Research Cyberinfrastructure Center

| Resources | Update | Utilization |

Philip Papadopoulos ppapadop@uci.edu

Joulien Tatar jtatar@uci.edu

Nadya Williams npw@uci.edu

What do we do?
Infrastructure for Researchers

RCIC builds and maintains real infrastructure for

- ✓ High-performance and high-throughput computing
- ✓ Research data storage and analysis,
- ✓ Scientific software tool integration.

Computing and data infrastructure is operated in a *shared financial model* where campus researchers are given no-cost access to a baseline level of computing and highly-reliable storage.

Faculty can also purchase additional capacity and capability using grant or other funds.

RCIC Faculty Oversight

Executive Committee – Chair Filipp Furche, Professor, Dept. of Chemistry

- Help with strategic guidance and direction
- Approval chain for large purchases (> $100K) and high-level policy
- Meet approximately quarterly

Advisory Committee

- About 30 researchers from disciplines across UCI
- Key feedback on what RCIC does right and wrong. They are not shy about expressing their views.

Formation of RCIC was the result of the UCI Cyberinfrastructure Vision 2016

Key Resources @ RCIC

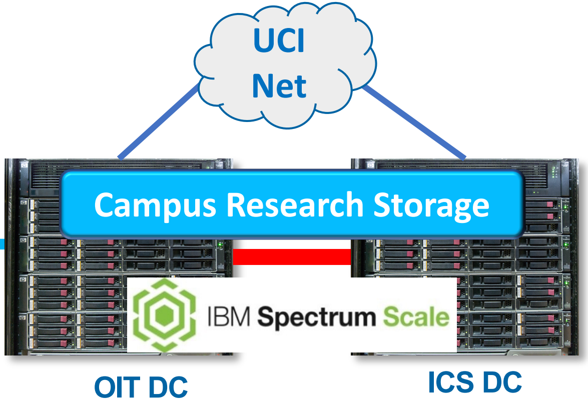
HPC3

- ~6900 Cores/162 Hosts (expanding to ~8500/200)
- 52 Tesla V100 16Gb Nvidia GPUs
- EDR (100Gbps) Infiniband
- 10GbE Ethernet
- Minimum
  - 4GB memory/core
  - AVX2 instruction set (Epyc/Intel CPUs)

Six Parallel File Systems

DFS2, DFS3a, DFS3b, …

- 3.9PB usable storage
- ~6GB/sec bandwidth/System
- Single Copy/No Snapshots



CRSP – Campus Research Storage Pool

- 1 PB usable storage
- Available anywhere on UCI Network
- Dual Copy of All Data
- Snapshots
- Highly available

High-level View of what things cost

No Cost Allocations

Role	HPC3 Core Hours	GPU Hours	Home Area Storage	DFS Storage	CRSP Storage
Faculty	200K hours/year[1]	By Request ~2K hours/year[1]	50GB	1TB in Pub	1 TB
Student	1000 hours	---	50GB	1 TB in Pub	---

Cloud-like Costs

	HPC3 Core Hours	GPU Hours	Home Area Storage	DFS Storage	CRSP Storage
Faculty	$.01/core hour	$0.32/GPU hour	Not expandable	$100/TB/5 years	$60/TB/year
AWS Equivalent	C5n.large $.063	P3.2xlarge $1.95	---	---	S3[2] Standard $242/TB/year

[1] Exact amounts dependent on # requests/available hardware

[2] Comparison difficult - S3 has higher durability, CRSP has no networking fee.

HPC^3 – High Performance Community Computing Cluster

- Short History – And Expansion
- Different Use Cases of HPC3
- How HPC3 is physically connected to UCI
- Queueing and Allocations
- Software Environment
  - What happens when you ask RCIC to install software
  - Organization
  - Insights to usage
- How has HPC3 been used since Jan 1, 2021

Short History of HPC3

Predecessor - HPC

- Catalyzed shared computing at UCI

  Hat-Tip to retired personnel: Joseph Farran , Harry Mangalam, Allen Schiano, Dana Roode, and Garr Updegraff
- Expanded primarily through faculty node purchases (condo computing)
- Reached end of life Dec 2020 – 10500 cores at its peak. Cores 1-9 years in age

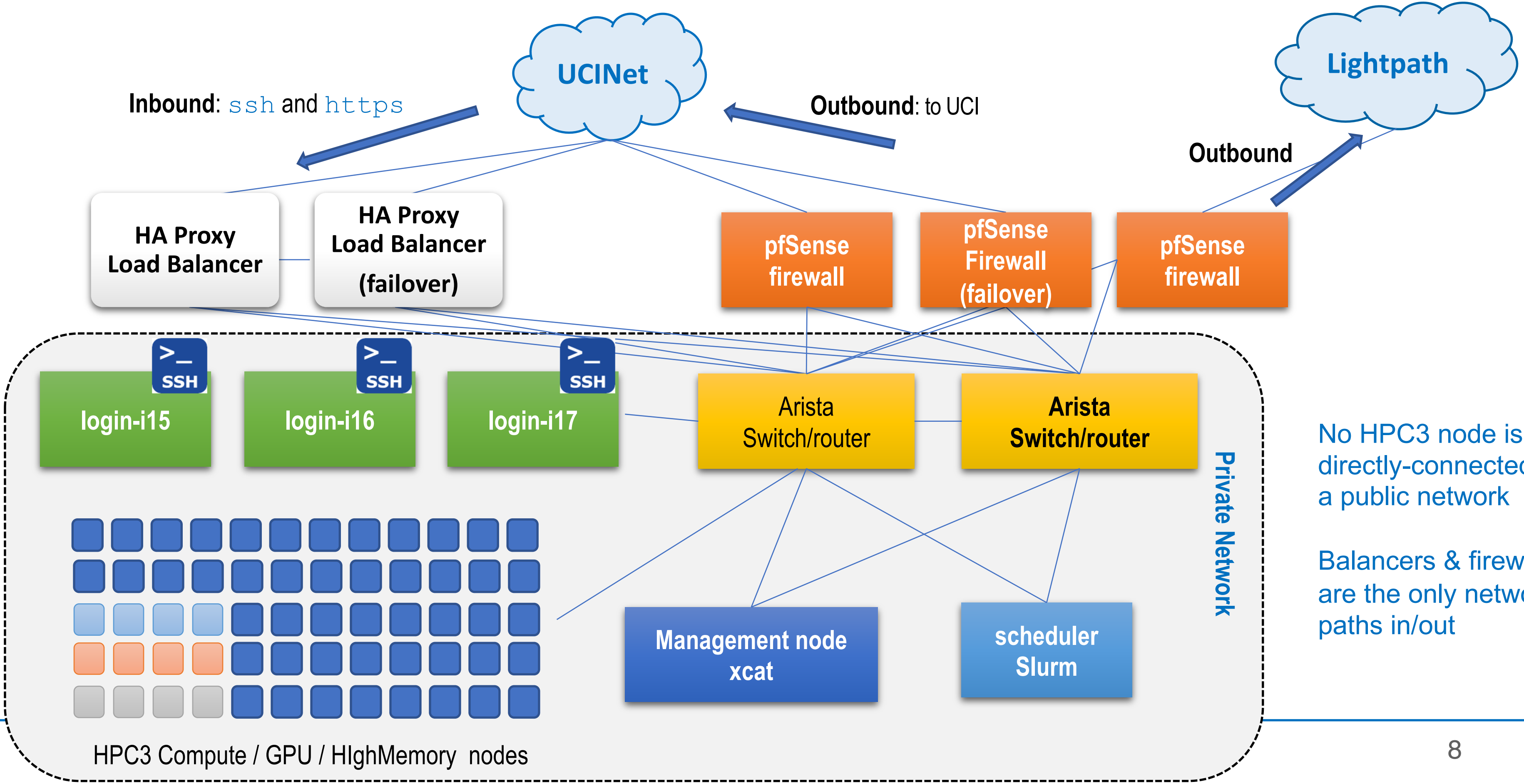
HPC3 catalyzed by NSF Major Research Instrumentation Grant

- PI: Chandramowlishwaran
- Co-PIs: Furche, Roode

Initially constructed from Grant, Faculty Purchase, and Significant UCI investment

- RFP (won by HPE) in Oct 2019. ~100 CPU and GPU nodes (4000 cores total).
- Most nodes arrived after March 2020 during shutdown
- Expanded through faculty/UCI purchase Oct/Dec 2020
- Expanded via compatible HPC nodes moved to HPC3 Jan 2021
- Expanded via UCI/Faculty purchase via April 2021 Competitive Bid (nodes arriving now)

Network Connectivity of HPC3



No HPC3 node is directly-connected to a public network

Balancers & firewall are the only network paths in/out

Different Ways People are Using HPC3

1. Most common: command-line, batch queue, job submission

   ssh hpc3.rcic.uci.edu

2. Teaching courses

   Quarter:

   Grad courses

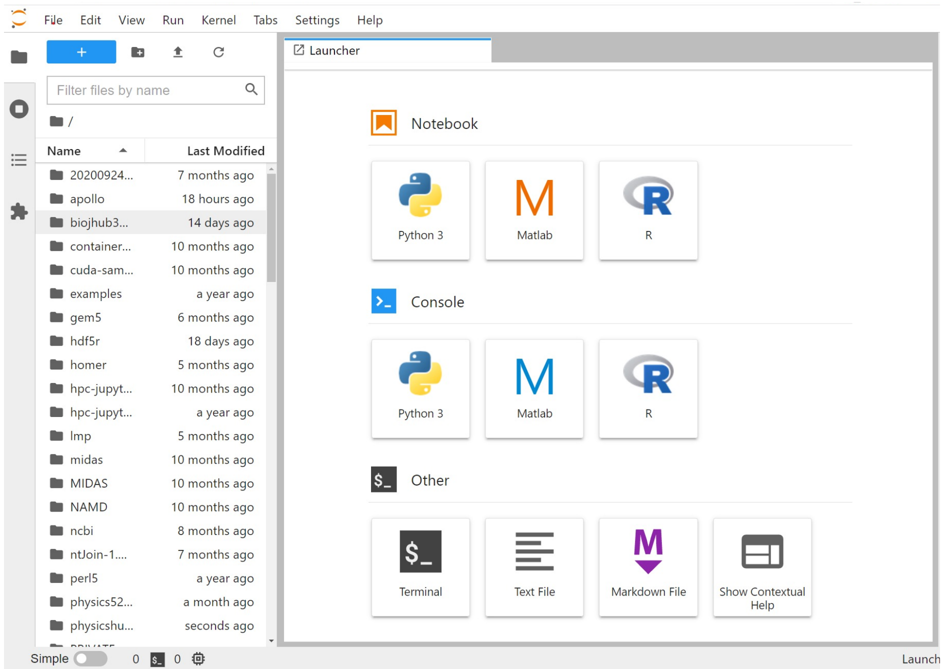
   Lower division physics labs

   Short-term:

   UCI machine learning hackathon

3. Specialized Jupyter Labs

4. Via Singularity Containers



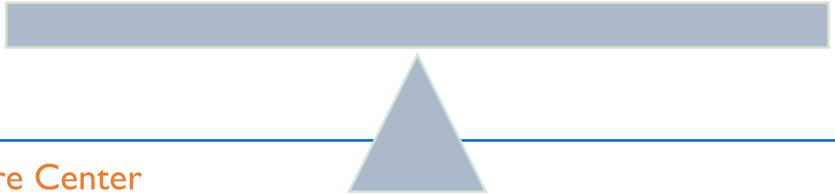
Name	Last Modified
20200924...	7 months ago
apollo	18 hours ago
biojhub3...	14 days ago
container...	10 months ago
cuda-sam...	10 months ago
examples	a year ago
gem5	6 months ago
hdf5r	18 days ago
homer	5 months ago
hpc-jupyt...	10 months ago
hpc-jupyt...	a year ago
lmp	5 months ago
midas	10 months ago
MIDAS	10 months ago
NAMD	10 months ago
ncbi	8 months ago
ntJoin-1....	7 months ago
perl5	a year ago
physics52...	a month ago
physicshu...	seconds ago

Two Types of Jobs

Q: Why is my job not running and squeue output shows AssocGrpCPUMinutesLimit ?

A: not enough balance in your slurm account

Allocated (accounted)	**Free**
Slurm account must have sufficient balance to fund the job to completion	**Slurm account is not debited**
Job once started cannot be pre-empted	**Allocated jobs can pre-epmt running free jobs at any time**
Standard, *mem, gpu, *debug partitions	**free, free-gpu partitions**



Q: Is there checkpointing?

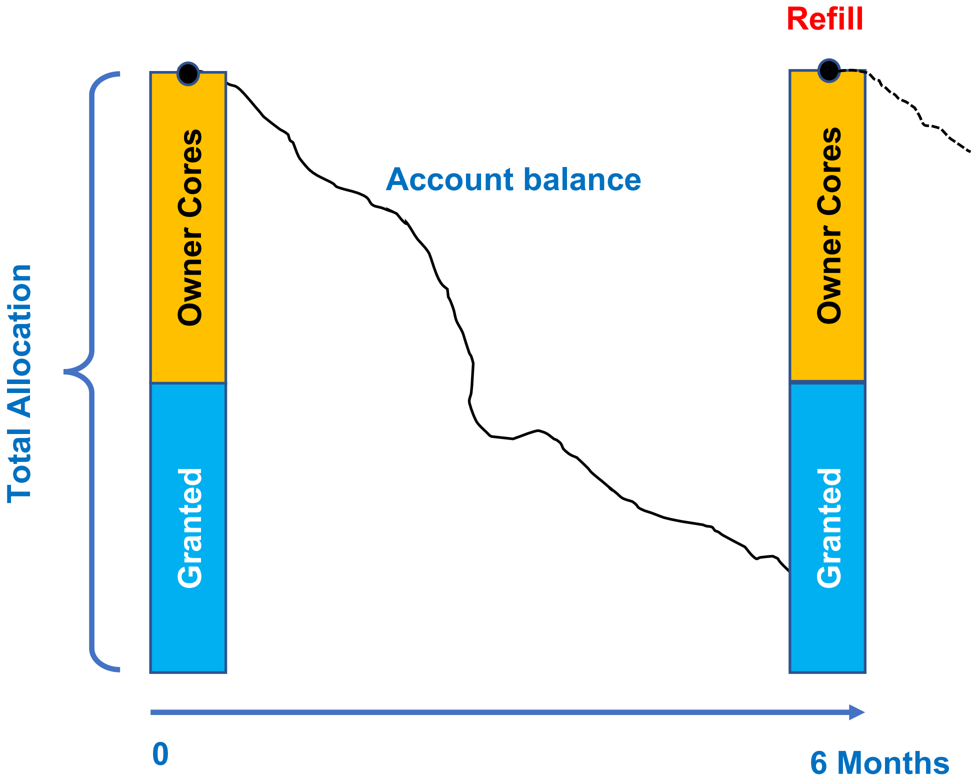
A: **NO checkpointing !**
not a viable technology

Allocated Jobs (standard, *mem, GPU queues)

- All allocated jobs use a common "currency" (SU)
- CPU cores cost 1 SU/Hour
  - There is no differentiation on memory used.
- GPUs cost 32 SUs/Hour
- What about owner hardware/queues?
  - Owner queues do NOT exist. Instead
  - Theoretical capacity of owner hardware is converted into SUs
    - 95% * (# cores) * 8760 hours/year – 40 core node ~ 325K SUs/year
    - 95% * (# gpus) * 32 * 8760 hours/year.
  - GPU SUs and CPU SUs are not "convertible". → need a GPU account to charge runs on the GPU queue.

Automated Refill of Allocations for labs

- Account balances are reset every 6 months
- Each Lab is on their own cycle
- Allocations are for "the next 6 months"
- **SUs not utilized in the previous 6 months are lost**
- Purchased cycles can be spent over 18 months.



Policy on allocating UCI-paid cycles

Ideal – every cycle allocated is utilized

Allocation Tiers for CPU Cores (6 months horizon):

- 100K, 75K, 50K, 25K, 12.5K

Your next allocation is based on your previous 6 months of usage

- > 80% of current allocation utilized, go up one tier
- 50% - 80%. Remain in same tier
- 25% - 50%. Go down on tier
- < 25% go down two tiers

Limits

- Philosophy

  Allow users to do what they need to do.
  Generally, only place limits to address: stability, fairness, responsiveness

- Example System-wide limits

  MaxArraySize = 100000
  MaxJobCount = 50000

- When we see a file system "under stress"

  1. Identify user/users
  2. Contact them to find out "what their applications are doing"
  3. Determine if

     limits (like maxjobs or maximum cores) are needed to mitigate
     or
     can a restructuring of jobs address the issue

- Example user-specific limit:

```
   Account        User  Partition     Share  MaxJobs       GrpTRES
----------- ---------- ---------- --------- ------- -------------
   uci_lab   panteater                    1     300       cpu=800
```

⚠ **The most vulnerable part of any cluster is storage**

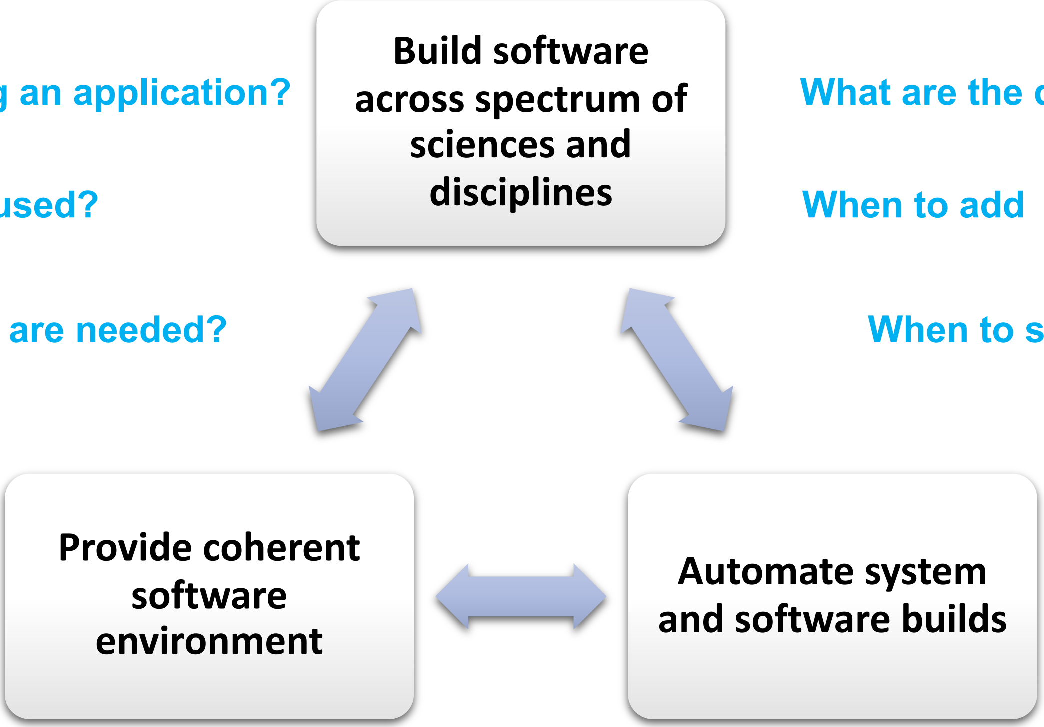
Software environment on HPC3

Software Applications

Who is using an application?

How often is it used?

Which versions are needed?



Build software across spectrum of sciences and disciplines

Provide coherent software environment

Automate system and software builds

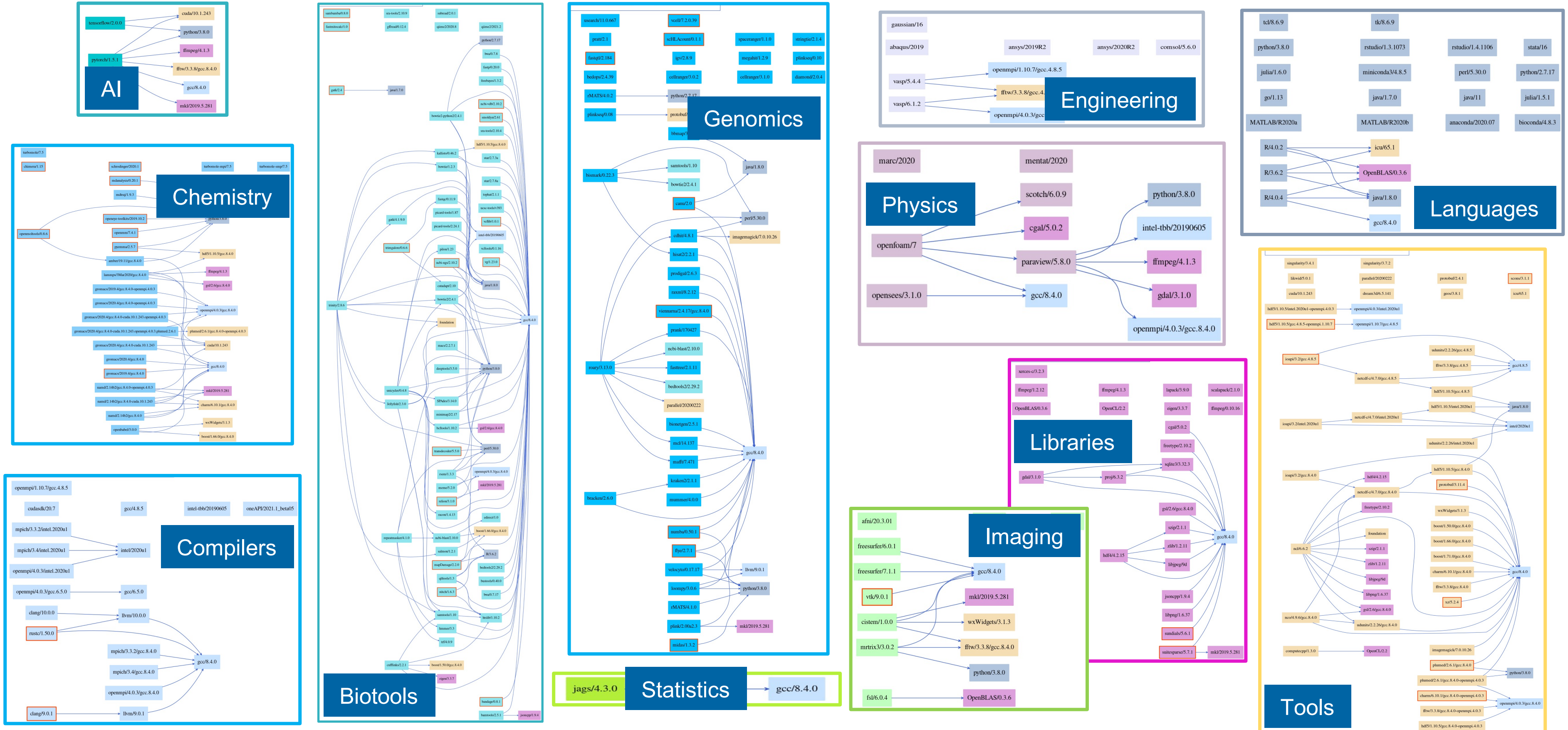
What are the dependencies?

When to add or update?

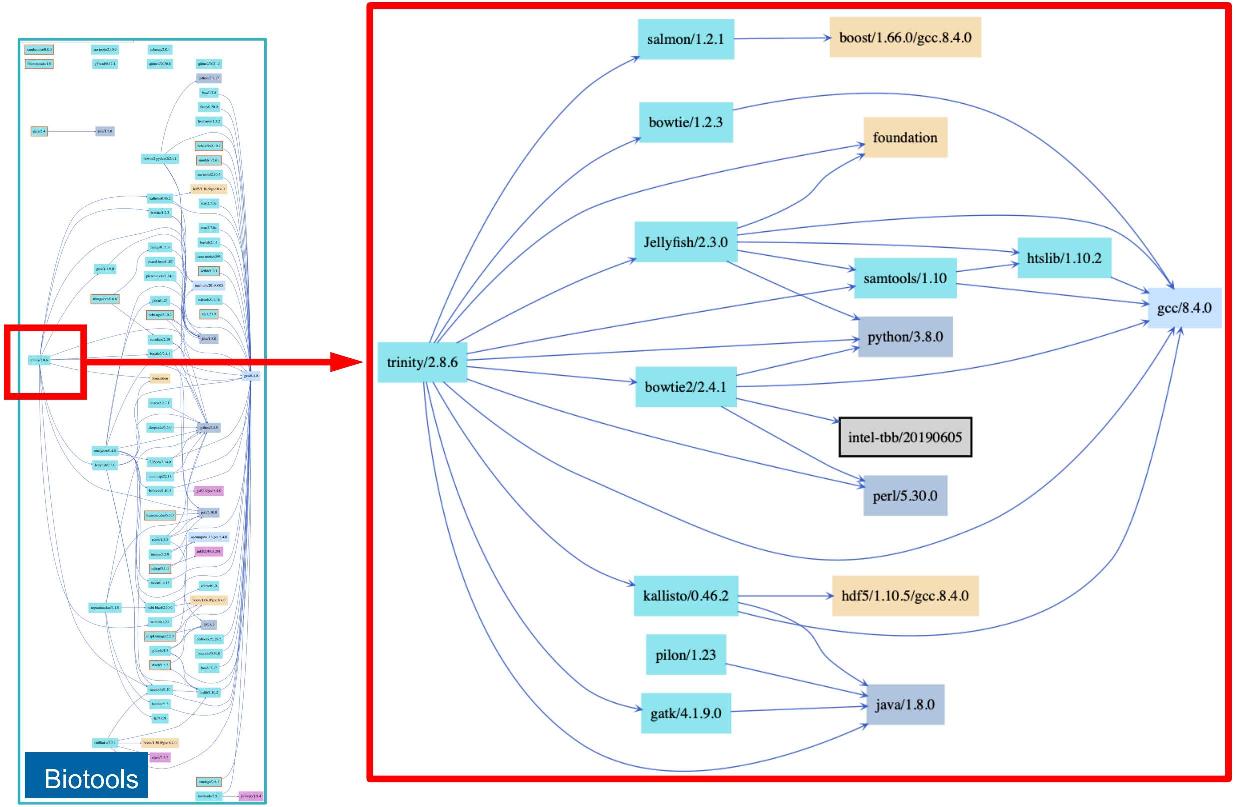
When to sunset?

We package most applications in the OS-native format: RPM

Software Map



Software Map Detail



- Some apps have very deep dependencies
- Capture dependencies during the build
- Enable auto loading of dependencies
- User needs to load a single module:

module load trinity/2.8.6

Module Dependencies

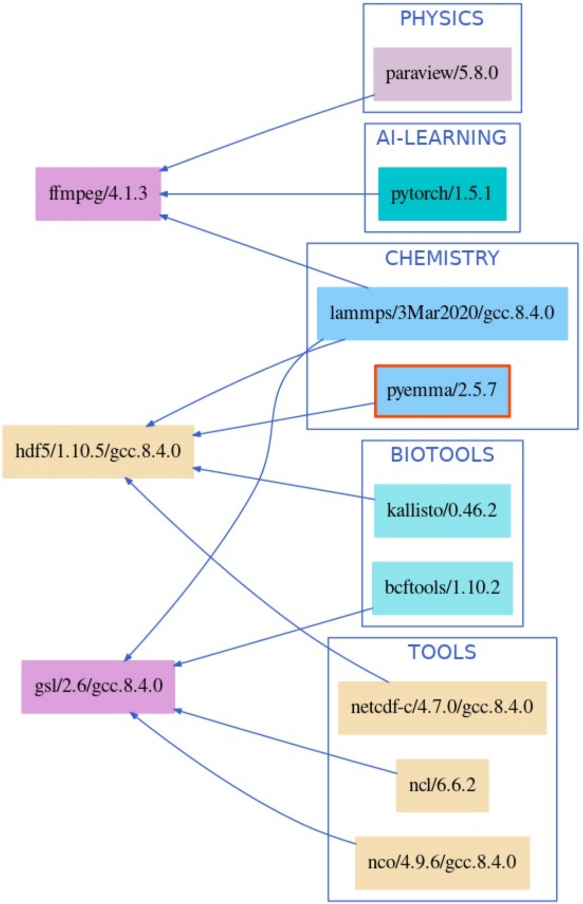
AI-LEARNING:	2
BIOTOOLS:	63
CHEMISTRY:	21
COMPILERS:	20
ENGINEERING:	7
GENOMICS:	40
IMAGING:	9
LANGUAGES:	22
LIBRARIES:	24
PHYSICS:	6
STATISTICS:	1
TOOLS:	42

Most used modules since Jan 1, 2021

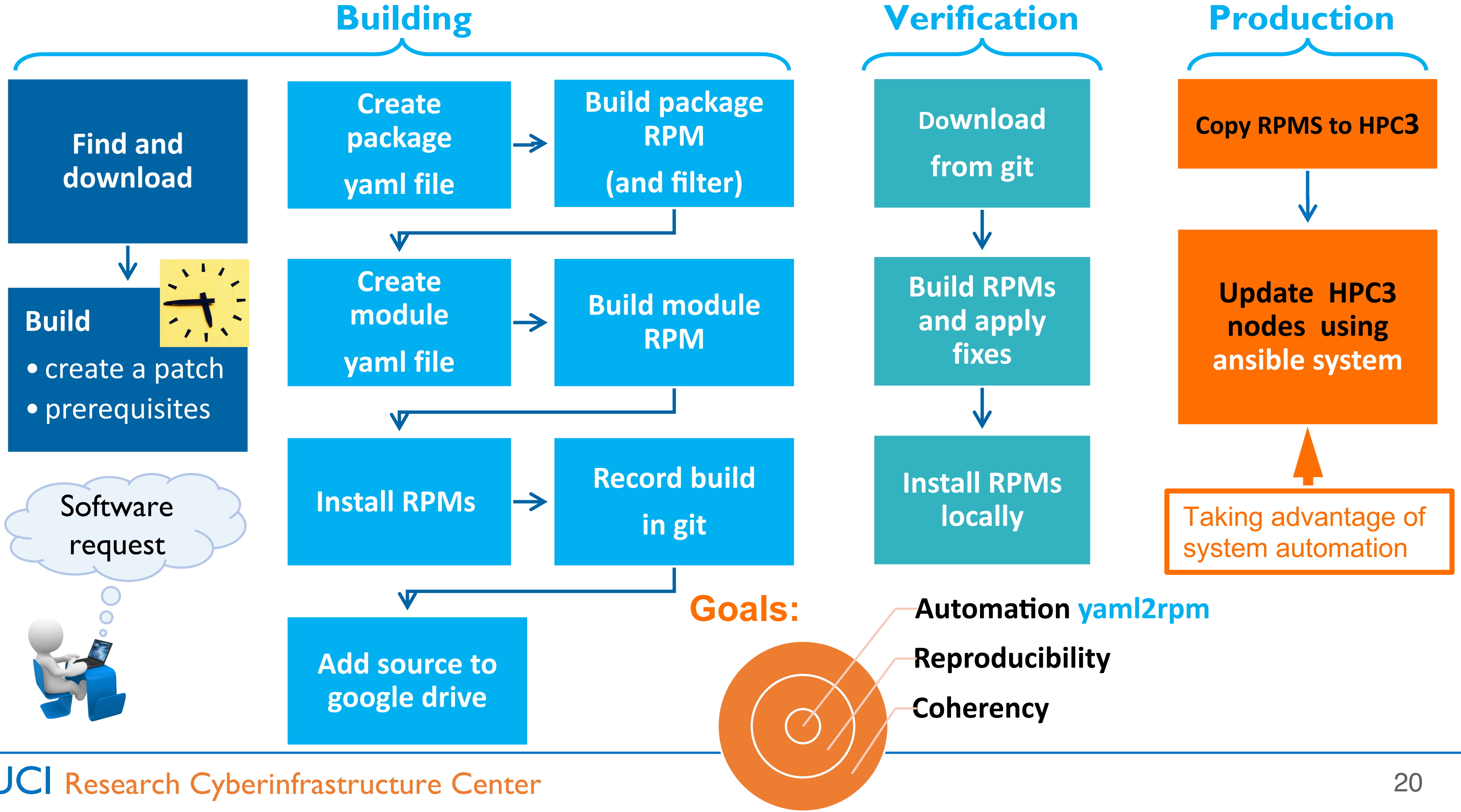
python/3.8.0	701,106
gcc/8.4.0	408,170

Updating a single package

Can affect many others, how?
Which ones ?



What we do to install software



Building

Find and download

Build
- create a patch
- prerequisites

Software request

Create package yaml file → Build package RPM (and filter)

Create module yaml file → Build module RPM

Install RPMs → Record build in git

Add source to google drive

Verification

Download from git

Build RPMs and apply fixes

Install RPMs locally

Production

Copy RPMS to HPC3

Update HPC3 nodes using ansible system

Taking advantage of system automation

Goals:

- Automation yaml2rpm
- Reproducibility
- Coherency

Modules Usage (since Jan 1)

Modules:

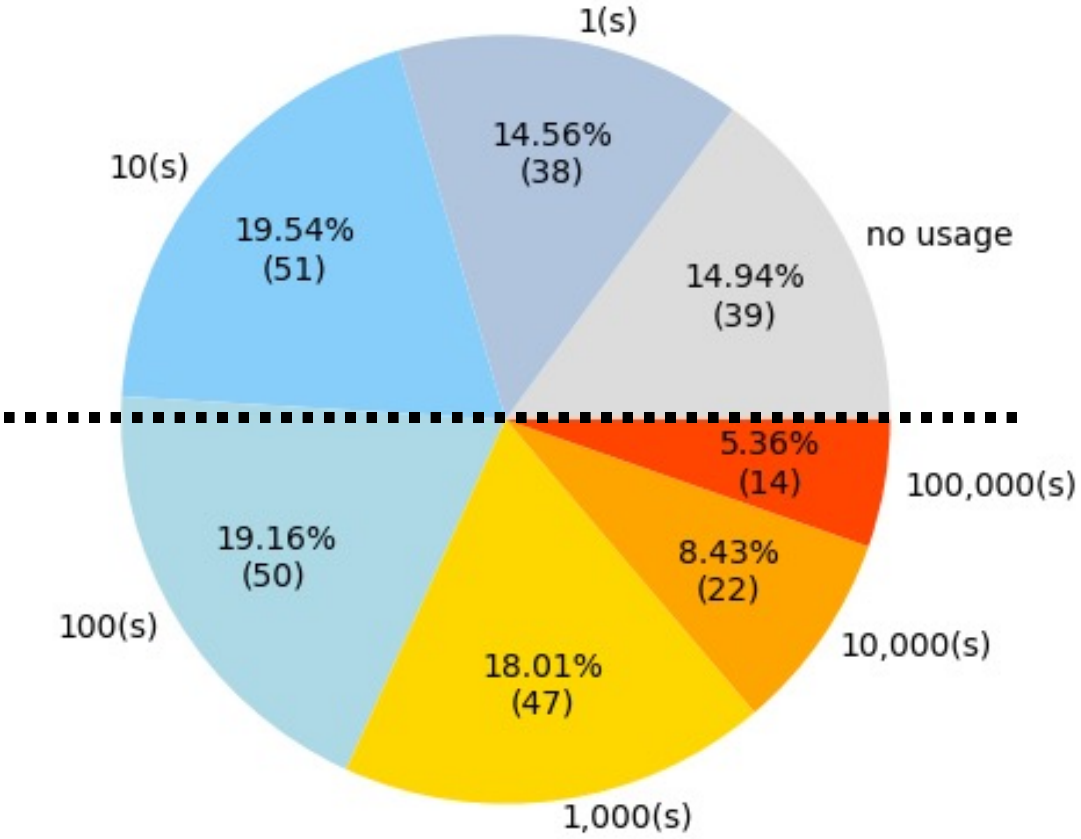
- **261 built**
- **39 unused**
- **6 user-authored**

RPMs:

- **~1700 built and installed**

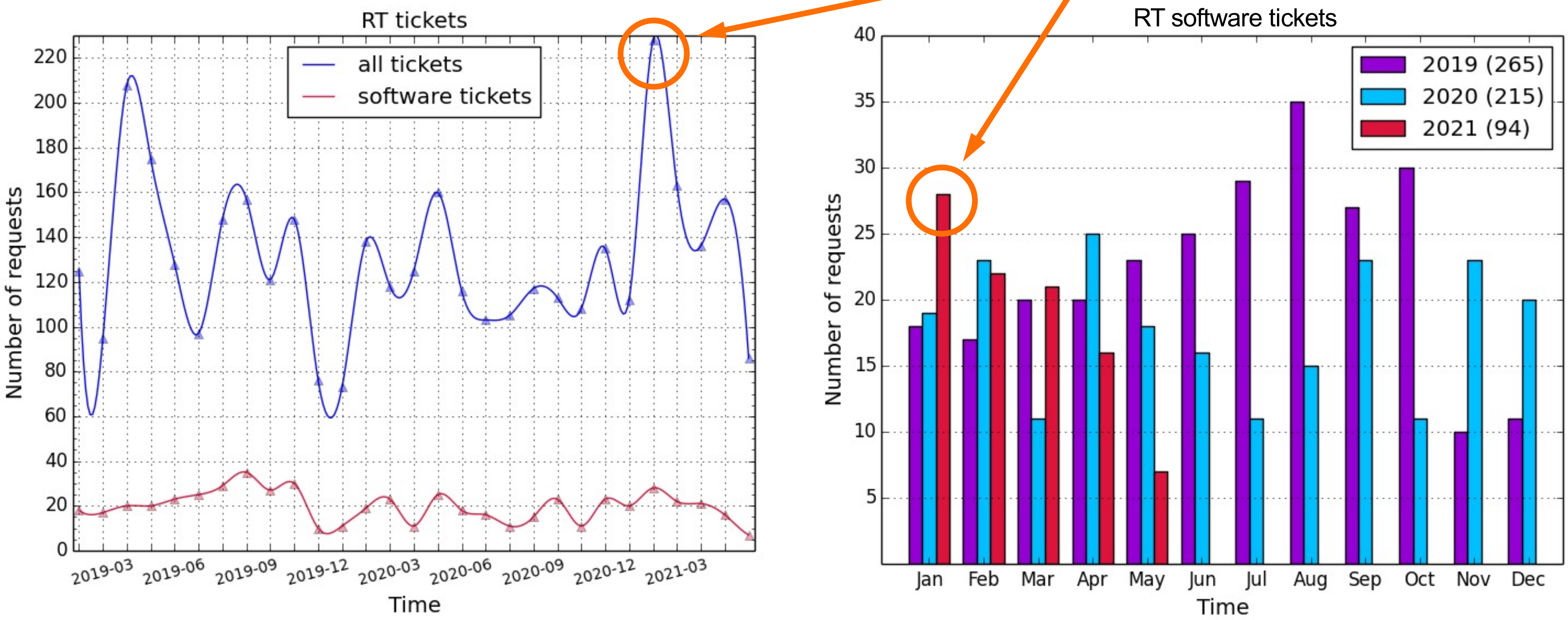
Summary of tracking:

1. All software was requested
2. Fair fraction (~ 49%) is rarely or never (15%) used
3. Helps to answer the question *“When to install, update, or sunset the software”*



Request Tracker (RT) History

Start: 2019-01-01
End: 2021-05-31

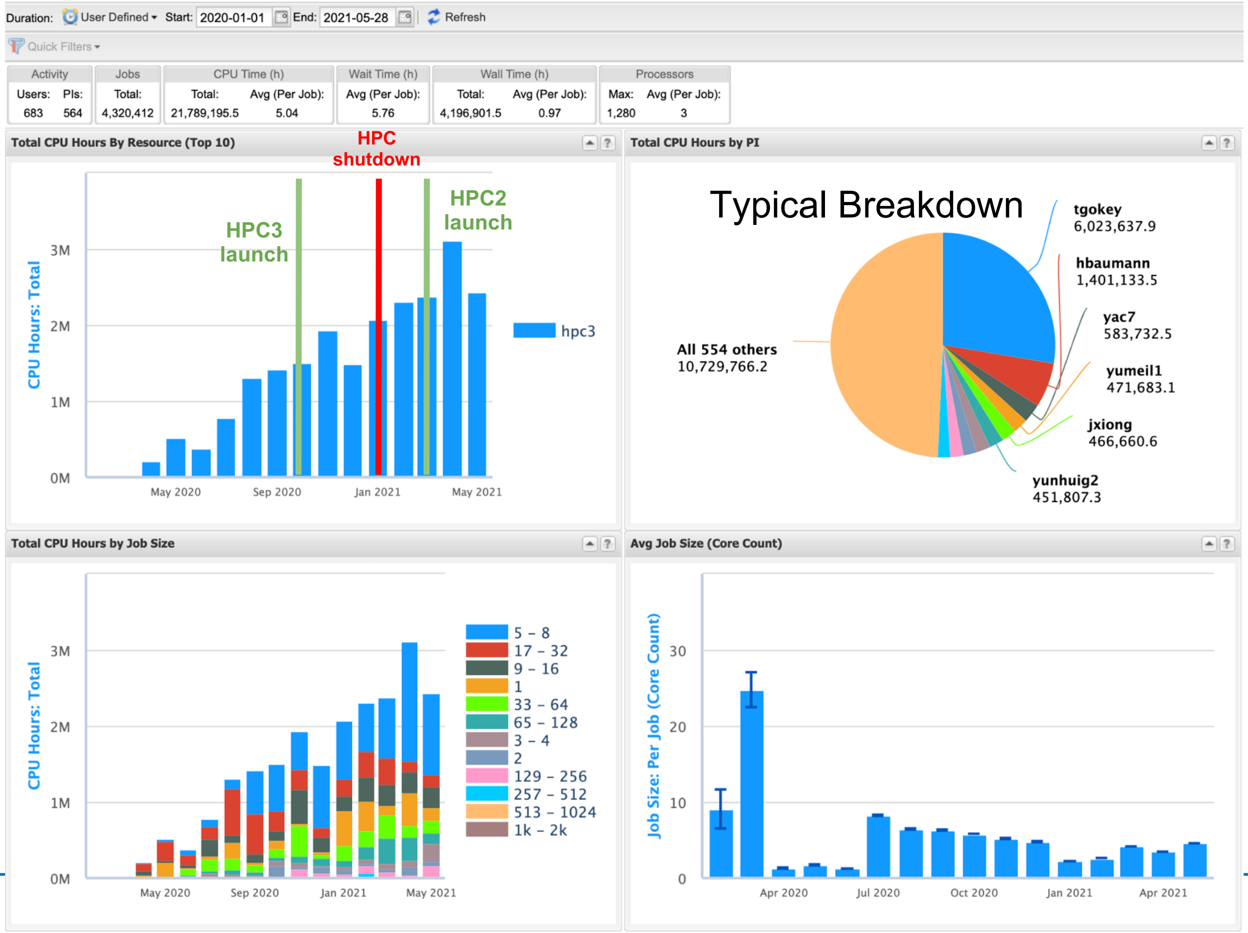


move to HPC3

Configuration automation (Ansible) + Applications → fewer issues (and usage has increased)

How has HPC3 been used since Jan 1, 2021

CPU Use Summary: Up Trend



Duration: User Defined ▾ Start: 2020-01-01 End: 2021-05-28 Refresh

Quick Filters ▾

Activity		Jobs	CPU Time (h)		Wait Time (h)	Wall Time (h)		Processors	
Users:	PIs:	Total:	Total:	Avg (Per Job):	Avg (Per Job):	Total:	Avg (Per Job):	Max:	Avg (Per Job):
683	564	4,320,412	21,789,195.5	5.04	5.76	4,196,901.5	0.97	1,280	3

2/1/2020 to 5/28/2021:

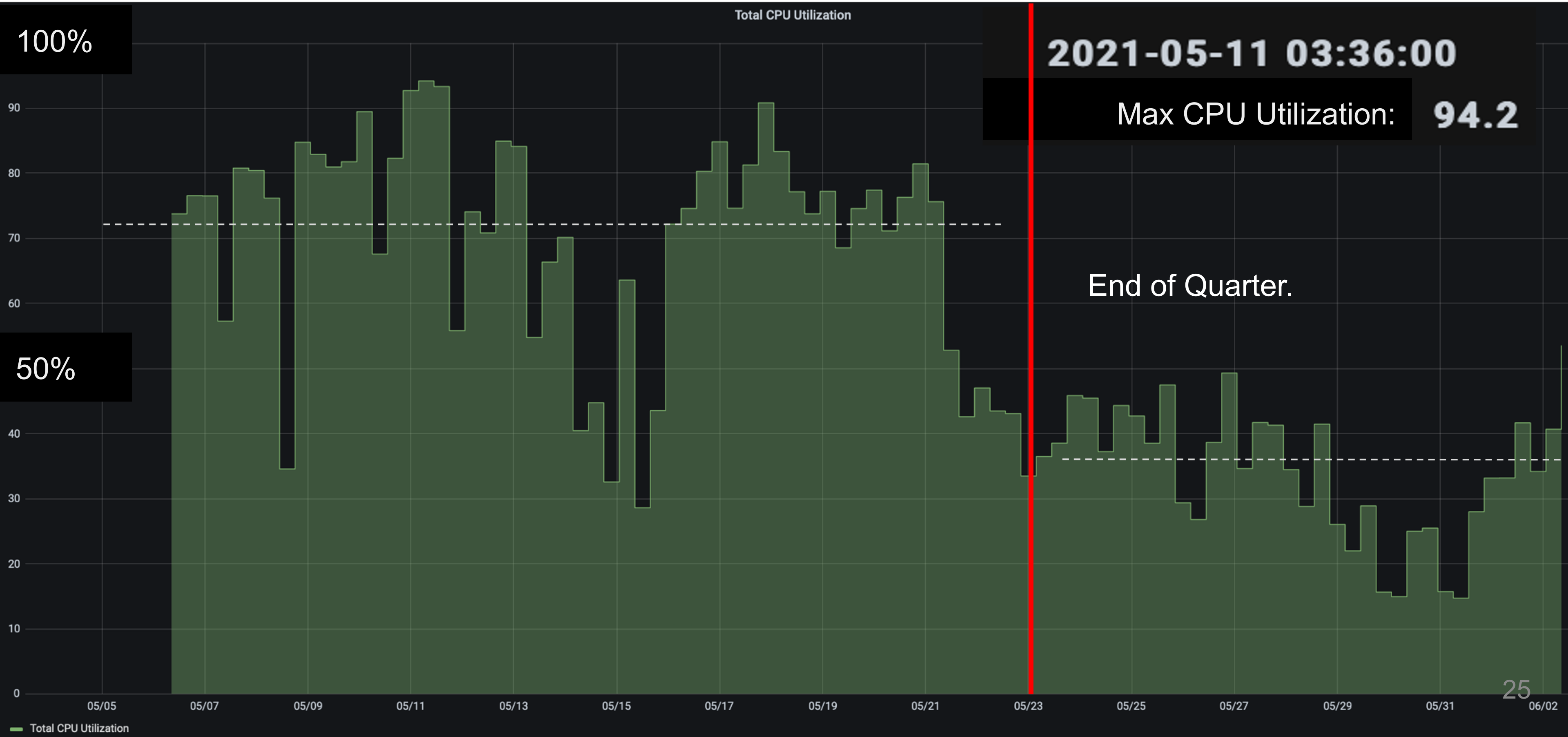
~ 500 active users
~ 1% [power] users

Total core count:
~ 4000 cores 2/2020)
~ 7000 cores 6/2021)
~ 9000 cores 9/2021)
! 16000 cores 1/2025)

~ 22M CPU hours delivered
~ 5 cores per job on average
- 1280 cores largest jobs

70% average CPU utilization

HPC3 CPU Utilization

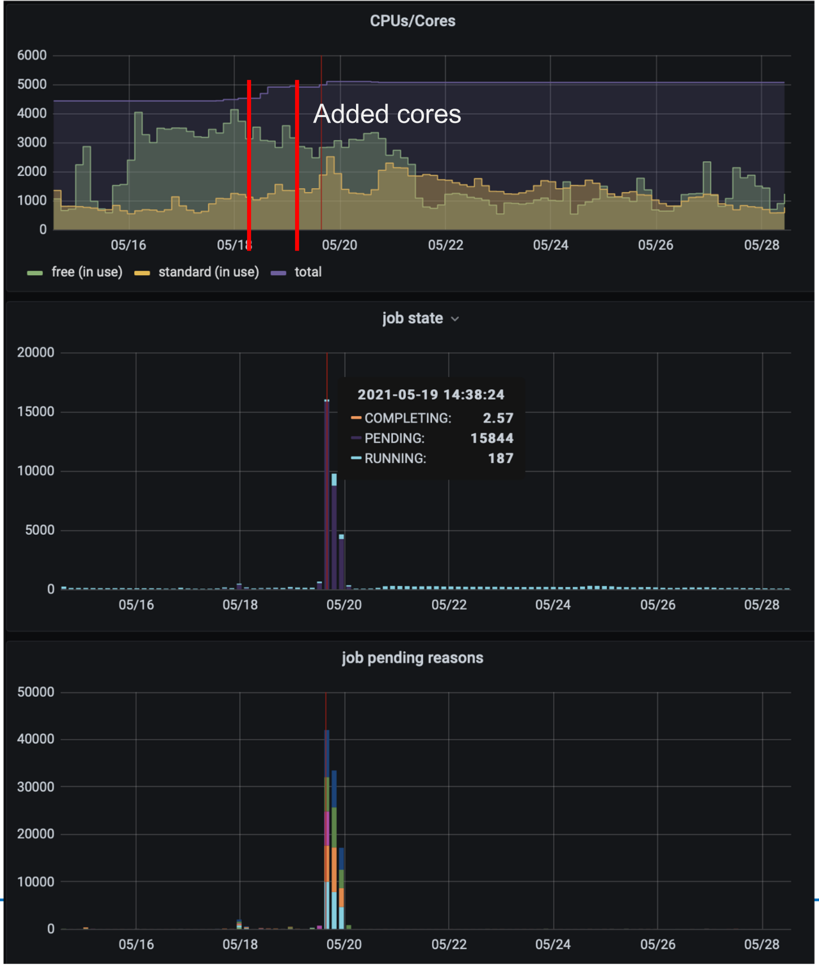


2021-05-11 03:36:00

Max CPU Utilization: 94.2

End of Quarter.

Standard Partition Jobs Summary



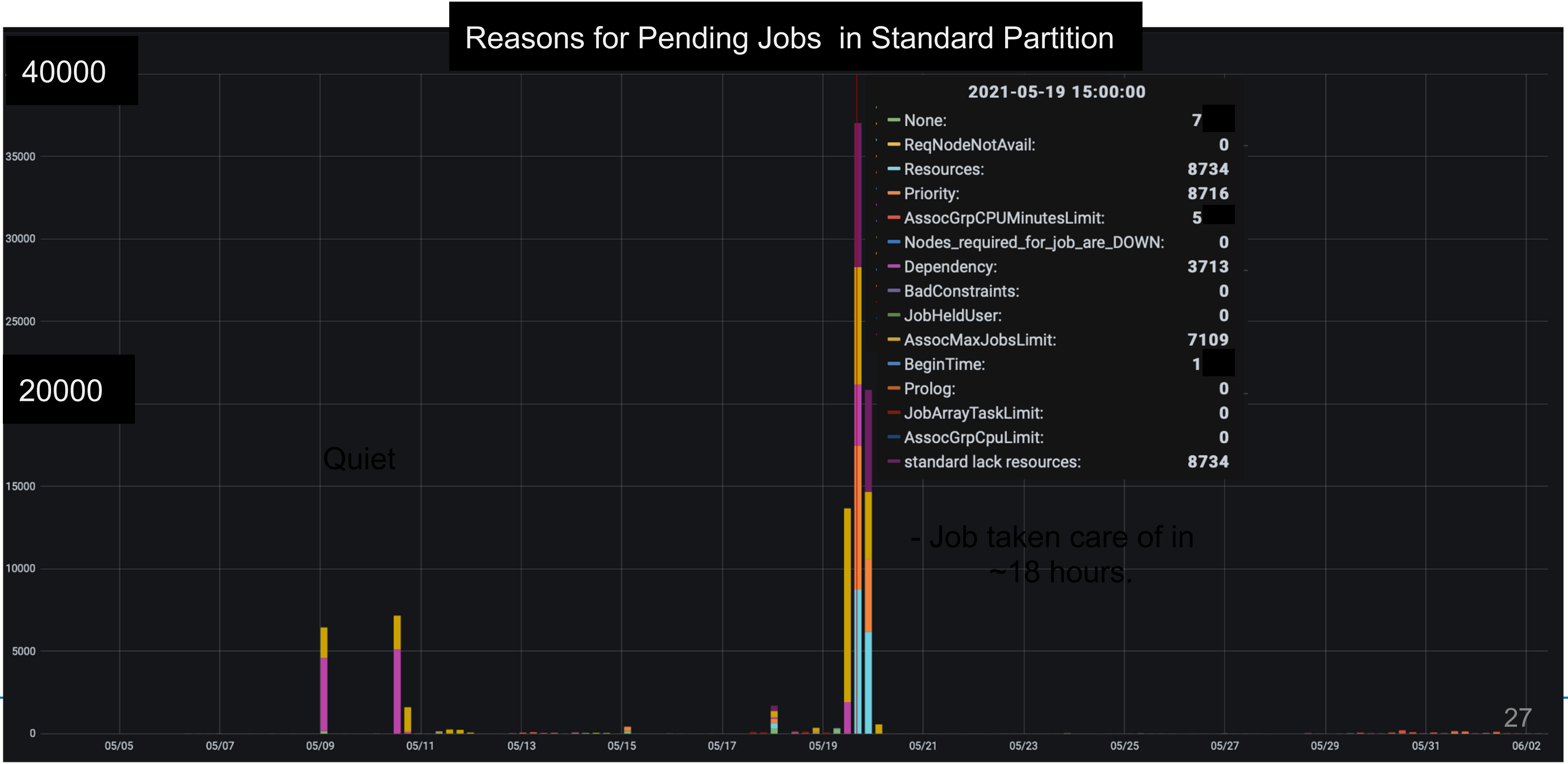
- Partition (queue) has ~5500 cores.
- Standard partition dominates by day
  - Free partition by night and weekend
- ~ 40% of jobs go through standard partition
  - Users not spending their allocations

Waiting for a job a Standard job to run is rare:

Example: Standard Pending Jobs on 5/19/21 ~ 14:35pm
~ 16,000 cores requested

Reason for pending job? See next slide...

Standard Jobs Wait Time: Minimal



Reasons for Pending Jobs in Standard Partition

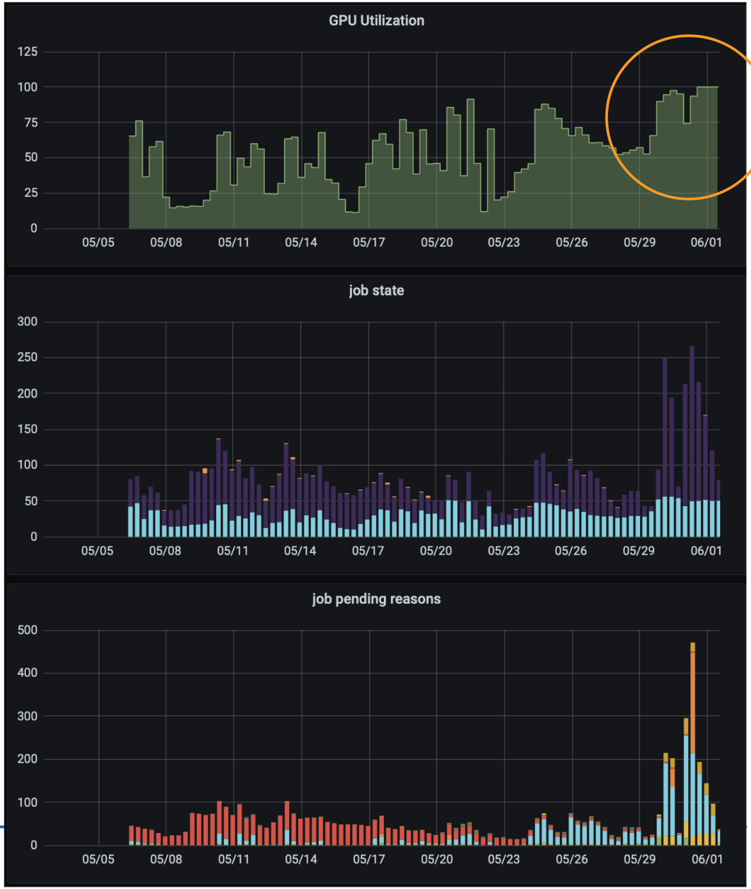
2021-05-19 15:00:00

None:	7[redacted]
ReqNodeNotAvail:	0
Resources:	8734
Priority:	8716
AssocGrpCPUMinutesLimit:	5[redacted]
Nodes_required_for_job_are_DOWN:	0
Dependency:	3713
BadConstraints:	0
JobHeldUser:	0
AssocMaxJobsLimit:	7109
BeginTime:	1[redacted]
Prolog:	0
JobArrayTaskLimit:	0
AssocGrpCpuLimit:	0
standard lack resources:	8734

Quiet

- Job taken care of in ~18 hours.

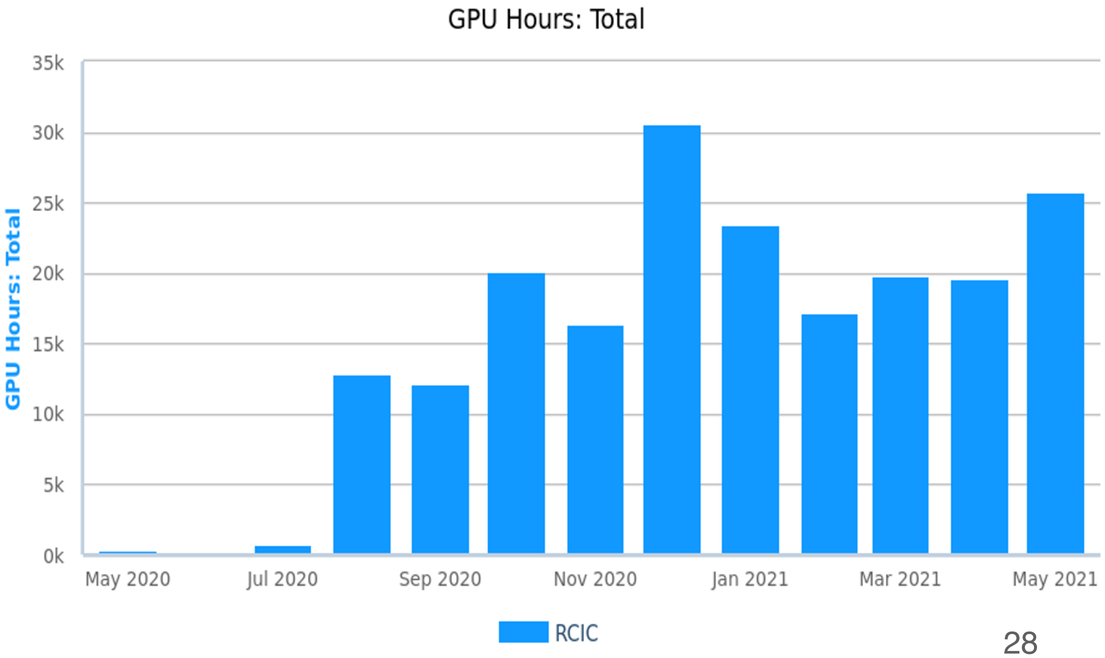
GPUs Summary



~ 14 nodes = 56 GPUs (V100)
~ 170,000 total gpu hours
~20k/month
~ 65% avg gpu utilization

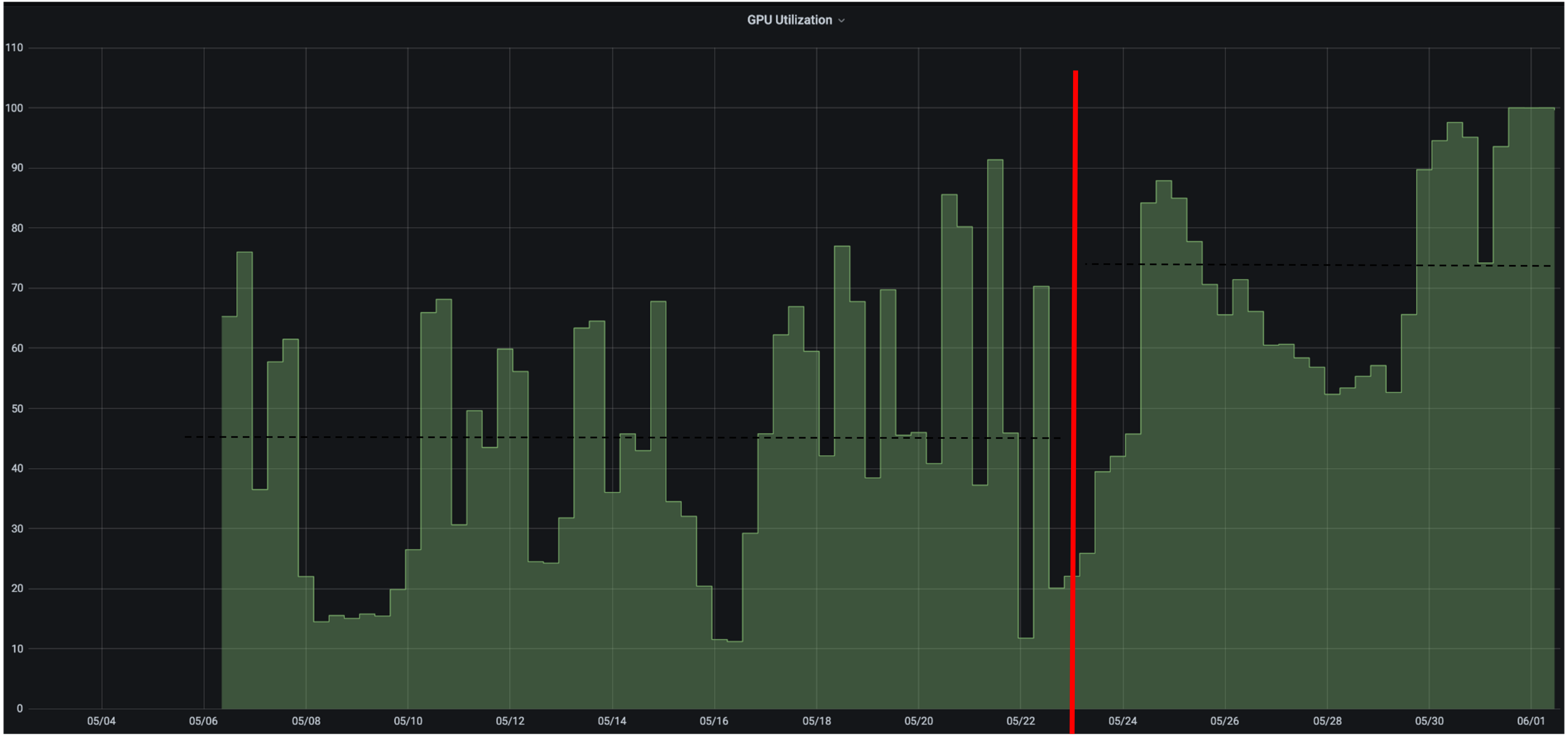
- Increasing as people learn how to use them
- There is a longer wait for GPUs than CPUs

=> Need more GPUs



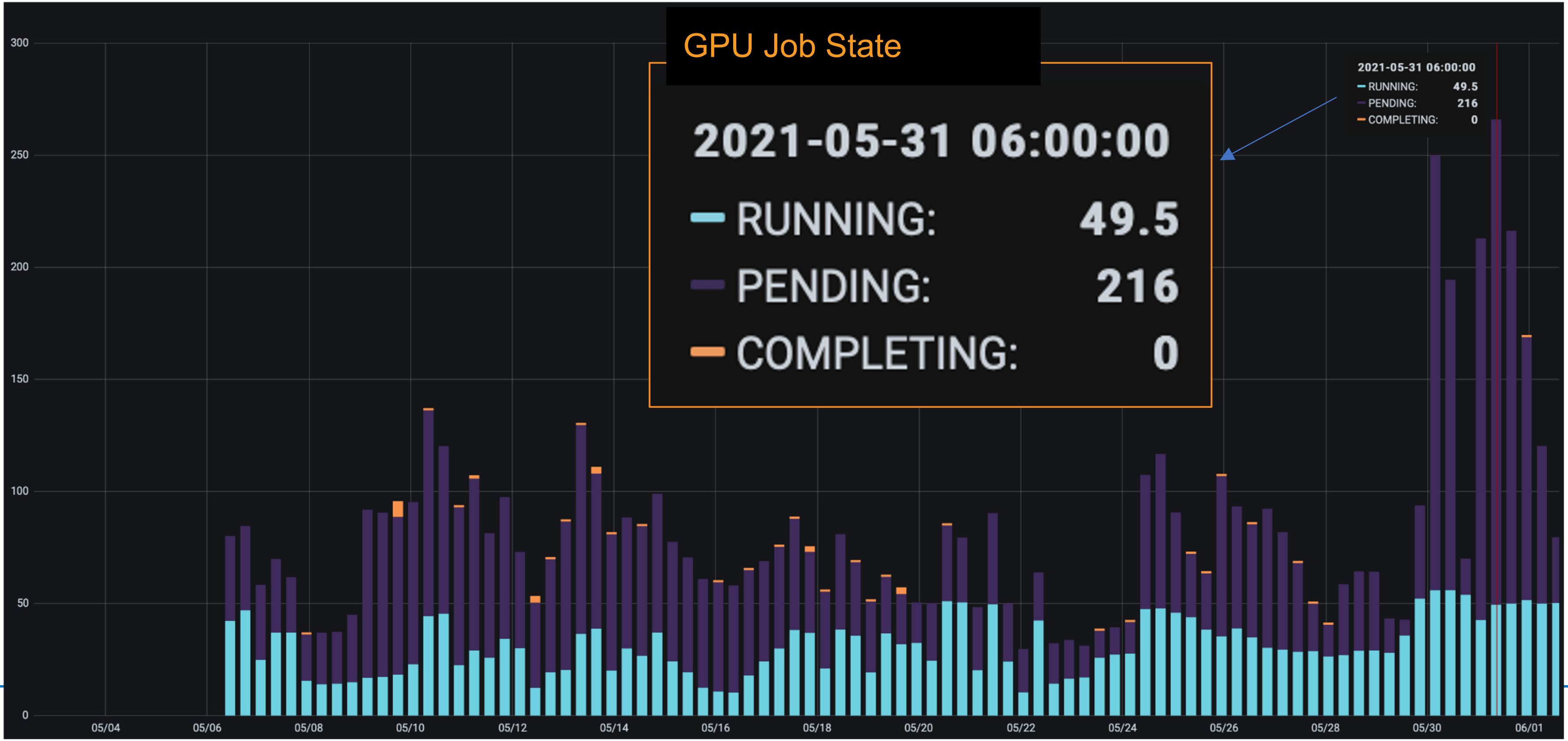
2020-05-01 to 2021-05-31 Src: HPcDB. Powered by XDMoD/Highcharts

GPU Partition Utilization - Detail

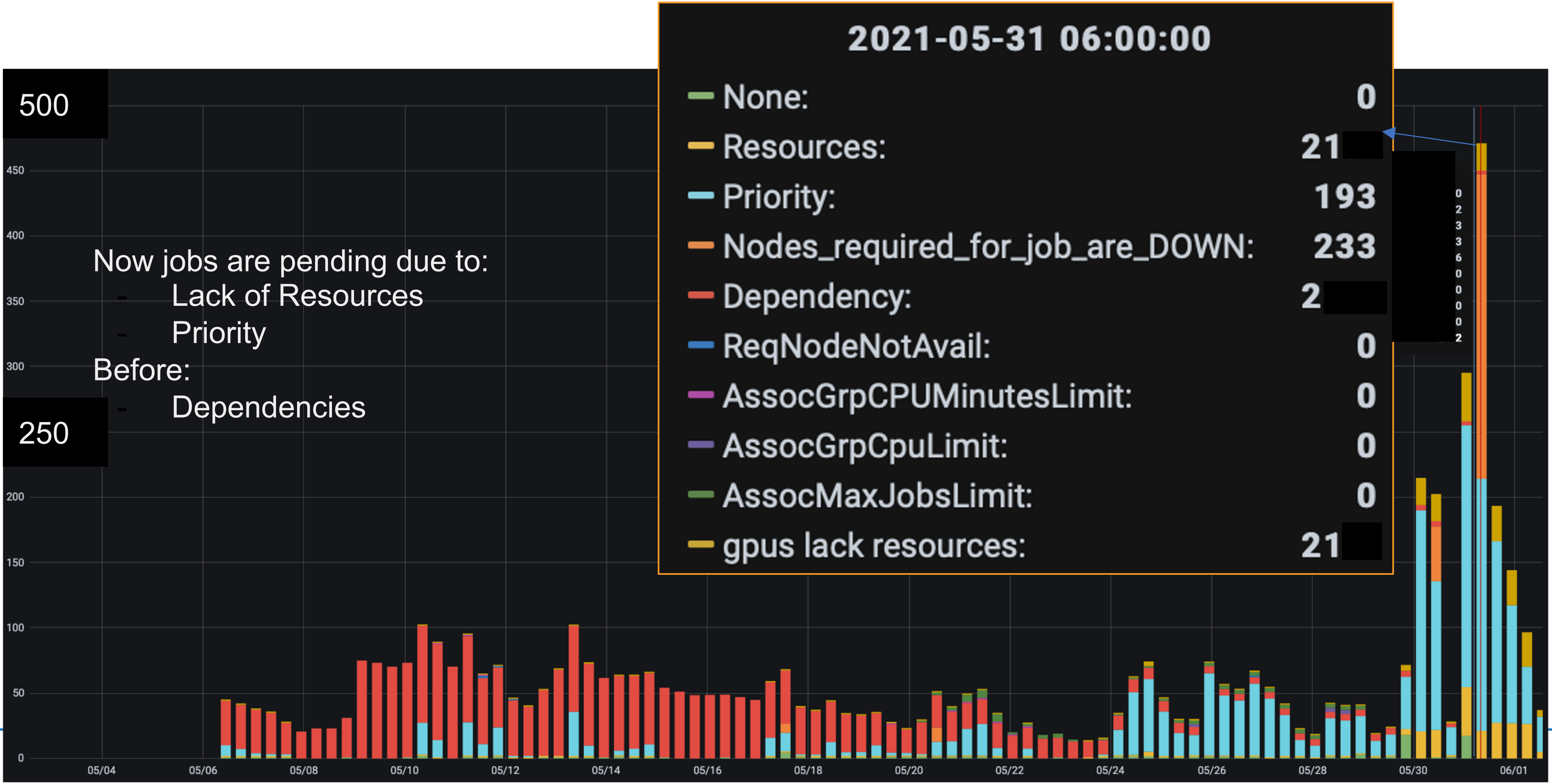


GPU Partition Job State

Different from Standard Partition: there are Always Pending Jobs



Reasons for Pending Jobs in GPU Partition



2021-05-31 06:00:00

None:	0
Resources:	21[redacted]
Priority:	193
Nodes_required_for_job_are_DOWN:	233
Dependency:	2[redacted]
ReqNodeNotAvail:	0
AssocGrpCPUMinutesLimit:	0
AssocGrpCpuLimit:	0
AssocMaxJobsLimit:	0
gpus lack resources:	21[redacted]

[redacted]

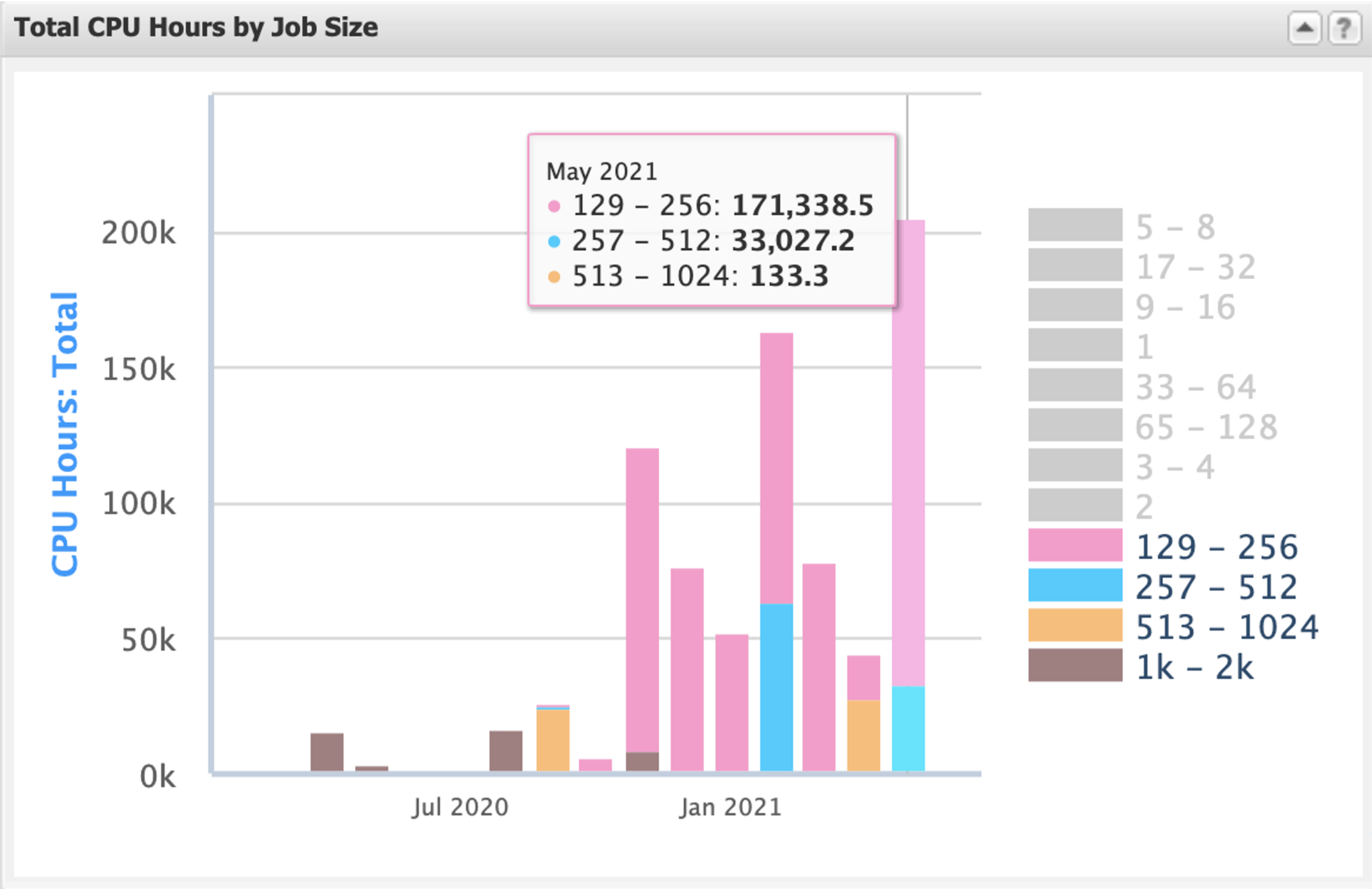
Now jobs are pending due to:

- Lack of Resources
- Priority

Before:

- Dependencies

Large Multi-node Jobs: No special provisions needed to run



10^3 range in core requests

- Avg. Job size: 5 cores

- Max Job size: 1280 cores

Large Jobs [129, 1280] cores

- Use ~3% of CPU time
- Reasonable wait time

Talking to RCIC and to Each Other

- **How do I ask for help/talk to RCIC?**
  - Send email to hpc-support@uci.edu

    This automatically creates a help ticket
  - Read that fine website: https://rcic.uci.edu

- **What about talking to RCIC and the other users at UCI?**
  - Join the new! Google group

    https://groups.google.com/a/uci.edu/g/rcic-users
  - Chat with us on Slack: https://rcicos.slack.com/

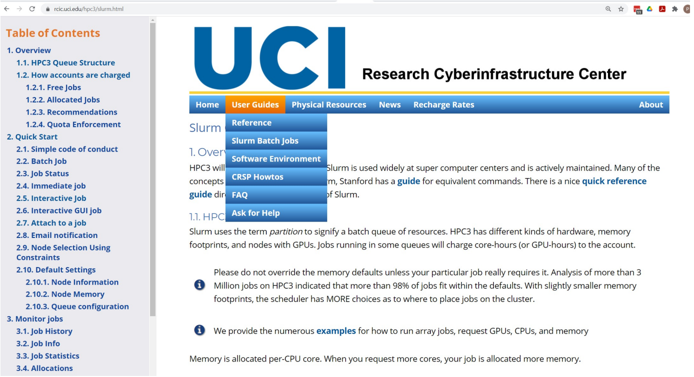


Table of Contents

Slurm

1. Over

HPC3 will Slurm is used widely at super computer centers and is actively maintained. Many of the concepts m, Stanford has a **guide** for equivalent commands. There is a nice **quick reference guide** dir of Slurm.

1.1. HPC

Slurm uses the term *partition* to signify a batch queue of resources. HPC3 has different kinds of hardware, memory footprints, and nodes with GPUs. Jobs running in some queues will charge core-hours (or GPU-hours) to the account.

ℹ Please do not override the memory defaults unless your particular job really requires it. Analysis of more than 3 Million jobs on HPC3 indicated that more than 98% of jobs fit within the defaults. With slightly smaller memory footprints, the scheduler has MORE choices as to where to place jobs on the cluster.

ℹ We provide the numerous **examples** for how to run array jobs, request GPUs, CPUs, and memory

Memory is allocated per-CPU core. When you request more cores, your job is allocated more memory.

Resources

- Github repositories for the software builds

  https://github.com/RCIC-UCI-Public
- RCIC website   https://rcic.uci.edu