CENTER FOR ADVANCED RESEARCH COMPUTINIG

SUPERCOMPUTING, STAR-CCM AND COMPUTATIONAL FLUID DYNAMICS WORKSHOP

Part 1: Matthew, Maisy, Sumaya, Nick, and JJ

Part 2: Keenan Kresl-Hotz,
Timothy M. Schriener
Mohamed S. El-Genk

INSTITUTE FOR SPACE AND NUCLEAR POWER STUDIES

Topics: Part 1

- **Introduction**
  HPC, CARC's mission, available clusters, and user support resources.
- **Linux and BASH**
  Core Linux concepts, BASH shell usage, and interacting with HPC systems.
- **Running Programs**
  Using Slurm commands to allocate hardware, submit jobs, and monitor workflows.
- **STAR-CCM**
  Setting up CFD models, launching STAR-CCM+ jobs, and understanding HPC job scripts.
- **Summary and**
  Use CARC resources efficiently, use tools to check code is running as expected.

NOTE: All these topics (and a lot more) are covered in our online video tutorials

Topics: Part 2

INSTITUTE FOR SPACE AND NUCLEAR POWER STUDIES

Introduction

Introduce STAR-CCM+ licensing at UNM; run on CARC HPC and local systems.

Background

Explain CFD: continuum mechanics, finite volume method, Direct Numerical Simulation, Reynolds-Averaged Navier-Stokes, Large Eddy Simulation.

STAR-CCM+ Workflow

Build geometry, assign physics, create mesh, set data collection, automate with macros, and checkpointing.

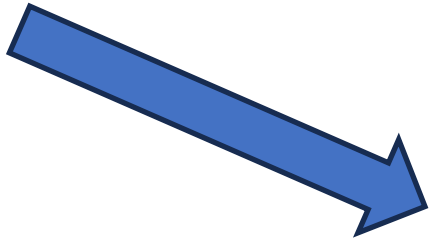
Example Problem: Laminar Pipe Flow

Simulate 3D pipe flow; apply meshing, boundary conditions, and analyse results.

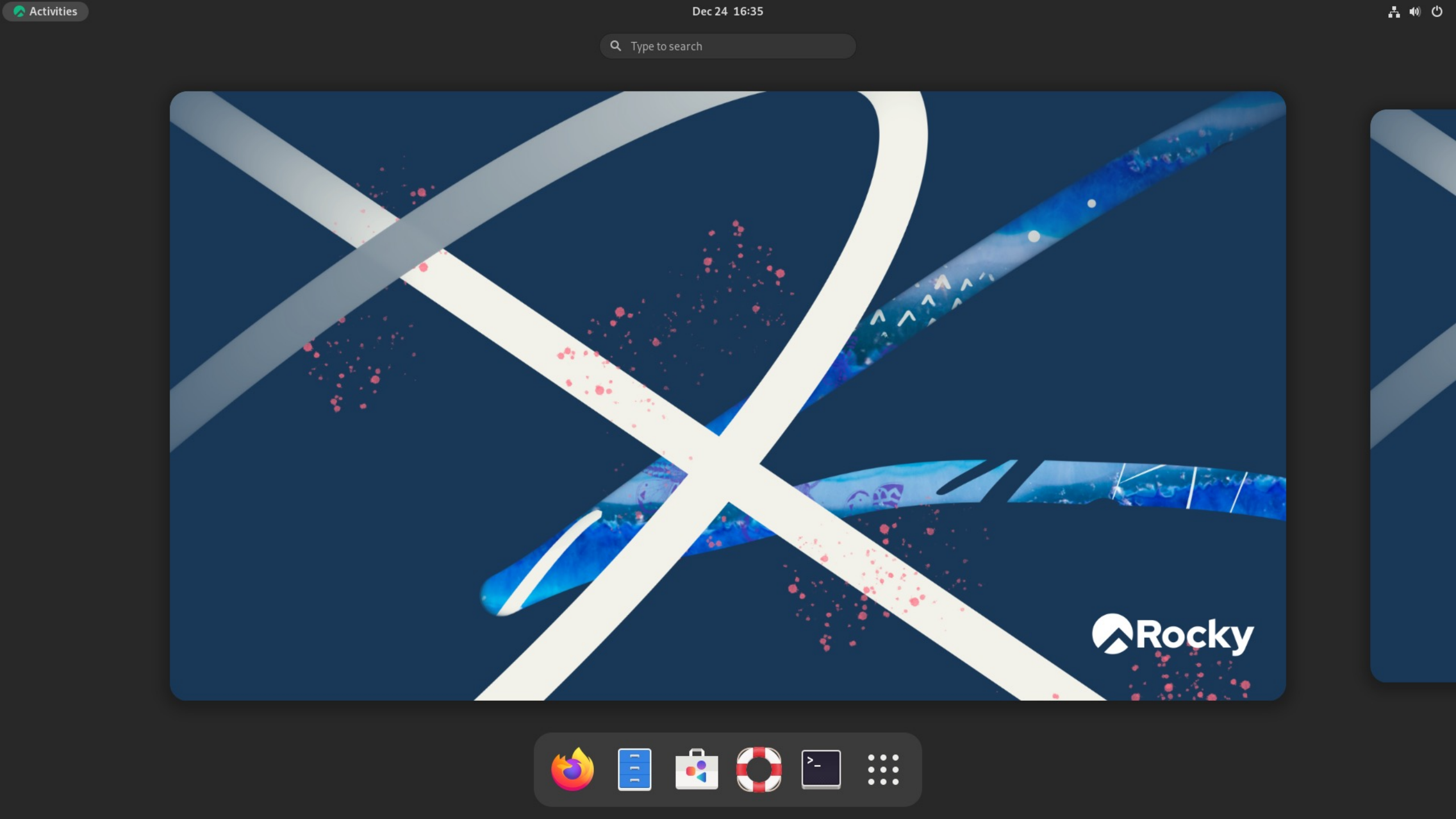
Summary and Discussion

Emphasise saving, verifying settings, analysing critically, and using HPC resources effectively.

Lookup your CARC username



Username	First Name	Last Name	Email	Role	Status	Notifications	Actions
abairagi	Anutam	Bairagi	abairagi@unm.edu	User	Active	☑	
acar	Zafer	Acar	acar@unm.edu	User	Active	☑	
ashaheen	Ahmad	Shaheen	anshaheen22@unm.edu	User	Active	☑	
bradenfod	Braden	Foderaro	bfoderaro2021@unm.edu	User	Active	☑	
dlei	Di	Lei	dlei@unm.edu	User	Active	☑	
fellis	Finn	Ellis	finnellis@unm.edu	User	Active	☑	
hfchen	Haifei	Chen	hfchen@unm.edu	User	Active	☑	
itsegok	JEREMIAH	ABOK	ABOKJEREMIAH@UNM.EDU	User	Active	☑	
mfricke	George Matthew	Fricke	mfricke@unm.edu	Manager	Active	☑	
mnu	Minghui	Chen	mnu@unm.edu	User	Active	☑	
mpathirage	Madura	Pathirage	mpathirage@unm.edu	User	Active	☑	
najmabms	Najma	Al-Kharousi	najmabms@gmail.com	User	Active	☑	
rogerd	Roger	Davis	rogerdavis85@unm.edu	User	Active	☑	
schrient	Timothy M	Schriener	schrient@unm.edu	User	Active	☑	
stzuniga	Steven G	Zuniga	stzuniga@unm.edu	User	Active	☑	
teshorty	Temair Ellen	Shorty	teshorty@unm.edu	User	Active	☑	
tundra	Steven	Kao	tundra@unm.edu	User	Active	☑	
yujianlu	Yujian	Lu	yujianlu@unm.edu	User	Active	☑	
yuqiliu	Yuqi	Liu	yuqiliu@unm.edu	User	Active	☑	


Activities
Dec 24 16:35
Type to search
Rocky

Logging into Hopper


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

You may be prompted for a password…

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

Anywhere you see vanilla replace it with your username

Logging into Hopper

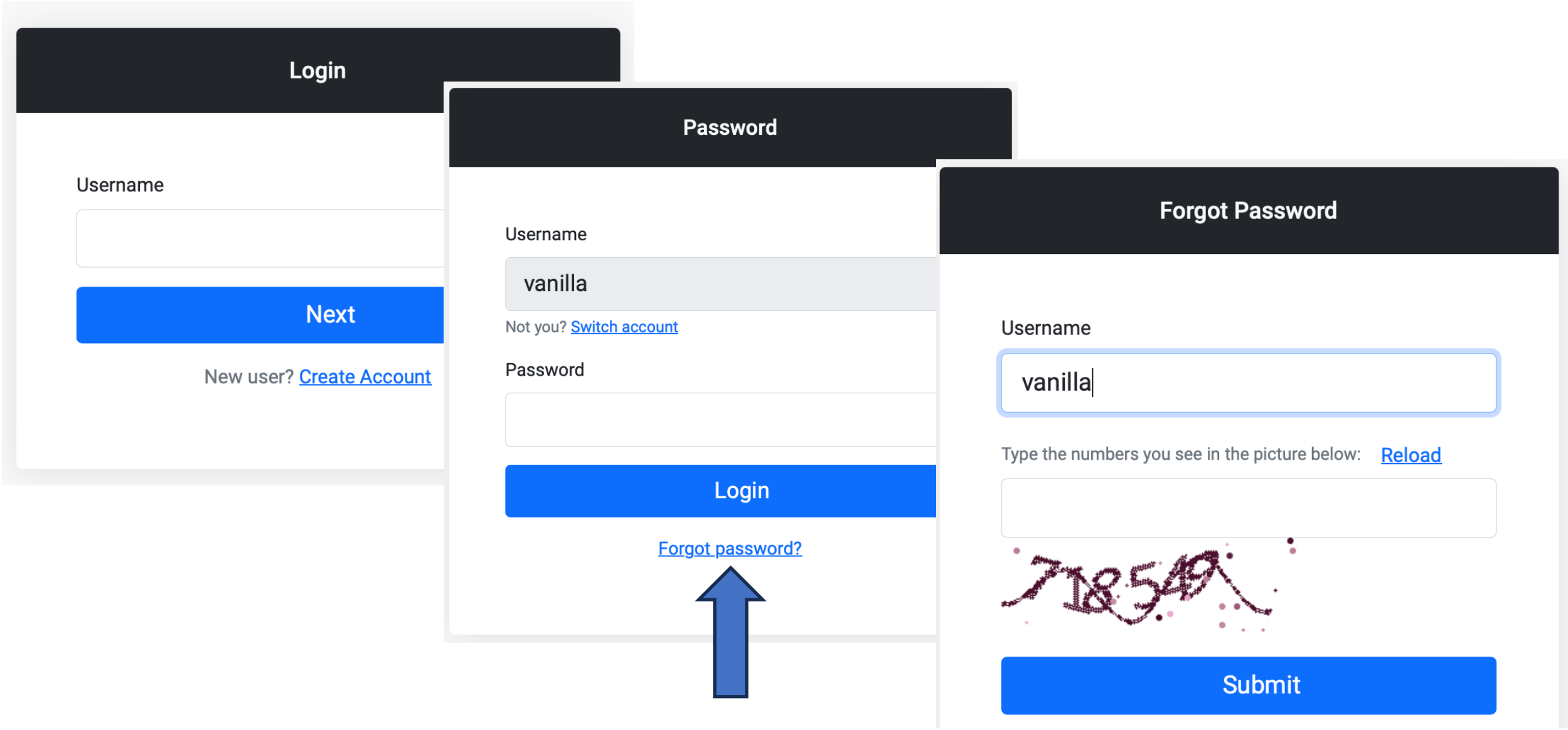
```
--------------------------------------------------------------------------------
  Welcome to Hopper

  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.

  Hopper is our newest general purpose cluster and member of the Taos/Hopper condo.

  There are four types of slurm partitions on Hopper:
  1) General - this partition is accessible by all CARC users.
  2) Debug - for testing your code and interactive jobs. Short time limits so that nodes are usually
available right away.
  3) Condo - this partition is accessible by users who are members of the Hopper/Taos condo. Users with
access to this partition also have access to the community partition on the Taos cluster. Jobs run here
may be interrupted by the hardware owner, so we recommend that your software support checkpointing so it
can recover.
  4) Private partitions - these partitions are dedicated to the condo grant/lab/center that purchased
the associated hardware.

  Type "qgrok" to get information about the partitions to which you have access.
  Enter "quotas" to see your storage usage and limits.
  For a list of software installed on Hopper enter "module spider".
--------------------------------------------------------------------------------
  ******************************************************************************
  * Logging in under another person's account is strictly forbidden and will *
  * result in the account being     locked.                                    *
  ******************************************************************************
```

A bit about CARC

- **Office Hours:** 327 meetings, **27 per month**
- **Help Tickets:** 900+ issues resolved in the past year
- **Workshops:** 20+ Intro & Subject matter

  e.g. Intro to Linux, Quantum Computing
- **New Projects:** 53 research, 17 courses, 70 total
- **Video Tutorials:** 4,000 views, 300 viewing hrs/yr
- 2 senior CS courses on HPC admin

If you don’t know your password: https://mokey.alliance.unm.edu



What is Supercomputing?

1 processor with 6 cores.
Can run 6 programs at a time.

1 processor 2 cores

What is Supercomputing?

1 processor with 6 cores.
Can run 6 programs at a time.

1 processor with 8 cores.
Can run 8 programs at a time.

1 processor 2 cores

At the time Cray Computers dominated supercomputers with Monolithic machines.

Seymore Cray was not impressed with Linux Clusters:

“If you were plowing a field, which would you rather use? Two strong oxen or 1024 chickens?”

Seymore Cray with Cray-1 (1976)

In 1993 NASA devel

These are Linux bas

At the time Cray Co
supercomputers witl

Seymore Cray was
Clusters:

“If you were
which would
Two strong c
chickens?”

Roadrunner 1999

Result of a “bake-off” between CARC and Linux and NCSA’s Windows NT Wolfpack cluster.

Roadrunner was made the first NSF nationally available Linux cluster.

In 1993 NASA developed the first **Beowulf** cluster. These are Linux based clusters of many computers.

CA
effe
Clu

ARTICLE

Design and Analysis of the Alliance/University of New Mexico Roadrunner Linux SMP SuperCluster

Authors: David A. Bader, Arthur B. MacCabe, Jason R. Mastaler, John K. McIver, III, Patricia A. Kovatch | Authors Info & Claims

IWCC '99: Proceedings of the 1st IEEE Computer Society International Workshop on Cluster Computing • Page 9

Published: 02 December 19 Publication History

1 0

Publisher Site

Abstract

This paper will discuss high performance clustering from a series of critical topics: architectural design, system software infrastructure, and programming environment. This will be accomplished through an overview of a large scale, high performance SuperCluster (named Roadrunner) in production at The University of New Mexico (UNM) Albuquerque High Performance Computing Center (AHPCC). This SuperCluster, sponsored by the U.S. National

	Easley (2025)	Hopper (2023)	Wheeler (2016)	Xena (2016)	Taos (2020)
OS	Rocky 9.5	Rocky 8	CentOS 7	CentOS 7	CentOS 7
Interconnect	Infiniband 400G	InfiniBand 200G/100G	InfiniBand 40G	InfiniBand 56G	InfiniBand 56G
# of Nodes	62	61	304	32	9
GPUs	8 H100*s, 24 L40*s	37 A100s	0	33 K40s	0
CPU Cores	**4,160**	**2,176**	**2,432**	**720**	**180**
RAM/node	256 GB – 2 TB	96 GB – 1.5 TB	48 GB	64 GB – 3 TB	64 GB – 1 TB
Local disk/node	1 TB	448 GB	Diskless	1 TB	1 TB
CPU Architecture	**Sapphire Rapids***	Intel Cascade Lake	Intel Nehalem EP	Intel Haswell	Intel Broadwell

*AMX – Native 2D registers and matrix operations, H100s for comp chem, L40s for ML

	Easley (2025)	Hopper (2023)	Wheeler (2016)	Xena (2016)	Taos (2020)
OS	Rocky 9.5	Rocky 8	CentOS 7	CentOS 7	CentOS 7
Interconnect	Infiniband 400G	InfiniBand 200G/100G	Inf		
# of Nodes	62	61			
GPUs	8 H100*s, 24 L40*s	37 A100s			
CPU Cores	**4,160**	**2,176**			
RAM/node	256 GB – 2 TB	96 GB – 1.5 TB			
Local disk/node	1 TB	448 GB	D		
CPU Architecture	**Sapphire Rapids***	Intel Cascade Lake	Intel Nehalem EP	Intel Haswell	Intel Broadwell

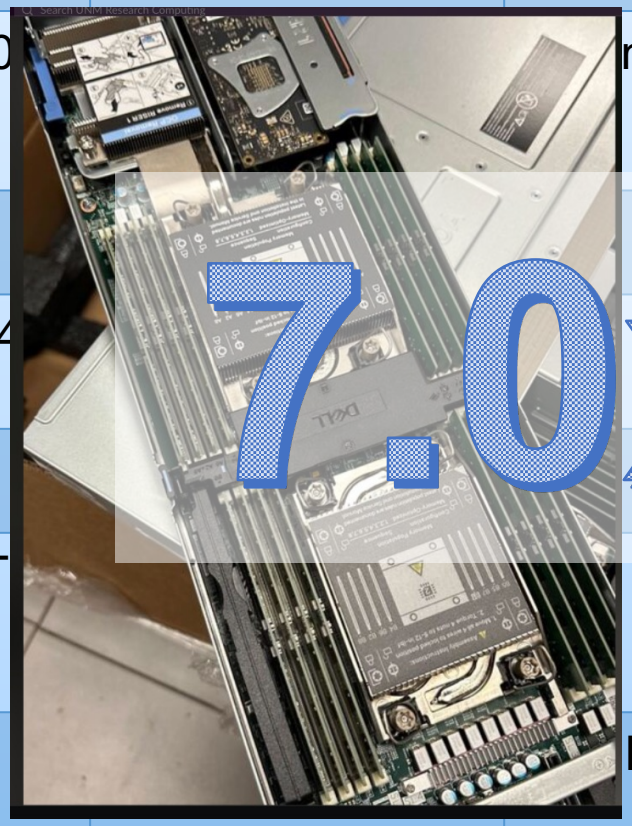
Wheeler 2400 cores in 306 nodes, 12 racks
Estimated 25 Terra Floating Point Operations/second (Tflops)

*AMX – Native 2D registers and matrix operations, H100s for comp chem, L40s for ML

	Easley (2025)	Hopper (2023)	Wheeler (2016)	Xena (2016)	Taos (2020)
OS	Rocky 9.5	Rocky 8	CentOS 7	CentOS 7	CentOS 7
			el Nehalem EP	Intel Haswell	Intel Broadwell
		Lake			

4000 Cores in 1 Rack

175 Tflops



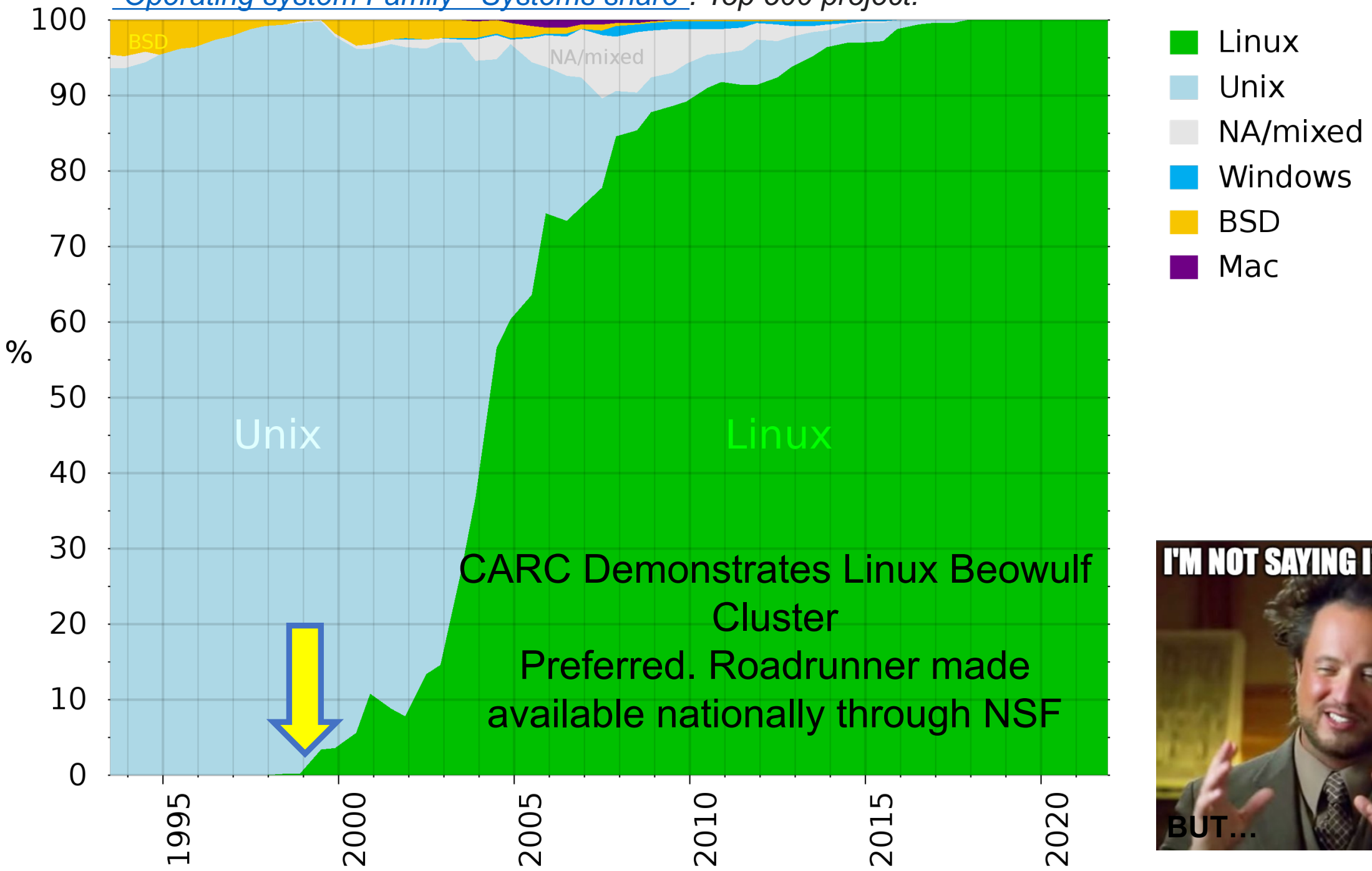
4 compute nodes per 2U slot

7.0x

Wheeler 2400 cores in 306 nodes, 12 racks
Estimated 25 Tera Floating Point Operations/second (Tflops)

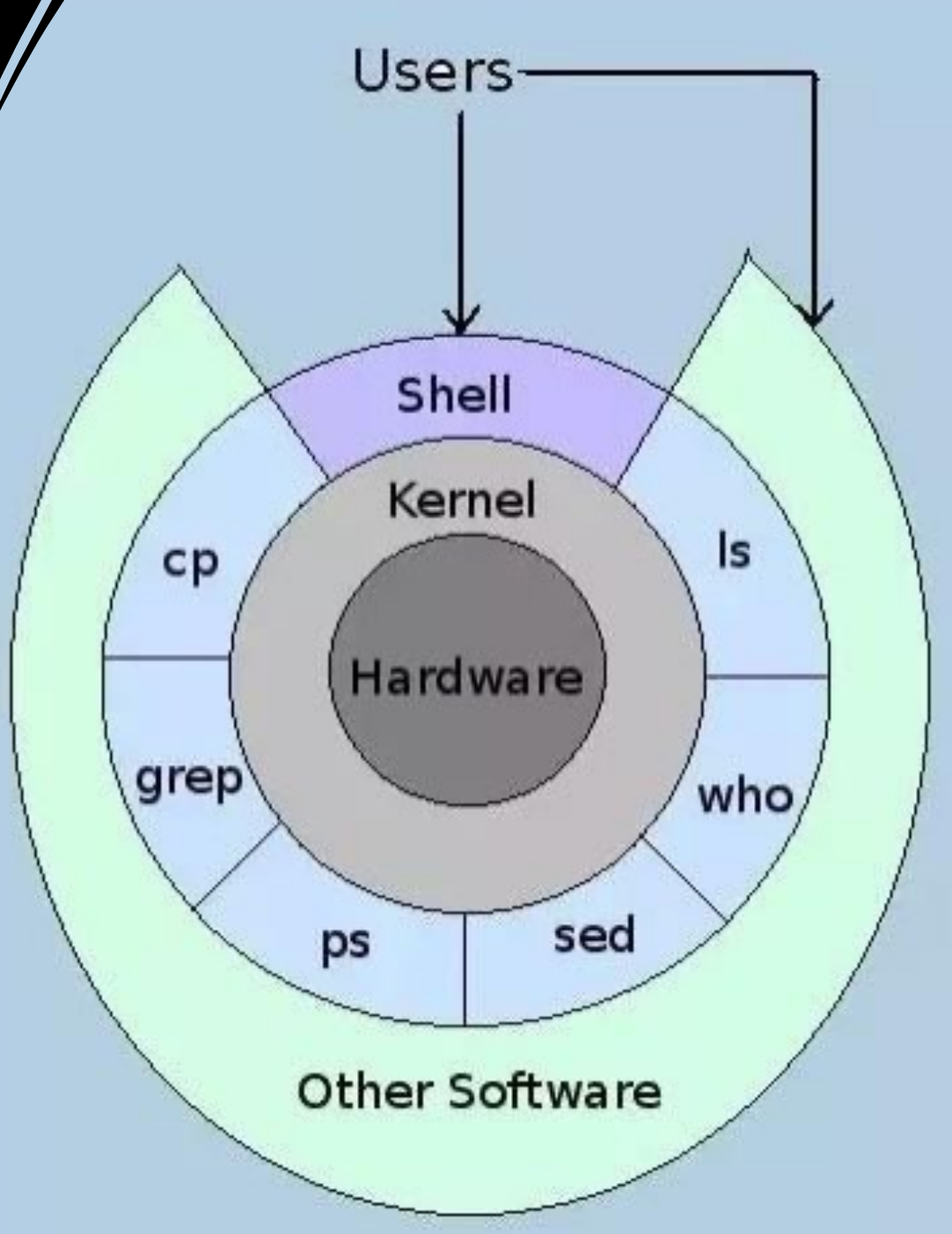
*AMX – Native 2D registers and matrix operations, H100s for comp chem, L40s for ML

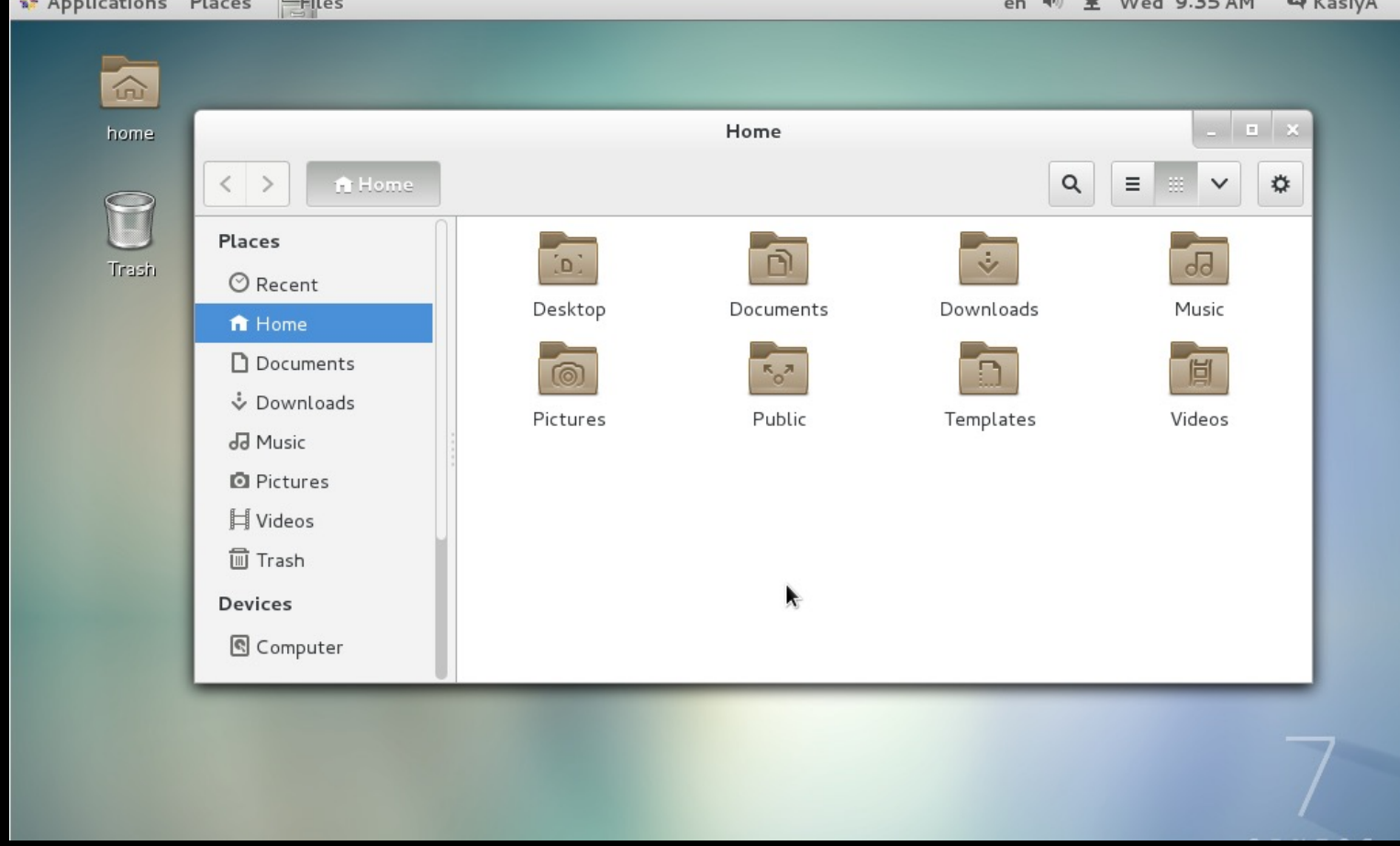
"Operating system Family - Systems share". Top 500 project.



Linux and the BASH Shell

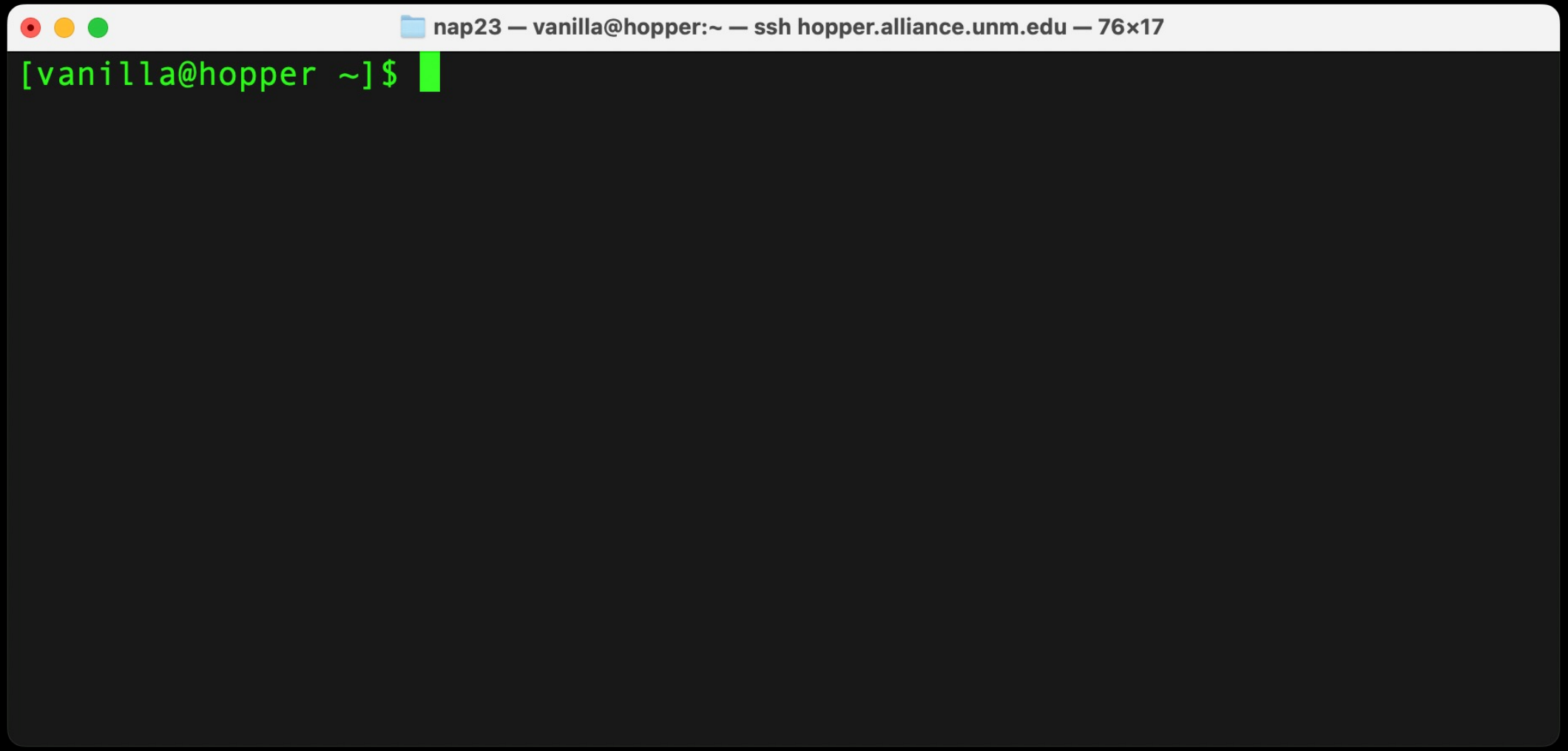
- The Kernel manages access to the hardware in a computer.
- An Operating System (OS) is the Kernel plus useful programs provided by the OS.
- The “shell” is the outermost layer of the OS.
- It is where the user interacts with the OS.





Graphical Shells (GUIs)

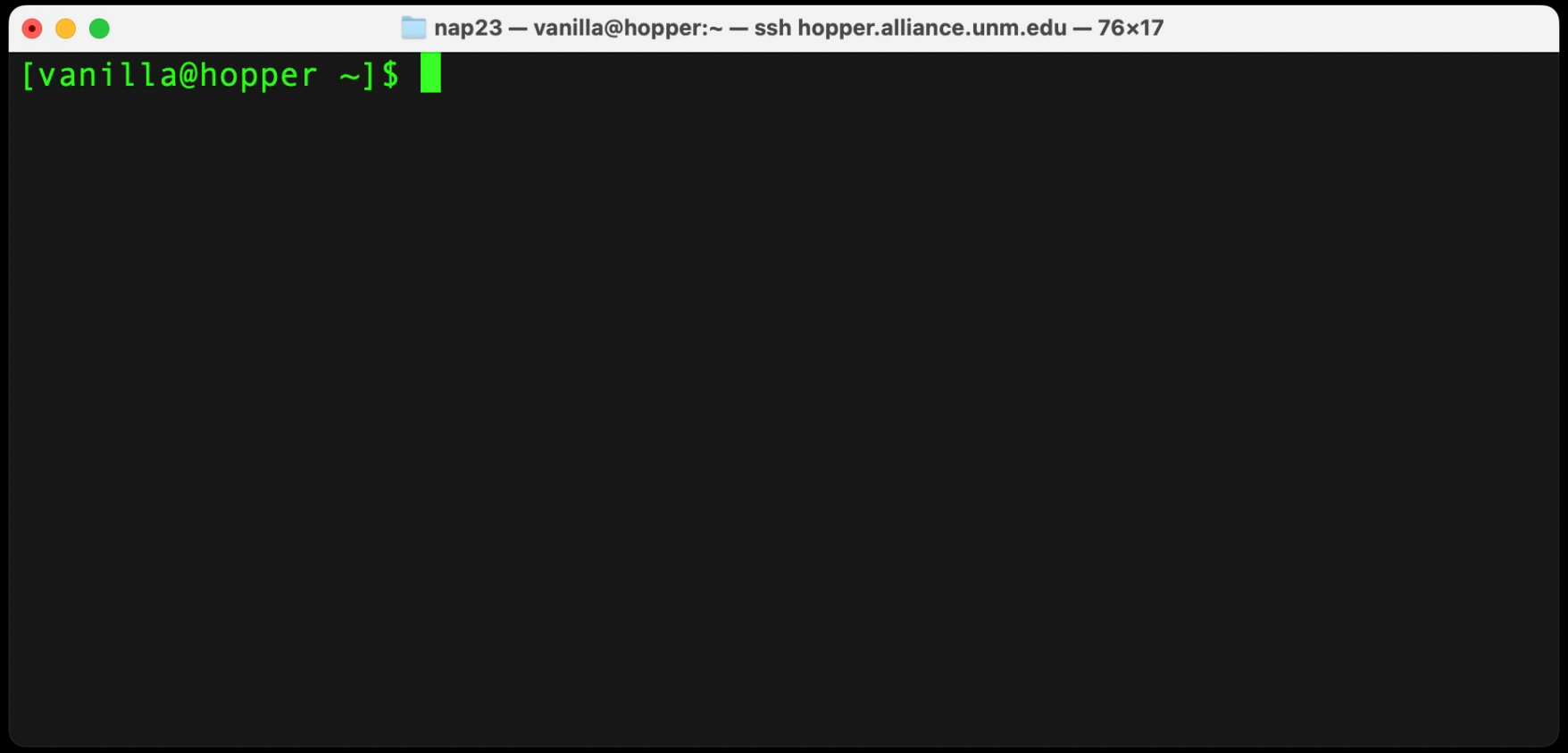
Logging into Hopper

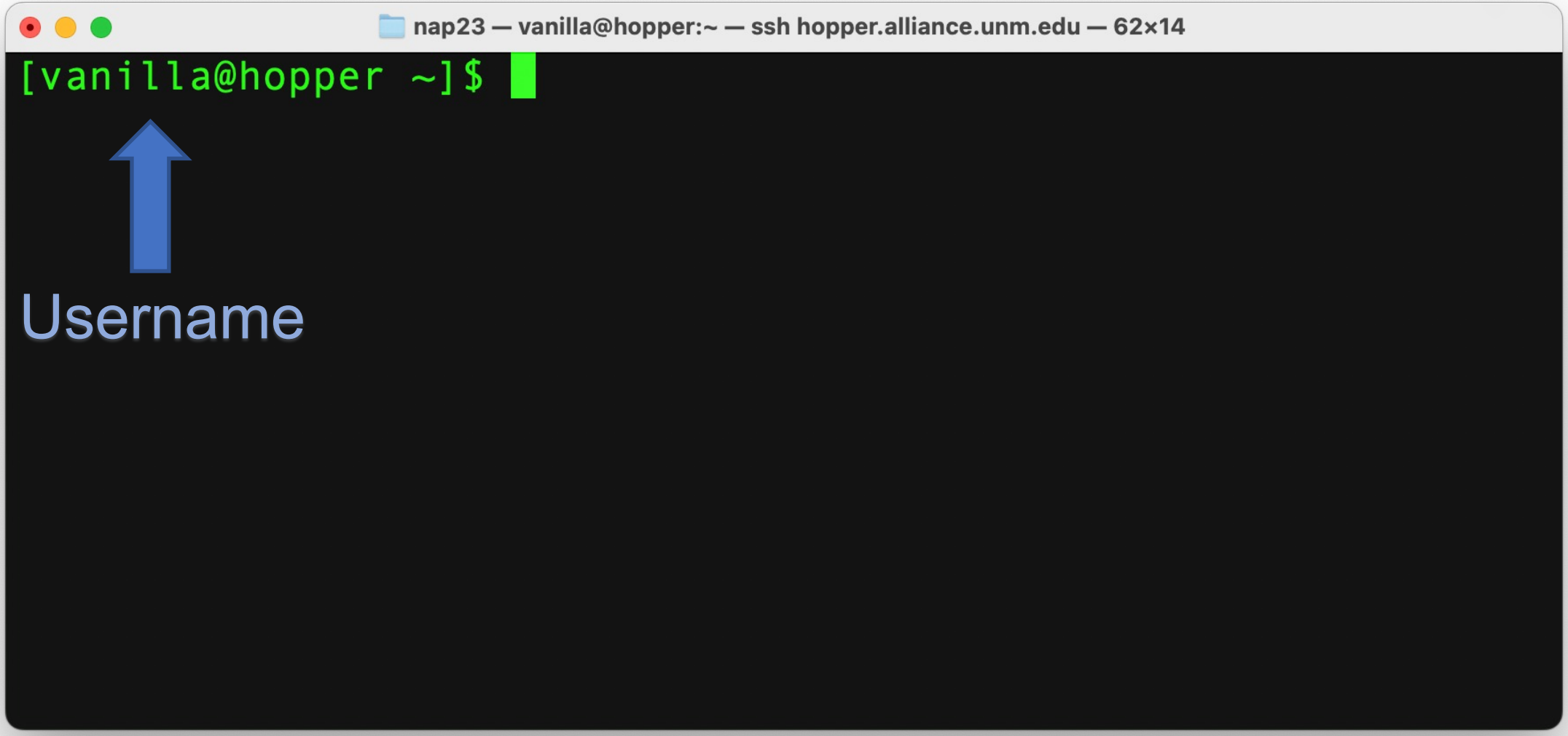


Linux and the BASH Shell

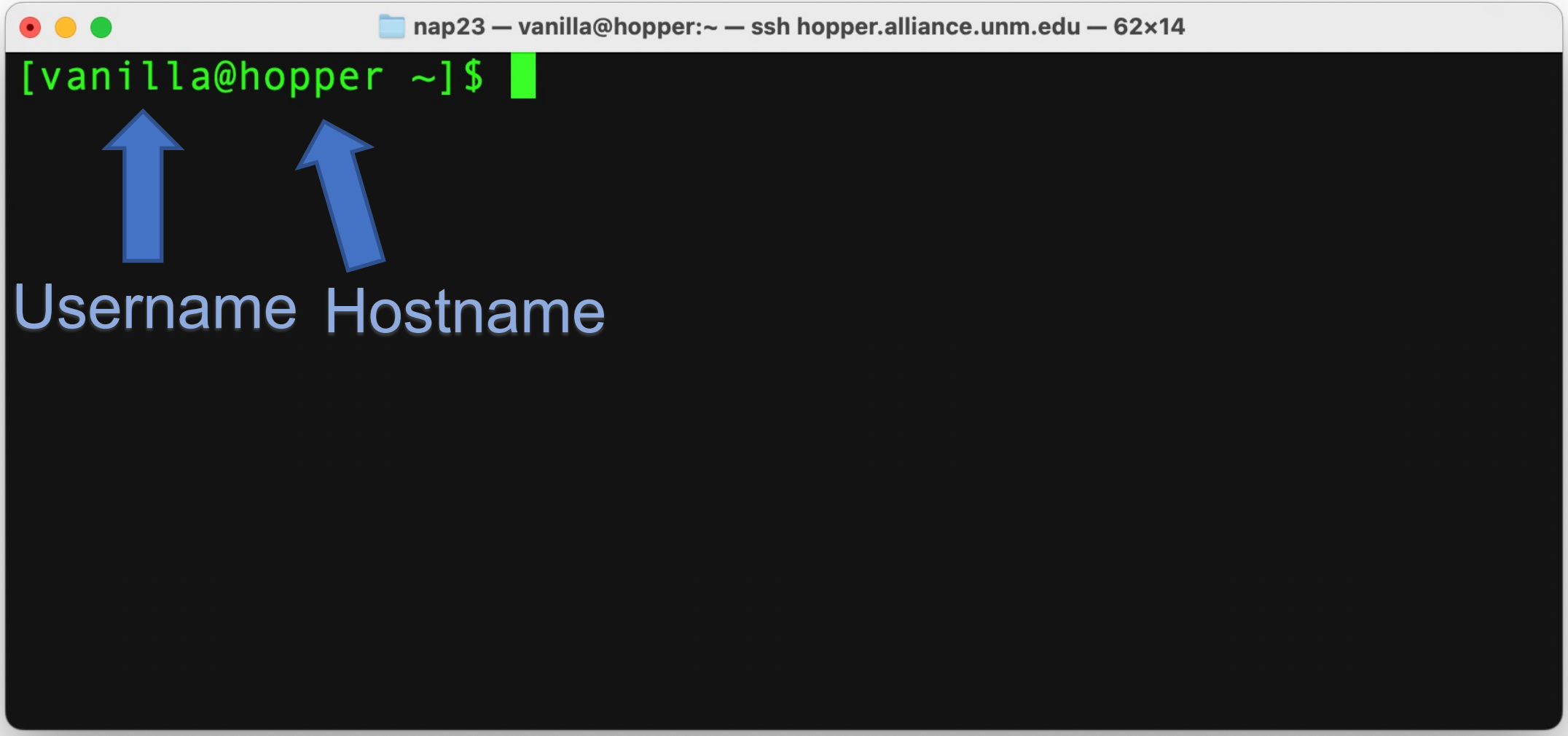
The Borne-Again Shell (BASH)

Written in 1976 by Stephen Bourne for UNIX version 7.

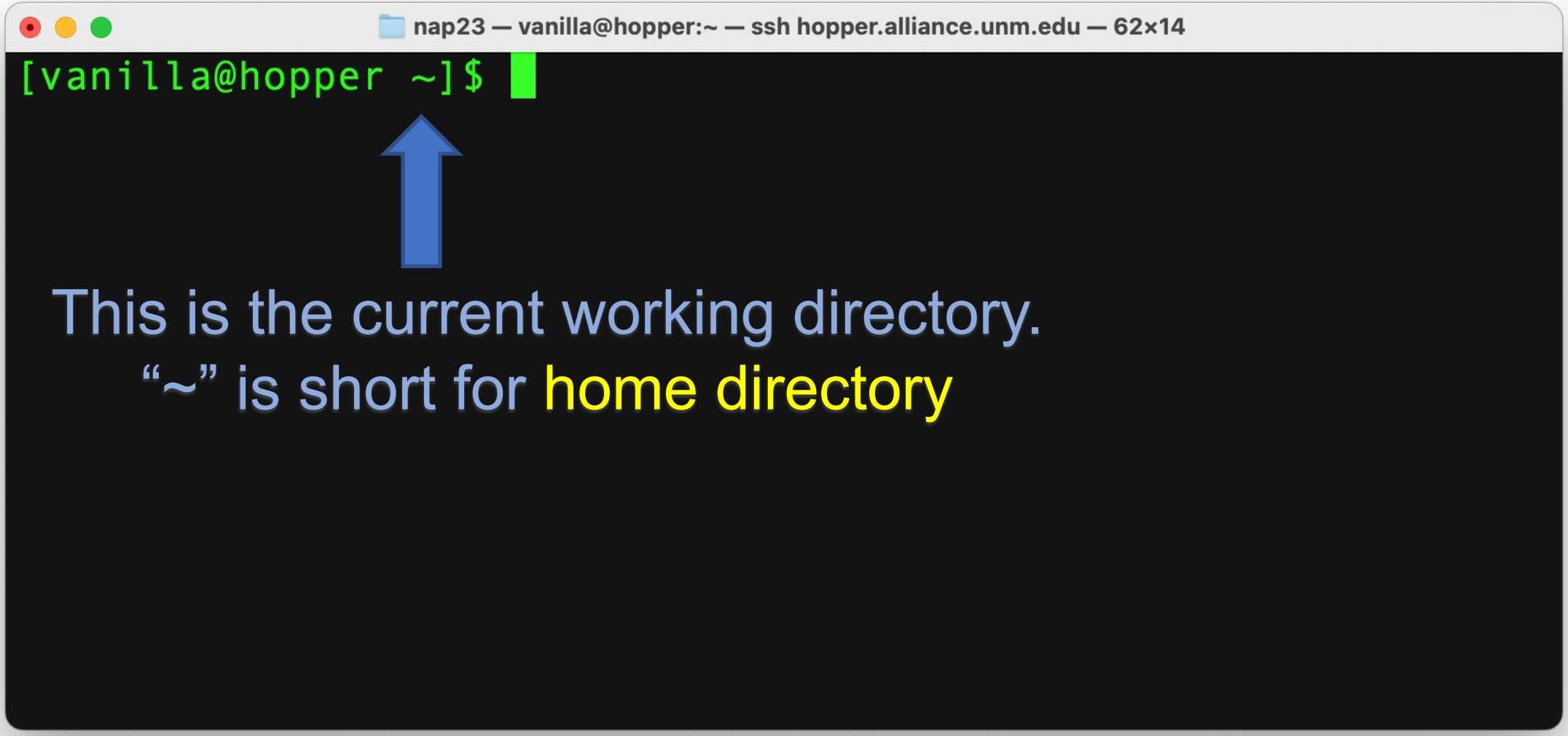




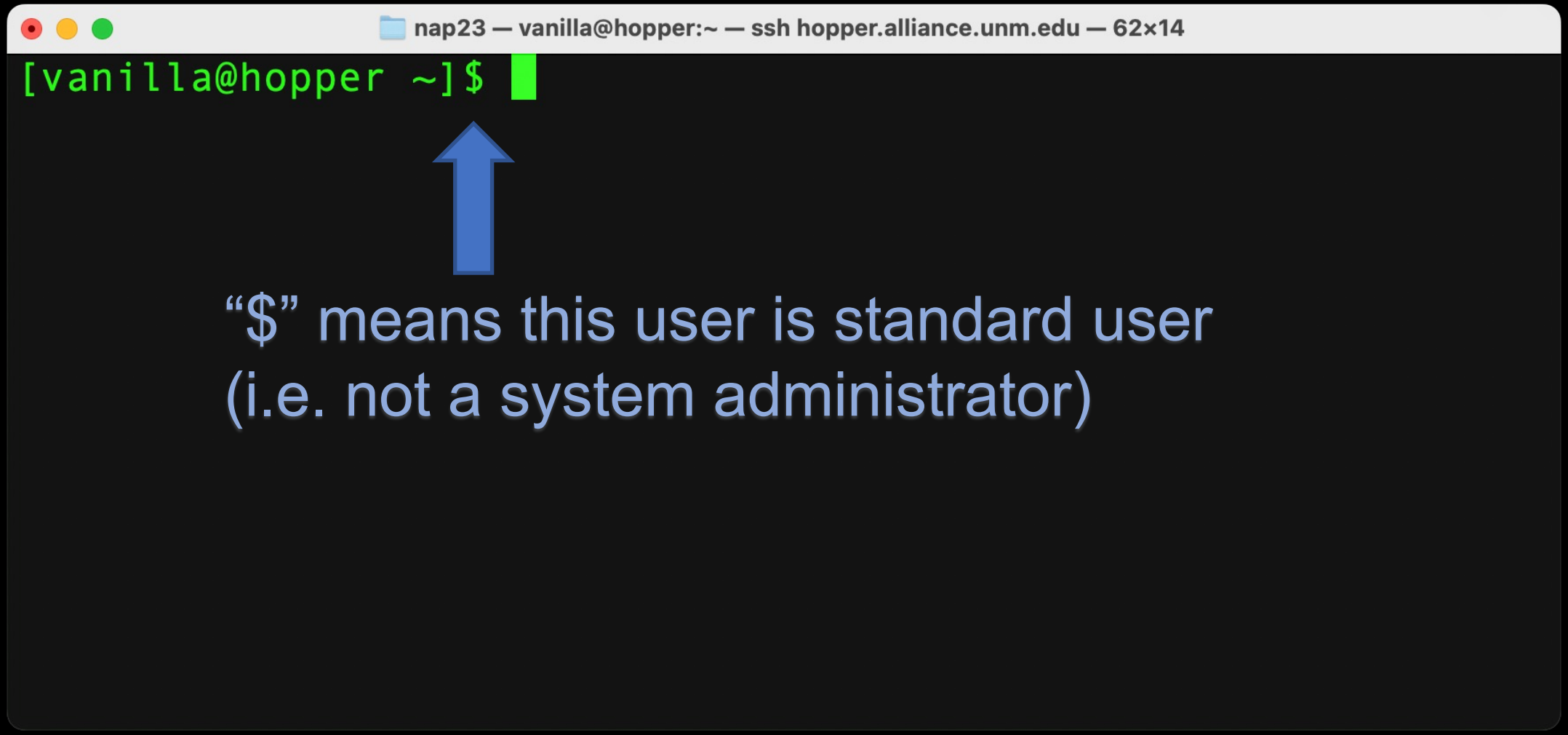
Understanding the BASH prompt…



Understanding the BASH prompt…



Understanding the BASH prompt…

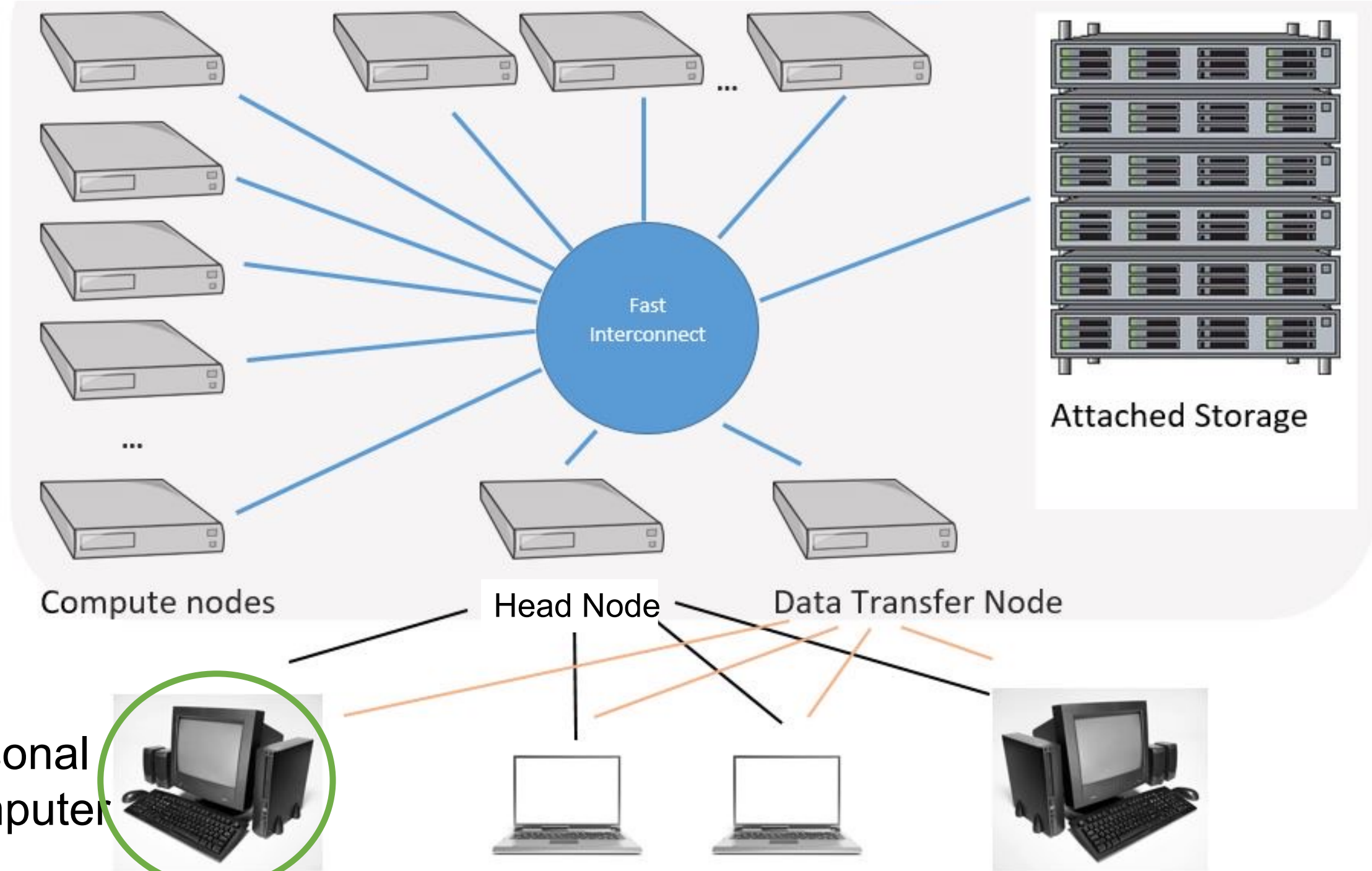


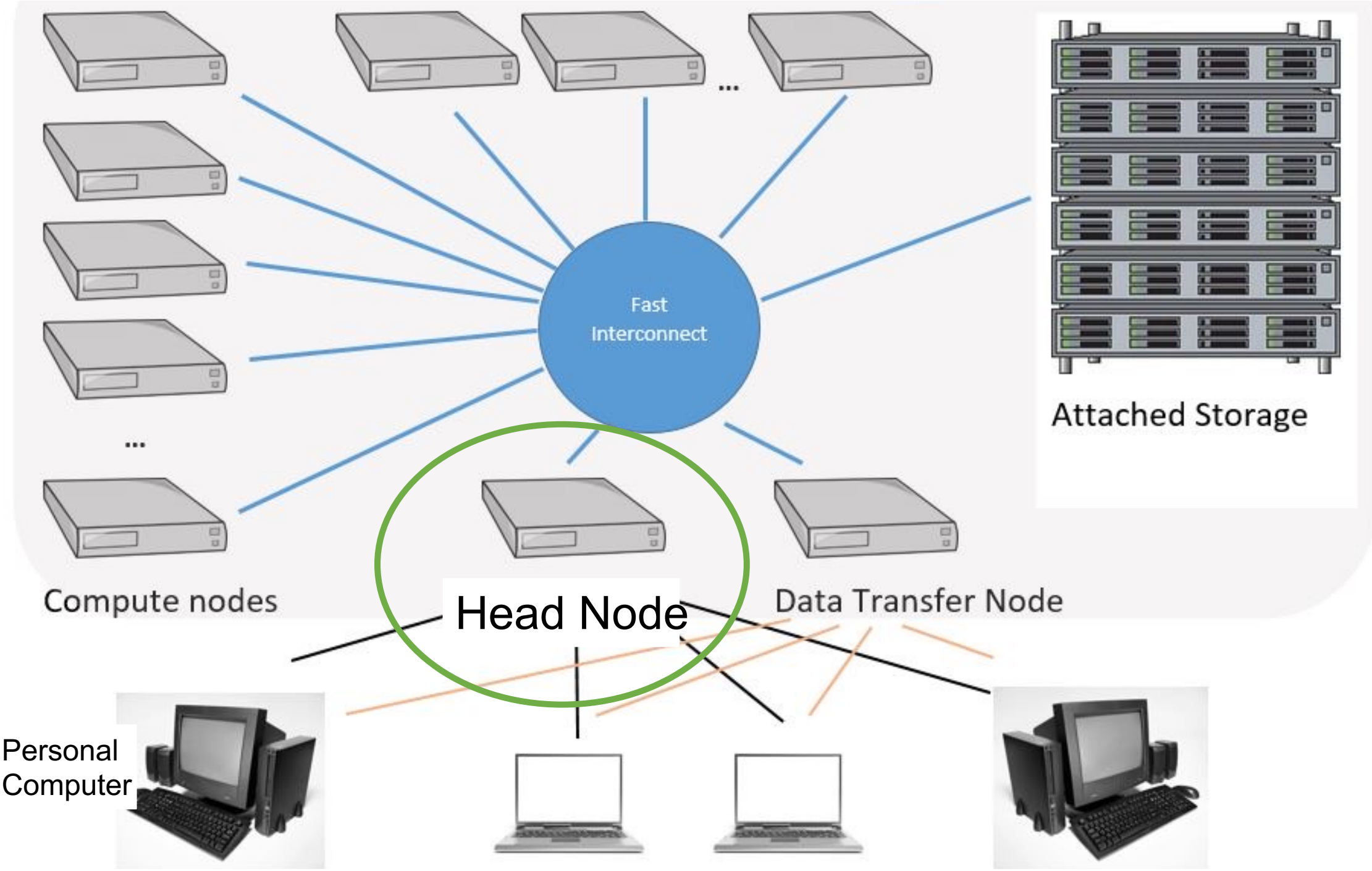
Understanding the BASH prompt…

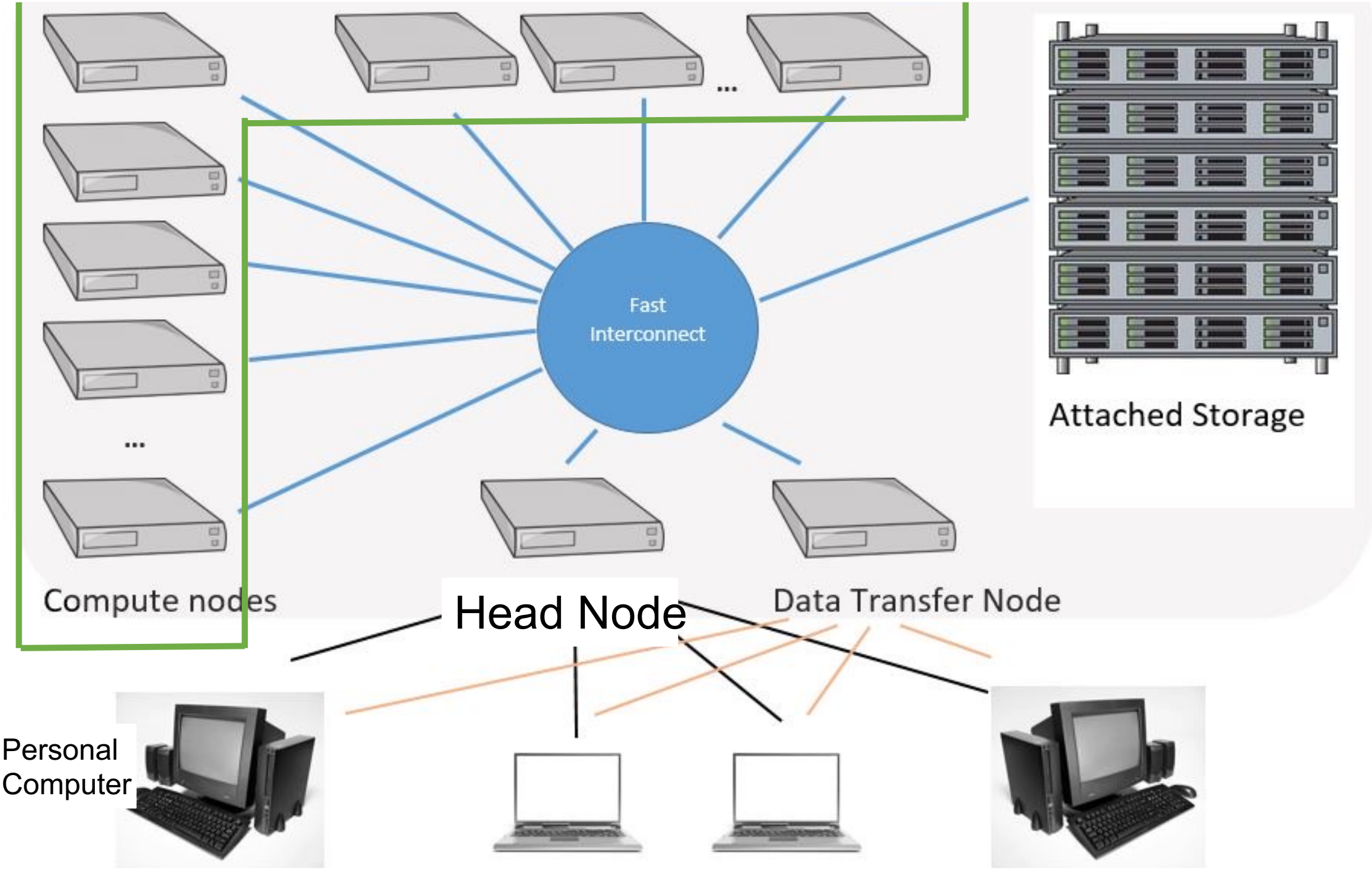
```
[vanilla@hopper ~]$ pwd
/users/vanilla
[vanilla@hopper ~]$
```

Figuring out where you are in the filesystem…


HPC
Cluster


Fast
Interconnect
Attached Storage
Compute nodes
Head Node
Data Transfer Node
Personal
Computer

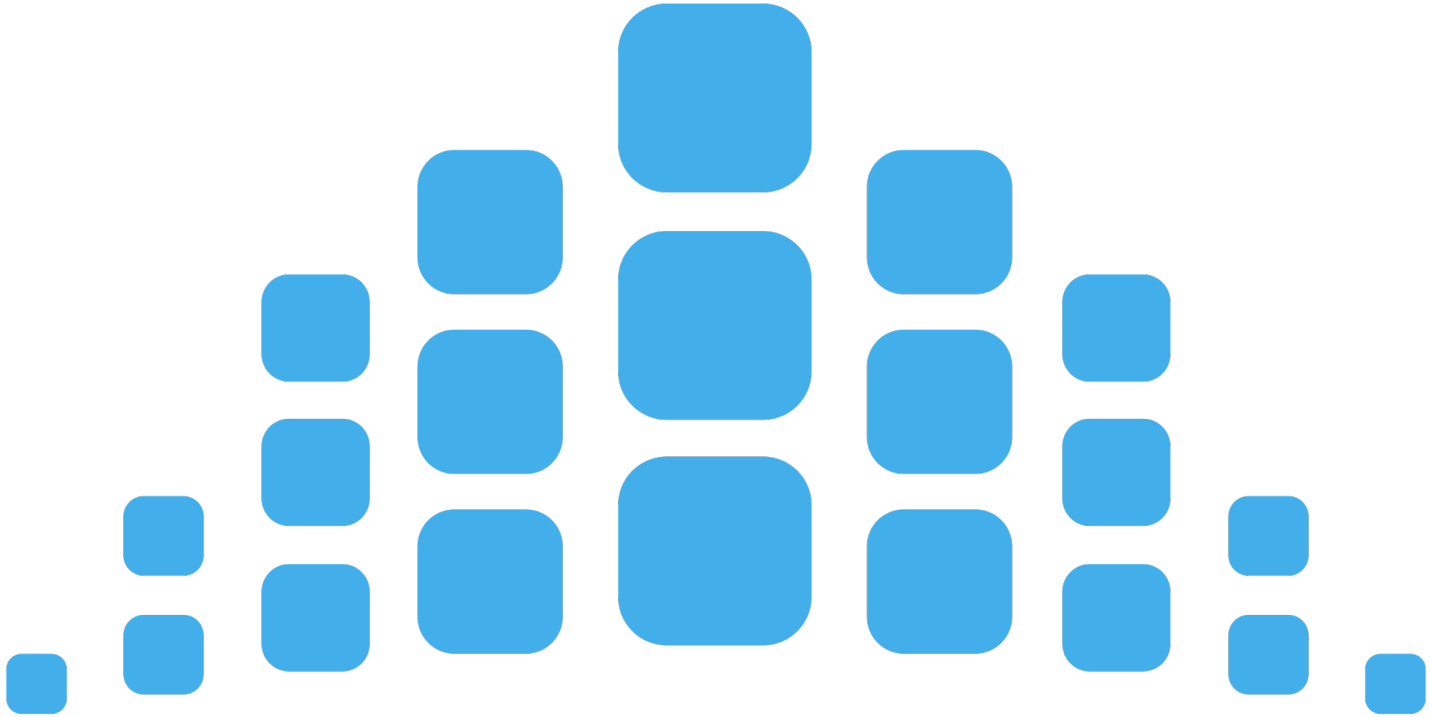

Fast
Interconnect
Attached Storage
Compute nodes
Head Node
Data Transfer Node
Personal
Computer


Fast
Interconnect
Attached Storage
Compute nodes
Head Node
Data Transfer Node
Personal
Computer

Never run computations on the head node

(This can make it so no one else can login)

Always use compute nodes



slurm

workload manager

Simple Linux Utility for Resource Management

Simple Linux

workload manager

```
[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 ~]$ 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

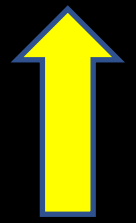
```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      1    mix hopper011
debug        up    4:00:00      1   idle hopper012
```

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
```

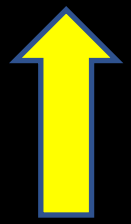
The name of the nodes in the partition

```
[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 ~]$
```



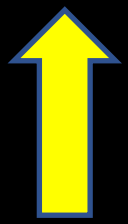
Running on the Head Node.

The head node’s name is “hopper”.

```
[vanilla@hopper ~]$ 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       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)
 4319   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4320   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4321   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4322   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4323   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4324   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4325   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4326   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4328   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4329   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4330   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4331   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4332   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4333   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4334   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4335   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4336   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
 4337   general      PRE erowland PD       0:00      2 (QOSMaxCpuPerUserLimit)
```

```
[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
 4319
 4320
 4321
 4322
 4323
 4324
 4325
 4326
 4328
 4329
 4330
 4331
 4332
 4333
 4334
 4335
 4336
 4337   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.

```
[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 ~]$ 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   11:4                07
              4313   general      PRE erowland                          003-004]
              5111   general    1stMA  mfricke  R   11:1                005-006]
              5025   general      c2n    jxzuo  R                       01
              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
```

Only one job running

```
[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 --mem 4G
--ntasks 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
```

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
--ntasks 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
```

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
--ntasks 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
```

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

Interactive vs Batch Mode

Interactive Mode

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

Batch Mode

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

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

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

```
[vanilla@hopper ~]$ tree workshops
workshops
├── crystallography
│   ├── data
│   │   └── thaumatin.pdb
│   └── slurm
│       ├── integrate_xray_images.slurm
│       └── solve_structure.slurm
├── escape
│   ├── fit_insar_data.ipynb
│   └── kml_functions.py
├── gaussian
│   ├── AcetylChloride.com
│   └── gaussian16_linda_acetylchloride.sh
├── intro_workshop
│   ├── code
│   │   ├── calc_pi_mpi.f90
│   │   ├── calcPiMPI.py
│   │   ├── calc_pi_serial.f90
│   │   ├── calcPiSerial.py

...

6 directories, 30 files
```

Run tree to see how the workshops directories are organized…

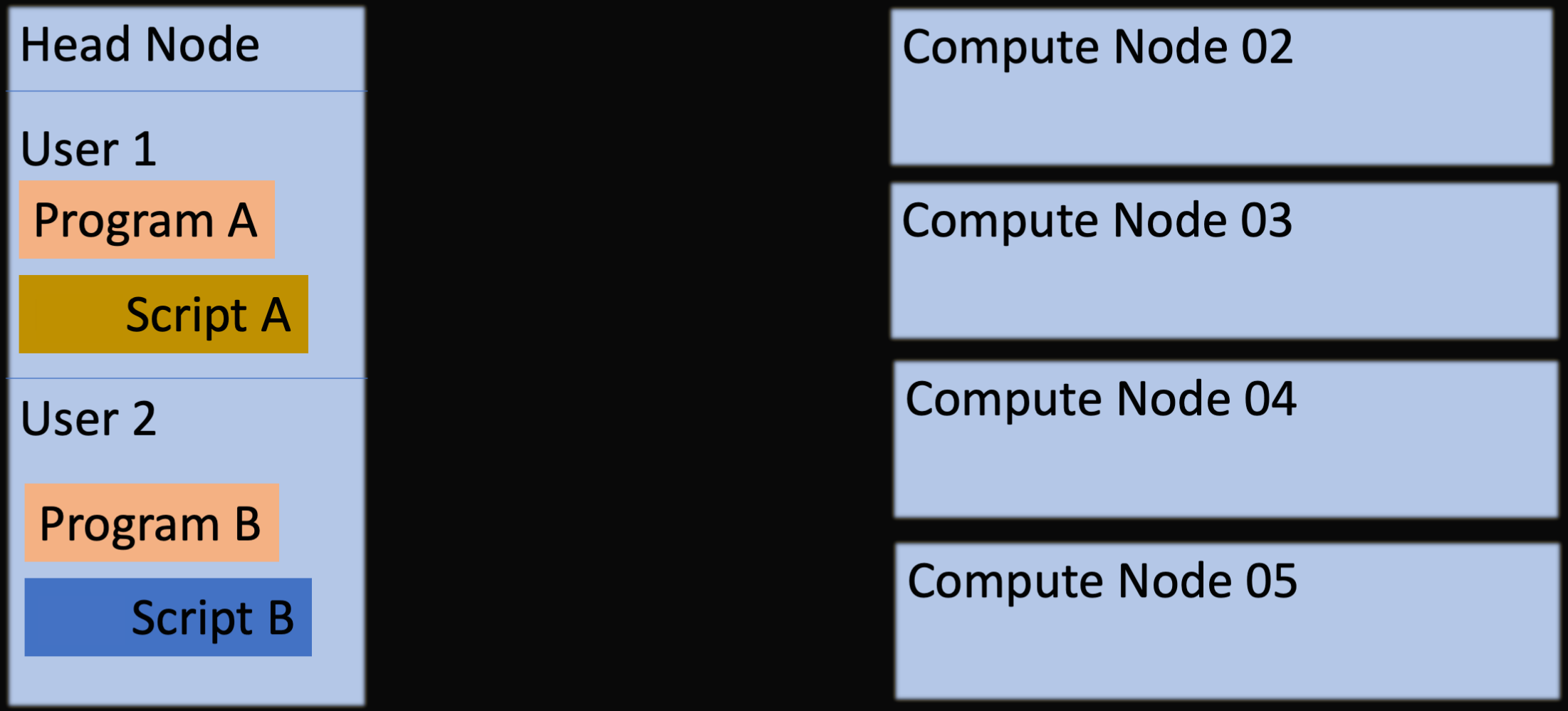
```
[vanilla@hopper intro_workshop]$ pwd
/users/vanilla/workshops/intro_workshop
[vanilla@hopper intro_workshop]$ cat slurm/workshop_example1.sh
#!/bin/bash
#SBATCH --partition debug
#SBATCH --ntasks 4
#SBATCH --time 00:05:00
#SBATCH --job-name ws_example
#SBATCH --mail-user your_username@unm.edu
#SBATCH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

srun hostname
```

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

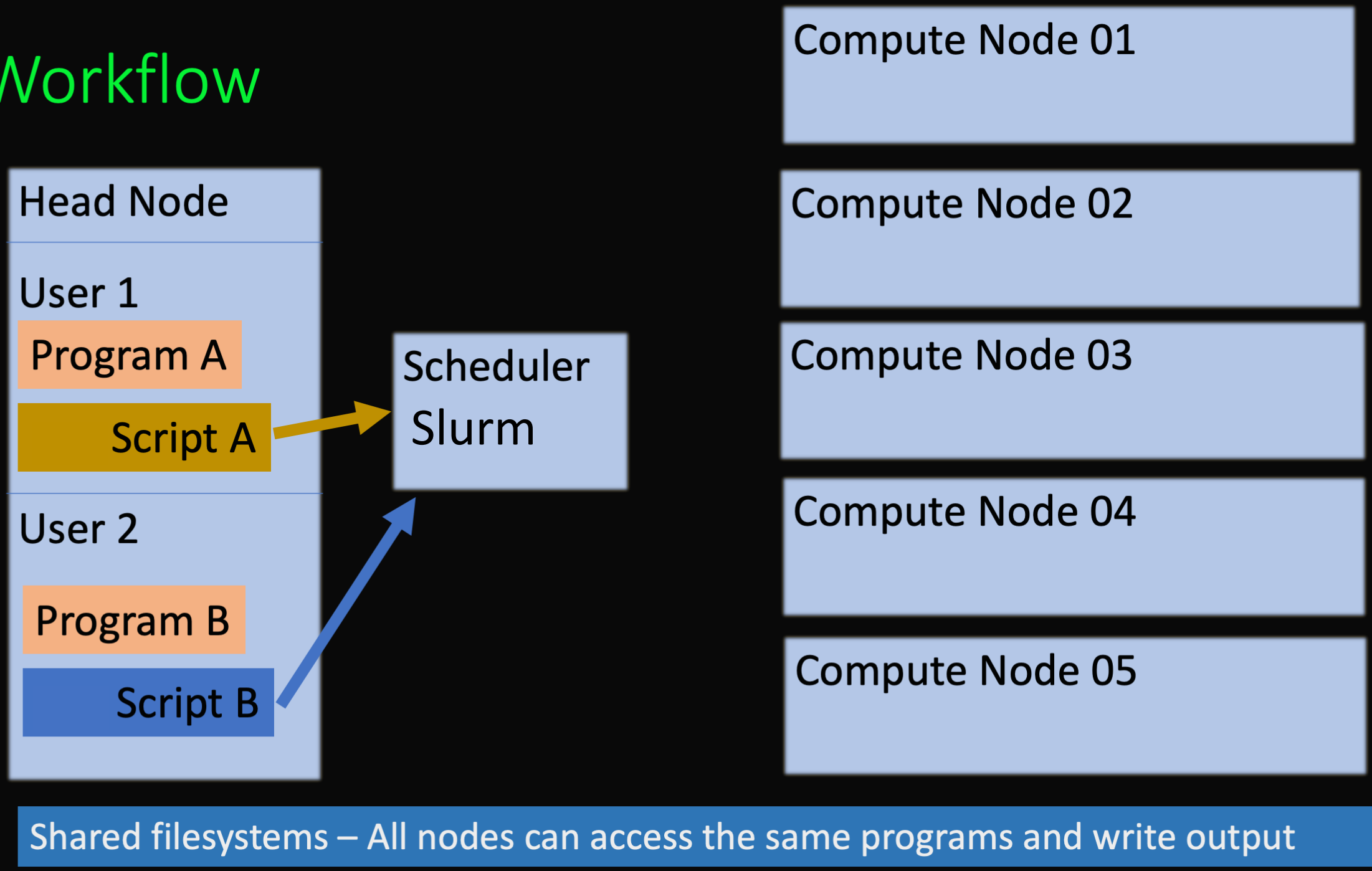
```
[vanilla@hopper intro_workshop]$ sbatch slurm/workshop_example1.sh
sbatch: Account not specified in script or ~/.default_slurm_account,
using latest project
Submitted batch job 5252
[vanilla@hopper intro_workshop]$
```

We submit our slurm shell script with the sbatch command.

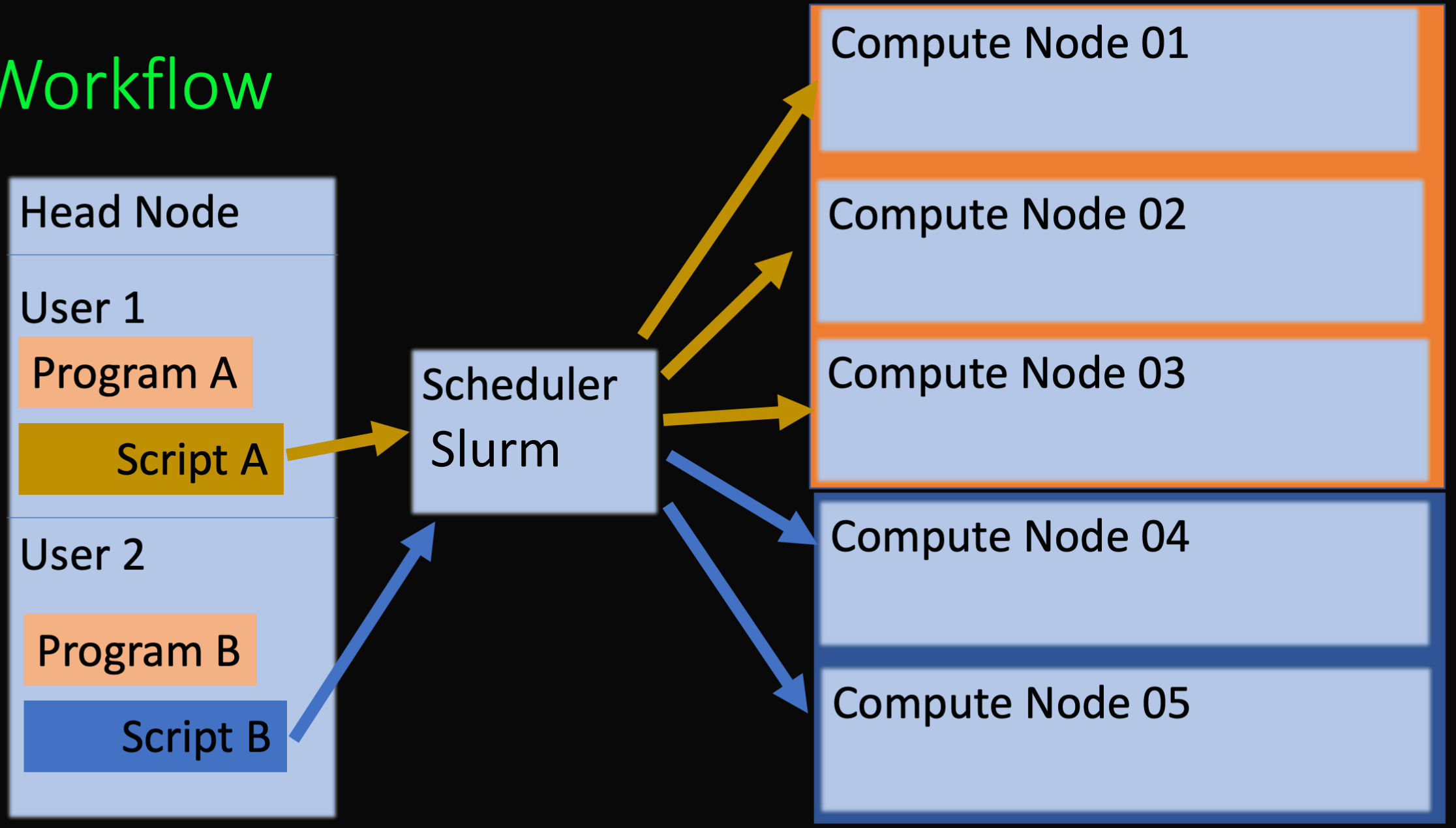
Workflow


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

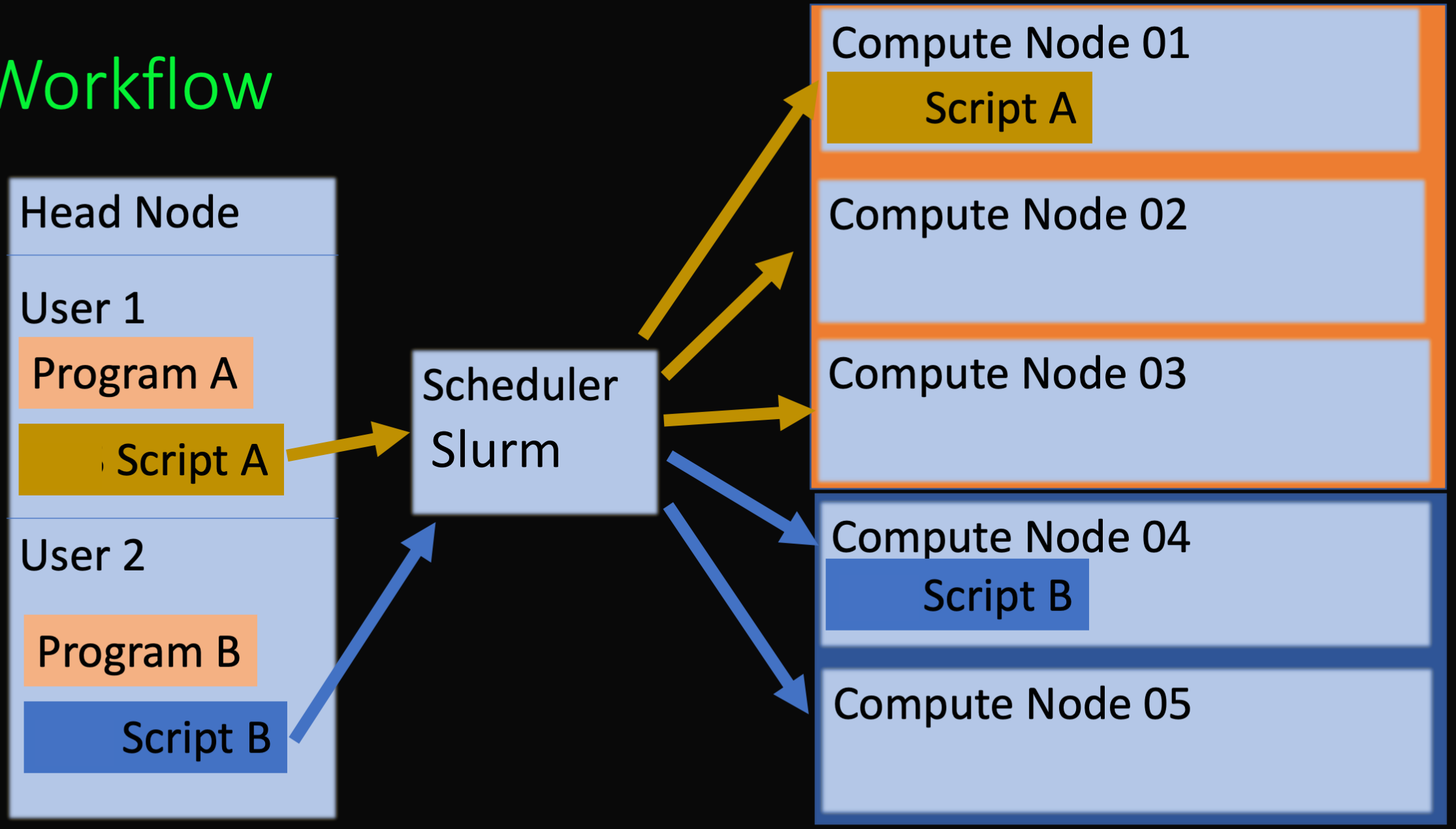

Head Node
User 1
Program A
Script A
User 2
Program B
Script B
Scheduler
Slurm
Compute Node 01
Compute Node 02
Compute Node 03
Compute Node 04
Compute Node 05
Shared filesystems – All nodes can access the same programs and write output

Workflow



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

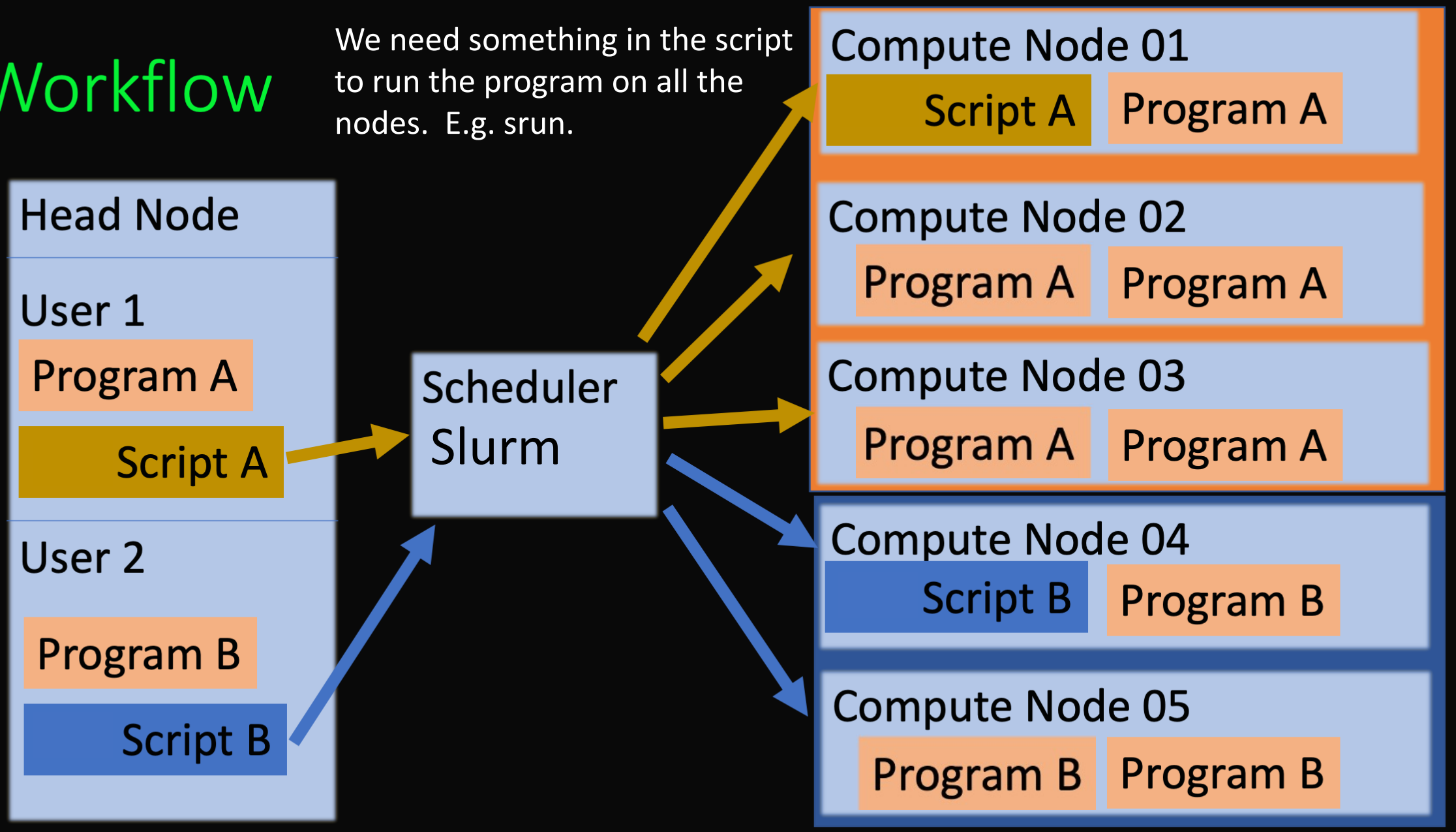
Workflow



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
```


```
[vanilla@hopper intro_workshop]$ cat slurm-5252.out
What do you see?
```

```
[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.

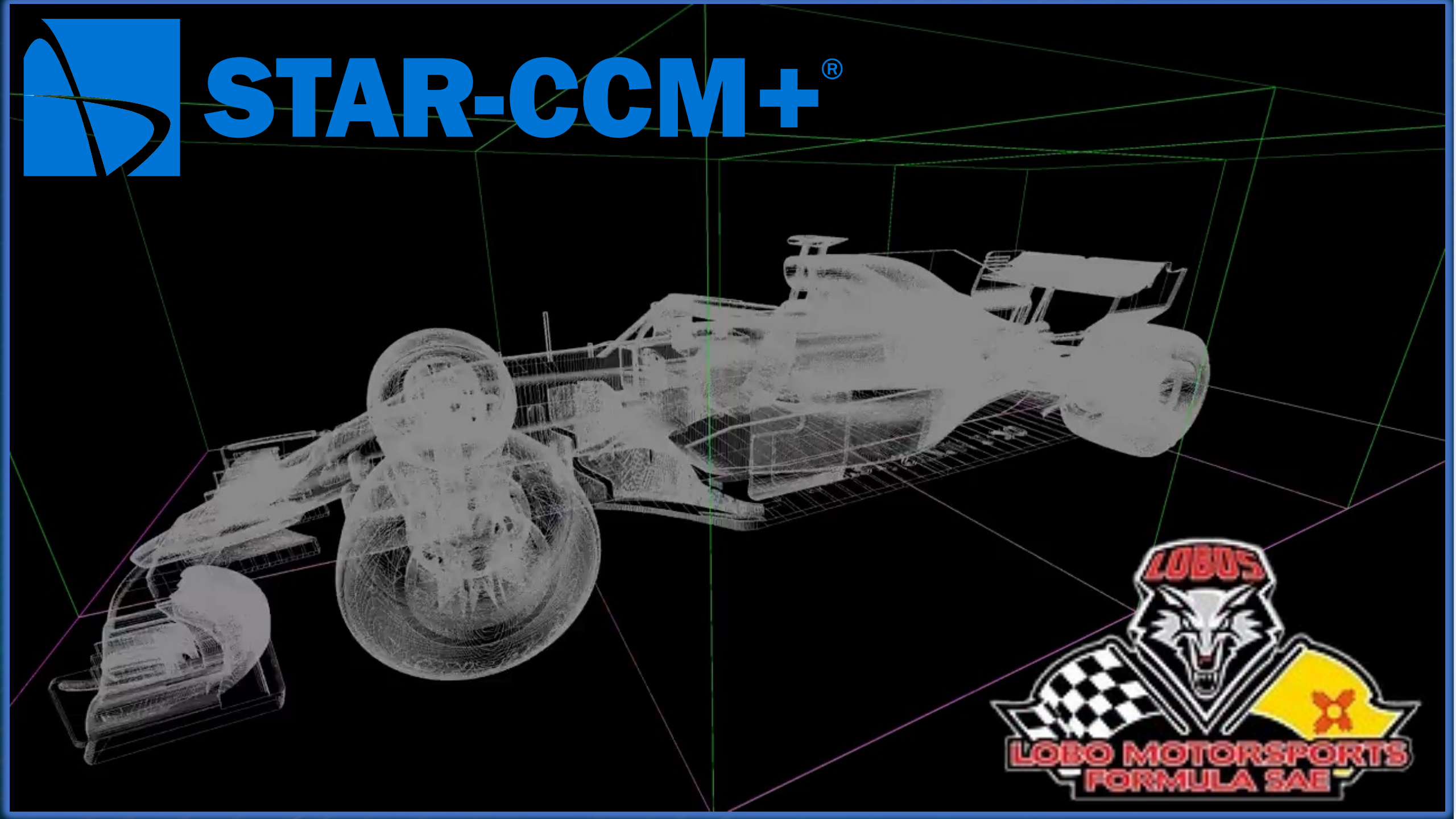
```
[vanilla@hopper intro_workshop]$ cat slurm-5252.out
To request GPUs, add --gpus-per-node X or --gpus X, where X is
the desired number of GPUs.
Job 3687623 running on hopper011
hopper011
hopper011
hopper011
hopper011
```

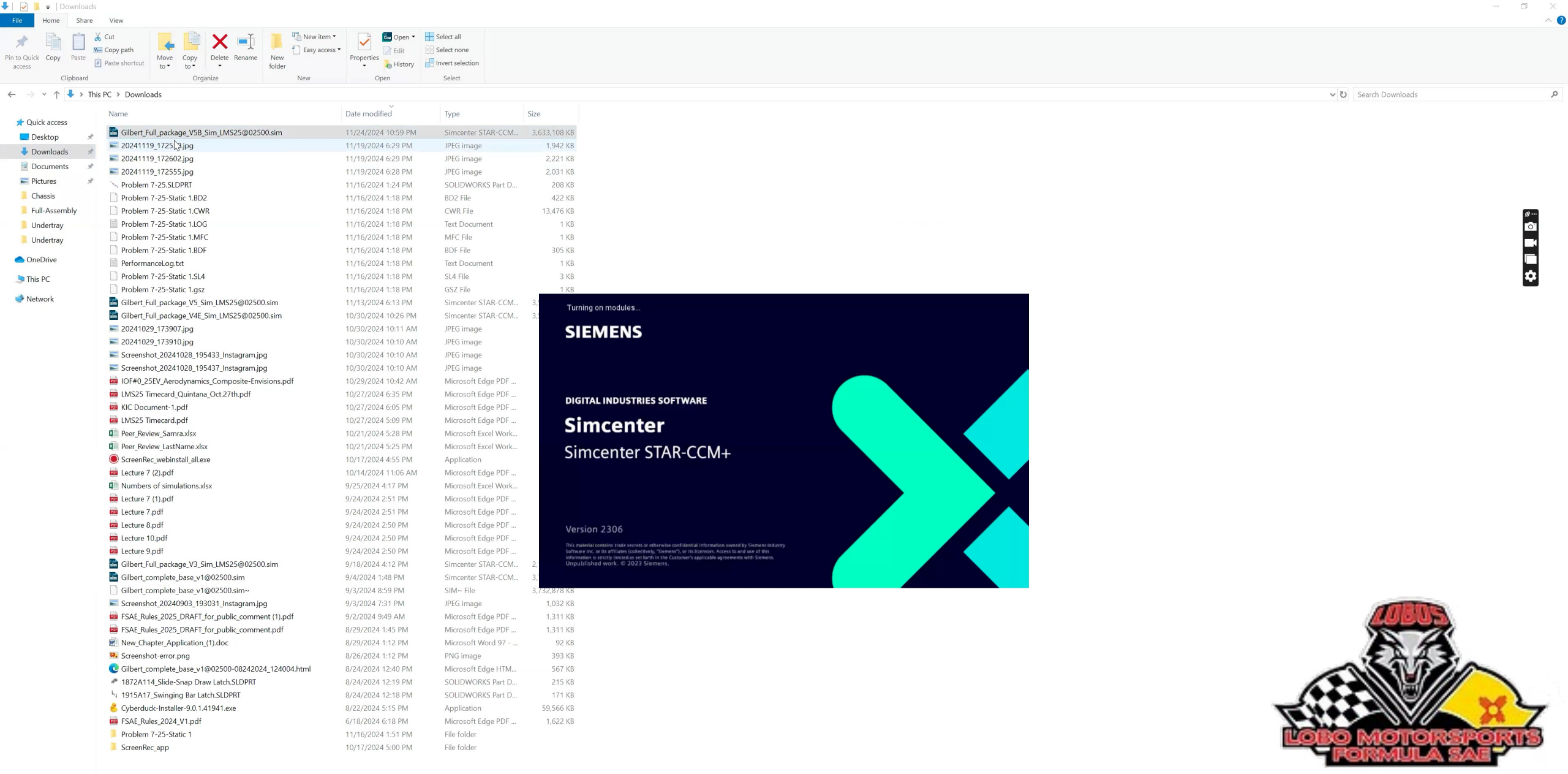
```
[vanilla@hopper intro_workshop]$ ls
code  data  pbs  slurm  slurm-5252.out
```

RACECAR CFD

LOBOS
LOBO MOTORSPORTS
FORMULA SAE

Aerodynamics Team, Amjad Samra (Top Center)


STAR-CCM+®
LOBOS
LOBO MOTORSPORTS
FORMULA SAE



Load racecar simulation, edit geometry, adjust settings, save changes, verify model updates.

```
[vanilla@hopper ~]$ ls /projects/shared/workshops/starccm/starccm_example/
L22_AE_FullCar_Cornering_v2@01400.sim  macroMeshAndRun.java  slurm-2746285.out
L22_AE_FullCar_Cornering_v2.sim        PBS_script            starccm_example.slurm
```

Copying data with scp…

Make good use of <tab> autocomplete for long paths ….

```
vanilla@hopper:~ $ cd /carc/scratch/users/vanulla/
vanilla@hopper:vanilla $ mkdir starccm_example
vanilla@hopper:vanilla $ cd starccm_example

rsync -P /projects/shared/workshops/starccm/starccm_example/L22_AE_FullCar_Cornering_v2.sim .
```

Source

Destination

```
    5,771,984,896  46%  359.46MB/s    0:00:18
```

Copy simulation into personal scratch drive…

StarCCM Script

```
vanilla@hopper: starccm_example $ cd ~/workshops/starccm/
vanilla@hopper: starccm $ cat starccm_debug_example.slurm
#!/bin/bash -l
#SBATCH --job-name starccm-example
#SBATCH --partition general
#SBATCH --time=4:00:00
#SBATCH --nodes 1
#SBATCH --ntasks-per-node 8
#SBATCH --mem=16GB
#SBATCH --mail-user your@email.address
#SBATCH --mail-type all

SIM_PATH=/carc/scratch/users/$USER/starccm_example/L22_AE_FullCar_Cornering_v2.sim

module load starccm
starccm+ -batchsystem slurm -batch $SIM_PATH -time -batch-report -licpath 1717@10.101.164.203
```

StarCCM Script

```
starccm+ -batchsystem slurm -batch $SIM_PATH -time -batch-report -licpath 1717@10.101.164.203
```

starccm+: This is the command to launch **STAR-CCM+**, Siemens' computational fluid dynamics (CFD) software.

-batchsystem slurm This tells STAR-CCM+ to use **SLURM** as the job scheduler for running batch jobs. It enables STAR-CCM+ to submit the job to the queue rather than running interactively.

-batch $SIM_PATH This runs STAR-CCM+ in **batch mode** using the simulation file or macro specified by $SIM_PATH. $SIM_PATH should point to a .sim file (or a .java macro). Batch mode is non-interactive—no GUI is opened.

-time This tells STAR-CCM+ to **measure and report the total wall-clock time** it takes to run the simulation.
•Useful for performance benchmarking and profiling.

-batch-report This generates a **batch report** automatically after the simulation completes. This typically includes solve performance, residuals, mesh statistics, etc. Output is usually a .pdf file summarising the run.

-licpath 1717@10.101.164.203 Specifies the **license server** to use for STAR-CCM+.
717 is the **port number**.
10.101.164.203 is the **IP address** of the license server.

Make good use of <tab> autocomplete for long paths ….

```
vanilla@hopper: starccm_example $ sbatch starccm_debug_example.slurm
sbatch: Account not specified in script or ~/.default_slurm_account, using
latest project
Submitted batch job 3700627
vanilla@hopper:~/workshops/starccm [master !x?]$ squeue --me
  JOBID     PARTITION   NAME      USER     ST    TIME  NODES NODELIST(REASON)
  3700627   general     starccm-  vanilla  R     1:13      1 hopper004
```

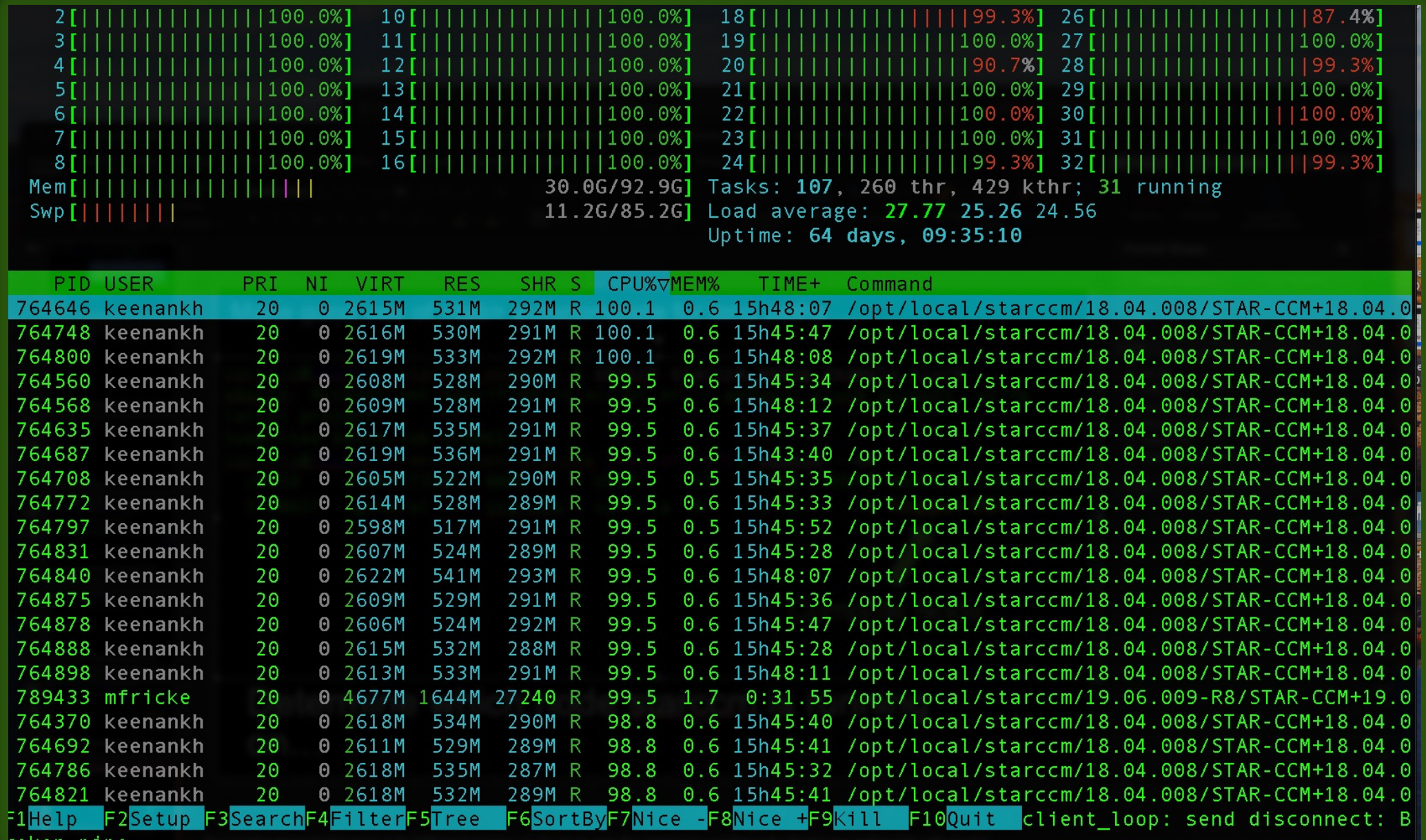
Compute Node

Determine which node starccm is running on…

```
vanilla@hopper: starccm_example $ ssh hopper004
vanilla@hopper004:~ $ htop
```

‘q’ to quit

Exit to logout of the compute node

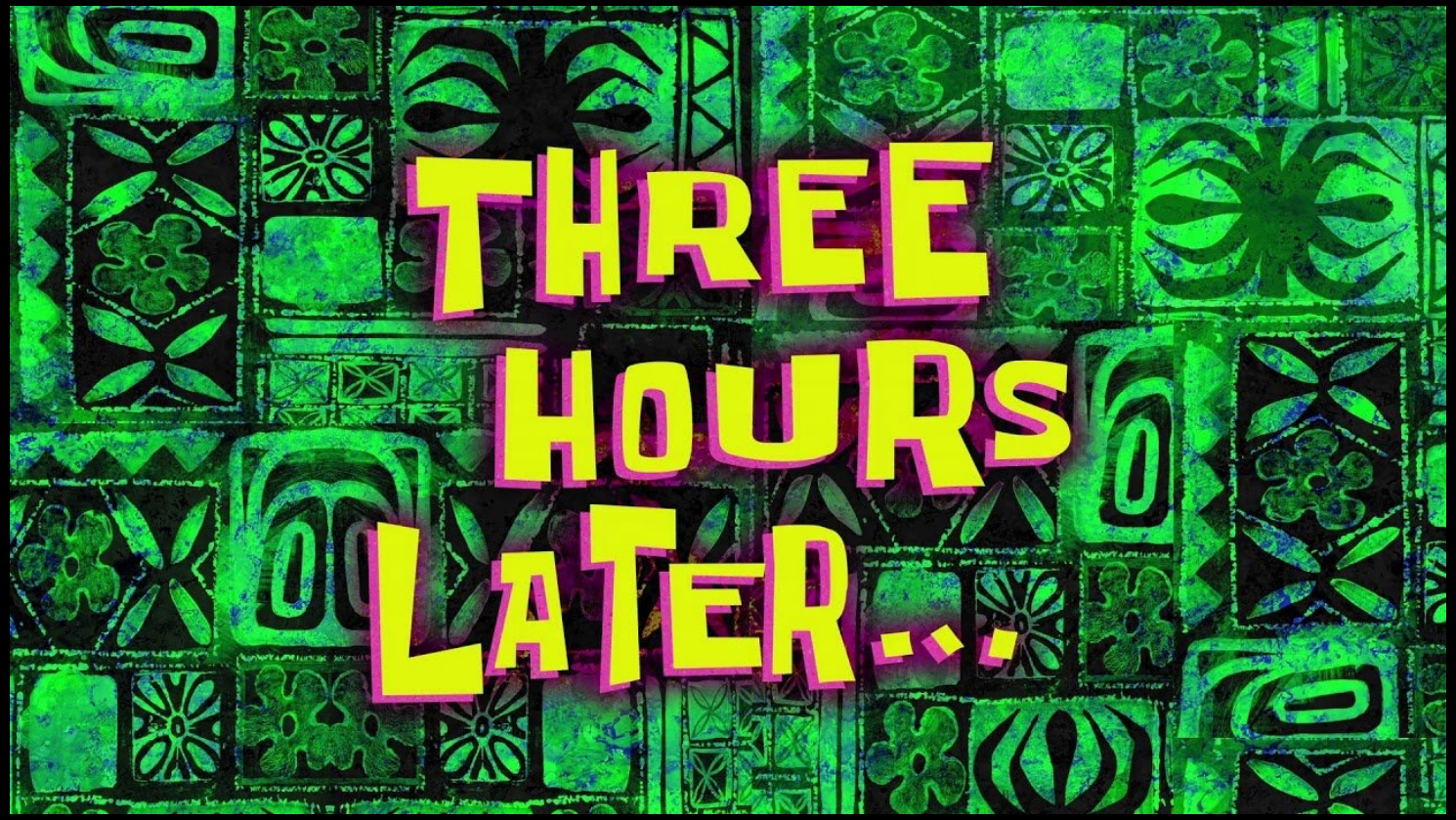


```
vanilla@hopper:starccm $ ls slurm-*
slurm-3700634.out
vanilla@hopper:starccm $ tail -f slurm-3700634.out

Starting local server: /opt/local/starccm/19.06.009-R8/STAR-CCM+19.06.009-
R8/star/bin/starccm+ -batchsystem slurm -licpath 1717@10.101.164.203 -server  -
xencoded-session
L2NhcmMvc2NyYXRjaC91c2Vycy9tZnJpY2tlL3N0YXJjY21fZXhhbXBsZS9MMjJfQUVfRnVsbENhcl9Db3JuZX
JpbmdfdjIuc2lt
Attention: The distribution of Simcenter STAR-CCM+ compiled with the GNU Compiler
           Collection (GCC) will no longer be available starting with the 2506
           release. The distribution compiled with Clang will replace it and is
           already available for download. Please contact your support engineer
           in case of questions.
MPI Distribution : Open MPI-4.1.5
Host 0 -- hopper004 -- Ranks 0-7
Process rank 0 hopper004 801332
Total number of processes : 8
Simcenter STAR-CCM+ 2410.0001 Build 19.06.009 (linux-x86_64-2.28/gnu11.4-r8)
License build date: 15 December 2023
This version of the code requires license version 2024.10 or greater.
Checking license file: 1717@10.101.164.203
```

```
vanilla@hopper:starccm $ ls slurm-*
slurm-3700634.out
vanilla@hopper:starccm $ tail -f slurm-3700634.out

Starting local server: /opt/local/starccm/19.06.009-R8/STAR-CCM+19.06.009-
R8/star/bin/starccm+ -batchsystem slurm -licpath 1717@10.101.164.203 -server  -
xencoded-session
L2NhcmMvc2NyYXRjaC91c2Vycy9tZnJpY2tlL3N0YXJjY21fZXhhbXBsZS9MMjJfQUVfRnVsbENhcl9Db3JuZX
JpbmdfdjIuc2lt
Attention: The distribution of Simcenter STAR-CCM+ compiled with the GNU Compiler
           Collection (GCC) will no longer be available starting with the 2506
           release. The distribution compiled with Clang will replace it and is
           already available for download. Please contact your support engineer
           in case of questions.
MPI Distribution : Open MPI-4.1.5
Host 0 -- hopper004 -- Ranks 0-7
Process rank 0 hopper004 801332
Total number of processes : 8
Simcenter STAR-CCM+ 2410.0001 Build 19.06.009 (linux-x86_64-2.28/gnu11.4-r8)
License build date: 15 December 2023
This version of the code requires license version 2024.10 or greater.
Checking license file: 1717@10.101.164.203
```

```
vanilla@hopper: starccm$ seff 3700634
Job ID: 3700634
Cluster: hopper
User/Group: vanilla/users
State: OUT_OF_MEMORY (exit code 0)
Nodes: 1
Cores per node: 8
CPU Utilized: 00:04:52
CPU Efficiency: 60.83% of 00:08:00 core-walltime
Job Wall-clock time: 00:01:00
Memory Utilized: 692.64 MB
Memory Efficiency: 4.23% of 16.00 GB
```

```
vanilla@hopper: starccm$ sbatch starccm_example.slurm
sbatch: Account not specified in script or ~/.default_slurm_account, using latest project
Submitted batch job 3700644
```



```
vanilla@hopper: starccm$ sbatch starccm_example.slurm
sbatch: Account not specified in script or ~/.default_slurm_account, using latest project
Submitted batch job 3700644

vanilla@hopper: starccm$ seff 3700635
Job ID: 3700635
Cluster: hopper
User/Group: vanilla/users
State: COMPLETED (exit code 0)
Nodes: 2
Cores per node: 32
CPU Utilized: 7-20:58:40
CPU Efficiency: 99.09% of 7-22:43:12 core-walltime
Job Wall-clock time: 02:58:48
Memory Utilized: 40.30 GB
Memory Efficiency: 22.39% of 180.00 GB
```

```
vanilla@hopper: starccm$ sbatch starccm_example.slurm
sbatch: Account not specified in script or ~/.default_slurm_account, using latest project
Submitted batch job 3700644

vanilla@hopper: starccm$ seff 3700635
Job ID: 3700635
Cluster: hopper
User/Group: vanilla/users
State: COMPLETED (exit code 0)
Nodes: 2
Cores per node: 32
CPU Utilized: 7-20:58:40
CPU Efficiency: 99.09% of 7-22:43:12 core-walltime
Job Wall-clock time: 02:58:48
Memory Utilized: 40.30 GB
Memory Efficiency: 22.39% of 180.00 GB
```

```
vanilla@hopper: starccm$ sbatch starccm_example.slurm
sbatch: Account not specified in script or ~/.default_slurm_account, using latest project
Submitted batch job 3700644

vanilla@hopper: starccm$ seff 3700635
Job ID: 3700635
Cluster: hopper
User/Group: vanilla/users
State: COMPLETED (exit code 0)
Nodes: 2
Cores per node: 32
CPU Utilized: 7-20:58:40
CPU Efficiency: 99.09% of 7-22:43:12 core-walltime
Job Wall-clock time: 02:58:48
Memory Utilized: 40.30 GB
Memory Efficiency: 22.39% of 180.00 GB
```

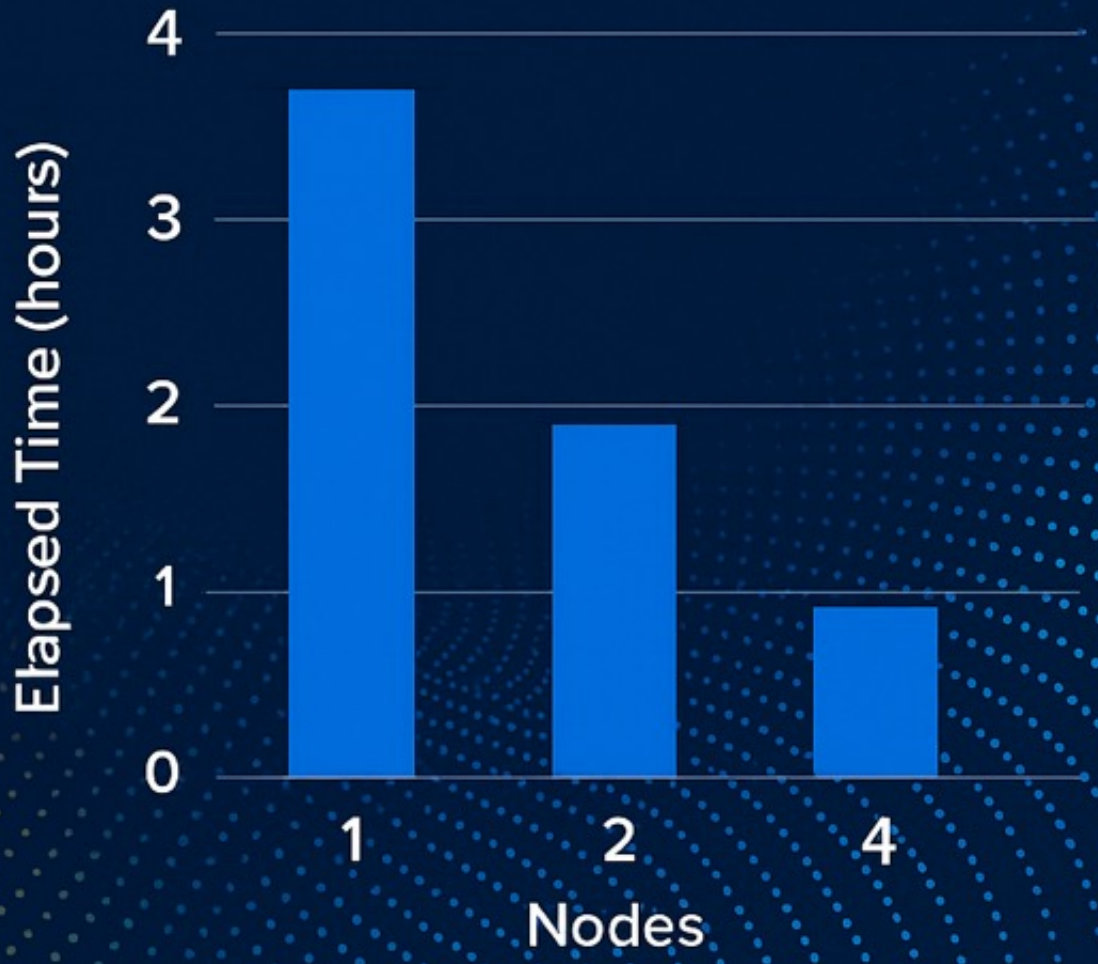
```
vanilla@hopper: starccm$ sbatch starccm_example.slurm
sbatch: Account not specified in script or ~/.default_slurm_account, using latest project
Submitted batch job 3700644

vanilla@hopper: starccm$ seff 3700635
Job ID: 3700635
Cluster: hopper
User/Group: vanilla/users
State: COMPLETED (exit code 0)
Nodes: 2
Cores per node: 32
CPU Utilized: 7-20:58:40
CPU Efficiency: 99.09% of 7-22:43:12 core-walltime
Job Wall-clock time: 02:58:48
Memory Utilized: 40.30 GB
Memory Efficiency: 22.39% of 180.00 GB
```

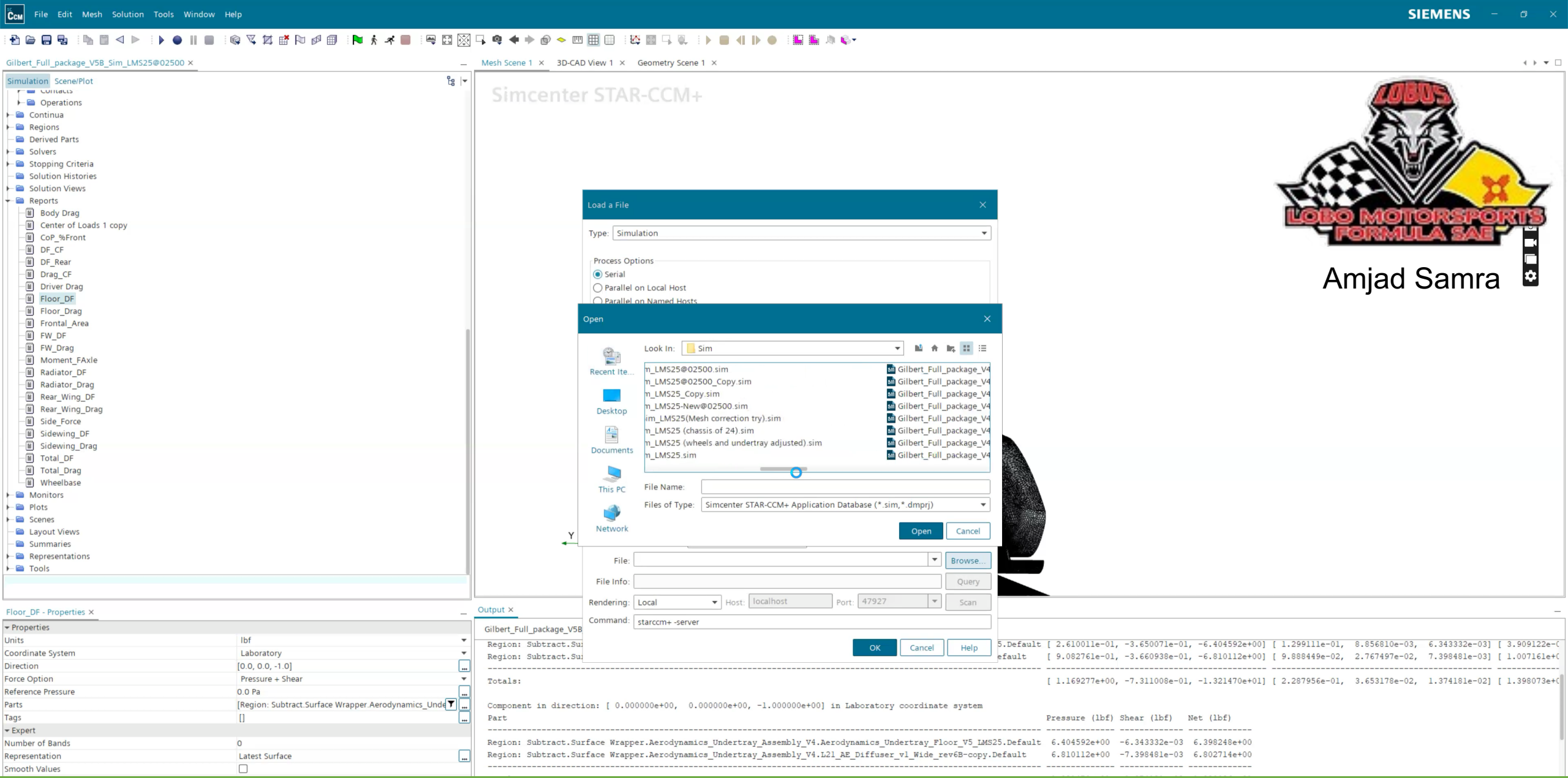
```
vanilla@hopper: starccm$ sbatch starccm_example.slurm
sbatch: Account not specified in script or ~/.default_slurm_account, using latest project
Submitted batch job 3700644

vanilla@hopper: starccm$ seff 3700635
Job ID: 3700635
Cluster: hopper
User/Group: vanilla/users
State: COMPLETED (exit code 0)
Nodes: 2
Cores per node: 32
CPU Utilized: 7-20:58:40
CPU Efficiency: 99.09% of 7-22:43:12 core-walltime
Job Wall-clock time: 02:58:48
Memory Utilized: 40.30 GB
Memory Efficiency: 22.39% of 180.00 GB
```

Timing on Hopper

- On Wheeler 8 cores: about 14 hours
- 12 GB Racecar Model Input
- Hopper 1 node 32 tasks – about 4 hours

  Hopper 4 nodes 128 tasks **1 hour**

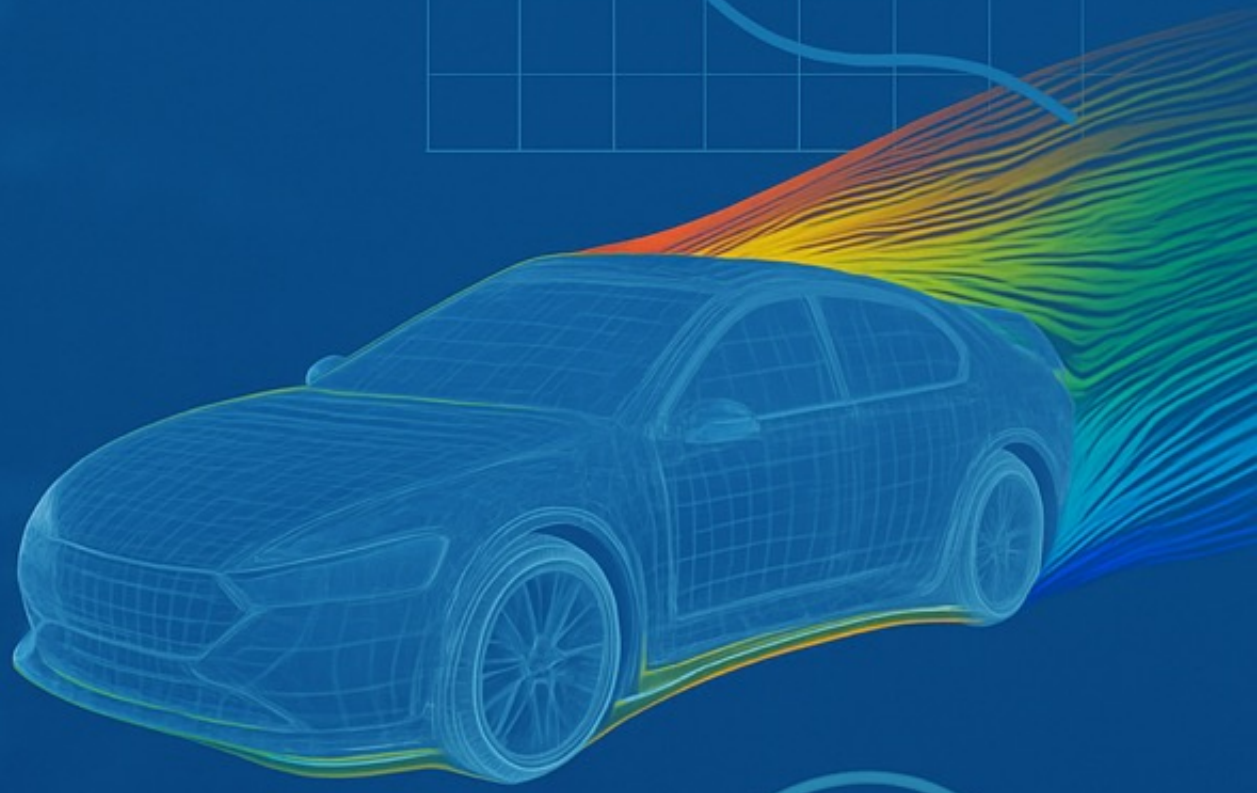




Reviewing Scalar Fields and Flow Features Calculated on Hopper

Colour-mapped results (e.g., velocity or pressure contours) are shown.

TIME FOR A BREAK



End of Part 1