HIGH PERFORMANCE RESEARCH COMPUTING

Introduction to AlphaFold for 3D Protein Structure Prediction on ACES

Texas A&M Research Computing Symposium

May 21, 2024

High Performance
Research Computing

DIVISION OF RESEARCH

AlphaFold for 3D Protein Structure Prediction on ACES

1. Running AlphaFold
   - Resources and Limitations
   - Database Files
   - Submit an AlphaFold job
     - ParaFold on ACES
     - Google Colab
2. Introduction to AlphaFold (Dr. Devon Boland)
3. Visualization of Results
   - job resource usage
   - view predictions in Jmol
   - plotting pLDDT values
   - efficient GPU Usage

Resource Limitations

- AlphaFold
  - currently AlphaFold can only utilize one GPU
  - about 90% of processing is done on CPU
    - multiple sequence alignments are CPU-only
    - structure prediction step is on GPU
- AlphaFold in Google Colab (web browser)
  - 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 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**		

Resources for Running AlphaFold

- Login to ACES using your ACCESS ID
  - https://hprc.tamu.edu/kb/Helpful-Pages/ACCESS-ID
  - https://hprc.tamu.edu/kb/User-Guides/ACES/Access
- Run as a Slurm job script on ACES
  - https://portal-aces.hprc.tamu.edu
  - use ParaFlow workflow
    - first job of multiple sequence alignments is run on CPU-only
    - second job of structure predictions is run on a GPU node
    - the ParaFlow software module also contains the AlphaFold/2.3.2 software module
    - reduced_dbs parameter is not supported yet by ParaFold
- Run as a Jupyter Notebook on Google Colab in web browser
  - Google account required

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

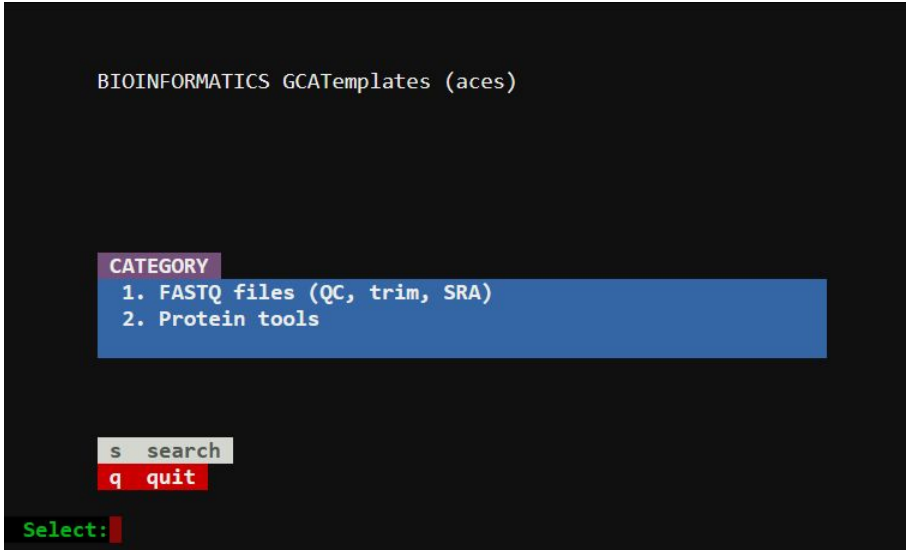
AlphaFold ACES Job Scripts

Finding AlphaFold template job scripts using GCATemplates on ACES

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

```
mkdir $SCRATCH/rcs2024
cd $SCRATCH/rcs2024
gcatemplates
```

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




```
sbatch run_parafold_alphafold_2.3.2_monomer_ptm_h100_aces.sh
```

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=24          # CPUs (threads) per command
#SBATCH --mem=122G                  # 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
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/1L2Y.fasta

# First, run CPU-only steps to get multiple sequence alignments
run_alphafold.sh -d $ALPHAFOLD_DATA_DIR -o pf_output_dir -p monomer_ptm -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
module load ParaFold/2.0-CUDA-11.8.0 AlphaPickle/1.4.1
jobstats &
run_alphafold.sh -g -u 0 -d $ALPHAFOLD_DATA_DIR -o pf_output_dir -p monomer_ptm -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
```

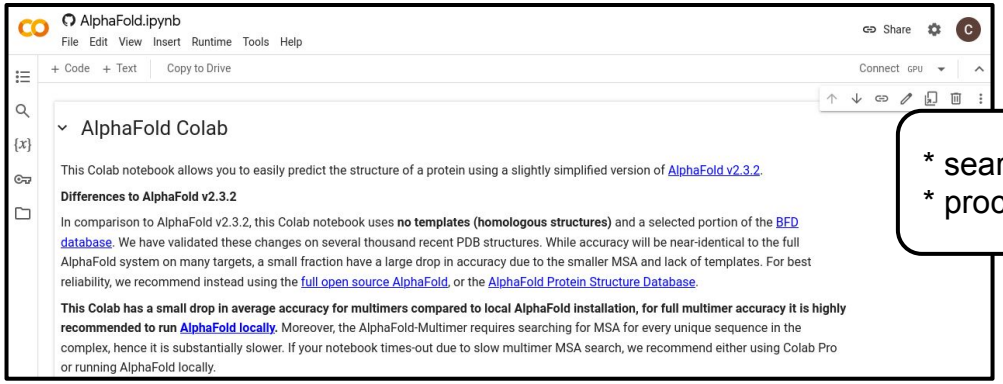
Unified Memory

```
#!/bin/bash
#SBATCH --job-name=parafold-gpu     # 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 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 ParaFold/2.0-CUDA-11.8.0
```

- Unified memory can be used 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`
    - automatically configured to 3.0 in the ParaFold 2.0-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.

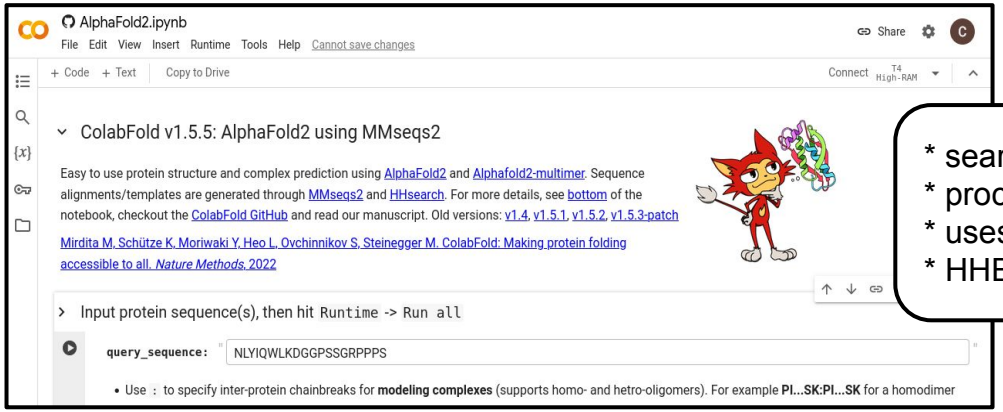
https://jax.readthedocs.io/en/latest/gpu_memory_allocation.html

AlphaFold Colab or ColabFold Jupyter Notebooks



* search for **alphafold deepmind colab**
* processing time approximately 45 minutes for this sequence

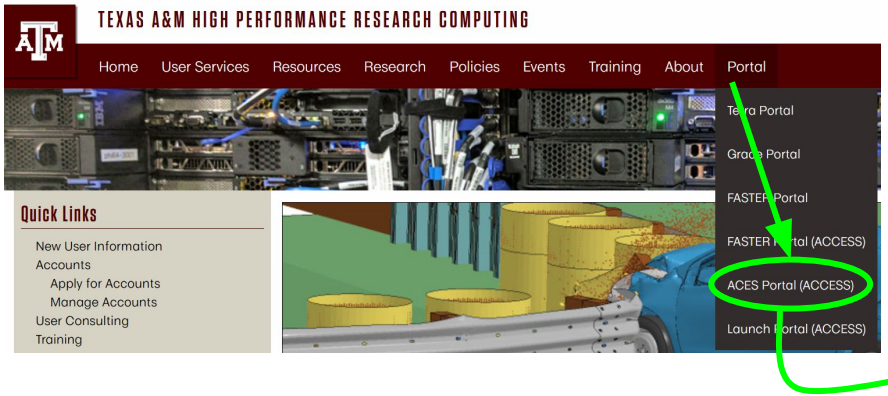
Enter 1L2Y amino acid sequence
NLYIQWLKDGGPSSGRPPPS
Click Runtime -> Run all

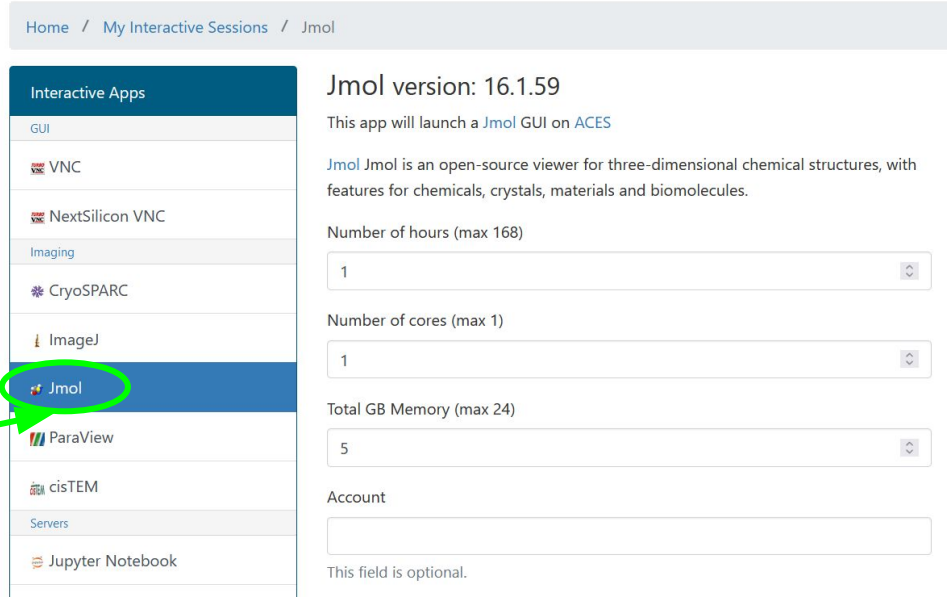


* search for **alphafold sokrypton colabfold**
* processing time approximately 5 min for this sequence
* uses existing predictions if available
* HHBlits and BLAST replaced by MMSeq2

Jmol Portal App

- Launch a Jmol job on the ACES portal portal-aces.hprc.tamu.edu
- Jmol will be used later to visualize results from an ParaFold/AlphaFold job





Running AlphaFold DeepMind Workflow on Grace

This example was run as a job script requesting one GPU

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

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  memverge  nextsilicon
  ACES GPUs in gpu partition:  a30:2  h100:2  h100:4  h100:8  pvc:2  pvc:4  pvc:8

  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
```

Viewing Maximum Available Resources

See the recommended Slurm parameters for requesting 1 x H100 GPU with a fraction the total CPUs and memory since there are multiple H100 GPUs per node

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

  ACES partitions:   cpu  gpu  pvc  bittware  memverge  nextsilicon
  ACES GPUs in gpu partition:  a30:2  h100:2  h100:4  h100:8  pvc:2  pvc:4  pvc:8

  Showing 1/8 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=12
#SBATCH --mem=61G
#SBATCH --gres=gpu:h100:1
#SBATCH --output=stdout.%x.%j
#SBATCH --error=stderr.%x.%j
```

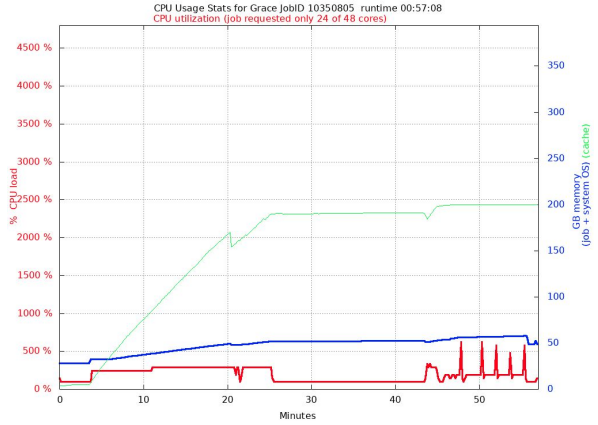
AlphaFold Results Visualization

Review GPU and CPU usage for a Job

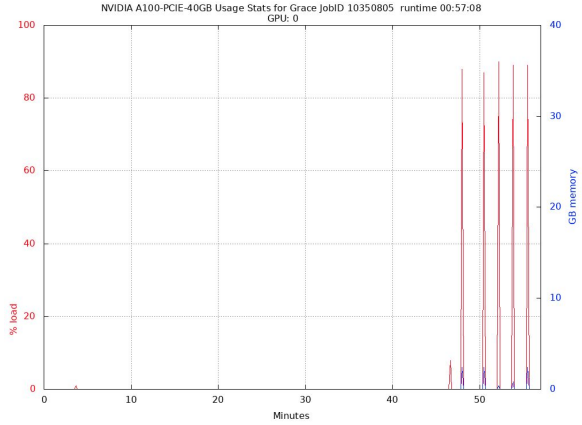
The **jobstats** command monitors GPU and CPU resource usage and create graphs.

```
#!/bin/bash
#SBATCH --job-name=my_job
#SBATCH --time=1-00:00:00
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=24
#SBATCH --mem=122G
#SBATCH --gres=gpu:h100:1
#SBATCH --output=stdout.%x.%j
#SBATCH --error=stderr.%x.%j

module purge
module load sw_modules
# 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.10350805.png



stats_gpu.10350805.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

Monitor a Running Job

The **myjob** command displays information about your job including time elapsed since starting

```
[username@aces ~]$ myjob 154568

              Job ID: 154568
             Cluster: aces
          User/Group: username/username
               State: RUNNING
           Partition: cpu
          Node Count: 1
            NodeList: ac014
      Cores per node: 24
      CPU Efficiency: 0.00% of 15:37:36 core-walltime
         Submit time: 2024-05-20 09:48:14
          Start time: 2024-05-20 09:48:16
            End time: Unknown
          Time limit: 10:00:00
        Time elapsed: 00:39:04
      Time remaining: 9:20:56
   Memory Efficiency: 0.00% of 122.00 GB (122.00 GB/node)
             WARNING: Efficiency statistics may be misleading for RUNNING jobs.
            Job Name: parafold-cpu
Job Submit Directory: /scratch/user/username/templates/aces/dev/parafold/rcs2024
         Submit Line: sbatch run_parafold_alphafold_2.3.2_monomer_ptm_h100_aces.sh
```

Review a Completed Job

The **myjob** command displays CPU resource usage and provides more information than just using **seff**

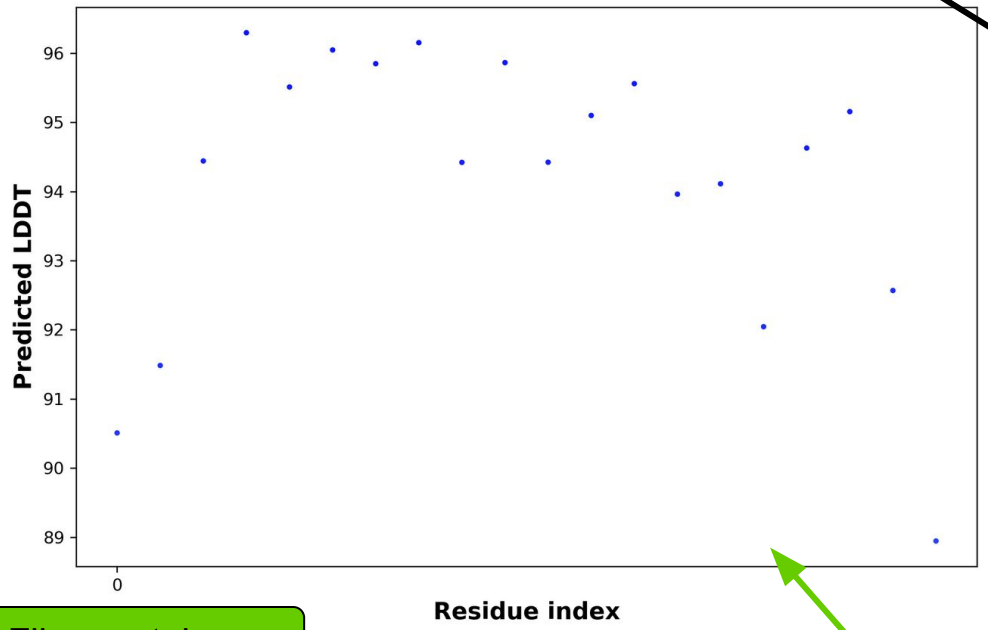
```
[username@aces ~]$ myjob 154568

               Job ID: 154568
              Cluster: aces
           User/Group: username/username
                State: COMPLETED (exit code 0)
            Partition: cpu
           Node Count: 1
             NodeList: ac014
      Cores per node: 24
        CPU Utilized: 00:26:57
      CPU Efficiency: 0.29% of 6-08:34:24 core-walltime
         Submit time: 2024-05-20 09:48:14
          Start time: 2024-05-20 09:48:16
            End time: 2024-05-20 16:09:42
 Job Wall-clock time: 06:21:26
     Memory Utilized: 17.26 GB
   Memory Efficiency: 14.15% of 122.00 GB
            Job Name: parafold-cpu
Job Submit Directory: /scratch/user/username/templates/aces/dev/parafold/rcs2024
         Submit Line: sbatch run_parafold_alphafold_2.3.2_monomer_ptm_h100_aces.sh
```

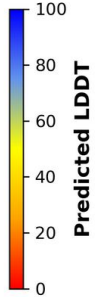
AlphaFold Confidence Metrics

Visualize AlphaFold Results

Jmol portal app

```
/scratch/training/parafold/monomer_ptm/out_parafold_1L2Y_monomer_ptm/1L2Y/ranked_0.pdb
```



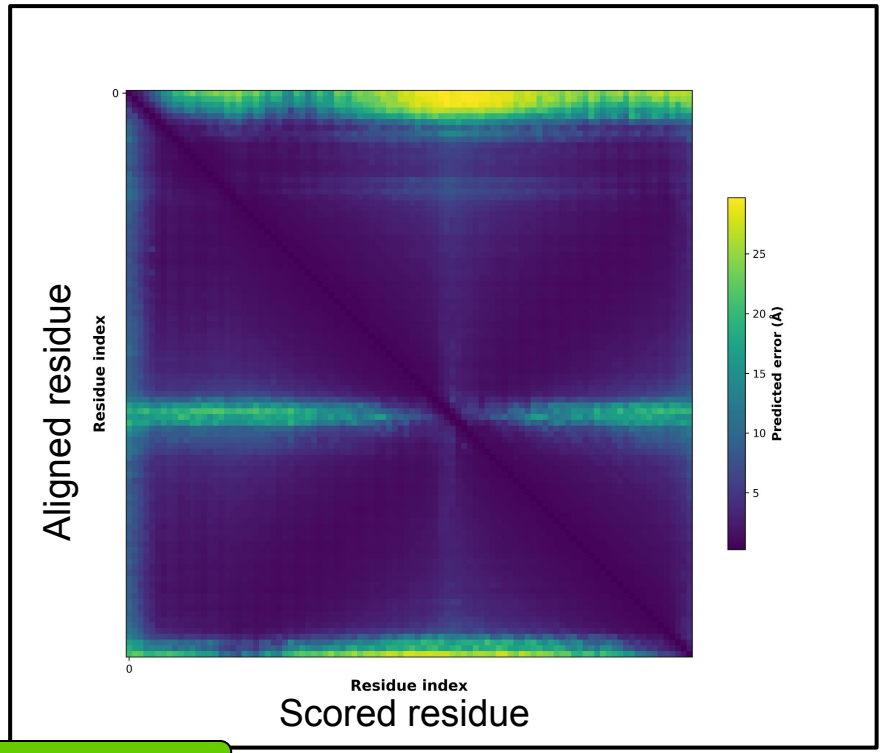


> 90 = Very high
70 - 90 = Confident
50 - 70 = Low
< 50 = Very low

Files portal app

```
/scratch/training/parafold/monomer_ptm/out_parafold_1L2Y_monomer_ptm/1L2Y/ranked_0_pLDDT.png
```

Visualize AlphaFold PAE Results (monomer_ptm)



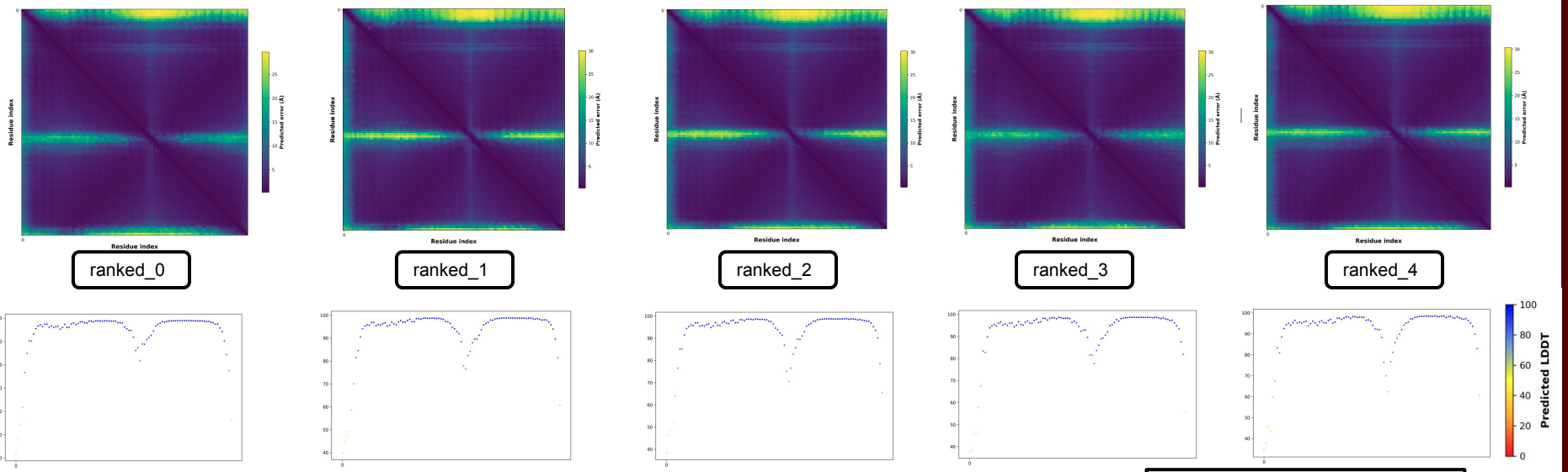
- Low Predicted Aligned Error (PAE) value has higher confidence of accuracy
- Must use monomer_ptm or multimer as model_preset to create PAE image
- The colour at position (x, y) indicates AlphaFold's expected position error at residue x, when the predicted and true structures are aligned on residue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

https://alphafold.ebi.ac.uk

Portal Files app

```
/scratch/training/parafold/monomer_ptm/out_parafold_1L2Y_monomer_ptm/1L2Y/ranked_0_PAE.png
```

Evaluating Models



see which model has the top rank based on pLDDT score

Portal Shell access

```
cat out_parafold_1L2Y_monomer_ptm/1L2Y/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

ParaFold Results

ParaFold (ParallelFold)

- The ParaFold software module includes the AlphaFold software 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

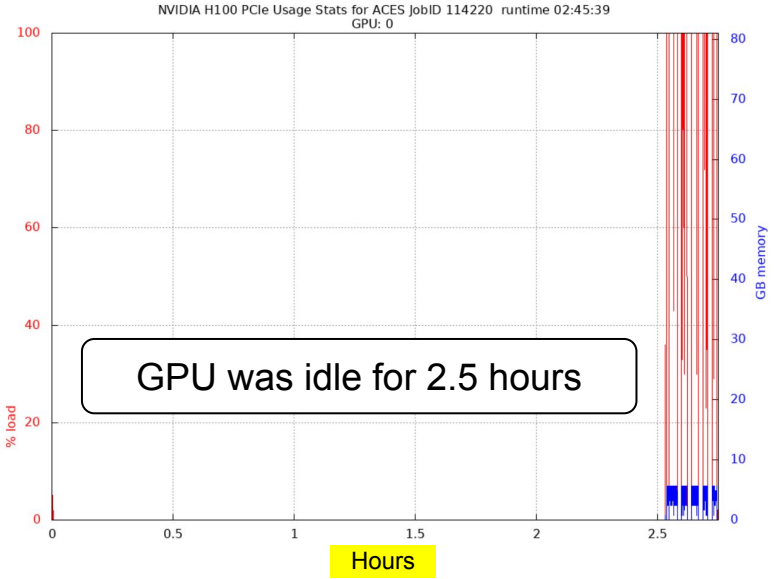
* ACES cluster
** combined time for the separate CPU 3 hour job and GPU 10 min job
*** measure of similarity between two protein structures

https://github.com/Zuricho/ParallelFold

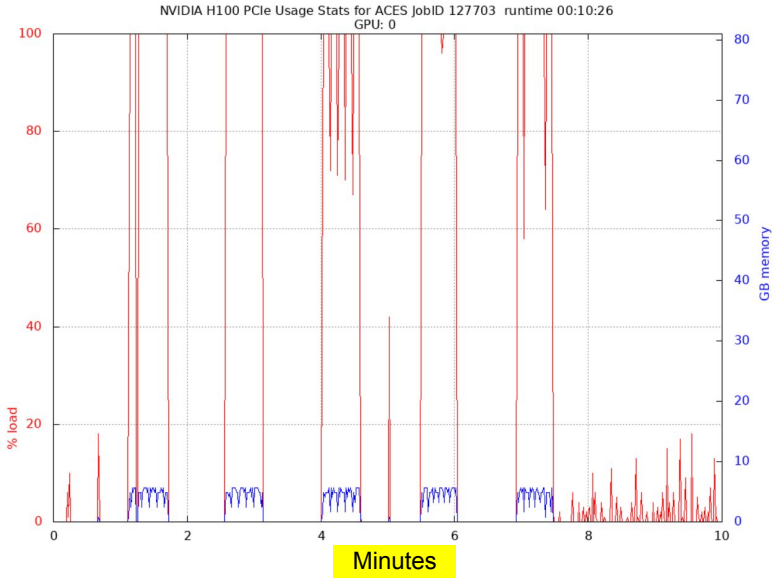
Comparison of DeepMind vs ParaFold Workflows

- AlphaFold DeepMind's workflow (1 job on a GPU node) 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



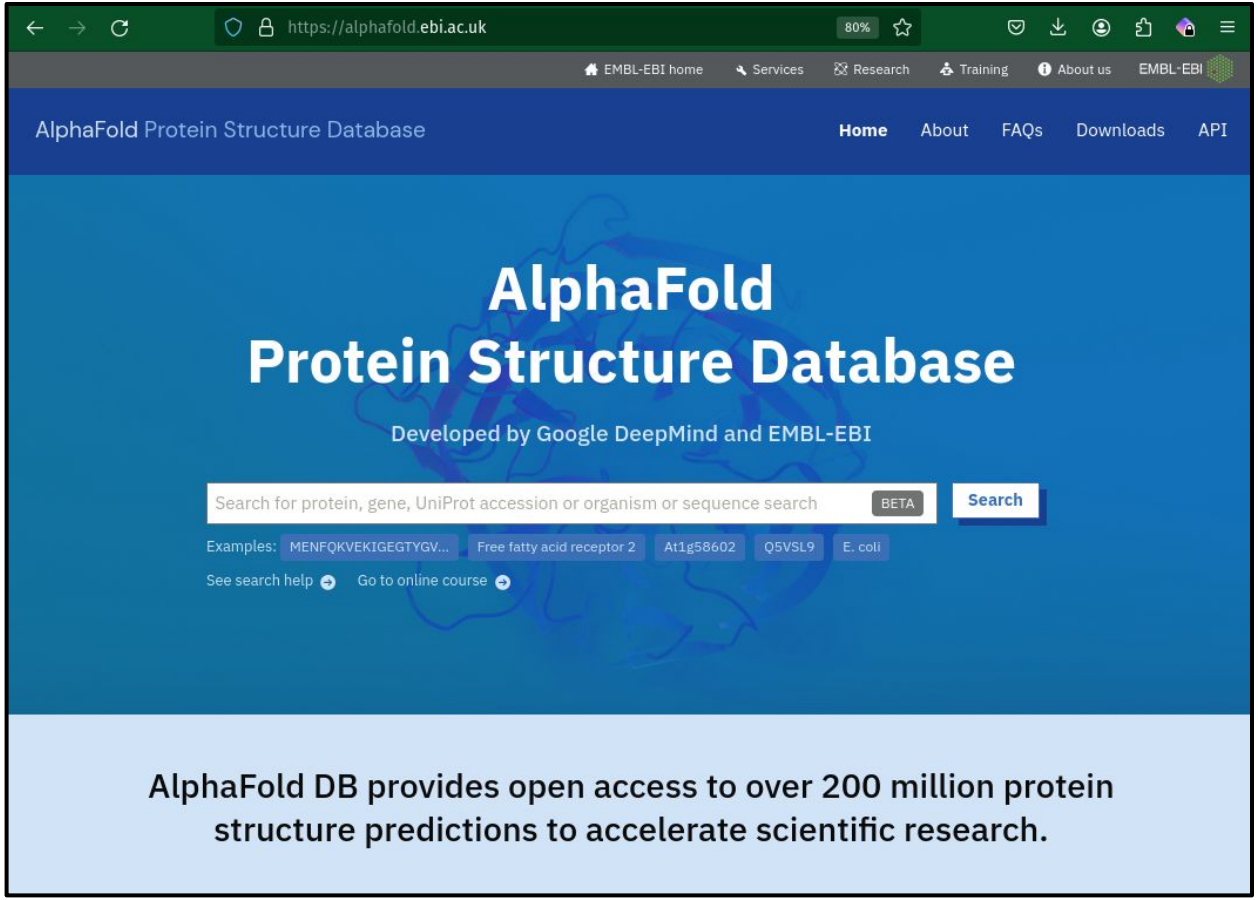
ParaFold Workflow (GPU job)



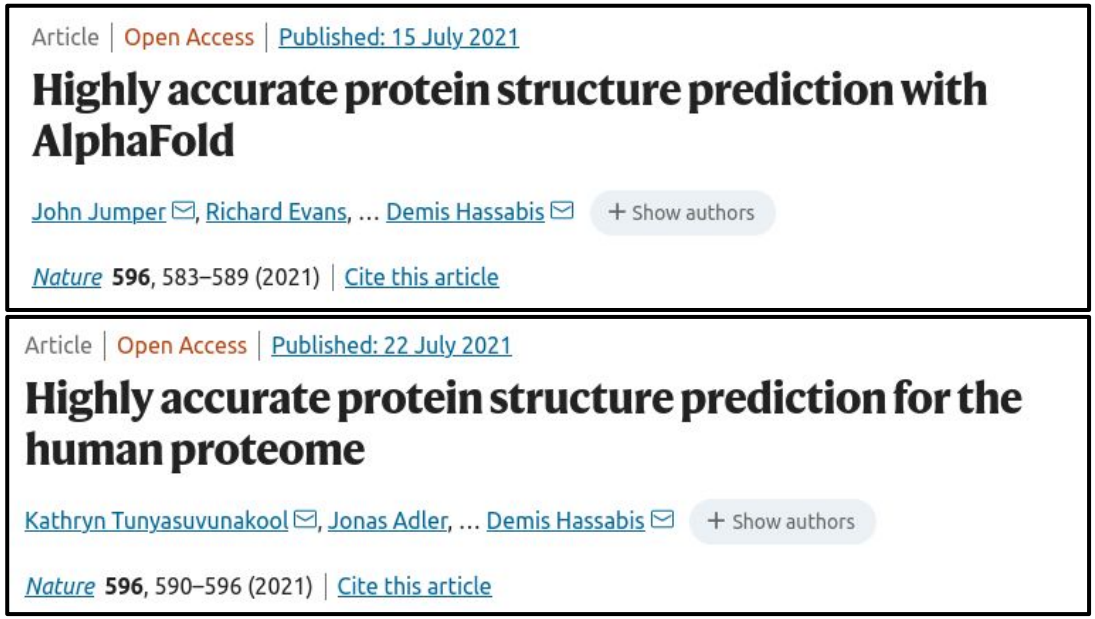
The first job of the ParaFold workflow (CPU-only) completed in 3 hours

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

Bozitao Zhong, Xiaoming Su, Minhua Wen, Sichen Zuo, Liang Hong, James Lin. ParaFold: Paralleling AlphaFold for Large-Scale Predictions. 2021. arXiv:2111.06340. doi.org/10.48550/arXiv.2111.06340

HPRC Resources

- Free Help
  - Send an email to help@hprc.tamu.edu if you have any questions regarding Bioinformatics tools usage on HPRC clusters or to schedule a Zoom or in-person visit.
    - First spend some time investigating the error.
      - use the **`myjob`** utility
      - read log files, stdout file, stderr file, tool manual
      - Google search
      - Google user groups: many are software specific
    - Include details about your issue.
      - Which **cluster** you are using
      - The **JobID**
      - Which software you are using
      - Which modules you have loaded
      - Error messages you are seeing

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

- HPRC NGS data analysis tools Documentation
  - https://hprc.tamu.edu/kb/Software/Bioinformatics