

Intermediate Level Introduction to Computing at CARC

1 hour version with Neuromorphic Computing

Matthew Fricke

GPT Astra-6 assisted with developing notebooks, python code, and Slurm scripts.

Version 0.1

A series of five thick, yellow, curved lines arranged in a diagonal pattern from the bottom right towards the center of the slide.

Why neuromorphic computing?

Potential advantages for sensing and intelligent systems

Energy efficiency

Exploit sparse activity and reduce data movement.

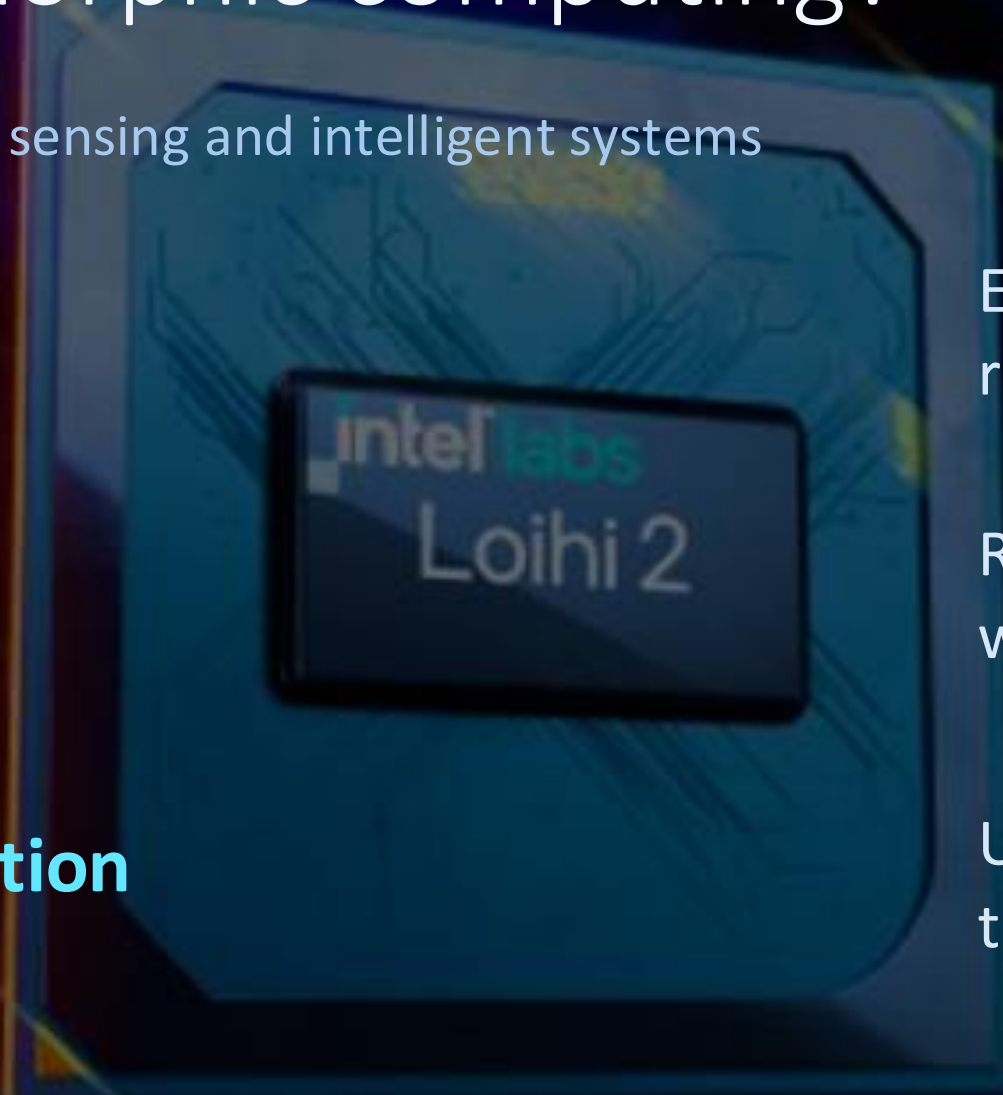
Fast responses

React to incoming events with low latency.

Temporal information

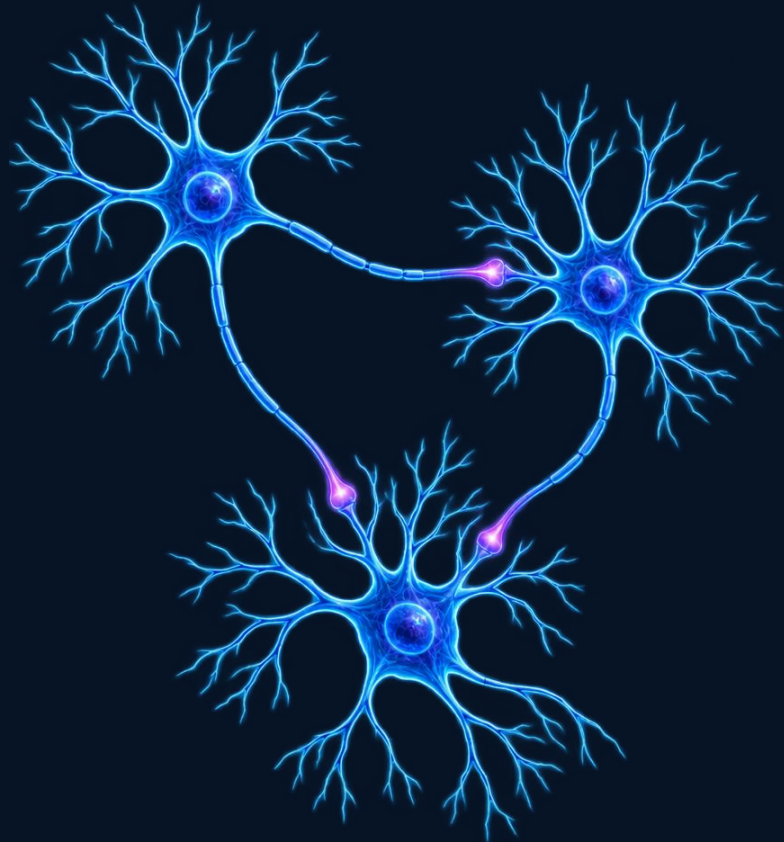
Use timing and internal state to process changing signals.

Today: test spiking-network ideas on CARC.

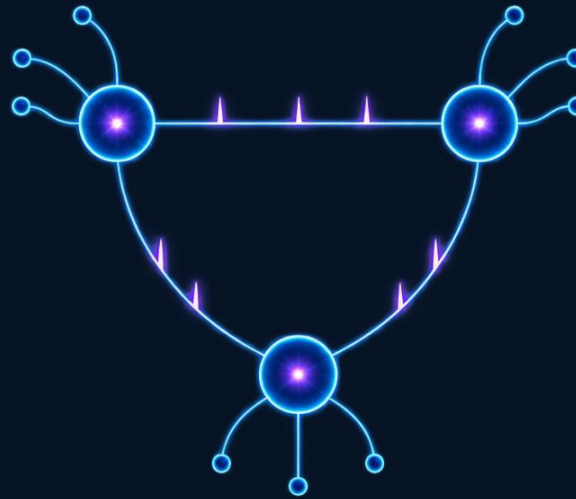


Neuromorphic computing: inspiration from the brain

Biological neurons



Spiking model



Spikes

Neurons communicate through brief electrical events.

Synapses

Connections influence how strongly neurons affect one another.

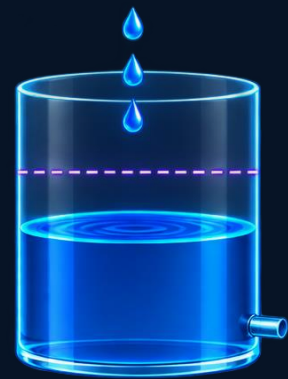
Parallel activity

Many connected neurons process information together.

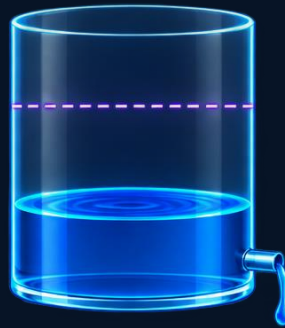
How a spiking neuron works

Integration, leakage, threshold, reset

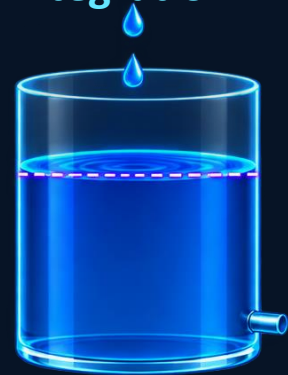
A leaky bucket analogy



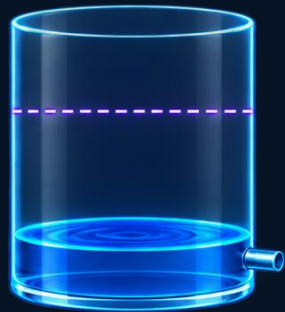
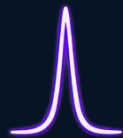
Integration



Leakage



Threshold



Reset

Integration

Inputs accumulate in the membrane potential.

Leakage

Without input, the membrane potential decays toward rest.

Threshold

Crossing a threshold triggers an output spike.

Reset

Our example subtracts the threshold after each spike.

Example 1: four time steps and two spikes

Start at 0 Input = 0.75 $\beta = 0.5$ Threshold = 1.0

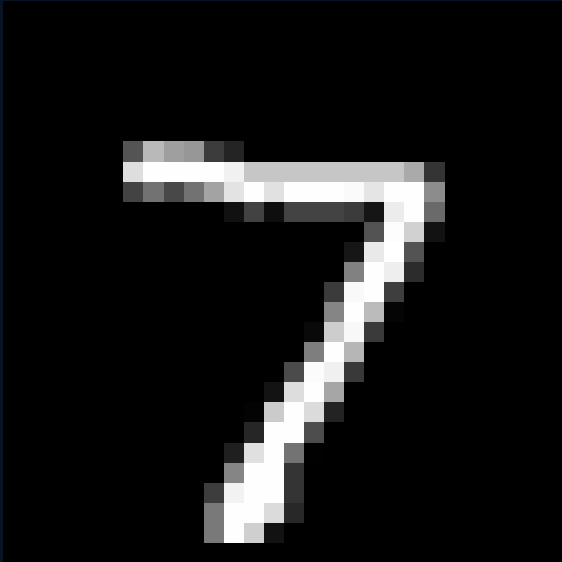
Step	Leak + input	Before reset	Spike	After reset
1	$0.5 \times 0 + 0.75$	0.75	0	0.75
2	$0.5 \times 0.75 + 0.75$	1.125	1	0.125
3	$0.5 \times 0.125 + 0.75$	0.8125	0	0.8125
4	$0.5 \times 0.8125 + 0.75$	1.15625	1	0.15625

Spikes: 0, 1, 0, 1

A spike subtracts 1.0.
The remainder carries into the next step.

Example 2: Our image classifier

Recognizing handwritten digits with a spiking neural network



28 × 28 grayscale image

Input

Pixel brightness supplies input current.

Spiking network

128 hidden neurons process the image for 15 simulated time steps.

Prediction

10 output neurons represent digits 0–9.
The highest spike count selects the digit.

In Jupyter: train on MNIST, inspect spikes, then try a digit image.

Where HPC helps

More compute and memory for spiking-network research

Training

Run longer training jobs
on allocated CPUs or GPUs.

Larger models

Access more memory for
networks and longer sequences.

Repeated experiments

Run parameter sweeps and
random seeds as separate jobs.

Slurm allocates resources and schedules the runs.

Today's tools

snnTorch on conventional CPUs and GPUs

snnTorch

Spiking neuron models and learning with surrogate gradients.

PyTorch

Tensor operations and gradients on CPUs or CUDA GPUs.

Slurm

Allocates Easley resources and launches our jobs.

Today's examples simulate spiking networks on Easley CPUs.

Goals

- 1) SLURM scheduler literacy
- 2) Example Gaussian
- We wont cover file transfer, storage systems, module system, or conda. (These are all covered in depth in the video tutorials)

Logging into CARC



First login to the Linux **workstation** in front of you.

Use the password you chose when you created your account.

If you forgot your password go to:

<https://mokey.alliance.unm.edu/auth/forgotpw>

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

Read all updates

Welcome to the Center for Advanced Research Computing

The UNM Center for Advanced Research Computing is the hub of computational research at UNM and one of the largest computing centers in the State of New Mexico. It is an interdisciplinary community that uses computational resources to create new research insights. The goal is to lead and grow the computational research community at UNM..

CARC provides not just the computing resources but also the expertise and support to help the university’s researchers. This service is available to faculty, staff, and student researchers free of charge through support from the UNM Office of the Vice President for Research.



- Getting started at CARC
- Good Neighbor Use Policy
- Video: Introduction to Computing at CARC tutorials

Get started at CARC

Create a help ticket

Downtime notices



ever...
New users - Video: Introduction to Computing at CARC tutorials

UNM launches search for the next director of the CARC

UNM awarded prestigious grant to lead national research on high-performance computing

Genome sequencing reveals a newly identified virus

Undergraduate researcher leverages CARC for molecular modeling breakthrough



CARC Video Tutorials: Logging in

Home

Shorts

Subscriptions

You

YouTube

Search

Sign in



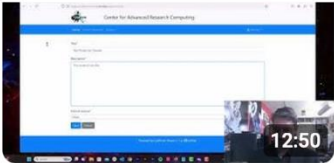
QuickBytes

by UNMCARC

Playlist • 19 videos • 8,136 views

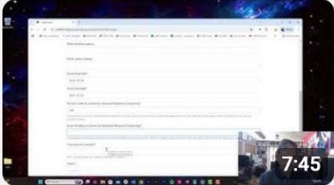
▶ Play all

3



UNMCARC • 469 views • 1 year ago

4



Annual Project Review
UNMCARC • 205 views • 1 year ago

5



Logging into CARC Systems
UNMCARC • 1.3K views • 5 years ago

6



Slurm Job Scheduler
UNMCARC • 700 views • 10 months ago

7



Storage Systems
UNMCARC • 674 views • 4 years ago



CARC Video Tutorials: Slurm Scheduler

Home

Shorts

Subscriptions

You

YouTube

Search

Q

Sign in



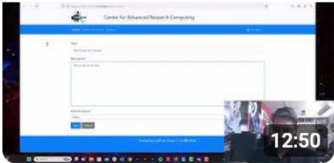
QuickBytes

by UNMCARC

Playlist • 19 videos • 8,136 views

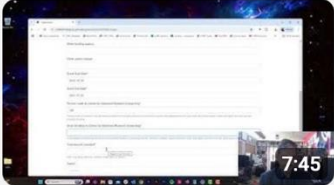
▶ Play all

3



UNMCARC • 469 views • 1 year ago

4



Annual Project Review
UNMCARC • 205 views • 1 year ago

5



Logging into CARC Systems
UNMCARC • 1.3K views • 5 years ago

6



Slurm Job Scheduler
UNMCARC • 700 views • 10 months ago

7



Storage Systems
UNMCARC • 674 views • 4 years ago



Logging into Easley



```
ssh vanilla@easley.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 Easley

▶ 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>

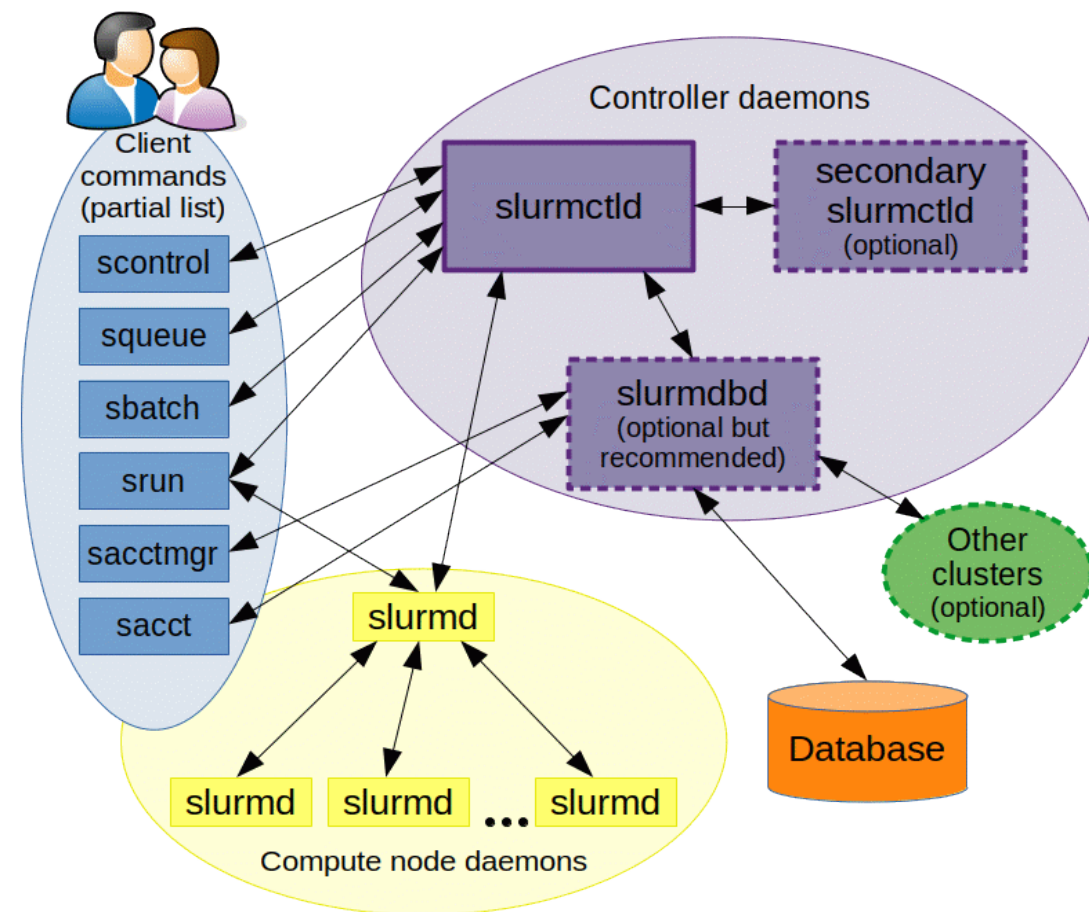
▶ Need help? Email: help@carc.unm.edu

▶ Please include an acknowledgement of CARC in any publications that make use of computations performed here.

Last login: Tue Feb 10 17:03:07 2026 from 50.188.169.22

Warning: Your home directory (~) usage is 187G*/100G. Please reduce your usage. Use the `ncdu` command to see details.

Simple Linux Utility for Resource Management





VOTED **#1** SOFT DRINK OF THE 31ST CENTURY!



[vanilla@easley ~]\$ qgrok

queues	free	busy	offline	jobs	nodes	CPUs	idle_CPUS	GPUs	CPUs/node	memory/node	time_limit	CPU_limit	GPU_limit	RAM_limit
general	3	31	14	17	48	3072	427	0	64	250G	2-00:00:00	512	0	2059472M
bigmem	2	0	0	0	2	128	128	0	64	1.9T	2-00:00:00	64	0	2063765M
h100	0	4	0	1	4	256	63	8	64	1006G	2-00:00:00	64	2	1031578M
l40s	0	3	0	0	3	192	189	12	64	250G	2-00:00:00	96	3	171624M
interactive	0	7	0	0	7	448	252	20	64	250G-1006G	4:00:00	64	1	
debug	5	38	14	0	57	3648	807	20	64	250G-1006G	1:00:00	128	2	
scavenger	9	40	14	3	63	4032	1186	44	64	250G-1006G	2-00:00:00	4032	44	
liulab	4	2	0	1	6	384	379	24	64	250G	7-00:00:00	384	24	
totals:	5	38	14	18	57	3648	1186	24	64					

General:

For longer jobs

Debug:

For short small jobs to test code

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

queues	free	busy	offline	jobs	nodes	CPUs	idle_CPUS	GPUs	CPUs/node	memory/node	time_limit	CPU_limit	GPU_limit	RAM_limit
general	3	31	14	17	48	3072	427	0	64	250G	2-00:00:00	512	0	2059472M
bigmem	2	0	0	0	2	128	128	0	64	1.9T	2-00:00:00	64	0	2063765M
h100	0	4	0	1	4	256	63	8	64	1006G	2-00:00:00	64	2	1031578M
l40s	0	3	0	0	3	192	189	12	64	250G	2-00:00:00	96	3	171624M
interactive	0	7	0	0	7	448	252	20	64	250G-1006G	4:00:00	64	1	
debug	5	38	14	0	57	3648	807	20	64	250G-1006G	1:00:00	128	2	
scavenger	9	40	14	3	63	4032	1186	44	64	250G-1006G	2-00:00:00	4032	44	
liulab	4	2	0	1	6	384	379	24	64	250G	7-00:00:00	384	24	
totals:	5	38	14	18	57	3648	1186	44	64	250G-1006G				

“scavenger” partition

- Allows you to use idle nodes from any partition even special hardware
- May be interrupted at any time someone decides to use that special hardware

[vanilla@easley ~]\$ **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@easley ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug      up    1:00:00    2  idle easley[011-012]
```

sinfo reports information about
partitions

```
[vanilla@easley ~]$ sinfo --partition debug  
PARTITION AVAIL TIMELIMIT  NODES  STATE NODELIST  
debug      up    1:00:00    2  idle easley[011-012]
```

The debug queues are intended
for testing your programs.

```
[vanilla@easley ~]$ sinfo --partition debug
```

```
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
```

```
debug      up    1:00:00    2 idle easley[011-012]
```



Name

```
[vanilla@easley ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug      up      1:00:00    2  idle easley[011-012]
```



You can run a “job” for up to 4 hrs.

```
[vanilla@easley ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug      up    1:00:00    2  idle easley[011-012]
```



There are two nodes in this partition.

```
[vanilla@easley ~]$ sinfo --partition debug
PARTITION AVAIL TIMELIMIT  NODES  STATE NODELIST
debug      up    1:00:00    2  idle easley[011-012]
```



The names of the nodes in the
partition

```
[vanilla@easley ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug      up    1:00:00    2  idle easley[011-012]
```



The names of the nodes in the
partition

```
[vanilla@easley ~]$ sinfo --partition general
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
general*   up 2-00:00:00    9  alloc easley[001-009]
general*   up 2-00:00:00    1  idle easley010
```



Easley001 through 009 are running jobs.

Easley010 is waiting to be used.

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



Running on the Head Node.
The head node's name is "easley".

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

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

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

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

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

```
[vanilla@easley ~]$ 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@easley ~]\$ sinfo --all

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	NODELIST
general*	up	2-00:00:00	5	down*	easley[002,023,027,037,048]
general*	up	2-00:00:00	8	drain	easley[005,007-008,017,035-036,039-040]
general*	up	2-00:00:00	3	idle	easley[009-010,042]
general*	up	2-00:00:00	1	down	easley016
bigmem	up	2-00:00:00	2	idle	easley[049-050]
h100	up	2-00:00:00	1	mix	easley054
h100	up	2-00:00:00	3	alloc	easley[051-053]
l40s	up	2-00:00:00	3	mix	easley[055-057]
interactive	up	4:00:00	4	mix	easley[054-057]
interactive	up	4:00:00	3	alloc	easley[051-053]
debug	up	1:00:00	5	idle	easley[009-010,042,049-050]
debug	up	1:00:00	1	down	easley016
scavenger	up	2-00:00:00	5	down*	easley[002,023,027,037,048]
scavenger	up	2-00:00:00	8	drain	easley[005,007-008,017,035-036,039-040]
scavenger	up	2-00:00:00	17	mix	easley[001,003-004,006,011-013,018,024-026,054-059]
scavenger	up	2-00:00:00	23	alloc	easley[014-015,019-022,028-034,038,041,043-047,051-053]
scavenger	up	2-00:00:00	9	idle	easley[009-010,042,049-050,060-063]
scavenger	up	2-00:00:00	1	down	easley016

[vanilla@easley ~]\$ sinfo --all

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	NODELIST
general*	up	2-00:00:00	5	down*	easley[002,023,027,037,048]
general*	up	2-00:00:00	8	drain	easley[005,007-008,017,035-036,039-040]
general*	up	2-00:00:00	3	idle	easley[009-010,042]
general*	up	2-00:00:00	1	down	easley016
bigmem	up	2-00:00:00	2	idle	easley[049-050]
h100	up	2-00:00:00	1	mix	easley054
h100	up	2-00:00:00	3	alloc	easley[051-053]
l40s	up	2-00:00:00	3	mix	easley[055-057]
interactive	up	4:00:00	4	mix	easley[054-057]
interactive	up	4:00:00	3	alloc	easley[051-053]
debug	up	1:00:00	5	idle	easley[009-010,042,049-050]
debug	up	1:00:00	1	down	easley016
scavenger	up	2-00:00:00	5	down*	easley[002,023,027,037,048]
scavenger	up	2-00:00:00	8	drain	easley[005,007-008,017,035-036,039-040]
scavenger	up	2-00:00:00	17	mix	easley[001,003-004,006,011-013,018,024-026,054-059]
scavenger	up	2-00:00:00	23	alloc	easley[014-015,019-022,028-034,038,041,043-047,051-053]
scavenger	up	2-00:00:00	9	idle	easley[009-010,042,049-050,060-063]
scavenger	up	2-00:00:00	1	down	easley016

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



Tell slurm to run a program
on a compute node...

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



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

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



The program
to run.

```
[vanilla@easley ~]$ 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.
```

```
easley011
```

```
[vanilla@easley ~]$ queue
```

```
[vanilla@easley ~]$ queue
```

(- ' " . 02' 2' - , , + # 31#0 12 2' +

□□□□ ?=F=J 9D . 0# =J GOD9F< . " ← □□□□

□□□□ ?=F=J 9D . 0# =J GOD9F< . " □□□□

□□□□ ?=F=J 9D . 0# =J GOD9F< . " □□□□

□□□□ ?=F=J 9D . 0# =J GOD9F< . " □□□□

**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.

[illegible]

```
[vanilla@easley ~]$ queue
```

□□□□	?=F=J9D	. 0# =J GOD9F< . "	□ǫ □□	□ ɹ/ - 1+9P! HM =J3K=J * AEALɹ
□□□□	?=F=J9D	. 0# =J GOD9F< . "	□ǫ □□	□ ɹ/ - 1+9P! HM =J3K=J * AEALɹ
□□□□	?=F=J9D	. 0# =J GOD9F< . "	□ǫ □□	□ ɹ/ - 1+9P! HM =J3K=J * AEALɹ
□□□□	?=F=J9D	. 0# =J GOD9F< . "	□ǫ □□	□ ɹ/ - 1+9P! HM =J3K=J * AEALɹ
□□□□	?=F=J9D	. 0# =J GOD9F< . "	□ǫ □□	□ ɹ/ - 1+9P! HM =J3K=J * AEALɹ

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

0000 (-F-J9D : U# -JGD9P< . ° □□ □ ٤/- 1+9P! HM =J3K=J * AEAL٥

```
[vanilla@easley ~]$ squeue -t R --all
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
4405	condo	2ndMA	mfricke	R	1-07:48:30	6	easley[031-036]
5208	condo	NN	kgu	R	5:48:49	1	easley015
5210	condo	NN	kgu	R	6:30:13	1	easley014
5209	condo	NN	kgu	R	6:31:13	1	easley013
5206	condo	NN	kgu	R	6:32:13	1	easley051
5207	condo	NN	kgu	R	6:32:13	1	easley052
5205	condo	NN	kgu	R	6:32:43	1	easley028
4595	cup-ecs	golConfi	aalasand	R	2-06:51:59	1	easley050
4594	cup-ecs	golConfi	aalasand	R	2-06:52:03	1	easley049
5120	general	jupyterh	jacobm	R	11:45:47	1	easley007
4313	general	PRE	erowland	R	1:17:29	2	easley[003-004]
5111	general	1stMA	mfricke	R	11:15:28	2	easley[005-006]
5025	general	c2n	jxzuo	R	1:50	1	easley001
5024	general	c2n	jxzuo	R	31:28	1	easley002
5203	general	NN	kgu	R	6:37:50	1	easley009
5201	general	NN	kgu	R	6:38:14	1	easley008
4390	tc	UCsTpCyd	lepluart	R	2-15:18:18	3	easley[018-020]
5198	tc	NN	kgu	R	6:40:19	1	easley030
5196	tc	NN	kgu	R	6:40:31	1	easley029

```
[vanilla@easley ~]$ 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.
```

```
easley011
```

```
easley011
```

```
[vanilla@easley ~]$ 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.
```

```
easley011
```

```
easley011
```

You ran two **copies** of your program.

ntasks is the number of copies to run.

```
[vanilla@easley ~]$ srun --partition debug --ntasks 8 hostname
```

```
srun: Account not specified in script or ~/.default_slurm_account, using latest project
```

```
easley011
```

```
easley011
```

```
easley011
```

```
You have not been allocated GPUs. To request GPUs, use the -G option in your submission script.
```

```
easley011
```

```
easley011
```

```
easley011
```

```
easley011
```

```
easley011
```

You ran eight **copies** of your program.

ntasks is the number of copies to run.

```
[vanilla@easley ~]$ srun --partition debug --ntasks 8 hostname
```

```
srun: Account not specified in script or ~/.default_slurm_account, using latest project
```

```
easley011
```

```
easley011
```

```
easley011
```

```
You have not been allocated GPUs. To request GPUs, use the -G option in your submission script.
```

```
easley011
```

```
easley011
```

```
easley011
```

```
easley011
```

```
easley011
```

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@easley ~]$ 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.
```

```
easley011
```

```
easley011
```

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

```
[vanilla@easley ~]$ srun --partition debug --nodes 2 --ntasks-per-node 4 hostname  
srun: Account not specified in script or ~/.default_slurm_account, using latest project  
easley012
```

You have not been allocated GPUs. To request GPUs, use the -G option in your submission script.

```
easley012
```

```
easley011
```

```
easley011
```

```
easley012
```

```
easley012
```

```
easley011
```

```
easley011
```

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@easley ~]$ 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  
easley011  
You have not been allocated GPUs. To request GPUs, use the -G option in your  
submission script.  
easley011  
easley012  
easley012
```

And we can combine all three.

```
[vanilla@easley ~]$ 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
```

```
easley012
```

```
easley012
```

```
You have not been all  
submission script.
```

```
easley011
```

```
easley011
```

**And we can specify how much
memory we want.**

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

```
[vanilla@easley ~]$ 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
```

```
easley012
```

```
easley012
```

```
You have not been allocated  
submission script.
```

```
easley011
```

```
easley011
```

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@easley ~]$ 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
```

```
easley012
```

```
easley012
```

```
You have not been all  
submission script.
```

```
easley011
```

```
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@easley ~]$ srun --partition debug --mem 4G  
--nodes 2 --ntasks-per-node 2 --cpus-per-Easley hostname  
srun: Account not specified in script or ~/.default_slurm_account, using latest  
project  
easley012  
easley012  
You have not been allocated a submission script.  
easley011  
easley011
```

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@easley ~]$ 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, and only if, you already have a workshops directory:

```
cd workshops
git pull
```

```
[vanilla@easley ~]$ cd workshops  
[vanilla@easley ~]$ cd intro_workshop  
[vanilla@easley intro_workshop]$ pwd  
/users/vanilla/workshops/intro_workshop  
[vanilla@easley 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@easley 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@easley intro_workshop]$
```

We **submit** our slurm
shell script with the
sbatch command.

```
[vanilla@easley 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@easley intro_workshop]$
```

We **submit** our slurm
shell script with the
sbatch command.

Easley 250K jobs (6 mo beta, 1 mo production)

Hopper: 4 million+ jobs (4 yr)

Xena: 1.6 million+ jobs (5 yrs, retiring)

```
[vanilla@easley 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@easley intro_workshop]$
```

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.

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

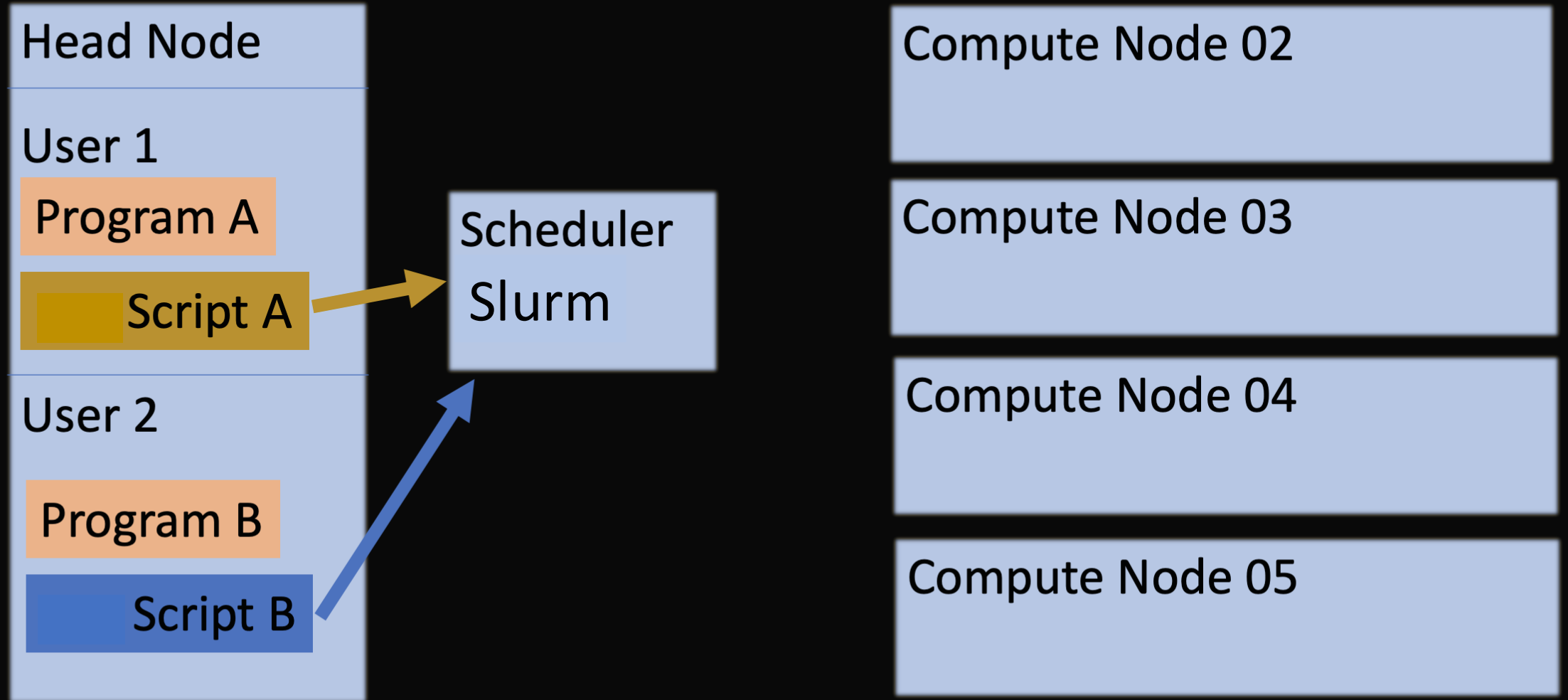
Compute Node 03

Compute Node 04

Compute Node 05

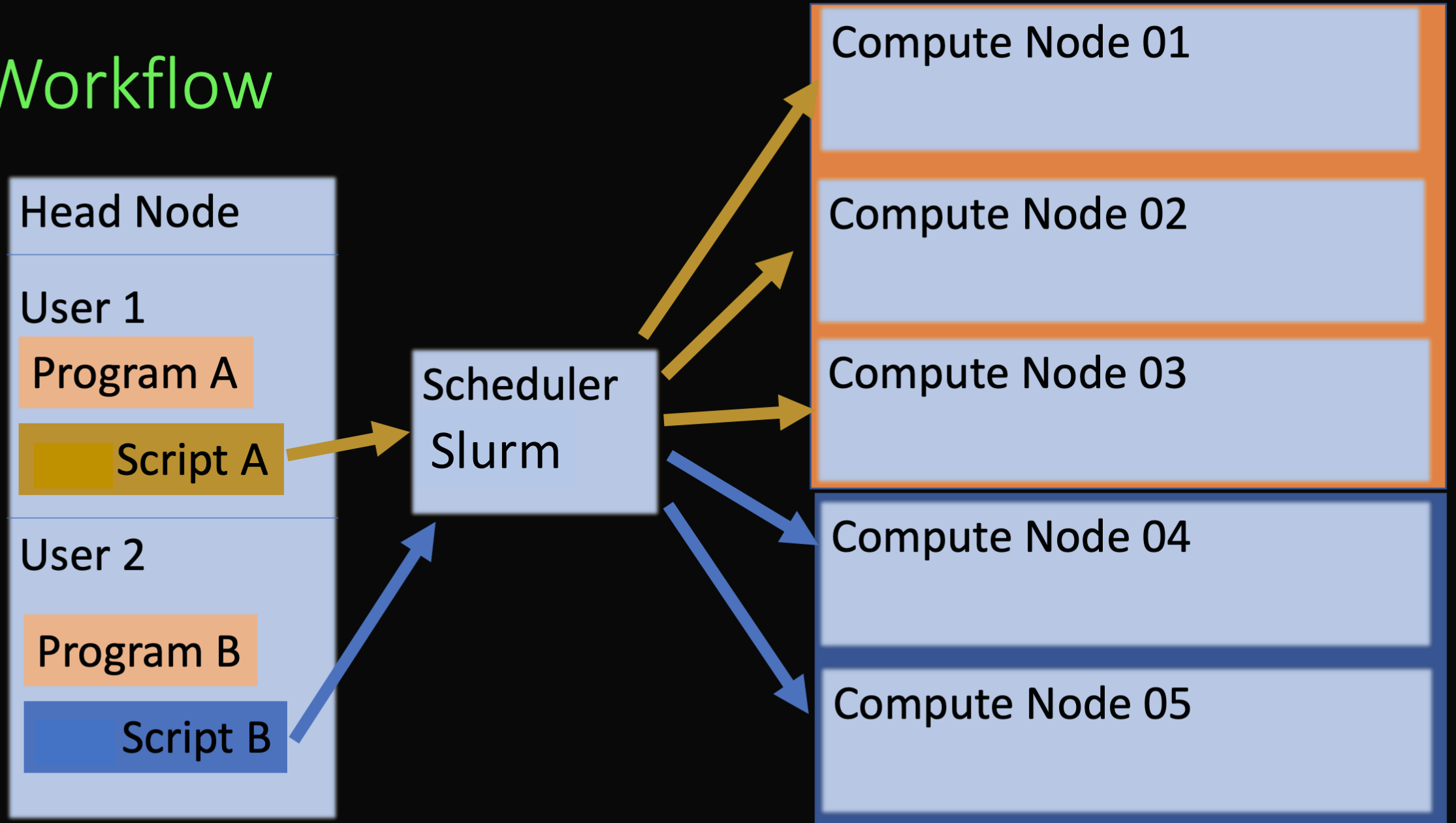
Shared filesystems – All nodes can access the same programs and write output

Workflow



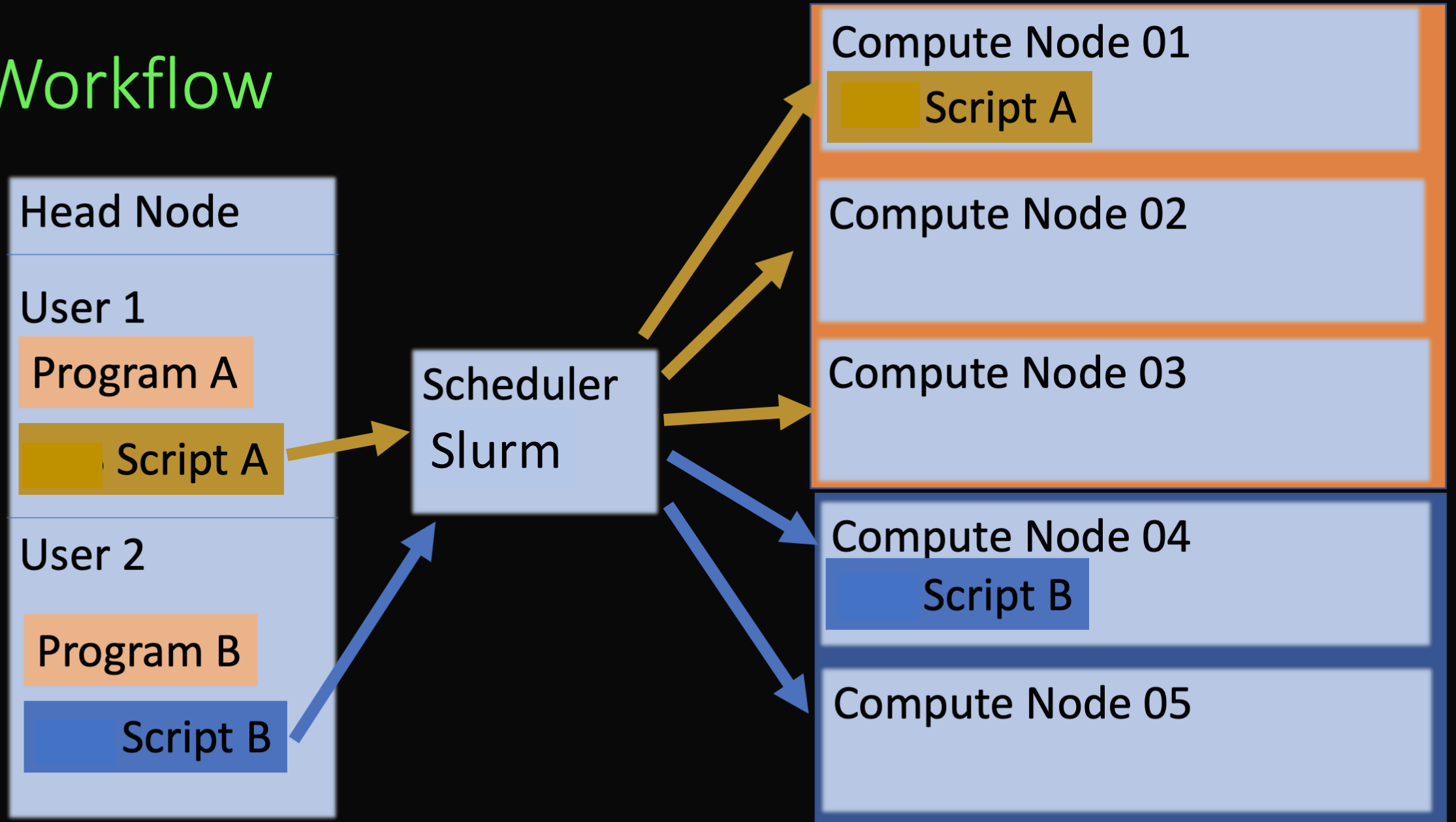
Shared filesystems – All nodes can access the same programs and write output

Workflow



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

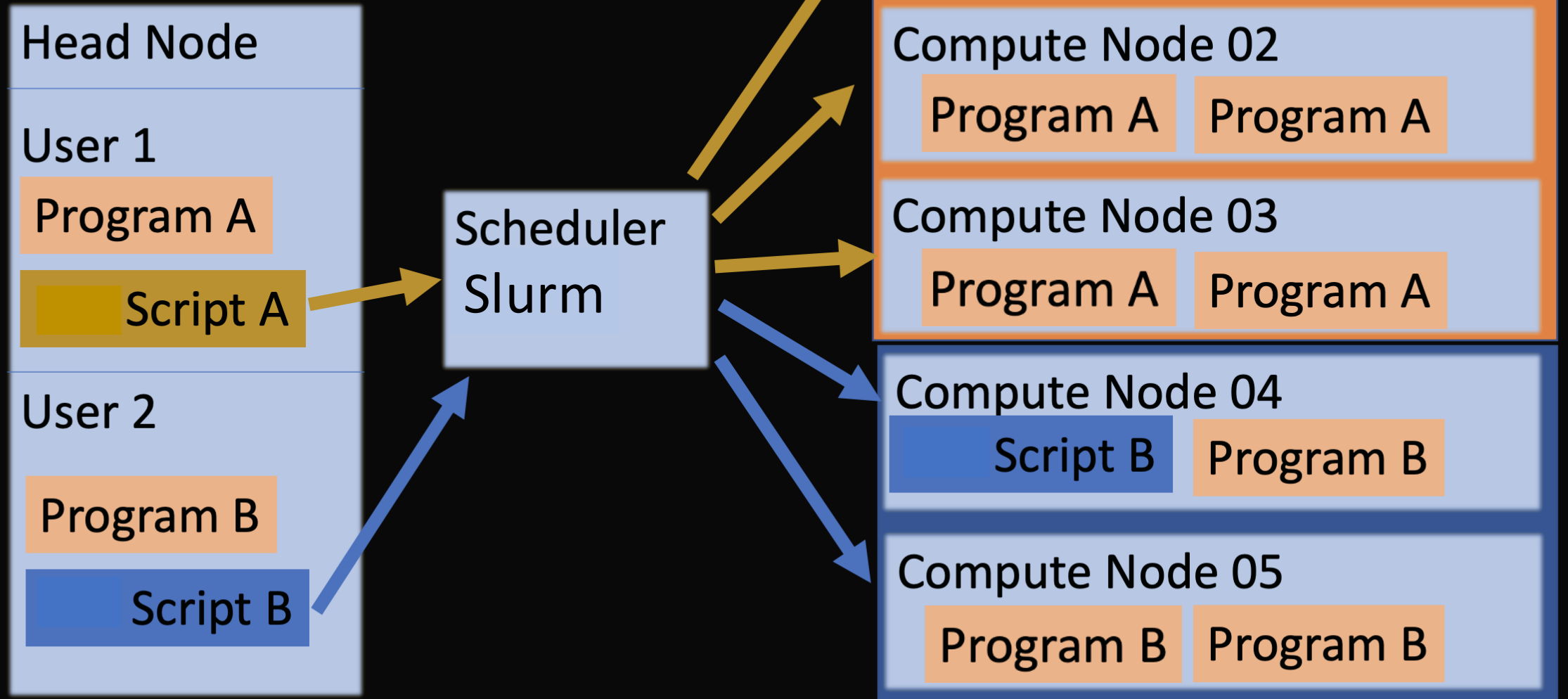
Workflow



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@easley 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@easley 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@easley intro_workshop]$ cat slurm-5252.out  
easley011
```

```
[vanilla@easley 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@easley intro_workshop]$ cat slurm-5252.out  
easley011
```

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

```
[vanilla@easley 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@easley intro_workshop]$
```

**Take a look at the
output file.**



- Python library built on PyTorch
- Simulates and trains spiking neural networks on CPUs and GPUs
- Models neuron dynamics and learns weights using surrogate gradients
- Exports compatible networks for neuromorphic hardware through NIR and chip-specific tools
- I've installed it for you: `conda activate pytorch-2.5.1`

Our snnTorch example and its expected results

snntorch_test.py

Neuron dynamics

Four time steps produce spikes [0, 1, 0, 1].

Learning

Loss falls from 1.0 to 0.0 within 25 optimizer steps.

Batched computation

Forward and backward checks pass with 1 or 4 CPU threads.

Successful run: SNNTORCH_TEST_PASS

Saved summary: result.json

Loading the shared Python environment on Easley

The batch runner loads this environment for every job.

```
module purge  
module load miniconda3  
conda activate /projects/shared/conda/envs/pytorch-2.5.1
```

Shared environment

PyTorch and snnTorch come from the same installation.

Anatomy of the snnTorch submission script

Resource request (excerpt)

```
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=4G
#SBATCH --time=00:05:00
```

slurm-test.slurm

run-test.sh

snntorch_test.py

1 compute node / 1 Python process / 1 CPU core

Demonstration: submit a one-core job

From the directory containing `~/workshops/neuromorphic/`

```
cd Easley/single-node  
sbatch slurm-test.slurm
```

Slurm prints a job ID. Use it to follow your job.

```
squeue --me
```

PD = pending **R = running**

The job may finish before the next queue check.

Reading the logs and result.json

Default output location

```
/easley/scratch/users/<username>/app-test-runs/snntorch/  
test-snntorch-single-node-<jobid>/
```

test.log

Runner output and errors

python.log

Checks and SNNTORCH_TEST_PASS

result.json

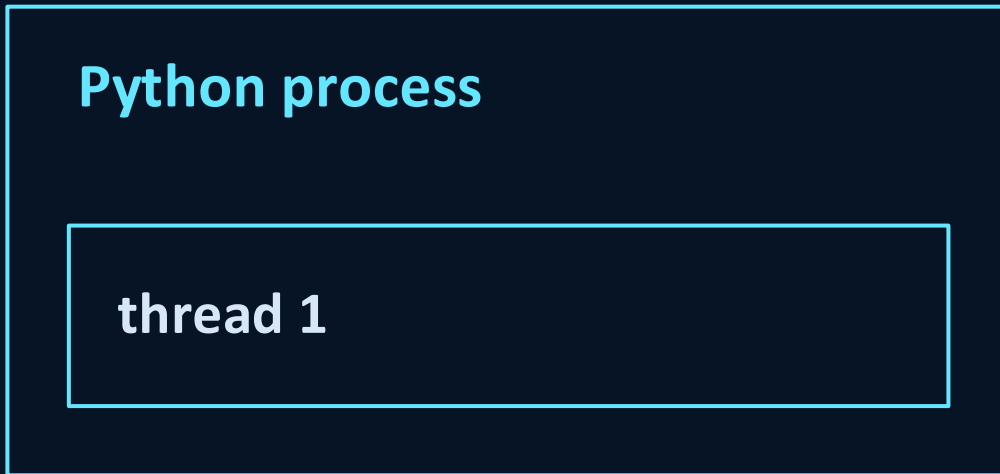
Versions, threads, spikes and loss

Expected: threads = 1 spikes = [0, 1, 0, 1] final_loss = 0

One CPU core versus four CPU cores

One node and one Python process in both jobs

1 CPU core



4 CPU cores



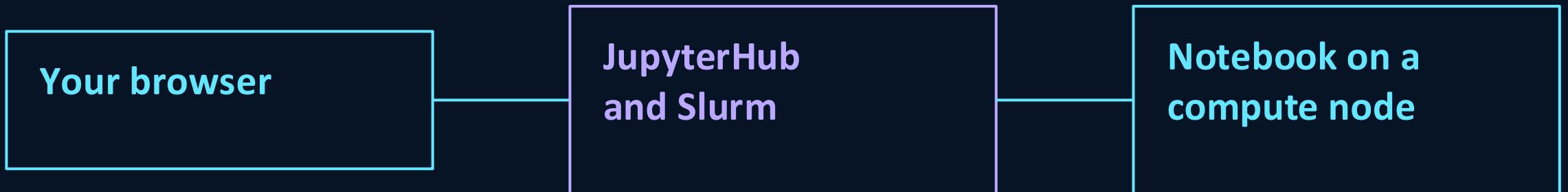
PyTorch uses the requested thread count for CPU work.

Both should pass the same checks. More cores may not be faster.

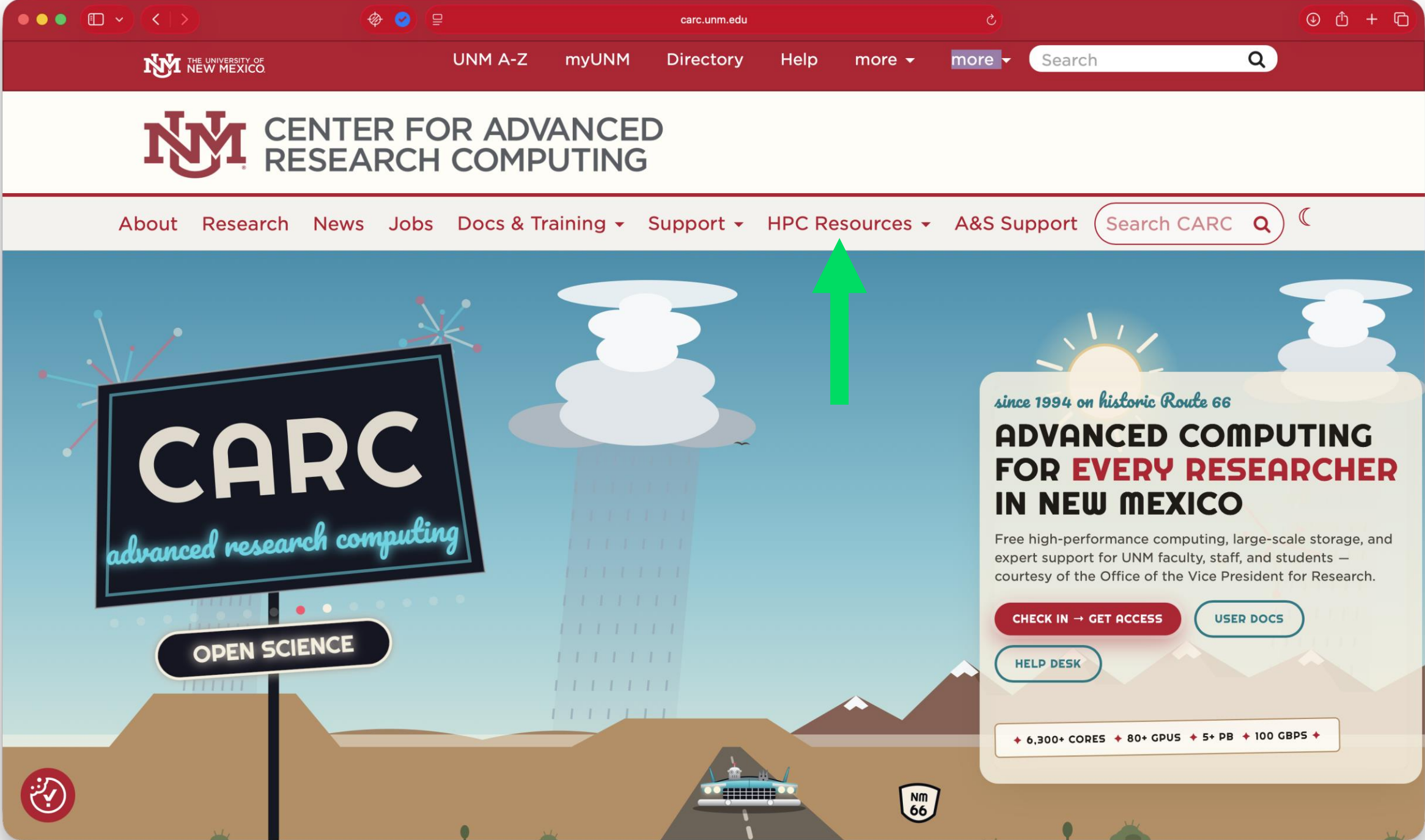
Starting Jupyter on Easley

<https://easley.alliance.unm.edu/jupyter>

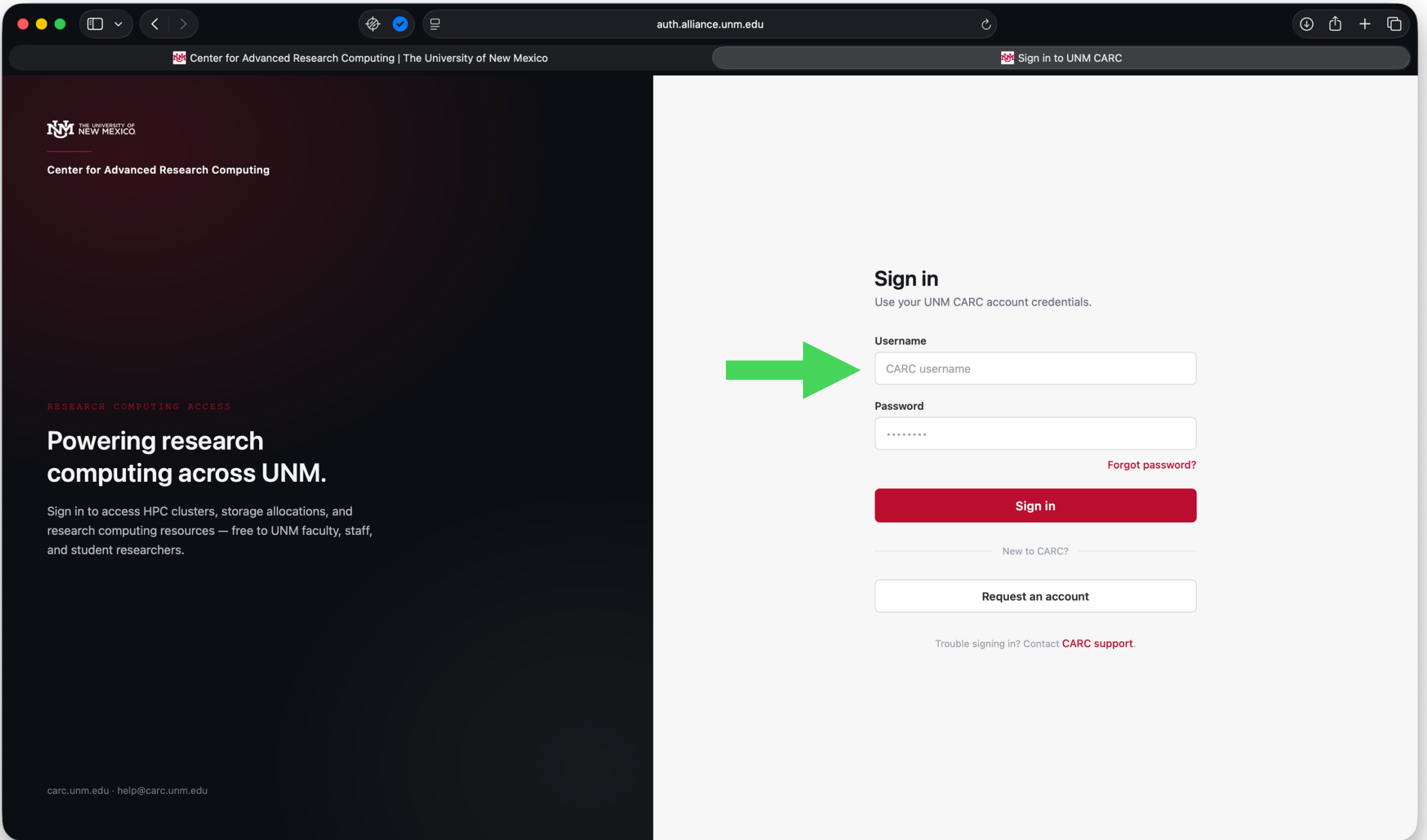
Sign in with your CARC account, then choose resources and session duration in the launch form.



Your notebook server is a Slurm job and may wait in the queue.



10+ GPUS ♦ 5+ PB ♦ 100 GBPS ♦



Center for Advanced Research Computing

RESEARCH COMPUTING ACCESS

Powering research computing across UNM.

Sign in to access HPC clusters, storage allocations, and research computing resources — free to UNM faculty, staff, and student researchers.

carc.unm.edu · help@carc.unm.edu

Sign in

Use your UNM CARC account credentials.

Username

CARC username

Password

.....

[Forgot password?](#)

Sign in

[New to CARC?](#)

Request an account

Trouble signing in? Contact [CARC support](#).

ood.alliance.unm.edu

Center for Advanced Research Computing | The University of New Mexico

Dashboard - Open OnDemand

Open OnDemand

Files

Jobs

Clusters

Interactive Apps

My Interactive Sessions

Help

Logged in as vanilla

Log Out

OPEN

OnDemand

OnDemand provides

Recent



Jupyter Note
System Install



Visual Studio Code
System Installed App



RStudio Server
System Installed App

Desktops

Desktop

Development

Visual Studio Code

IDE

RStudio Server

Servers

Jupyter Notebook

JupyterHub (Easley)

JupyterHub (Hopper)

powered by

OPEN OnDemand

OnDemand version: 4.1.5

[Software License Notice](#)

Interactive Apps

Desktops

 Desktop

Development

 Visual Studio Code

IDE

 RStudio Server

Servers

 Jupyter Notebook

Jupyter Notebook

This app will launch a Jupyter Notebook server on one or more nodes.

Cluster

Easley

☐ Show specialised partitions

Specialised partitions require a ColdFront allocation request from a project manager or PI.

Partition

interactive

Number of nodes (limit: 1)

1

Number of CPU cores (limit: 64)

1

Number of GPUs (limit: 1)

0

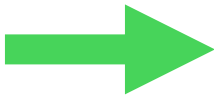
Memory per node (GB) (limit: 235 GB)

4

Walltime (limit: 4h)

04:00:00

Format: HH:MM:SS or DD-HH:MM:SS.

 Launch

* The Jupyter Notebook session data for this session can be accessed under the [data root directory](#).

Session was successfully created.



[Home](#) / My Interactive Sessions



Interactive Apps

Desktops

 Desktop

Development

 Visual Studio Code

IDE

 RStudio Server

Servers

 Jupyter Notebook

Jupyter Notebook (1349224)

1 node | 1 core | Running

Host: [>_ localhost](#)

 Delete

Created at: 2026-10-20 21:12:52 MDT

Time Remaining: 3 hours and 59 minutes

Session ID: [3dcb1754-4679-4c5e-a565-92dc23bc3cd1](#)

 [Connect to Jupyter](#)



[vanilla@easley ~]\$ squeue --me

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
1351342	interacti	ood-jupy	vanilla	R	0:11	1	easley001

[vanilla@easley ~]\$

Name	Modified
carc-scratch	4y ago
jp	11mo ago
mystuff	4y ago
ondemand	6mo ago
opt	10mo ago
simcov	3y ago
test-programs	last yr.
vecadd	last yr.
workshops	
myscript.sh	4y ago
test.slurm	3y ago
Untitled.ipynb	3y ago

> Console

☐ \$ _ Other

Center for Advanced Research Computing | The University of New Mexico

My Interactive Sessions - Open OnDemand

JupyterLab

FileEditViewRunKernelTabsSettingsHelp

+

/ workshops /

Name	Modified
abinit	9m ago
alphafold3	10m ago
amesim	9m ago
crystallography	10m ago
escape	3y ago
gaussian	9m ago
intro_workshop	10m ago
machine_learning	9m ago
mesastar	9m ago
namd	9m ago
neuromorphic	9m ago
quantum_computing	9m ago
quantum_vis	9m ago
radio_astronomy	3y ago
starccm	9m ago
README.md	9m ago

Launcher

workshops

Notebook



Python 3
(ipykernel)



escape (Python,
personal)



haskell-8.10.7
(Haskell, shared)



humann-3.9
(Python, shared)



humann-3.9 (R,
shared)



humann-4.0.0a1
(Python, shared)



humann-4.0.0a1
(R, shared)



intro_workshop
(Python, shared)



julia-1.10.4 (Julia,
shared)



plotting (Python,
shared)



pytorch-2.5.1
(Python, shared)



qiime2-2022.11
(Python, shared)

Simple

0

Launcher 0

o.d.alliance.unm.edu

Center for Advanced Research Computing | The University of New Mexico

My Interactive Sessions - Open OnDemand

JupyterLab

File Edit View Run Kernel Tabs Settings Help

/ workshops / neuromorphic /

Name	Modified
assets	12 minutes ago
Easley	12 minutes ago
image_classifier	12 minutes ago
Neuromorphic_HPC_snnTorch_workshop.ipynb	12 minutes ago
run-test.sh	12 minutes ago

Launcher

workshops/neuromorphic

Notebook



Python 3
(ipykernel)



escape (Python,
personal)



haskell-8.10.7
(Haskell, shared)



humann-3.9
(Python, shared)



humann-3.9 (R,
shared)



humann-4.0.0a1
(Python, shared)

Simple

0

Launcher 0

FileEditViewRunKernelTabsSettingsHelp

+

+

↑

↺

⏏

/ workshops / neuromorphic /

Name	Modified
assets	38m ago
Easley	38m ago
image_classifier	38m ago
Neuromorphic_HPC_...	38m ago
run-test.sh	38m ago

Neuromorphic_HPC...

+

✂

📄

📋

Neuromorphic_HPC...

CARC w

A spiking

visible, t

Run cell

Use No Kernel

No Kernel

Start Kernel

haskell-8.10.7 (Haskell, shared)

humann-3.9 (R, shared)

humann-4.0.0a1 (R, shared)

julia-1.10.4 (Julia, shared)

r-4.5 (R, shared)

Connect to Existing python Kernel

✓ Neuromorphic_HPC_snnTorch_workshop.ipynb (b110706d)

Notebook ↗ ⚙️ pytorch-2.5.1 (Python, shared) ○

snnTorch notebook

t crosses a threshold, and resets. Here we make every part

, numpy , and matplotlib . On Easley, use the workshop's

s not require a GPU or a particular Slurm environment

g below its cell. You can use those files directly in slides or

ion.

nel matters just as much as activating the correct

Simple 1 ⚙️ pytorch-2.5.1 (Python, shared) | Idle

Mode: Command ⚠️ Ln 1, Col 1 Neuromorphic_HPC_snnTorch_workshop.ipynb 1 🔔

```
[1]: import os
import sys
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
```

File Explorer View:

- Path: /workshops/neuromorphic/
- Files:
 - assets (38m ago)
 - Easley (38m ago)
 - image_classifier (38m ago)
 - Neuromorphic_HPC_...** (38m ago) - Highlighted with a blue bar and a green arrow pointing to it.
 - run-test.sh (38m ago)

Neuromorphic computing on HPC: a first snnTorch notebook

CARC workshop · Easley · CPU example

A spiking neuron accumulates input, loses some stored state, emits a spike when it crosses a threshold, and resets. Here we make every part visible, then explore learning and why an HPC system helps.

Run cells from top to bottom. Select a kernel that contains `torch`, `snntorch`, `numpy`, and `matplotlib`. On Easley, use the workshop's shared `pytorch-2.5.1` environment. This notebook uses CPU computation and does not require a GPU or a particular Slurm environment variable.

Each figure saves a high-resolution PNG in `snntorch_figures/` as well as appearing below its cell. You can use those files directly in slides or take screenshots of the notebook. All examples are small enough for an interactive session.

1. Our Python environment

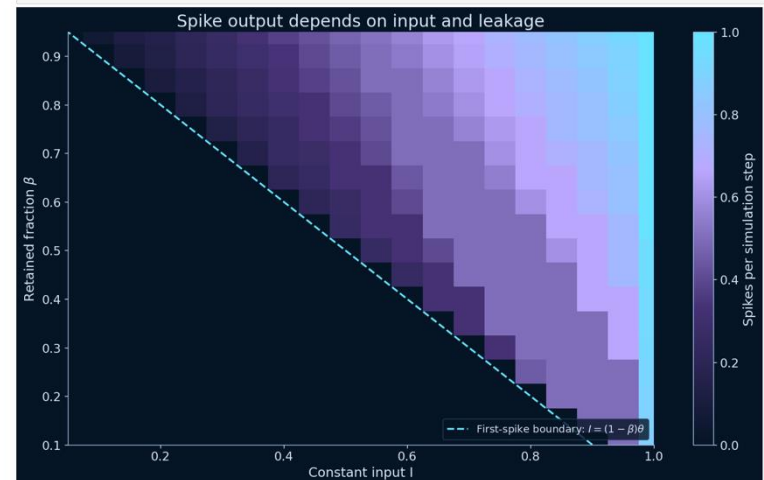
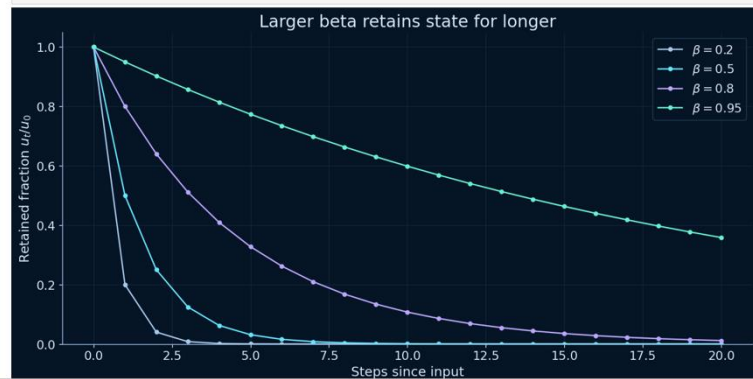
A Jupyter **kernel** is the Python process executing these cells. Choosing the correct kernel matters just as much as activating the correct environment in a batch job.

On Easley, `sys.executable` and `snn.__file__` should come from:

```
/projects/shared/conda/envs/pytorch-2.5.1/
```

The version report below records the packages actually used. If an import fails, switch to the workshop kernel or ask the instructor to check it. Do not install into the shared environment.

```
[1]: import os
import sys
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
```



360 independent experiments, 80 steps each

9. Learning with a hard spike and a surrogate gradient

The forward computation still makes a discrete decision:

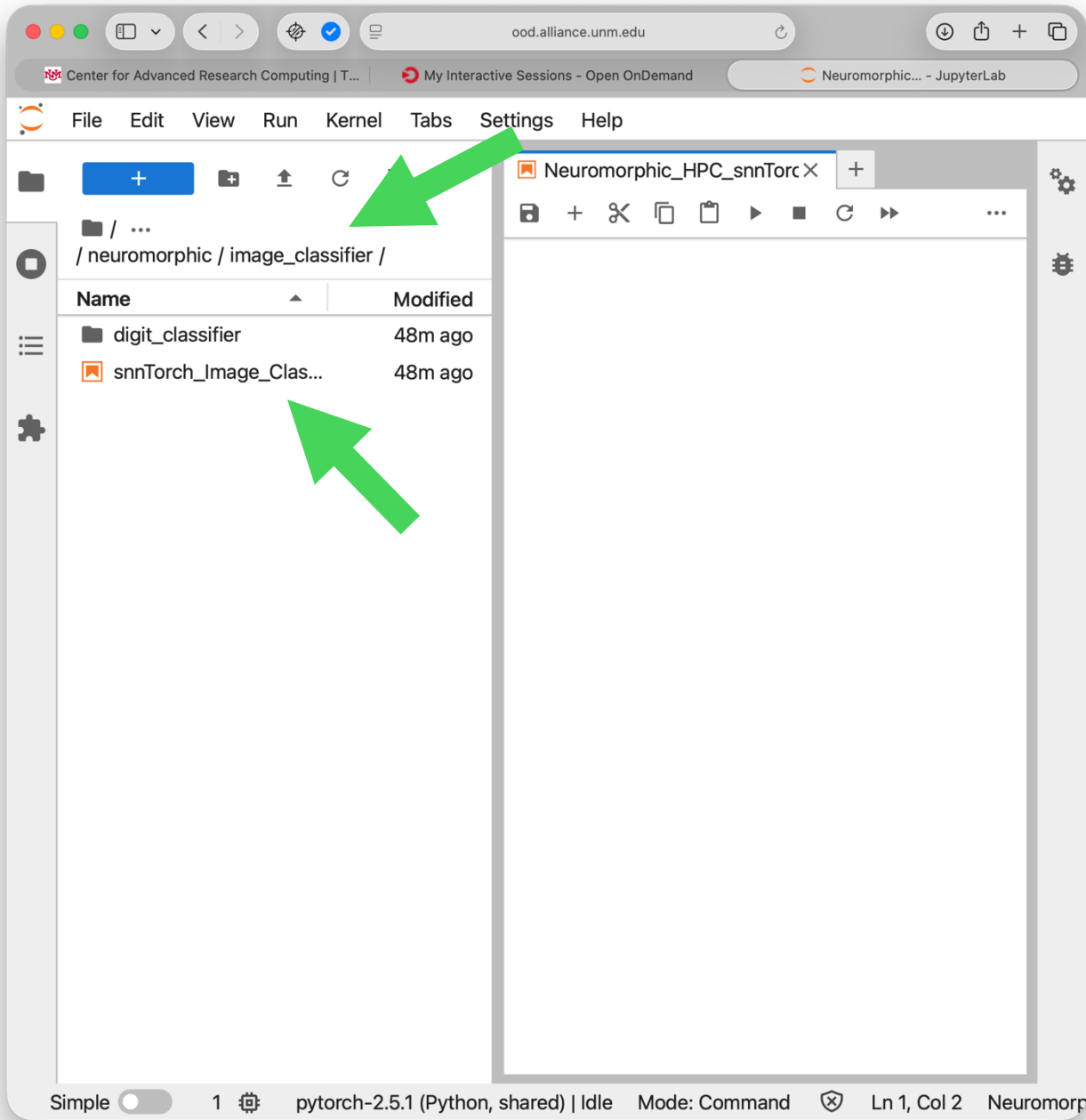
$$s = H(z), \quad z = \bar{u} - \theta.$$

The true step function has zero derivative almost everywhere and is discontinuous at zero, which makes ordinary gradient learning difficult. During the **backward pass**, `snnTorch` can substitute a smooth derivative. With the fast-sigmoid surrogate used here:

$$\frac{\partial s}{\partial z} \approx \frac{1}{(1+k|z|)^2}, \quad k=5.$$

This changes the derivative used for learning. It does **not** turn the forward spike into a smooth output. The next figure plots the hard forward decision and the substitute backward derivative separately.

```
(12.) z = np.linspace(-1, 1, 600)
k = 5
fig, ax = plt.subplots(1, 2, figsize=(12, 4.0), layout="constrained")
ax[0].plot(z, z - theta_0.astype(float), color='CN', lw=3)
ax[0].set(title="Forward pass: a hard spike", xlabel="$z$=widetilde{u}-\theta_{t-1}$",
        ylabel="Spikes", yticks=(0, 1))
ax[1].plot(z, 1 / (1 + k * np.abs(z)) ** 2, color='Violet', lw=3)
ax[1].set(title="Backward pass: surrogate derivative", xlabel="$z$=widetilde{u}-\theta_{t-1}$",
        ylabel="Approximate $\partial$partial $z$/$\partial$partial $z$")
for a in ax:
    a.axvline(0, color='MUTED', ls='--')
    a.grid(alpha=0.3)
show_figure(fig, "07 surrogate gradient.png")
```



Workflow:

1. Load the dataset: Import labeled images and inspect the input data.
2. Preprocess images: Resize, normalize, and prepare images for the network.
3. Encode images as spikes: Convert pixel intensities into spike trains that represent the input over time.
4. Construct the spiking neural network (SNN) — Define layers, neuron models, synaptic connections, and parameters.
5. Train the network: Adjust synaptic weights to improve classification accuracy.
6. Run inference: Present test images and simulate neuron activity over successive time steps.
7. Decode the output : Convert output neuron activity into predicted image classes.
8. Evaluate performance: Compare predictions with labels and calculate classification accuracy.
9. Explore parameters: Investigate how spike encoding, simulation duration, and network parameters affect performance.

Center for Advanced Research Computing | T...

File Edit View Run Kernel Tabs Set

File Edit View Run Kernel

+

/ neuromorphic / image_classifier /

Name Modified

digit_classifier 48m ago

snnTorch_Image_Clas... 48m ago

Python 3 (ipykernel)

pytorch-2.5.1 (Python, shared)

qiime2-2022.11 (Python, shared)

qiime2-amplicon-2024.5 (Python, shared)

qiime2-amplicon-2026.1 (Python, shared)

qiskit-1.0.2 (Python, shared)

tetrad-0.9.14 (Python, shared)

vllm-cuda-0.19.1 (Python, shared)

Use No Kernel

No Kernel

Start Kernel

haskell-8.10.7 (Haskell, shared)

humann-3.9 (R, shared)

humann-4.0.0a1 (R, shared)

julia-1.10.4 (Julia, shared)

r-4.5 (R, shared)

Connect to Existing python Kernel

Neuromorphic_HPC_snnTorch_workshop.ipynb (b110706d)

✓ snnTorch_Image_Classifier_Easley.ipynb (4040f3ce)

Notebook Python 3 (ipykernel)

its with a spiking

dwritten digit, predict its value from 0

0-image test set, then accepts your

ne most spikes wins. We will visualize

rch, snnTorch, NumPy, Matplotlib, and

ell downloads about 12 MB from the

ork works offline with its cached data. If the

compute node cannot access the internet, transfer the four .gz data files into

digit_classifier/data/ first.

This is a learning demonstration on conventional CPU/GPU hardware. It measures classification

rary

[1]: import os, sys, json, gzip, hashlib, struct, time, urllib.request

from pathlib import Path

Simple 2 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 snnTorch_Image_Classifier_Easley.ipynb 1

Simple 1 pytorch-2.5.1 (Python, shared) | Idle Mode: Command Ln 1, Col 2 Neuromorp

Choose the preinstalled ssntorch kernel (conda environment)

ood.alliance.unm.edu

Center for Advanced Research Computing | The University of New Mexico

My Interactive Sessions - Open OnDemand

snnTorch_Ima... (2) - JupyterLab

File Edit View Run Kernel Tabs Settings Help

Neuromorphic_HPC_snnToiX

snnTorch_Image_Classifier_X

Code

Notebook

pytorch-2.5.1 (Python, shared)

neuromorphic / image_classifier /

Name	Modified
digit_classifier	4s ago
snnTorch_Image_Cla...	next yr.

Test accuracy: 95.72%

True digit

0	972	0	0	0	1	2	1	1	2	1
1	0	1124	4	1	0	0	3	0	3	0
2	10	3	997	1	1	0	2	12	5	1
3	2	1	27	949	1	9	0	8	10	3
4	1	0	8	0	950	0	1	1	2	19
5	10	1	0	25	4	835	4	1	10	2
6	24	4	3	1	4	23	896	0	3	0
7	3	5	16	2	1	0	0	992	2	7
8	11	1	21	10	5	8	2	12	899	5
9	9	8	1	9	13	3	0	7	1	958

Number of images

True 7 / predicted 7

True 2 / predicted 2

True 1 / predicted 1

True 0 / predicted 0

True 4 / predicted 4

True 1 / predicted 1

Simple

2

pytorch-2.5.1 (Python, shared) | Idle

Mode: Edit

Ln 1, Col 1

snnTorch_Image_Classifier_Easley.ipynb

1

ood.alliance.unm.edu

Center for Advanced Research Computing | The University of New Mexico

My Interactive Sessions - Open OnDemand

snnTorch_Ima... (2) - JupyterLab

File Edit View Run Kernel Tabs Settings Help

+

+

↑

↺

🔍

/ neuromorphic / image_classifier /

Name	Modified
digit_classifier	4s ago
snnTorch_Image_Cla...	next yr.

⚙️

Neuromorphic_HPC_snnTorch

snnTorch_Image_Classifier_×

+

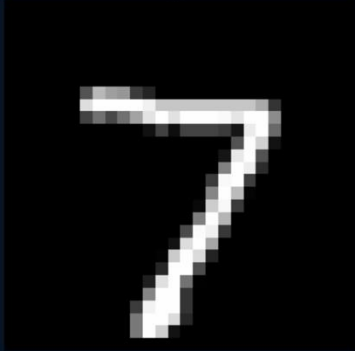
Code

Notebook

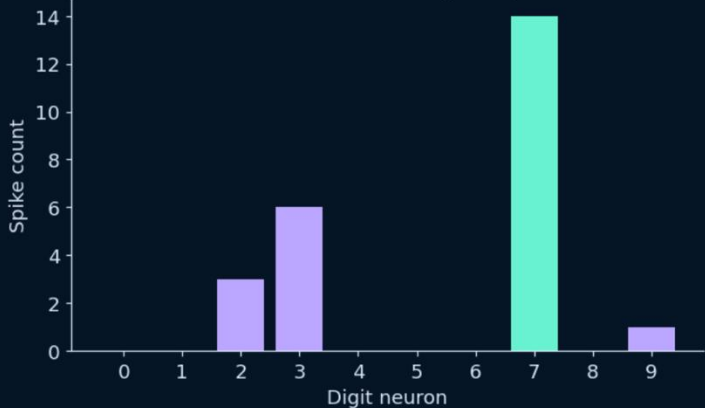
pytorch-2.5.1 (Python, shared)

MNIST test image 0, true label 7 | predicted digit: 7

The 28 x 28 image the model sees

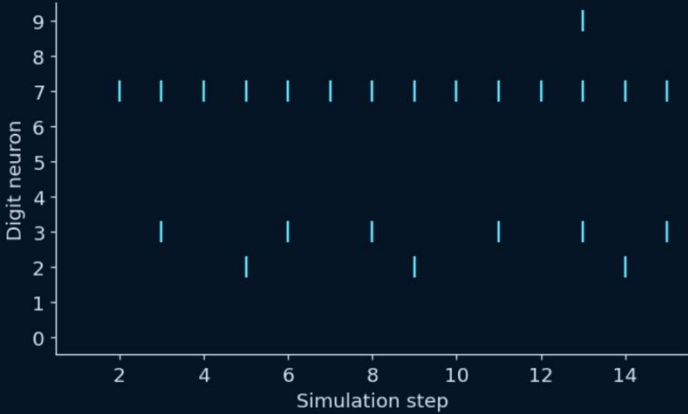


Decision: most output spikes



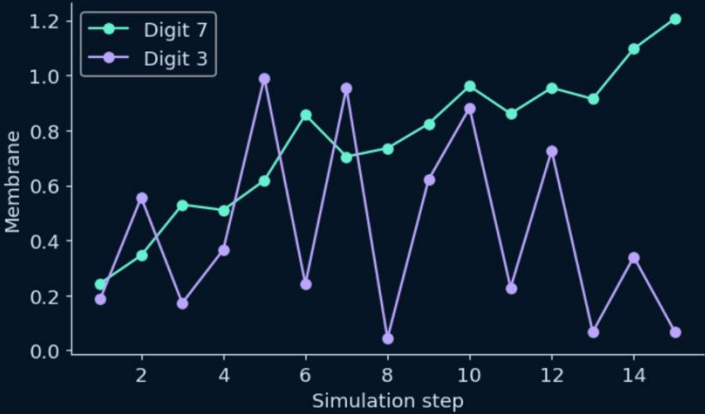
Digit neuron	Spike count
0	0
1	0
2	3
3	6
4	0
5	0
6	0
7	14
8	0
9	1

Output spike raster



Simulation step	0	1	2	3	4	5	6	7	8	9
2	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	1	0	0
12	0	0	0	0	0	0	0	1	0	0
13	0	0	0	0	0	0	0	1	0	0
14	0	0	0	0	0	0	0	1	0	0

Returned membrane after reset



Simulation step	Digit 7	Digit 3
2	0.25	0.20
3	0.35	0.55
4	0.50	0.18
5	0.60	0.35
6	0.85	0.98
7	0.70	0.25
8	0.75	0.95
9	0.80	0.05
10	0.95	0.60
11	0.85	0.90
12	0.95	0.22
13	0.90	0.72
14	1.10	0.08

Output spike counts: [0, 0, 3, 6, 0, 0, 0, 14, 0, 1]
Prediction: 7

Simple

2

pytorch-2.5.1 (Python, shared) | Idle

Mode: Edit

Ln 1, Col 1

snnTorch_Image_Classifier_Easley.ipynb

1

Selecting and checking the Python environment

Choose the kernel for the shared pytorch-2.5.1 environment.

```
import sys
import torch
import snntorch as snn

print(sys.executable)
print(snn.__file__)
```

Both paths should begin with:

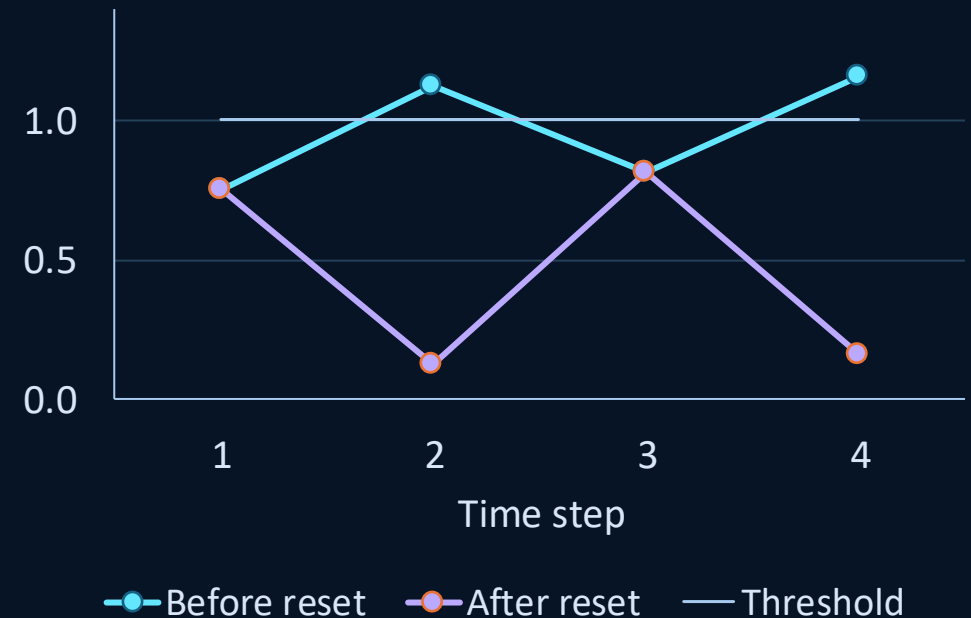
```
/projects/shared/conda/envs/pytorch-2.5.1/
```

Exploring membrane potential and spikes

The same four-step neuron in a notebook

```
neuron = snn.Leaky(
    beta=0.5, threshold=1.0,
    reset_delay=False)
mem = torch.zeros(1)
spikes, membranes = [], []
for t in range(4):
    spk, mem = neuron(
        torch.tensor([0.75]), mem)
    spikes.append(spk.item())
    membranes.append(mem.item())
```

Returned mem is the value after reset.



Spikes: [0, 1, 0, 1]

Exercise: change the input current or leakage

Predict the spikes, then rerun from zero membrane potential.

A Change the input

```
torch.tensor([0.60])
```

Keep $\beta = 0.5$. When does the first spike occur?

B Change the memory

Restore input 0.75 and use $\beta = 0.8$.

Larger β retains more of the previous state.

Four-step spike count



```
[vanilla@easley Easley]$ pwd
/users/vanilla/workshops/neuromorphic/Easley
[vanilla@easley Easley]$ tree
```

```
.
├── 140s
│   ├── digit-classifier.slurm
│   └── slurm-test.slurm
├── multi-cpu
│   ├── digit-classifier.slurm
│   └── slurm-test.slurm
└── single-node
    ├── digit-classifier.slurm
    └── slurm-test.slurm
```

Jupyter is great for
developing ML interactively

But if you want to run lots of
training jobs use slurm
scripts

3 directories, 6 files

```
[vanilla@easley Easley]$
```

```
#!/bin/bash -l
#SBATCH --job-name=neuromorphic-l40s
#SBATCH --output=neuromorphic-%j.out
#SBATCH --partition=l40s
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --gpus-per-node=1
#SBATCH --mem=4G
#SBATCH --time=00:05:00

# Stop if a command fails.
set -e
cd "$HOME/workshops/neuromorphic"
# Activate the shared workshop environment on Easley.
module load miniconda3
conda activate /projects/shared/conda/envs/pytorch-2.5.1

export OMP_NUM_THREADS="$SLURM_CPUS_PER_TASK"
export MKL_NUM_THREADS="$SLURM_CPUS_PER_TASK"
# Requesting a GPU allocates it; --device cuda moves the model and data onto it.
srun --cpu-bind=cores python -l assets/snn_example.py --device cuda
```

```
#!/bin/bash -l
#SBATCH --job-name=digits-l40s
#SBATCH --output=output/digits-%j.out
#SBATCH --partition=l40s
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --gpus-per-node=1
#SBATCH --mem=4G
#SBATCH --time=00:15:00

# Submit from ~/workshops/neuromorphic so Slurm can find output/.
set -e
cd "$HOME/workshops/neuromorphic/image_classifier"
# Use the same environment as the notebook.
module load miniconda3
conda activate /projects/shared/conda/envs/pytorch-2.5.1

export OMP_NUM_THREADS="$SLURM_CPUS_PER_TASK"
export MKL_NUM_THREADS="$SLURM_CPUS_PER_TASK"
# Share cached MNIST data, but give each job its own model, metrics, and figures.
srun --cpu-bind=cores python -l -u classify_digits.py --device cuda \
    --output-dir "$HOME/workshops/neuromorphic/output/digits-$SLURM_JOB_ID" "$@"
```

```
[vanilla@easley single-node]$ sbatch digit-classifier.slurm
```

```
Submitted batch job 1351364
```

```
[vanilla@easley single-node]$ watch squeue --me
```

```
[vanilla@easley single-node]$ squeue --me
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
1351364	debug	digits-s	vanilla	R	0:13	1	easley001
1351342	interacti	ood-jupy	vanilla	R	10:25	1	easley001

```
[vanilla@easley single-node]$
```

```
[vanilla@easley output]$ pwd
/users/vanilla/workshops/neuromorphic/Easley/single-node/output
[vanilla@easley output]$ cat digits-1351364.out
PyTorch: 2.5.1 snnTorch: 1.0.0
Device: cpu CPU threads: 1
Output directory: /users/vanilla/workshops/neuromorphic/output/digits-1351364
Downloading train-images-idx3-ubyte.gz
Downloading train-labels-idx1-ubyte.gz
Downloading t10k-images-idx3-ubyte.gz
Downloading t10k-labels-idx1-ubyte.gz
Training pool: 60000 Test set: 10000
Training on 20,000 images; evaluating on 10,000 separate test images.
Trainable parameters: 101770
Epoch 1/3: loss=0.4143, training accuracy=86.98%
Epoch 2/3: loss=0.1672, training accuracy=94.95%
Epoch 3/3: loss=0.1111, training accuracy=96.72%
Training time on this machine: 6.5 seconds
Held-out MNIST test accuracy: 95.72% (9572/10000)
Tied spike-count predictions: 200/10000
Saved /users/vanilla/workshops/neuromorphic/output/digits-1351364/digit_snn.pt
Output spike counts: [0, 0, 3, 6, 0, 0, 0, 14, 0, 1]
Prediction: 7
Output spike counts: [0, 0, 3, 6, 0, 0, 0, 14, 0, 1]
Prediction: 7
Results saved to: /users/vanilla/workshops/neuromorphic/output/digits-1351364
```

Figures are saved to file instead of rendered in jupyter notebook

```
~/workshops/neuromorphic/output/digits-1351334/figures [master !?]$ ls
```

01_training_examples.png

03_confusion_matrix.png

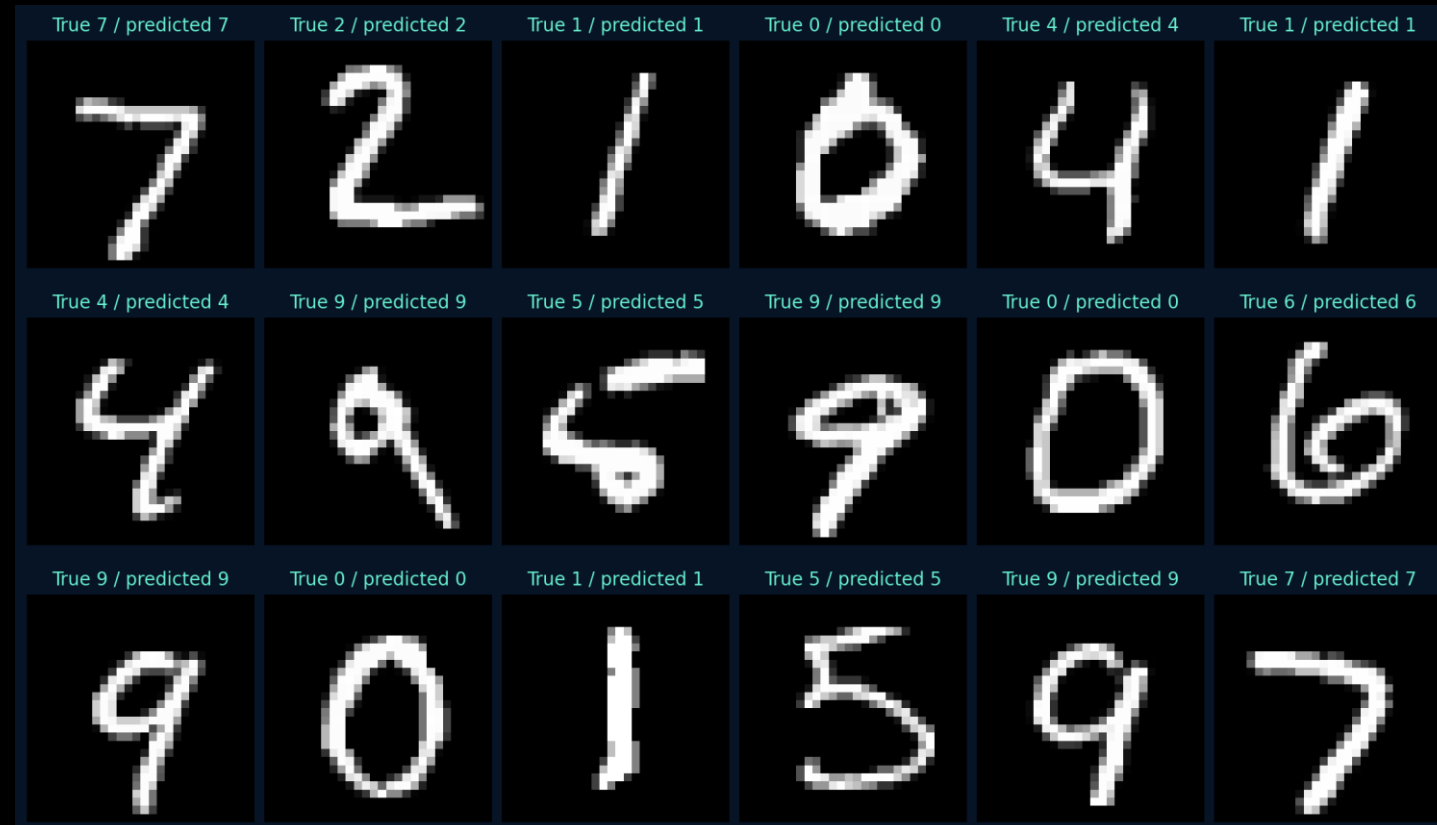
05_example_spikes.png

07_my_digit_spikes.png

02_learning_curves.png

04_test_predictions.png

06_image_preprocessing.png



Notebooks for exploration, batch jobs for longer runs

Jupyter notebook

- Change parameters
- Inspect membrane traces
- Check a small example

Slurm batch job

- Run a saved Python script
- Repeat training or experiments
- Keep logs and result files

Same model and Python environment

Save the notebook, then stop your Jupyter server when finished.

Ordinary PyTorch and our SNN classifier

Both learn weights from the same labeled digit images.

Feedforward PyTorch

Continuous activations

One forward pass per image

Highest output score wins

Ordinary activation derivatives

Our snnTorch network

Spikes and membrane state

15 simulated steps per image

Most output spikes wins

Surrogate gradients through time

Why SNNs? Sparse, event-driven computation can use much less energy on suitable neuromorphic hardware.

Why HPC today? We use CPUs and GPUs to develop, train, and study these networks; this exercise does not demonstrate the hardware's energy savings.

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 snntorch code using SLURM and Open OnDemand

