HIGH PERFORMANCE RESEARCH COMPUTING

ACES: AlphaFold Protein Structure Prediction

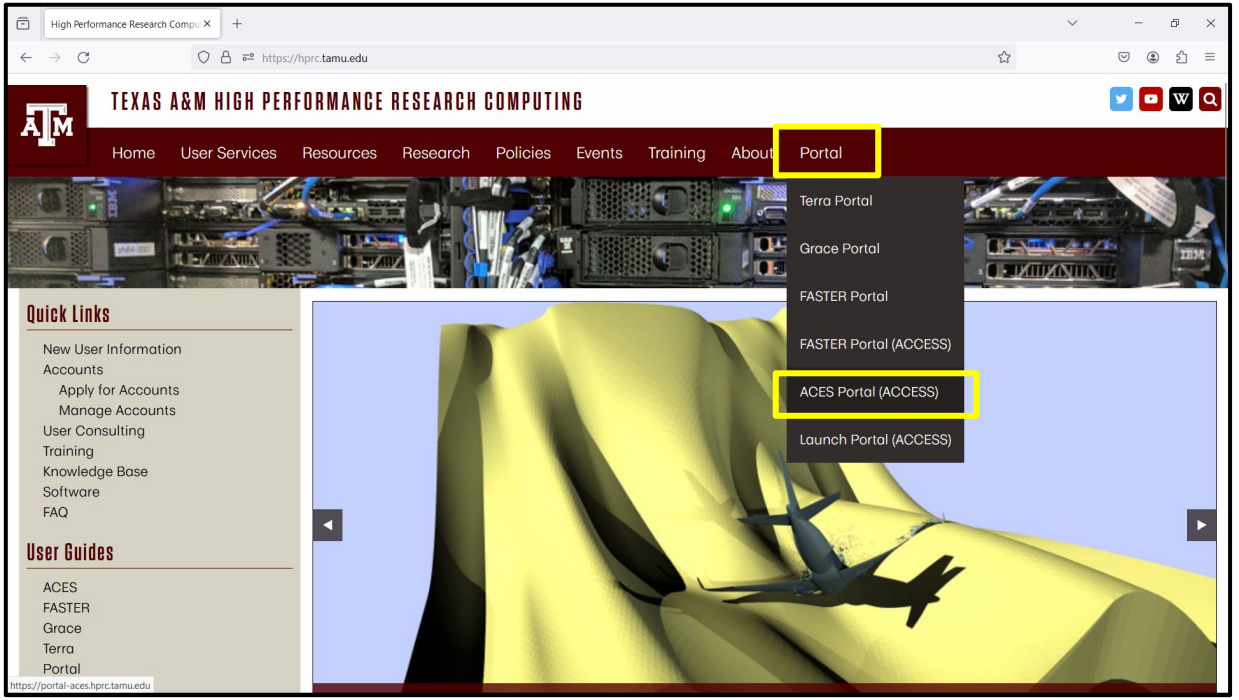
High Performance
Research Computing
DIVISION OF RESEARCH

Spring 2024

ACES: AlphaFold Protein Structure Prediction

- ACES Login and AlphaFold Job Submission
- AlphaFold History
- Selection and Limitations of Resources
- Database Files for Sequence Prediction
- AlphaFold Job Scripts
- AlphaFold Results Visualization
  - job resource monitoring and usage
  - view predicted structures in Jmol
  - plotting pLDDT values
- Alternative Workflows

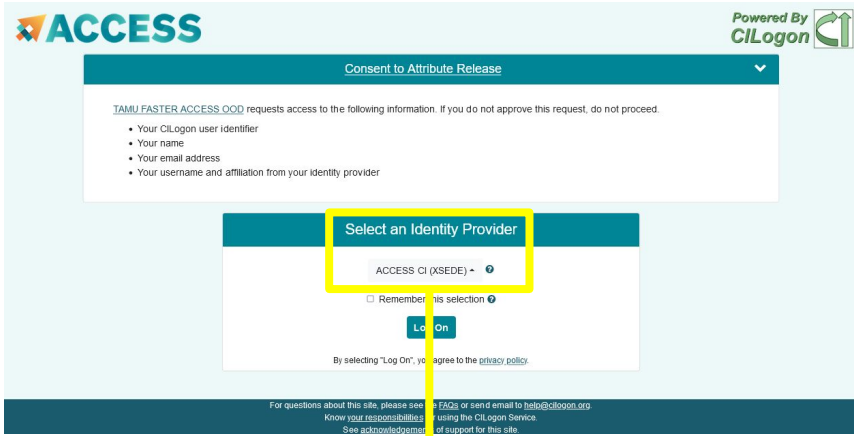
Accessing the HPRC ACES Portal

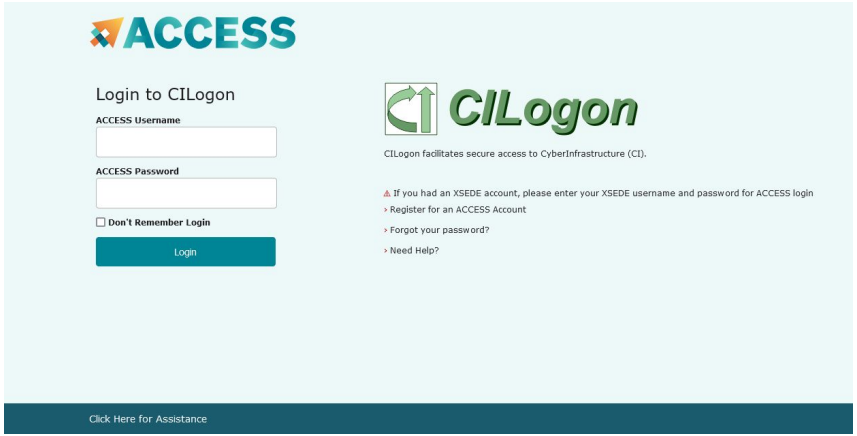


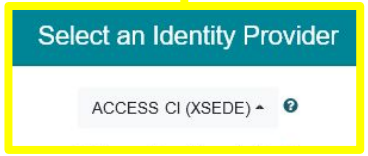
HPRC webpage: hprc.tamu.edu

Accessing ACES via the Portal (ACCESS)

Log-in using your ACCESS credentials.

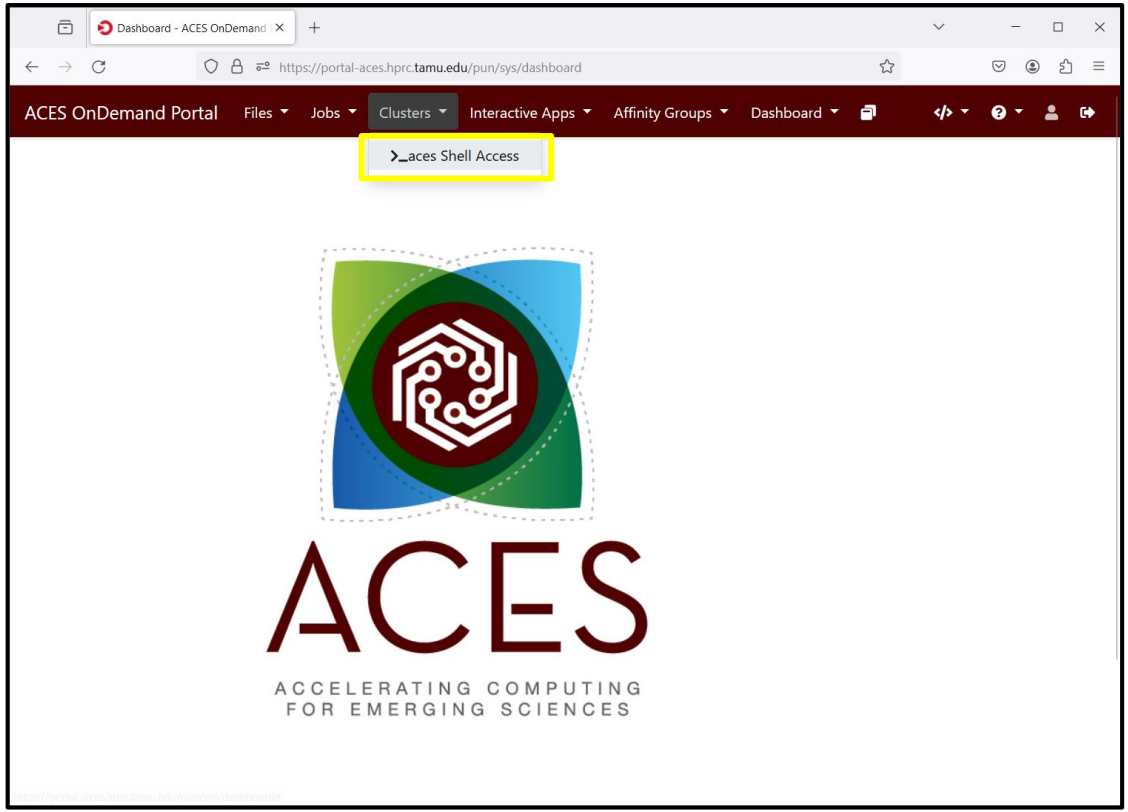






Select the Identity Provider appropriate for your account.

Shell Access via the Portal

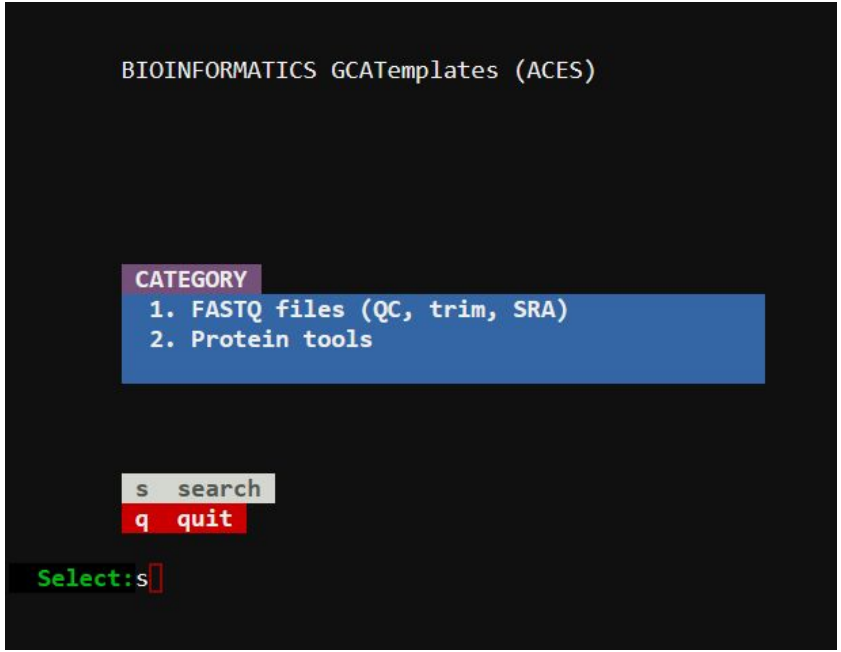


Finding AlphaFold template job scripts using GCATemplates on ACES

- Genomic Computational Analysis Templates are job scripts that use examples input data, which you can run for demo purposes.

```
mkdir $SCRATCH/af_demo
cd $SCRATCH/af_demo
gcatemplates
```

- Type **s** for search, then enter **alphafold** to search for the alphafold **2.3.2** template script, and select the **reduced_dbs** script
- Review the script.



Submit and Monitor the Job

- Submit the job script to the Slurm scheduler.
  - completes in about 25 minutes

```
[username@aces ~]$ sbatch run_alphafold_2.3.2_reduced_dbs_monomer_ptm_h100_aces.sh

Submitted batch job 127863
```

- Monitor the job status.

```
[username@aces ~]$ squeue --me

JOBID   NAME  USER   PARTITION   NODES   CPUS   STATE   TIME   TIME_LEFT  START_TIME          REASON  NODELIST
127863  af2   myid   gpu         1       24     RUNNING 10:16  9:49:44    2024-02-27T15:58    None    ac044
```

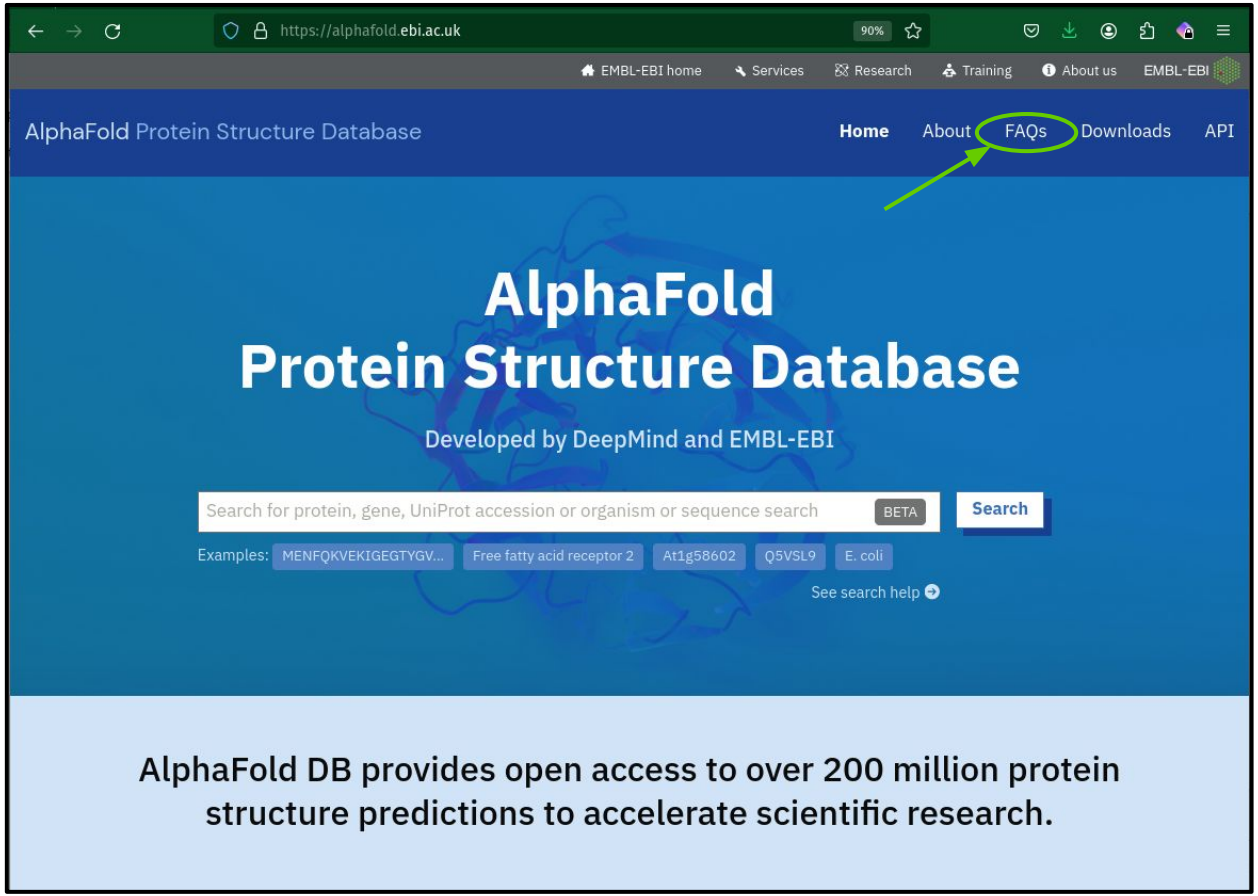
AlphaFold History

- 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

DeepMind and EMBL's European Bioinformatics Institute (EMBL-EBI) have partnered to create the AlphaFold Protein Structure Database to make over 200 million predictions freely available to the scientific community.

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

See the FAQs



Selection and Limitations of Resources

Resource Limitations

- AlphaFold
  - Currently AlphaFold can only utilize one GPU.
  - minimum amino acid length: 16
  - maximum amino acid length:
    - 2,700 proteomes / Swiss-Prot
    - 1,280 all other UniProt
- AlphaFold DeepMind workflow
  - Only about 10% of job runtime is done on GPUs when using DeepMind's workflow.
- ACES Job Script Configuration
  - In your job script, request only ½ of the cores and memory when using 1 x H100 on a GPU node that has 2 x H100 installed so the other H100 GPU on that node is available for other jobs.

Checking GPU Configuration & Availability on ACES

- Use the command line (shell) to see the current GPU configuration and availability
- The GPU configuration can change since ACES is a composable resource cluster
- If there are no GPUs in the AVAILABILITY output, it means that a GPU job that you submit may take a while to start.
- AlphaFold does not support running on PVC GPUs

```
[username@aces ~]$ gpuavail

    CONFIGURATION
NODE           NODE
TYPE           COUNT
--------------------
gpu:pvc:4      16
gpu:h100:2     10
gpu:a30:2      2
gpu:h100:4     2
gpu:pvc:2      1


              AVAILABILITY
NODE   GPU     GPU    GPUs     CPUs   GB MEM
NAME   TYPE    COUNT  AVAIL    AVAIL  AVAIL
-----------------------------------------------
ac041  h100    4      1        87     421
ac045  h100    2      1        88     422
ac051  pvc     2      2        96     488
ac065  a30     2      2        96     488
```

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

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 ACES partition: cpu).

```
[username@aces ~]$ maxconfig

  ACES partitions:   cpu  gpu  pvc  bittware  d5005  memverge  nextsilicon
  ACES GPUs in gpu partition:  a30:2  h100:2  h100:4  pvc:2  pvc:4

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

#!/bin/bash
#SBATCH --job-name=my_job
#SBATCH --time=7-00:00:00
#SBATCH --nodes=1       # max 64 nodes for partition cpu
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=96
#SBATCH --mem=488G
#SBATCH --output=stdout.%x.%j
#SBATCH --error=stderr.%x.%j
```

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

Viewing Maximum Available Resources

See the recommended Slurm parameters for requesting 1 x H100 GPU with ¼ the total CPUs and memory since there are 4 x H100s per node.

```
[username@aces ~]$ maxconfig -g h100 -G 1

  ACES partitions:   cpu  gpu  pvc  bittware  d5005  memverge  nextsilicon
  ACES GPUs in gpu partition:  a30:2  h100:2  h100:4  pvc:2  pvc:4

  Showing 1/4 of total cores and memory for using 1 x h100 GPU

#!/bin/bash
#SBATCH --job-name=my_job
#SBATCH --time=2-00:00:00
#SBATCH --partition=gpu
#SBATCH --nodes=1       # max 8 nodes for partition gpu
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=24
#SBATCH --mem=125G
#SBATCH --gres=gpu:h100:1
#SBATCH --output=stdout.%x.%j
#SBATCH --error=stderr.%x.%j
```

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

AlphaFold Databases for Structure Predictions on ACES

`/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**		

AlphaFold 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 to allow another job to use the other GPU.
  - ACES compute nodes have 488 GB of available memory and 96 cores.

https://hprc.tamu.edu/kb/Software/AlphaFold

```
#!/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=48          # CPUs (threads) per command
#SBATCH --mem=244G                  # total memory per node
#SBATCH --gres=gpu:h100:1           # request 1 H100 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/2.3.2-CUDA-11.8.0

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

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

https://hprc.tamu.edu/kb/Software/AlphaFold

```
#!/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=48          # CPUs (threads) per command
#SBATCH --mem=244G                  # total memory per node
#SBATCH --gres=gpu:h100:1           # request 1 H100 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/2.3.2-CUDA-11.8.0

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

- **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 to allow another job to use the other GPU.

https://hprc.tamu.edu/kb/Software/AlphaFold

```
#!/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=48          # CPUs (threads) per command
#SBATCH --mem=244G                  # total memory per node
#SBATCH --gres=gpu:h100:1           # request 1 H100 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/2.3.2-CUDA-11.8.0

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=48            # CPUs (threads) per command
#SBATCH --mem=244G                    # total memory per node
#SBATCH --gres=gpu:h100:1             # request 1 H100 GPU
#SBATCH --output=stdout.%x.%j         # save stdout to file
#SBATCH --error=stderr.%x.%j          # save stderr to file

# version 2.3.2 has the unified memory variable already configured
module purge
module load GCC/11.3.0  OpenMPI/4.1.4 AlphaFold/2.3.2-CUDA-11.8.0
```

- Unified memory is automatically enabled to request more than just the total GPU memory for the JAX step in AlphaFold.
  - H100 GPU has 80GB memory.
  - `XLA_PYTHON_CLIENT_MEM_FRACTION` (configured to 3.0 in the AlphaFold 2.3.2 module)
- this example script has 240 GB of unified preallocated memory:
  - 80 GB from H100 GPU + 160 GB DDR from motherboard.

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

AlphaFold Runtimes on ACES

- Comparison of GPU vs CPU-only job scripts
- full_dbs

AlphaFold 2.3.2	monomer_ptm 20 aa	multimer 98 & 73 aa
H100 GPU	1 hour 34 minutes	2 hours 25 minutes
CPU-only	1 hour 38 minutes	13 hours 4 minutes

AlphaFold Results Visualization

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

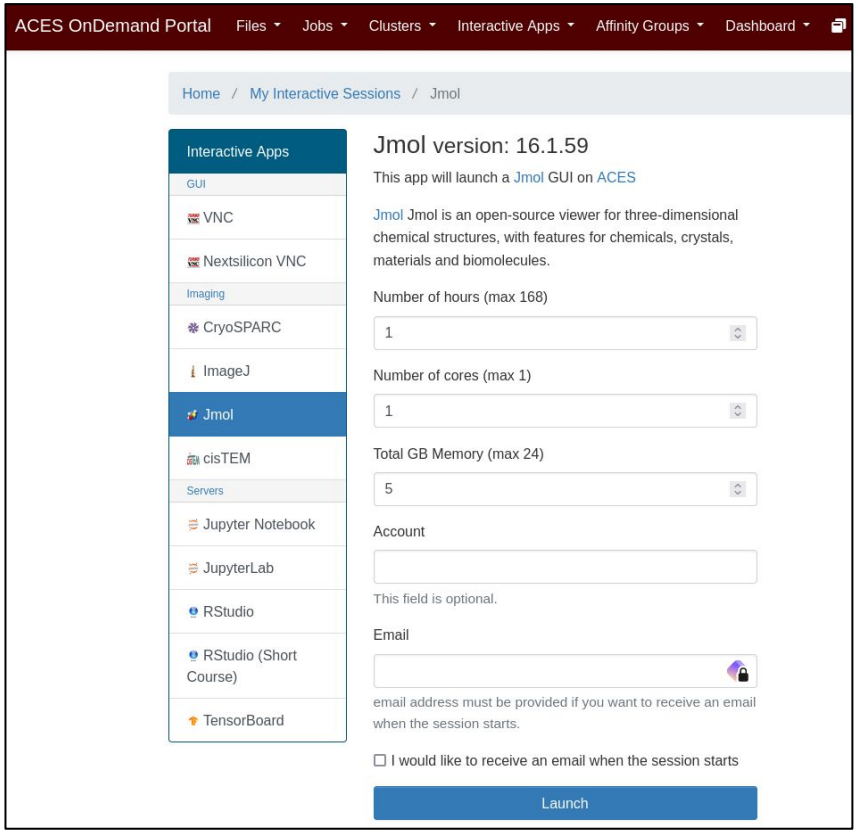
Show Your Job Details using myjob

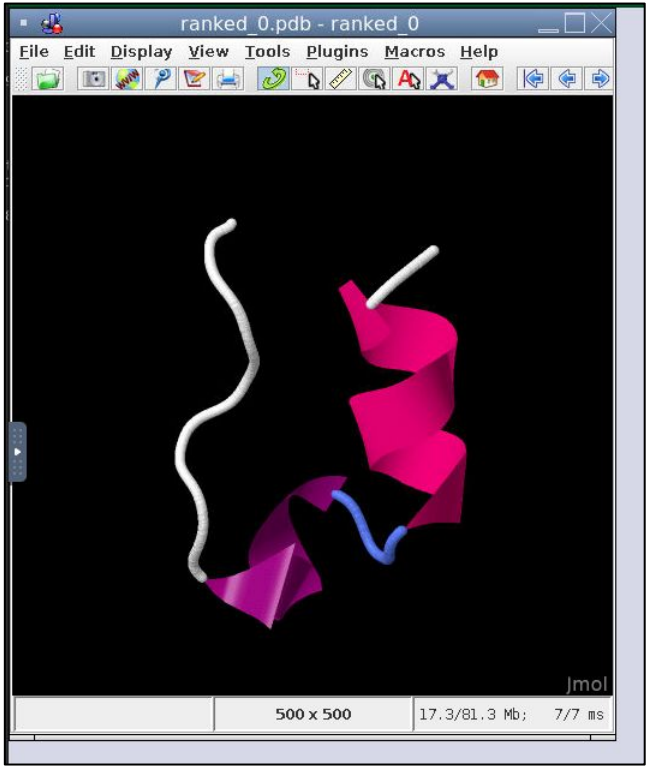
```
[username@aces ~]$ myjob 127863

              Job ID: 127863
             Cluster: aces
          User/Group: userid/userid
               State: COMPLETED (exit code 0)
           Partition: gpu
          Node Count: 1
            NodeList: ac044
      Cores per node: 24
        CPU Utilized: 00:27:34
      CPU Efficiency: 6.02% of 07:38:00 core-walltime
         Submit time: 2024-02-27 15:57:47
          Start time: 2024-02-27 15:58:03
            End time: 2024-02-27 16:17:08
 Job Wall-clock time: 00:19:05
     Memory Utilized: 11.22 GB
   Memory Efficiency: 9.19% of 122.00 GB
            Job Name: alphafold-2.3.2
Job Submit Directory: /scratch/user/userid/classes/alphafold
         Submit Line: sbatch run_alphafold_2.3.2_h100_reduced_dbs_monomer_aces.sh
```

use the -h flag to view usage

```
myjob -h
```

Visualize AlphaFold Results with Jmol on the ACES Portal





AlphaFold Confidence Metrics

AlphaPickle for Visualization of Confidence Scores

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

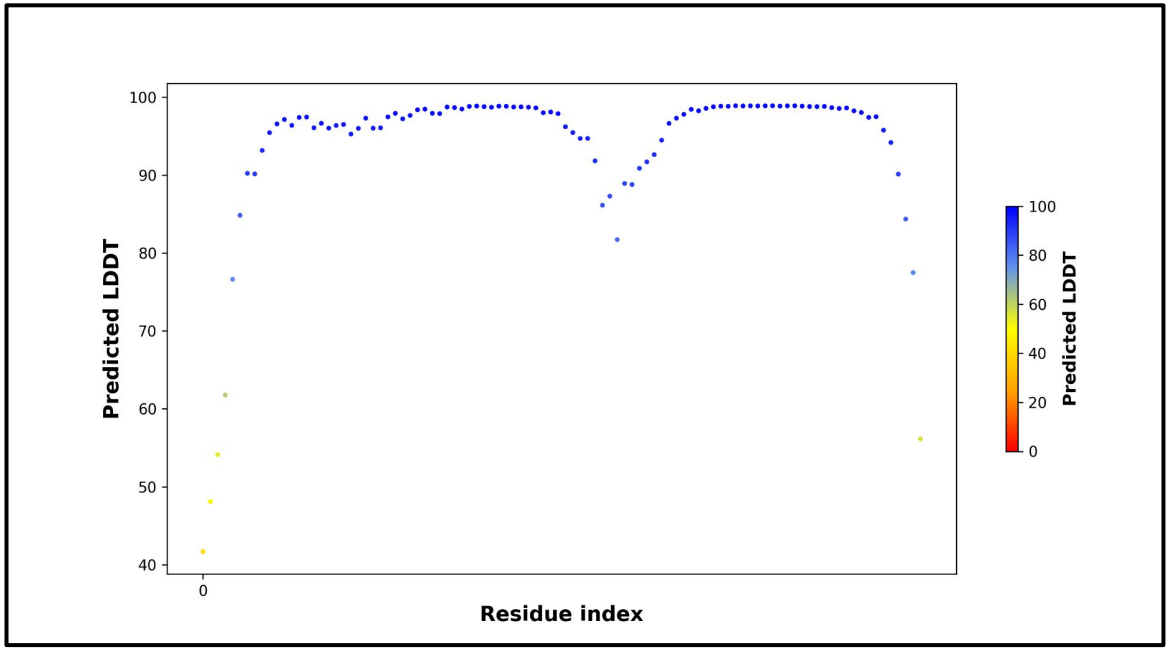
- Graphing PAE scores is only available for the **monomer_ptm** and **multimer** model presets.
- Load the AlphaPickle module at the beginning of the job script.
- Run AlphaPickle at the end, specifying the output directory used in the run_alphafold.py command.

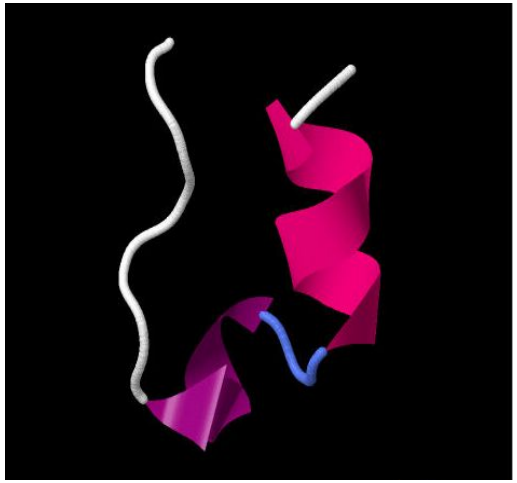
- pLDDT: scale from 0 - 100 of per-residue estimate of prediction confidence
- PAE: Predicted Alignment Error

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=48          # CPUs (threads) per command
#SBATCH --mem=244G                  # total memory per node
#SBATCH --gres=gpu:h100:1           # request 1 H100 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/2.3.2-CUDA-11.8.0
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_2.3.2 \
 --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_2.3.2/1L2Y

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

Visualize AlphaFold pLDDT Scores



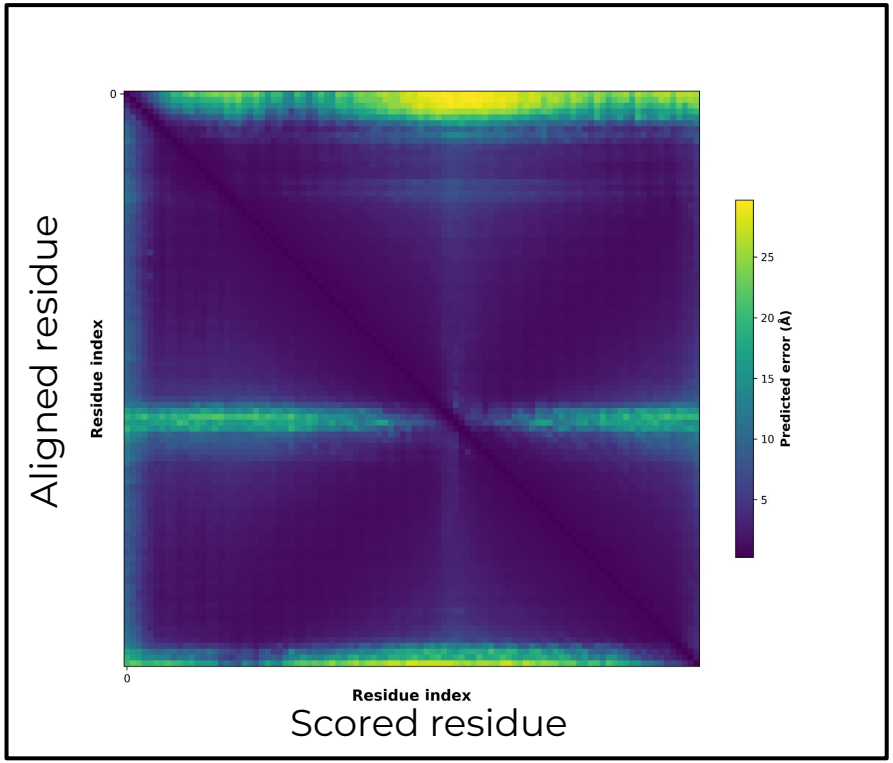


`out_1L2Y_monomer_ptm_reduced_dbs/1L2Y/ranked_0_pLDDT.png`

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

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

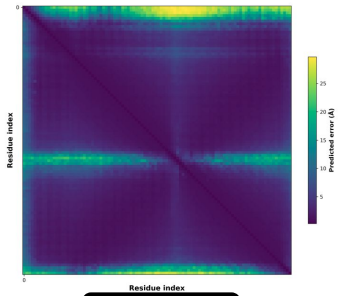
Visualize AlphaFold PAE Results (monomer_ptm)



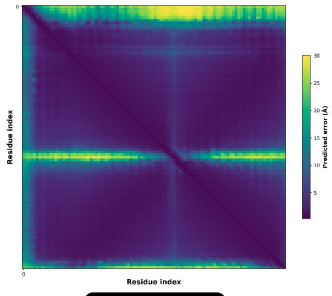
- Low Predicted Aligned Error (PAE) value has a higher confidence of accuracy.
- Must use monomer_ptm or multimer as model_preset to create PAE image
- The color at position (x, y) indicates AlphaFold's expected position error at residue x, when the predicted and true structures are aligned on residue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

`out_1L2Y_monomer_ptm_reduced_dbs/1L2Y/ranked_0_PAE.png`

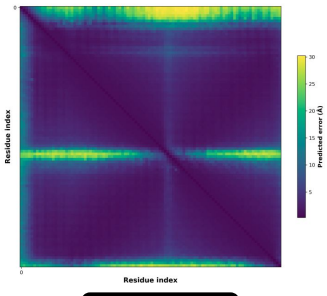
Evaluating Models



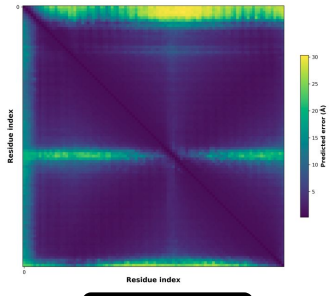
ranked_0



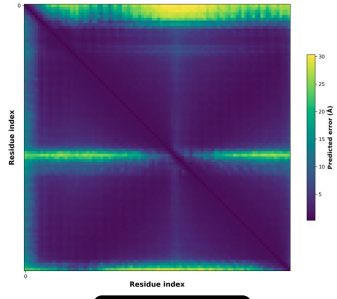
ranked_1



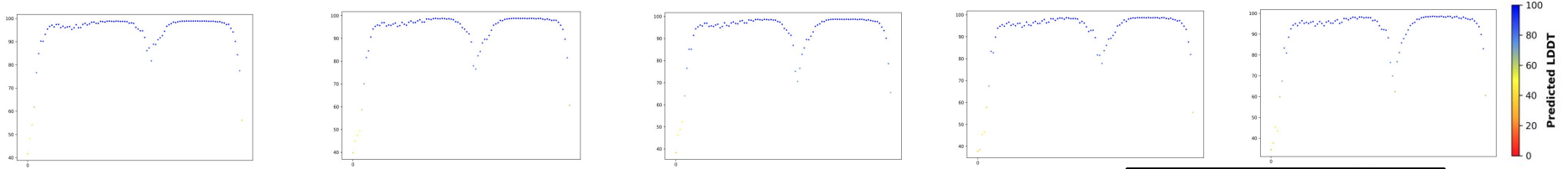
ranked_2



ranked_3



ranked_4



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

```
cat out_1L2Y_monomer_ptm_reduced_dbs/IL2Y/ranking_debug.json
```


```
"plddts": {
  "model_1_ptm_pred_0": 94.14318865567142,
  "model_2_ptm_pred_0": 94.83124839667653,
  "model_3_ptm_pred_0": 90.41209232774796,
  "model_4_ptm_pred_0": 90.73102198891624,
  "model_5_ptm_pred_0": 92.6319870089253
},
"order": [
  "model_2_ptm_pred_0",
  "model_1_ptm_pred_0",
  "model_5_ptm_pred_0",
  "model_4_ptm_pred_0",
  "model_3_ptm_pred_0"
]
```


ranked_0

ranked_4

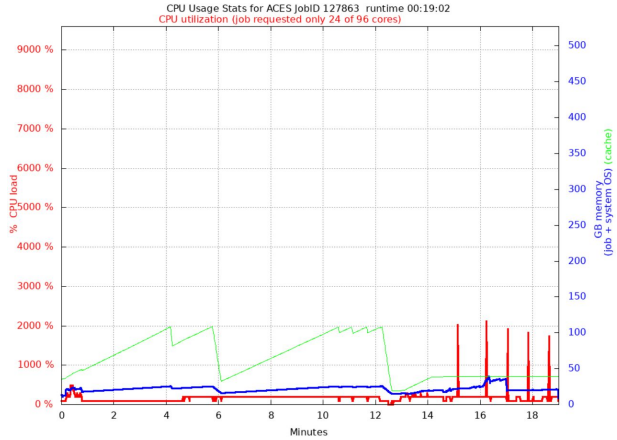
AlphaFold Job Resource Monitoring

Review GPU and CPU usage for a Job

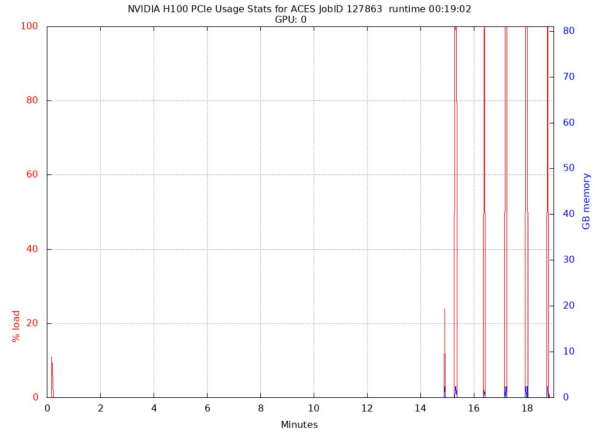
The **jobstats** command monitors GPU and CPU resource usage and create graphs.

```
#!/bin/bash
#SBATCH --job-name=my_gpu_job
#SBATCH --time=1-00:00:00
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=48
#SBATCH --mem=244G
#SBATCH --gres=gpu:h100:1
#SBATCH --output=stdout.%x.%j
#SBATCH --error=stderr.%x.%j
module purge
module load GCC/11.3.0  OpenMPI/4.1.4
module load AlphaFold/2.3.2-CUDA-11.8.0

# 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
```

view images in Portal Files app



stats_cpu.127863.png



stats_gpu.127863.png

- CPU stats are only accurate for jobs using the entire compute node resources (CPUs, memory), but we primarily want to make sure GPUs were used.
- GPU stats are accurate if using fewer than max CPUs and memory because your job will be the only job running on the requested GPU.

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

AlphaFold Workflow Alternatives

ParaFold (ParallelFold)

- The ParaFold module uses the same AlphaFold installation as the AlphaFold module
- 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
- 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

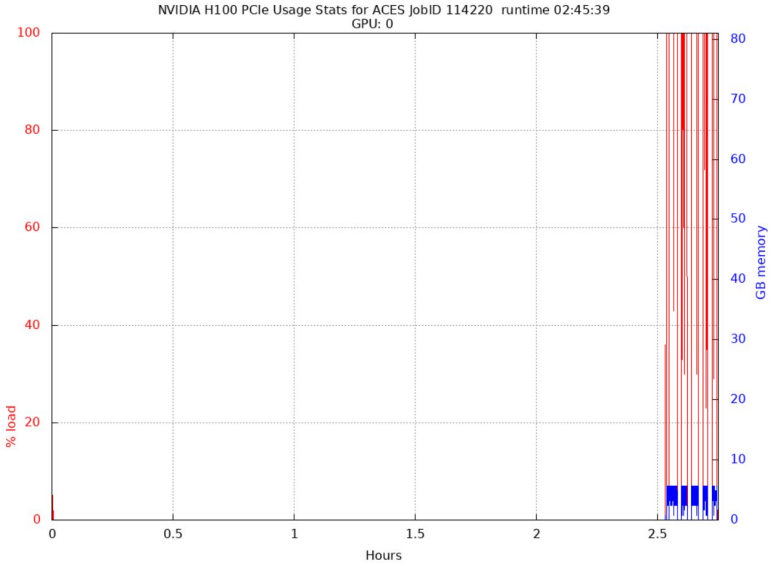
Example ParaFold Job Script

```
#!/bin/bash
#SBATCH --job-name=parafold-cpu     # job name
#SBATCH --time=7-00:00:00           # max job run time dd-hh:mm:ss
#SBATCH --ntasks-per-node=1         # tasks (commands) per compute node
#SBATCH --cpus-per-task=48          # CPUs (threads) per command
#SBATCH --mem=244G                  # total memory per node
#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/2.3.2-CUDA-11.8.0
module load ParaFold/2.0-CUDA-11.8.0

ALPHAFOLD_DATA_DIR=/scratch/data/bio/alphafold/2.3.2
protein_fasta=/scratch/data/bio/alphafold/example_data/T1083_T1084_multimer.fasta

# First, run CPU-only steps to get multiple sequence alignments
run_alphafold.sh -d $ALPHAFOLD_DATA_DIR -o pf_output_dir -p multimer -i $protein_fasta -t 2024-1-1 -f

# Second, run GPU steps as a separate job after the first part completes successfully
sbatch --job-name=parafold-gpu --time=2-00:00:00 --ntasks-per-node=1 --cpus-per-task=24 --mem=122G \
--gres=gpu:h100:1 --partition=gpu --output=stdout.%x.%j --error=stderr.%x.%j \
--dependency=afterok:$SLURM_JOBID<<EOF
#!/bin/bash
module purge
module load GCC/11.3.0  OpenMPI/4.1.4 AlphaFold/2.3.2-CUDA-11.8.0
module load ParaFold/2.0-CUDA-11.8.0 AlphaPickle/1.4.1
jobstats -i 1 &
run_alphafold.sh -g -u 0 -d $ALPHAFOLD_DATA_DIR -o pf_output_dir -p multimer -i $protein_fasta -t 2024-1-1
# graph pLDDT and PAE .pkl files
run_AlphaPickle.py -od pf_output_dir/T1083_T1084_multimer
jobstats
EOF
```

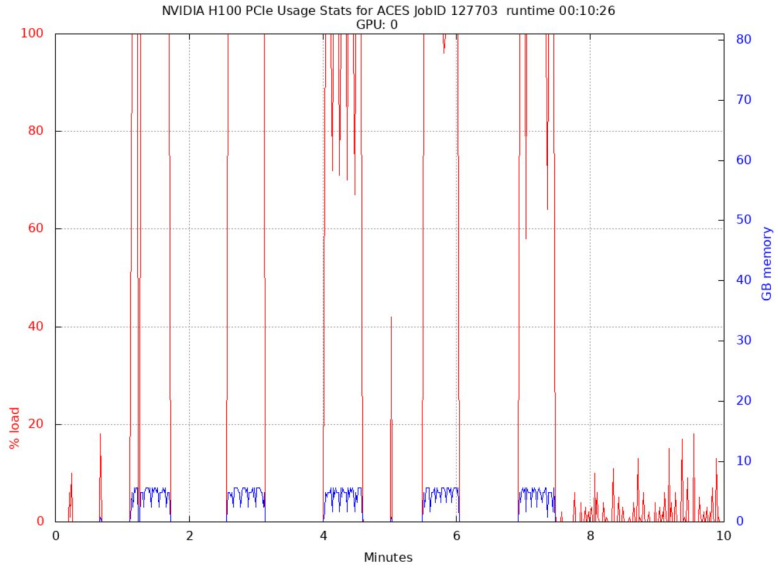
Comparison of DeepMind vs ParaFold Workflows

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

DeepMind Workflow



ParaFold Workflow (GPU job)



The first job of the ParaFold workflow (CPU-only) completed in 3 hours

DeepMind vs ParaFold Workflows

- Compare the runtimes and TM-scores of the DeepMind vs Parafold workflows
- Use gcatemplates to get the AlphaFold multimer and ParaFold multimer scripts for version 2.3.2.
- The ParaFold job script is configured to submit a CPU job that will start a separate GPU job when the CPU job is complete
- Use a multimer sequence file of your choice or use the preconfigured script.
- Launch each job and review the final runtimes and TM-scores found in the ranking_debug.json file in the output directory

AlphaFold 2.3.2	Runtime	Highest Scoring Model	TM-score
ParaFold	? hrs ? min*	?	?
DeepMind	? hrs ? min	?	?

***** combined time for the CPU job and GPU job

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

HPRC ACES Support

First check the KnowledgeBase Documentation hprc.tamu.edu/kb

- ACES User Guide hprc.tamu.edu/kb/User-Guides/ACES
- Email your questions to help@hprc.tamu.edu

Help us help you -- we need more info

- Which **Cluster**
- Username
- **JobID**(s) if any
- Location of your jobfile, input/output files
- Software Application used if any
- Module(s) loaded if any
- Error messages
- Steps you have taken, so we can reproduce the problem

Let us know when the issue has been resolved so we can close the helpdesk ticket.

High Performance
Research Computing

DIVISION OF RESEARCH

Thank you

Questions?