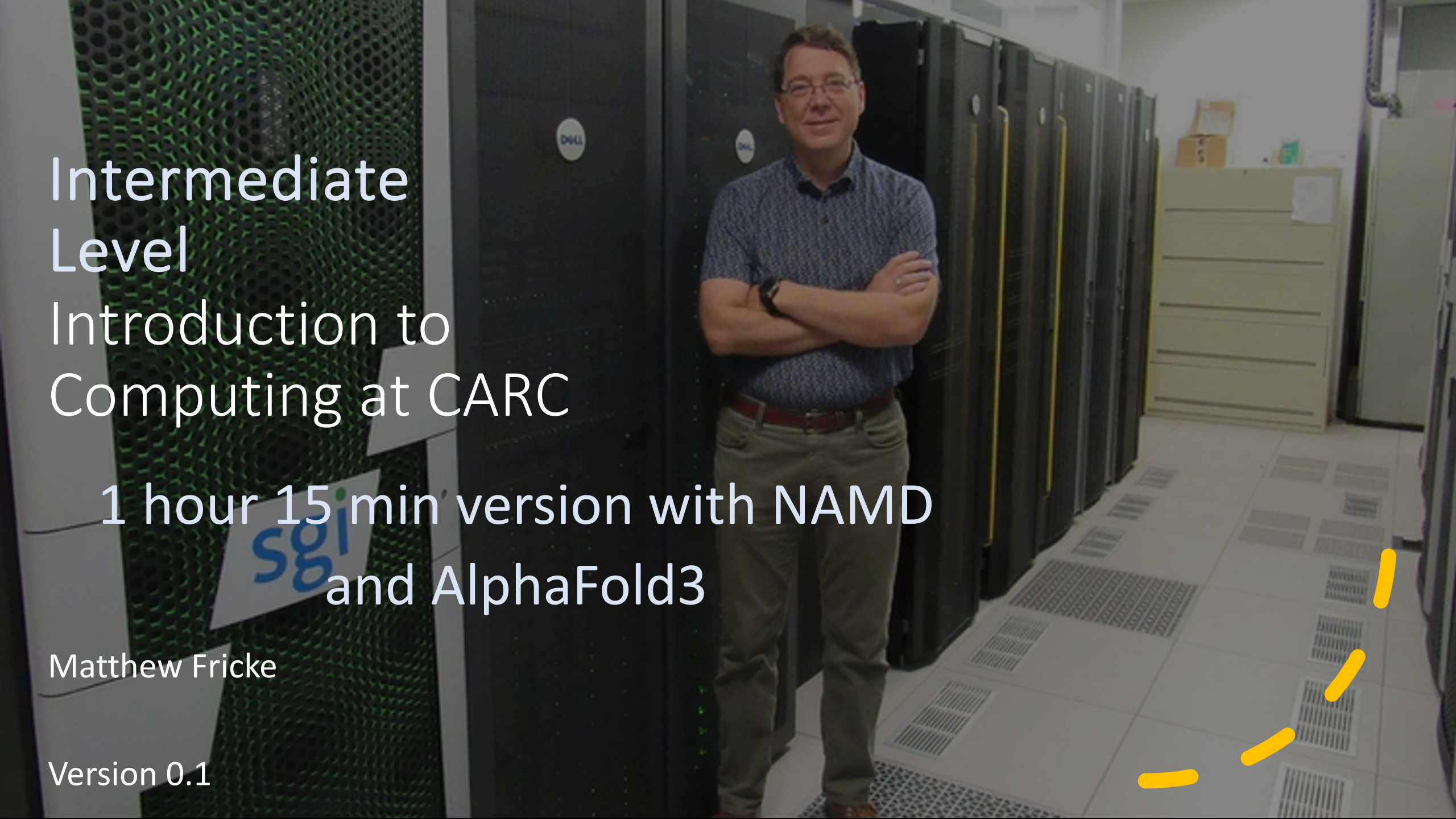


Intermediate Level Introduction to Computing at CARC

1 hour 15 min version with NAMD and AlphaFold3

Matthew Fricke

Version 0.1

Goals

- 1) SLURM scheduler literacy
- 2) Example NAMD
- 3) Example Alphafold3 run
- 4) Visualise output of Alphafold3 in Jupyter

- We wont cover file transfer, storage systems, module system, conda, PBS. (These are all covered in depth in the video tutorials)

Logging in

First login to the Linux **workstation** in front of you. Your CARC username is on the next screen.

If you have logged in before use your existing password

Otherwise, you will go to https://mokey.alliance.unm.edu/auth/forgotpw

This is an “important step” so don’t let me move on until you have logged in

Logging into Hopper


```
ssh vanilla@hopper.alliance.unm.edu
```

Should prompt you for a password…

Don’t let me move on until you are able to login.

Replace vanilla with your name (unless your last name is Ice)

Logging into Hopper

Welcome to Hopper
▶️ By using CARC systems you agree to the CARC Good Neighbor Guide
▶️ Tutorials and the Good Neighbor Guide are at https://goto.unm.edu/carc-start
▶️ Need help? Email: help@carc.unm.edu

Last login: Sun Aug 24 18:04:03 2025 from 50.188.169.22

vanilla@hopper:~


ENJOY
Slurm
SODA
IT'S HIGHLY ADDICTIVE!
VOTED #1 SOFT DRINK OF THE 31ST CENTURY!
YOU WIN!
WHIMMY WHAM WHAM WOZZLE!
SLURMS McKENZIE

```
[vanilla@hopper ~]$ qgrok
partition          nodes  nodes  nodes  total  total  free   total  free   CPUs   RAM/node   time   CPU    GPU    RAM
name        jobs   free   busy   down   nodes  CPUs   CPUs   GPUs   GPUs   /node             limit  limit  limit  limit
----------  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----  ---------  -----  -----  -----  -----
general     4      4      6      0      10     320    156    0      0      32     93G        2d     128    0      372G
debug       1      1      1      0      2      64     63     0      0      32     93G        4h     8      0      25G
totals:     5      5      7      0      12     384    219    0      0
```

```
[vanilla@hopper ~]$ qgrok
```

partition name	jobs	nodes free	nodes busy	nodes down	total nodes	total CPUs	free CPUs	total GPUs	free GPUs	CPUs /node	RAM/node	time limit	CPU limit	GPU limit	RAM limit
general	4	4	6	0	10	320	156	0	0	32	93G	2d	128	0	372G
debug	1	1	1	0	2	64	63	0	0	32	93G	4h	8	0	25G
condo	29	30	25	0	55	1760	1060	37	8	32	94G-1.5T	2d	512	4	1.5T
bugs	0	0	2	0	2	64	0	0	0	32	93G	1w	64	0	
pcnc	0	0	2	0	2	64	0	0	0	32	93G	1w	64	0	
pathogen	0	1	0	0	1	32	32	0	0	32	93G	1w	32	0	
tc	8	1	9	0	10	320	104	0	0	32	93G-1.5T	1w	320	0	
gold	0	2	0	0	2	64	64	0	0	32	93G	1w	64	0	
fishgen	1	0	1	0	1	32	12	0	0	32	377G	1w	32	0	
neuro-hsc	0	14	0	0	14	448	448	0	0	32	93G	1w	448	0	
pna	0	1	0	0	1	32	32	0	0	32	93G	1w	32	0	
geodef	0	4	0	0	4	128	128	0	0	32	504G	1w	128	0	
cup-ecs	0	0	2	0	2	64	0	13	8	32	188G	1w	64	13	
tid	0	0	1	0	1	32	0	2	0	32	188G	1w	32	2	
biocomp	0	1	0	0	1	32	32	1	0	32	188G	1w	32	1	
chakra	0	0	1	0	1	32	0	1	0	32	188G	1w	32	1	
quark	0	3	7	0	10	320	112	20	0	32	377G	1w3d	320	20	
toadpole	0	1	0	0	1	32	32	0	0	32	94G	1w	32	0	
insar	0	2	0	0	2	64	64	0	0	32	504G	1w	64	0	
totals:	34	35	32	0	67	2144	1279	37	8						

```
[vanilla@hopper ~]$ qgrok
partition          nodes   nodes   nodes   total   total   free    total   free    CPUs    RAM/node    time    CPU     GPU     RAM
name        jobs   free    busy    down    nodes   CPUs    CPUs    GPUs    GPUs    /node               limit   limit   limit   limit
----------  -----  -----   -----   -----   -----   -----   -----   -----   -----   -----   ---------   -----   -----   -----   -----
general     4      4       6       0       10      320     156     0       0       32      93G         2d      128     0       372G
debug       1      1       1       0       2       64      63      0       0       32      93G         4h      8       0       25G
condo       29     30      25      0       55      1760    1060    37      8       32      94G-1.5T    2d      512     4       1.5T
bugs        0      0       2       0       2       64      0       0       0       32      93G         1w      64      0
pcnc        0      0       2       0       2       64      0       0       0       32      93G         1w      64      0
pathogen    0      1       0       0       1       32      32      0       0       32      93G         1w      32      0
tc          8      1       9       0       10      320     104     0       0       32      93G-1.5T    1w      320     0
gold        0      2       0       0       2
fishgen     1      0       1       0       1
neuro-hsc   0      14      0       0       1
pna         0      1       0       0       1
geodef      0      4       0       0       4
cup-ecs     0      0       2       0       2
tid         0      0       1       0       1
biocomp     0      1       0       0       1
chakra      0      0       1       0       1
quark       0      3       7       0       1
toadpole    0      1       0       0       1
insar       0      2       0       0       2
totals:     34     35      32      0       6
```

Open partitions for use by everyone with a CARC account.

Purchased by the Office for the Vice President for Research.

```
[vanilla@hopper ~]$ qgrok
```

partition name	jobs	nodes free	nodes busy	nodes down	total nodes	total CPUs	free CPUs	total GPUs	free GPUs	CPUs /node	RAM/node	time limit	CPU limit	GPU limit	RAM limit
general	4	4	6	0	10	320	156	0	0	32	93G	2d	128	0	372G
debug	1	1	1	0	2	64	63	0	0	32	93G	4h	8	0	25G
condo	29	30	25	0	55	1760	1060	37	8	32	94G-1.5T	2d	512	4	1.5T
bugs	0	0	2	0	2	64	0	0	0	32	93G	1w	64	0	
pcnc	0	0	2	0	2	64	0	0	0	32	93G	1w	64	0	
pathogen	0	1	0	0	1	32	32	0	0	32	93G	1w	32	0	
tc	8	1	9	0	10	320	104	0	0	32	93G-1.5T	1w	320	0	
gold	0	2	0	0	2	64	64	0	0	32	93G	1w	64	0	
fishgen	1	0	1	0	1	32	12	0	0	32	377G	1w	32	0	
neuro-hsc	0	14	0	0	14	448	448	0	0	32	93G	1w	448	0	
pna	0	1	0	0	1	32	32	0	0	32	93G	1w	32	0	
geodef	0	4	0	0	4	128	128	0	0	32	504G	1w	128	0	
cup-ecs	0	0	2	0	2	64	0	13	8	32	188G	1w	64	13	
tid	0	0	1	0	1	32	0	2	0	32	188G	1w	32	2	
biocomp	0	1	0	0	1	32	32	1	0	32	188G	1w	32	1	
chakra	0	0	1	0	1	32	0	1	0	32	188G	1w	32	1	
quark	0	3	7	0	10	320	112	20	0	32	377G	1w3d	320	20	
toadpole	0	1	0	0	1	32	32	0	0	32	94G	1w	32	0	
insar	0	2	0	0	2	64	64	0	0	32	504G	1w	64	0	
totals:	34	35	32	0	67	2144	1279	37	8						

```
[vanilla@hopper ~]$ qgrok
partition          nodes   nodes   nodes   total   total   free    total   free    CPUs    RAM/node    time    CPU     GPU     RAM
name        jobs   free    busy    down    nodes   CPUs    CPUs    GPUs    GPUs    /node               limit   limit   limit   limit
----------  -----  -----   -----   -----   -----   -----   -----   -----   -----   -----   ---------   -----   -----   -----   -----
general     4      4       6       0       10      320     156     0       0       32      93G         2d      128     0       372G
debug       1      1       1       0       2       64      63      0       0       32      93G         4h      8       0       25G
condo       29     30      25      0       55      1760    1060    37      8       32      94G-1.5T    2d      512     4       1.5T
bugs        0      0       2       0       2       64      0       0       0       32      93G         1w      64      0
pcnc        0      0       2       0       2       64      0       0       0       32      93G         1w      64      0
pathogen    0      1       0       0       1       32      32      0       0       32      93G         1w      32      0
tc          8      1       9       0       10      320     104     0       0       32      93G-1.5T    1w      320     0
gold        0      2       0       0       2
fishgen     1      0       1       0       1
neuro-hsc   0      14      0       0       1
pna         0      1       0       0       1
geodef      0      4       0       0       4
cup-ecs     0      0       2       0       2
tid         0      0       1       0       1
biocomp     0      1       0       0       1
chakra      0      0       1       0       1
quark       0      3       7       0       1
toadpole    0      1       0       0       1
insar       0      2       0       0       2
totals:     34     35      32      0       6
```

Private partitions

- Reserved for use by the purchaser.
- Request access by emailing support@carc.unm.edu and CC the partition owner.

```
[vanilla@hopper ~]$ qgrok
partition          nodes   nodes   nodes   total   total   free    total   free    CPUs    RAM/node    time    CPU     GPU     RAM
name        jobs   free    busy    down    nodes   CPUs    CPUs    GPUs    GPUs    /node               limit   limit   limit   limit
----------  -----  -----   -----   -----   -----   -----   -----   -----   -----   -----   ---------   -----   -----   -----   -----
general     4      4       6       0       10      320     156     0       0       32      93G         2d      128     0       372G
debug       1      1       1       0       2       64      63      0       0       32      93G         4h      8       0       25G
condo       29     30      25      0       55      1760    1060    37      8       32      94G-1.5T    2d      512     4       1.5T
bugs        0      0       2       0       2       64      0       0       0       32      93G         1w      64      0
pcnc        0      0       2       0       2       64      0       0       0       32      93G         1w      64      0
pathogen    0      1       0       0       1       32      32      0       0       32      93G         1w      32      0
tc          8      1       9       0       10      320     104     0       0       32      93G-1.5T    1w      320     0
gold        0      2       0       0       2
fishgen     1      0       1       0       1
neuro-hsc   0      14      0       0       14
pna         0      1       0       0       1
geodef      0      4       0       0       4
cup-ecs     0      0       2       0       2
tid         0      0       1       0       1
biocomp     0      1       0       0       1
chakra      0      0       1       0       1
quark       0      3       7       0       10
toadpole    0      1       0       0       1
insar       0      2       0       0       2
totals:     34     35      32      0       67
```

Condo “scavenger” partition

- Allows you to use compute nodes purchased by another group that are currently idle.
- May be interrupted at any time if the owners start to use it.

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      1    mix hopper011
debug        up    4:00:00      1   idle hopper012
```

sinfo reports information about partitions

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      1    mix hopper011
debug        up    4:00:00      1   idle hopper012
```

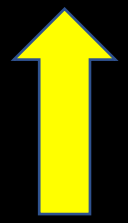
The debug queues are intended for testing your programs.

And for interactive jobs.

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      1    mix hopper011
debug        up    4:00:00      1   idle hopper012
```

Name

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      1    mix hopper011
debug        up    4:00:00      1   idle hopper012
```



You can run a “job” for up to 4 hrs.

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      1    mix hopper011
debug        up    4:00:00      1   idle hopper012
```

There are two nodes in this partition.

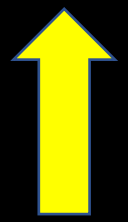
```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      1    mix hopper011
debug        up    4:00:00      1   idle hopper012
```

The state of the nodes in the partition

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      1    mix hopper011
debug        up    4:00:00      1   idle hopper012
```

The name of the nodes in the partition

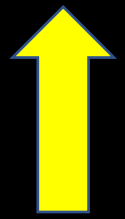
```
[vanilla@hopper ~]$ sinfo --partition general
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
general*     up 2-00:00:00      5  alloc hopper[001-009]
general*     up 2-00:00:00      4   idle hopper010
```



Hopper001 through 009 are running jobs.

Hopper010 is waiting to be used.

```
[vanilla@hopper ~]$ hostname
hopper
[vanilla@hopper ~]$
```



Running on the Head Node.

The head node’s name is “hopper”.

```
[vanilla@hopper ~]$ hostname
hopper
[vanilla@hopper ~]$ man hostname
```

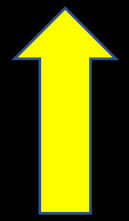
```
[vanilla@hopper ~]$ hostname
hopper
[vanilla@hopper ~]$ man hostname
(‘q’ to quit)

[vanilla@hopper ~]$ man man
(‘q’ to quit)
```

[vanilla@hopper ~]$ man sinfo

```
sinfo(1)                                                    Slurm
Commands                                                    sinfo(1)

NAME
       sinfo - View information about Slurm nodes and partitions.

SYNOPSIS
       sinfo [OPTIONS...]

DESCRIPTION
       sinfo is used to view partition and node information for a system running
Slurm

OPTIONS
       -a, --all
              Display  information  about  all partitions. This causes information
to be displayed about partitions that are configured as hidden and partitions
that are unavailable to the user's group.
```

```
[vanilla@hopper ~]$ sinfo --all
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
general*     up 2-00:00:00      9  alloc hopper[001-009]
general*     up 2-00:00:00      1   idle hopper010
debug        up    4:00:00      2   idle hopper[011-012]
condo        up 2-00:00:00      1  down* hopper045
condo        up 2-00:00:00      3    mix hopper[018-020]
condo        up 2-00:00:00     16  alloc hopper[013-015,028-036,049-052]
condo        up 2-00:00:00     18   idle hopper[016-017,021-027,037-044,053]
bugs         up 7-00:00:00      2  alloc hopper[013-014]
pcnc         up 7-00:00:00      1  alloc hopper015
pcnc         up 7-00:00:00      1   idle hopper016
pathogen     up 7-00:00:00      1   idle hopper017
tc           up 7-00:00:00      3    mix hopper[018-020]
tc           up 7-00:00:00      2  alloc hopper[029-030]
tc           up 7-00:00:00      5   idle hopper[021-025]
gold         up 7-00:00:00      2   idle hopper[026-027]
fishgen      up 7-00:00:00      1  alloc hopper028
neuro-hsc    up 7-00:00:00      6  alloc hopper[031-036]
neuro-hsc    up 7-00:00:00      8   idle hopper[037-044]
cup-ecs      up 7-00:00:00      2  alloc hopper[049-050]
tid          up 7-00:00:00      1  alloc hopper051
biocomp      up 7-00:00:00      1  alloc hopper052
chakra       up 7-00:00:00      1   idle hopper053
pna          up 7-00:00:00      1  down* hopper045
```

```
[vanilla@hopper ~]$ srun --partition debug hostname
```



Tell slurm to run a program on a compute node…

```
[vanilla@hopper ~]$ srun --partition debug hostname
```

Run the program on a compute node in the debug partition.

```
[vanilla@hopper ~]$ srun --partition debug hostname
```

The program to run.

```
[vanilla@hopper ~]$ srun --partition debug hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
You have not been allocated GPUs. To request GPUs,
use the -G option in your submission script.
hopper011
```

```
[vanilla@hopper ~]$ squeue
```

```
[vanilla@hopper ~]$ squeue
 JOBID PARTITION     NAME     USER ST       TIM
 4314    general     PRE erowland PD       0:00
 4315    general     PRE erowland PD       0:00
 4317    general     PRE erowland PD       0:00
 4318    general     PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
                                                     2 (QOSMaxCpuPerUserLimit)
```

PD means programs that are waiting their turn.

Shows you what the slurm scheduler is doing right now.

Here we can see that user 'erowland' has a lot of programs waiting to run.

```
[vanilla@hopper ~]$ squeue
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             4314   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
             4315   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
             4317   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
             4318   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
                                                                 2 (QOSMaxCpuPerUserLimit)
             4337   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
```

The reason these jobs are not running is that ‘erowland’ is already using the maximum number of CPUs they are allowed.

```
[vanilla@hopper ~]$ squeue -t R --all
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              4405     condo    2ndMA  mfricke  R 1-07:48:30      6 hopper[031-036]
              5208     condo       NN      kgu  R    5:48:49      1 hopper015
              5210     condo       NN      kgu  R    6:30:13      1 hopper014
              5209     condo       NN      kgu  R    6:31:13      1 hopper013
              5206     condo       NN      kgu  R    6:32:13      1 hopper051
              5207     condo       NN      kgu  R    6:32:13      1 hopper052
              5205     condo       NN      kgu  R    6:32:43      1 hopper028
              4595   cup-ecs golConfi aalasand  R 2-06:51:59      1 hopper050
              4594   cup-ecs golConfi aalasand  R 2-06:52:03      1 hopper049
              5120   general jupyterh   jacobm  R   11:45:47      1 hopper007
              4313   general      PRE erowland  R    1:17:29      2 hopper[003-004]
              5111   general    1stMA  mfricke  R   11:15:28      2 hopper[005-006]
              5025   general      c2n    jxzuo  R       1:50      1 hopper001
              5024   general      c2n    jxzuo  R      31:28      1 hopper002
              5203   general       NN      kgu  R    6:37:50      1 hopper009
              5201   general       NN      kgu  R    6:38:14      1 hopper008
              4390        tc UCsTpCyd lepluart  R 2-15:18:18      3 hopper[018-020]
              5198        tc       NN      kgu  R    6:40:19      1 hopper030
              5196        tc       NN      kgu  R    6:40:31      1 hopper029
```

```
[vanilla@hopper ~]$ srun --partition debug --ntasks 2 hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
You have not been allocated GPUs. To request GPUs, use the -G
option in your submission script.
hopper011
hopper011
```

```
[vanilla@hopper ~]$ srun --partition debug --ntasks 2 hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
You have not been allocated GPUs. To request GPUs, use the -G
option in your submission script.
hopper011
hopper011
```

You ran two copies of your program.

ntasks is the number of copies to run.

```
[vanilla@hopper ~]$ srun --partition debug --ntasks 8 hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
hopper011
hopper011
hopper011
You have not been allocated GPUs. To request GPUs, use the -G
option in your submission script.
hopper011
hopper011
hopper011
hopper011
hopper011
hopper011
```

You ran eight copies of your program.

ntasks is the number of copies to run.

```
[vanilla@hopper ~]$ srun --partition debug --ntasks 8 hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
hopper011
hopper011
hopper011
You have not been allocated GPUs. To request GPUs, use the -G
option in your submission script.
hopper011
hopper011
hopper011
hopper011
hopper011
```

By default, each task (copy of your program) is allowed to use one CPU.

Many programs are able to use more than one CPU at a time.

```
[vanilla@hopper ~]$ srun --partition debug --ntasks 2 --cpus-per-task 2 hostname
srun: Account not specified in script or ~/.default_slurm_account, using latest project
You have not been allocated GPUs. To request GPUs, use the -G option in your submission
script.
hopper011
hopper011
```

Here we are telling SLURM to run 2 copies of our program and let each copy of our program use 2 CPUs.

```
[vanilla@hopper ~]$ srun --partition debug --nodes 2 --ntasks-per-node 4 hostname
srun: Account not specified in script or ~/.default_slurm_account, using
latest project
hopper012
You have not been allocated GPUs. To request GPUs, use the -G option in
your submission script.
hopper012
hopper011
hopper011
hopper012
hopper012
hopper011
hopper011
```

Here we are telling SLURM to run 4 copies of our program on 2 different compute nodes.

This is useful when our programs need a bigger share of the compute node.

```
[vanilla@hopper ~]$ srun --partition debug --nodes 2
--ntasks-per-node 2 --cpus-per-task 2 hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
hopper011
You have not been allocated GPUs. To request GPUs, use
the -G option in your submission script.
hopper011
hopper012
hopper012
```

And we can combine all three.

```
[vanilla@hopper ~]$ srun --partition debug --mem 4G
--nodes 2 --ntasks-per-node 2 --cpus-per-task 2
hostname
srun: Account not specified in script or
~/.default_slurm
hopper012
hopper012
You have not be                                    se
the -G option i
hopper011
Hopper011
```

And we can specify how much memory we want.

--mem 4G means give me 4 gigabytes of memory per node.

```
[vanilla@hopper ~]$ srun --partition debug --mem 4G
--nodes 2 --ntasks-per-node 2 --cpus-per-task 2
hostname
srun: Account not specified in script or
~/.default_slurm
hopper012
hopper012
You have not be                                    se
the -G option i
hopper011
Hopper011
```

Why does all this matter?

The purpose of SLURM is to provide you the hardware your programs need.

So you have to understand what those requirements are really well.

```
[vanilla@hopper ~]$ srun --partition debug --mem 4G
--nodes 2 --ntasks-per-node 2 --cpus-per-task 2 hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
hopper012
hopper012
You have not be                                    se
the -G option i
hopper011
Hopper011
```

1) **Can my program use multiple CPUs?**
2) **How much memory does my program need?**
3) **Can my program use multiple compute nodes (MPI*, GNU Parallel*)?**
4) **Can my program use GPUs?**

Interactive vs Batch Mode

Interactive Mode

- Everything so far has been interactive. You request hardware, run your program, and get the output on your screen right away.

Batch Mode

- Most programs at an HPC center are run in “batch” mode.
- Batch mode means we write a shell script that the SLURM scheduler runs for us. The script requests hardware just like we did with salloc and then runs the commands in the script.
- Whatever would have been written to the screen is saved to a file instead.

```
[vanilla@hopper ~]$ git clone https://lobogit.unm.edu/CARC/workshops.git
Cloning into 'workshops'...
remote: Enumerating objects: 132, done.
remote: Counting objects: 100% (75/75), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 132 (delta 33), reused 74 (delta 32), pack-reused 57
Receiving objects: 100% (132/132), 57.58 KiB | 3.60 MiB/s, done.
Resolving deltas: 100% (51/51), done.
```

Rather than make you write shell scripts lets just download some we wrote for this workshop…

```
vanilla@hopper:~ $ tree workshops
workshops
├── alphafold3
│   ├── data
│   │   ├── sample_output
│   │   │   ├── 2pv7.cif
│   │   │   └── short.cif
│   │   ├── short.json
│   │   └── tyrA_flu.json
│   ├── notebooks
│   │   └── molecule_vis.ipynb
│   ├── slurm
│   │   ├── short_gpu_l40s.slurm
│   │   ├── test_gpu_a100.slurm
│   │   ├── test_gpu_h100.slurm
│   │   ├── test_gpu_l40s.slurm
│   │   └── tyra_gpu_h100.slurm
│   ├── test_a100.out
│   ├── test_l40s.out
│   └── tyrA_h100.out
│
…
├── namd
│   ├── namd3_multinode.slurm
│   ├── namd_multinode.slurm
│   ├── namd.slurm
│   ├── namd_xena.slurm
│   ├── name_multinode.slurm
│   ├── par_all27_prot_lipid.inp
│   ├── ubq_ws_eq.conf
│   ├── ubq_ws.pdb
│   └── ubq_ws.psf
```

Run tree to see how the workshops directories are organized…

```
[vanilla@hopper intro_workshop]$ pwd
/users/vanilla/workshops/intro_workshop
[vanilla@hopper intro_workshop]$ cat slurm/workshop_example1.sh
#!/bin/bash
#SBATCH --partition debug
#SBATCH --ntasks 4
#SBATCH --time 00:05:00
#SBATCH --job-name ws_example
#SBATCH --mail-user your_username@unm.edu
#SBATCH --mail-type ALL

srun hostname
```

Let's take a look at the workshop_example.sh script in the slurm directory…

```
[vanilla@hopper intro_workshop]$ sbatch slurm/workshop_example1.sh
sbatch: Account not specified in script or ~/.default_slurm_account,
using latest project
Submitted batch job 5252
[vanilla@hopper intro_workshop]$
```

We submit our slurm shell script with the sbatch command.

```
[vanilla@hopper intro_workshop]$ sbatch slurm/workshop_example.sh
sbatch: Account not specified in script or ~/.default_slurm_account,
using latest project
Submitted batch job 5252
[vanilla@hopper intro_workshop]$
```

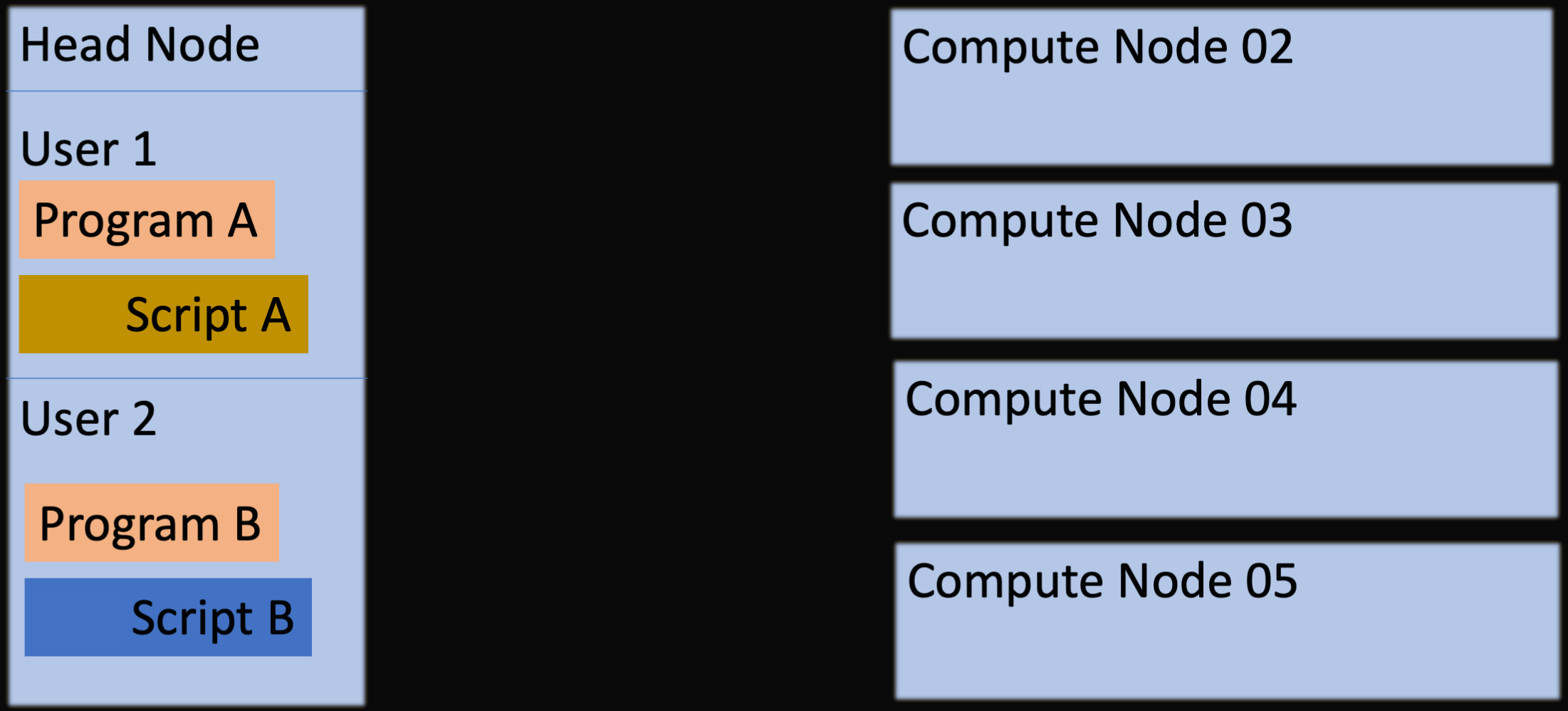
We submit our slurm shell script with the sbatch command.

Notice that the only output we get is a job id.

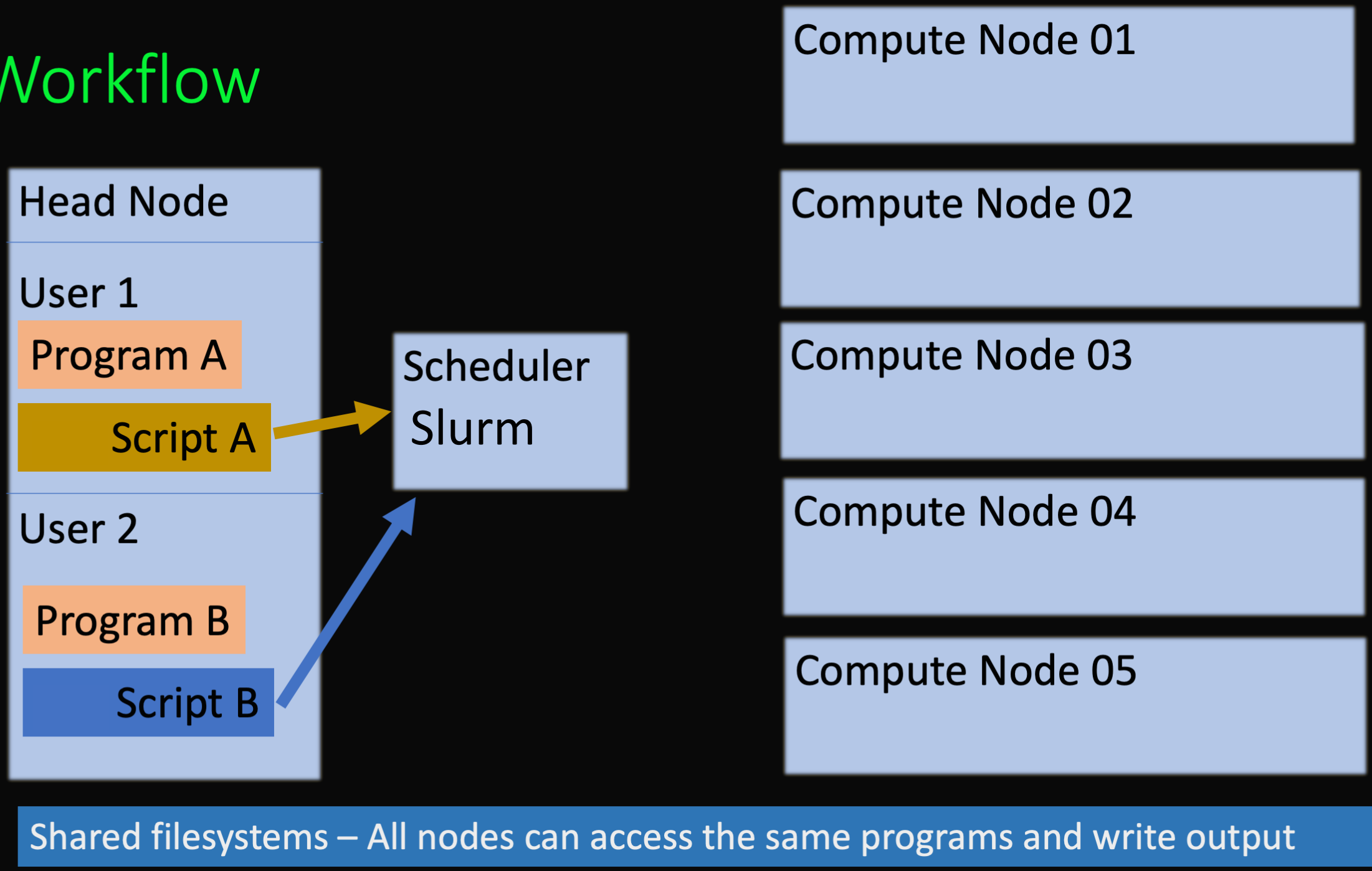
This indicates that the script was successfully sent to the scheduler.

The commands in the script will run as soon as the hardware requested is available.

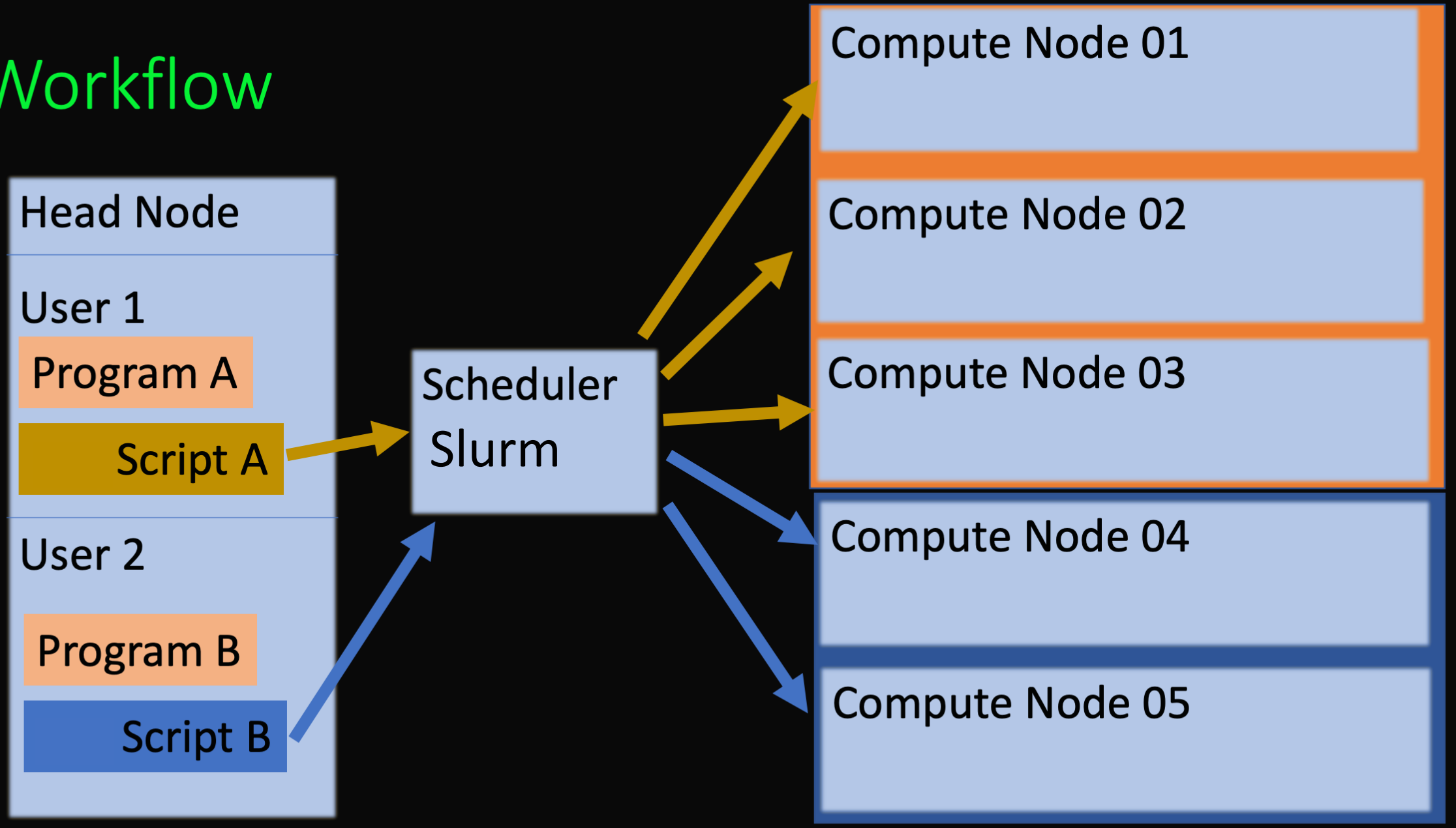
Workflow


Head Node
User 1
Program A
Script A
User 2
Program B
Script B
Compute Node 01
Compute Node 02
Compute Node 03
Compute Node 04
Compute Node 05
Shared filesystems – All nodes can access the same programs and write output

Workflow

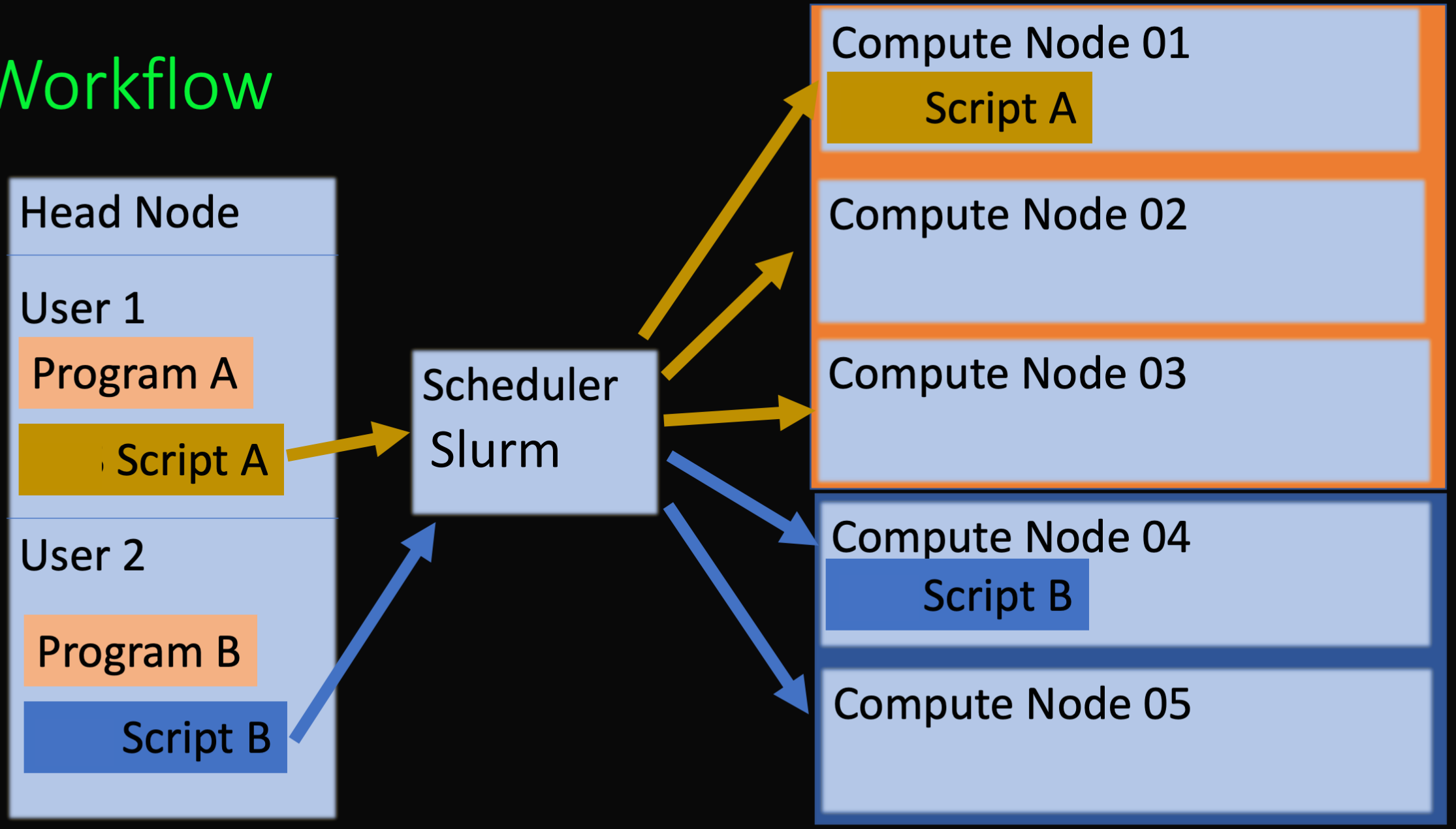

Head Node
User 1
Program A
Script A
User 2
Program B
Script B
Scheduler
Slurm
Compute Node 01
Compute Node 02
Compute Node 03
Compute Node 04
Compute Node 05
Shared filesystems – All nodes can access the same programs and write output

Workflow



Shared filesystems – All nodes can access the same programs and write output

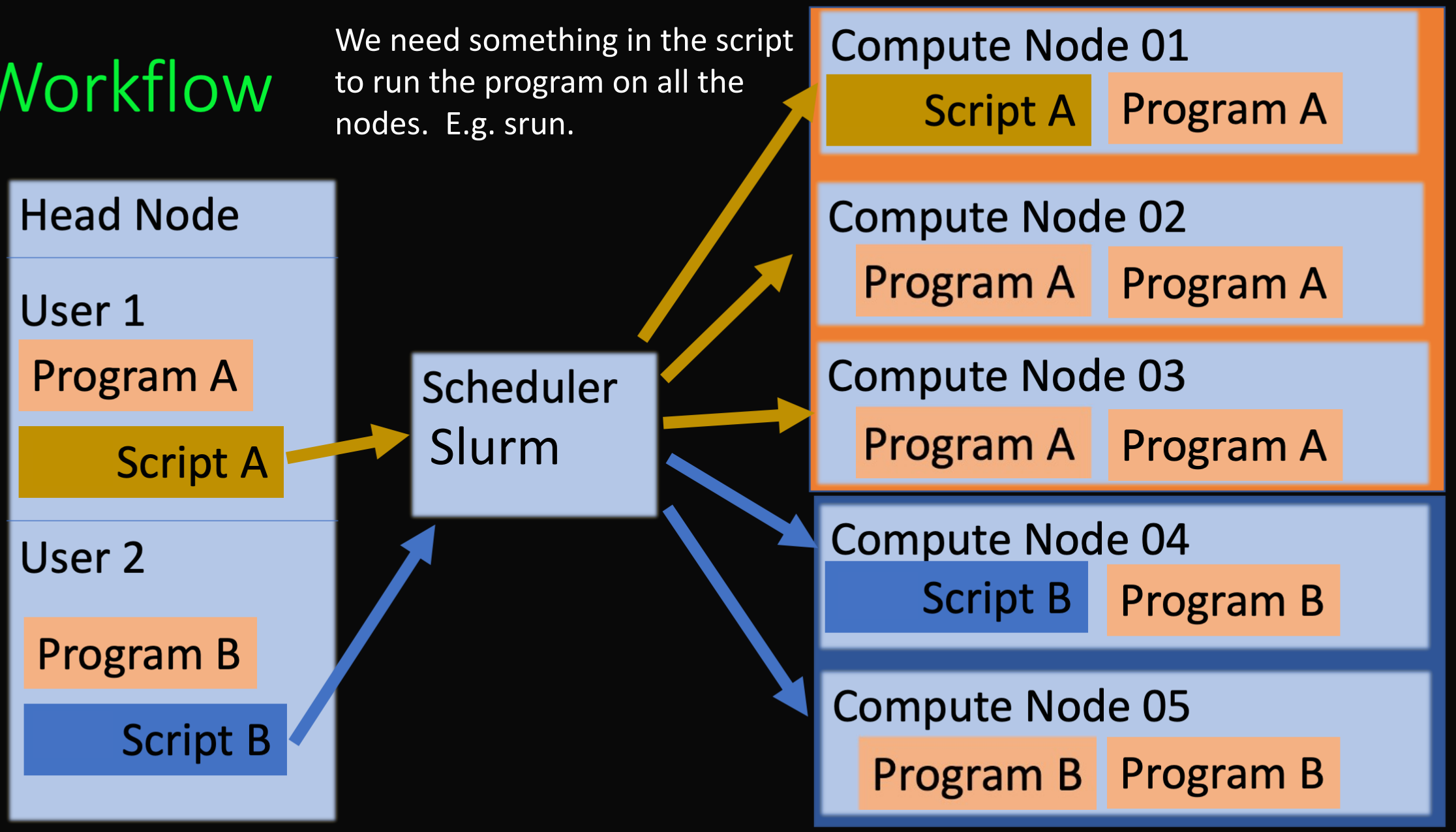
Workflow



Shared filesystems – All nodes can access the same programs and write output

Workflow

We need something in the script to run the program on all the nodes. E.g. srun.



Shared filesystems – All nodes can access the same programs and write output

```
[vanilla@hopper intro_workshop]$ ls
code  data  pbs  slurm  slurm-5252.out
```

The **hostname** command is very fast so everyone's job should finish in a few seconds.

When it is finished you will have a new file named slurm-{your job id}.out.

```
[vanilla@hopper intro_workshop]$ ls
code  data  pbs  slurm  slurm-5252.out
```

When it is finished you will have a new file named slurm-{your job id}.out.

```
[vanilla@hopper intro_workshop]$ cat slurm-5252.out
hopper011
```

```
[vanilla@hopper intro_workshop]$ ls
code  data  pbs  slurm  slurm-5252.out
```

When it is finished you will have a new file named slurm-{your job id}.out.

```
[vanilla@hopper intro_workshop]$ cat slurm-5252.out
hopper011
```

Why did it only run the program once instead of 4 times?

```
[vanilla@hopper intro_workshop]$ sbatch slurm/workshop_example1.sh
sbatch: Account not specified in script or ~/.default_slurm_account,
using latest project
Submitted batch job 5252
[vanilla@hopper intro_workshop]$
```

Take a look at the output file.

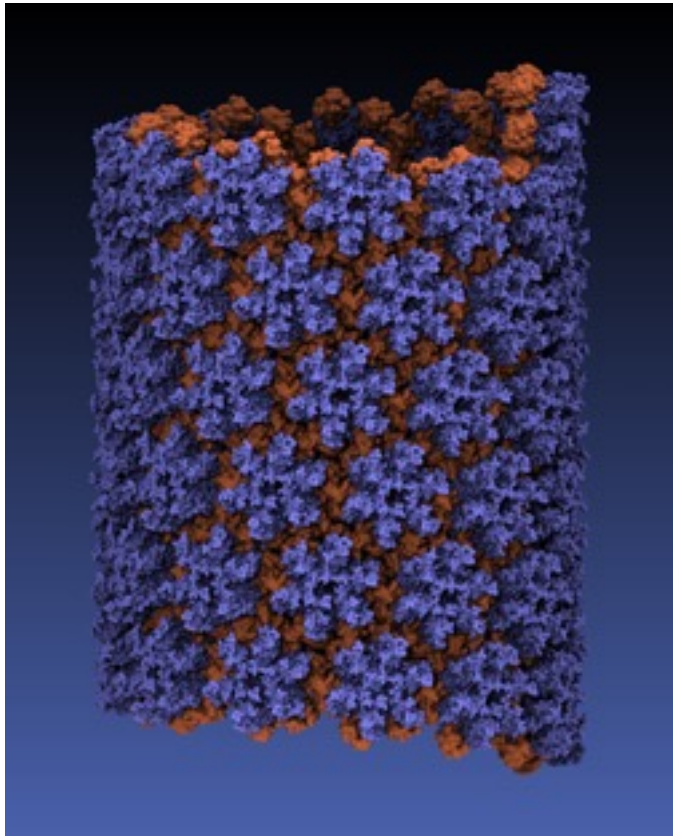
```
[vanilla@hopper intro_workshop]$ sbatch slurm/workshop_example1.sh
sbatch: Account not specified in script or ~/.default_slurm_account,
using latest project
Submitted batch job 5252
[vanilla@hopper intro_workshop]$
```

Take a look at the output file.

```
[vanilla@hopper intro_workshop]$ ls
code  conda_envs  data  pbs  slurm  slurm-5252.out
mfricke@hopper:~/workshops/intro_workshop [master ?]$ cat slurm-4038014.out
To request GPUs, add --gpus-per-node X or --gpus X, where X is the desired number
of GPUs.
Job 5252 running on hopper011
hopper011
hopper011
hopper011
hopper011
```

NAMD
Scalable Molecular Dynamics

NAMD and VMD are part of the team winning the **2020 ACM Gordon Bell Special Prize** for high performance computing-based **COVID-19 research**, for the paper **AI-Driven Multiscale Simulations Illuminate Mechanisms of SARS-CoV-2 Spike Dynamics**, presented at Supercomputing 2020, Nov 18, 2020.



```
vanilla@hopper:~ $ cd workshops/namd
vanilla@hopper:~/workshops/namd $ ls
namd.slurm  ubq_ws_eq.conf  ubq_ws.pdb  ubq_ws.psf par_all27_prot_lipid.inp
```

```
namd.slurm      -    SLURM Batch Script

ubq_ws_eq.conf  -    NAMD Configuration
                     Simulation info in TCL

ubq_ws.pdb      -    Protein Databank File
                     Metadata and atom locations

ubq_ws.psf      -    Protein Structure File
                     Molecular force field Info

par_all27_prot_lipid.inp - CHARMM Parameter File
                           Protein/Lipid force fields
```

```
vanilla@hopper:~/workshop/namd $ cat namd.slurm
#!/bin/bash
#SBATCH --partition debug
#SBATCH --job-name namd-test
#SBATCH --time 00:05:00
#SBATCH --nodes 1
#SBATCH --ntasks-per-node 4
#SBATCH --mail-user your@email.address
#SBATCH --mail-type ALL

module load namd/2.14/verbs

# NAMD Binaries are not SLURM aware - so we use charmrun directly
charmrun $(which namd2) ++auto-provision ubq_ws_eq.conf
```

SLURM Batch Script

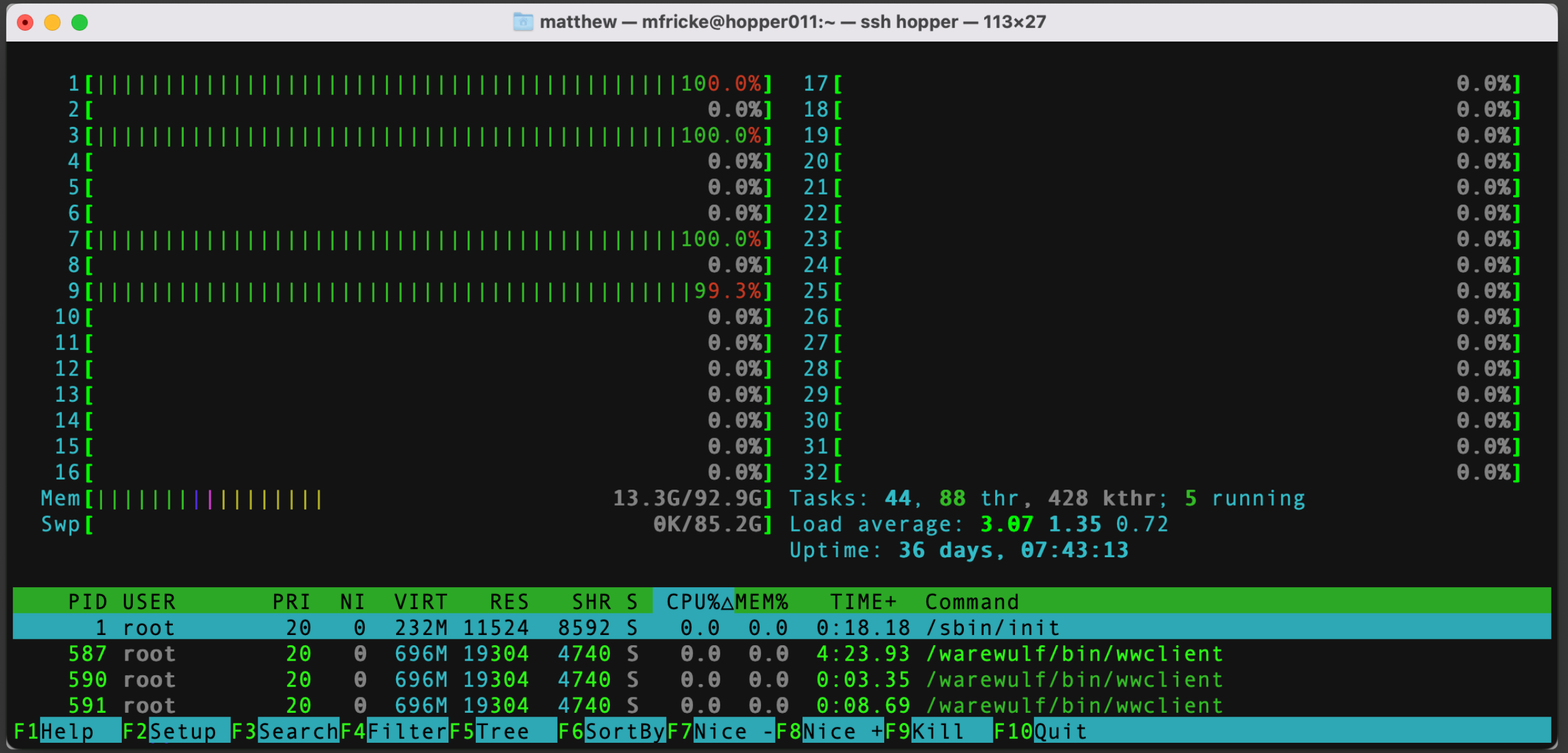
```
vanilla@hopper:~/workshops/namd $ sbatch namd.slurm
sbatch: Account not specified in script or ~/.default_slurm_account, using
latest project
Submitted batch job 1814848
```

```
vanilla@hopper:~/workshops/namd $ squeue --me
```

```
vanilla@hopper:~/workshops/namd $ squeue --me
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1814848     debug namd-tes  vanilla  R       0:04      1 hopper011

vanilla@hopper:~/workshops/namd $ ssh hopper011
Last login: Fri Feb 17 10:04:34 2023 from 172.17.0.2
vanilla@hopper011:~ $ htop
```

Check the status of your job…

Note the hostname of the compute node you were allocated. In the example it is hopper011: SSH to that hostname and run the htop command.



```
vanilla@hopper:~/workshops/namd $ ls
namd.slurm                  ubq_ws_eq.dcd           ubq_ws_eq.vel.BAK
par_all27_prot_lipid.inp    ubq_ws_eq.dcd.BAK       ubq_ws_eq.xsc
slurm-1814849.out           ubq_ws_eq.restart.coor  ubq_ws_eq.xsc.BAK
ubq_ws_eq.conf              ubq_ws_eq.restart.vel   ubq_ws.pdb
ubq_ws_eq.coor              ubq_ws_eq.restart.xsc   ubq_ws.psf
ubq_ws_eq.coor.BAK          ubq_ws_eq.vel
vanilla@hopper:~/workshops/namd $
```

Check your output after the job completes…

ubq_ws_eq.restart.vel – atomic velocities

ubq_ws_eq.restart.coor - atomic coordinates

.restart. - checkpoint files

```
vanilla@hopper:~/workshops/namd $ ssh xena
Last login: Fri Feb 17 10:08:12 2023 from hopper.alliance.unm.edu
<snip>
```

Xena is our GPU and large memory cluster.
NAMD can use GPU acceleration.

```
vanilla@xena:~/workshops/namd $ module spider namd
----------------------------------------------------------------------------
  namd/2.14:
----------------------------------------------------------------------------
     Versions:
        namd/2.14/cuda-multicore
        namd/2.14/cuda-verbs
```

Find the NAMD module names on Xena with the “spider” command

```
vanilla@xena:~ $ qgrok
queues      free   busy   offline  jobs   nodes  CPUs   GPUs   CPUs/node  GPUs/node  Memory/node  time_limit   CPU
_limit
-----       -----  -----  -------  -----  -----  -----  -----  ---------  ---------  -----------  ----------   ---
--
singleGPU   0      23     0        13     23     368    23     16         1          61G          2-00:00:00   192
dualGPU     1      2      1        1      4      64     8      16         2          61G          2-00:00:00   192
bigmem-1TB  1      1      0        1      2      128    0      64         0          1006G        2-00:00:00   192
bigmem-3TB  2      0      0        0      2      128    0      64         0          3.0T         2-00:00:00   192
debug       2      0      0        0      2      32     2      16         1          61G          4:00:00      8
systems     1      0      0        0      1      16     1      16         1          61G          2-00:00:00   192
totals:     6      26     1        15     33     720    33
```

Find the available partitions with “qgrok”

```
vanilla@xena:~/~/workshops/namd$ cat namd_xena.slurm
#!/bin/bash
#SBATCH --partition debug
#SBATCH --gpus 1 # Request 1 GPU
#SBATCH --job-name namd-test
#SBATCH --time 00:05:00
#SBATCH --nodes 1
#SBATCH --ntasks-per-node 4
#SBATCH --mail-user your@email.address
#SBATCH --mail-type ALL

module load namd/2.14/cuda-verbs

# NAMD Binaries are not SLURM aware - so we use charmrun directly
charmrun $(which namd2) ++auto-provision ubq_ws_eq.conf
```

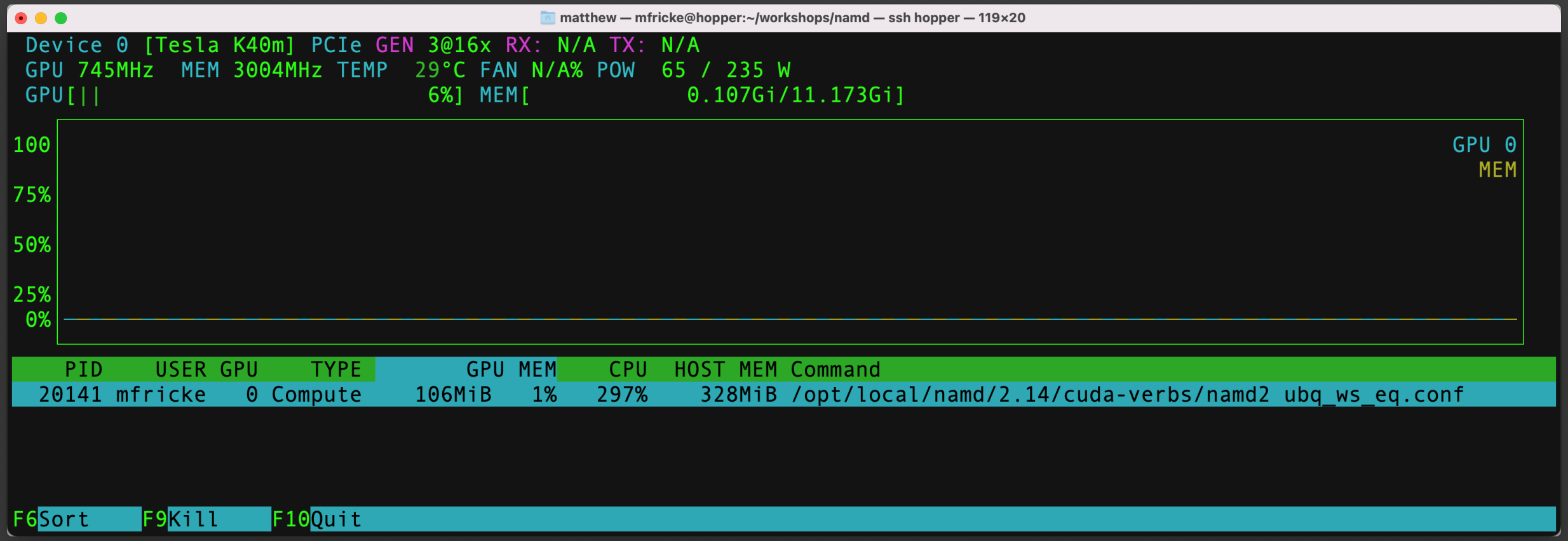
Now we can write a new SLURM script for Xena and just modify the module name and request a GPU

```
vanilla@xena:~/workshops/namd $ sbatch namd_xena.slurm
sbatch: Using account 2016199 from ~/.default_slurm_account
Submitted batch job 333532
vanilla@xena:~/workshops/namd [master ?]$ squeue --me
JOBID PARTITION NAME      USER     ST       TIME  NODES NODELIST(REASON)
333532 debug namd-tes vanilla  R       0:04      1 xena-test

vanilla@xena:~/workshops/namd [master ?]$ ssh xena-test
Last login: Fri Feb 17 10:19:37 2023 from xena
vanilla@xena-test:~ $ module load nvtop
vanilla@xena-test:~ $ nvtop
```

Check the status of your job…

Note the hostname of the compute node you were allocated. In the example it is xena-test:

SSH to that hostname and run the nvtop command.



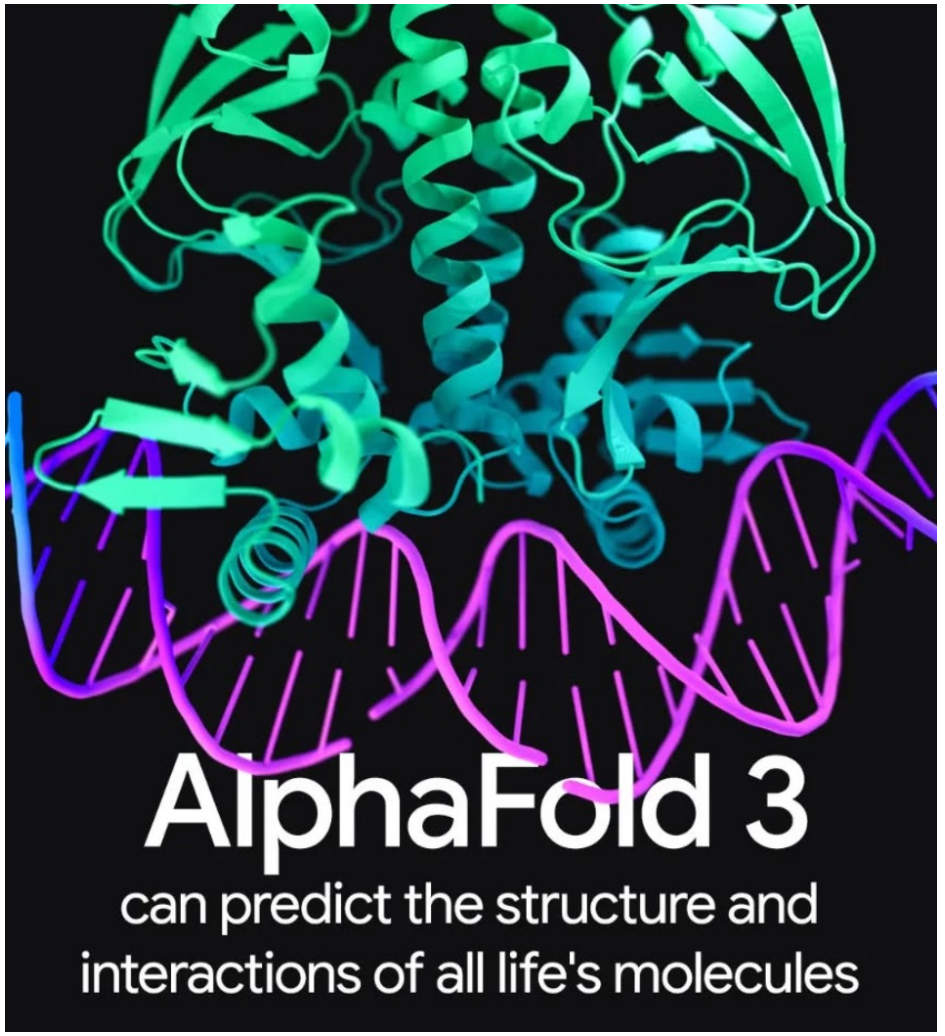
matthew — mfricke@hopper:~/workshops/namd — ssh hopper — 119×20

```
Device 0 [Tesla K40m] PCIe GEN 3@16x RX: N/A TX: N/A
GPU 745MHz  MEM 3004MHz TEMP  29°C FAN N/A% POW  65 / 235 W
GPU[||                          6%] MEM[            0.107Gi/11.173Gi]

100                                                              GPU 0
                                                                  MEM
75%
50%
25%
 0%

    PID    USER GPU     TYPE         GPU MEM     CPU  HOST MEM Command
  20141 mfricke   0 Compute     106MiB   1%    297%     328MiB /opt/local/namd/2.14/cuda-verbs/namd2 ubq_ws_eq.conf

F6Sort     F9Kill     F10Quit
```

Google DeepMind

AlphaFold is an artificial intelligence system developed by DeepMind that predicts three-dimensional protein structures from amino acid sequences with near-experimental accuracy.

AlphaFold 3 improves on AlphaFold 2 by extending predictions beyond proteins to include nucleic acids, ligands, and other biomolecular interactions, enabling accurate modelling of whole complexes rather than just isolated protein structures.

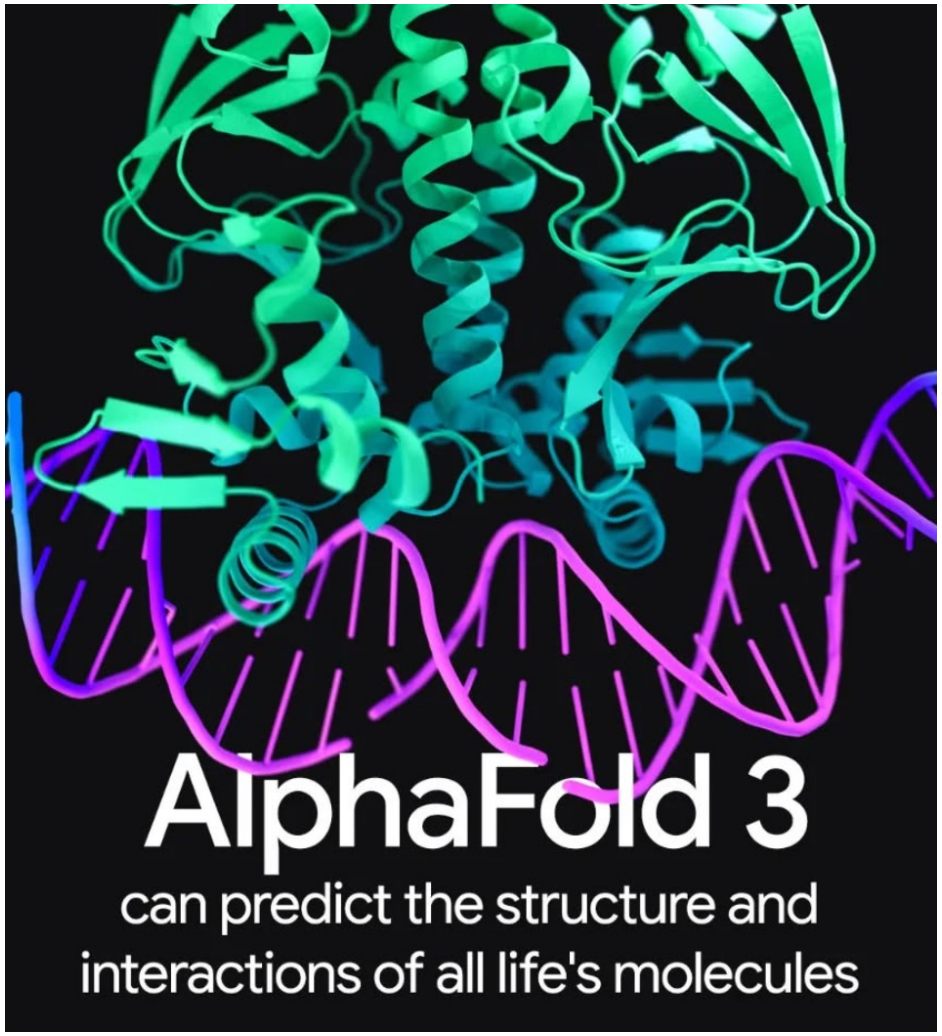


Google DeepMind

AlphaFold is an artificial intelligence system developed by DeepMind that predicts three-dimensional protein structures from amino acid sequences with near-experimental accuracy.

AlphaFold 3 improves on AlphaFold 2 by extending predictions beyond proteins to include nucleic acids, ligands, and other biomolecular interactions, enabling accurate modelling of whole complexes rather than just isolated protein structures.

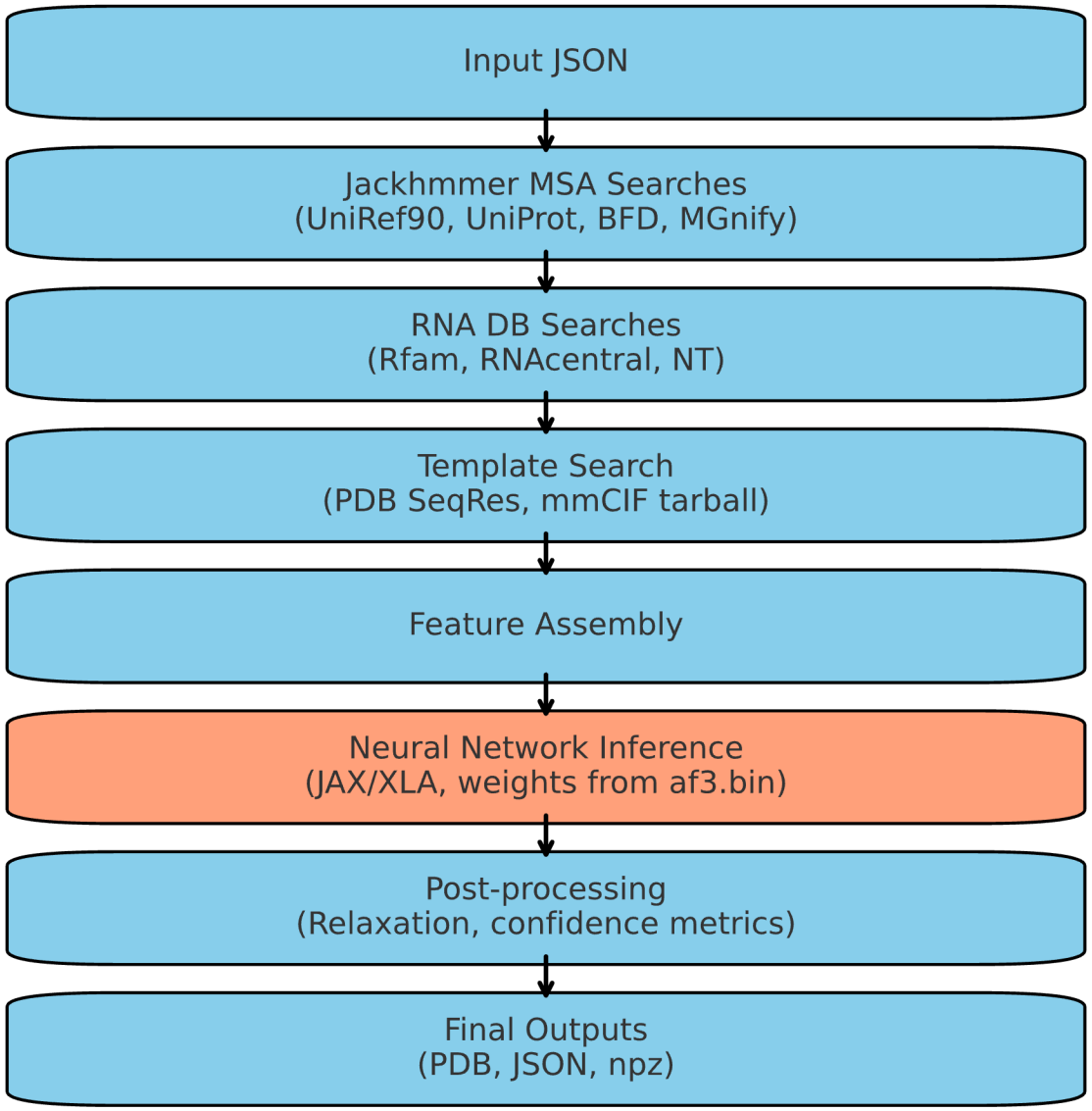
Access to DeepMind's AlphaFold 3 inference model is legally restricted to non-commercial use and may not be shared with anyone lacking explicit ColdFront approval, with all outputs watermarked and traceable to UNM under a legal agreement managed by the UNM Chief Information Officer with Google DeepMind.



```
vanilla@xena:~ $ ssh easley
```

We will use the Easley cluster to run Alphafold3 since it has the modern GPUs it needs.

Alphafold3 Pipeline



- **MSA search (jackhmmer stage):** dominates wall time on CPU (30–90 min for medium proteins, longer for huge DBs).

- **Template search:** 5–15 min once .tar is present.
- **Inference:** seconds to minutes on GPU; hours on CPU.

- **Relaxation:** optional, adds ~10–30 min.

This directory contains the large sequence and structure databases required by AlphaFold3, including clustered metagenomic (MGnify, BFD), protein (UniProt, UniRef90, PDB seqres, PDB mmCIF), and RNA (nt, Rfam, RNAcentral) reference datasets used for multiple sequence alignment, template search, and structure prediction.

```
tree /carc/scratch/shared/alphafold/alphafold3/db
/carc/scratch/shared/alphafold/alphafold3/db
├── bfd-first_non_consensus_sequences.fasta
├── mgy_clusters_2022_05.fa
├── nt_rna_2023_02_23_clust_seq_id_90_cov_80_rep_seq.fasta
├── pdb_2022_09_28_mmcif_files.tar
├── pdb_seqres_2022_09_28.fasta
├── rfam_14_9_clust_seq_id_90_cov_80_rep_seq.fasta
├── rnacentral_active_seq_id_90_cov_80_linclust.fasta
├── uniprot_all_2021_04.fa
└── uniref90_2022_05.fa
```

af3.bin file is essentially the **“secret sauce”**: it contains the trained neural network weights from DeepMind, which encode all of the learning from vast protein structure and sequence data, and without which AlphaFold 3 cannot perform structure prediction.

```
tree /carc/scratch/shared/alphafold/alphafold3/model
/carc/scratch/shared/alphafold/alphafold3/model
├── af3.bin
```

af3.bin file is essentially the **“secret sauce”**: it contains the trained neural network weights from DeepMind, which encode all of the learning from vast protein structure and sequence data, and without which AlphaFold 3 cannot perform structure prediction.

```
tree /carc/scratch/shared/alphafold/alphafold3/model
/carc/scratch/shared/alphafold/alphafold3/model
├── af3.bin
```

This is the file you are not allowed to copy from its current location and which we protect with group membership.

If you have any questions about this please let us know.

```
mfricke@easley:~/workshops/alphafold3 [master ?]$ sbatch slurm/test_gpu_l40s.slurm
Submitted batch job 44987

mfricke@easley:~/workshops/alphafold3 [master ?]$ watch squeue --me

Every 2.0s: squeue --
me                                                  easley: Tue Sep 16
12:42:36 2025

             JOBID PARTITION     NAME     USER      ST TIME   NODES NODELIST(REASON)
             44987 l40s          alphafol  vanilla  R  1:13   1     easley056
```

test_gpu_l40s.slurm runs Alphafold3s test suite.

cat slurm/test_gpu_l40s.slurm

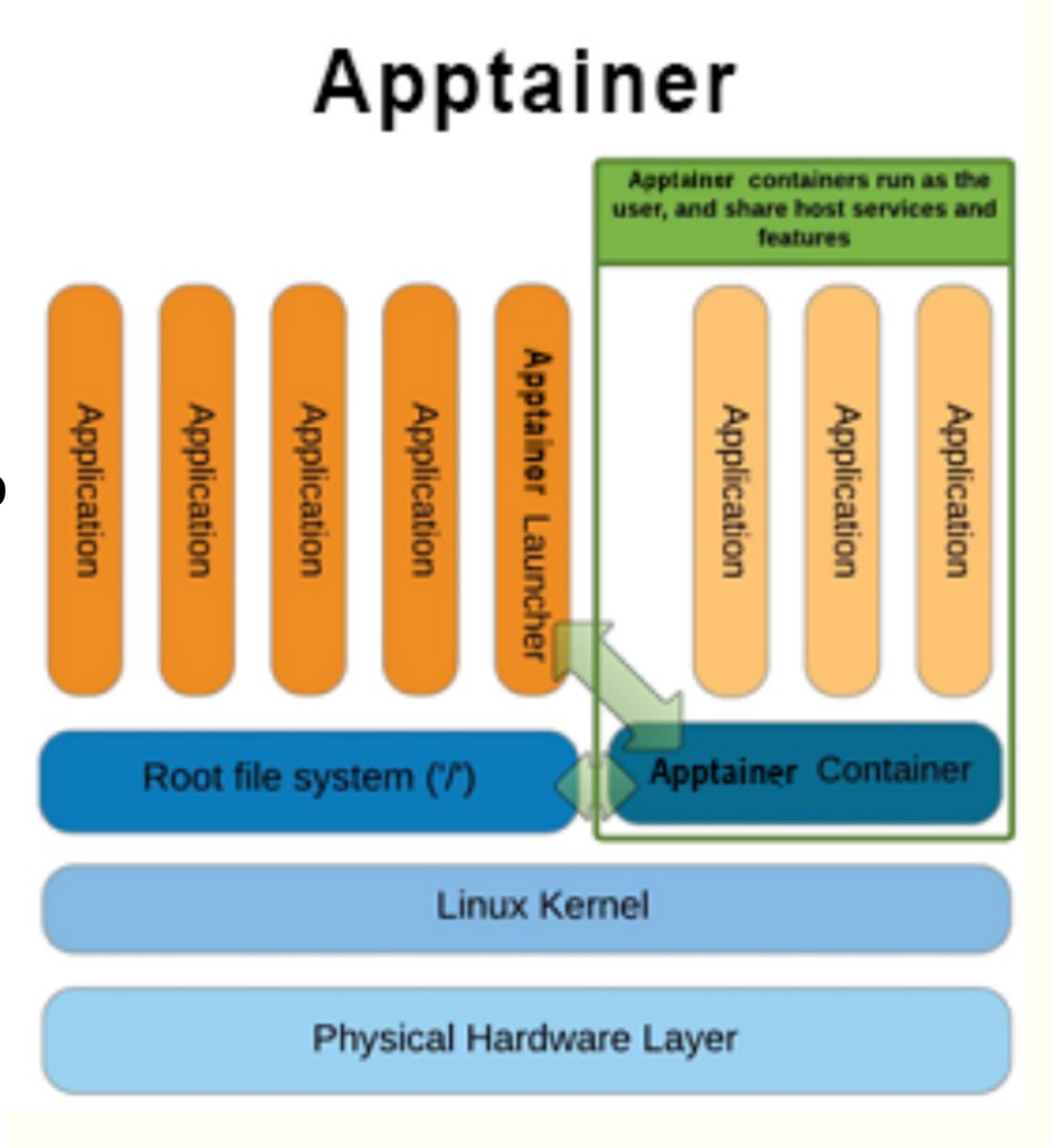
```
#!/bin/bash
#SBATCH --job-name=alphafold_run
#SBATCH --time=10:00
#SBATCH --partition=l40s
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --gres=gpu:l40s:1                # Request 1 L40S GPU
#SBATCH --cpus-per-task=1
#SBATCH --mem=32G
#SBATCH --mail-user=yourusername@unm.edu
#SBATCH --mail-type=ALL
```

```
# Fail on first error
set -euo pipefail

# Paths
MODEL_DIR=/carc/scratch/shared/alphafold/alphafold3/model/
DB_DIR=/carc/scratch/shared/alphafold/alphafold3/db/
APPTAINER_PATH=/projects/shared/apptainer/alphafold3.sif


echo Running with:
echo MODEL_DIR=$MODEL_DIR
echo DB_DIR=$DB_DIR
echo APPTAINER_PATH=$APPTAINER_PATH


# Load Apptainer
module load gcc/14.2.0-cuda-jeua apptainer


apptainer exec --nv \
    --bind $MODEL_DIR:/root/models \
    --bind $DB_DIR:/root/public_databases \
    $APPTAINER_PATH python /app/alphafold/run_alphafold_test.py \
    --model_dir=/root/models \
    --db_dir=/root/public_databases
```

- **Apptainer containers encapsulate all the supporting software needed for an application in a single file.**

- **We have to *bind* directories on the host to directories inside the container so the program inside the container can access data outside the container:**

```
--bind $MODEL_DIR:/root/models
--bind $DB_DIR:/root/public_databases
```

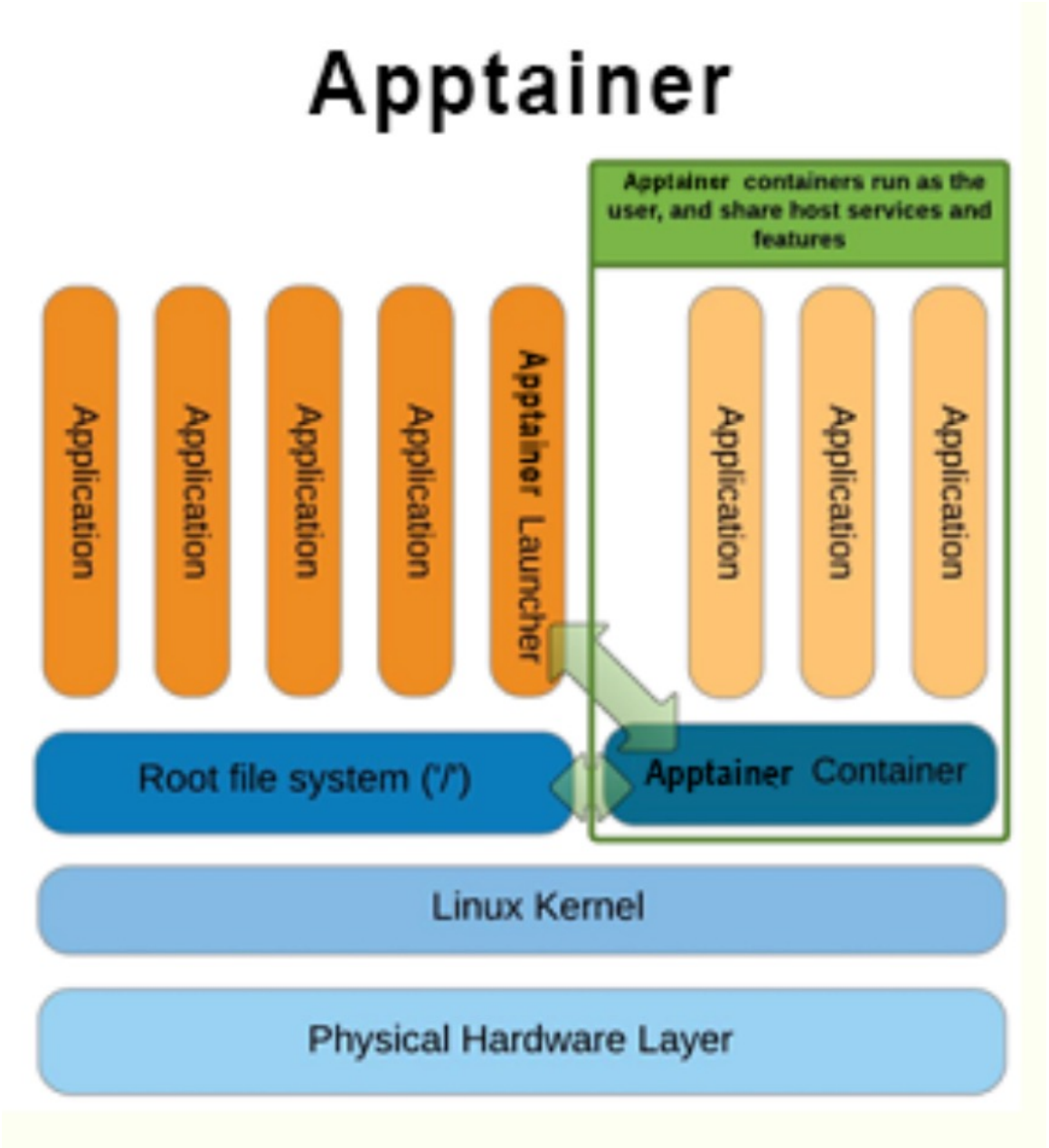


```
# Fail on first error
set -euo pipefail

# Paths
MODEL_DIR=/carc/scratch/shared/alphafold/alphafold3/model/
DB_DIR=/carc/scratch/shared/alphafold/alphafold3/db/
APPTAINER_PATH=/projects/shared/apptainer/alphafold3.sif


echo Running with:
echo MODEL_DIR=$MODEL_DIR
echo DB_DIR=$DB_DIR
echo APPTAINER_PATH=$APPTAINER_PATH


# Load Apptainer
module load gcc/14.2.0-cuda-jeua apptainer


apptainer exec --nv \
    --bind $MODEL_DIR:/root/models \
    --bind $DB_DIR:/root/public_databases \
    $APPTAINER_PATH python /app/alphafold/run_alphafold_test.py \
    --model_dir=/root/models \
    --db_dir=/root/public_databases
```

•**Apptainer executes a container (nv says use the GPU):**

```
apptainer exec --nv \

$APPTAINER_PATH python
/app/alphafold/run_alphafold_test.py \
    --model_dir=/root/models \
    --db_dir=/root/public_databases
```

The python code inside the container is referring to the internal directories we bound to the host's directories.



```
mfricke@easley:~/workshops/alphafold3 [master ?]$ cat slurm-44987.out


[       OK ] InferenceTest.test_model_inference
[ RUN      ] InferenceTest.test_no_chains_in_input
[       OK ] InferenceTest.test_no_chains_in_input
[ RUN      ] InferenceTest.test_process_fold_input_runs_only_data_pipeline
[       OK ] InferenceTest.test_process_fold_input_runs_only_data_pipeline
[ RUN      ] InferenceTest.test_process_fold_input_runs_only_inference
[       OK ] InferenceTest.test_process_fold_input_runs_only_inference
[ RUN      ] InferenceTest.test_write_input_json
[       OK ] InferenceTest.test_write_input_json
----------------------------------------------------------------------
Ran 9 tests in 243.287s


OK
```

```
Running data pipeline...
Processing chain A
Processing chain A took 0.00 seconds
Processing chain B
Processing chain B took 0.00 seconds
Output directory: <absl.testing.absltest._TempDir object at 0x149809d963d0>
Writing model input JSON to <absl.testing.absltest._TempDir object at 0x149809d963d0>
Skipping inference...
Done processing fold input 5tgy.
Processing fold input 5tgy
Checking we can load the model parameters...
Skipping data pipeline...
Output directory: <absl.testing.absltest._TempDir object at 0x149809d64f50>
Writing model input JSON to <absl.testing.absltest._TempDir object at 0x149809d64f50>
Predicting 3D structure for 5tgy for seed(s) (1234,)...
Featurising data for seeds (1234,)...
```

cat slurm/tyra_gpu_h100.slurm

```
#!/bin/bash
#SBATCH --job-name=alphafold_run
#SBATCH --time=8:00:00
#SBATCH --partition=h100
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --gres=gpu:h100:1    # Allocate H100 GPU
#SBATCH --cpus-per-task=64
#SBATCH --mem=32G
#SBATCH --mail-user=yourusername@unm.edu
#SBATCH --mail-type=ALL


# Fail on first error
set -euo pipefail


# Paths
INPUT_FILE=tyrA_flu.json
INPUT_DIR=$HOME/workshops/alphafold3/data/
OUTPUT_DIR=/carc/scratch/users/$USER/alphafold3/output/
MODEL_DIR=/carc/scratch/shared/alphafold/alphafold3/model/
DB_DIR=/carc/scratch/shared/alphafold/alphafold3/db/
APPTAINER_PATH=/projects/shared/apptainer/alphafold3.sif

echo Running with:
echo INPUT_DIR/INPUT_FILE=$INPUT_DIR/$INPUT_FILE
echo OUTPUT_DIR=$OUTPUT_DIR
echo MODEL_DIR=$MODEL_DIR
echo DB_DIR=$DB_DIR
echo APPTAINER_PATH=$APPTAINER_PATH


# Load Apptainer
module load gcc/14.2.0-cuda-jeua apptainer

# Ensure output directory exists
mkdir -p "$OUTPUT_DIR"

# Run AlphaFold 3 with Apptainer
apptainer exec --nv \
        --bind $INPUT_DIR:/root/af_input \
        --bind $OUTPUT_DIR:/root/af_output \
        --bind $MODEL_DIR:/root/models \
        --bind $DB_DIR:/root/public_databases \
        $APPTAINER_PATH \
        python /app/alphafold/run_alphafold.py \
        --json_path=/root/af_input/$INPUT_FILE \
        --model_dir=/root/models \
        --output_dir=/root/af_output \
        --db_dir=/root/public_databases
```

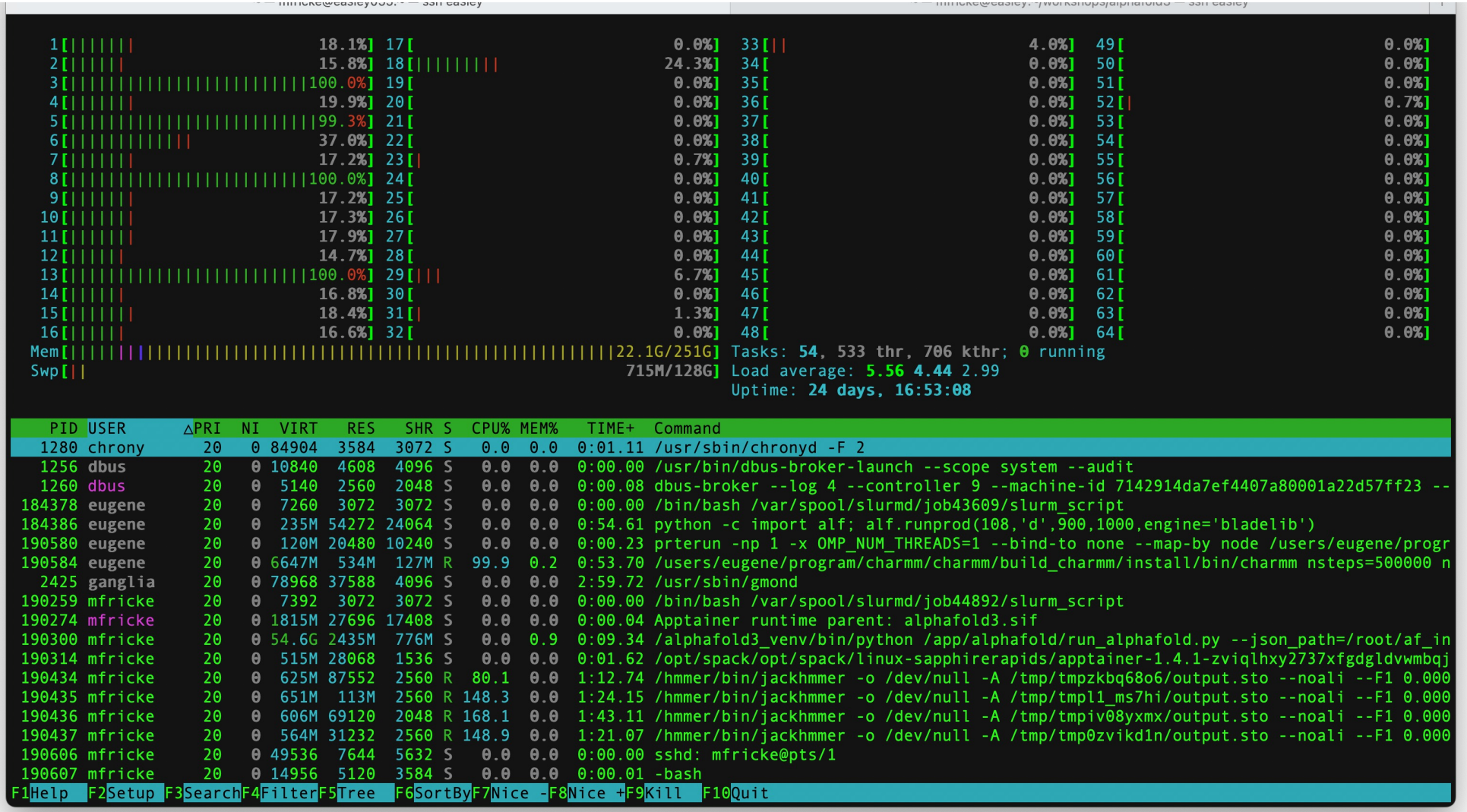
Call run_alphafold.py with a .json input file containing an amino acid sequence (tyrA)

```
mfricke@easley:~/workshops/alphafold3 [master ?]$ sbatch slurm/short_gpu_l40s.slurm
Submitted batch job 44889
mfricke@easley:~/workshops/alphafold3 [master ?]$ sbatch slurm/tyra_gpu_h100.slurm
Submitted batch job 44895
```

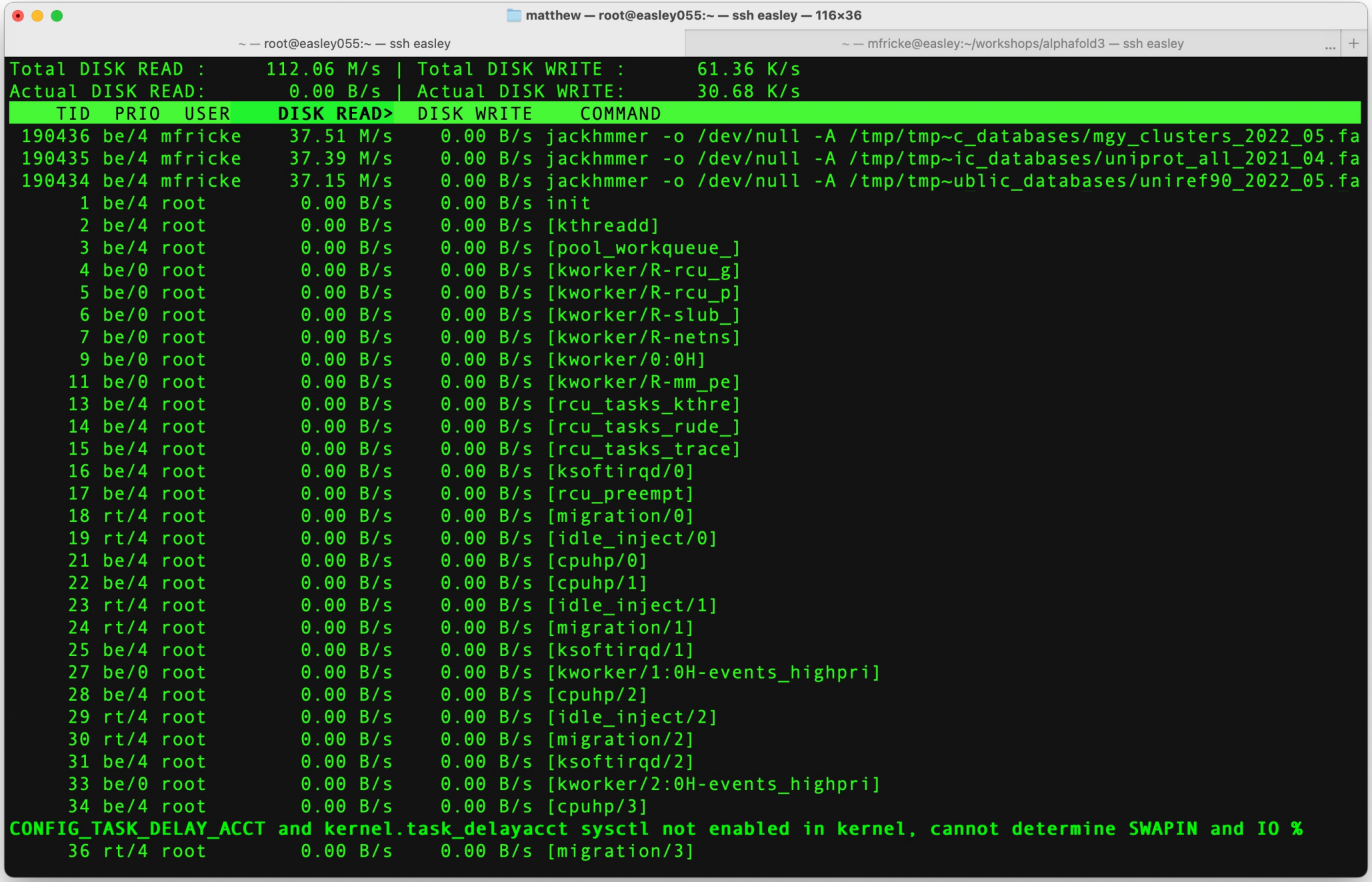
```
vanilla@easley:~/workshops/alphafold3 [master !?+]$ squeue --me
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
             44889      h100 alphafol  vanilla  R      55:35      1 easley052
             44895      l40s alphafol  vanilla  R       6:52      1 easley055
```

```
vanilla@easley:~/workshops/alphafold3 [master !?+]$ tail -f slurm-44892.out
I0916 07:10:57.953472 22786576286848 xla_bridge.py:895] Unable to initialize backend 'tpu': INTERNAL: Failed to
open libtpu.so: libtpu.so: cannot open shared object file: No such file or directory
I0916 07:11:09.198485 22786576286848 pipeline.py:40] Getting protein MSAs for sequence MKTAYIAKQRQISFVKSHFS
I0916 07:11:09.201956 22782817535552 jackhmmer.py:78] Query sequence: MKTAYIAKQRQISFVKSHFS
I0916 07:11:09.202092 22782800815680 jackhmmer.py:78] Query sequence: MKTAYIAKQRQISFVKSHFS
I0916 07:11:09.202167 22782796613184 jackhmmer.py:78] Query sequence: MKTAYIAKQRQISFVKSHFS
I0916 07:11:09.202351 22782798714432 jackhmmer.py:78] Query sequence: MKTAYIAKQRQISFVKSHFS
I0916 07:11:09.203024 22782817535552 subprocess_utils.py:68] Launching subprocess "/hmmer/bin/jackhmmer -o
/dev/null -A /tmp/tmpzkbq68o6/output.sto --noali --F1 0.0005 --F2 5e-05 --F3 5e-07 --cpu 8 -N 1 -E 0.0001 --incE
0.0001 /tmp/tmpzkbq68o6/query.fasta /root/public_databases/uniref90_2022_05.fa"
I0916 07:11:09.203206 22782796613184 subprocess_utils.py:68] Launching subprocess "/hmmer/bin/jackhmmer -o
/dev/null -A /tmp/tmpl1_ms7hi/output.sto --noali --F1 0.0005 --F2 5e-05 --F3 5e-07 --cpu 8 -N 1 -E 0.0001 --incE
0.0001 /tmp/tmpl1_ms7hi/query.fasta /root/public_databases/uniprot_all_2021_04.fa"
I0916 07:11:09.203538 22782800815680 subprocess_utils.py:68] Launching subprocess "/hmmer/bin/jackhmmer -o
/dev/null -A /tmp/tmpiv08yxmx/output.sto --noali --F1 0.0005 --F2 5e-05 --F3 5e-07 --cpu 8 -N 1 -E 0.0001 --incE
0.0001 /tmp/tmpiv08yxmx/query.fasta /root/public_databases/mgy_clusters_2022_05.fa"
I0916 07:11:09.203737 22782798714432 subprocess_utils.py:68] Launching subprocess "/hmmer/bin/jackhmmer -o
/dev/null -A /tmp/tmp0zvikd1n/output.sto --noali --F1 0.0005 --F2 5e-05 --F3 5e-07 --cpu 8 -N 1 -E 0.0001 --incE
0.0001 /tmp/tmp0zvikd1n/query.fasta /root/public_databases/bfd-first_non_consensus_sequences.fasta"
I0916 07:18:20.054697 22782798714432 subprocess_utils.py:97] Finished Jackhmmer in 430.851 seconds
```

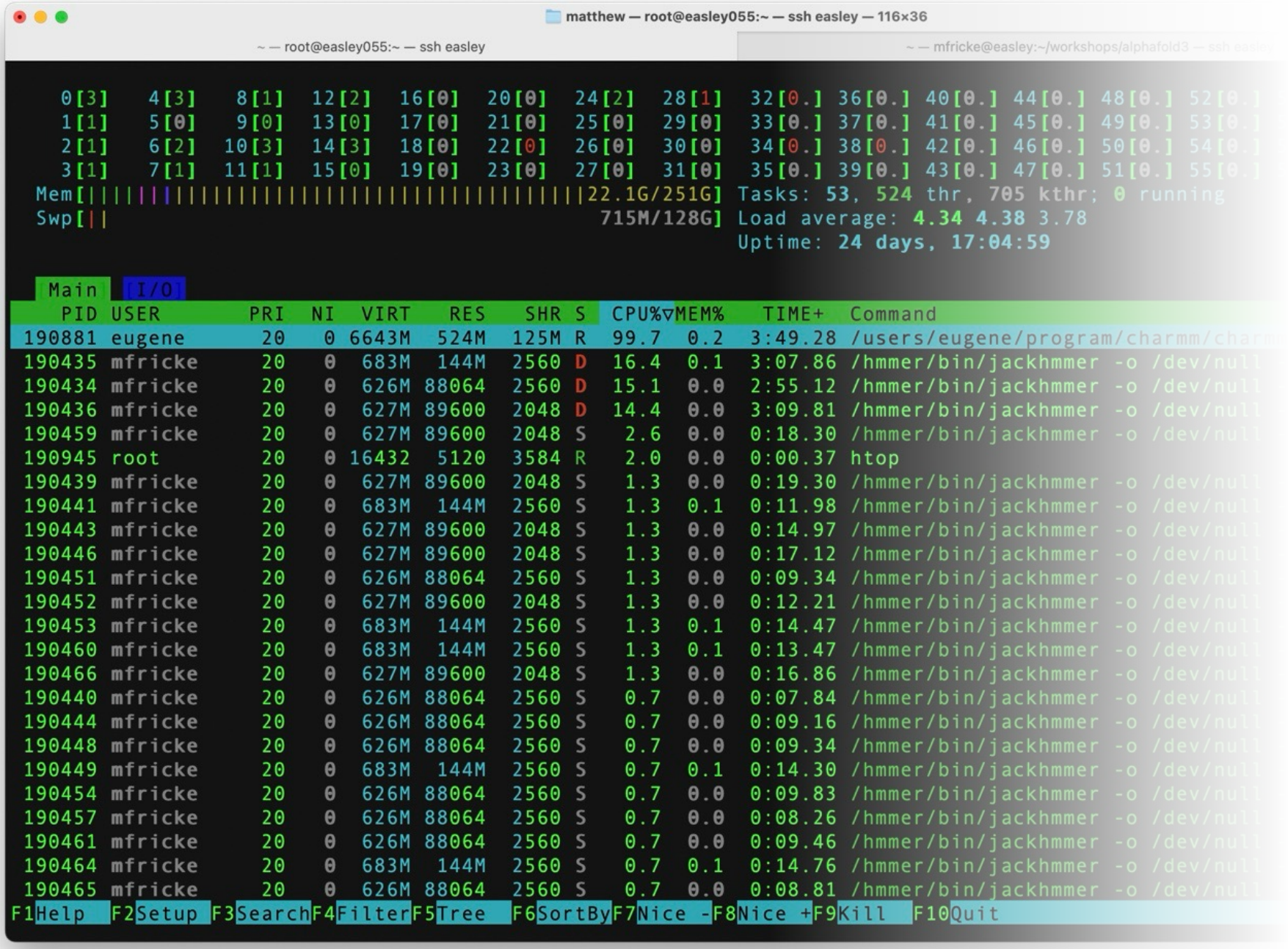
ssh into the compute node you were allocated (easley055 in this case)
So we can look at resource usage with tools like **htop**:



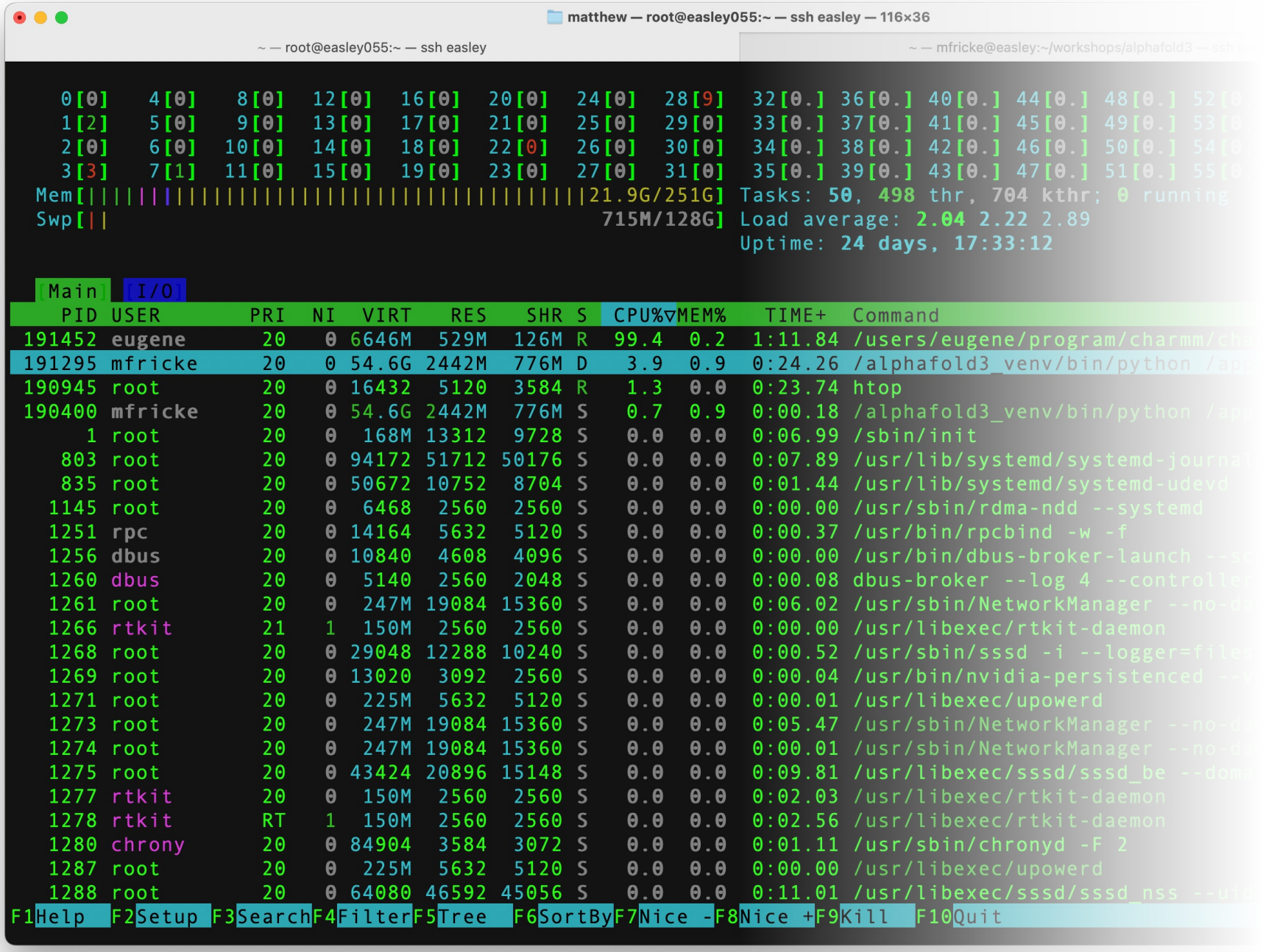
- The jackhmmer processes are each using 140–170% CPU, indicating they are multi-threaded and keeping cores busy.



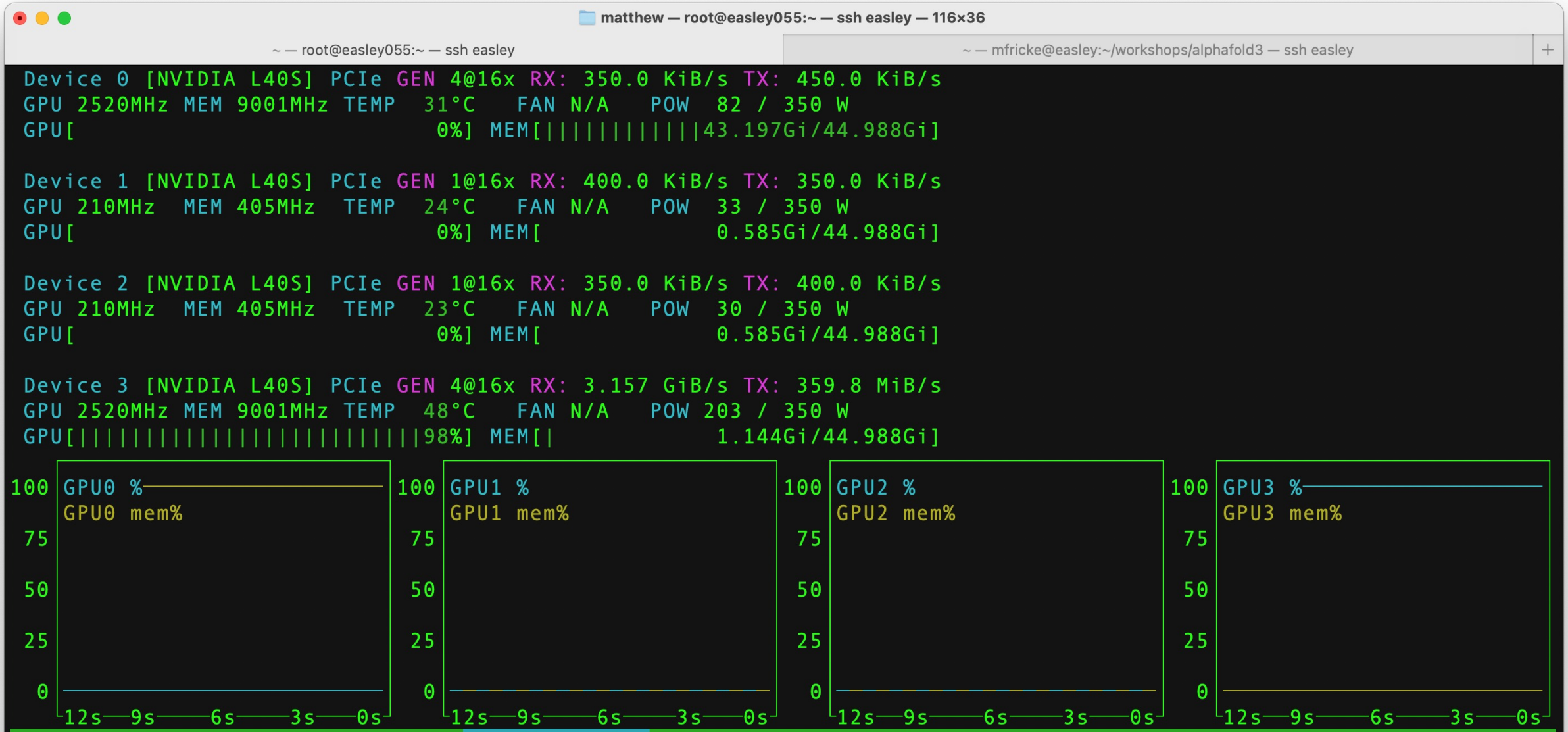
jackhmmer processes are doing **heavy sequential reads** from the sequence databases



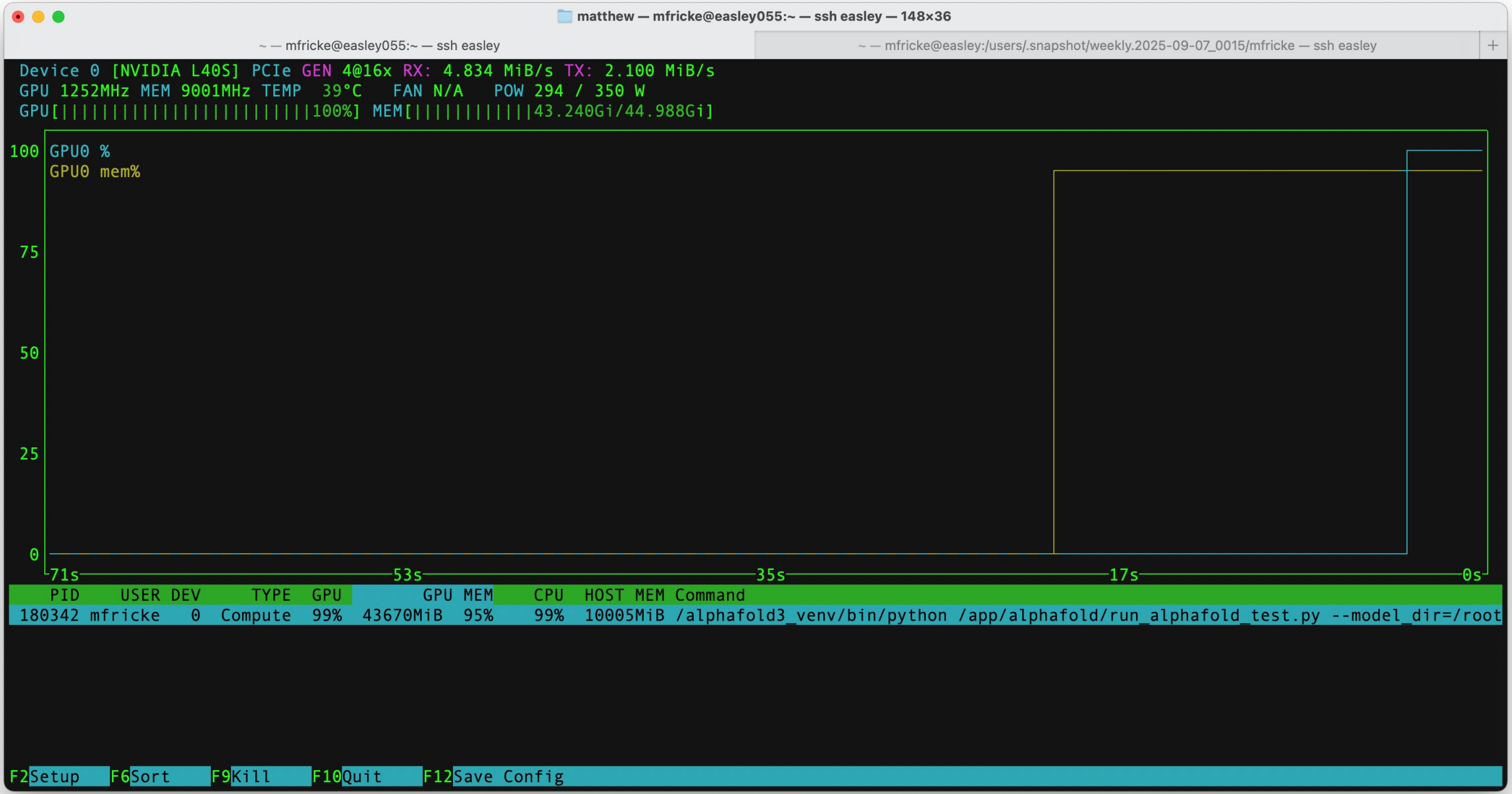
- Most of the jackhmmr processes are in the S (sleeping) or D (uninterruptable sleep) status, meaning they are waiting for data to be read from disk before resuming.



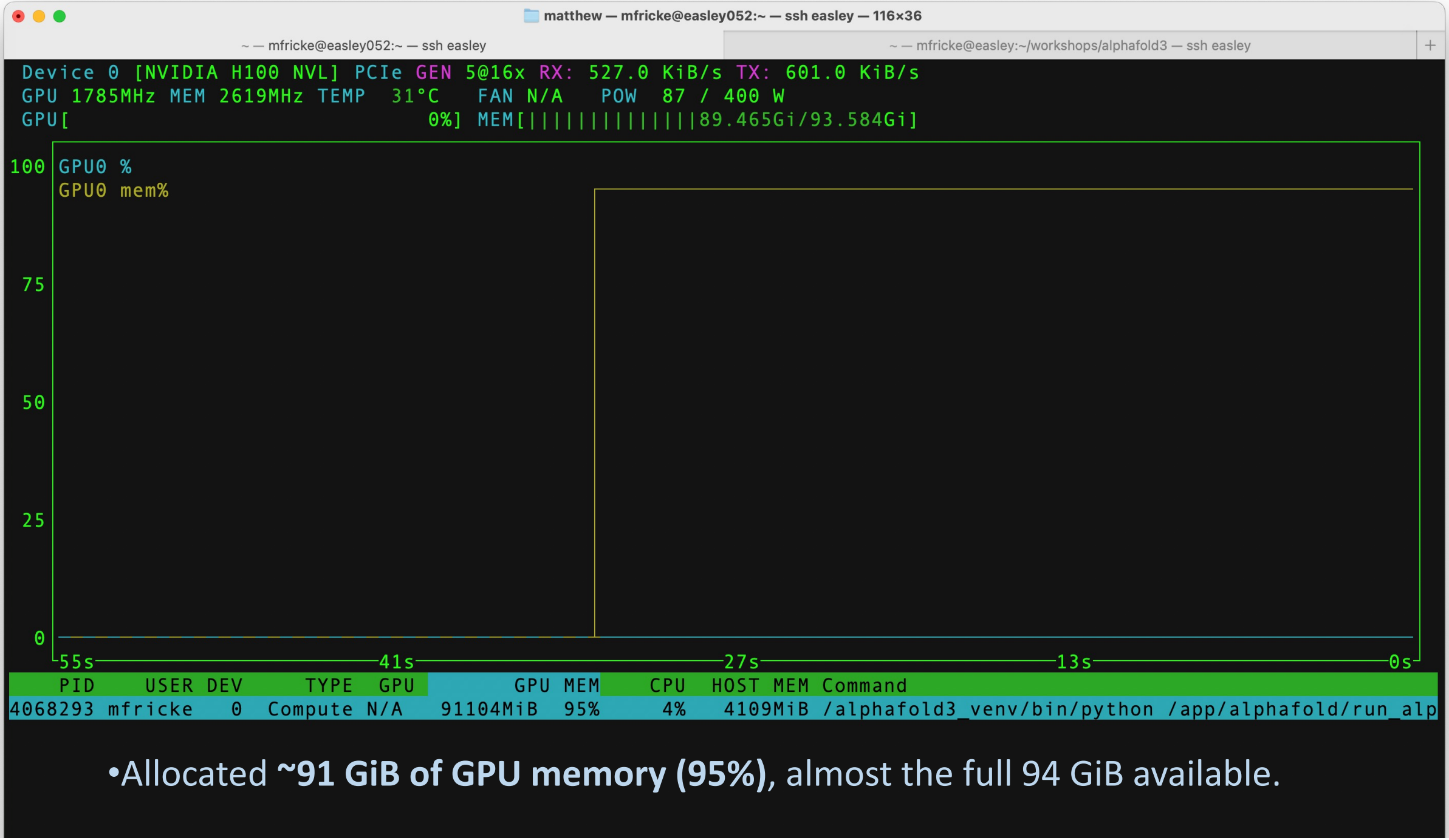
AlphaFold3 run is **I/O-bound at this stage**, stalled on disk access rather than compute



nvtop :GPU0 (your AlphaFold3 run) is consuming **~43.6 GiB VRAM (95%)**, but the **compute utilization is still 0%** → meaning it has loaded the model weights but isn't actively running inference yet.



nvtop: GPU0 utilization is at 99% → the model inference is now fully running on the GPU.



- Allocated **~91 GiB of GPU memory (95%)**, almost the full 94 GiB available.

```
vanilla@easley:~/workshops/alphafold3 [master !?+]$ cat short_l40s.out
Found local devices: [CudaDevice(id=0)]
Building model from scratch...
Processing 1 fold inputs.
Processing fold input test_peptide
Checking we can load the model parameters...
Running data pipeline...
Processing chain A
Processing chain A took 3531.70 seconds
Output directory: /root/af_output/test_peptide
Writing model input JSON to /root/af_output/test_peptide
Predicting 3D structure for test_peptide for seed(s) (1,)...
Featurising data for seeds (1,)...
Featurising test_peptide with rng_seed 1.
Featurising test_peptide with rng_seed 1 took 0.30 seconds.
Featurising data for seeds (1,) took  6.10 seconds.
Running model inference for seed 1...
Running model inference for seed 1 took  39.11 seconds.
Extracting output structures (one per sample) for seed 1...
Extracting output structures (one per sample) for seed 1 took  0.04 seconds.
Running model inference and extracting output structures for seed 1 took  39.15 seconds.
Running model inference and extracting output structures for seeds (1,) took  39.15 seconds.
Writing outputs for test_peptide for seed(s) (1,)...
Done processing fold input test_peptide.
Done processing 1 fold inputs.
```

```
vanilla@easley:~/workshops/alphafold3 [master !?+]$ cat tyrA_h100.out
Found local devices: [CudaDevice(id=0)]
Building model from scratch...
Processing 1 fold inputs.
Processing fold input 2PV7
Checking we can load the model parameters...
Running data pipeline...
Processing chain A
Processing chain A took 4743.06 seconds
Processing chain B
Processing chain B took 0.07 seconds
Output directory: /root/af_output/2pv7
Writing model input JSON to /root/af_output/2pv7
Predicting 3D structure for 2PV7 for seed(s) (1,)...
Featurising data for seeds (1,)...
Featurising 2PV7 with rng_seed 1.
Featurising 2PV7 with rng_seed 1 took 7.71 seconds.
Featurising data for seeds (1,) took  12.75 seconds.
Running model inference for seed 1...
Running model inference for seed 1 took  56.23 seconds.
Extracting output structures (one per sample) for seed 1...
Extracting output structures (one per sample) for seed 1 took  0.50 seconds.
Running model inference and extracting output structures for seed 1 took  56.74 seconds.
Running model inference and extracting output structures for seeds (1,) took  56.74 seconds.
Writing outputs for 2PV7 for seed(s) (1,)...
Done processing fold input 2PV7.
Done processing 1 fold inputs.
```

L40S (short peptide: 20 aa, short.json)

- **MSA generation (Jackhmmer)**: ~2,266 s (≈ 38 min)
- **Template search/deduplication**: ~1,265 s (≈ 21 min)
- **Chain preprocessing**: 3,532 s (≈ 59 min)
- **GPU inference**: **39 s**
- **Total walltime**: ≈ 2 hr 10 min

Sequence: MKTAYIAKQRQISFVKSHFS
That's **20 amino acids** long.

H100 (2PV7, 596 aa, tyrA_flu.json)

- **MSA generation (Jackhmmer)**: ~2,795 s (≈ 46 min)
- **Template search/deduplication**: ~1,948 s (≈ 32 min)
- **Chain A preprocessing**: 4,743 s (≈ 79 min)
- **GPU inference**: **56 s**
- **Total walltime**: ≈ 2 hr 45 min

- **596 amino acids**

MSA stage dominates runtime on both cards (jackhmmer + template search).
1.That's CPU-bound and I/O-heavy, so the GPU type doesn't change those numbers much.
2.Both runs spent ~60–70% of wall time in MSAs.
3.Chain preprocessing (bucket sizing, padding, feature construction) is similar scale on both runs:
4.~59 min on short peptide vs ~79 min on the 596-aa protein.
5.The scaling is more sensitive to sequence length than GPU model.

GPU inference speed:
1.L40S: 39 s (tiny peptide, 20 aa).
2.H100: 56 s (much larger, 596 aa).
3.So inference scales with sequence length, and the H100 only had to handle the larger input; the raw inference throughput is probably **much higher** than the L40S (it just wasn't stressed by the small peptide).
4.Overall takeaway:
5.H100 gives only a small runtime difference for these small-to-medium sequences because most time is in MSAs (CPU).
6.L40S vs H100 inference: if you ran a **larger protein (e.g. >1,500 aa)** or multiple chains, you'd see the H100 pull ahead significantly in GPU inference (minutes vs tens of minutes).

```
vanilla@easley:/carc/scratch/users/vanilla/alphafold3/output $ tree 2pv7/
2pv7/
├── 2pv7_confidences.json
├── 2pv7_data.json
├── 2pv7_model.cif
├── 2pv7_summary_confidences.json
├── ranking_scores.csv
├── seed-1_sample-0
│   ├── confidences.json
│   ├── model.cif
│   └── summary_confidences.json
├── seed-1_sample-1
│   ├── confidences.json
│   ├── model.cif
│   └── summary_confidences.json
├── seed-1_sample-2
│   ├── confidences.json
│   ├── model.cif
│   └── summary_confidences.json
├── seed-1_sample-3
│   ├── confidences.json
│   ├── model.cif
│   └── summary_confidences.json
├── seed-1_sample-4
│   ├── confidences.json
│   ├── model.cif
│   └── summary_confidences.json
└── TERMS_OF_USE.md
```

This output directory contains the predicted 3D structures of protein **2PV7**, with multiple sampled models, their confidence metrics, ranking scores, input/output JSONs, and an accompanying terms-of-use file.

```
vanilla@easley:/carc/scratch/users/vanilla/alphafold3/output $ head
2pv7/ranking_scores.csv
seed,sample,ranking_score
1,0,0.6282640031092119
1,1,0.6348824466104794
1,2,0.6306750479708 41
1,3,0.6300505713451457
1,4,0.6638215342510709
```

This file lists the confidence-based ranking scores for each sampled model of protein **2PV7**, showing which prediction is considered most reliable.

```
vanilla@easley:/carc/scratch/users/vanilla/alphafold3/output $
cp 2pv7/seed-1_sample-4/model.cif
~/workshops/alphafold3/data/sample_output/2pv7.cif
```

Copy the highest ranked output cif somewhere it is easy to find later.
You already have it in your data/sample_output folder so you don’t need to type this

dkoes

Joined May 9, 2016

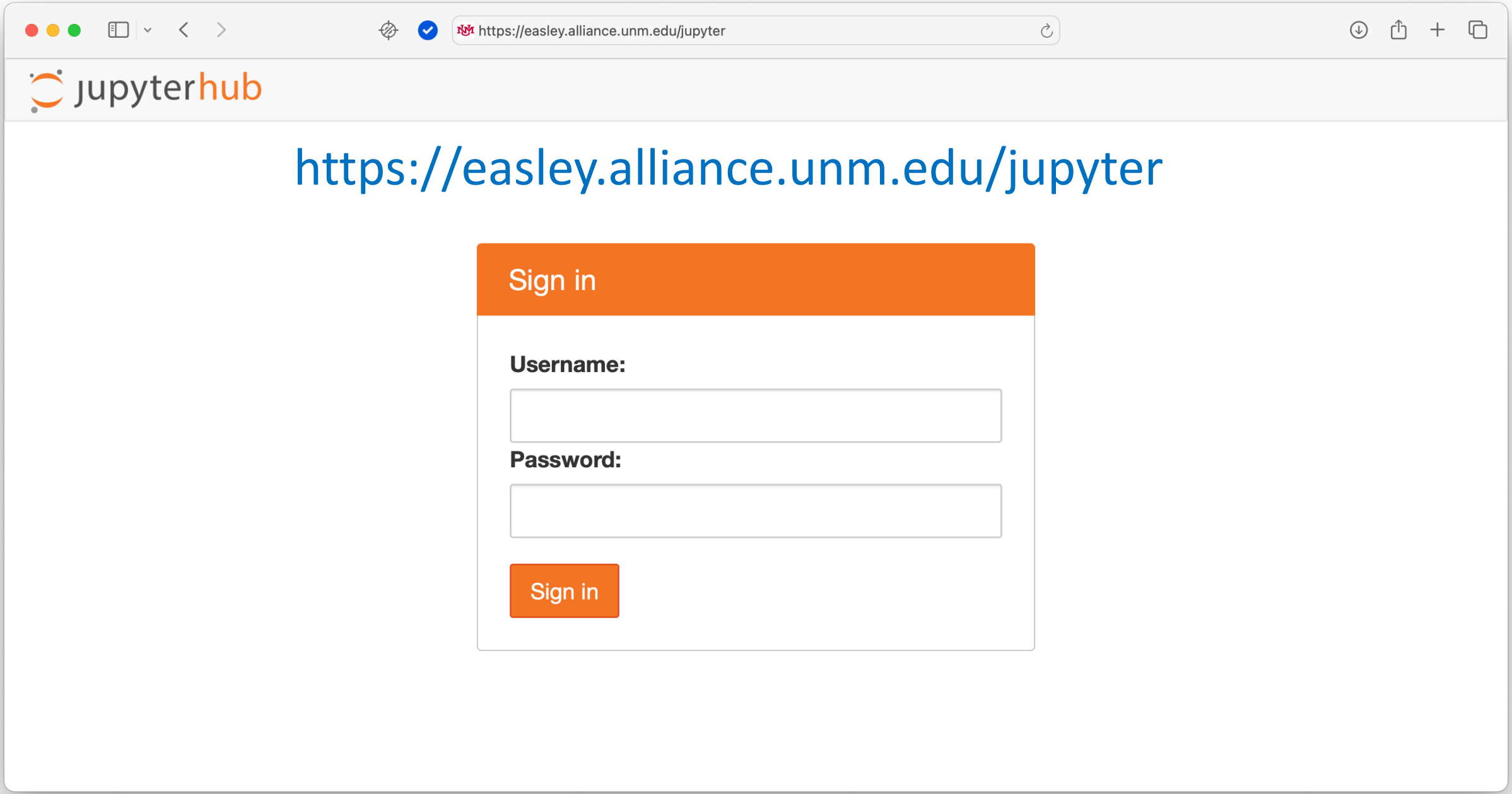
2 projects

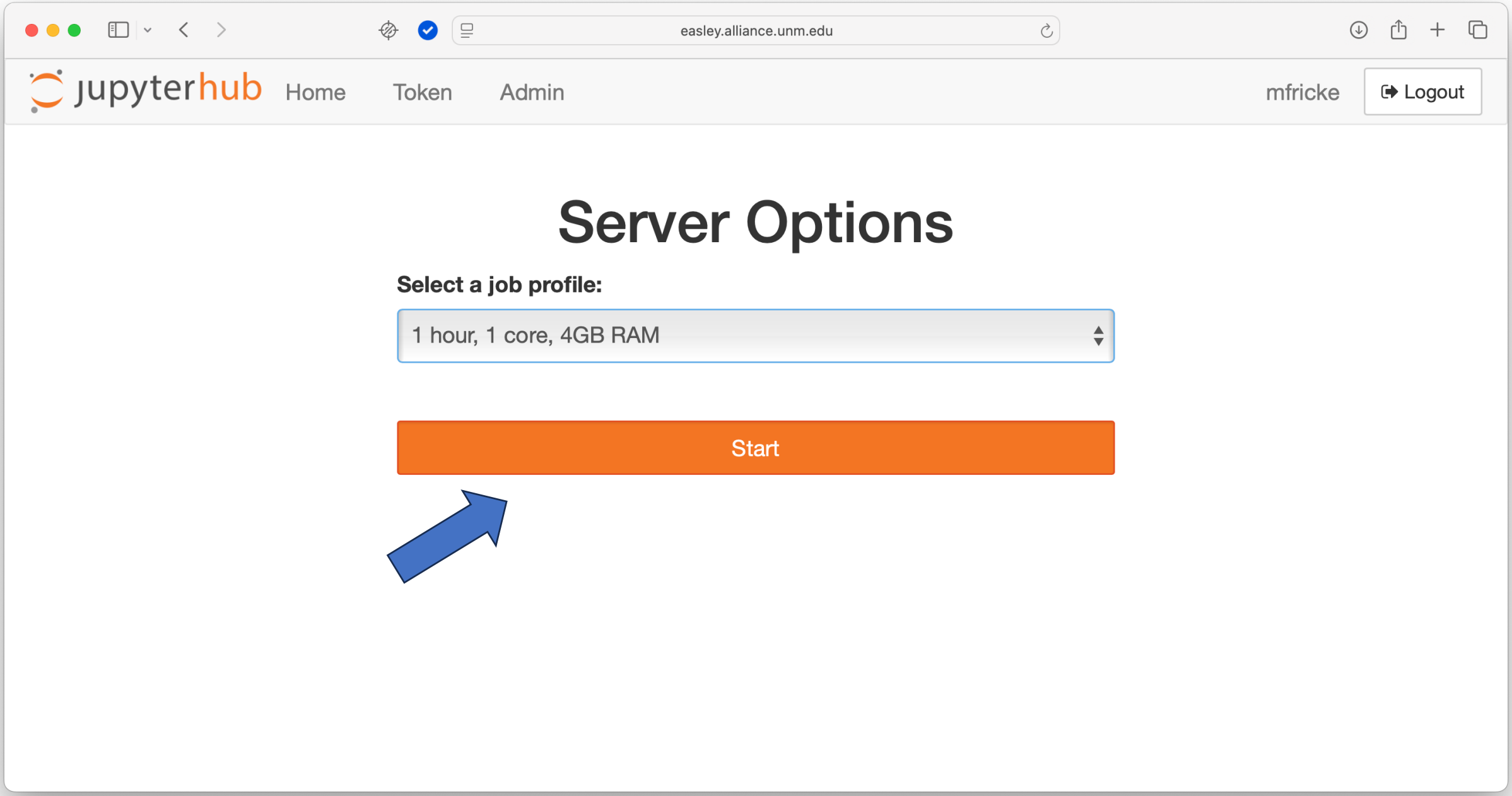
py3Dmol
Last released Jul 31, 2025
An IPython interface for embedding 3Dmol.js views in Jupyter notebooks

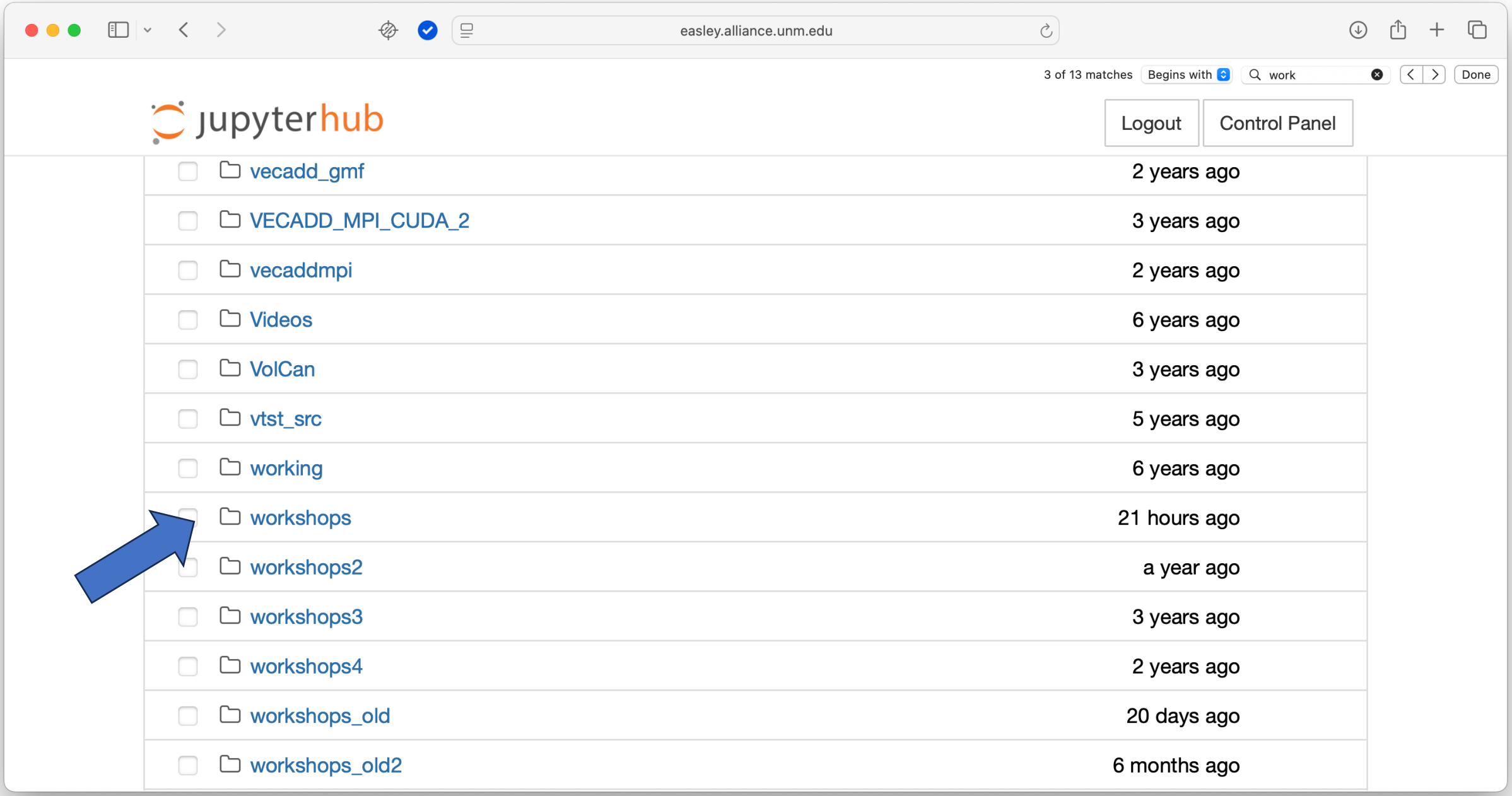
molgrid
Last released Jul 12, 2024
Grid-based molecular modeling library

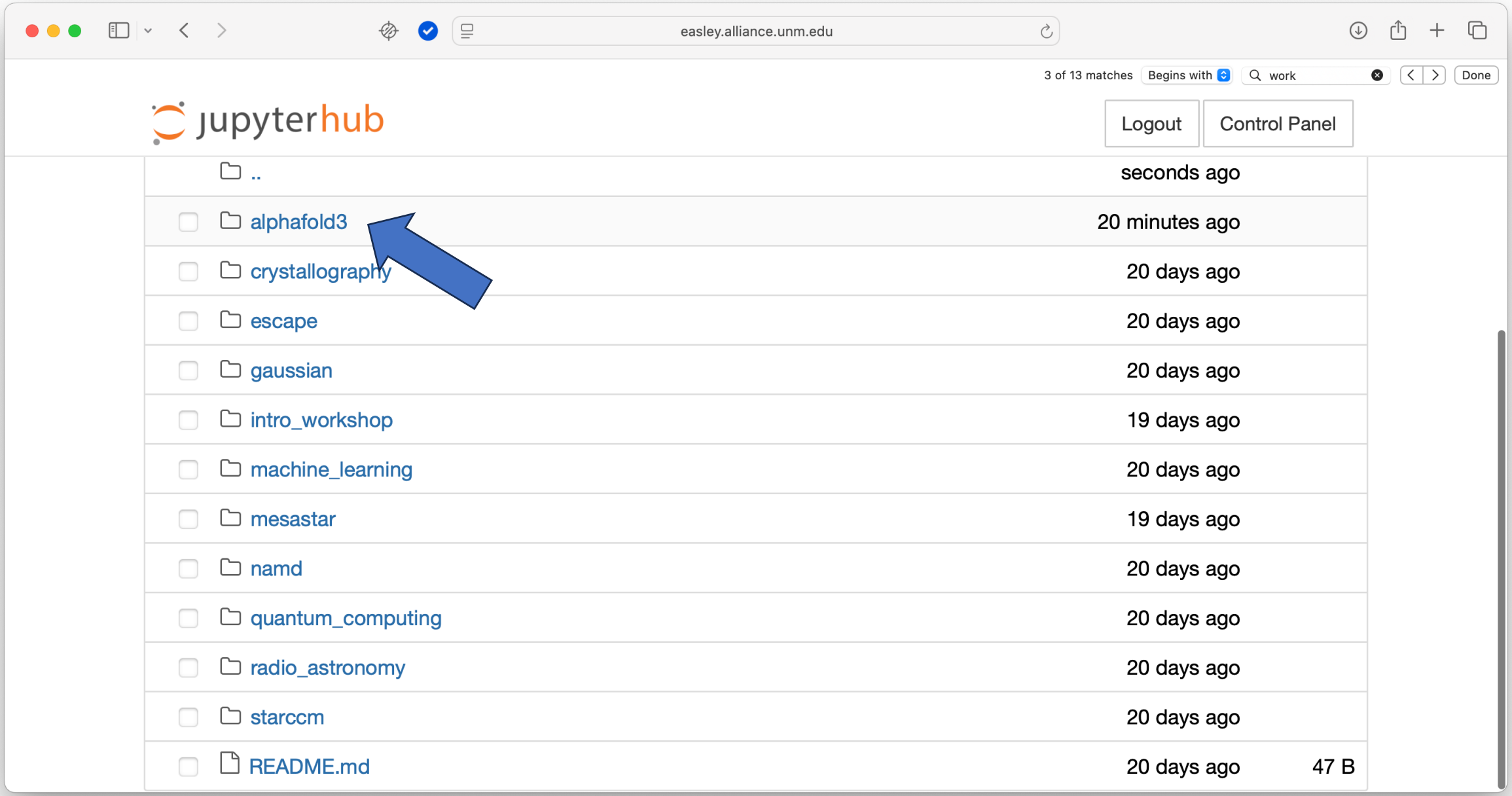
```
vanilla@easley:/carc/scratch/users/vanilla/alphafold3/output $
module load miniconda3


vanilla@easley:/carc/scratch/users/vanilla/alphafold3/output $ conda create --
name mol_view py3Dmol ipykernel --channel conda-forge -y
```

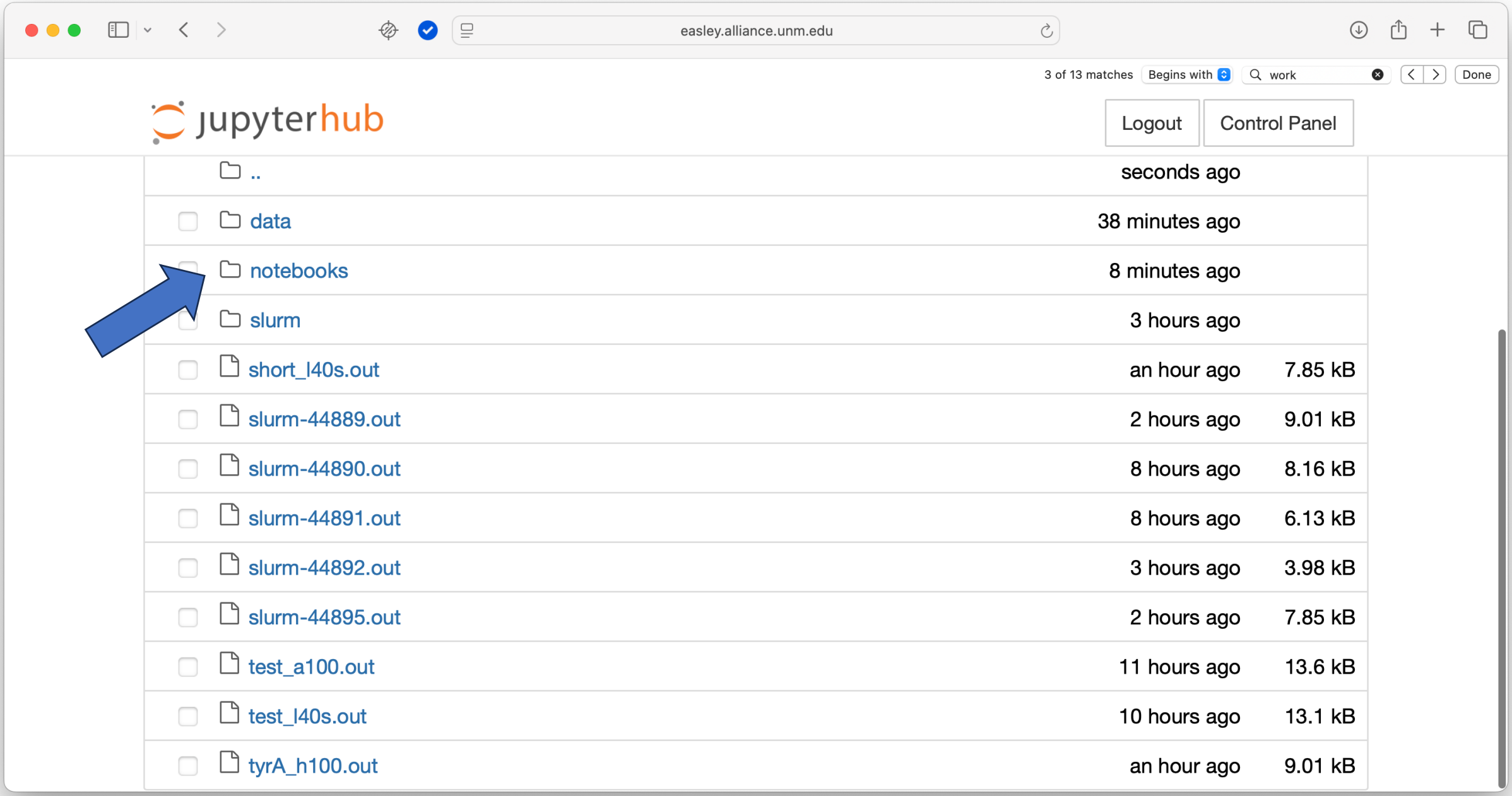
Install the py3Dmol packages

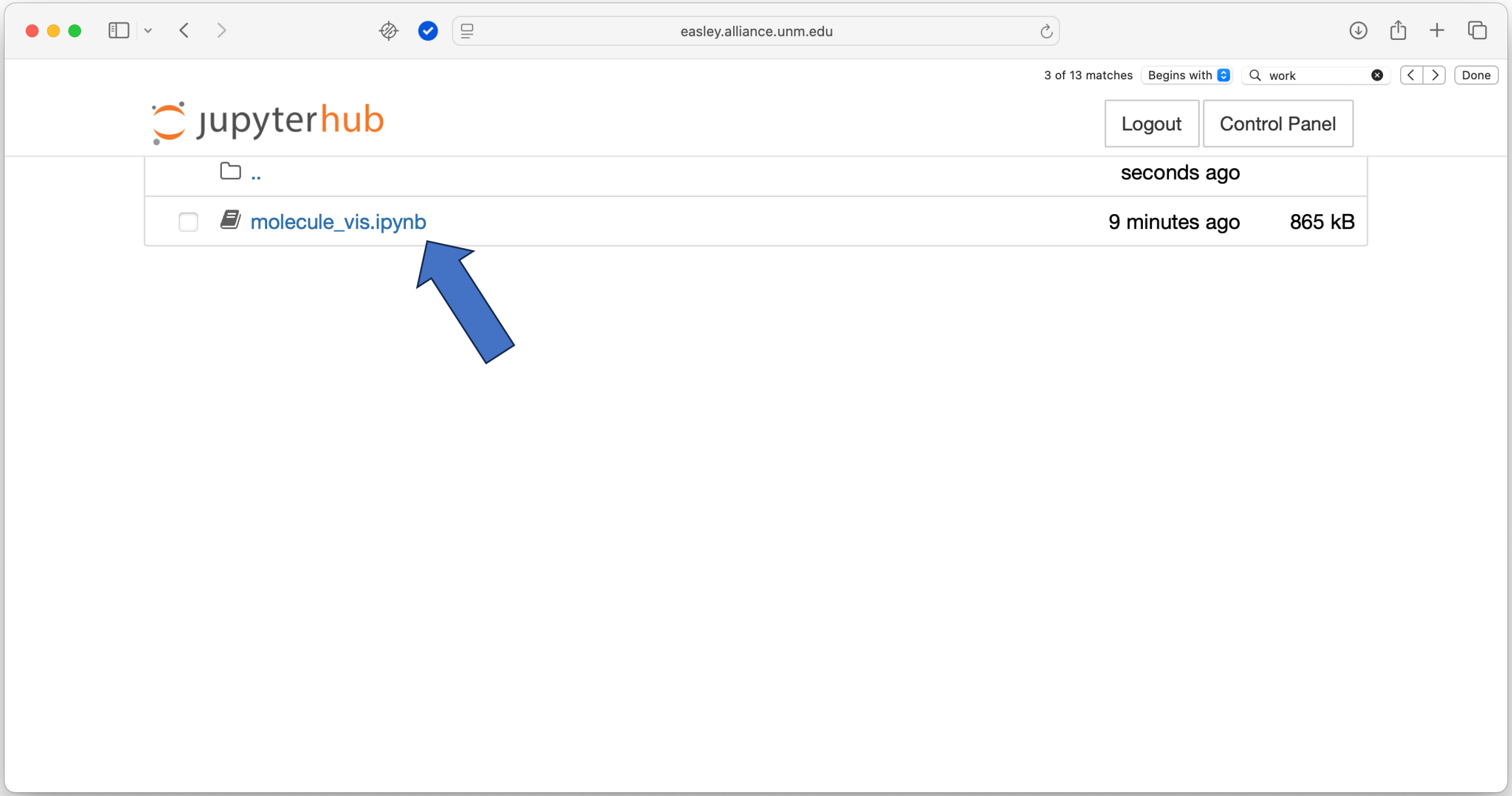

https://easley.alliance.unm.edu/jupyter
jupyterhub
https://easley.alliance.unm.edu/jupyter
Sign in
Username:
Password:
Sign in


easley.alliance.unm.edu
jupyterhub
Home
Token
Admin
mfricke
Logout
Server Options
Select a job profile:
1 hour, 1 core, 4GB RAM
Start


easley.alliance.unm.edu
3 of 13 matches
Begins with
work
Done
jupyterhub
Logout
Control Panel
vecadd_gmf 2 years ago
VECADD_MPI_CUDA_2 3 years ago
vecaddmpi 2 years ago
Videos 6 years ago
VolCan 3 years ago
vtst_src 5 years ago
working 6 years ago
workshops 21 hours ago
workshops2 a year ago
workshops3 3 years ago
workshops4 2 years ago
workshops_old 20 days ago
workshops_old2 6 months ago


easley.alliance.unm.edu
3 of 13 matches
Begins with
work
Done
jupyterhub
Logout
Control Panel
..
seconds ago
alphafold3
20 minutes ago
crystallography
20 days ago
escape
20 days ago
gaussian
20 days ago
intro_workshop
19 days ago
machine_learning
20 days ago
mesastar
19 days ago
namd
20 days ago
quantum_computing
20 days ago
radio_astronomy
20 days ago
starccm
20 days ago
README.md
20 days ago
47 B


easley.alliance.unm.edu
3 of 13 matches
Begins with
work
Done
jupyterhub
Logout
Control Panel
..
seconds ago
data
38 minutes ago
notebooks
8 minutes ago
slurm
3 hours ago
short_l40s.out
an hour ago
7.85 kB
slurm-44889.out
2 hours ago
9.01 kB
slurm-44890.out
8 hours ago
8.16 kB
slurm-44891.out
8 hours ago
6.13 kB
slurm-44892.out
3 hours ago
3.98 kB
slurm-44895.out
2 hours ago
7.85 kB
test_a100.out
11 hours ago
13.6 kB
test_l40s.out
10 hours ago
13.1 kB
tyrA_h100.out
an hour ago
9.01 kB


easley.alliance.unm.edu
3 of 13 matches
Begins with
work
Done
jupyterhub
Logout
Control Panel
..
seconds ago
molecule_vis.ipynb
9 minutes ago
865 kB

```
import py3Dmol

cif_file = "/users/vanilla/workshops/alphafold3/data/sample_output/2pv7.cif"

with open(cif_file) as f:
    cif_data = f.read()

view = py3Dmol.view(width=600, height=400)
view.addModel(cif_data, "cif")
view.setStyle({"cartoon": {"color": "spectrum"}})
view.zoomTo()
view.show()
```

jupyterhub molecule_vis Last Checkpoint: 11 minutes ago (autosaved)

Logout | Control Panel

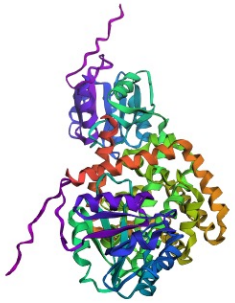
Not Trusted | Python [conda env:.conda-mol_view]

File Edit View Insert Cell Kernel Widgets Help

Run | Code

In [1]:

```
import py3Dmol

cif_file = "/users/mfricke/workshops/alphafold3/data/sample_output/2pv7.cif"

with open(cif_file) as f:
    cif_data = f.read()

view = py3Dmol.view(width=600, height=400)
view.addModel(cif_data, "cif")
view.setStyle({"cartoon": {"color": "spectrum"}})
view.zoomTo()
view.show()
```



In [2]:

```
cif_file = "/users/mfricke/workshops/alphafold3/data/sample_output/short.cif"

with open(cif_file) as f:
    cif_data = f.read()

view = py3Dmol.view(width=600, height=400)
view.addModel(cif_data, "cif")
view.setStyle({"cartoon": {"color": "spectrum"}})
view.zoomTo()
view.show()
```

jupyterhub molecule_vis Last Checkpoint: 11 minutes ago (autosaved)

Logout Control Panel

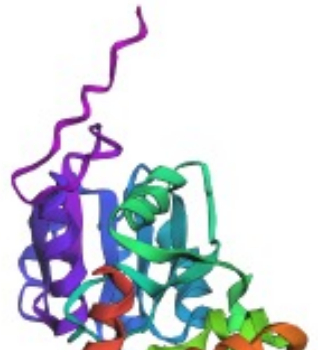
Notebook saved Not Trusted Python [conda env:.conda-mol_view]

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [1]:

```
1  import py3Dmol
2
3  cif_file = "/users/mfricke/workshops/alphafold3/data/sample_output/2pv7.cif"
4
5  with open(cif_file) as f:
6      cif_data = f.read()
7
8  view = py3Dmol.view(width=600, height=400)
9  view.addModel(cif_data, "cif")
10 view.setStyle({"cartoon": {"color": "spectrum"}})
11 view.zoomTo()
12 view.show()
```



jupyterhub molecule_vis Last Checkpoint: 11 minutes ago (autosaved)

Logout | Control Panel

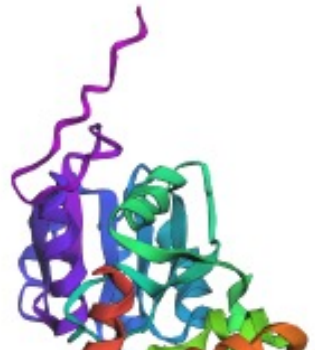
Notebook saved | Not Trusted | Python [conda env:.conda-mol_view]

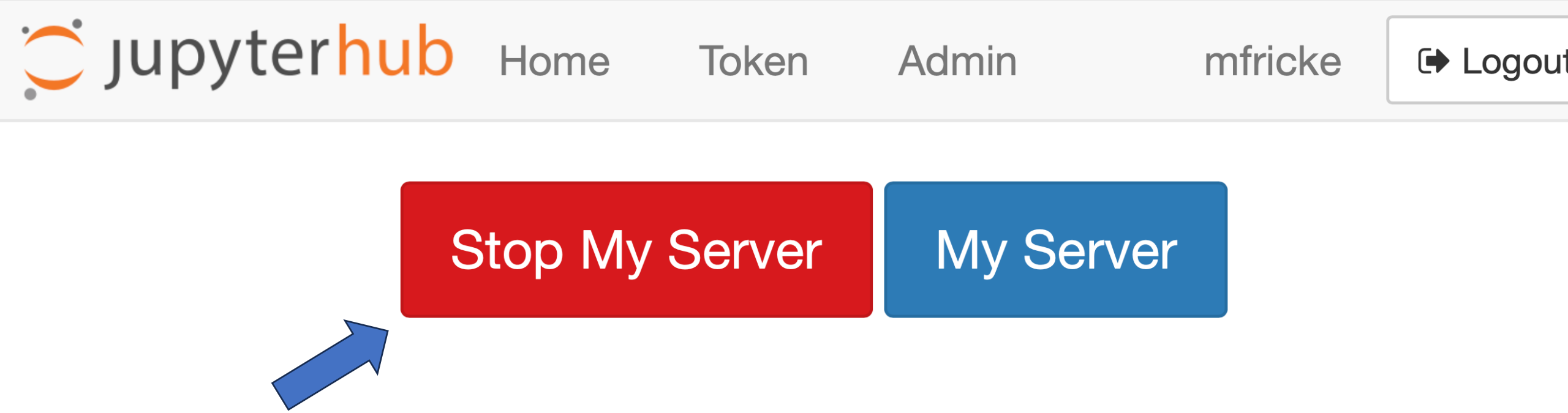
File Edit View Insert Cell Kernel Widgets Help

Run | Code

In [1]:

```
1  import py3Dmol
2
3  cif_file = "/users/mfricke/workshops/alphafold3/data/sample_output/2pv7.cif"
4
5  with open(cif_file) as f:
6      cif_data = f.read()
7
8  view = py3Dmol.view(width=600, height=400)
9  view.addModel(cif_data, "cif")
10 view.setStyle({"cartoon": {"color": "spectrum"}})
11 view.zoomTo()
12 view.show()
```




jupyterhub
Home
Token
Admin
mfricke
Logout
Stop My Server
My Server

You have learned

- how to run programs using the SLURM scheduler
- the difference between interactive and batch jobs
- how to check the status of your jobs
- how to select debug vs general SLURM partitions
- how to ask for the hardware resources you need
- you ran the NAMD molecular dynamics code using SLURM
- You know how to run AlphaFold3 and visualize the results!