# Changing landscape of computing at BNL

Shared Pool and New Users and Tools

HEPiX Spring
May 2018

William Strecker-Kellogg <willsk@bnl.gov>

70 YEARS OF DISCOVERY
A CENTURY OF SERVICE

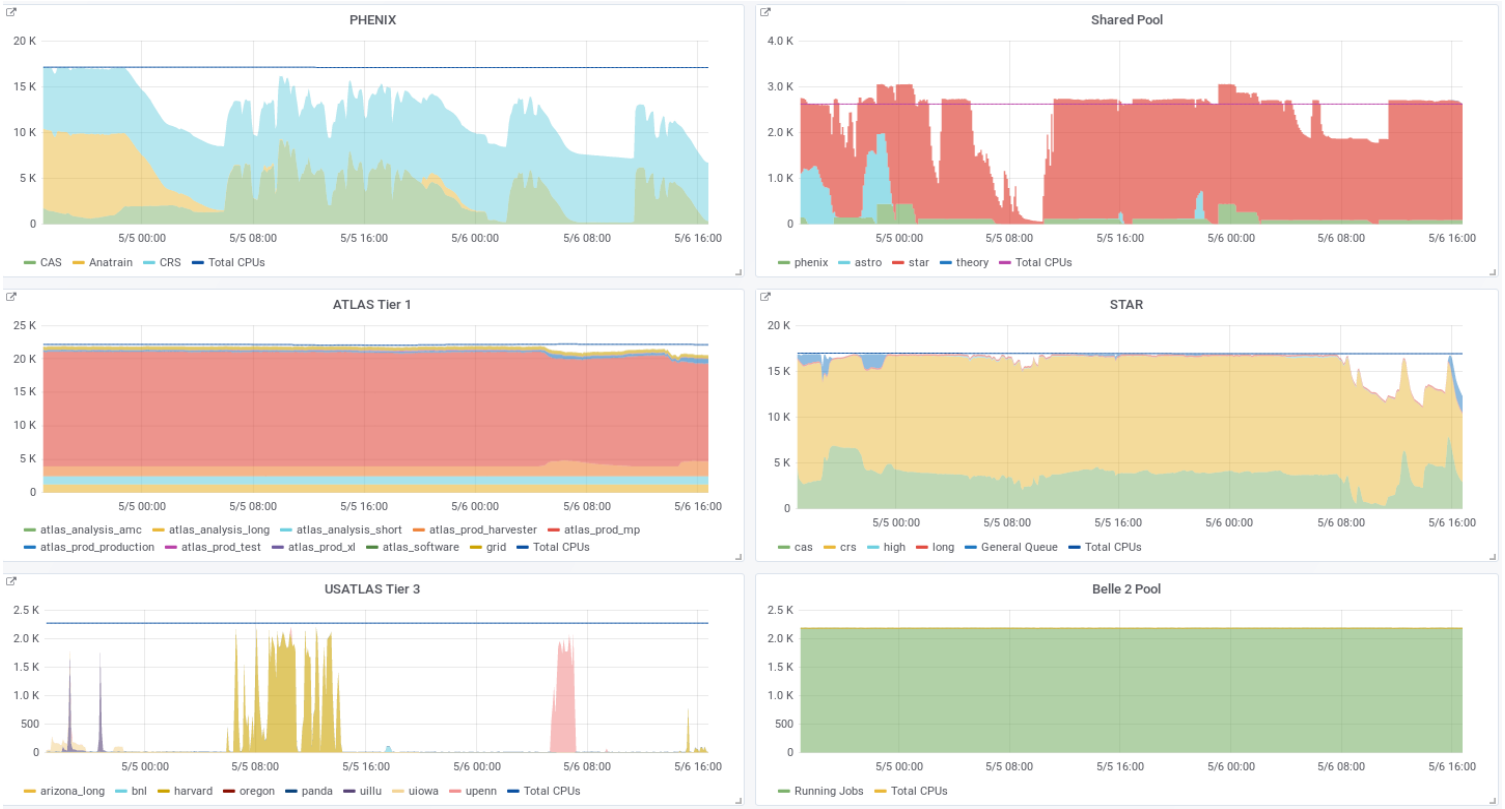
U.S. DEPARTMENT OF ENERGY
BROOKHAVEN NATIONAL LABORATORY

# Shared Pool

Merging 6 HTCondor Pools into 1

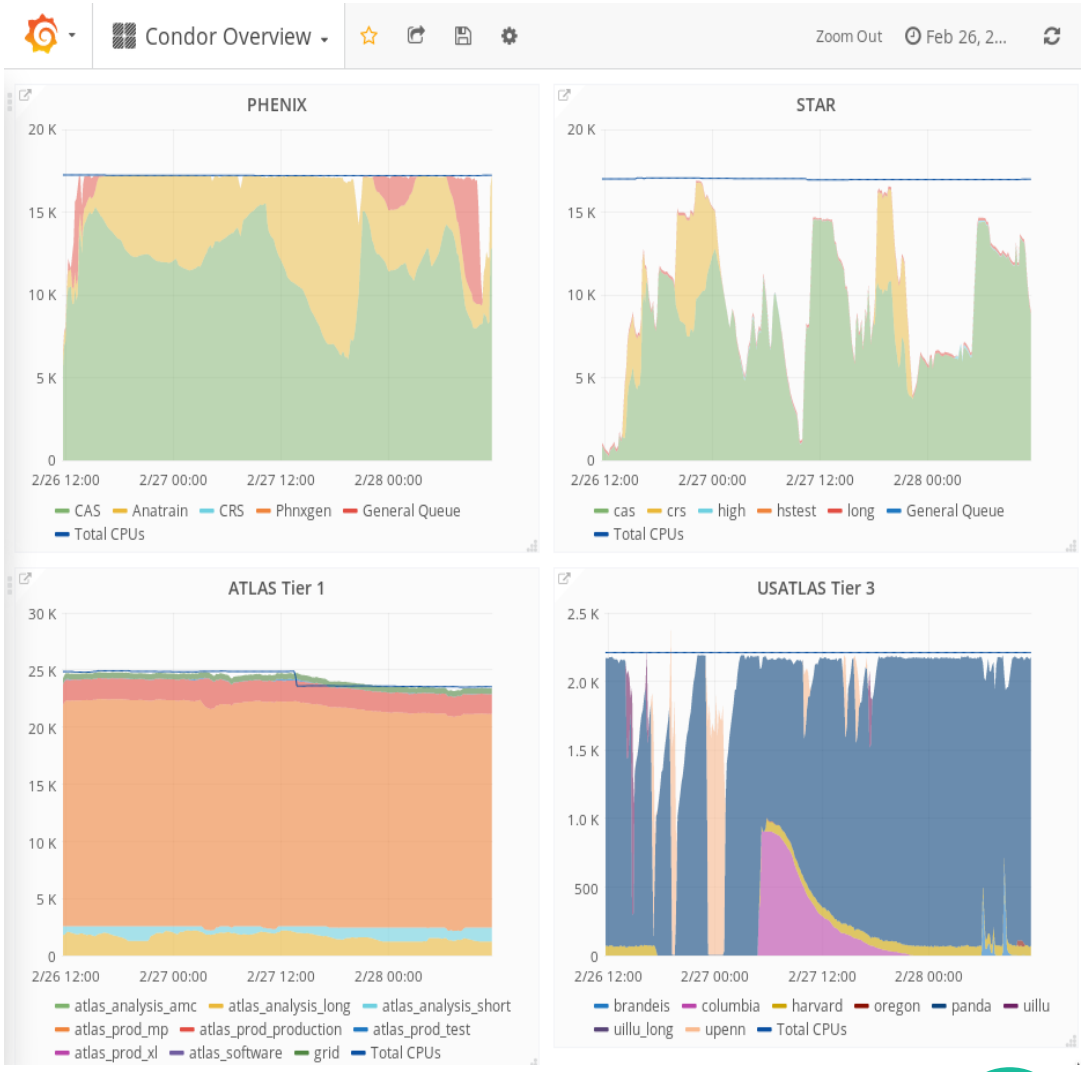
# What?

- **Current Situation**
  - Many pools, not all as well utilized as possible



# Current Situation

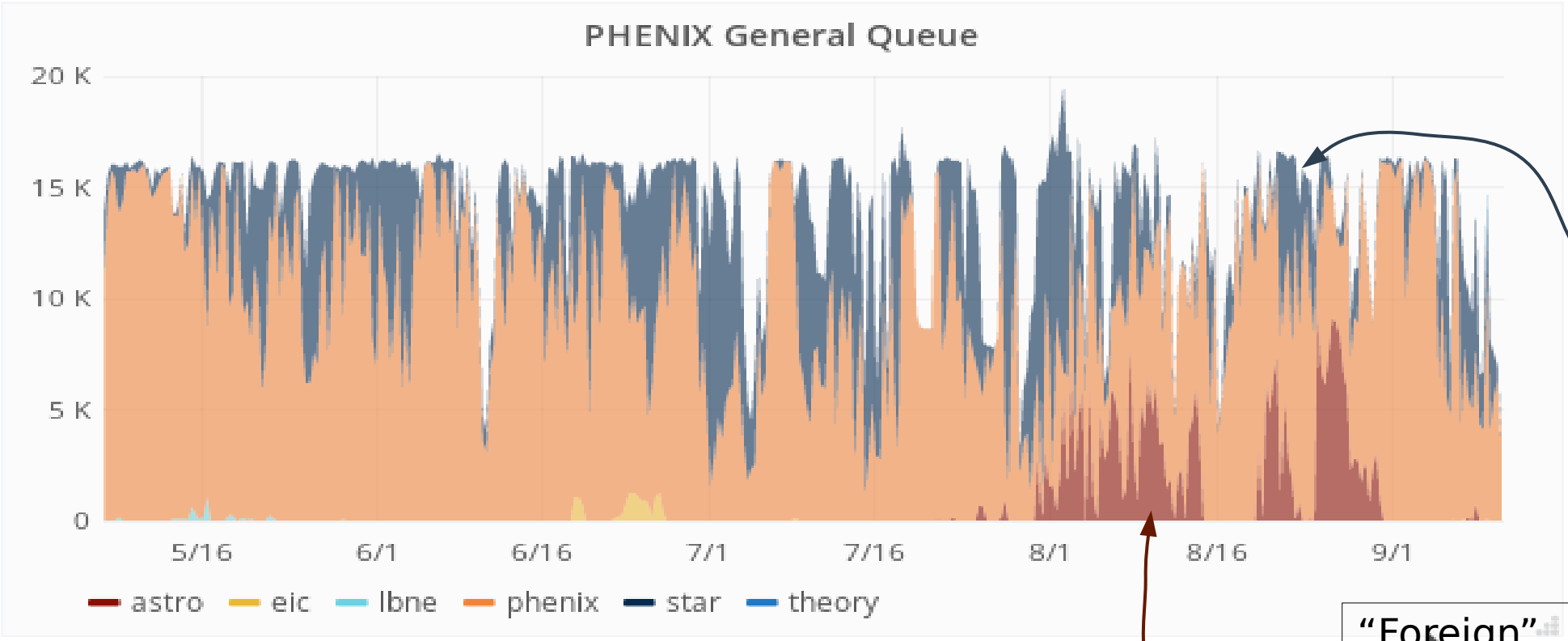
- **Machines owned by each experiment**
  - STAR: ~700 systems
  - PHENIX: ~700 systems
  - ATLAS: ~600 systems
  - Etc…
- **Each experiment runs its own jobs**
  - Custom policies
    - Job length, preemption, etc..
  - Emulated “queues” with policy



# What?

- **Sharing done between some pools with HTCondor Flocking**
  - Policy requirements for “general queue” jobs much stricter than for native jobs
  - Collaborations negotiate with other over these parameters
- **Not possible for many stakeholders**
  - E.g. ATLAS, using group quotas and auto-balancing (see my previous talks)

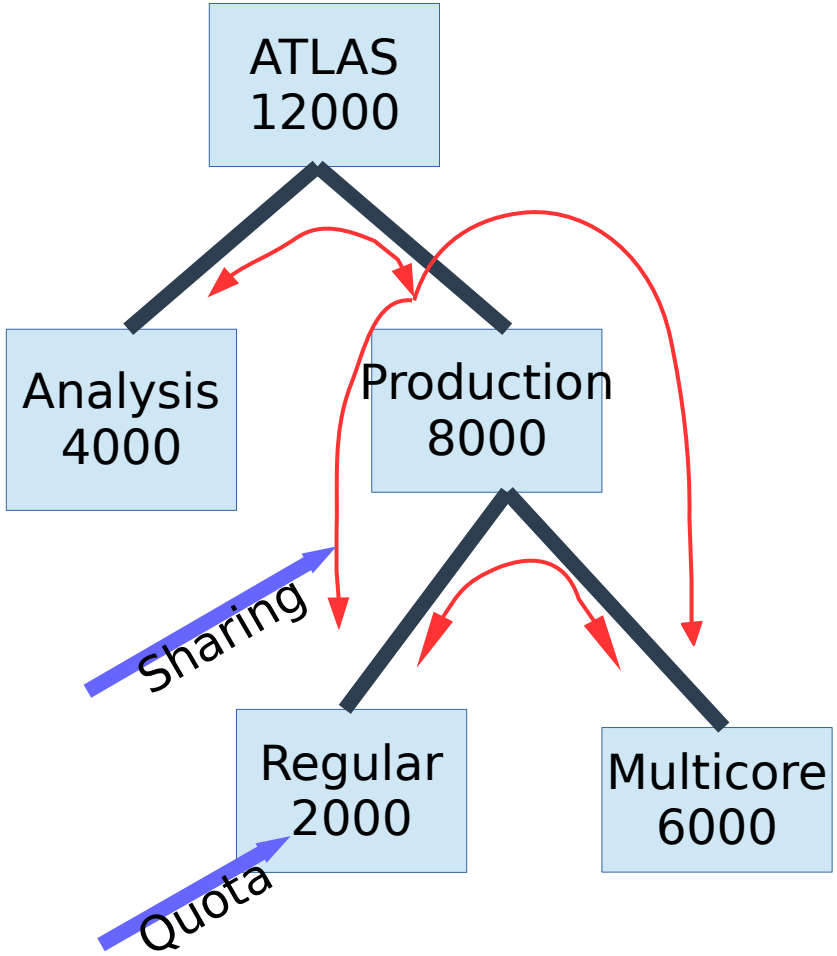
# Flocking



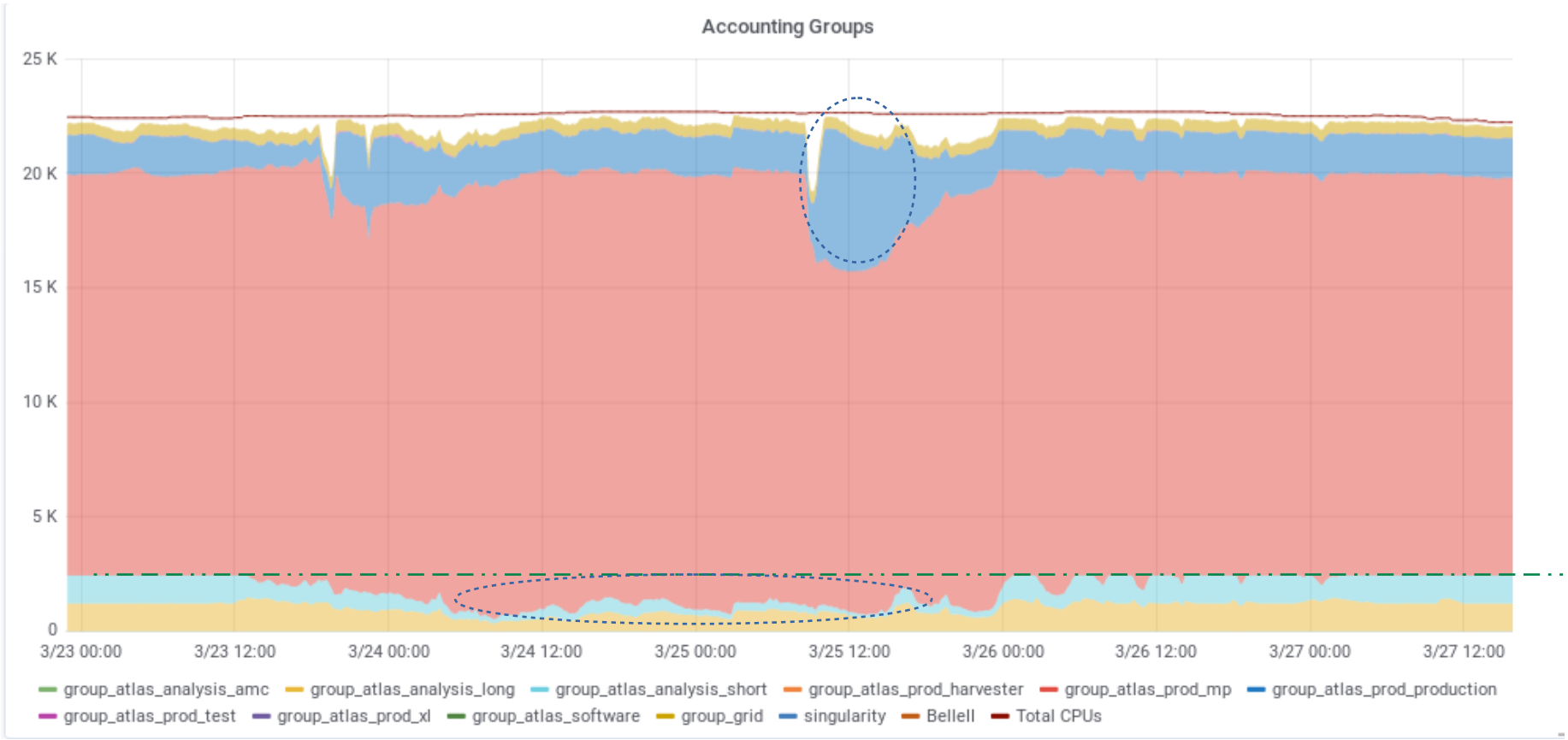

"Foreign" experiments flocked to PHENIX

# Group Quota Model

- **Hierarchy of job classes with “quotas” assigned to each class (how many CPUs can they get)**
  - Jobs “spill” between groups freely



# Group Surplus in ATLAS




"level" for analysis

Areas where other queues "fill in" with surplus-sharing automatically

# How?

- **Will adopt the group quota model**
  - Experiments→Top-Level Groups
  - Quotas→Set by contribution
  - Flocking→Surplus sharing
  - Queues→Sub-Groups in experiment
  - Fair-share→Fair-Share (between users within group)

# Why?

- More standard setup—everyone gets same features
- Sub-groups give experiments flexibility to define own policies / “queues”
- Surplus sharing automatically ensures maximum occupancy
- One unified policy (helps manage user expectations during growth)
- Easy scaling of offloading work. E.g. Backfilling HPC (slurm) resources with routed overflow jobs

# Implementation Details

- **Partitionable Slots**
  - Each node becomes divisible along several dimensions until one is exhausted (RAM, CPU, Disk...)
  - More and more users are requiring high memory or multicore
  - Serving more customers clearly requires enabling this feature
  - Pool fragmentation helped with condor_defrag

# Implementation Details

- **Preemption**
  - Needed for two reasons (assuming latency constraints)
    1) Intra-group: most collaborations want to be able to evict a resource hog sooner than the maximum runtime allowed (Latency)
    2) Inter-group: if surplus sharing is on, a group can monopolize the pool, not acceptable for all other colloborations to wait the maximum time to get their own resources back (preempt a group down to its quota)
- Currently has major issues with Partitionable-Slots
  - HTCondor team promises progress here…

# Limitations

- **Latency vs. Throughput:**
  - Most fundamental limit—all groups need to agree on allowable job run-time and acceptable latency
    - Currently able to be set per-experiment, but by its nature it is pool-wide
  - Manifests in several places
    - Maximum job lengths, how to fairly allow differences
    - How much to defragment to allow “large” jobs

# Implementation Details

- **Partitionable Slots w/ Preemption**
  - How to make room for larger jobs?
  - All slots that meet Preempt-Requirements
    - How to choose what to evict
      - Users with worst integrated priority?
      - Users belonging to groups most over their quota?
    - This is non-trivial and will require experimentation
- **Do we really need preemption?**
  - Who is using it now?
  - What are your time limits or latency requirements?

# New Computing Paradigms

HTC and Jupyter for Iteractive Scaling

# Jupyter

- **Came from IPython project**
  - Ipython→Jupyter (Hub/Lab/whatever)
- **Interactive Python Interperter and Login Shell in Browser**
- **Why not just log in normally?**
  - This is (to most admins inexplicably) a major barrier to science getting done
- **Very useful session management and portability via browser**

# BatchSpawner

- **Using Condor BatchSpawner**
  - Jupyter sessions spawned in batch jobs that proxy back to the jupyter node
  - Allows reduction of dedicated interactive nodes
  - Greatly-enhanced scalability of interactive workload
- **This is timely, considering growth of userbase and the shared pool**
  - Raises issues of latency again!
- **Ofer will talk in detail tomorrow**

**Questions?**
**Comments?**

**Thank you!**

GPU Hackathon @ BNL
This September
See this site for info