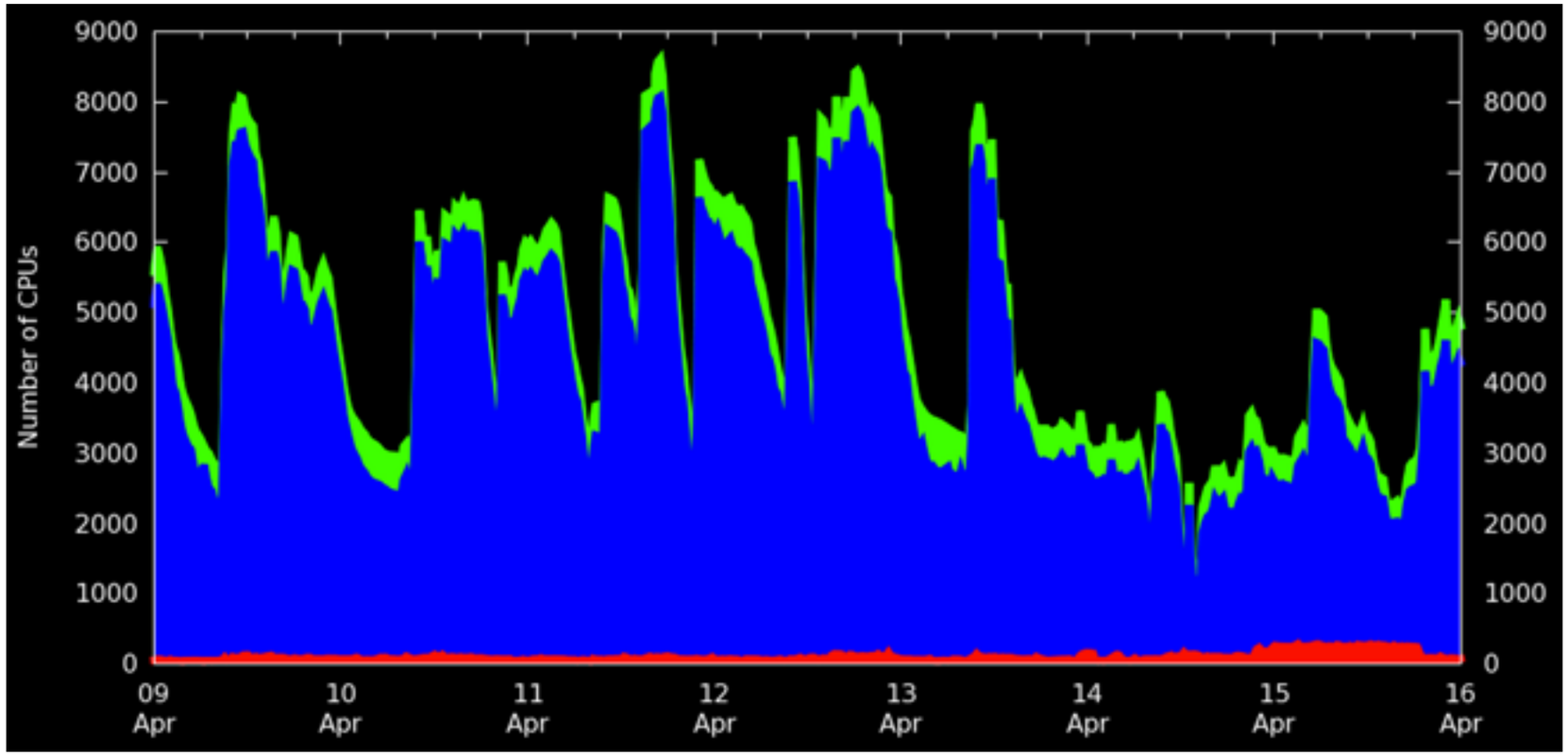
Exploiting Volatile Opportunistic Computing Resources with Lobster

Anna Woodard, Matthias Wolf, Charles Nicholas Mueller, Ben Tovar, Patrick Donnelly, Kenyi Hurtado Anampa, Paul Brenner, Kevin Lannon, Michael Hildreth, Douglas Thain

Why Lobster?

- **We have access to a lot of CPUs!**
  - Notre Dame (ND) Center for Research Computing: ~21k CPU cores + 2.5 PB storage
  - They belong to individual PIs— available opportunistically



Why Lobster?

- **We also have access to CCTools people!**
  - Team at ND, led by Doug Thain, that develops CCTools suite (WorkQueue, Parrot, Chirp, etc) is ‘down the hall’
- **Free CPUs + CCTools people = Lobster**

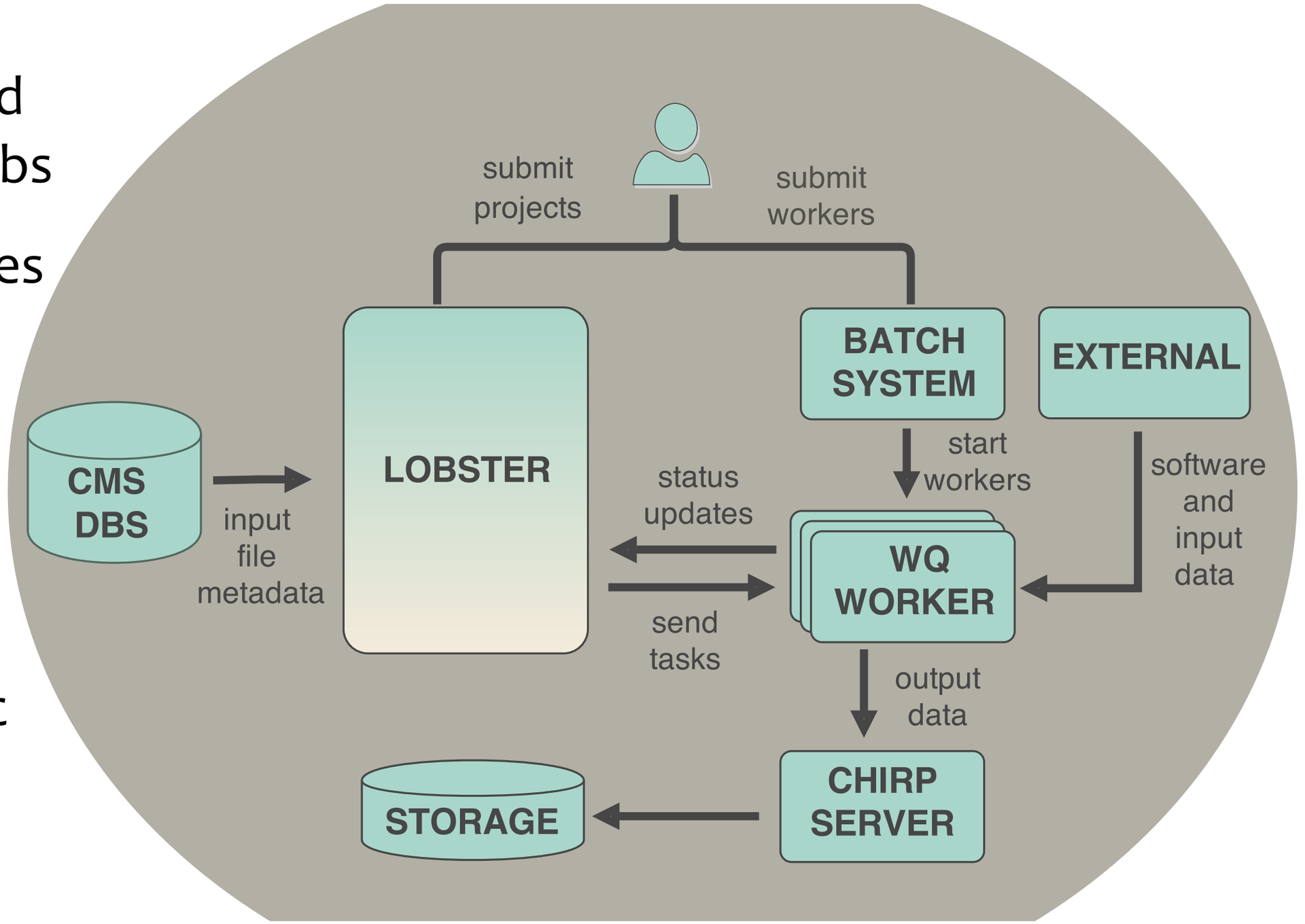
Why Lobster?

- **Doing physics on the ND cluster is challenging:**
  - Only for members of the university
  - Does not have our software, and we do not have root privileges
  - Evicted when owners reclaim resources

Lobster is an opportunistic workflow submission and management tool

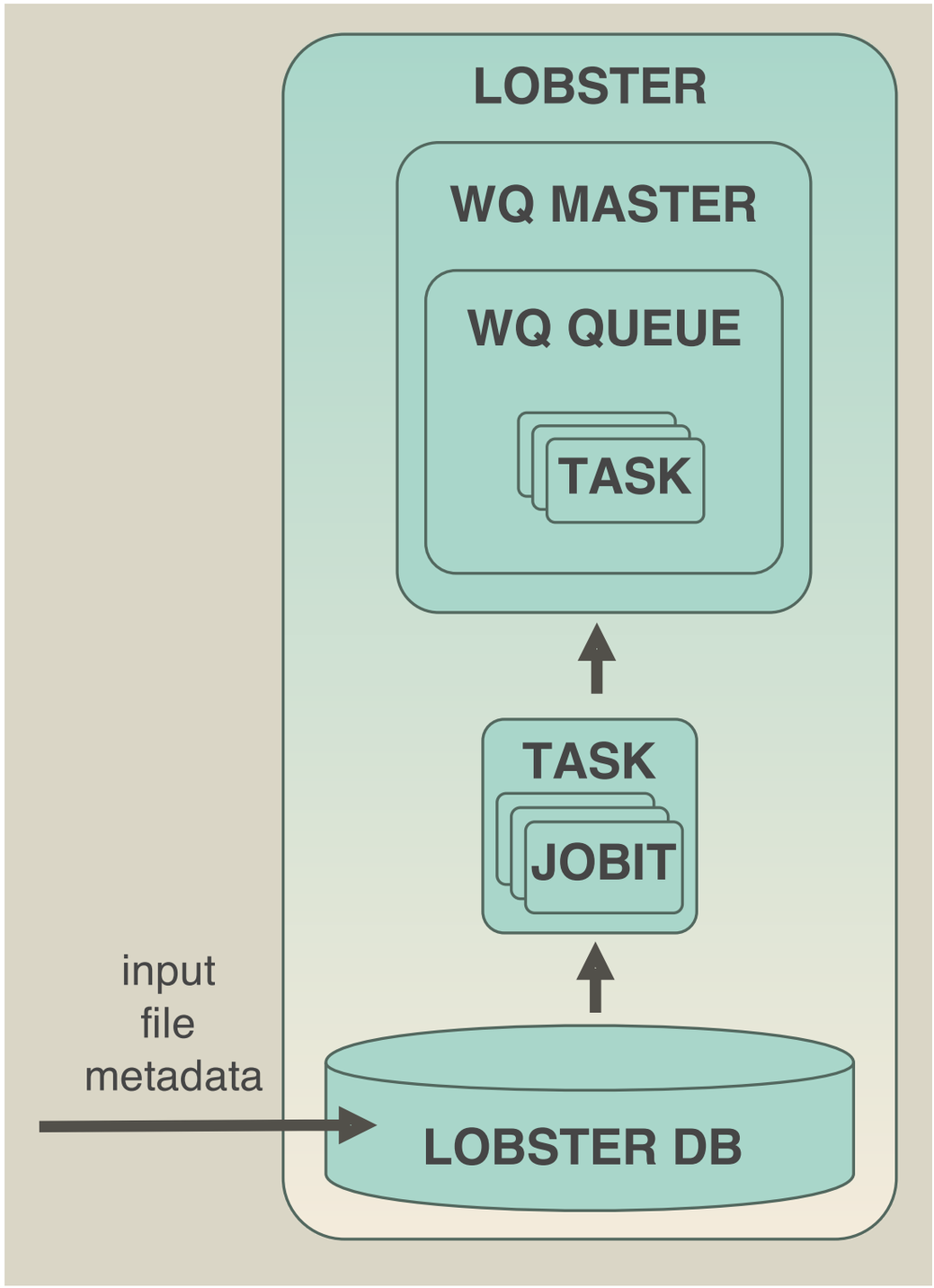
What does Lobster do?

- **Scheduling:** schedules and dispatches jobs
- **Data:** manages input/output data and software
- **Execution:** runs tasks on opportunistic or dedicated resources



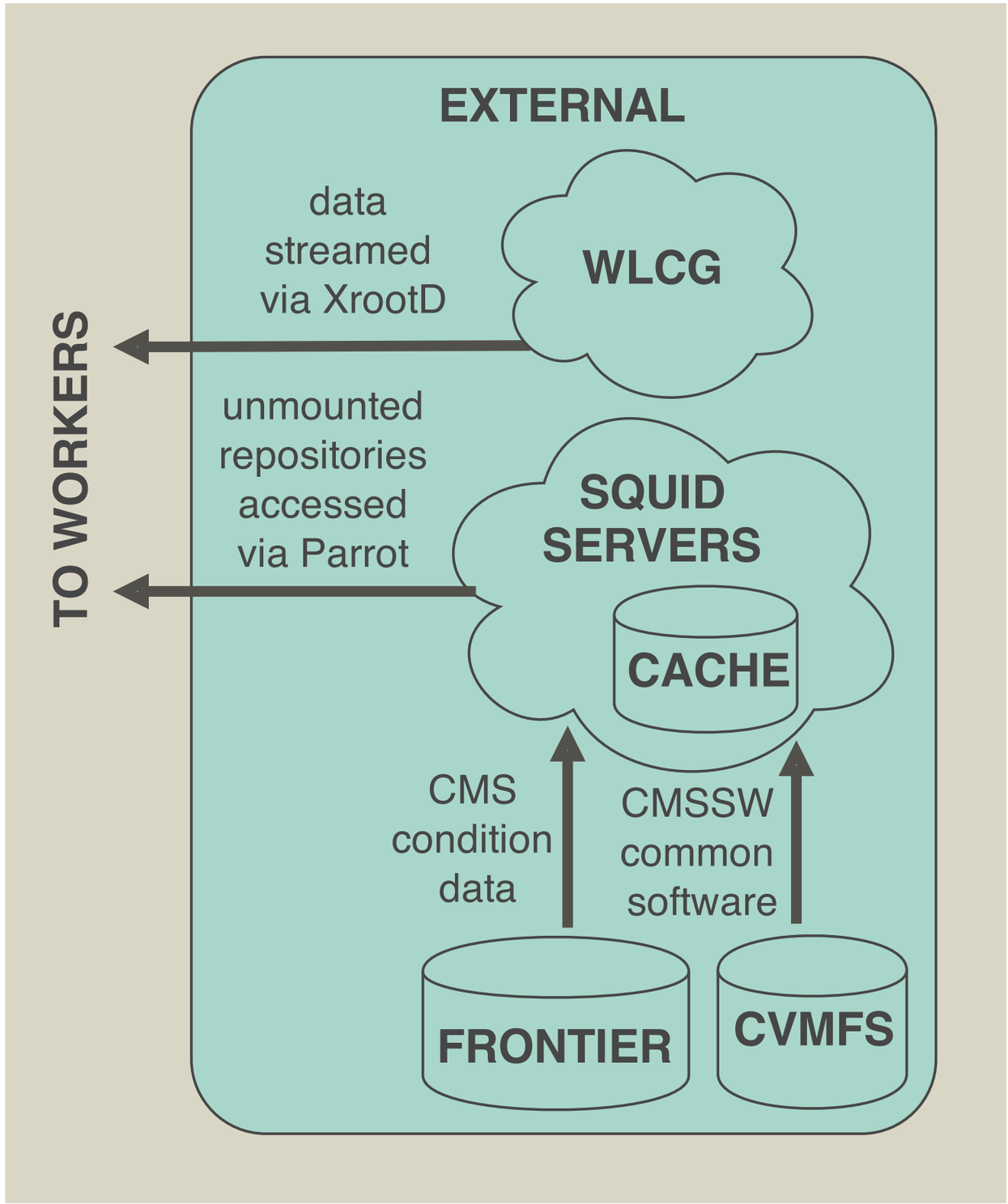
Scheduling

- Work broken down into smallest sensible quantum: **jobit**
- Jobs assembled from jobits **on-the-fly**—user does not need to think about jobs
- Job length can be adjusted **while running**



Data

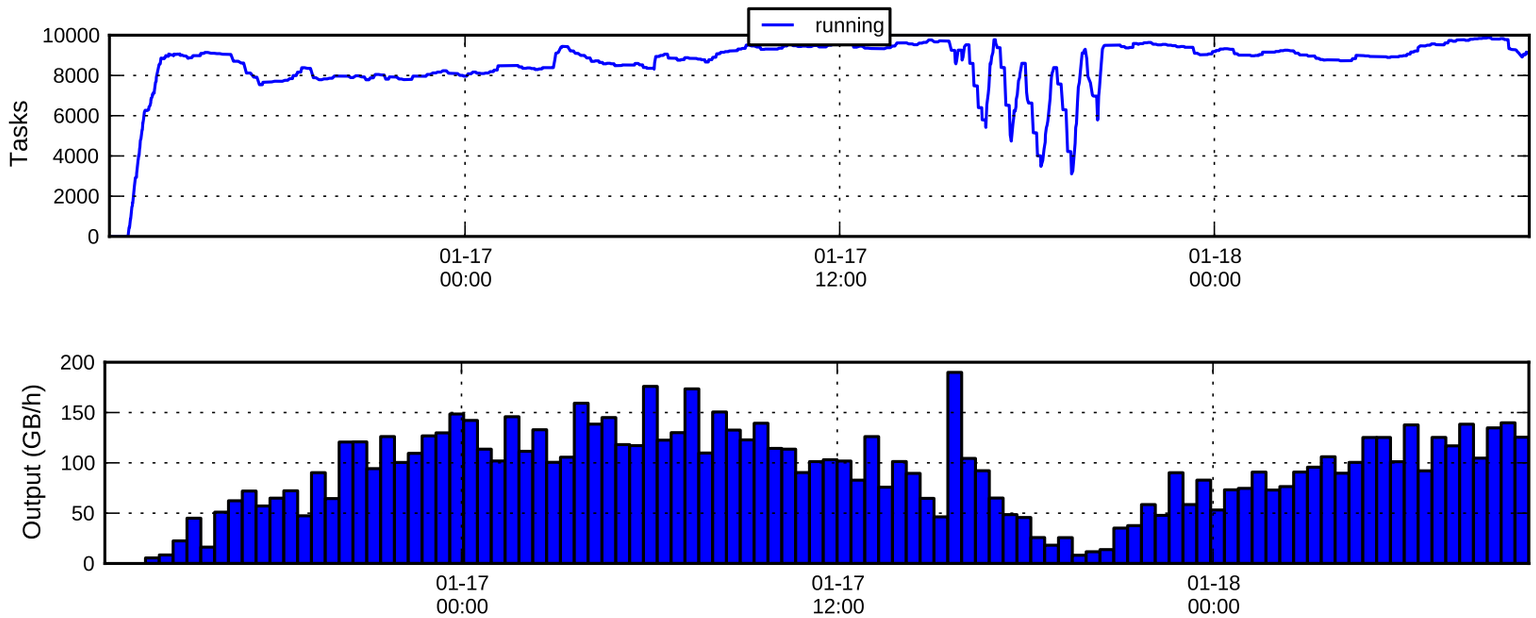
- Leverage a variety of tools (CVMFS, Parrot, Chirp, XrootD, WQ) in order to get data to jobs



Execution

- Workers can be submitted via HTCondor, SGE, PBS, etc— **only user permissions required!**
- Workers hold resources: they set up the environment and then run tasks until eviction or the work is finished
- Multi-core workers share local cache

Conclusions



Tasks running

Output transferred

- Lobster has enabled ND team to exploit opportunistic campus resources to ~10k core scale
- Similar in scale to some USCMS T2 sites