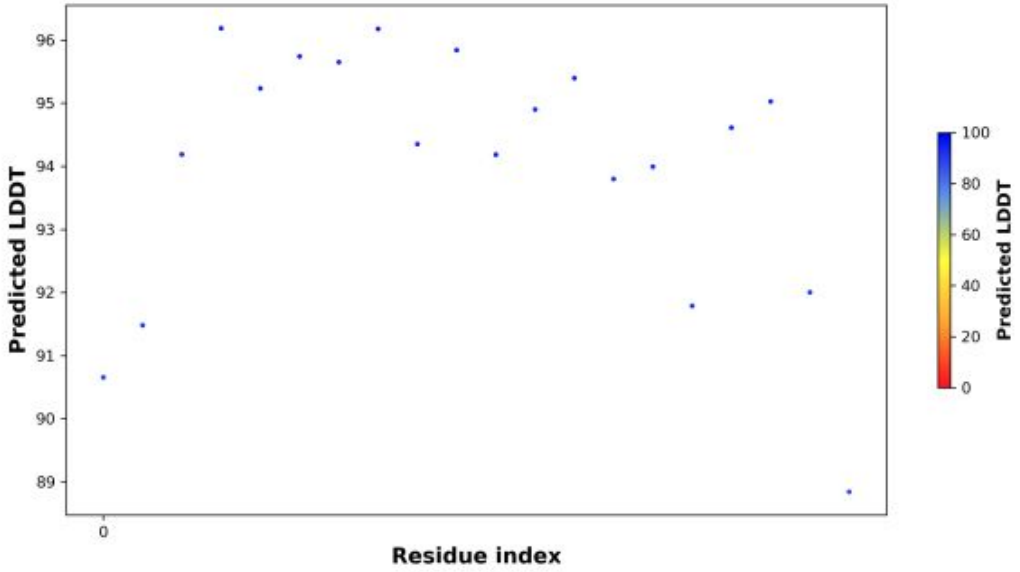
- graphing PAE scores is only available for the **monomer_ptm** and **multimer** model presets.
- load the AlphaPickle module at the beginning of the job script
- run AlphaPickle at the end, specifying the output directory used in the run_alphafold.py command.

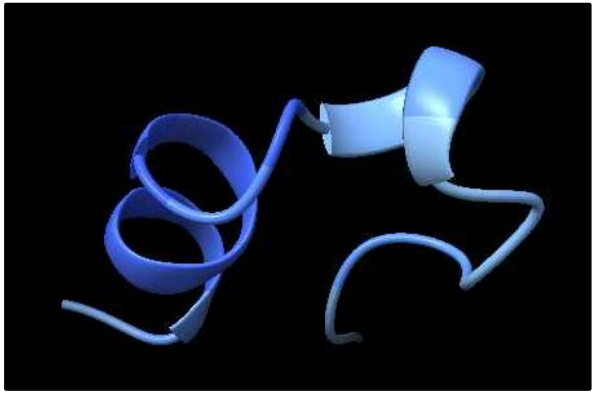
- pLDDT: scale from 0 - 100 of per-residue estimate of prediction confidence
- PAE: Predicted Alignment Error

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/11.3.0  OpenMPI/4.1.4 AlphaFold/2.3.1-CUDA-11.7.0
module load AlphaPickle/1.4.1

ALPHAFOLD_DATA_DIR=/scratch/data/bio/alphafold/2.3.2

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

run_alphafold.py \
 --use_gpu_relax \
 --data_dir=$ALPHAFOLD_DATA_DIR \
 --uniref90_database_path=$ALPHAFOLD_DATA_DIR/uniref90/uniref90.fasta \
 --mgnify_database_path=$ALPHAFOLD_DATA_DIR/mgnify/mgy_clusters_2022_05.fa \
 --small_bfd_database_path=$ALPHAFOLD_DATA_DIR/small_bfd/bfd-first_non_consensus_sequences.fasta \
 --model_preset=monomer_ptm \
 --pdb70_database_path=$ALPHAFOLD_DATA_DIR/pdb70/pdb70 \
 --template_mmcif_dir=$ALPHAFOLD_DATA_DIR/pdb_mmcif/mmcif_files \
 --obsolete_pdbs_path=$ALPHAFOLD_DATA_DIR/pdb_mmcif/obsolete.dat \
 --max_template_date=2024-1-1 \
 --db_preset=reduced_dbs \
 --output_dir=out_alphafold_2.3.1 \
 --fasta_paths=/scratch/data/bio/alphafold/example_data/1L2Y.fasta

# graph pLDDT and PAE .pkl files; part of the fasta file name will be the pickle directory name
run_AlphaPickle.py -od out_alphafold_2.3.1/1L2Y

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

Visualize AlphaFold pLDDT Scores



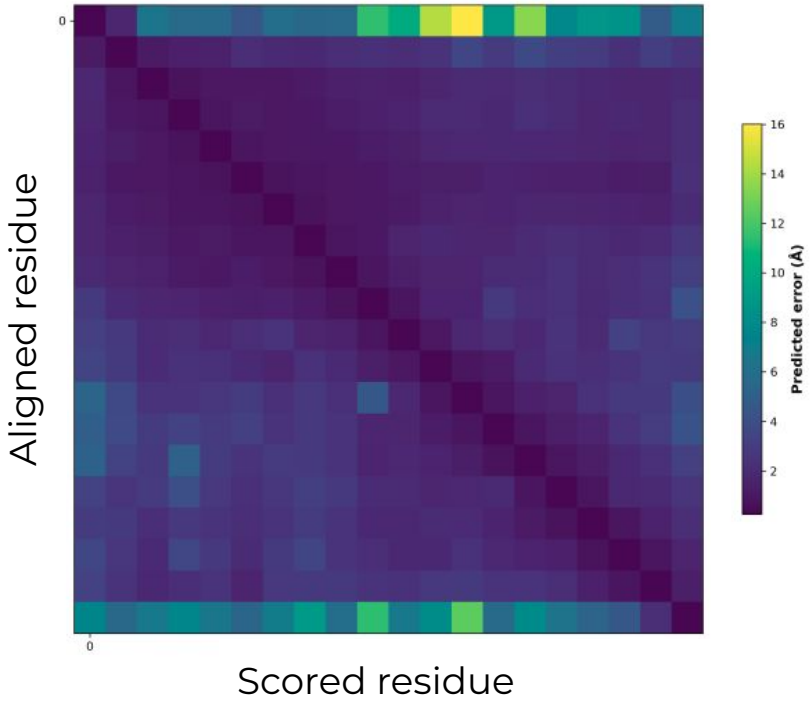


> 90 = Very high
70 - 90 = Confident
50 - 70 = Low
< 50 = Very low

```
eog out_parafold_1L2Y_monomer_ptm/1L2Y/ranked_0_pLDDT.png
```

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

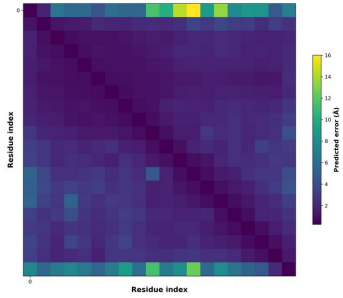
Visualize AlphaFold PAE Results (monomer_ptm)



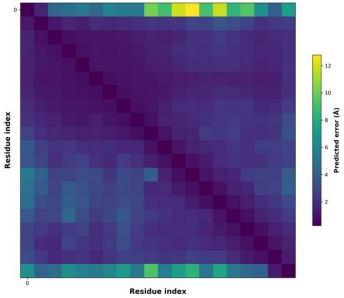
- Low Predicted Aligned Error (PAE) value has higher confidence of accuracy
- Must use monomer_ptm or multimer as model_preset to create PAE image
- The colour at position (x, y) indicates AlphaFold's expected position error at residue x, when the predicted and true structures are aligned on residue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

```
eog out_parafold_1L2Y_monomer_ptm/1L2Y/ranked_0_PAE.png
```

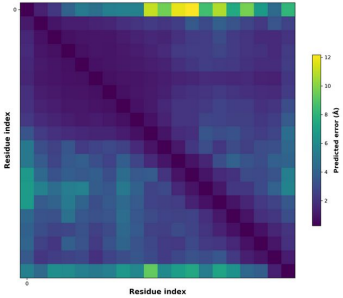
Evaluating Models



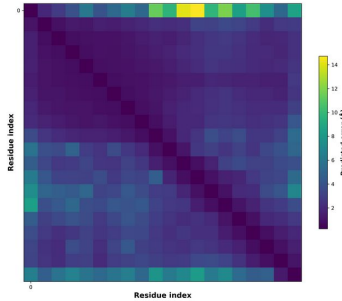
ranked_0



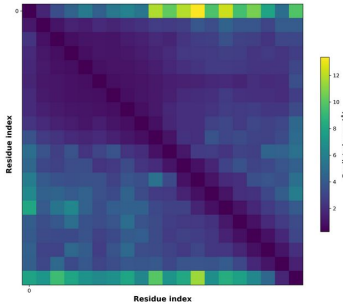
ranked_1



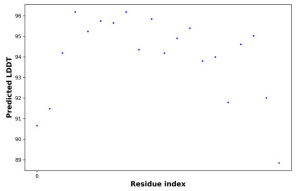
ranked_2

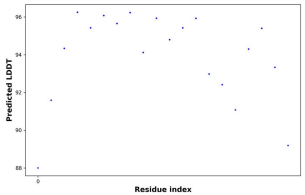


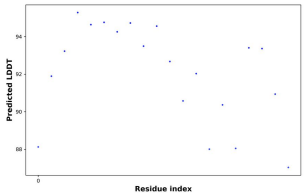
ranked_3

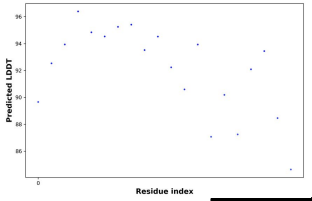


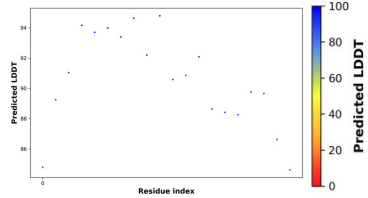
ranked_4











See which model has the top rank based on pLDDT score.

```
cat out_parafold_1L2Y_monomer_ptm/IL2Y/ranking_debug.json
```


```
"plddts": {
  "model_1_ptm_pred_0": 94.00143084962215,
  "model_2_ptm_pred_0": 93.92314232455674,
  "model_3_ptm_pred_0": 90.57195481381758,
  "model_4_ptm_pred_0": 92.07012004352153,
  "model_5_ptm_pred_0": 92.0243899816539
},
"order": [
  "model_1_ptm_pred_0",
  "model_2_ptm_pred_0",
  "model_4_ptm_pred_0",
  "model_5_ptm_pred_0",
  "model_3_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 11760808
```

```
Job ID: 11760808
Cluster: grace
User/Group: userid/userid
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 24
CPU Utilized: 01:01:54
CPU Efficiency: 0.57% of 7-12:11:12 core-walltime
Job Wall-clock time: 07:30:28
Memory Utilized: 37.08 GB
Memory Efficiency: 20.06% of 180.00 GB
```

Usage stats are not accurate until the job is complete.

AlphaFold History

- An Artificial Intelligence program developed by DeepMind
- 2018 AlphaFold 1 placed 1st at CASP 13
- 2020 AlphaFold 1 code released as open source
- 2020 AlphaFold 2 placed 1st at CASP 14
- 2021 AlphaFold publication in Nature
  - Highly accurate protein structure prediction with AlphaFold
- 2021 AlphaFold 2 code released as open source on GitHub
- 2024 AlphaFold 3 available on the deepmind AlphaFold Server
  - 20 jobs per day allowed for academic researchers

AlphaFold 3 Server

AlphaFold 3 vs AlphaFold 2

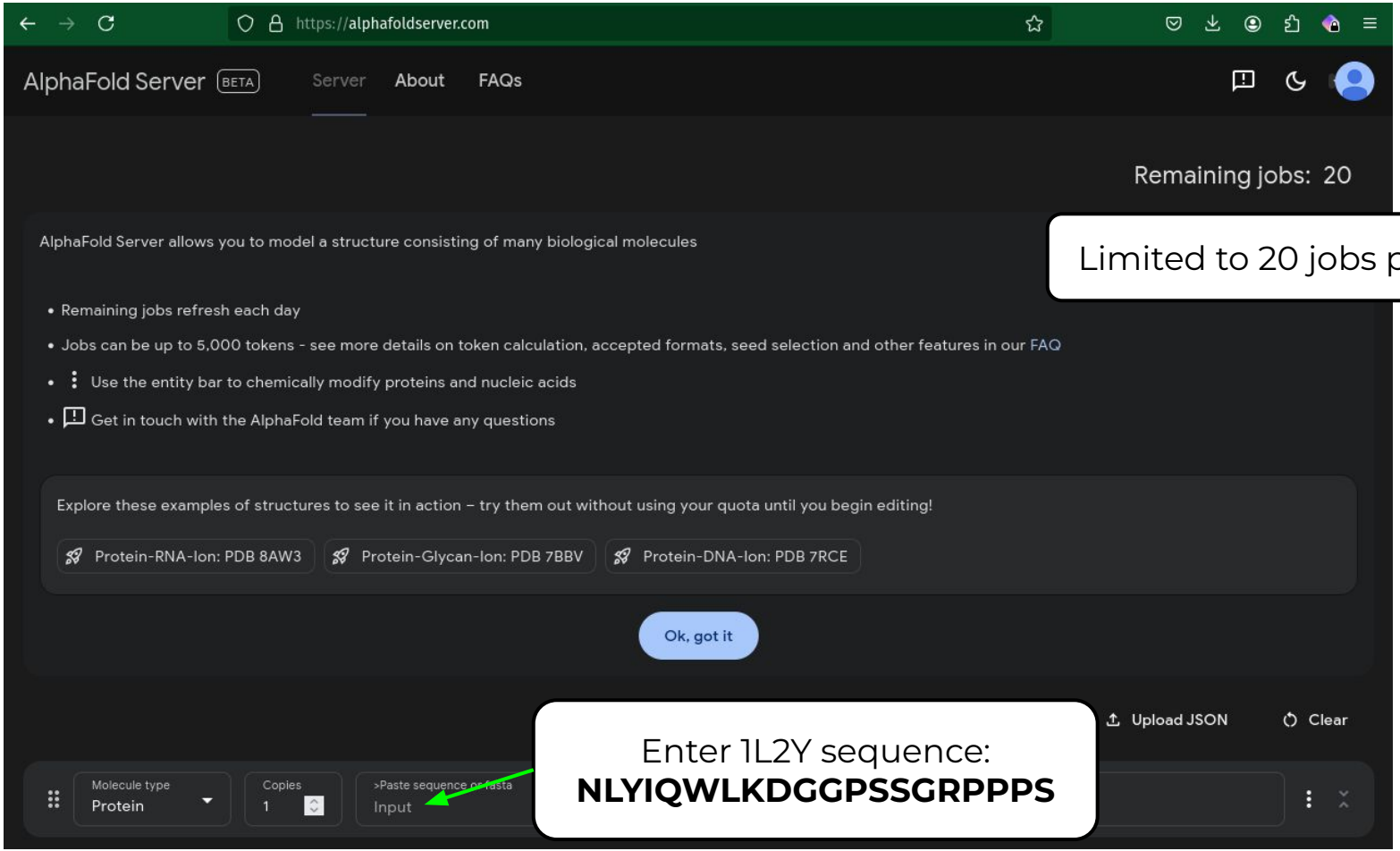
The new AlphaFold model demonstrates substantially improved accuracy over many previous specialized tools: far greater accuracy for protein–ligand interactions compared with state-of-the-art docking tools, much higher accuracy for protein–nucleic acid interactions compared with nucleic-acid-specific predictors and substantially higher antibody–antigen prediction accuracy compared with AlphaFold-Multimer v.2.37,8.

(from Abstract)

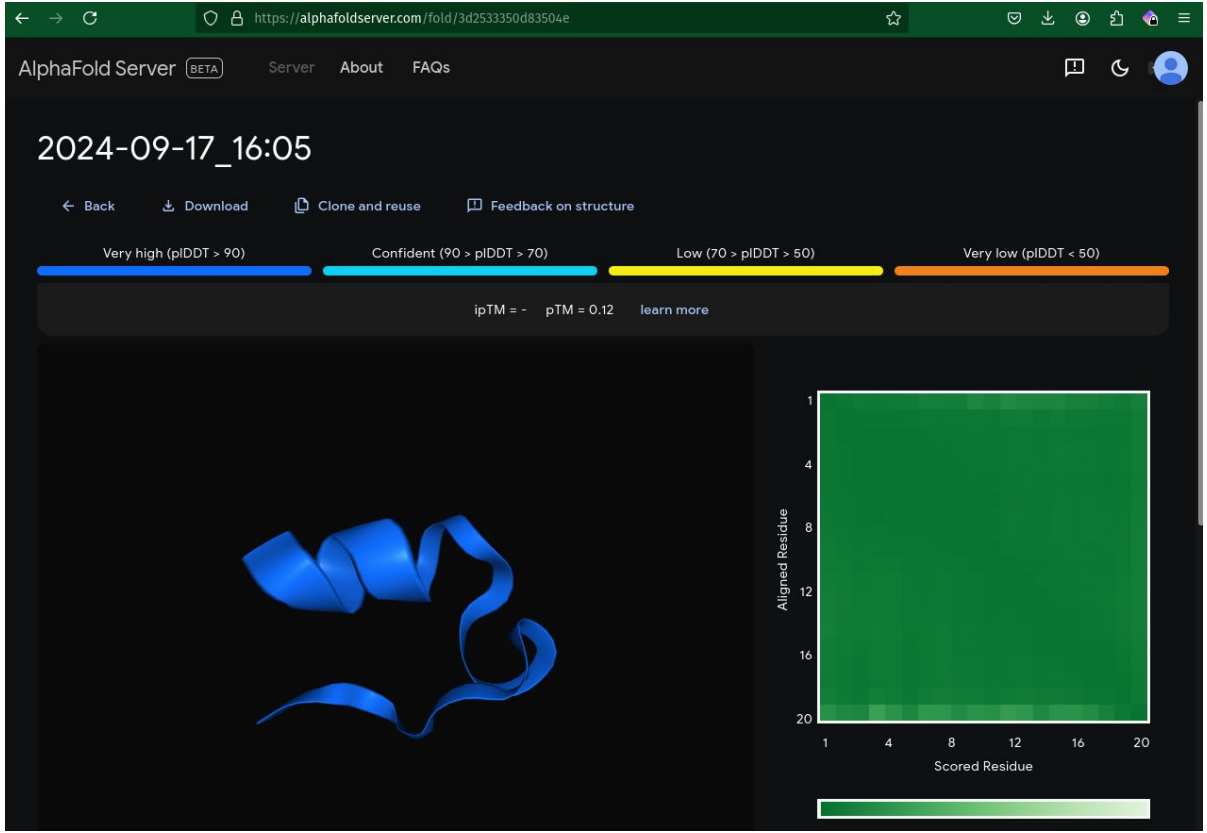
Abramson, J., Adler, J., Dunger, J. *et al.* Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* 630, 493–500 (2024). https://doi.org/10.1038/s41586-024-07487-w

AlphaFold 3 Server Terms

- AlphaFold Server is only available for non-commercial use by individuals and non-commercial organizations (universities, non-profit organizations and research institutes, educational and government bodies), or for journalism.
- You must not use AlphaFold Server or its outputs:
  - in connection with any commercial activities, including research on behalf of commercial organizations;
  - in any automated system that predicts the binding or interaction of the protein with ligands or peptides, such as Glide or AutoDock; or
  - to train machine learning models or related technology for biomolecular structure prediction similar to AlphaFold Server.
- You can publish, share and adapt AlphaFold Server output in accordance with our terms, including the requirement to provide clear notice that ongoing use is subject to AlphaFold Server Output Terms of Use and of any modifications you make.


https://alphafoldserver.com
AlphaFold Server BETA
Server
About
FAQs
Remaining jobs: 20
AlphaFold Server allows you to model a structure consisting of many biological molecules
Limited to 20 jobs per day
• Remaining jobs refresh each day
• Jobs can be up to 5,000 tokens - see more details on token calculation, accepted formats, seed selection and other features in our FAQ
• Use the entity bar to chemically modify proteins and nucleic acids
• Get in touch with the AlphaFold team if you have any questions
Explore these examples of structures to see it in action – try them out without using your quota until you begin editing!
Protein-RNA-Ion: PDB 8AW3
Protein-Glycan-Ion: PDB 7BBV
Protein-DNA-Ion: PDB 7RCE
Ok, got it
Upload JSON
Clear
Molecule type
Protein
Copies
1
>Paste sequence or fasta
Input
Enter 1L2Y sequence:
NLYIQWLKDGGPSSGRPPPS

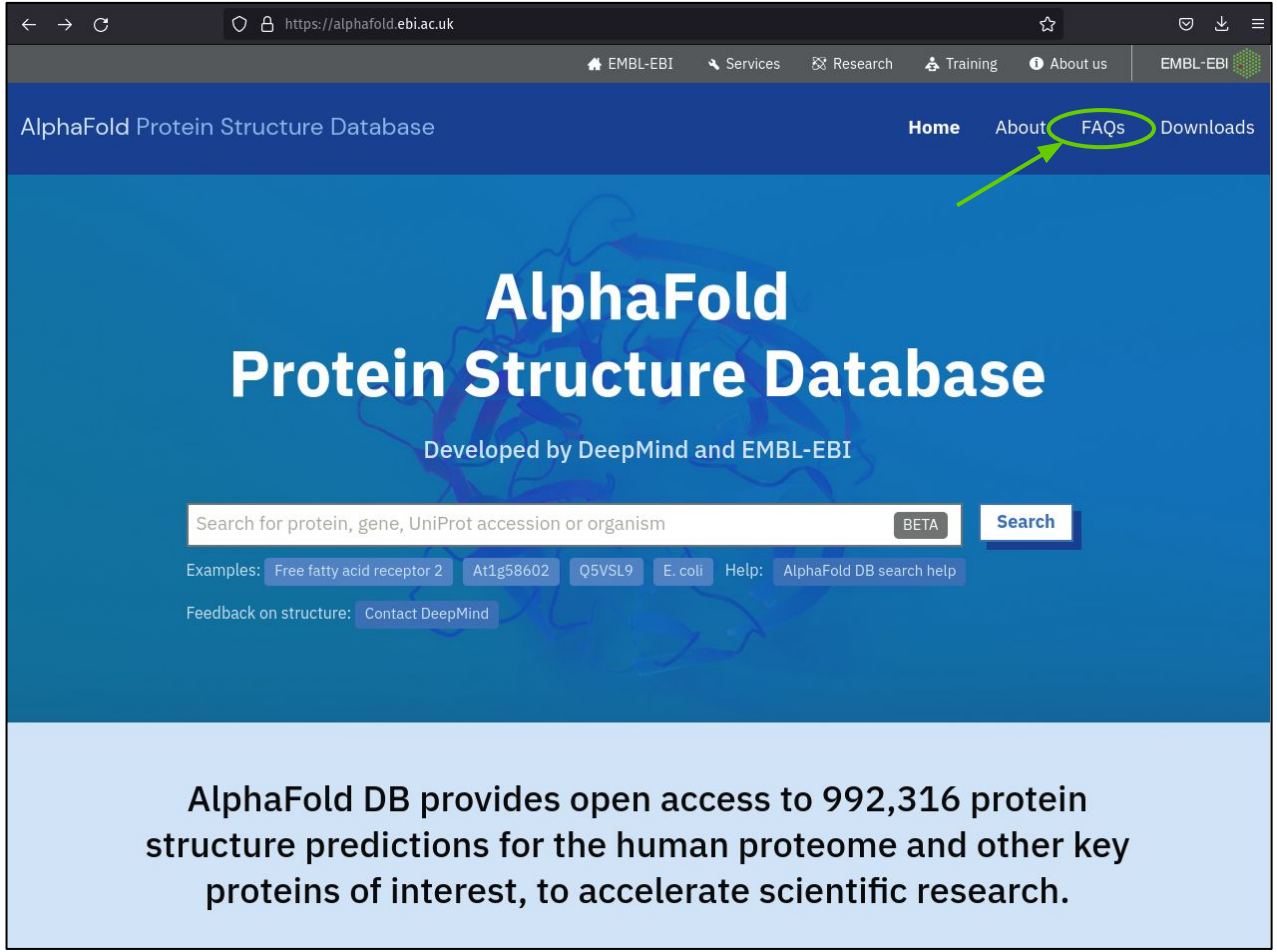
AlphaFold 3 Server Results



Structure Database 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.

Search for your protein to see if it the structure has already been predicted using AlphaFold.



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

Zhong, B, et al. (2021) ParaFold doi.org/10.48550/arXiv.2111.06340

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

HIGH PERFORMANCE RESEARCH COMPUTING
TEXAS A&M UNIVERSITY

https://hprc.tamu.edu

HPRC Helpdesk:

help@hprc.tamu.edu
Phone: 979-845-0219

Take our short course survey!

HPRC Survey
https://u.tamu.edu/hprc_shortcourse_survey

Help us help you. Please include details in your request for support, such as, **Cluster** (ACES, FASTER, Grace, Launch), NetID (UserID), Job information (**JobID**(s), Location of your jobfile, input/output files, Application, Module(s) loaded, Error messages, etc), and Steps you have taken, so we can reproduce the problem.