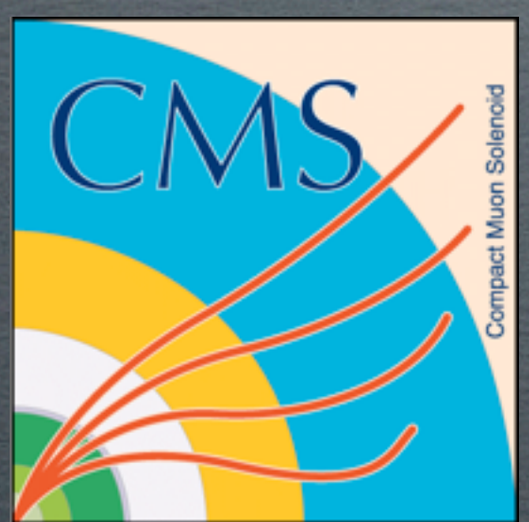
Lobster: Scaling Opportunistic CMS Workflows to 10k cores

Anna Woodard, Matthias Wolf, Charles Mueller, Nil Valls, Ben Tovar, Patrick Donnelly, Peter Ivie, Paul Brenner, Douglas Thain, Kevin Lannon, Michael Hildreth

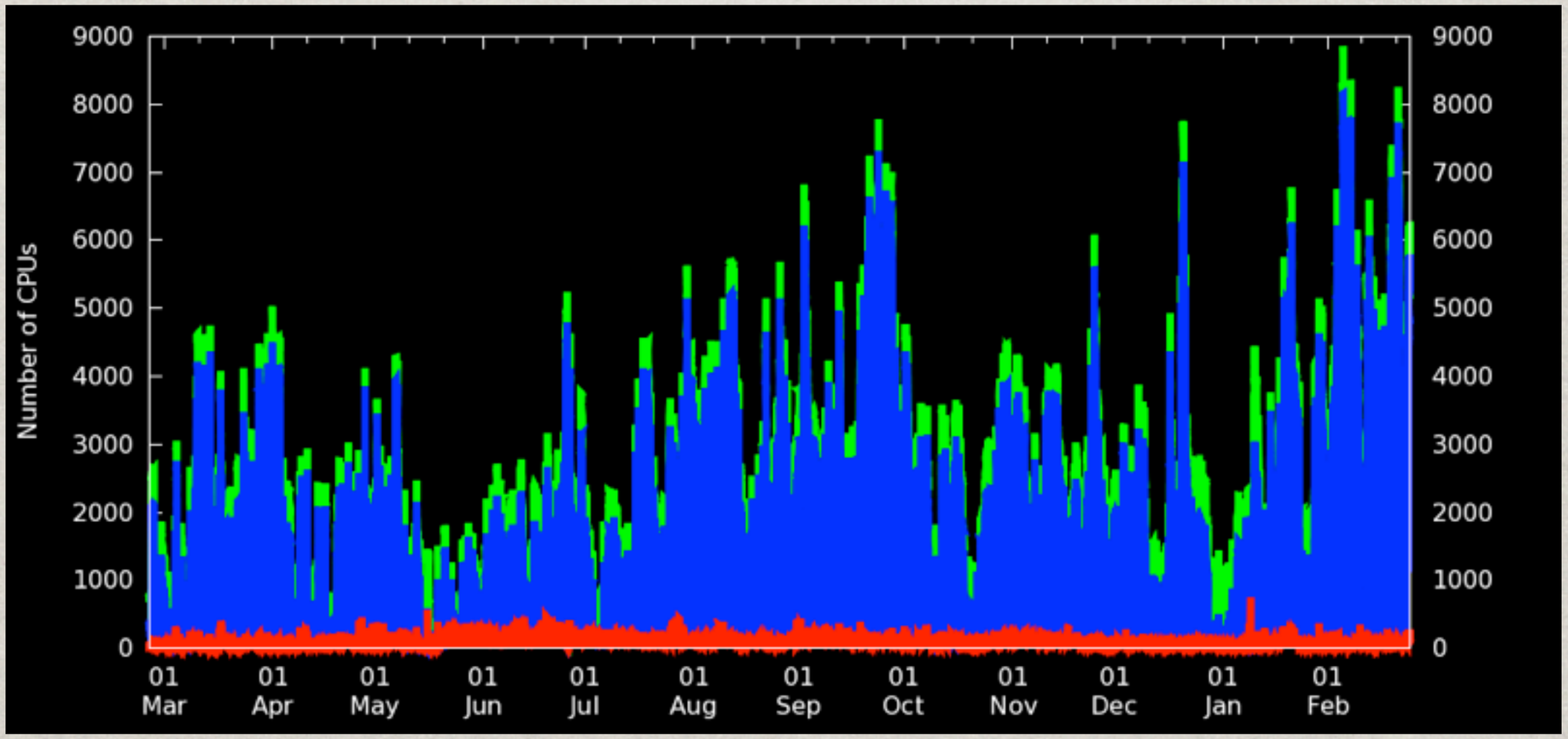


What is Lobster?

- **L**arge-scale **O**pportunistic **B**atch **S**ubmission **T**oolkit for **E**xploiting **R**esources
- Workflow submission and management tool written from scratch by two ND grad students (Matthias Wolf and Anna Woodard)
- Borrows ideas from CRAB2/3 and grid-control
- Based on CCTools suite (WorkQueue, Parrot, Chirp) from ND team led by Doug Thain
- Primary goal: Get access to ND's opportunistic computing resources

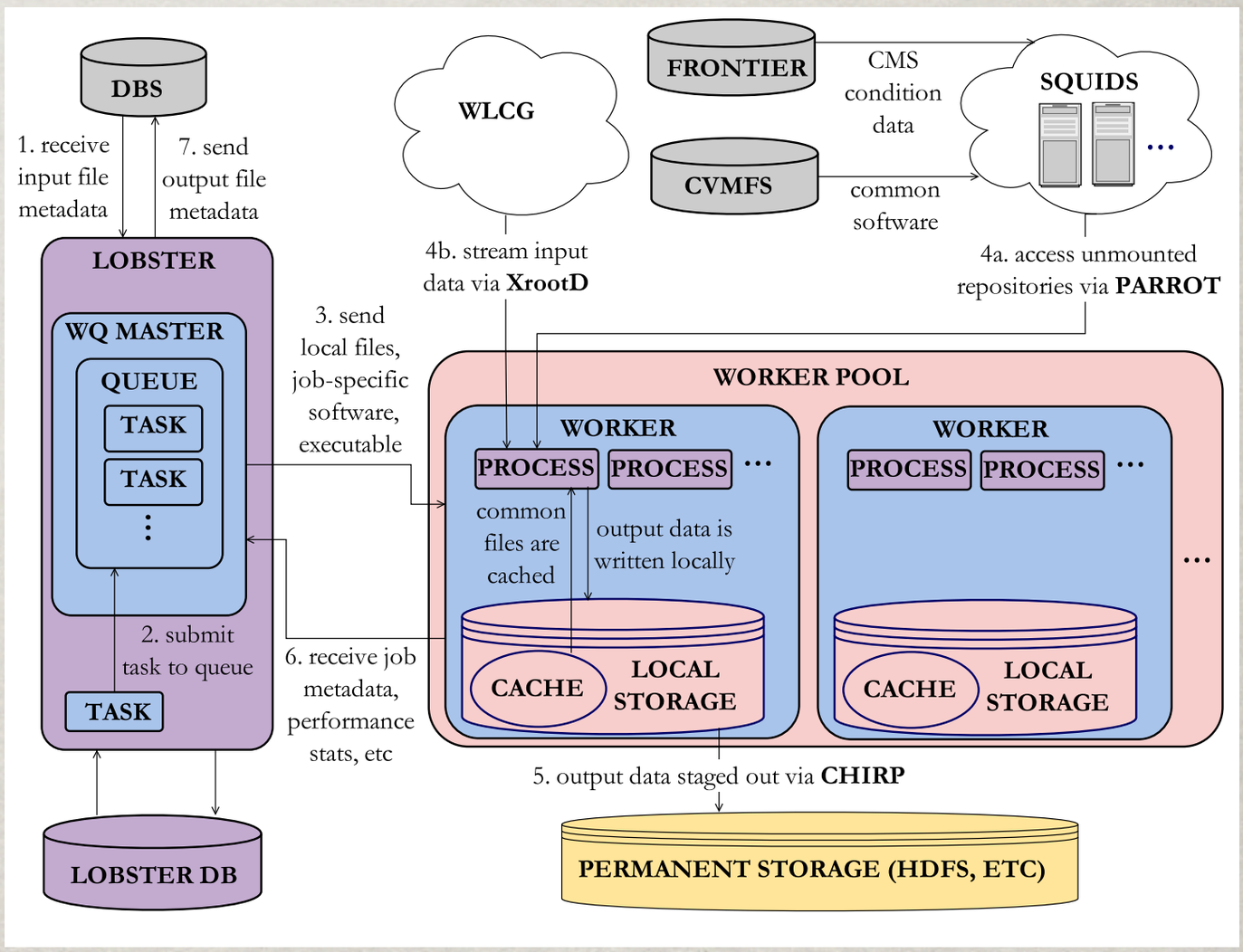
ND CRC Resources

- ND Center for Research Computing houses ~21k CPU cores and 2.5 PB of storage
  - Most resources belong to individual PIs
  - Available for opportunistic usage when idle (evicted when owners reclaim resources)



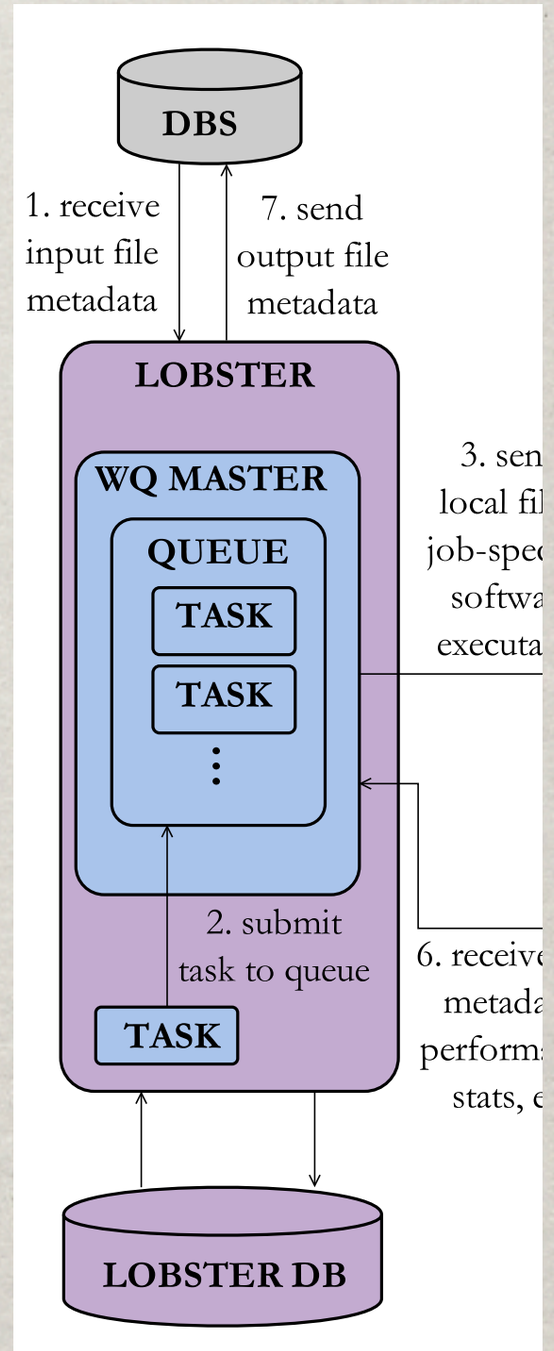
Lobster Architecture

- Main components
  - Scheduling: schedules and dispatches jobs
  - Data: managing input/output data and software
  - Execution: runs tasks on opportunistic resources
- Master-worker architecture

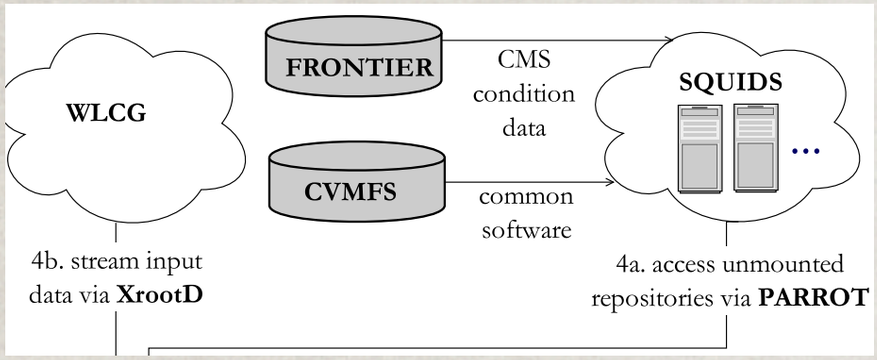


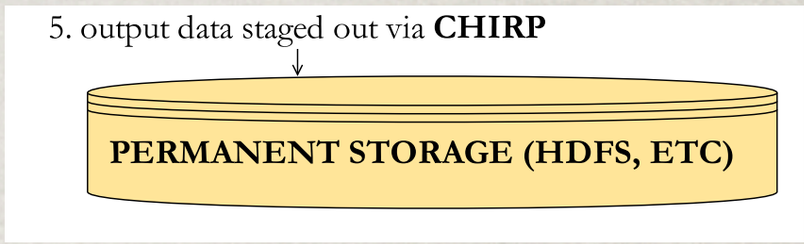
Scheduling

- Lobster master uses DBS to build database of work to be done
  - Work broken down into smallest sensible quantum: jobit
  - Lobster master schedules assigns jobits to tasks and schedules in Work Queue (WQ)
- WQ master handles distributing tasks to workers and tracks task progress
  - Optionally, system may include "foremen" to mediate between WQ master and workers
- Lobster communicates with WQ master to track jobit completion



Data

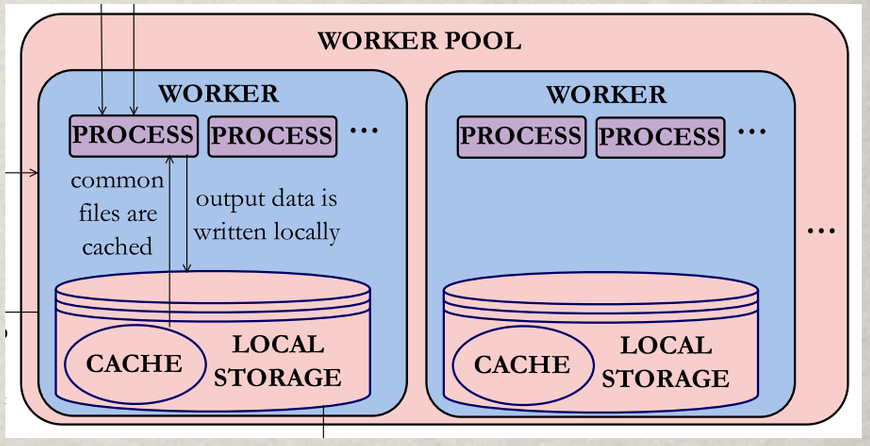




- Leverage wide variety of tools (CVMFS, Parrot, Chirp, XrootD, WQ) to distribute data to jobs
  - CMSSW distributed to workers via CVMFS+Parrot (squid cache, worker cache)
  - Job scripts and sandbox transferred via WQ (worker cache)
  - Conditions via Frontier (squid cache)
  - Input data delivered via AAA (XrootD) or ND T3 storage (XrootD or Chirp)
  - Outputs transferred via Chirp

Execution: Workers

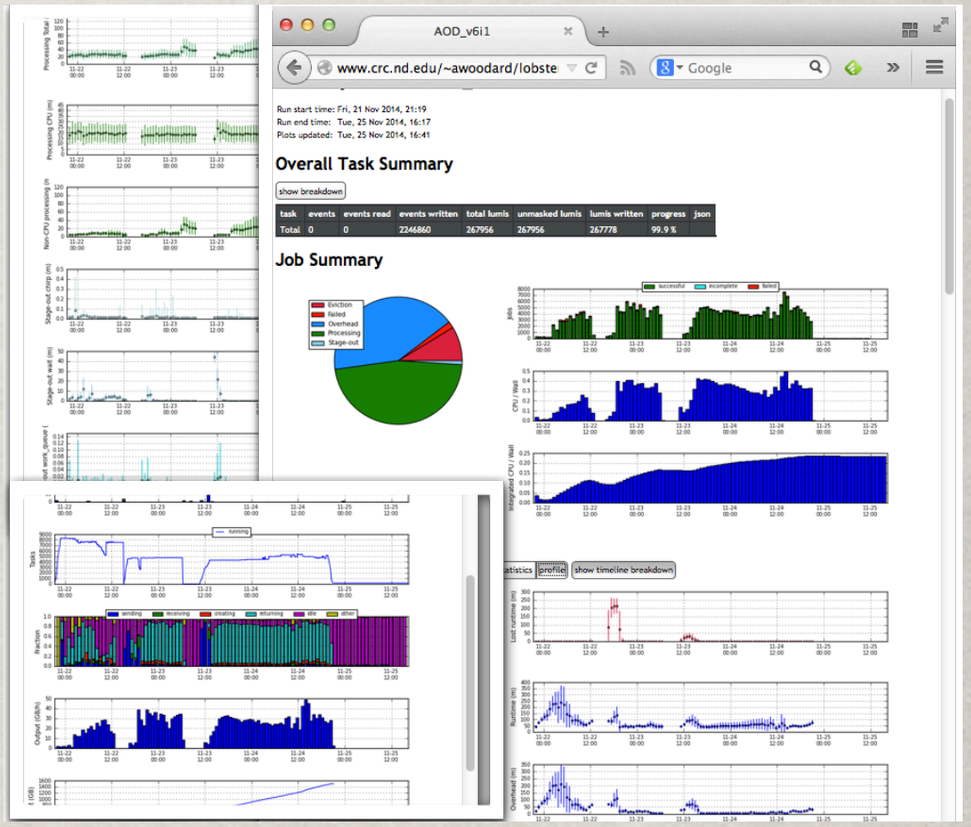
- Can be submitted via whatever batch system is available (HTCondor, SGE, PBS, etc.)
  - CCTools includes tools for managing worker pools
- Responsible for configuring resource to accept CMS tasks (setup CMSSW, etc.)
- Holds resources and runs tasks for master until work is finished or worker is evicted
- Multicore workers will run multiple tasks in parallel, sharing local cache for CVMFS and WQ files



UNIVERSITY OF NOTRE DAME

Monitoring

- Opportunistic resources change dynamically (chaotically)
  - Resources come and go depending on owner activity
  - Heterogeneous quality
  - Can fail randomly
- Monitoring critical to Lobster success
  - Lobster tracks time stamps of every phase of task setup and execution
  - Collects information in plots and tables on webpage
  - Gives comprehensive picture of system components so that bottlenecks and failures can be diagnosed and overcome

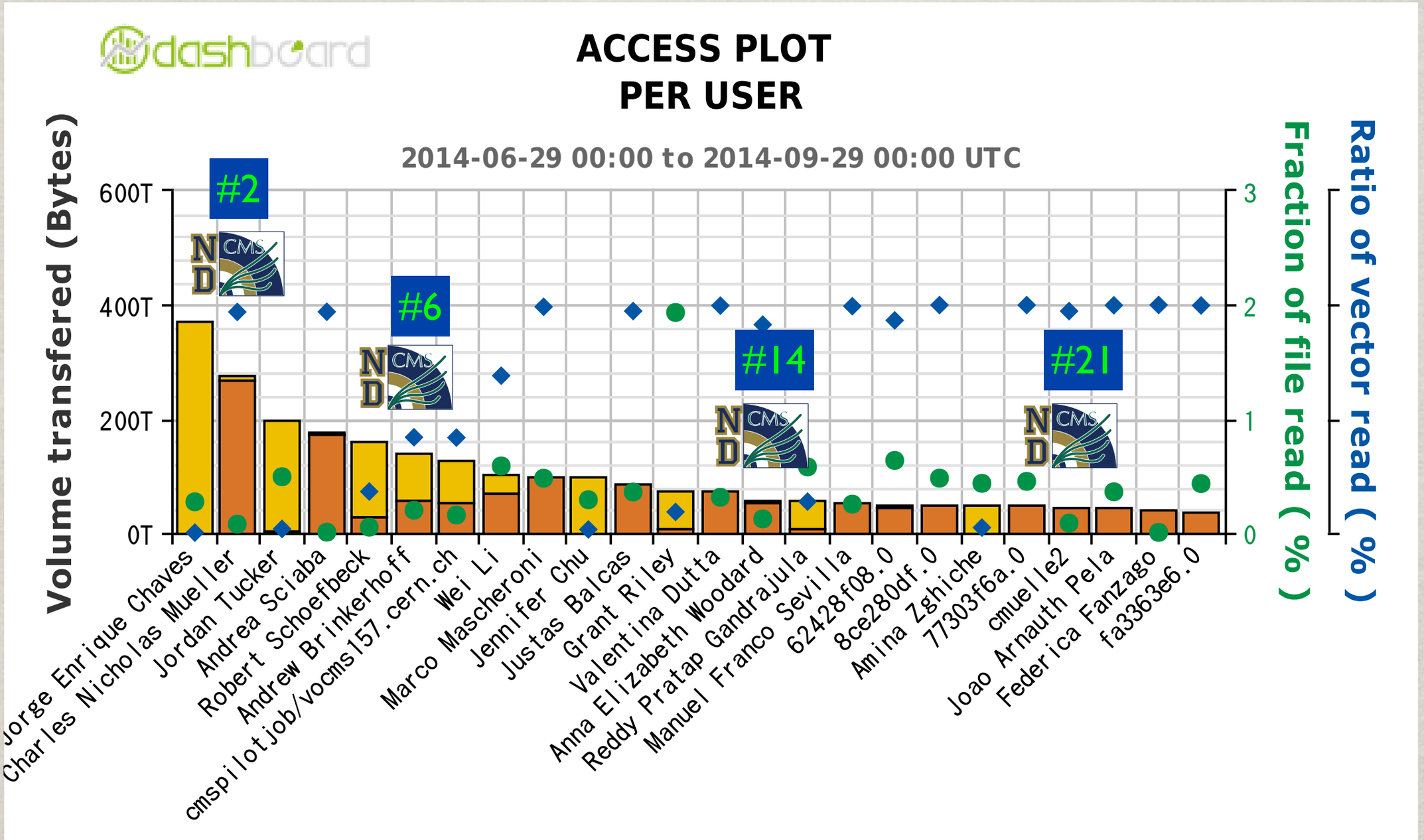


Design Motivations

- Maximize opportunistic spirit!
- CCTools suite operates completely in user space: no admin intervention required to use opportunistic resources
- Eviction requires agility: Decouple job size from output size and user task management
  - Lobster works in jobits, tracks splitting, handles resubmission without user intervention
    - Often leads to really small output files--merged automatically
    - When processing multiple datasets, jobit execution randomized to level load on AAA
  - Persistence pays off
    - Workers try very hard to get tasks started: Use local CVMFS if present, switch to Parrot if not, etc.
    - Lobster retries failed tasks until you stop it: needs to be resilient against transient failure in opportunistic system
    - Work Queue Pool: if workers die/lost, resubmit more

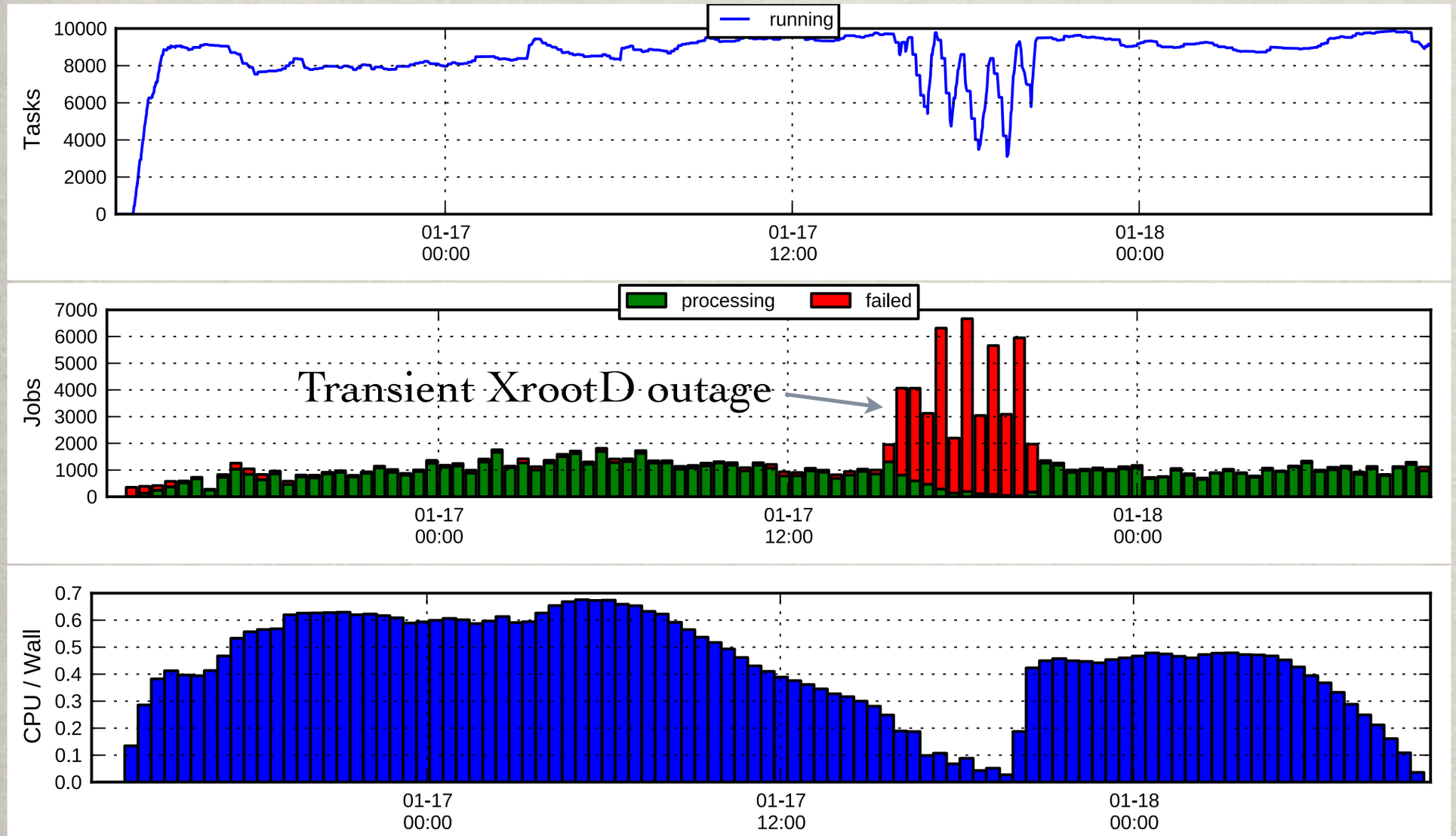
Processing Lots of Data

- ND users running only on ND resources competitive with CSA14 activity



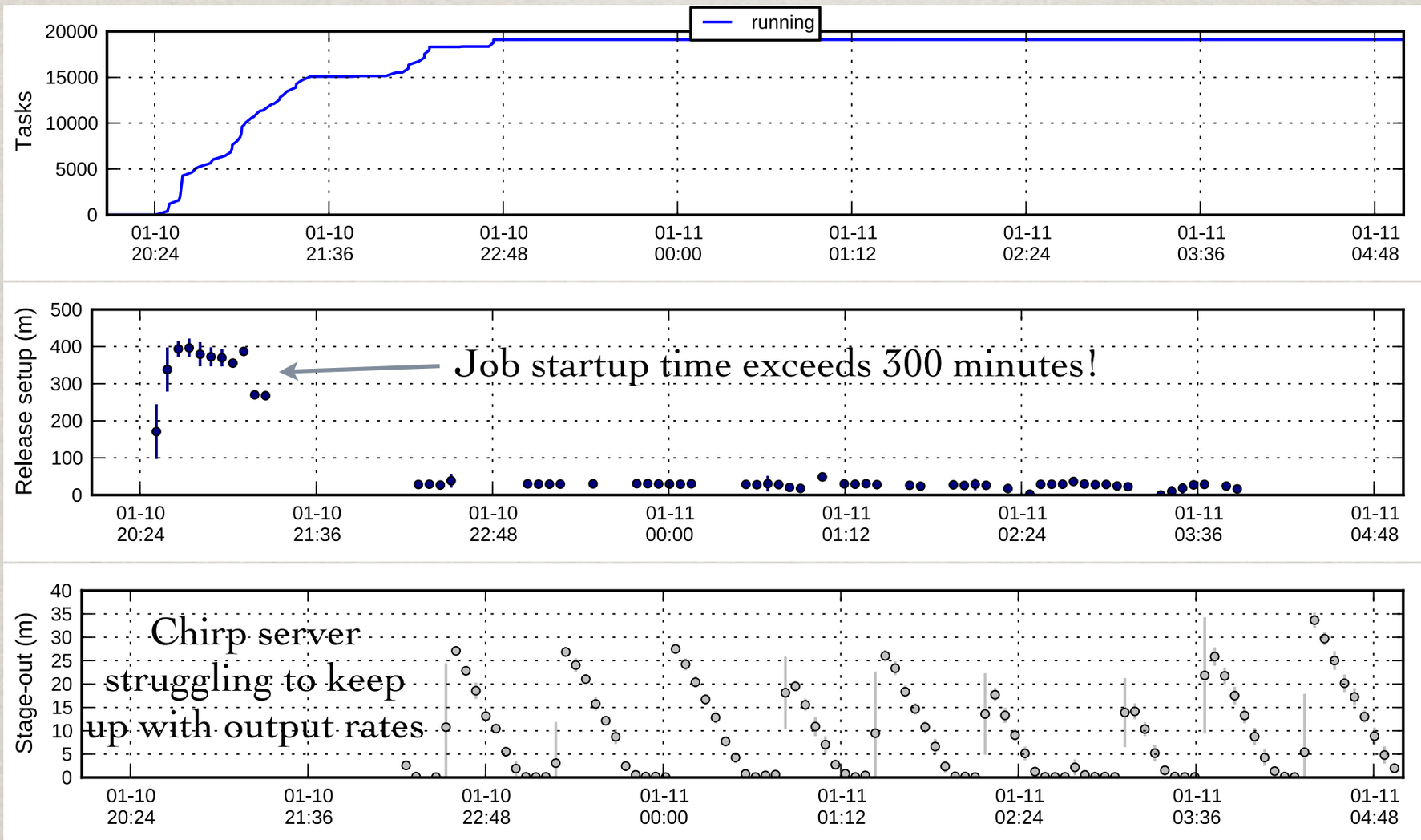
Reaching ~10k running jobs

- This is ~10% of size of CMS global pool
- Comparable to scale of one US CMS T2 site
- More than total of all US CMS dedicated T3 resources



Challenges

- Bottlenecks when running at large scales
  - Job overhead time increases non-linearly: bottleneck in squid cache?
  - Output Chirp server can get overloaded and fail



Near Term Future

- Resolve bottlenecks preventing scaling to 20k cores
  - Also important for decreasing overhead to minimize eviction losses
- Explore possibilities for Lobster to dynamically adapt to running conditions
  - E.g. Automatically adjust jobits/task to optimize for current running conditions (eviction)
- Improvements in reliability and robustness
  - E.g. Run Chirp server as service instead of user process to keep users from overwhelming login node with Chirp processes

Conclusions

- Lobster has enabled ND team to exploit opportunistic campus resources to 10k core scale
- Successful collaboration between physics and CS teams
  - Learned a number of useful things
  - Motivated improvements to CCTools suite
- Potential for lessons learned and innovations to be translated more widely to CMS
  - Anna and Matthias now working as CRAB3 developers
  - See Brian's talk for longer term vison