Cluster Resource Management for Sustainable and Efficient Computing

Andrei Bersatti – School of ECE
Aaron Jezghani – PACE
J. Eric Coulter – PACE

Georgia Institute of Technology

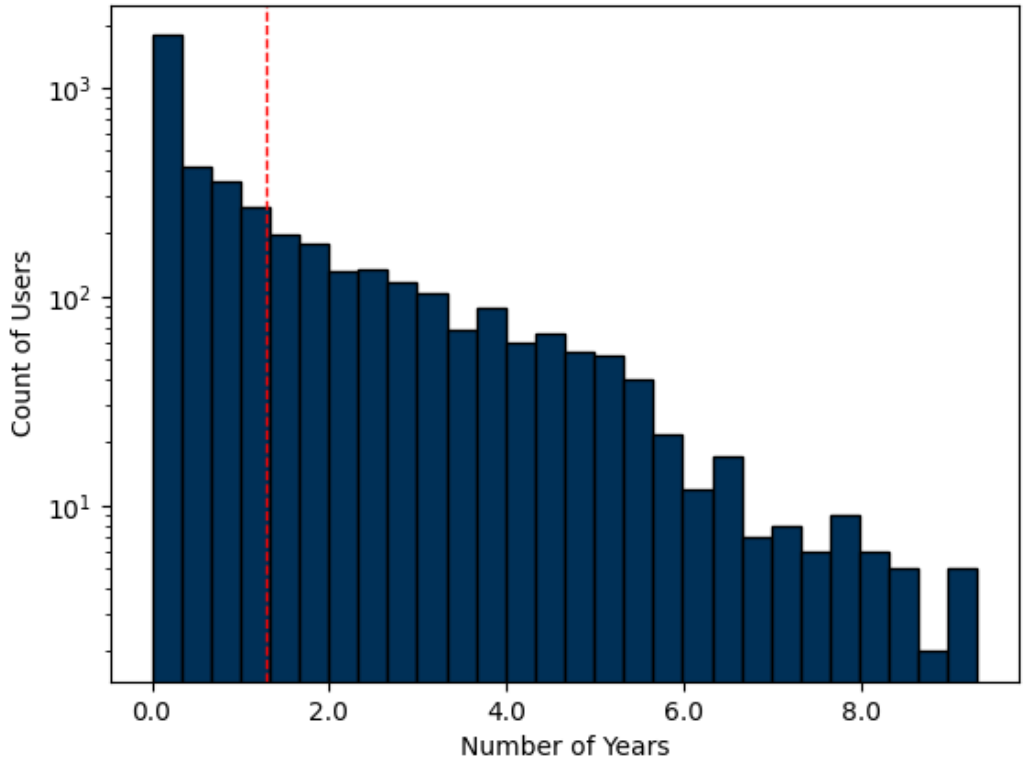
Georgia Tech College of Engineering
School of Electrical and Computer Engineering

I. Introduction

- Need for Energy Efficiency:
  - Carbon Footprint
  - New technologies and techniques
    → Resource hungry applications.
  - Increased efficiency
    → Jevon's Paradox
    → Increased demand
  - Utilization (85%)
- Challenges:
  - Hardware and Software considerations
  - Transparency in reporting
  - Specific to Academia

Phoenix in Coda datacenter in Midtown Atlanta.

I. Introduction



Distribution of User lifetimes on GT research clusters (time between first and last job submission). The average user lifetime is only 1.38 years. Data spans 2015 through present.

II. Phoenix Cluster at GT

- April ‘21 - Transition away from Condominium:
  - Dedicated resources → Long job wait times
  - Lack of flexibility → 35% allocation
  - Hardware refresh rate limited → Electricity/Rack space
  - Out of warranty/Heterogeneous → Maintenance
  - Condominium → Consumption-based model
    - Less restrictive queues
    - Larger collective pool of resources
    - First year → 35% to 55% utilization
    - Average Job Sizes → Increased 40%.
    - Average queue times → Decreased (>>1 Hr.)

II. Phoenix Cluster at GT

- Torque/Moab:
  - Moab – Workload scheduling
  - Torque – Resource Management
  - MAM – Accounting
  - Late 2021 updated scheduling policy → allow spillover of smaller memory jobs to idle larger memory architectures.
  - Use Node feature sets → Align jobs based on InfiniBand fabric topology → prioritized less dense racks → better locality.
- Challenges:
  - Newer OS than RHEL7 incompatibility friction with newer architectures.
  - Communication failures (scheduler components) → increased timeouts → reduced overall throughput.

II. Phoenix Cluster at GT

- Cluster Architecture:
  - 1,400 Servers
  - 35,000 CPU Cores
  - 330 GPUs
  - 7 PB mixed storage
  - Nodes → 100 Gbps InfiniBand
    → QM8700 → 8x100 Gbps
    → CS8500 director switch
  - Mainly Intel Cascade Lake (192GB, 348GB, 768GB)
  - Local Storage: NVMe (1.5TB), SAS (8.0TB)
  - 2x Nvidia Tesla V100-PCIe (16GB, 32GB);
    4x Nvidia Quadro Pro RTX6000
    Nvidia Ampere A100 (40GB, 80GB PCIe)
    Hopper H100 (SXM5)

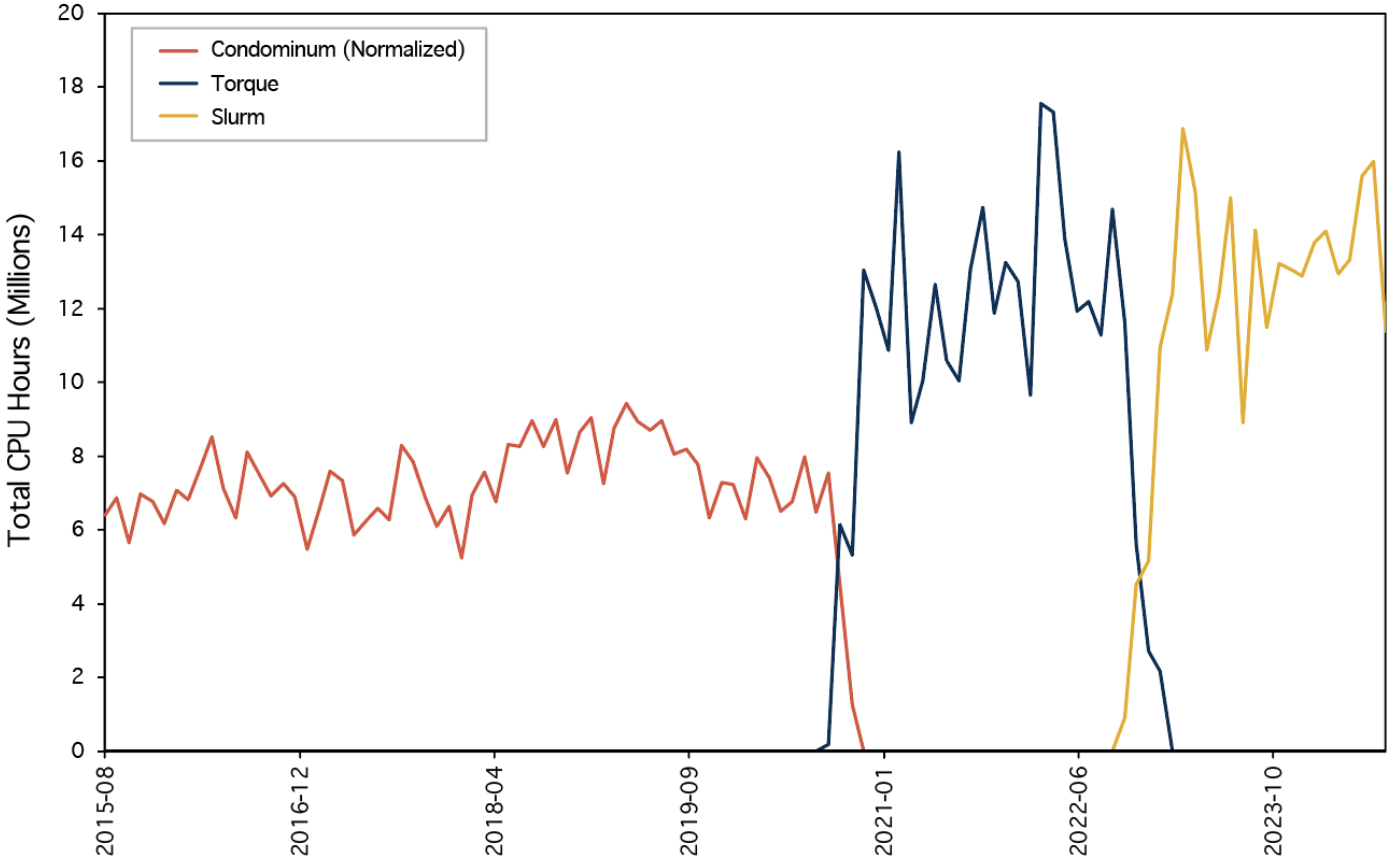
III. Migration to SLURM

- 2022 – Staged Migration to SLURM:
  - Accounting: Leverage TRESBillingWeights on a per-partition basis → set charge rate per node class → 10 partitions for charge rates.
  - Accounting: Switch from QOS to partitions under SLURM → "inferno" and "embers" submissions moved from Torque queues to SLURM QOS designations.
  - Enhancements and opportunities:
    - Transition to local PMIx installation
    - Hwloc and NVML support where appropriate → intra-node resource affinity
    - Enabled GPU SM and memory occupancy included in TRESUsageIn
    - Pam_slurm_adopt → addressed resource contention by interactive sessions on computational nodes → Places external login sessions under the most recent job step
    - -prefer flag in LUA job submit plugin → re-prioritizes node class features over topology → Updated job submit plugin: 1. append constraints requested and 2. include conditional logic to ensure viability of node class option.

III. Migration to SLURM

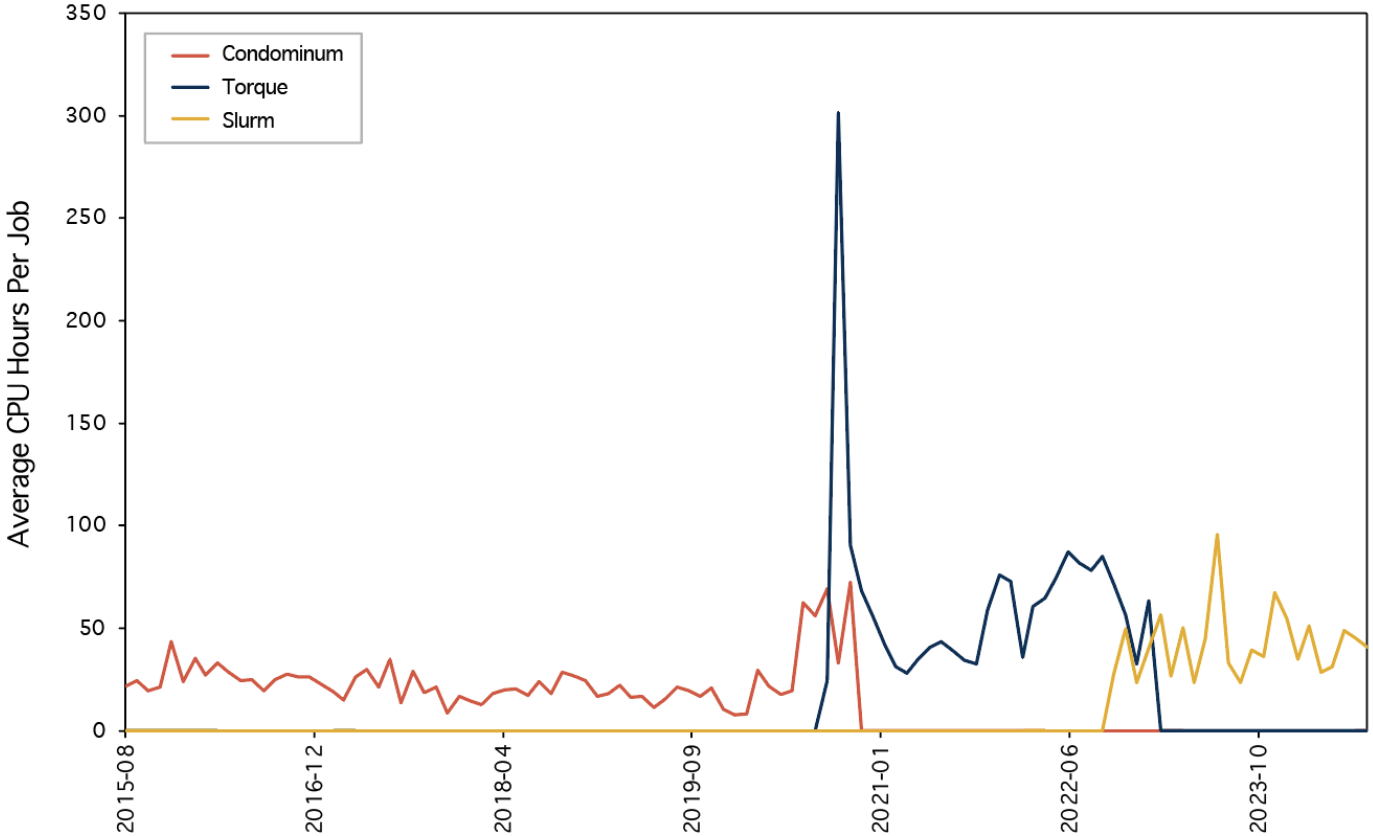
- 2022 – Staged Migration to SLURM:
  - Interactive CPU Partition:
    - On-demand resources for interactive code development and debugging is extremely bursty.
      - 1. SLURM supports partition-based over-subscription.
      - 2. SLURM supports prolog to manage kernel niceness on the compute node
      - 1 and 2 together allow for an interactive CPU partition atop the GPU nodes while preserving execution priority for the jobs specifically needing GPUs.
      - Use niceness to relinquish interactive jobs to GPU jobs as appropriate.
      - Use PriorityTier=15 setting on the partition ensures that interactive CPU jobs are scheduled with high priority, as we want to ensure that these jobs are promptly executed.

Results



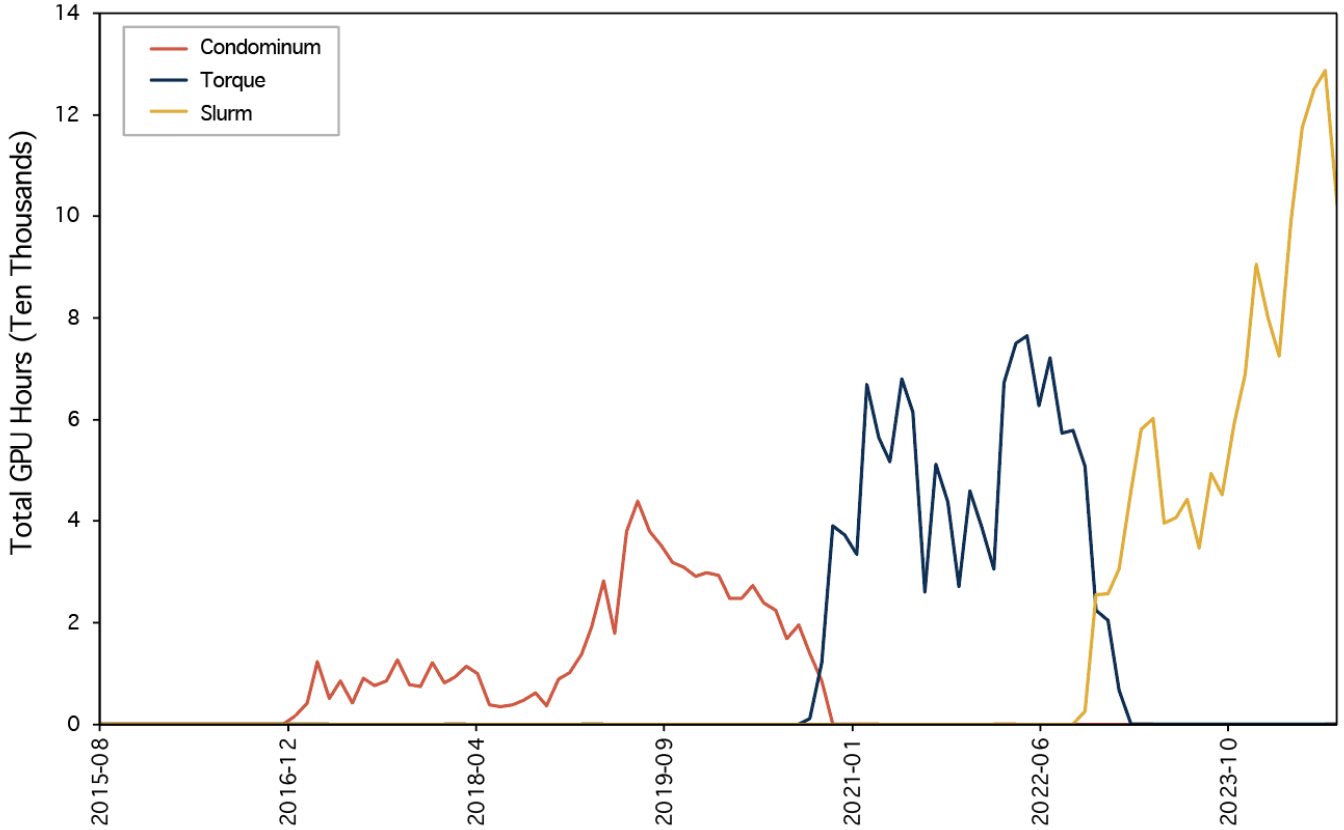
Scheduled CPU Hours Torque (navy) and Slurm (gold), with historical research cluster data under the condominium model (red) to highlight the increased utilization with the transition to the consumption model. Condominium model data is normalized by relative CPU performance following the cluster refresh.

Results



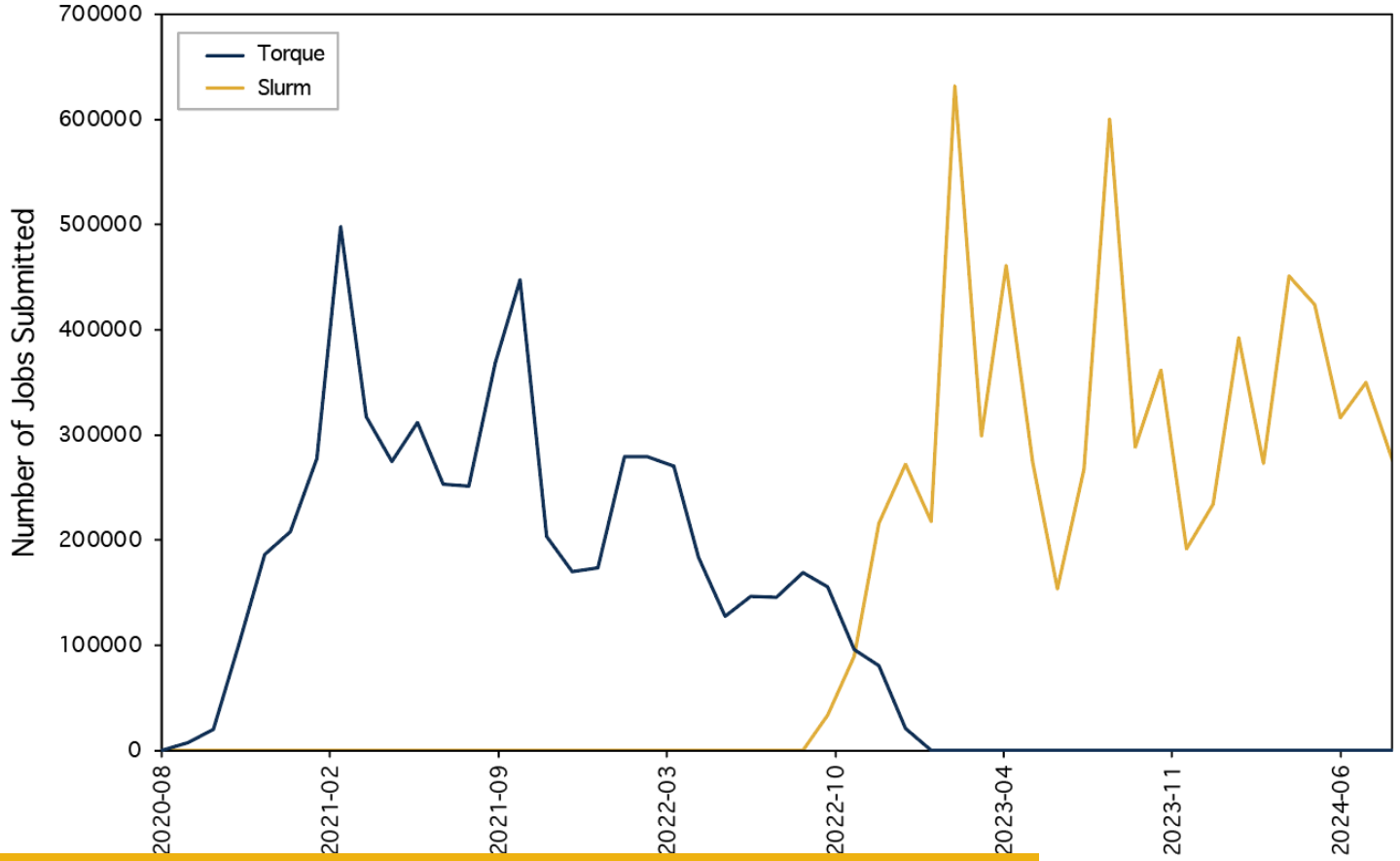
Average CPU Hours per job under the condominium model (red) and consumption model for Torque (navy) and Slurm (gold). The configurations under Torque seem to have favored larger jobs, while recent trends towards GPU compute seem to have transitioned cluster usage back to smaller jobs on average

Results



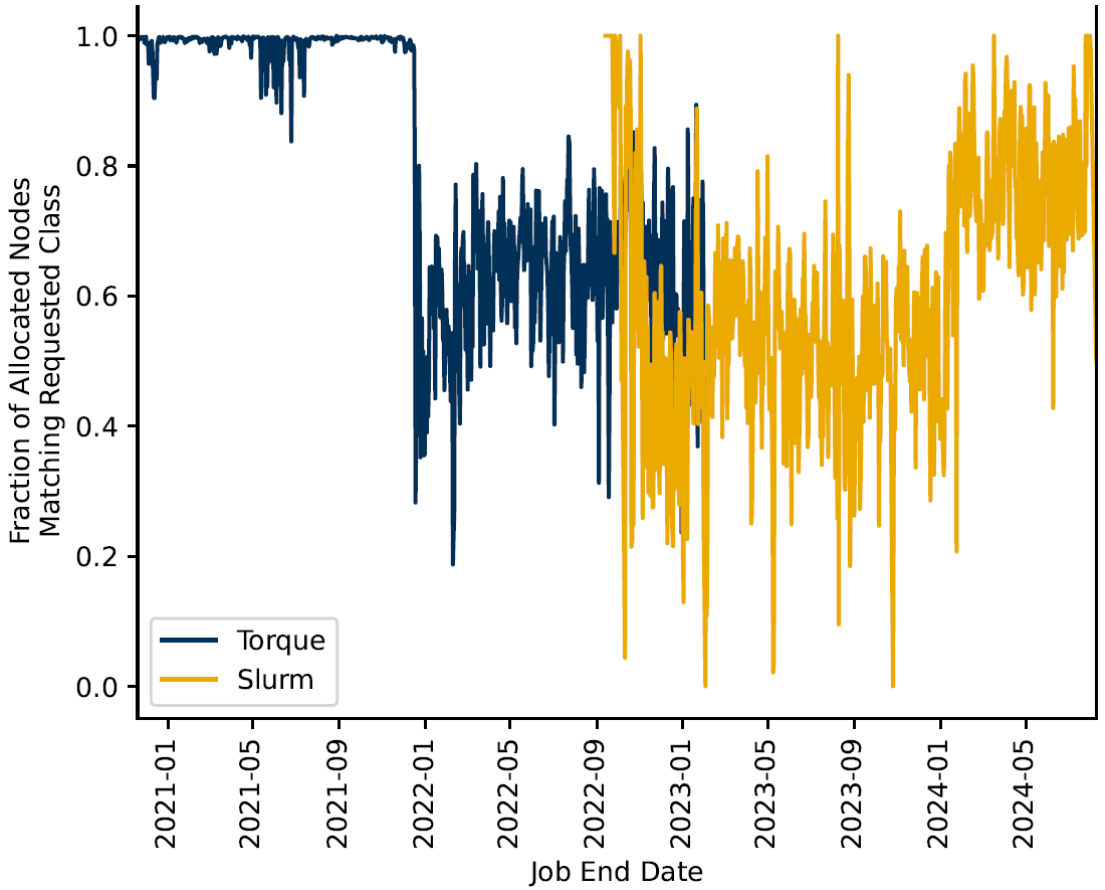
Scheduled GPU Hours under Torque (navy) and Slurm (gold), with historical research cluster data under the condominium model (red). Note the significant uptick in GPU utilization. Besides this, the capabilities of each accelerator have increased dramatically, as the architectures have evolved from NVIDIA Kepler series to the more recent Hopper series GPUs.

Results



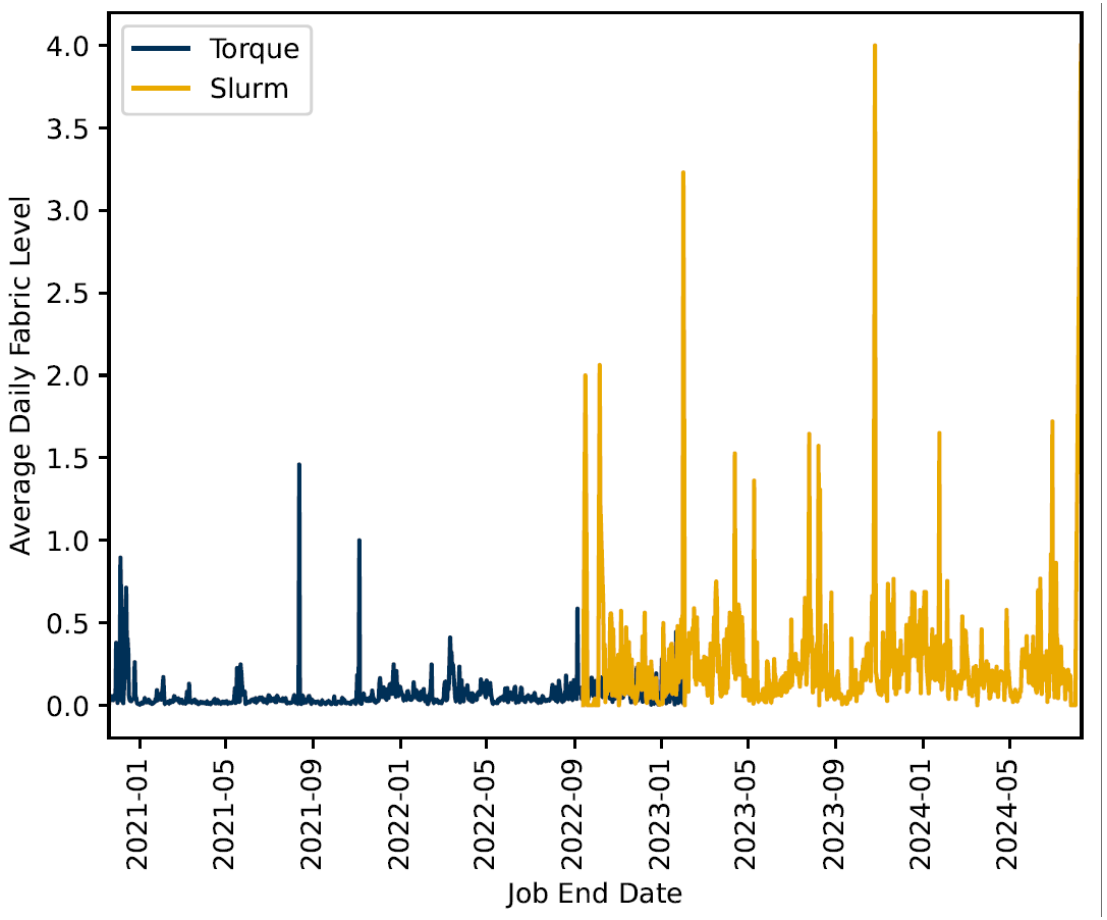
Job throughput based on number of jobs submitted to Torque (navy) and Slurm (gold). The number of jobs received each day has increased following the transition to Slurm, which we suspect can be attributed to changing user workflows as well as the more efficient scheduling, as the artificial throttling is no longer required.

Results



Node class affinity as impacted by scheduling method. At the end of 2021, memory spillover was enabled on Torque, which allowed for better overall utilization of resources, but reduced node class affinity considerably. With the transition to Slurm, the same trend was observed until the '–prefer' flag was introduced in early 2024.

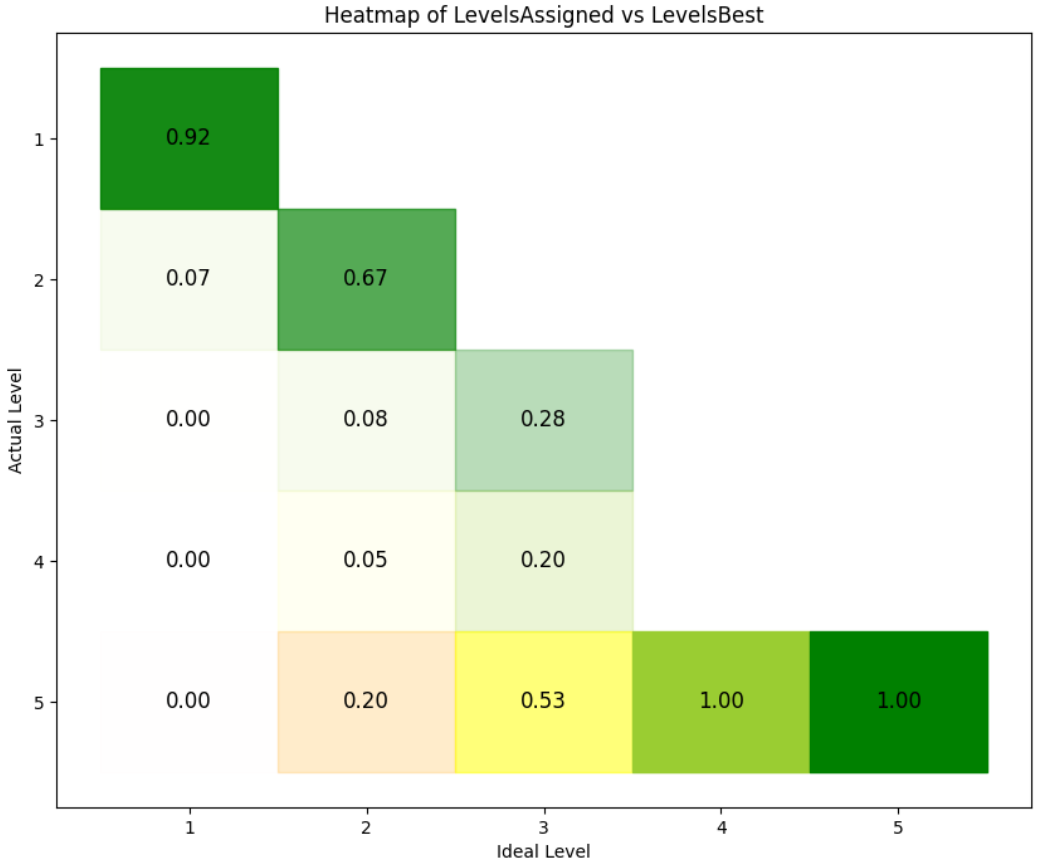
Results



Results from topology aware scheduling implemented using node feature sets in Moab and the fat-tree topology plugin for Slurm, averaged across all jobs per day. Greater values indicate more hops in the interconnect fabric, which results in reduced overall efficiency from higher latency in communication. On average, the approach in Moab was more effective at localizing jobs.

Georgia Tech College of Engineering
School of Electrical and Computer Engineering

Results



Heatmap of topology-aware scheduling for Slurm. Actual Level, y-axis, represents the actual number of hops in the interconnect fabric. The x-axis is the ideal number of hops assuming there is no contention and not taking into consideration command line information such as enforcing separate nodes.

Thank you!
Questions?

Backup

Power of Better Utilization

- Significance of Utilization - Phoenix
  - First year → 35% to 55% utilization
  - Overhead
    PUE – Power Usage Effectiveness
    PUE = Total Facility / IT Equipment
  - Idle Servers (not off)
  - Given same amount of compute:
    Better utilization → less time (less overhead energy) → less carbon footprint
    And/Or
    Better utilization → less number of devices → less carbon footprint

- **Today**: over 85% Utilization