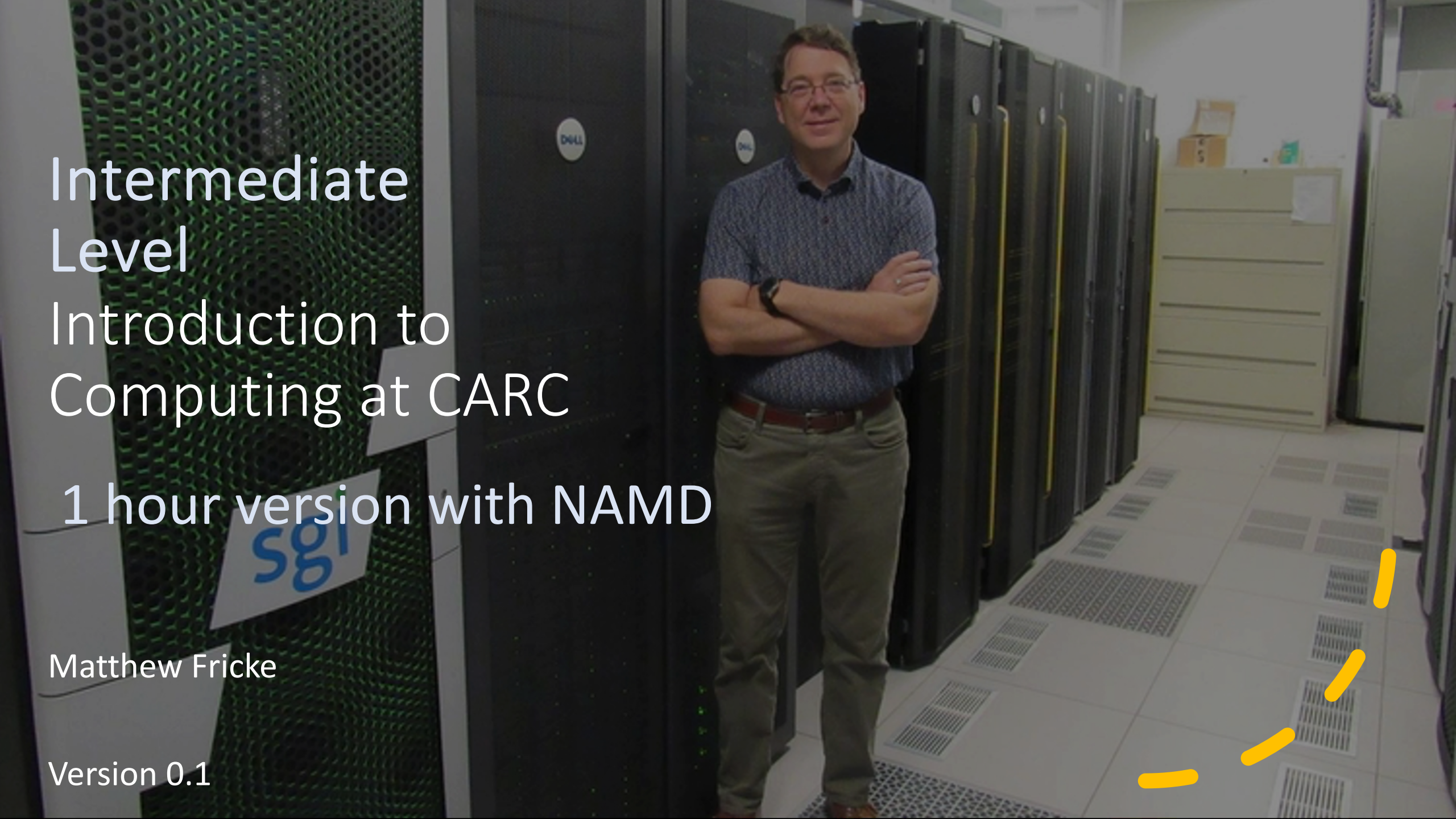


Intermediate Level Introduction to Computing at CARC

1 hour version with NAMD

Matthew Fricke

Version 0.1

Goals

- 1) SLURM scheduler literacy
- 2) Example NAMD

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

Logging into Hopper

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

If you have logged in before use your existing password

Otherwise, your initial password is Welcome2CARC

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.

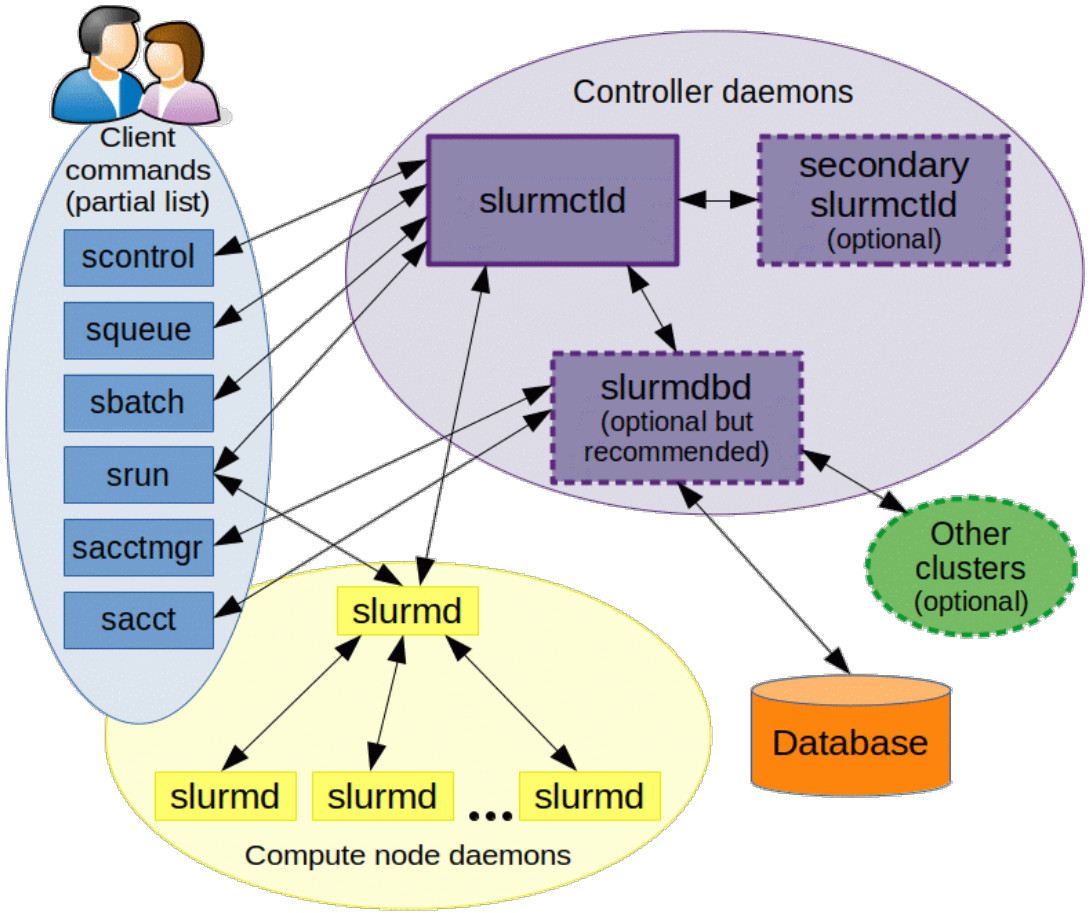
Replace vanilla with your name (unless your last name is Ice)

Logging into Hopper

```
Welcome to Hopper

  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".

  Warning: By default home directories are world readable. Use the chmod command
    to restrict access.

  Don't forget to acknowledge CARC in publications, dissertations, theses and
    presentations that use CARC computational resources:

  "We would like to thank the UNM Center for Advanced Research Computing,
  supported in part by the National Science Foundation, for providing the
  research computing resources used in this work."

  Please send citations to publications@carc.unm.edu.

  There are three types of slurm partitions on Hopper:
  1) General - this partition is accessible by all CARC users.

  2) Condo - preemtable scavenger queue available to all condo users. Your job must use checkpointing to use this
queue or you will lose any work you have done if it is preempted by the partition's owner.

  3) Named partitions - these partitions are available to condo users working under the grant/lab/center that
purchased the associated hardware.

  Type "qgrok" to get the status of the partitions.
--------------------------------------------------------------------------------
Last login: Wed Jul 27 17:46:13 2022 from 129.24.246.68
mfricke@hopper:~ $
```

Simple Linux Utility for Resource Management

slurm
workload manager




ENJOY
Slurm
SODA
IT'S HIGHLY ADDICTIVE!
VOTED #1 SOFT DRINK OF THE 31ST CENTURY!

```
[vanilla@hopper ~]$ qgrok
queues      free   busy   offline  jobs   nodes  CPUs   GPUs   CPUs/node  GPUs/node  Memory/node  time_limit   CPU_limit
-----       -----  -----  -------  -----  -----  -----  -----  ---------  ---------  -----------  -----------  ---------
general     4      6      0        4      10     320    0      32         0          93G          2-00:00:00   64
debug       2      0      0        0      2      64     0      32         0          93G          4:00:00      8
condo       22     25     4        7      51     1632   28     32         2          93G-1.5T     2-00:00:00   192
bugs        2      0      0        0      2      64     0      32         0          93G          7-00:00:00
```

```
vanilla@hopper:~ $ qgrok
queues      free   busy   offline  jobs   nodes  CPUs   GPUs
-----       -----  -----  -----    -----  -----  -----  -----
general     1      9      0        7      10     320    0
debug       2      0      0        0      2      64     0
condo       18     19     1        7      38     1216   8
bugs        0      2      0        0      2      64     0
pcnc        1      1
pathogen    1      0
tc          5      5
gold        2      0
fishgen     0      1
neuro-hsc   8      6
cup-ecs     0      2
tid         0      1
biocomp     0      1
chakra      1      0
pna         0      0
totals:     19     28
```

Open partitions for use by everyone with a CARC account.

Purchased by the Office for the Vice President for Research.

```
vanilla@hopper:~ $ qgrok
queues      free   busy   offline  jobs   nodes  CPUs   GPUs
-----       -----  -----  -----    -----  -----  -----  -----
general     1      9      0        7      10     320    0
debug       2      0      0        0      2      64     0
condo       18     19     1        7      38     1216   8
bugs        0      2      0        0      2      64     0
pcnc        1      1      0        0      2      64     0
pathogen    1      0      0        0      1      32     0
tc          5      5      0        3      10     320    0
gold        2      0      0        0      2      64     0
fishgen     0      1      0        0      1      32     0
neuro-hsc   8      6      0        0      14     448    0
cup-ecs     0      2      0        2      2      64     4
tid         0      1      0        0      1      32     2
biocomp     0      1      0        0      1      32     1
chakra      1      0      0        0      1      32     1
pna         0      0      1        0      1      32     0
totals:     19     28     1        14     48     1536   8
```

```
vanilla@hopper:~ $ qgrok
queues      free   busy   offline  jobs   nodes  CPUs   GPUs
-----       -----  -----  -----    -----  -----  -----  -----
general     1      9      0        7      10     320    0
debug       2      0      0        0      2      64     0
condo       18     19     1        7      38     1216   8
bugs        0      2      0        0      2      64     0
pcnc        1      1
pathogen    1      0
tc          5      5
gold        2      0
fishgen     0      1
neuro-hsc   8      6
cup-ecs     0      2
tid         0      1
biocomp     0      1
chakra      1      0
pna         0      0
totals:     19     28
```

Private partitions

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

```
mfricke@hopper:~ $ qgrok
queues      free   busy   offline  jobs   nodes  CPUs   GPUs
-----       -----  -----  -----    -----  -----  -----  -----
general     1      9      0        7      10     320    0
debug       2      0      0        0      2      64     0
condo       18     19     1        7      38     1216   8
bugs        0      2      0        0      2      64     0
pcnc        1      1
pathogen    1      0
tc          5      5
gold        2      0
fishgen     0      1
neuro-hsc   8      6
cup-ecs     0      2
tid         0      1
biocomp     0      1
chakra      1      0
pna         0      0
totals:     19     28
```

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 ~]$ quotas
Home Directory (/users/vanilla):
quota: Cannot resolve mountpoint path /root/.spack: Permission denied
Disk quotas for user vanilla (uid 659):
     Filesystem   space   quota   limit   grace   files   quota   limit   grace
chama:/home/homes
                 1527M    100G    200G           14913   4295m   4295m
---------------------------
Centerwide user scratch (/carc/scratch/users/mfricke)

Quota information for storage pool Default (ID: 1):

      user/group     ||           size          ||    chunk files
     name     |  id  ||    used    |    hard    ||  used   |  hard
--------------|------||------------|------------||---------|---------
       mfricke|  1512||  592.71 GiB | 1024.00 GiB ||    32784|unlimited

Centerwide scratch quota for project mfricke2016174 (/carc/scratch/projects/mfricke2016174)

Quota information for storage pool Default (ID: 1):
      user/group     ||           size          ||    chunk files
     name     |  id  ||    used    |    hard    ||  used   |  hard
--------------|------||------------|------------||---------|---------
mfricke2016174|2016142||  190.97 GiB| 1024.00 GiB||    23704|unlimited
```

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      2   idle hopper[011-012]
```

sinfo reports information about partitions

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      2   idle hopper[011-012]
```

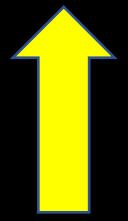
The debug queues are intended for testing your programs.

And for interactive jobs.

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      2   idle hopper[011-012]
```

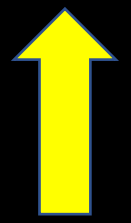
Name

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      2   idle hopper[011-012]
```



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      2   idle hopper[011-012]
```



There are two nodes in this partition.

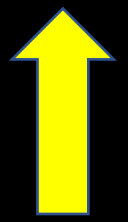
```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      2   idle hopper[011-012]
```

The names of the nodes in the partition

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      2   idle hopper[011-012]
```

The names of the nodes in the partition

```
[vanilla@hopper ~]$ sinfo --partition general
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
```



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)
```

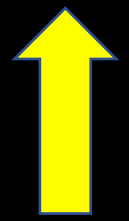
[vanilla@hopper ~]$ man sinfo

```
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
```

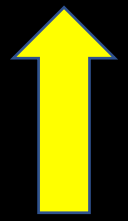


Tell slurm to run a program
on a compute node…

```
[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
```

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.

```
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)
```

```
[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)
                                                                 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?**

```
[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 shell scripts to automate all this in batch mode.

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/
├── 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_example.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

hostname
```

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

```
[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.

```
[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]$
```

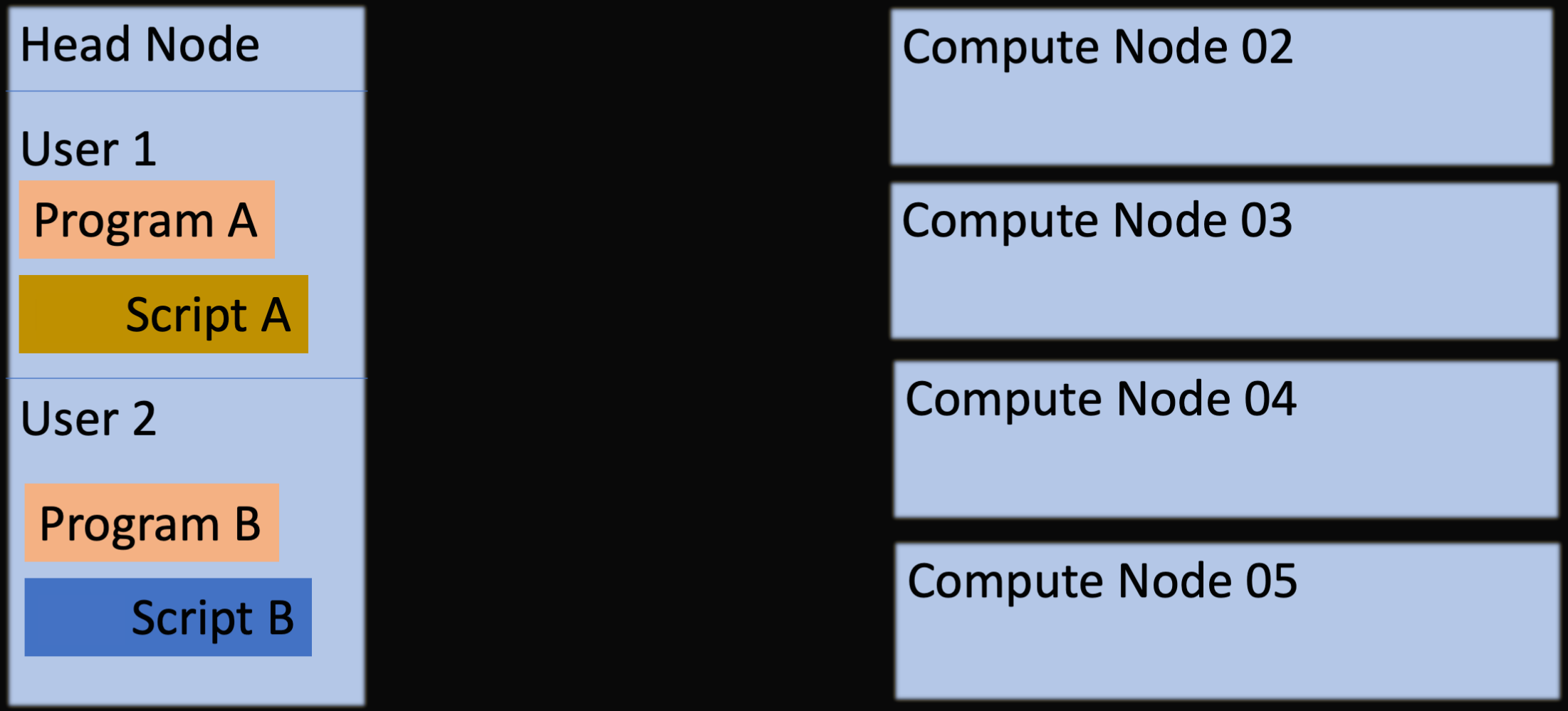
We submit our slurm shell script with the sbatch command.

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

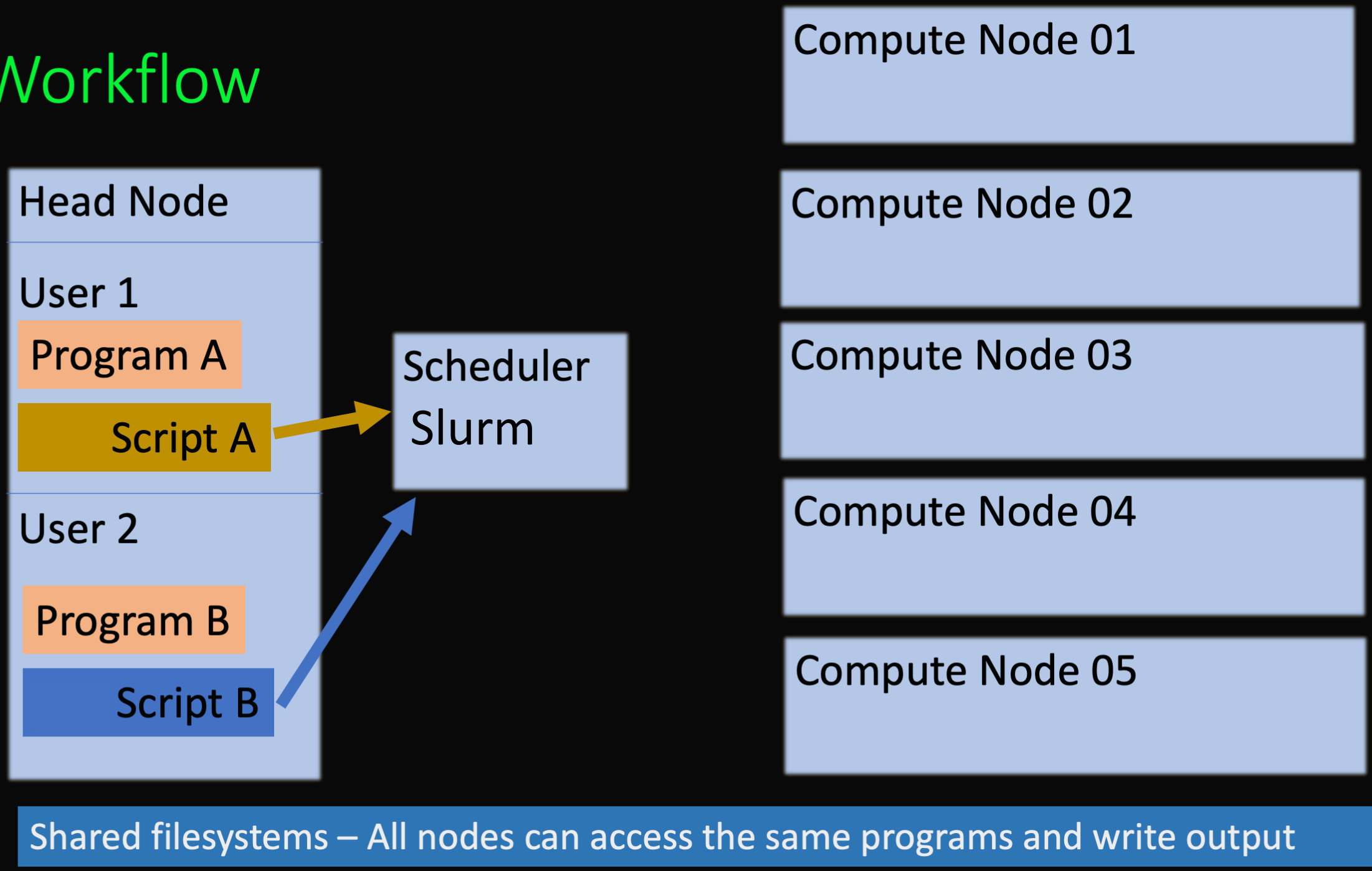
This indicates that the script was successfully sent to the scheduler.

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

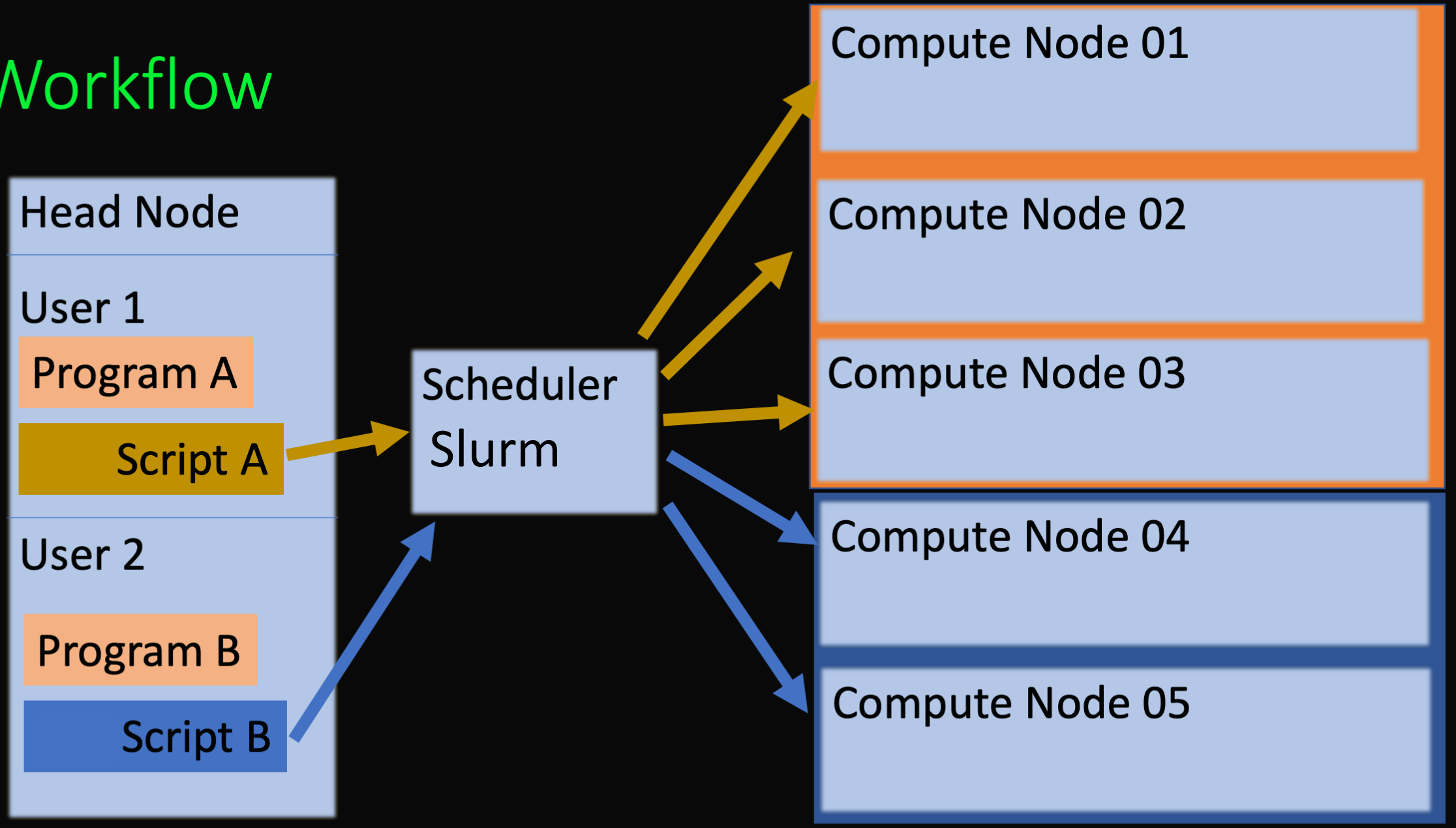
Workflow


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

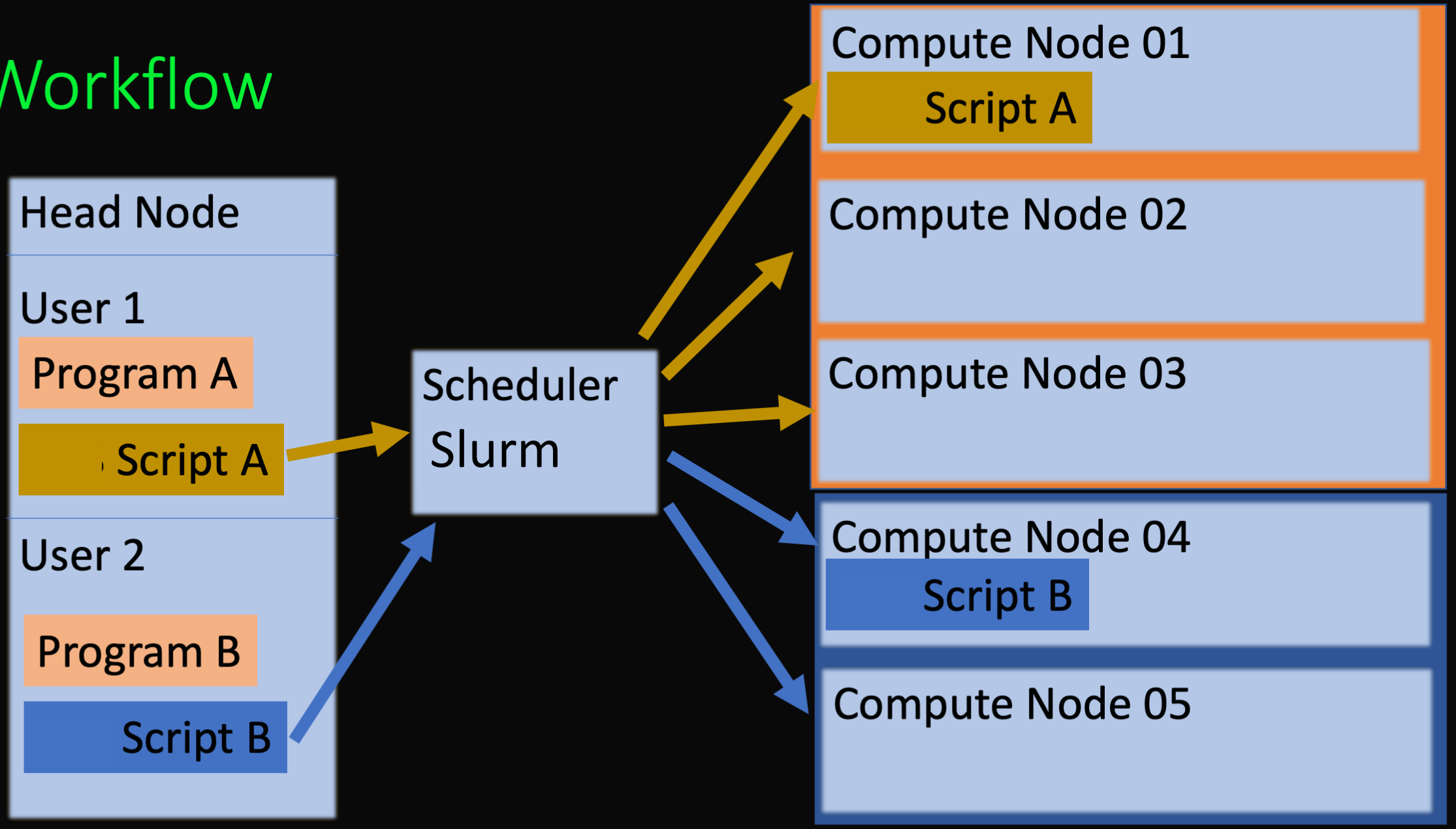

Head Node
User 1
Program A
Script A
User 2
Program B
Script B
Scheduler
Slurm
Compute Node 01
Compute Node 02
Compute Node 03
Compute Node 04
Compute Node 05
Shared filesystems – All nodes can access the same programs and write output

Workflow



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

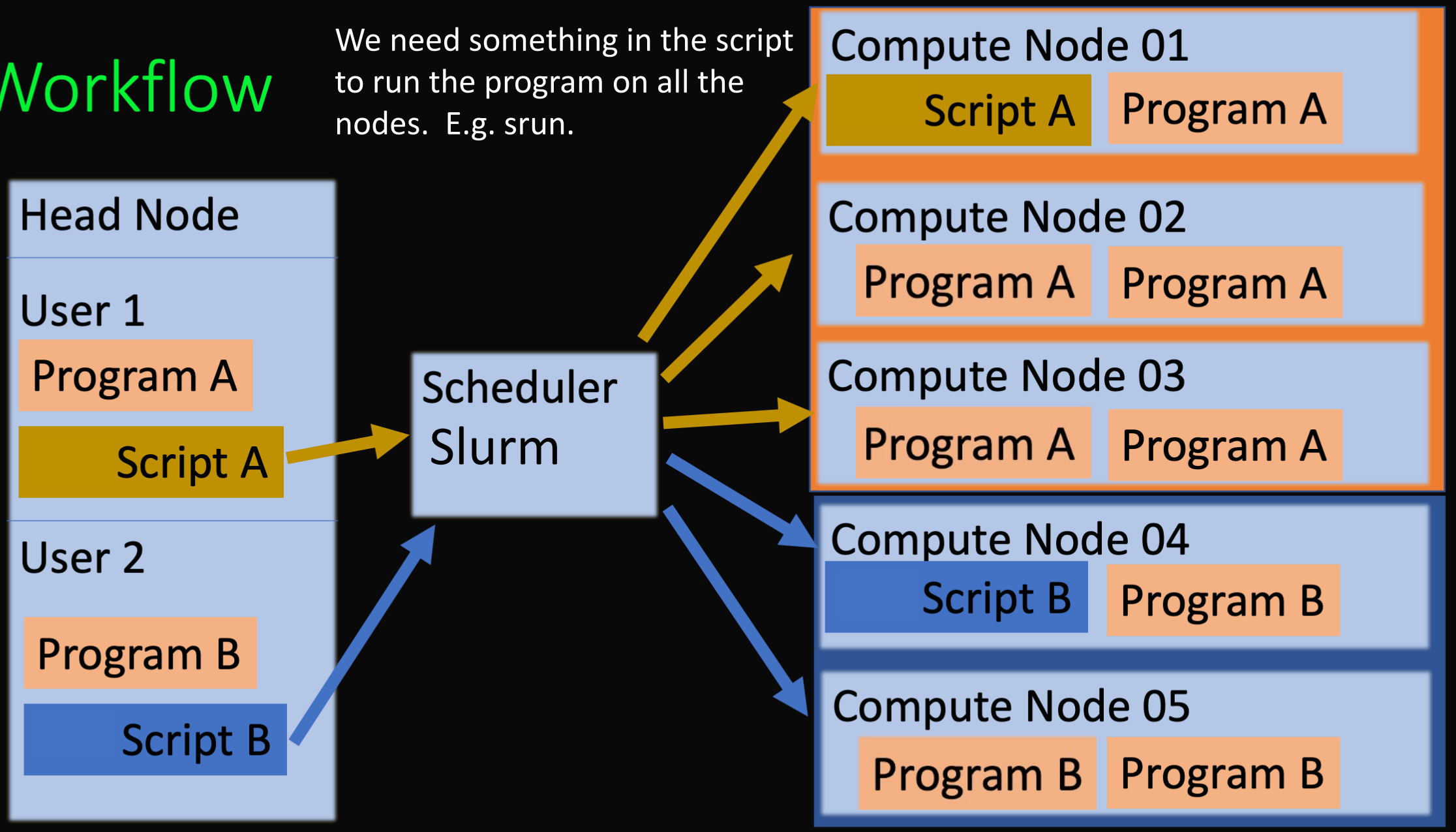
Workflow



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]$ 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]$ cat slurm/calc_pi_serial.sh
#!/bin/bash
#SBATCH --partition debug
#SBATCH --ntasks 1
#SBATCH --time 00:05:00
#SBATCH --job-name calc_pi_serial
#SBATCH --mail-user your_username@unm.edu
#SBATCH --mail-type ALL

module load miniconda3
source activate numpy

cd $SLURM_SUBMIT_DIR
python code/calcPiSerial.py 1000000000
[vanilla@hopper intro_workshop]$
```

```
[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
```

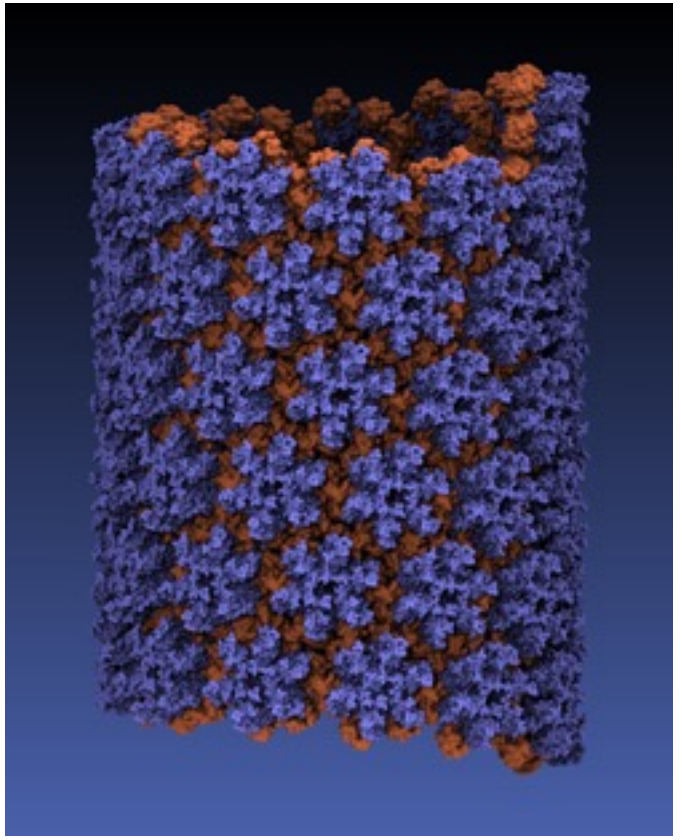
Edit slurm/calc_pi_serial.sh.
Change the email address to your address and submit the script.

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

Take a look at the job output.

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
mfricke@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
```

```
mfricke@hopper:~/workshops/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
```

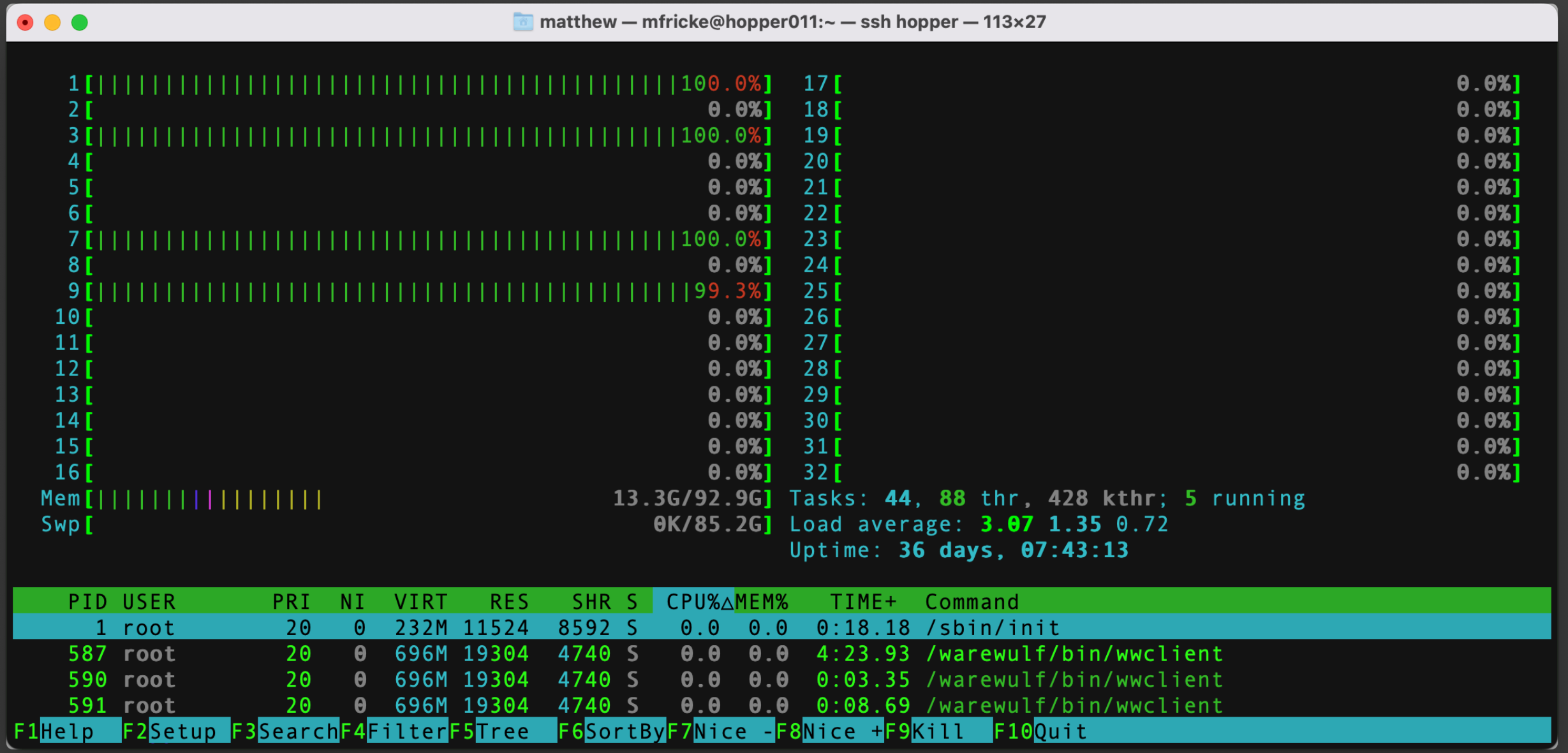
SLURM Batch Script

```
vanilla@hopper:~/workshops/namd $ 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@hopper:~ $ 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”

```
mfricke@hopper:~/~/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
```

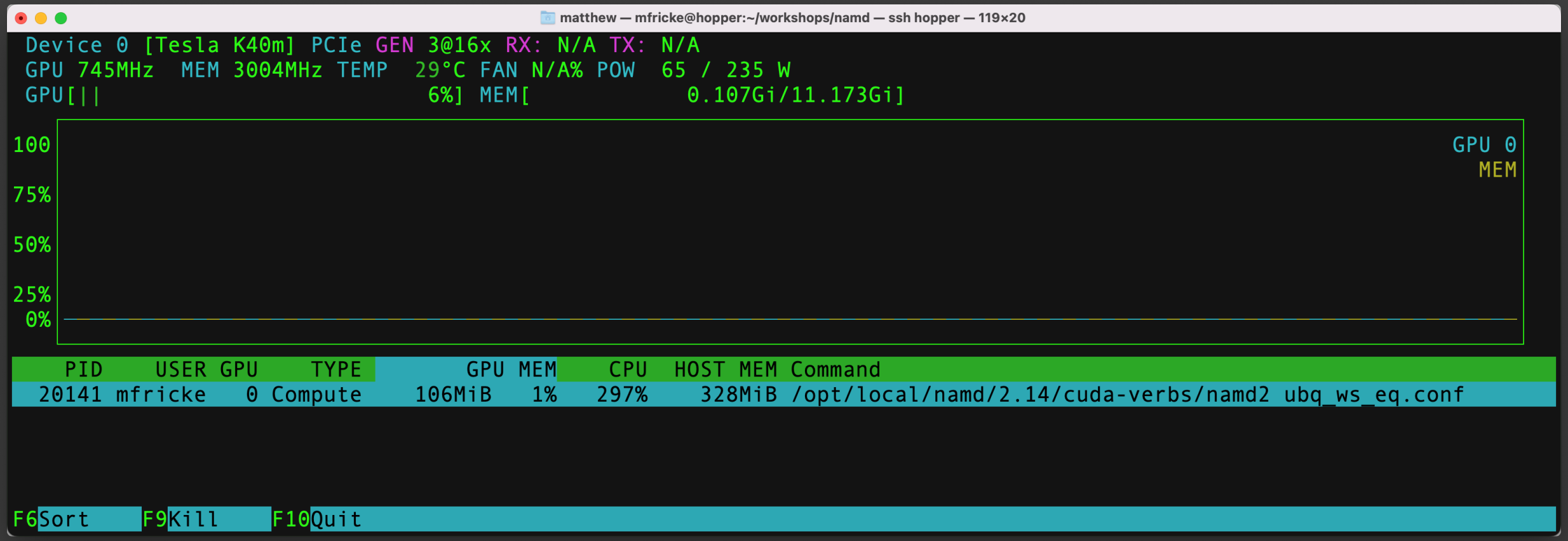
Now we can write a new SLURM script for Xena and just modify the module name and request a GPU

```
vanilla@xena:~/workshops/namd [master ?]$ 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
```

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