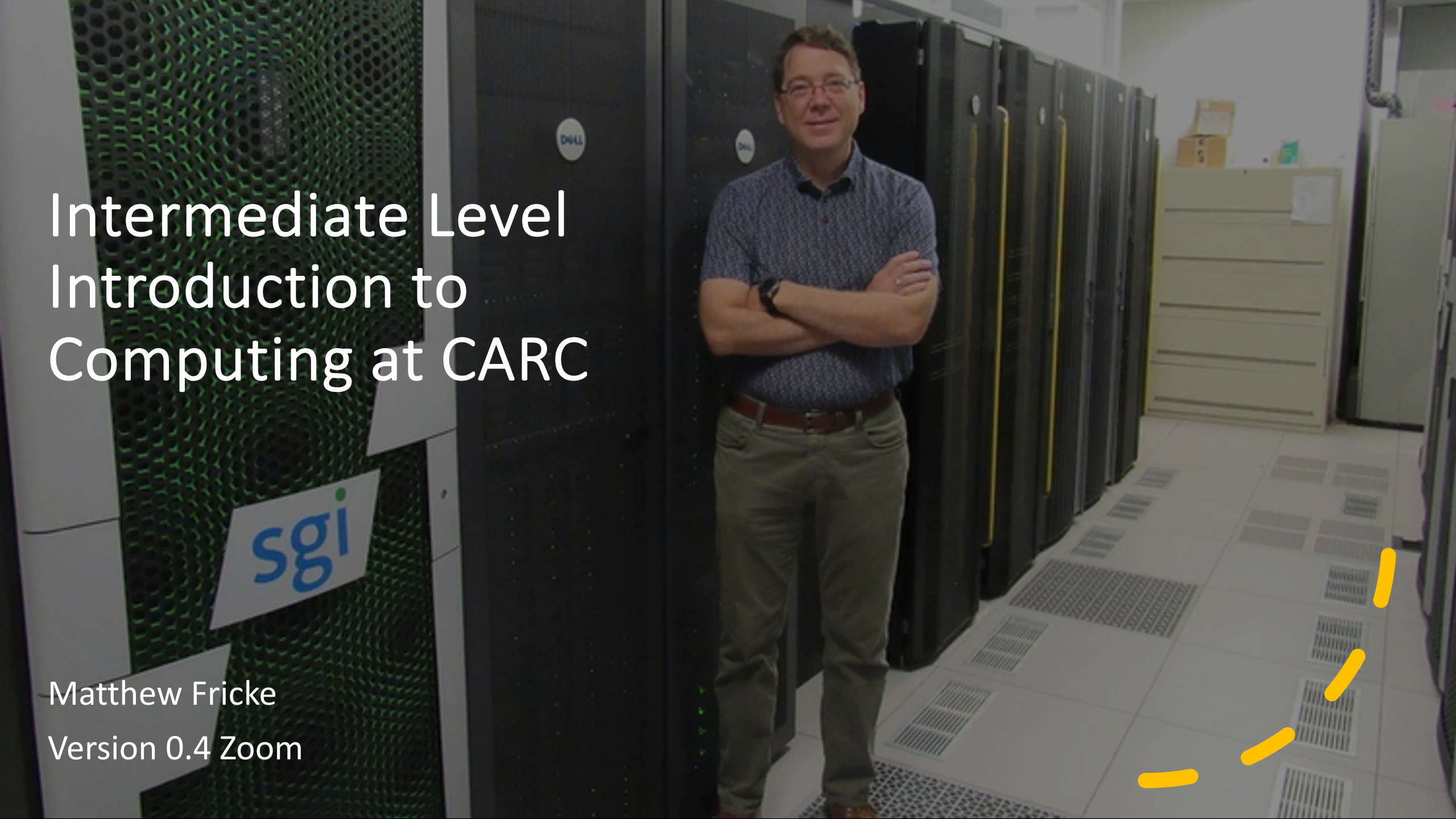


Intermediate Level Introduction to Computing at CARC

Matthew Fricke

Version 0.4 Zoom

Goals

- 1) SLURM scheduler literacy
- 2) Intro to HPC and Supercomputing
- 3) Run Gaussian
- 4) Run Abinit

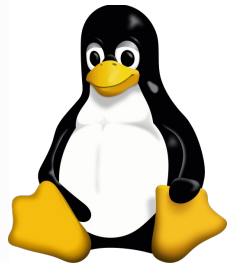
NOTE: We won't cover file transfer, storage systems, module system, conda, PBS. (These are all covered in depth in the video tutorials)

Agenda

MPI

Logging into Easley


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


On windows use PowerShell to get to
the command line.


On Mac or Linux machines run
terminal to access the command line.
```



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       (
```

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      0
```

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

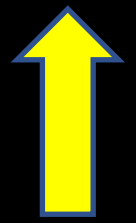
```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      1    mix hopper011
debug        up    4:00:00      1   idle hopper012
```

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
```

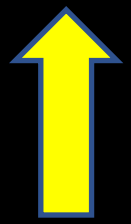
The name of the nodes in the partition

```
[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

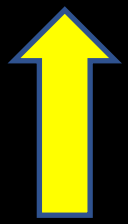
```
sinfo(1)                                                    Slurm
Commands                                                    sinfo(1)

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)
```

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)
                                                                 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  vanilla  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  vanilla  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?**

```
[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 been                                    se
the -G option in
hopper011
Hopper011
```

This command is getting pretty long.

We can use salloc to avoid asking for the same resources every time we use srun.

```
[vanilla@hopper ~]$ salloc --partition debug --nodes 2
--ntasks-per-node 2
salloc: Account not specified in script or
~/.default_slurm_account, using latest project
salloc: Granted job allocation 5251
salloc: Waiting for resource configuration
salloc: Nodes hopper[011-012] are ready for job
[vanilla@hopper ~]$
```

This command is getting pretty long.

We can use salloc to avoid asking for the same resources every time we use srun.

```
[vanilla@hopper ~]$ srun hostname
hopper012
hopper012
hopper011
You have not been allocated GPUs. To request GPUs, use
the -G option in your submission script.
hopper011
[vanilla@hopper ~]$ srun hostname
hopper012
hopper011
hopper012
hopper011
[vanilla@hopper ~]$
```

Now we can use srun over and over without having to ask for a new hardware allocation each time.

```
[vanilla@hopper ~]$ exit
exit
salloc: Relinquishing job allocation 5251
```

Always type exit when you are done with the hardware.
Running salloc inside an allocation gets very confusing.

BREAK

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…

If you already have a workshops directory from a previous CARC workshop then
cd ~/workshops
git pull

```
[vanilla@hopper ~]$ tree workshops
workshops/
├── intro_workshop
│   ├── code
│   │   ├── calcPiMPI.py
│   │   ├── calcPiSerial.py
│   │   └── vecadd
│   │       ├── Makefile
│   │       ├── vecadd_gpu.cu
│   │       ├── vecadd_mpi_cpu
│   │       ├── vecadd_mpi_cpu.c
│   │       ├── vecaddmpi_cpu.sh
│   │       └── vecadd_mpi_gpu.c
│   ├── data
│   │   ├── H2O.gjf
│   │   └── step_sizes.txt
│   └── slurm
│       ├── calc_pi_array.sh
│       ├── calc_pi_mpi.sh
│       ├── calc_pi_parallel.sh
│       ├── calc_pi_serial.sh
│       ├── gaussian.sh
│       ├── hostname_mpi.sh
│       ├── vecadd_hopper.sh
│       ├── vecadd_xena.sh
│       ├── workshop_example2.sh
│       ├── workshop_example3.sh
│       └── workshop_example.sh
└── README.md
```

Run tree to see how the workshops directories are organized…

```
[vanilla@hopper ~]$ tree workshops
workshops/
├── intro_workshop
│   ├── code
│   │   ├── calcPiMPI.py
│   │   ├── calcPiSerial.py
│   │   └── vecadd
│   │       ├── Makefile
│   │       ├── vecadd_gpu.cu
│   │       ├── vecadd_mpi_cpu
│   │       ├── vecadd_mpi_cpu.c
│   │       ├── vecaddmpi_cpu.sh
│   │       └── vecadd_mpi_gpu.c
│   ├── data
│   │   ├── H2O.gjf
│   │   └── step_sizes.txt
│   └── slurm
│       ├── calc_pi_array.sh
│       ├── calc_pi_mpi.sh
│       ├── calc_pi_parallel.sh
│       ├── calc_pi_serial.sh
│       ├── gaussian.sh
│       ├── hostname_mpi.sh
│       ├── vecadd_hopper.sh
│       ├── vecadd_xena.sh
│       ├── workshop_example2.sh
│       ├── workshop_example3.sh
│       └── workshop_example.sh
└── README.md
```

Run tree to see how the workshops directories are organized…

The workshop files are divided into “code”, “slurm”, and “data” directories.

```
[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_example1.sh
sbatch: Account not specified in script or ~/.default_slurm_account,
using latest project
Submitted batch job 5252
[vanilla@hopper intro_workshop]$
```

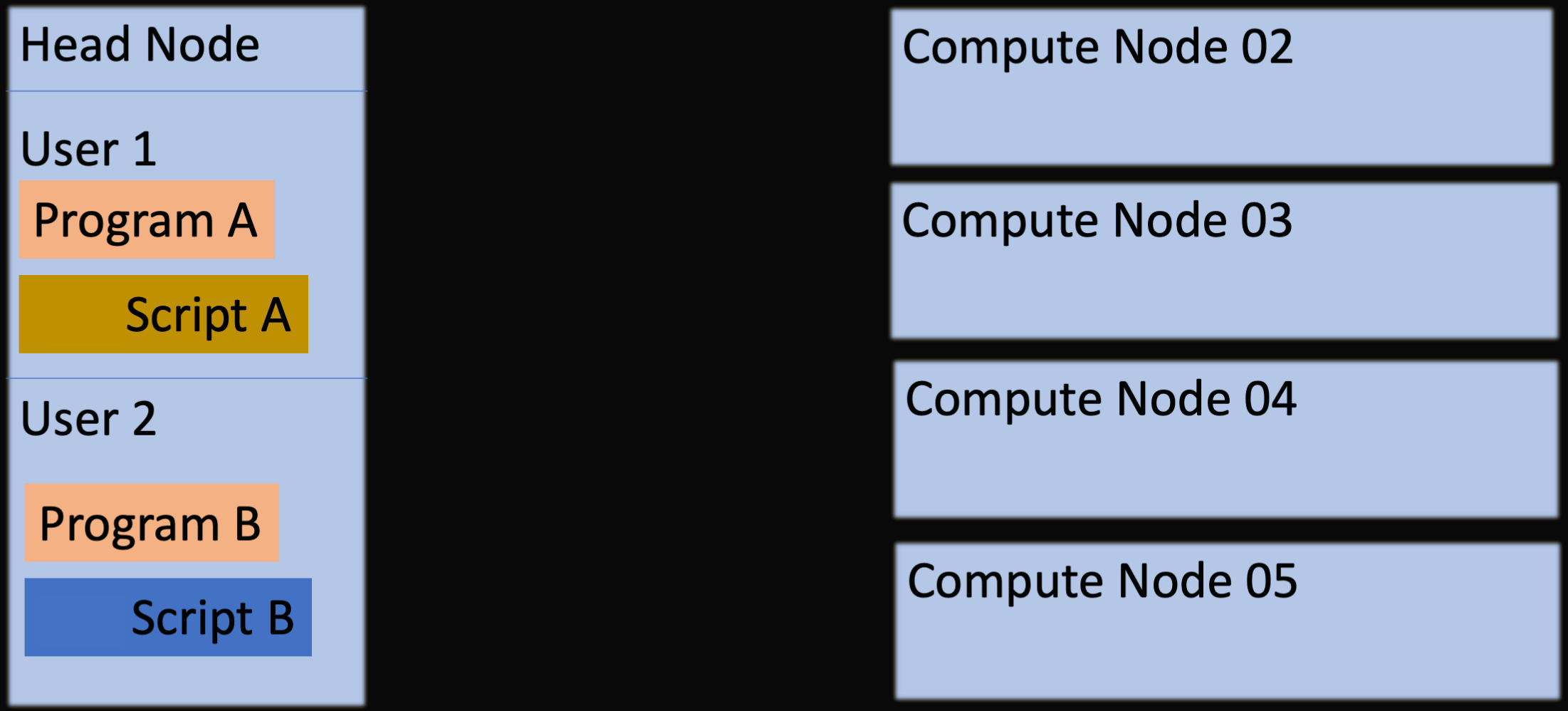
We submit our slurm shell script with the sbatch command.

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

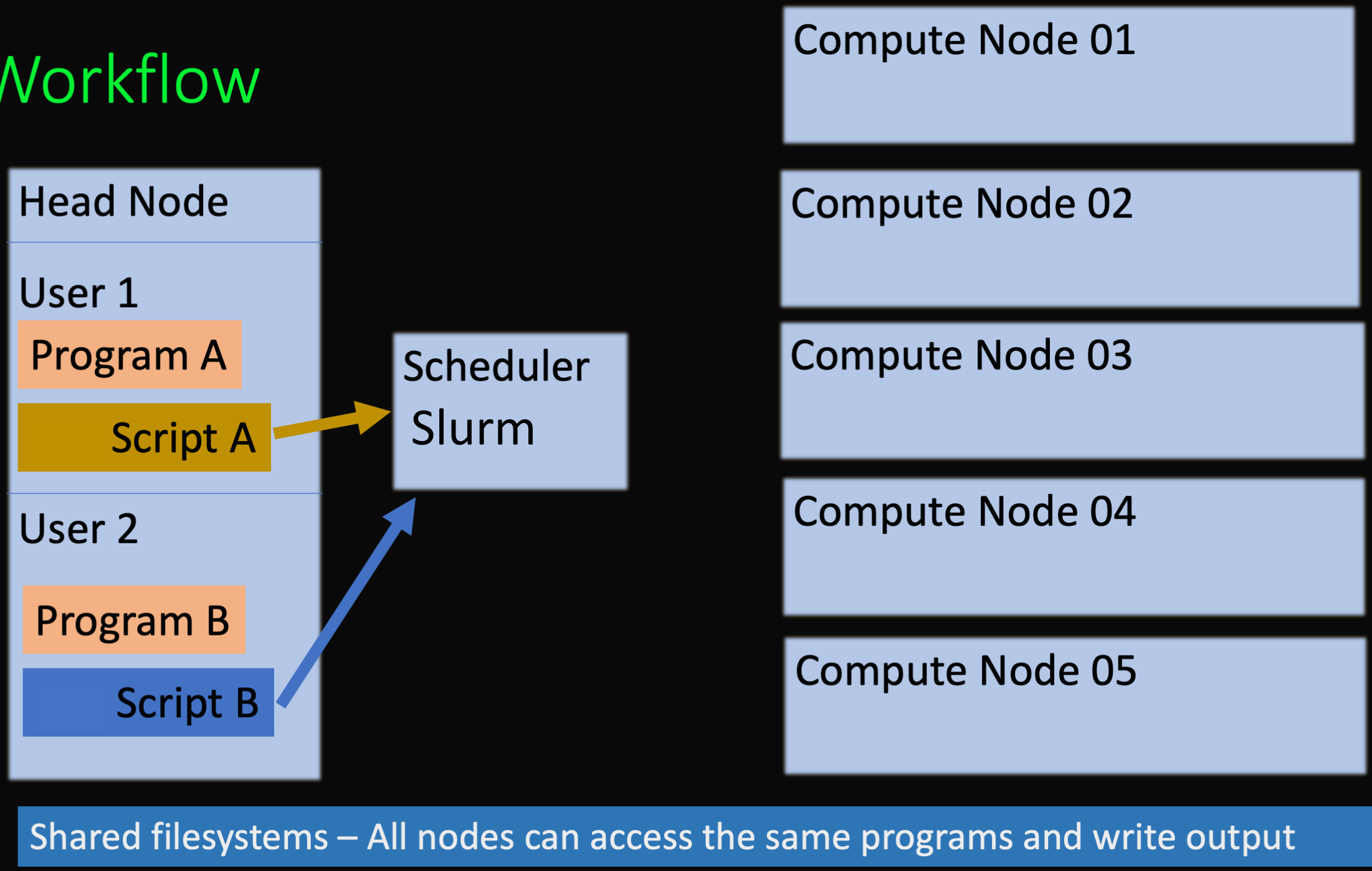
This indicates that the script was successfully sent to the scheduler.

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

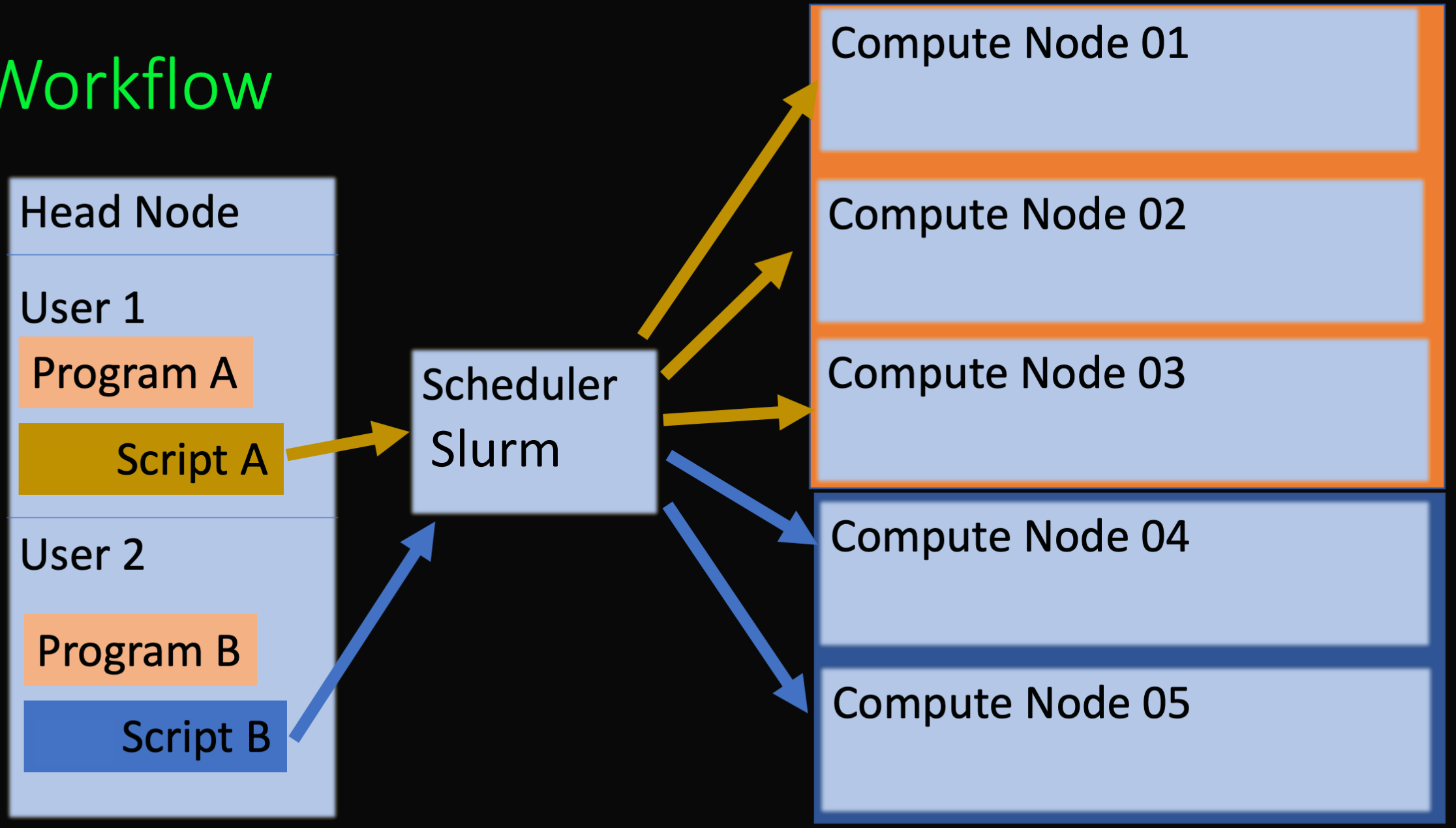
Workflow


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

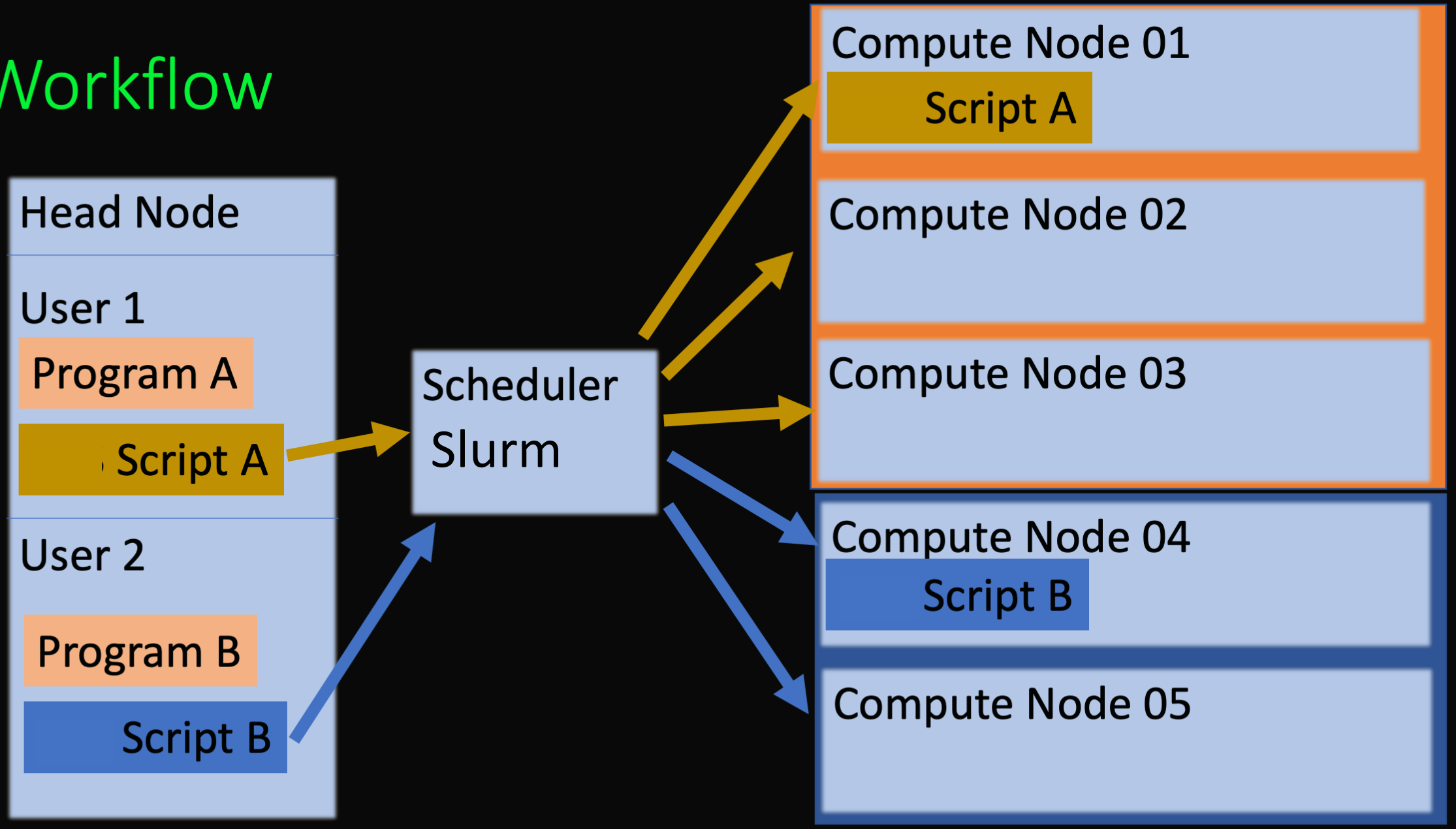

Head Node
User 1
Program A
Script A
User 2
Program B
Script B
Scheduler
Slurm
Compute Node 01
Compute Node 02
Compute Node 03
Compute Node 04
Compute Node 05
Shared filesystems – All nodes can access the same programs and write output

Workflow



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

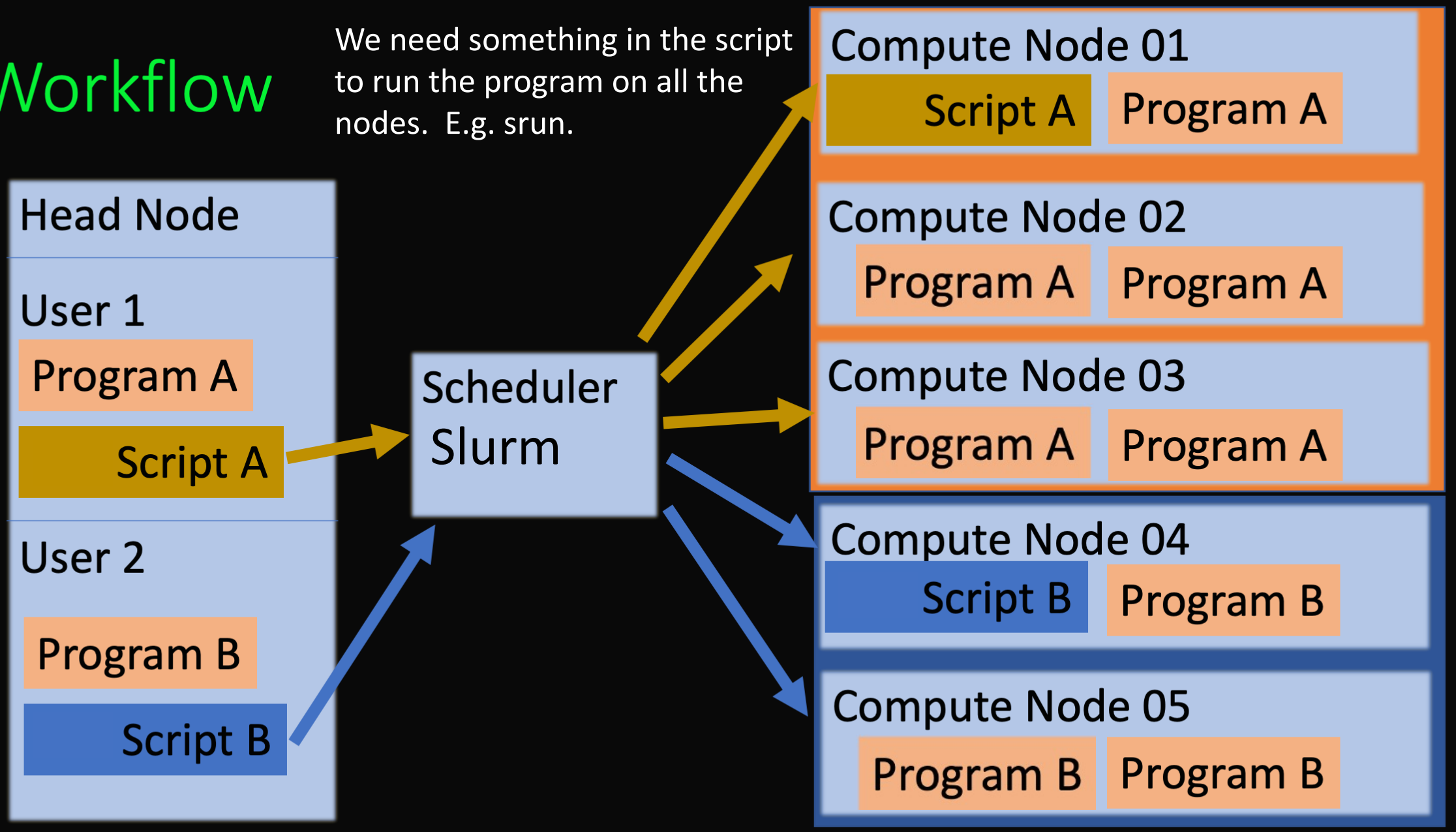
Workflow



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.

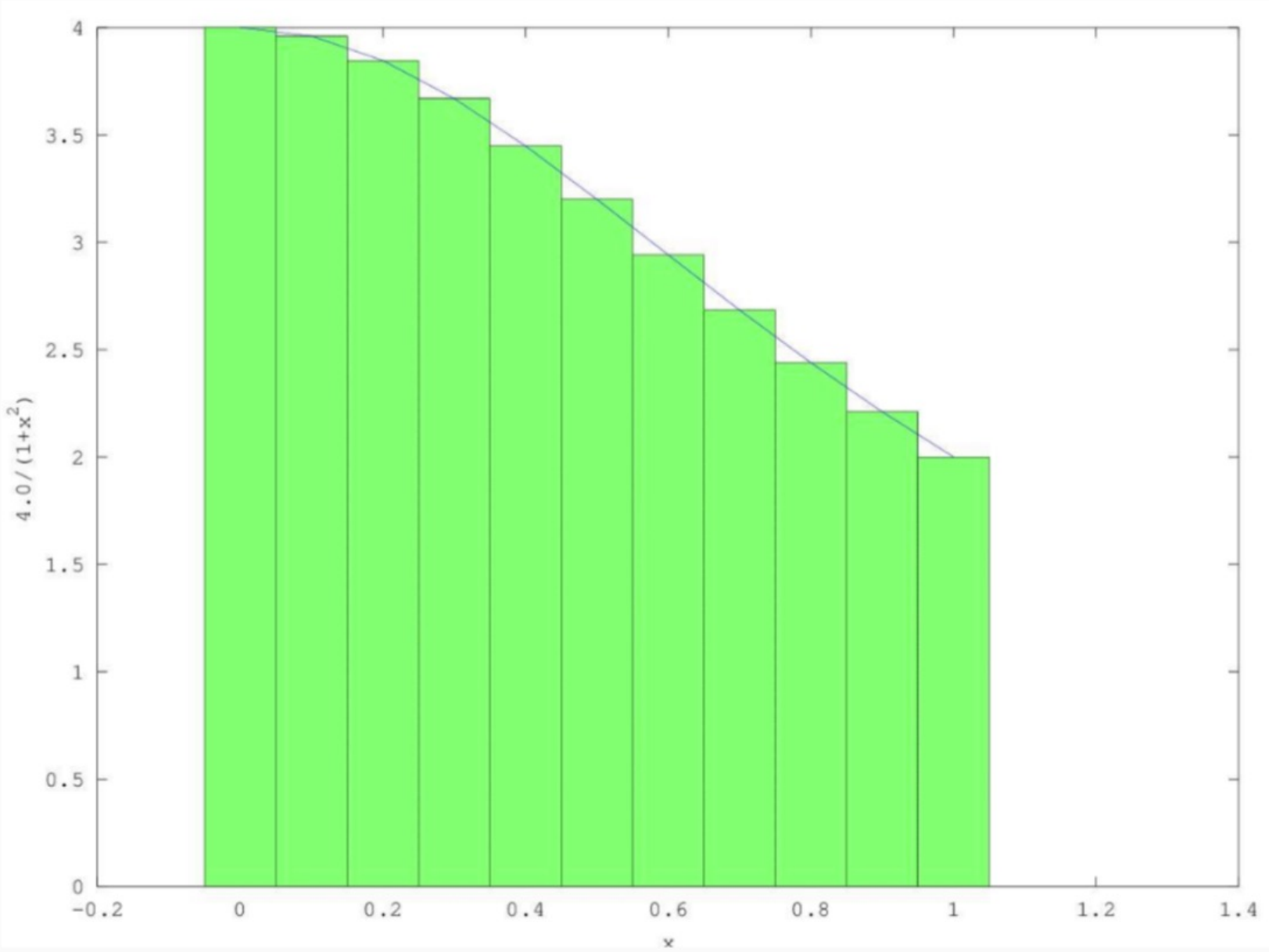
```
[vanilla@hopper intro_workshop]$ cat slurm-5252.out
hopper011
hopper011
hopper011
hopper011
```

```
[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
What do you see?
```

Serial Program to Calculate π



$$\int \frac{4}{1+x^2} = \pi$$

$$\Sigma_{i=1}^{N} \frac{4}{1+\left(\frac{i+\frac{1}{2}}{N}\right)^2} \approx \pi$$

```
[vanilla@hopper intro_workshop]$ module load miniconda3
[vanilla@hopper intro_workshop]$ conda create -n numpy numpy
```

Wait a while – introduce yourselves to your neighbor…

Conda allows you to install software into your home directory. In this case we need the numerical python libraries for calcPiSerial.py

Let's experiment with a program that does slightly more than print the hostname.

```
[vanilla@hopper intro_workshop]$ source activate numpy
[vanilla@hopper intro_workshop]$ srun --partition debug python
code/calcPiSerial.py 10
srun: Using account 2016199 from ~/.default_slurm_account
You have not been allocated GPUs. To request GPUs, use the -G
option in your submission script.
Pi = 3.142425985001098709 40, (Diff=0.0008333314113055934 1)
(calculated in 0.000005 secs with 10 steps)
```

Activate the numpy environment and
Run calcPiSerial.py on a compute node.

For our example program the more steps it takes the more accurate it is, but the longer it takes.

```
[vanilla@hopper intro_workshop]$ sbatch slurm/calc_pi_serial.sh
 sbatch: Using account 2016199 from ~/.default_slurm_account
Submitted batch job 5263
vanilla@hopper:~/workshops/intro_workshop$ squeue --me
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
5263     debug calc_pi_  vanilla  R       0:44      1 hopper011
```

Edit slurm/calc_pi_serial.sh.
Change the email address to your address and submit the script.

Then enter squeue --me to see the job status.

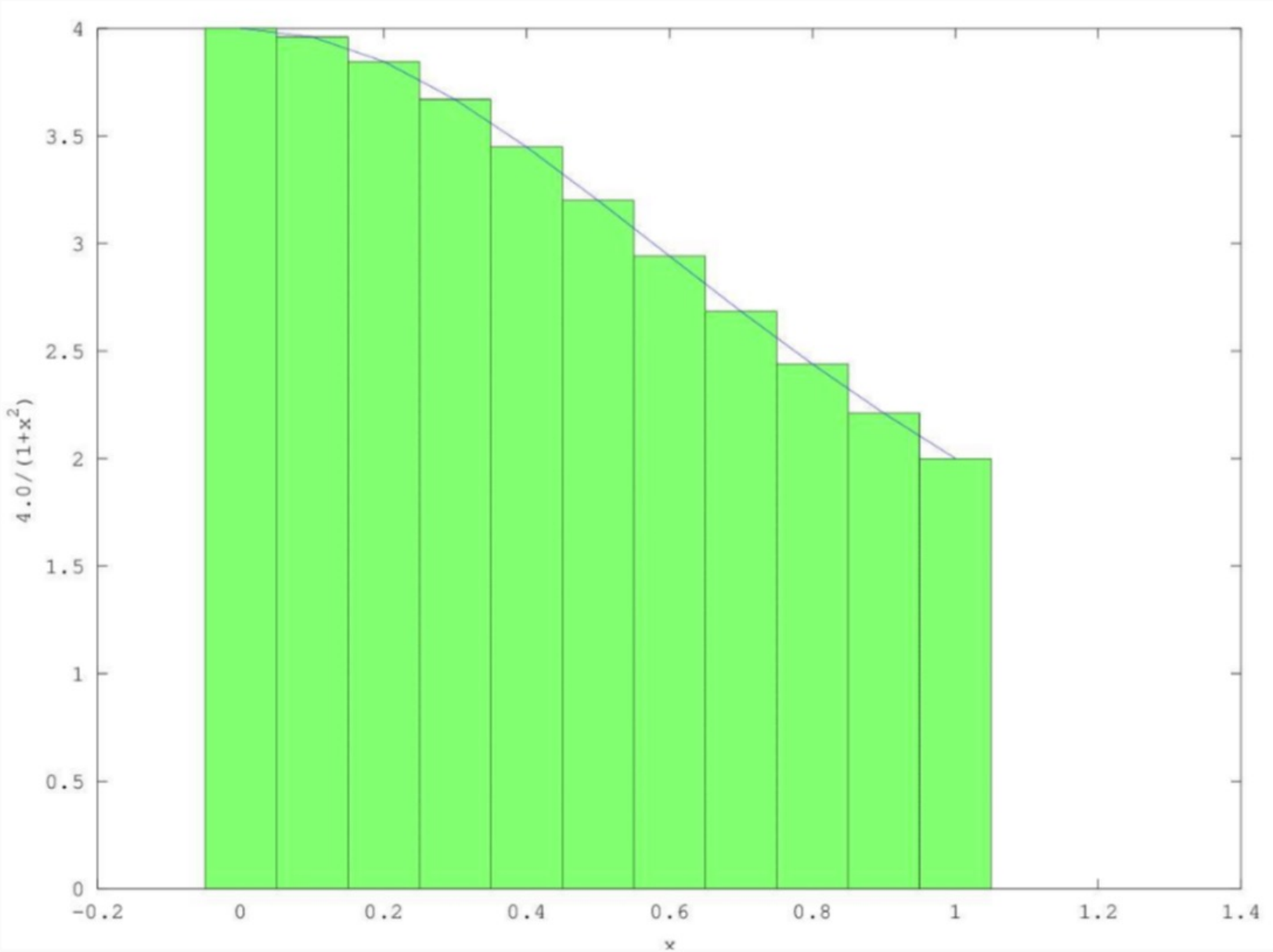
Take a look at the job output.

Parallelism – Coupled Parallelism

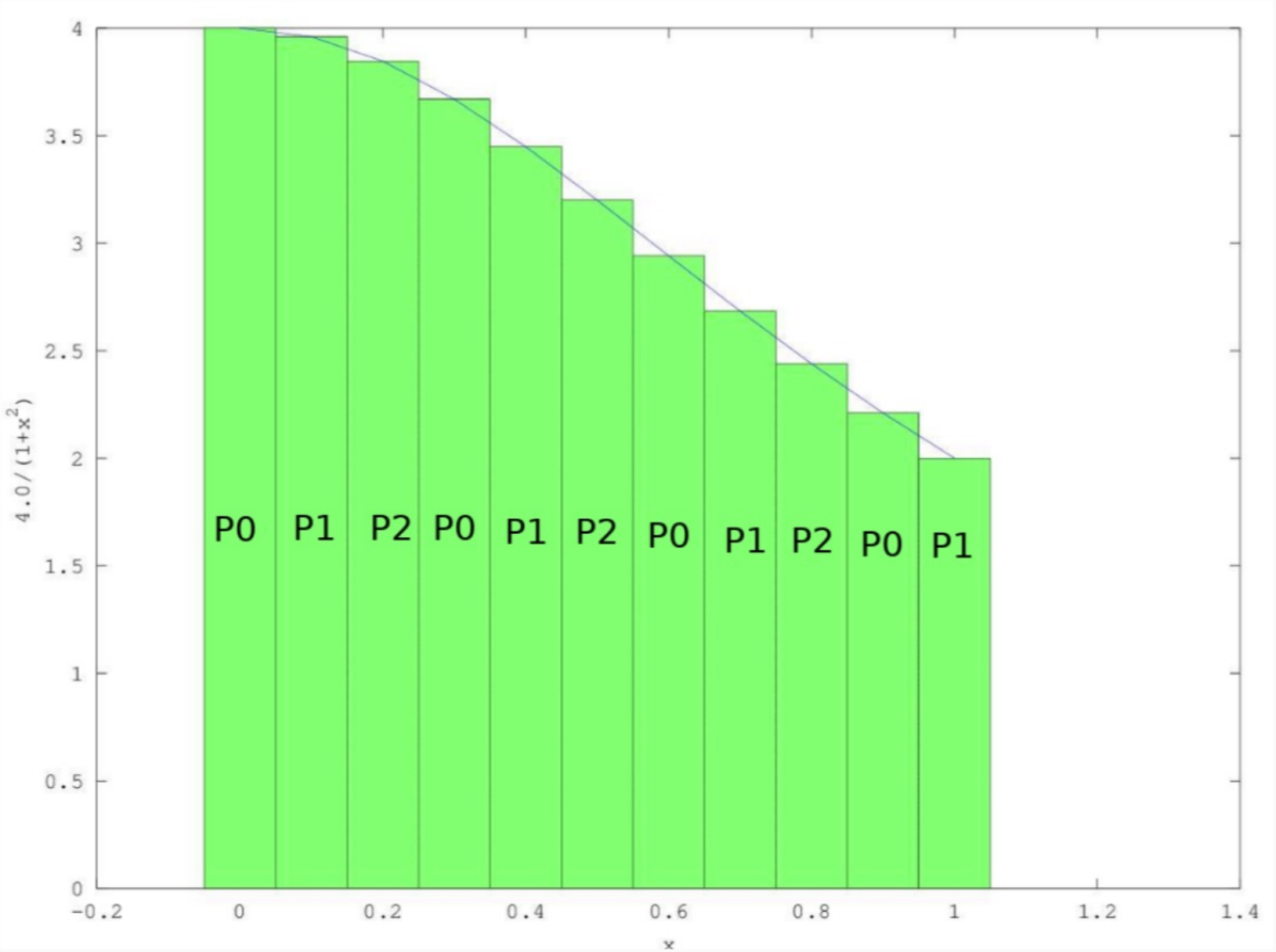
- Coupled problems are those where the CPUs need to work together to solve a problem by communicating with each other.
- Many commercial and research programs designed to run on HPC systems like CARC use a library called the message passing interface (MPI) to do this.

- We have written an MPI version of our python pi calculator to demonstrate.

Serial Program to Calculate π



Parallel Program to Calculate π



MPI: Message Passing Interface

When programs need to run on many processors but also communicate with one another.

Here the parallel version of calcPi needs to communicate the partial sums computed by each process so they can all be added up.
To communicate we will use the MPI library.

Make sure you are in the ~/workshops/intro_workshops directory.
Install your mpi conda environment from file:

```
conda env create –f conda_envs/mpi_numpy.yml
```

```
#!/bin/bash
#SBATCH --partition debug
#SBATCH --nodes 2
#SBATCH --ntasks-per-node 4
#SBATCH --time 00:05:00
#SBATCH --job-name calc_pi_mpi
#SBATCH --mail-user your_username@unm.edu
#SBATCH --mail-type ALL

module load miniconda3
source activate mpi_numpy

cd $SLURM_SUBMIT_DIR
srun --mpi=pmi2 python code/calcPiMPI.py 1000000000
```

sbatch slurm/calc_pi_mpi.sh

```
srun --mpi=pmi2 python code/calcPiMPI.py 1000000000
```

srun understands MPI programs!

If you ever used mpirun or mpiexec you had to provide a lot of parameters to describe how many compute nodes you had and what their names are, etc.

But srun is part of SLURM so it already knows all that.

The only thing you have to specify is the communication library to use. In our case “pmi2”.

Experiment

Run calc_pi_mpi.sh.

Vary the number of tasks it uses.

Use squeue to monitor the state of your job,

Look at your output files.

What is the relationship between the number of tasks and how cast it calculates pi?

Gaussian and ABINIT are both *ab initio* quantum chemistry/physics packages, but they focus on different domains:

- **Gaussian** is mainly used in **quantum chemistry** for molecules
  - emphasises molecular orbitals, reaction energies, spectroscopy, and chemical properties

- **ABINIT** is mainly used in **condensed-matter physics and materials science**
  - focuses on periodic solids, crystals, band structures, phonons, and material properties.

ABINIT is an open-source quantum chemistry and solid-state physics software package for simulating materials at the electronic structure level using density functional theory (DFT).

Ab initio is Latin for **“from the beginning” i.e. first principles**

```
vanilla@hopper intro_workshop $ ssh easley
vanilla@easley: ~ $ cd workshops/abinit
vanilla@easley: abinit $ pwd
/users/vanilla/workshops/abinit
```

Login to Easley and change directory into the abinit directory

```
vanilla@easley: abinit $ pwd
/users/vanilla/workshops/abinit
vanilla@easley: abinit $ tree
.
├── easley_debug.slurm
├── input
│   ├── O.psp8
│   ├── Si_r.psp8
│   ├── t41.abi
│   ├── t41.abo
│   └── t41.files
└── output
```

Tutorial 41 quartz (SiO_2)

Oxygen Pseudopotential (valence electron model)

Silicon Pseudopotential

Defines the system, parameters, and calculation settings

ABINIT setup with input files (t41.abi, t41.files, t41.abo, Si_r.psp8, O.psp8) in the input/ folder, an output/

cat easley_debug.slurm

```
#!/bin/bash
#SBATCH --partition debug
#SBATCH --time 10:00     # 10 minutes
#SBATCH --nodes 2           # number of compute nodes to use
#SBATCH --ntasks-per-node 2 # number of copies of Abinit (process) to run per node
#SBATCH --cpus-per-task 2 # number of CPUs to allocate to each process
#SBATCH --mail-user yourusername@unm.edu
#SBATCH --mail-type all
#SBATCH --job-name abinit

# the input file to use
export INPUTDIR=$HOME/workshops/abinit/input/
export INPUTFILES=$INPUTDIR/t41.files
export ABI_PSPDIR=$INPUTDIR

echo Using input directory $INPUTDIR
echo Using input file list  $INPUTFILE
echo Looking for pseudopotentials in $ABI_PSPDIR

# Load abinit
module load abinit/10.4.5

# Hybrid (mix MPI process level and OpenMP thread level parallelism)
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Execute abinit using MPI pmi2 comm. libraries
srun --mpi=pmi2 abinit < $INPUTFILES
```

```
#!/bin/bash
#SBATCH --partition debug
#SBATCH --time 10:00     # 10 minutes
#SBATCH --nodes 2           # number of compute nodes to use
#SBATCH --ntasks-per-node 2 # number of copies of Abinit (process) to run per node
#SBATCH --cpus-per-task 2 # number of CPUs to allocate to each process
#SBATCH --mail-user yourusername@unm.edu
#SBATCH --mail-type all
#SBATCH --job-name abinit
```

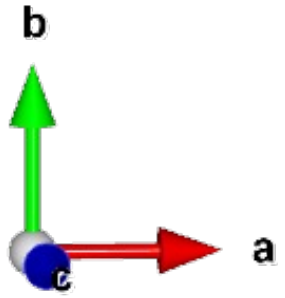
```
# the input file to use
export INPUTDIR=$HOME/workshops/abinit/input/
export INPUTFILES=$INPUTDIR/t41.files
export ABI_PSPDIR=$INPUTDIR

echo Using input directory $INPUTDIR
echo Using input file list  $INPUTFILE
echo Looking for pseudopotentials in $ABI_PSPDIR

# Load abinit
module load abinit/10.4.5

# Hybrid (mix MPI process level and OpenMP thread level
parallelism)
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Execute abinit using MPI pmi2 comm. libraries
srun --mpi=pmi2 abinit < $INPUTFILES
```

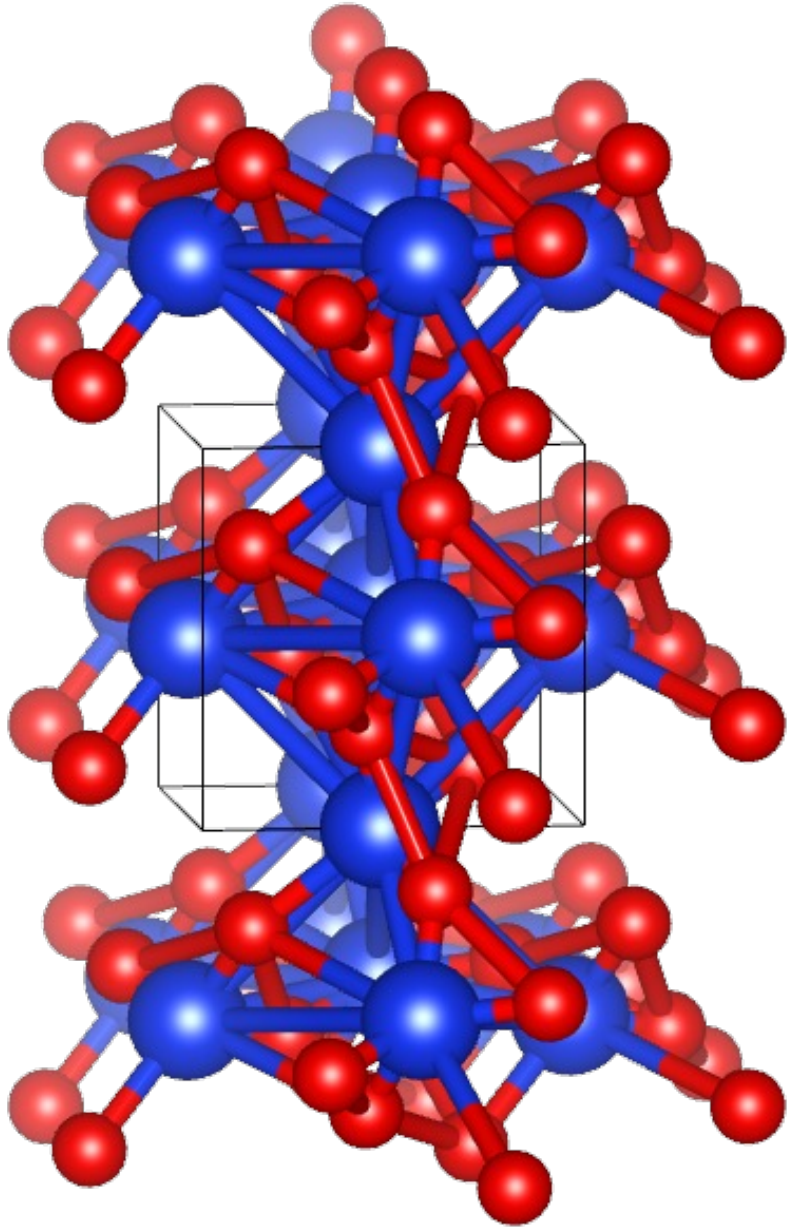
```
vanilla@easley:~/workshops/abinit [master !?]$ sbatch easley_debug.slurm
Submitted batch job 46555
vanilla@easley:~/workshops/abinit [master !?]$ squeue --me
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             46555     debug   abinit  vanilla  R       0:05      2 easley[026-027]
vanilla@easley:~/workshops/abinit [master !?]$ watch squeue --me
```

```
vanilla@easley:~/workshops/abinit [master !?]$ ls
easley_debug.slurm  input  output  slurm-46555.out
vanilla@easley:~/workshops/abinit [master !?]$ cat slurm-46555.out

--- !FinalSummary
program: abinit
version: 10.4.5
start_datetime: Thu Sep 18 16:39:31 2025
end_datetime: Thu Sep 18 16:39:58 2025
overall_cpu_time:          108.2
overall_wall_time:          108.2
exit_requested_by_user: no
timelimit: 0
pseudos:
    Si   : 72a6e885b1ac02cd8530f7649e452e4a
    O    : 733e923716a9eadd6cde8fc80d866281
usepaw: 0
mpi_procs: 4
omp_threads: 2
num_warnings: 0
num_comments: 0
...
```

Crystal structure of α-quartz (SiO_2), showing silicon atoms (blue) and oxygen atoms (red) arranged in a repeating 3D lattice unit cell, rendered with VESTA.

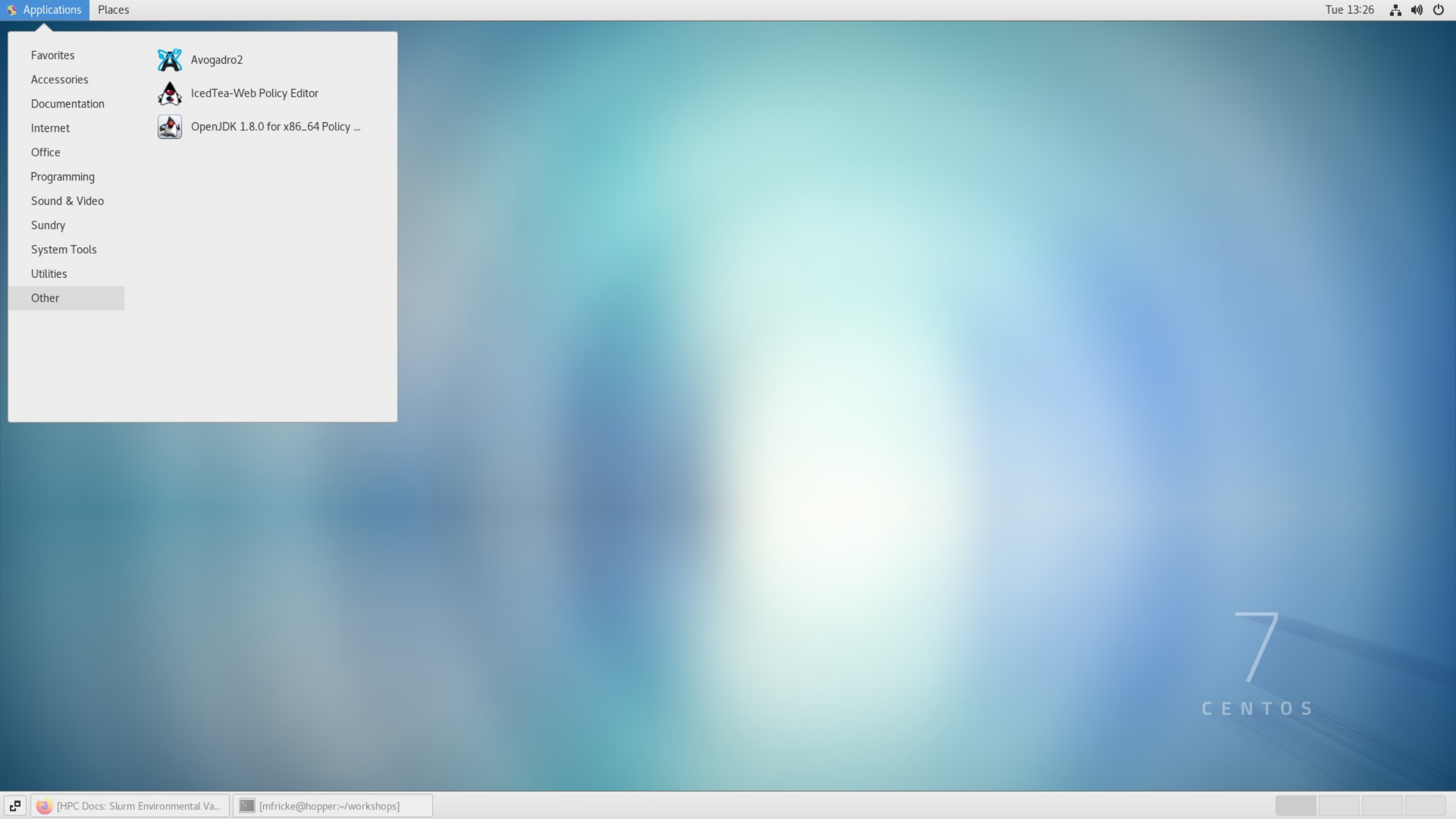
VESTA is a 3D visualization program for structural models, volumetric data such as electron/nuclear densities, and crystal morphologies.





General Purpose Computational Chemistry Package

We will design a molecule using Avogadro and then optimize its geometry with Gaussian


Applications
Places
Tue 13:26
Favorites
Accessories
Documentation
Internet
Office
Programming
Sound & Video
Sundry
System Tools
Utilities
Other
Avogadro2
IcedTea-Web Policy Editor
OpenJDK 1.8.0 for x86_64 Policy ...
7
CENTOS
[HPC Docs: Slurm Environmental Va...
[mfricke@hopper:~/workshops]

Untitled - Avogadro 1.90.0

File Edit View Crystal Extensions Quantum Help

New Open Save Save As Molecule

Draw

Element: Carbon (6)

Bond Order: Automatic

Adjust Hydrogens

Display Types

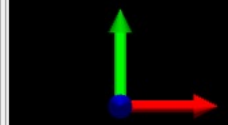
- Crystal Lattice
- Ball and Stick
- Licorice
- Van der Waals
- Wireframe
- Meshes
- Reference Axes Overlay
- Van der Waals (AO)

View Configuration | Display Types

Molecules

Untitled ()

Split Horizontal Split Vertical Close



Untitled* - Avogadro 1.90.0

File Edit View Crystal Extensions Quantum Help

New Open Save Save As Molecule

Draw

Element: Carbon (6)

Bond Order: Automatic

Adjust Hydrogens

Display Types

- [x] Crystal Lattice
- [x] Ball and Stick
- [ ] Licorice
- [ ] Van der Waals
- [ ] Wireframe
- [ ] Meshes
- [x] Reference Axes Overlay
- [ ] Van der Waals (AO)

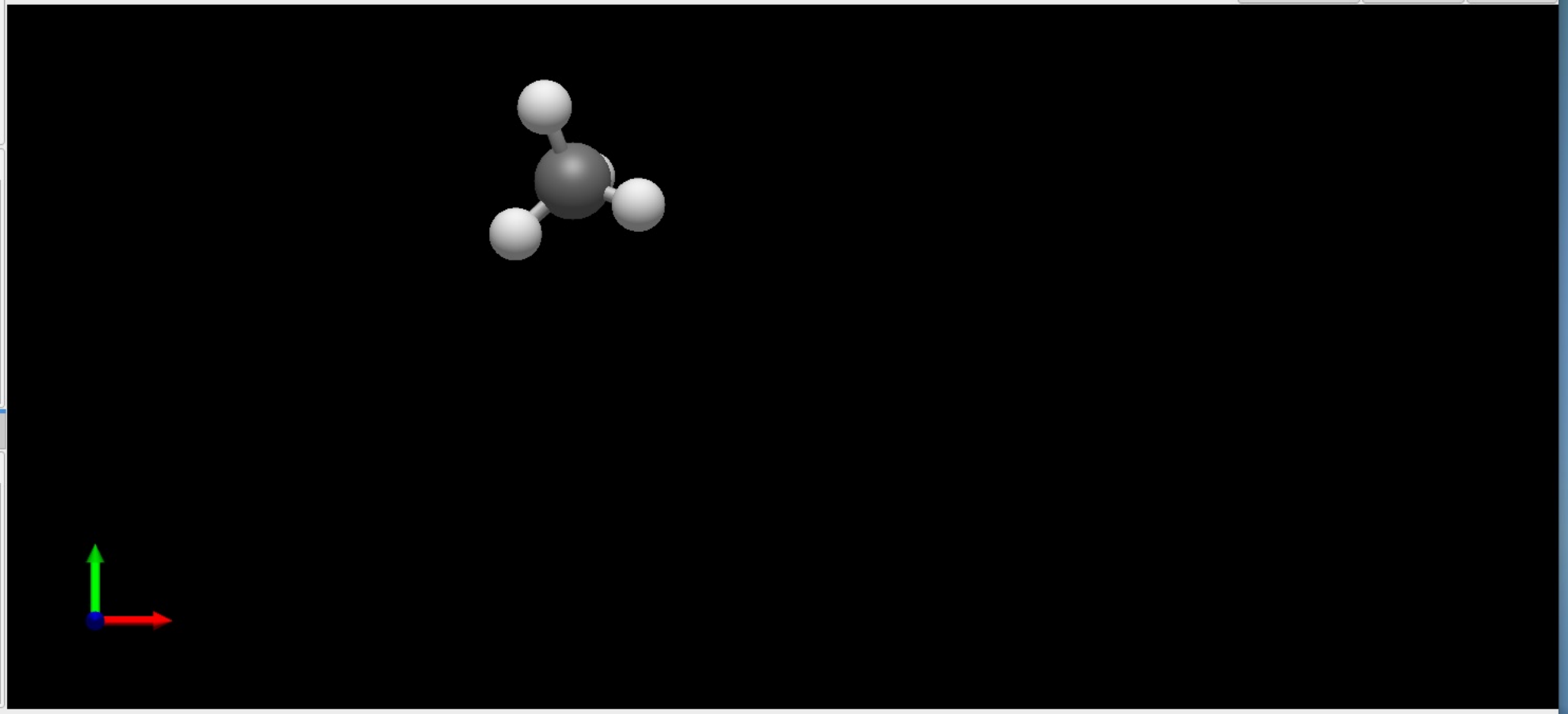
View Configuration | Display Types

Molecules

Untitled (CH4)

*

Split Horizontal Split Vertical Close



Applications Places Avogadro2

Untitled* - Avogadro 1.90.0

File Edit View Crystal Extensions Quantum Help

New Open Save Save As Molecule

Draw

Element: Carbon (6)

Bond Order: Automatic

Adjust Hydrogens

Display Types

- [x] Crystal Lattice
- [x] Ball and Stick
- [ ] Licorice
- [ ] Van der Waals
- [ ] Wireframe
- [ ] Meshes
- [x] Reference Axes Overlay
- [ ] Van der Waals (AO)

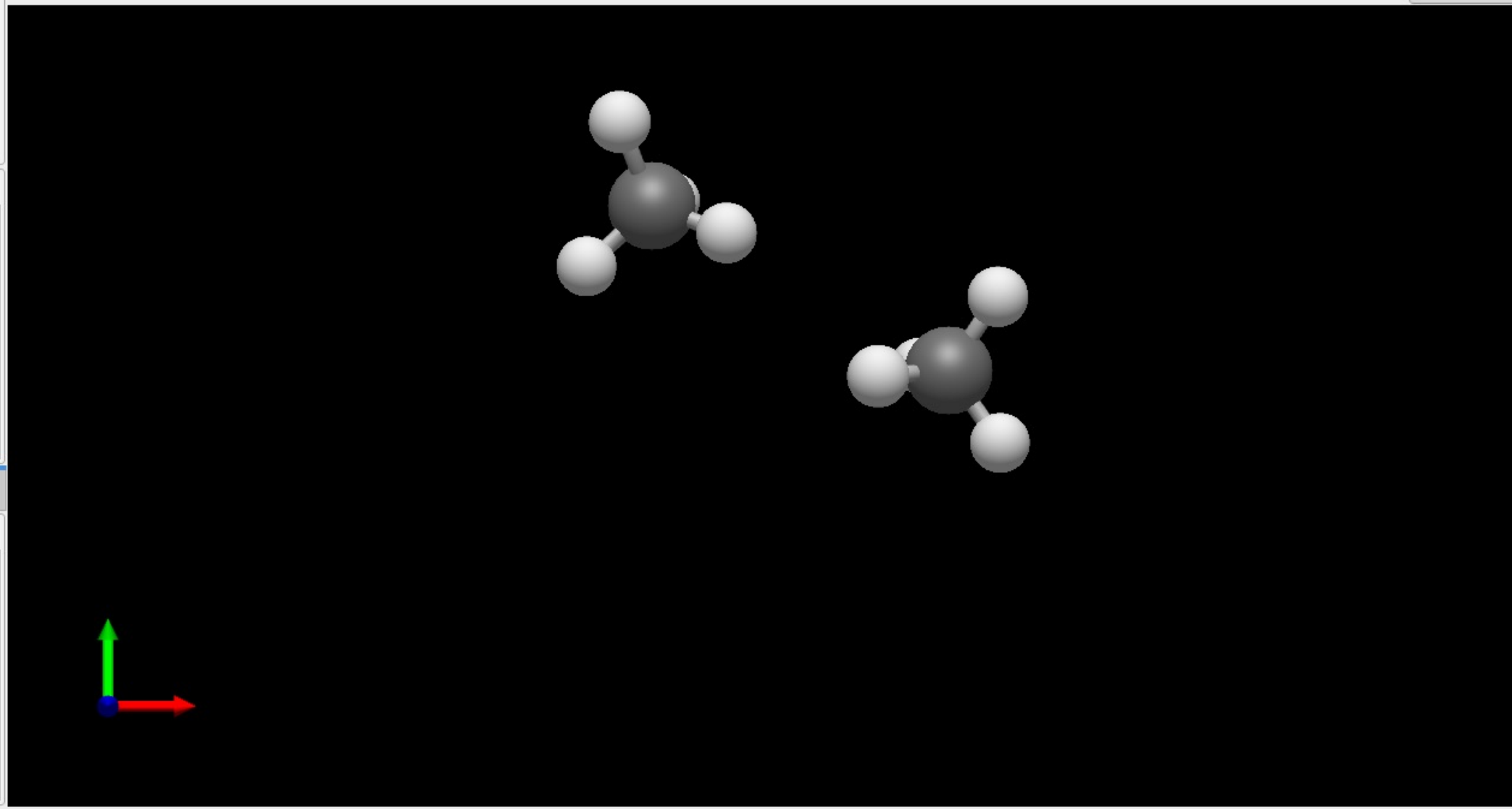
View Configuration | Display Types

Molecules

Untitled (C2H8)

*

Split Horizo



Untitled* - Avogadro 1.90.0

dit View Crystal Extensions Quantum Help

New Open Save Save As Molecule

*

Split Horizon

nent: Carbon (6)

rder: Automatic

Adjust Hydrogens

Types

stal Lattice

and Stick

rice

der Waals

eframe

shes

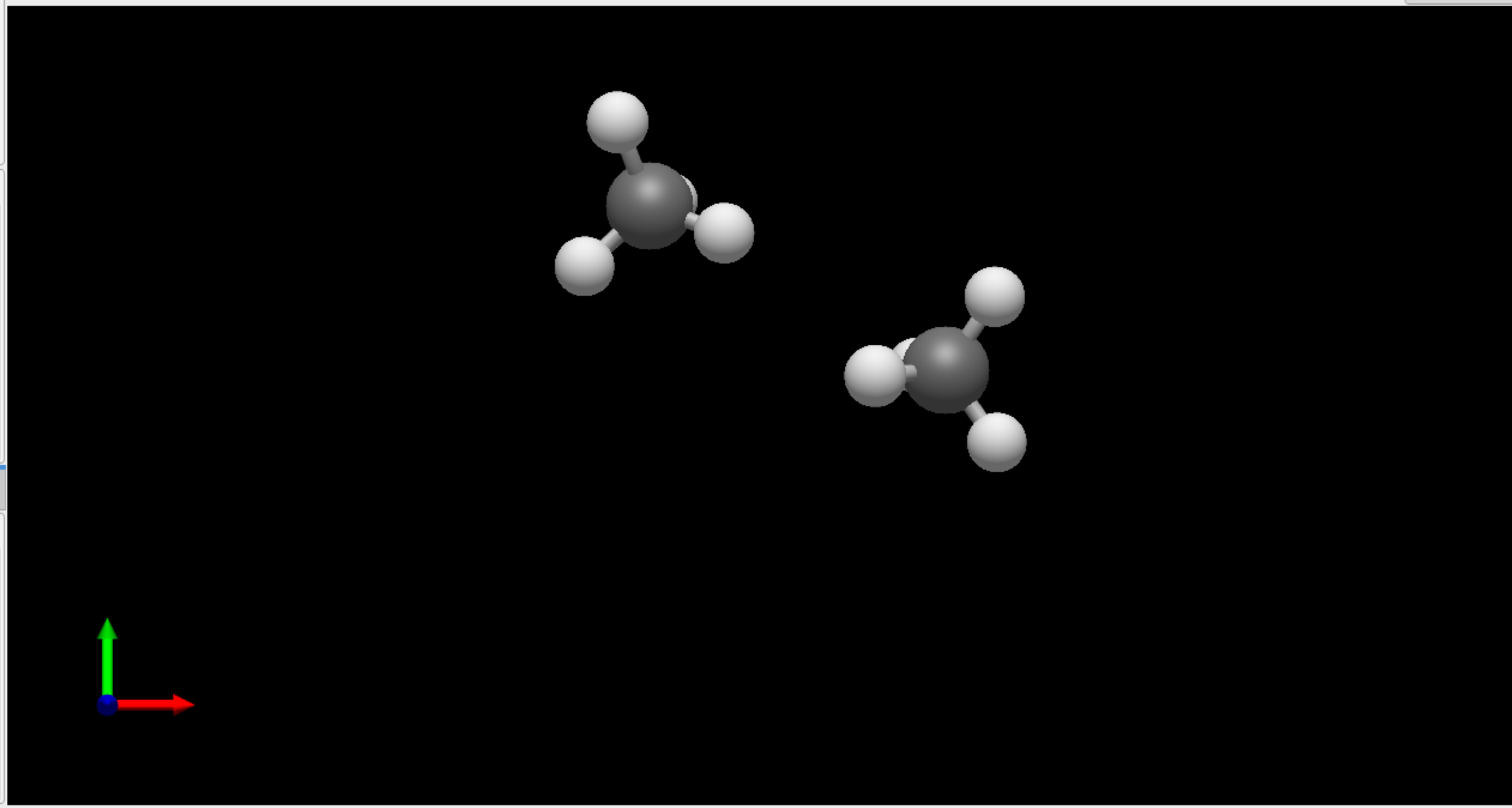
erence Axes Overlay

der Waals (AO)

Configuration Display Types

es

(C2H8)



Untitled* - Avogadro 1.90.0

t View Crystal Extensions Quantum Help

ew Open Save Save As Molecule

*

Split Horizont

ent: Oxygen (8)

der: Automatic

Adjust Hydrogens

ypes

tal Lattice

and Stick

ice

der Waals

frame

nes

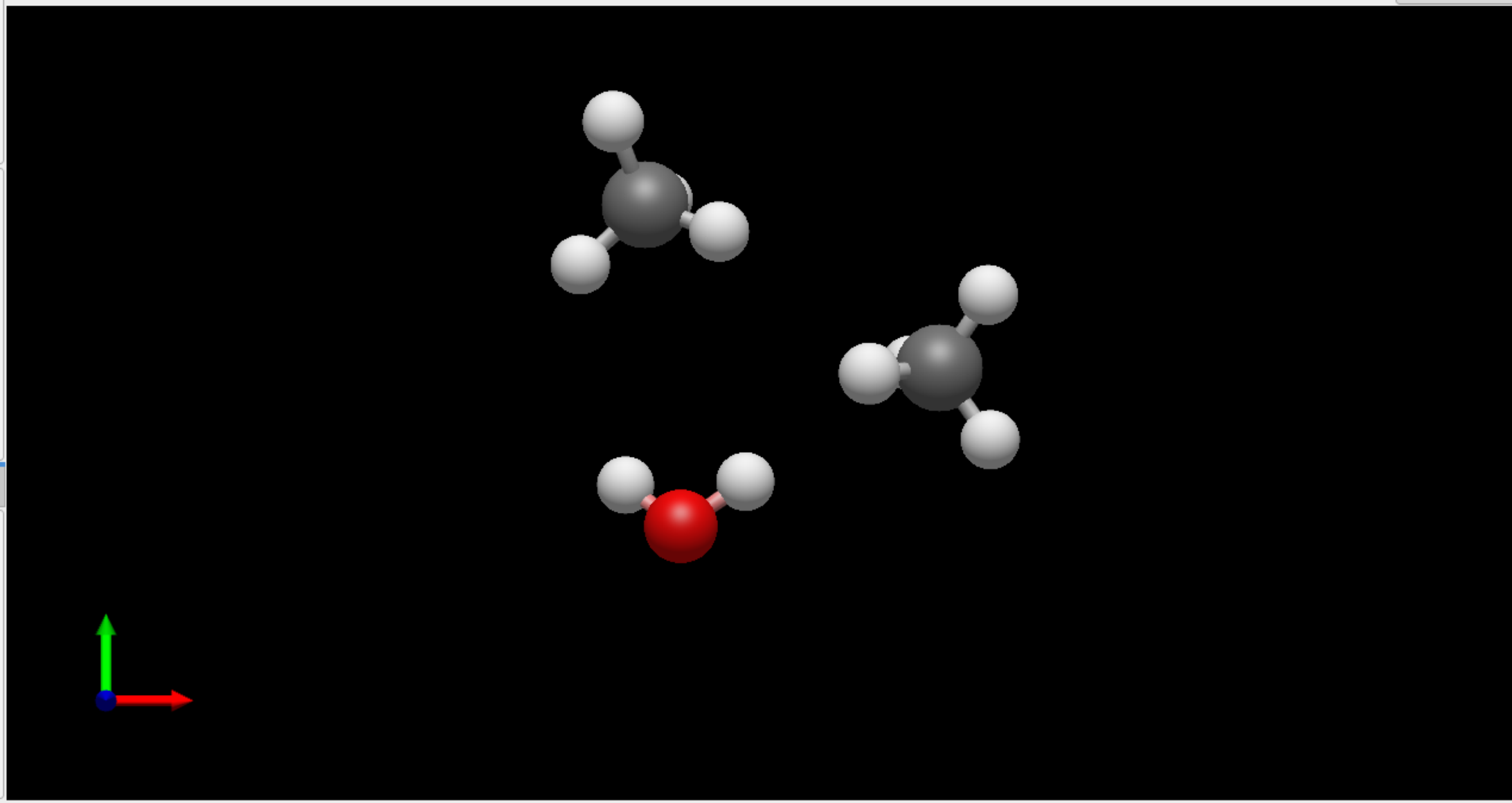
rence Axes Overlay

der Waals (AO)

onfiguration Display Types

s

(C2H10O)



Untitled* - Avogadro 1.90.0

View Crystal Extensions Quantum Help

w Open Save Save As Molecule

ent: Chlorine (17)

ler: Automatic

Adjust Hydrogens

ypes

al Lattice

nd Stick

ce

ler Waals

frame

es

ence Axes Overlay

ler Waals (AO)

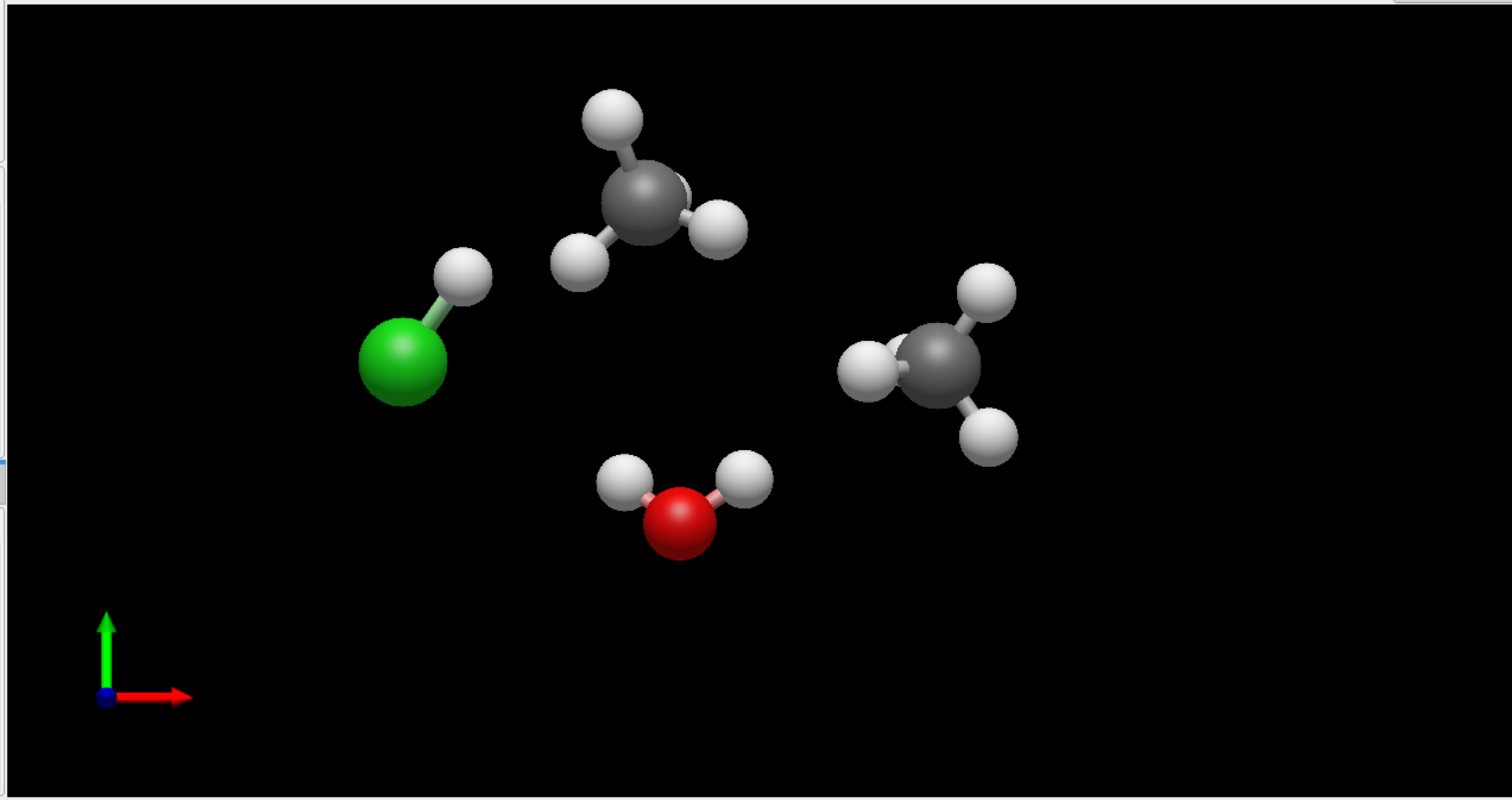
onfiguration Display Types

s

C2H11OCl)

*

Split Horizonta



Untitled* - Avogadro 1.90.0

File Edit View Crystal Extensions Quantum Help

New Open Save Save As Molecule

Draw

Element: Chlorine (17)

Bond Order: Automatic

Adjust Hydrogens

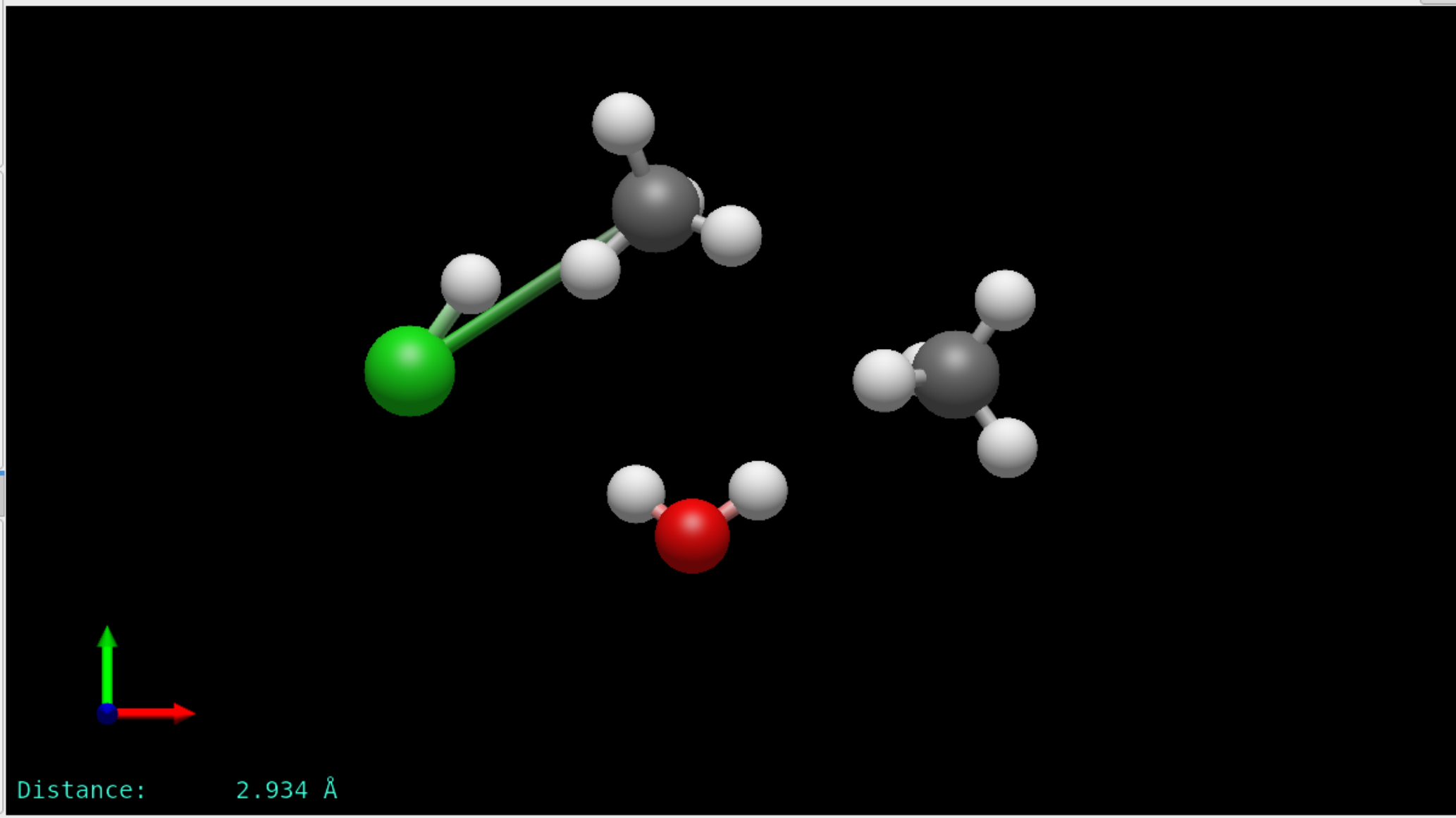
Display Types

- Crystal Lattice
- Ball and Stick
- Licorice
- Van der Waals
- Wireframe
- Meshes
- Reference Axes Overlay
- Van der Waals (AO)

View Configuration | Display Types

Molecules

Untitled (C2H11OCl)

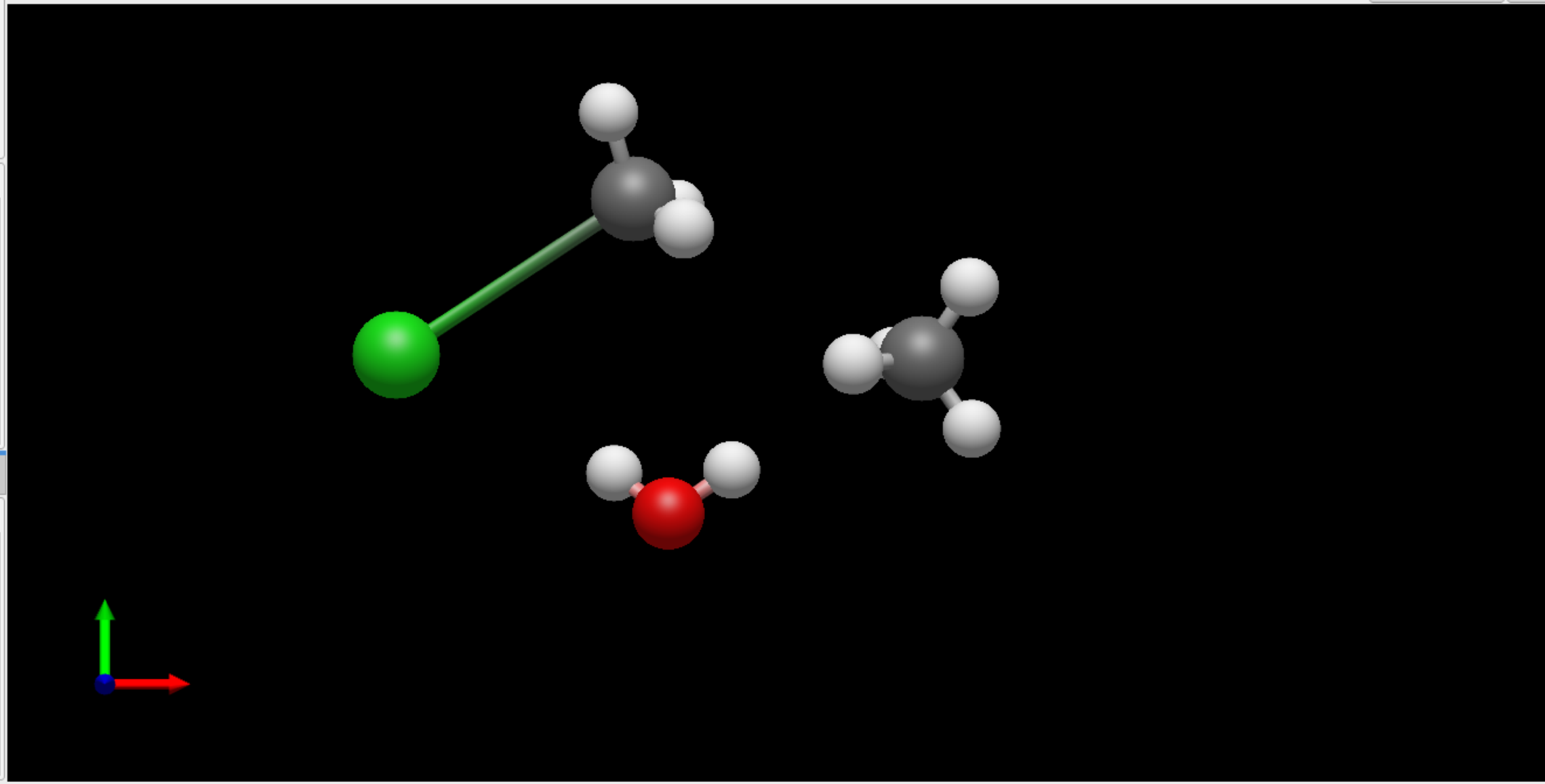


Untitled* - Avogadro 1.90.0

Crystal Extensions Quantum Help

Open Save Save As Molecule

Split Horizontal Spli



Applications Places Avogadro2

Untitled* - Avogadro 1.90.0

File Edit View Crystal Extensions Quantum Help

New Open Save Save As Molecule

Draw

Element: Chlorine (17)

Bond Order: Double

Adjust Hydrogens

Display Types

- [x] Crystal Lattice
- [x] Ball and Stick
- [ ] Licorice
- [ ] Van der Waals
- [ ] Wireframe
- [ ] Meshes
- [x] Reference Axes Overlay
- [ ] Van der Waals (AO)

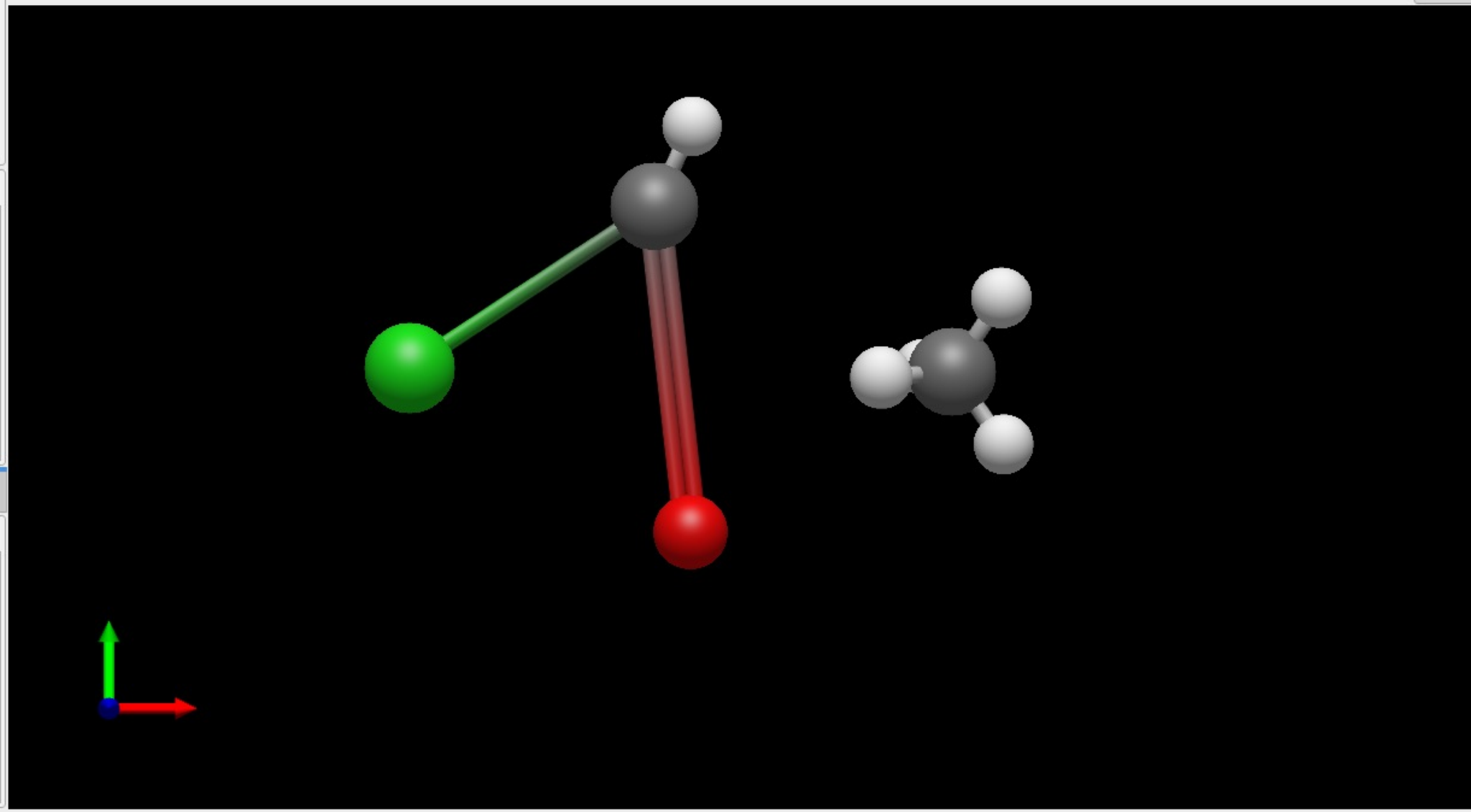
View Configuration | Display Types

Molecules

Untitled (C2H5OCl)

*

Split H



Untitled* - Avogadro 1.90.0

le Edit View Crystal Extensions Quantum Help

New Open Save Save As Molecule

raw

Element: Chlorine (17)

ond Order: Single

Adjust Hydrogens

isplay Types

Crystal Lattice
Ball and Stick
Licorice
Van der Waals
Wireframe
Meshes
Reference Axes Overlay
Van der Waals (AO)

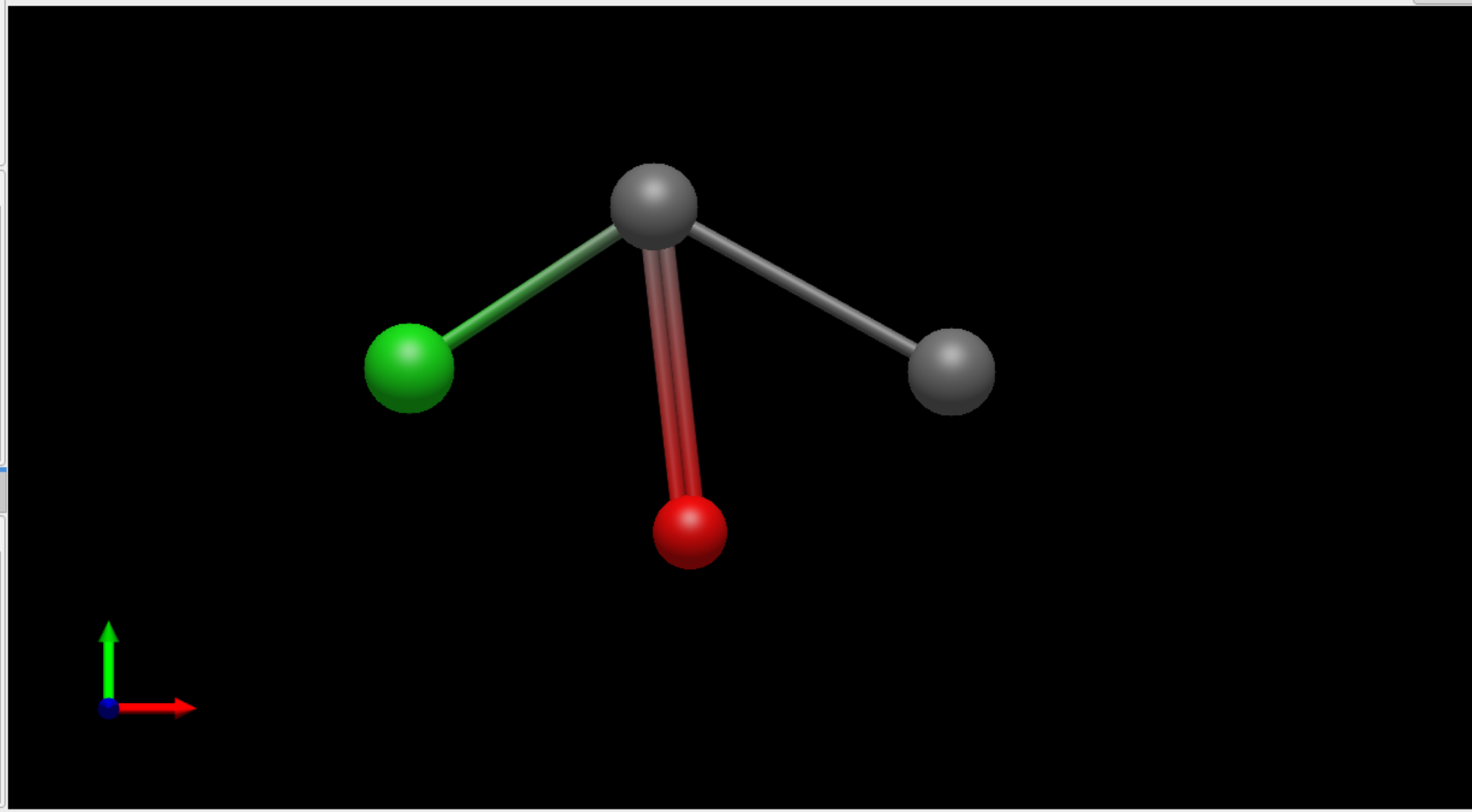
View Configuration | Display Types

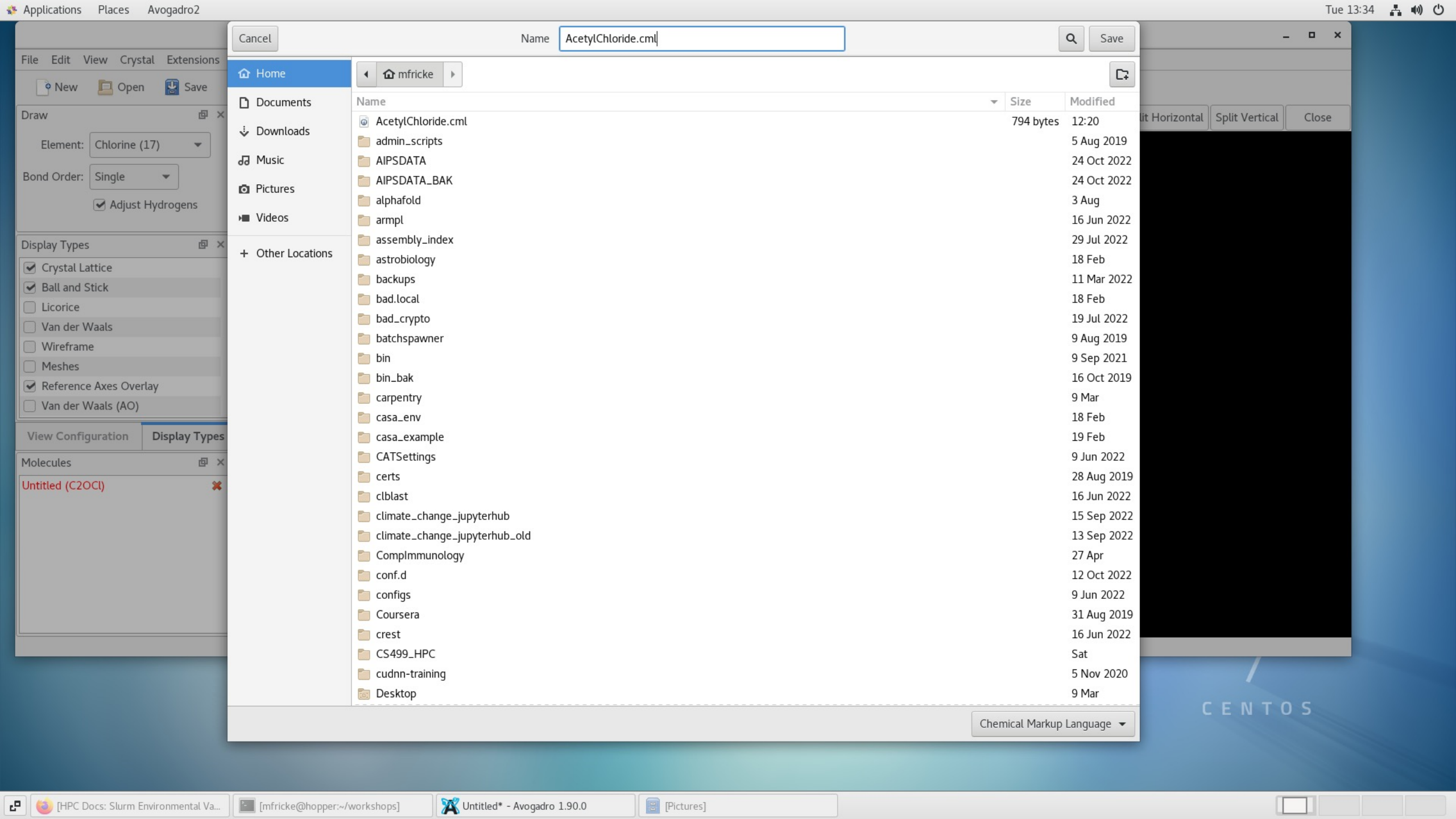
olecules

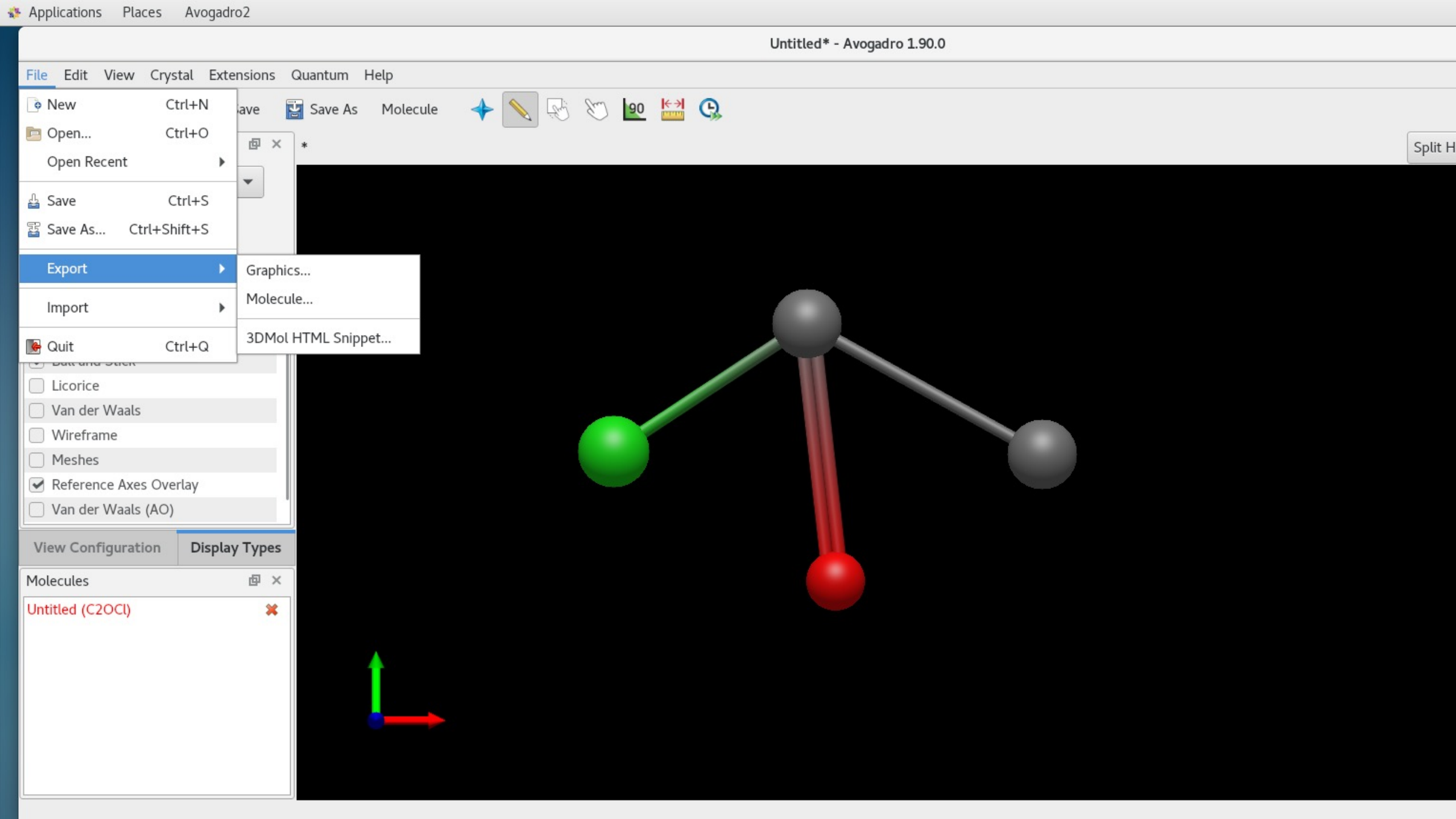
ntitled (C2OCl)

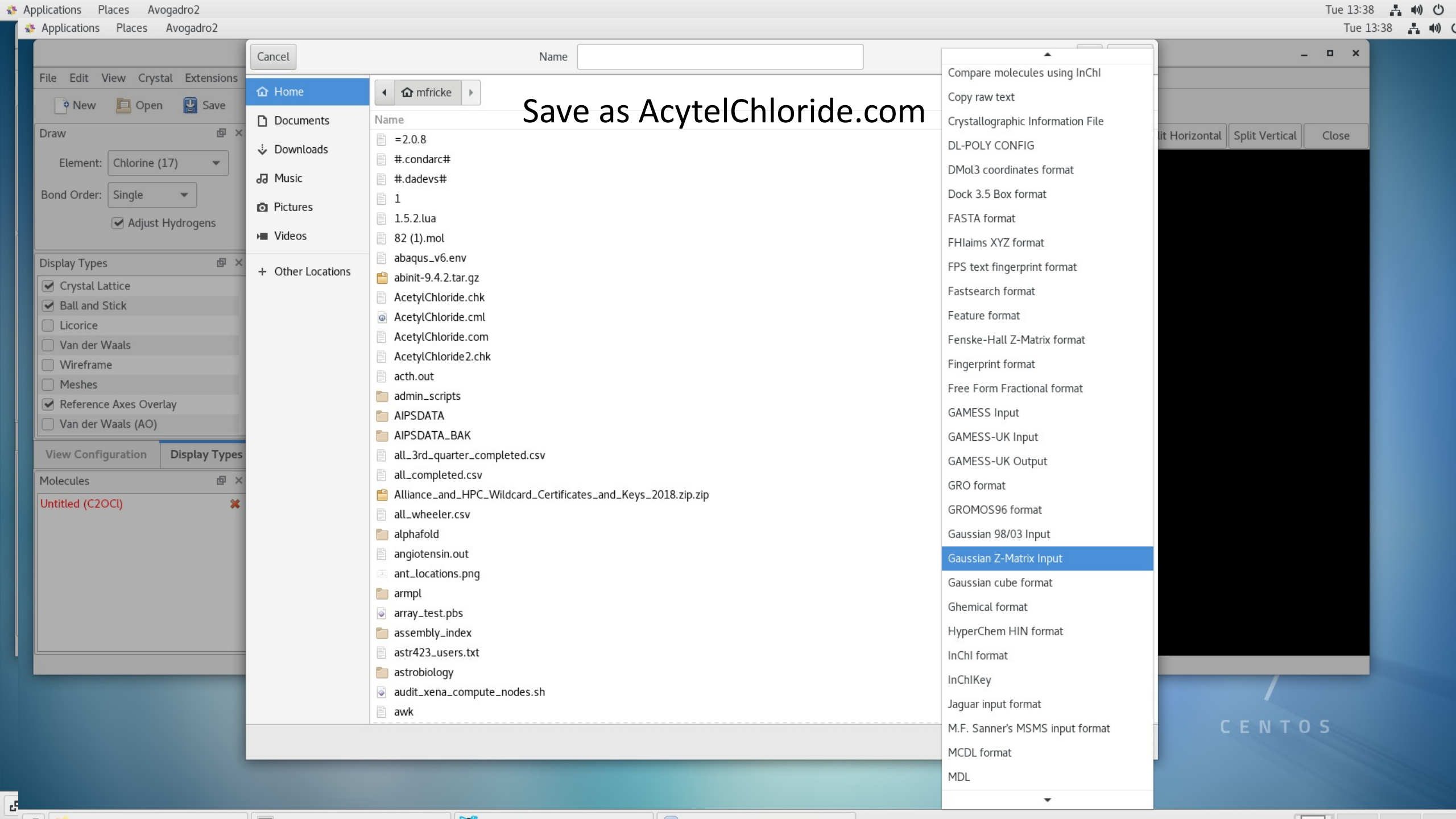
*

Split H




Applications
Places
Avogadro2
Tue 13:34
File
Edit
View
Crystal
Extensions
New
Open
Save
Draw
Element:
Chlorine (17)
Bond Order:
Single
Adjust Hydrogens
Display Types
Crystal Lattice
Ball and Stick
Licorice
Van der Waals
Wireframe
Meshes
Reference Axes Overlay
Van der Waals (AO)
View Configuration
Display Types
Molecules
Untitled (C2OCl)
Split Horizontal
Split Vertical
Close
Cancel
Name
AcetylChloride.cml
Save
Home
Documents
Downloads
Music
Pictures
Videos
Other Locations
mfricke
Name
Size
Modified
AcetylChloride.cml 794 bytes 12:20
admin_scripts 5 Aug 2019
AIPSDATA 24 Oct 2022
AIPSDATA_BAK 24 Oct 2022
alphafold 3 Aug
armpl 16 Jun 2022
assembly_index 29 Jul 2022
astrobiology 18 Feb
backups 11 Mar 2022
bad.local 18 Feb
bad_crypto 19 Jul 2022
batchspawner 9 Aug 2019
bin 9 Sep 2021
bin_bak 16 Oct 2019
carpentry 9 Mar
casa_env 18 Feb
casa_example 19 Feb
CATSettings 9 Jun 2022
certs 28 Aug 2019
clblast 16 Jun 2022
climate_change_jupyterhub 15 Sep 2022
climate_change_jupyterhub_old 13 Sep 2022
ComplImmunology 27 Apr
conf.d 12 Oct 2022
configs 9 Jun 2022
Coursera 31 Aug 2019
crest 16 Jun 2022
CS499_HPC Sat
cudnn-training 5 Nov 2020
Desktop 9 Mar
Chemical Markup Language
CENTOS
[HPC Docs: Slurm Environmental Va...
[mfricke@hopper:~/workshops]
Untitled* - Avogadro 1.90.0
[Pictures]


Applications
Places
Avogadro2
Untitled* - Avogadro 1.90.0
File
Edit
View
Crystal
Extensions
Quantum
Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Save Ctrl+S
Save As... Ctrl+Shift+S
Export
Import
Quit Ctrl+Q
Graphics...
Molecule...
3DMol HTML Snippet...
Save As
Molecule
Split H
Licorice
Van der Waals
Wireframe
Meshes
Reference Axes Overlay
Van der Waals (AO)
View Configuration
Display Types
Molecules
Untitled (C2OCl)


Applications Places Avogadro2
Tue 13:38
Cancel
Name
Save as AcytelChloride.com
File Edit View Crystal Extensions
New Open Save
Draw
Element: Chlorine (17)
Bond Order: Single
Adjust Hydrogens
Display Types
Crystal Lattice
Ball and Stick
Licorice
Van der Waals
Wireframe
Meshes
Reference Axes Overlay
Van der Waals (AO)
View Configuration
Display Types
Molecules
Untitled (C2OCl)
Split Horizontal
Split Vertical
Close
Home
Documents
Downloads
Music
Pictures
Videos
Other Locations
mfricke
Name
=2.0.8
#.condarc#
#.dadevs#
1
1.5.2.lua
82 (1).mol
abaqus_v6.env
abinit-9.4.2.tar.gz
AcetylChloride.chk
AcetylChloride.cml
AcetylChloride.com
AcetylChloride2.chk
acth.out
admin_scripts
AIPSDATA
AIPSDATA_BAK
all_3rd_quarter_completed.csv
all_completed.csv
Alliance_and_HPC_Wildcard_Certificates_and_Keys_2018.zip.zip
all_wheeler.csv
alphafold
angiotensin.out
ant_locations.png
armpl
array_test.pbs
assembly_index
astr423_users.txt
astrobiology
audit_xena_compute_nodes.sh
awk
Compare molecules using InChI
Copy raw text
Crystallographic Information File
DL-POLY CONFIG
DMol3 coordinates format
Dock 3.5 Box format
FASTA format
FHIaims XYZ format
FPS text fingerprint format
Fastsearch format
Feature format
Fenske-Hall Z-Matrix format
Fingerprint format
Free Form Fractional format
GAMESS Input
GAMESS-UK Input
GAMESS-UK Output
GRO format
GROMOS96 format
Gaussian 98/03 Input
Gaussian Z-Matrix Input
Gaussian cube format
Ghemical format
HyperChem HIN format
InChI format
InChIKey
Jaguar input format
M.F. Sanner's MSMS input format
MCDL format
MDL
CENTOS

```
scp AcetylChloride.com vanilla@easley.alliance.unm.edu:~
AcetylChloride.com                    100%  269   114.7KB/s   00:00
```

ssh vanilla@easley.alliance.unm.edu:~

```
--------------------------------------------------------------------------------
  Welcome to Easley

  Be sure to review the "Acceptable Use" guidelines posted on the CARC website.

  For assistance using this system email help@carc.unm.edu.

  Tutorial videos can be accessed through the CARC website: Go to
    http://carc.unm.edu, select the "New Users" menu and then click
      "Introduction to Computing at CARC".

  vanilla@easley:~ $
```

cd workshops/gaussian/
ls -l

```
total 1192
-rw-r--r-- 1 vanilla users     500 Sep 12 13:22 AcetylChloride.com
-rw-r-xr-- 1 vanilla users     795 Sep 12 14:15 gaussian16_linda_acetylchloride.sh
```

cd workshops/gaussian/

ls -l

```
total 1192
-rw-r--r-- 1 vanilla users     500 Sep 12 13:22 AcetylChloride.com
-rw-r-xr-- 1 vanilla users     795 Sep 12 14:15 gaussian16_linda_acetylchloride.sh
```

sbatch gaussian16_linda_acetylchloride.sh

```
sbatch: Using account 2016199 from ~/.default_slurm_account
Submitted batch job 2047647
```

cd workshops/gaussian/

ls -l

```
total 1192
-rw-r--r-- 1 vanilla users     500 Sep 12 13:22 AcetylChloride.com
-rw-r-xr-- 1 vanilla users     795 Sep 12 14:15 gaussian16_linda_acetylchloride.sh
```

cat gaussian16_linda_acetylchloride.sh

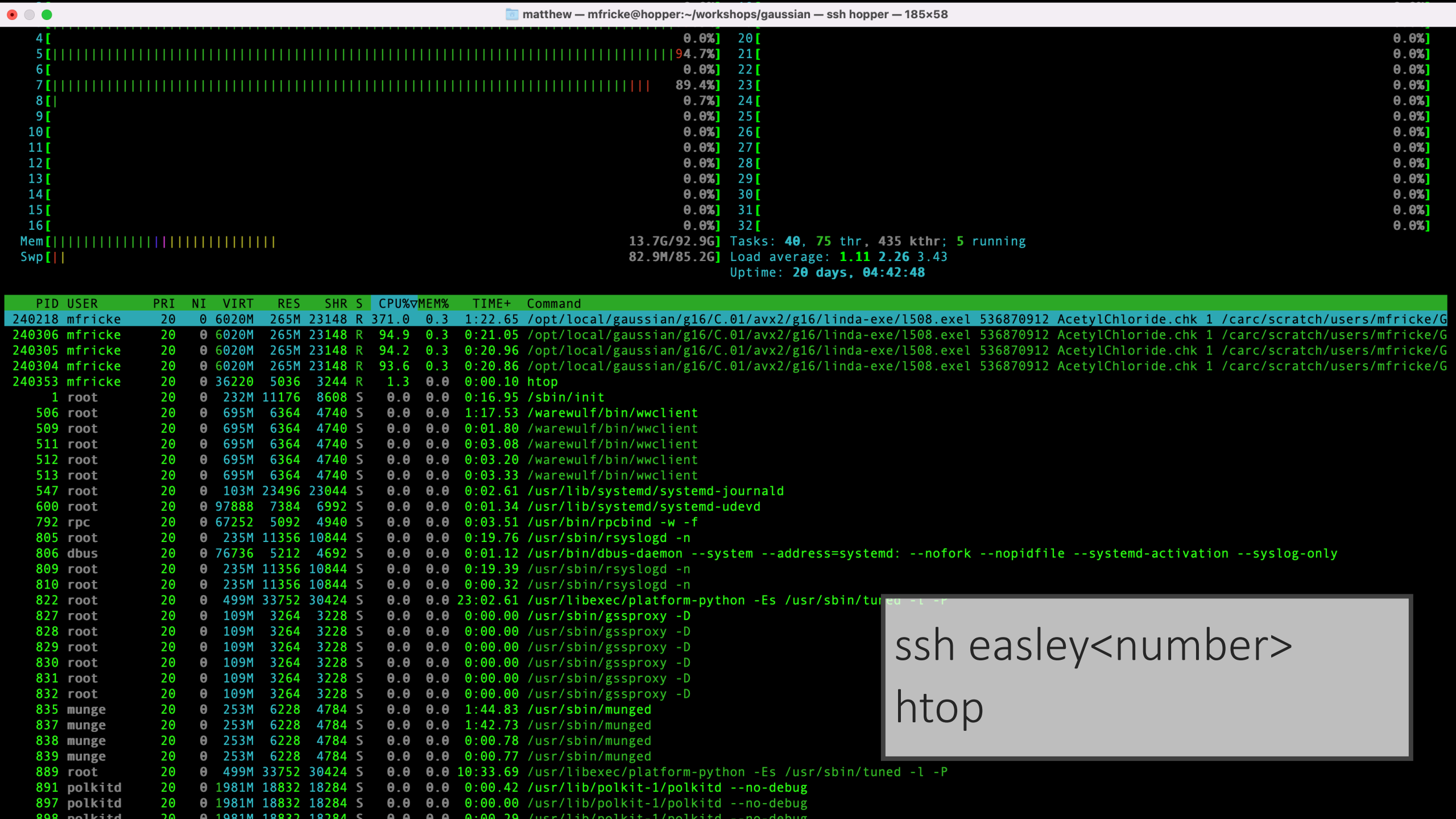
```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=4
#SBATCH --time=00:10:00
#SBATCH --job-name=g16_acetylchloride
#SBATCH --mail-user yourusername@unm.edu

module load gaussian/g16

# To make linda print verbose messages
export GAUSS_LFLAGS="-v"

INPUT_MOLECULE=$SLURM_SUBMIT_DIR/AcetylChloride.com
OUTPUT_FILE=$SLURM_SUBMIT_DIR/AcetylChloride.log

# Reformat the SLURM provided list of compute nodes to a format Gaussian can understand
# i.e. remove duplicates and replace newlines with commas
export GAUSS_WDEF=$(cat $CARC_NODEFILE | uniq | sed -z 's/\n/,/g;s/,$/\n/')

# Tell Linda to use as many nodes as requested by the user
export GAUSS_PDEF=$SLURM_CPUS_PER_TASK
echo "Parallelizing $GAUSS_PDEF processes across $GAUSS_WDEF nodes."

# Set the memory Gaussian variable
export GAUSS_MDEF=4GB

# Run Gaussian
g16 $INPUT_MOLECULE $OUTPUT_FILE
```

Gaussian uses a helper program called Linda to run multiple copies on multiple nodes. Linda does not understand SLURM so we have to spend some time converting SLURM information into a format Linda understand. That's most of this script.

watch squeue --me

```
Every 2.0s: squeue --me                                    hopper: Tue
Sep 12 14:37:13 2023

             JOBID PARTITION     NAME     USER ST TIME  NODES NODELIST(REASON)
           2047652   general g16_acet  vanilla  R 1:05   2 hopper[003-004]
```

Output will be written to AcetylChloride.log and slurm-<jobid>.out


matthew — mfricke@hopper:~/workshops/gaussian — ssh hopper — 185×58
4[0.0%] 20[0.0%]
5[94.7%] 21[0.0%]
6[0.0%] 22[0.0%]
7[89.4%] 23[0.0%]
8[0.7%] 24[0.0%]
9[0.0%] 25[0.0%]
10[0.0%] 26[0.0%]
11[0.0%] 27[0.0%]
12[0.0%] 28[0.0%]
13[0.0%] 29[0.0%]
14[0.0%] 30[0.0%]
15[0.0%] 31[0.0%]
16[0.0%] 32[0.0%]
Mem[13.7G/92.9G] Tasks: 40, 75 thr, 435 kthr; 5 running
Swp[82.9M/85.2G] Load average: 1.11 2.26 3.43
Uptime: 20 days, 04:42:48
PID USER PRI NI VIRT RES SHR S CPU%▽MEM% TIME+ Command
240218 mfricke 20 0 6020M 265M 23148 R 371.0 0.3 1:22.65 /opt/local/gaussian/g16/C.01/avx2/g16/linda-exe/l508.exel 536870912 AcetylChloride.chk 1 /carc/scratch/users/mfricke/G
240306 mfricke 20 0 6020M 265M 23148 R 94.9 0.3 0:21.05 /opt/local/gaussian/g16/C.01/avx2/g16/linda-exe/l508.exel 536870912 AcetylChloride.chk 1 /carc/scratch/users/mfricke/G
240305 mfricke 20 0 6020M 265M 23148 R 94.2 0.3 0:20.96 /opt/local/gaussian/g16/C.01/avx2/g16/linda-exe/l508.exel 536870912 AcetylChloride.chk 1 /carc/scratch/users/mfricke/G
240304 mfricke 20 0 6020M 265M 23148 R 93.6 0.3 0:20.86 /opt/local/gaussian/g16/C.01/avx2/g16/linda-exe/l508.exel 536870912 AcetylChloride.chk 1 /carc/scratch/users/mfricke/G
240353 mfricke 20 0 36220 5036 3244 R 1.3 0.0 0:00.10 htop
1 root 20 0 232M 11176 8608 S 0.0 0.0 0:16.95 /sbin/init
506 root 20 0 695M 6364 4740 S 0.0 0.0 1:17.53 /warewulf/bin/wwclient
509 root 20 0 695M 6364 4740 S 0.0 0.0 0:01.80 /warewulf/bin/wwclient
511 root 20 0 695M 6364 4740 S 0.0 0.0 0:03.08 /warewulf/bin/wwclient
512 root 20 0 695M 6364 4740 S 0.0 0.0 0:03.20 /warewulf/bin/wwclient
513 root 20 0 695M 6364 4740 S 0.0 0.0 0:03.33 /warewulf/bin/wwclient
547 root 20 0 103M 23496 23044 S 0.0 0.0 0:02.61 /usr/lib/systemd/systemd-journald
600 root 20 0 97888 7384 6992 S 0.0 0.0 0:01.34 /usr/lib/systemd/systemd-udevd
792 rpc 20 0 67252 5092 4940 S 0.0 0.0 0:03.51 /usr/bin/rpcbind -w -f
805 root 20 0 235M 11356 10844 S 0.0 0.0 0:19.76 /usr/sbin/rsyslogd -n
806 dbus 20 0 76736 5212 4692 S 0.0 0.0 0:01.12 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
809 root 20 0 235M 11356 10844 S 0.0 0.0 0:19.39 /usr/sbin/rsyslogd -n
810 root 20 0 235M 11356 10844 S 0.0 0.0 0:00.32 /usr/sbin/rsyslogd -n
822 root 20 0 499M 33752 30424 S 0.0 0.0 23:02.61 /usr/libexec/platform-python -Es /usr/sbin/tu
827 root 20 0 109M 3264 3228 S 0.0 0.0 0:00.00 /usr/sbin/gssproxy -D
828 root 20 0 109M 3264 3228 S 0.0 0.0 0:00.00 /usr/sbin/gssproxy -D
829 root 20 0 109M 3264 3228 S 0.0 0.0 0:00.00 /usr/sbin/gssproxy -D
830 root 20 0 109M 3264 3228 S 0.0 0.0 0:00.00 /usr/sbin/gssproxy -D
831 root 20 0 109M 3264 3228 S 0.0 0.0 0:00.00 /usr/sbin/gssproxy -D
832 root 20 0 109M 3264 3228 S 0.0 0.0 0:00.00 /usr/sbin/gssproxy -D
835 munge 20 0 253M 6228 4784 S 0.0 0.0 1:44.83 /usr/sbin/munged
837 munge 20 0 253M 6228 4784 S 0.0 0.0 1:42.73 /usr/sbin/munged
838 munge 20 0 253M 6228 4784 S 0.0 0.0 0:00.78 /usr/sbin/munged
839 munge 20 0 253M 6228 4784 S 0.0 0.0 0:00.77 /usr/sbin/munged
889 root 20 0 499M 33752 30424 S 0.0 0.0 10:33.69 /usr/libexec/platform-python -Es /usr/sbin/tuned -l -P
891 polkitd 20 0 1981M 18832 18284 S 0.0 0.0 0:00.42 /usr/lib/polkit-1/polkitd --no-debug
897 polkitd 20 0 1981M 18832 18284 S 0.0 0.0 0:00.00 /usr/lib/polkit-1/polkitd --no-debug
ssh easley<number>
htop

Useful Slurm Commands

squeue --me --long	shows information about jobs you submitted
squeue --me --start	shows when slurm expects your job to start
scancel jobid	cancels a job
scancel --u $USER	cancels all your jobs
sacct	shows your job history
seff jobid	shows how efficiently the hardware was used