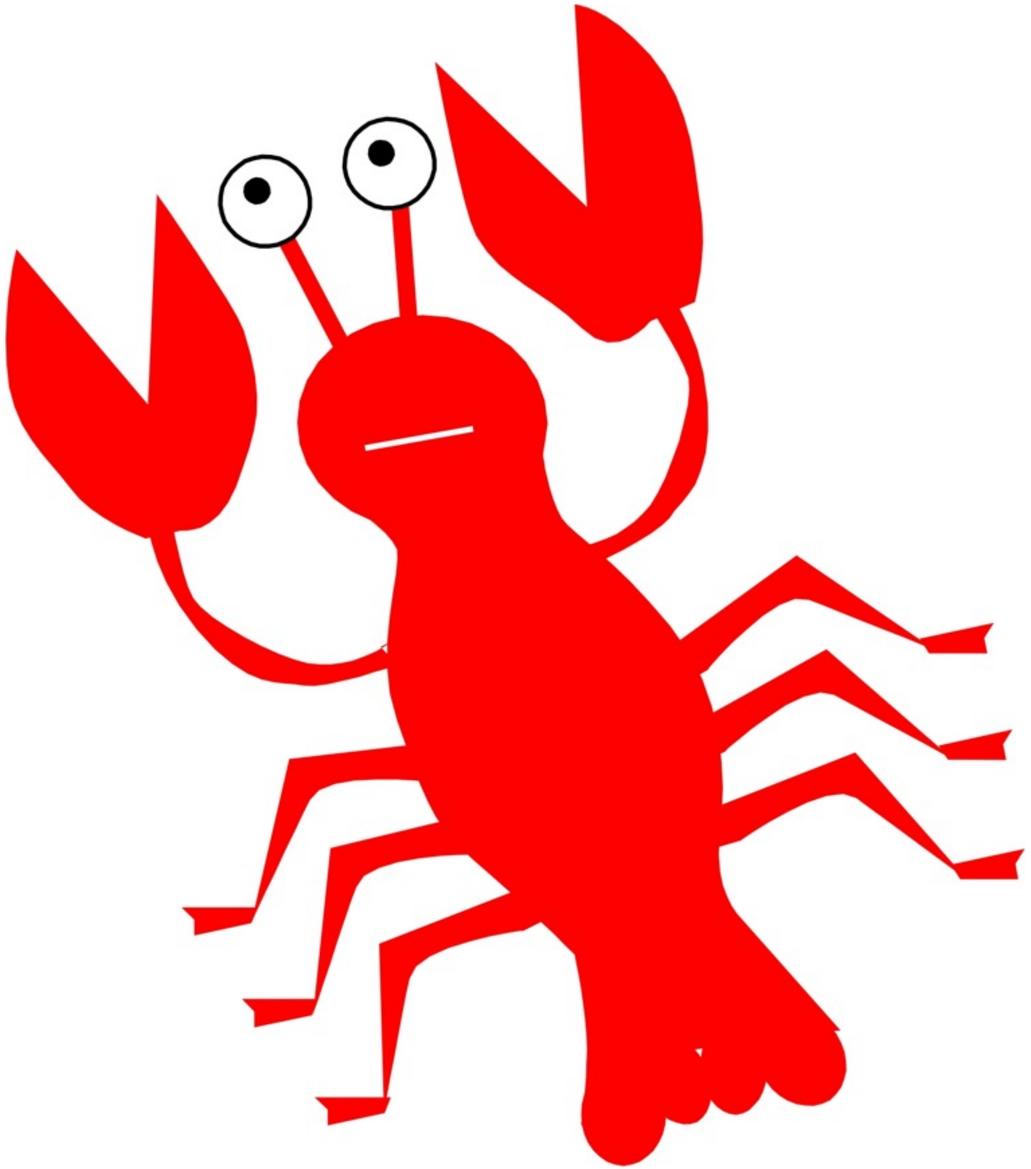
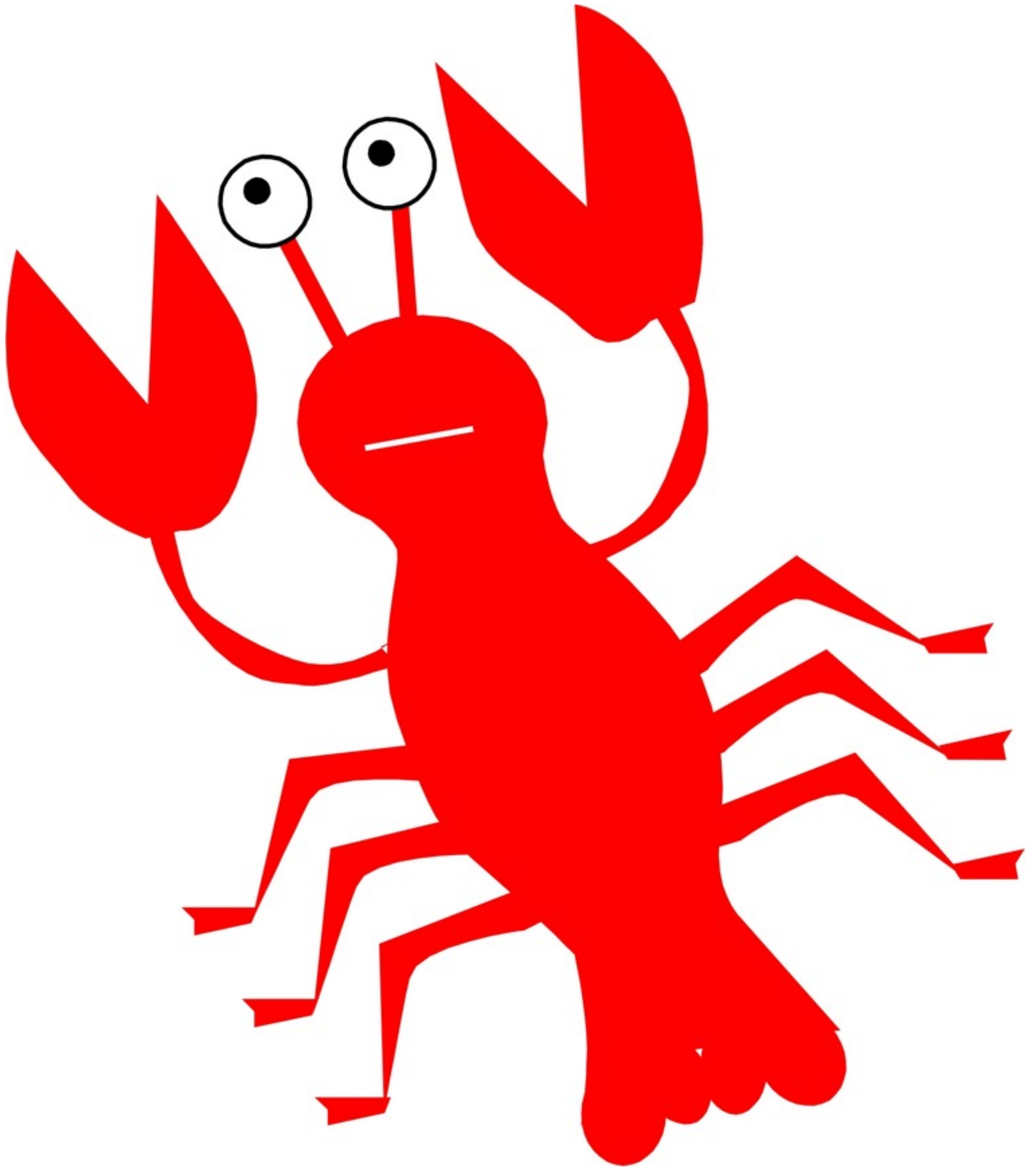
MATTHIAS WOLF, ANNA WOODARD, PAUL BRENNER, PATRICK DONNELLY, MIKE HILDRETH, KENYI PAOLO HURTADO ANAMPA, WENZHAO LI, KEVIN LANNON, DOUGLAS THAIN, BENJAMIN TOVAR, ANNA YANNAKOPOULOS

OPPORTUNISTIC COMPUTING WITH LOBSTER: LESSONS LEARNED FROM SCALING UP TO 25K NON-DEDICATED CORES



WHAT IS LOBSTER?

Lobster is an opportunistic workflow manager, built on top of CCTools.



WHAT IS LOBSTER?

Lobster is an opportunistic workflow manager, built on top of CCTools.

Basic idea introduced at CHEP2015 [1] and IEEE Cluster 2015 [2].

[1] http://iopscience.iop.org/article/10.1088/1742-6596/664/3/032035/pdf
[2] http://ccl.cse.nd.edu/research/papers/lobster-cluster-2015.pdf

WHY LOBSTER?

- We have access to a lot of CPUs!
  - ~25k cores at the Notre Dame Center for Research Computing
  - Machines belong to various PIs– available to Notre Dame users opportunistically
- We have access to Cooperative Computing Lab experts!
  - Team at Notre Dame, led by Doug Thain, that develops CCTools suite of software for large scale distributed systems
- Lobster is an R&D partnership between ND HEP and CS groups, making progress on general CS research problems in the context of real, complex, and demanding physics needs

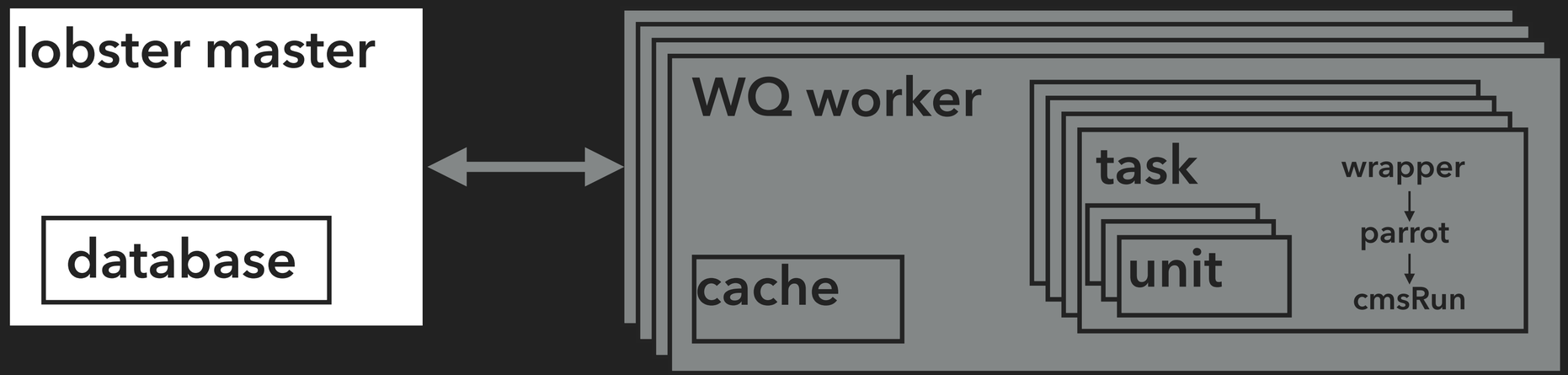
WHY IS THIS HARD?

- The Center for Research Computing is a heterogeneous environment:
  - Machines built and configured for owners' needs– all we can count on is (mostly) RHEL6
  - No OSG environment
  - No CMS software
  - No root access
  - Cluster owners' jobs evict opportunistic jobs without warning

CCTOOLS TO THE RESCUE!

- CCTools are user-level: no root access required!
  - parrot: transparent user-level virtual filesystem
    - allows us to access CVMFS
  - work queue: framework for building large scale master-worker applications
    - takes care of interactions with the batch system
  - chirp: userland file server
    - used for transferring input and output files

LOBSTER ANATOMY



master

- started by user
- tracks workers
- performs unit accounting
- assembles tasks on-the-fly (allows for dynamic task sizes, reconfigurable parameters, isolating problematic units…)

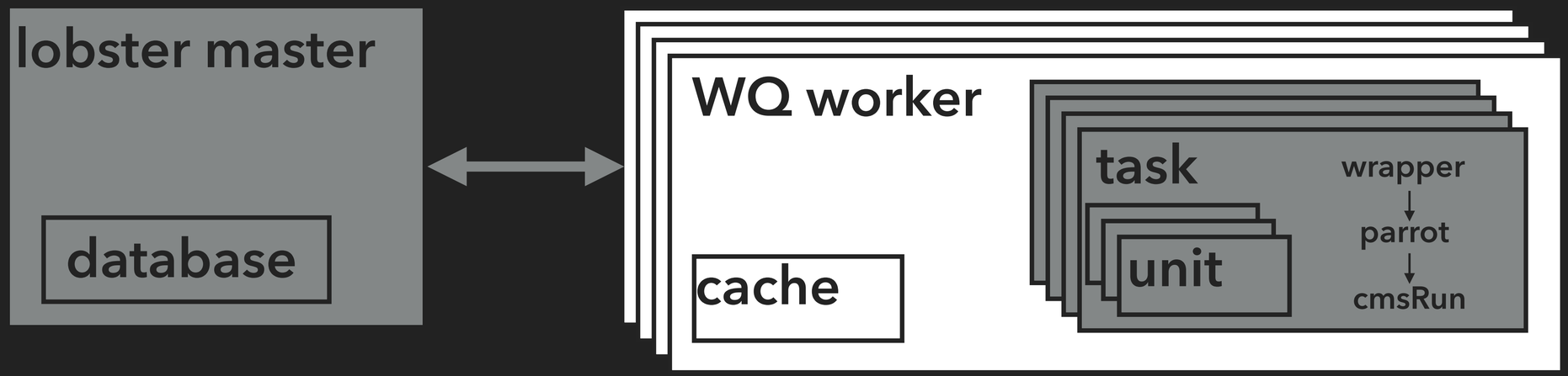
WQ worker

- user starts a factory to submit workers
- runs tasks
- provides cache (shared and reused by tasks)

task

- runs a wrapper which starts Parrot if needed
- sets up working environment
- executes cmsRun

LOBSTER ANATOMY



master

- started by user
- tracks workers
- performs unit accounting
- assembles tasks on-the-fly (allows for dynamic task sizes, reconfigurable parameters, isolating problematic units…)

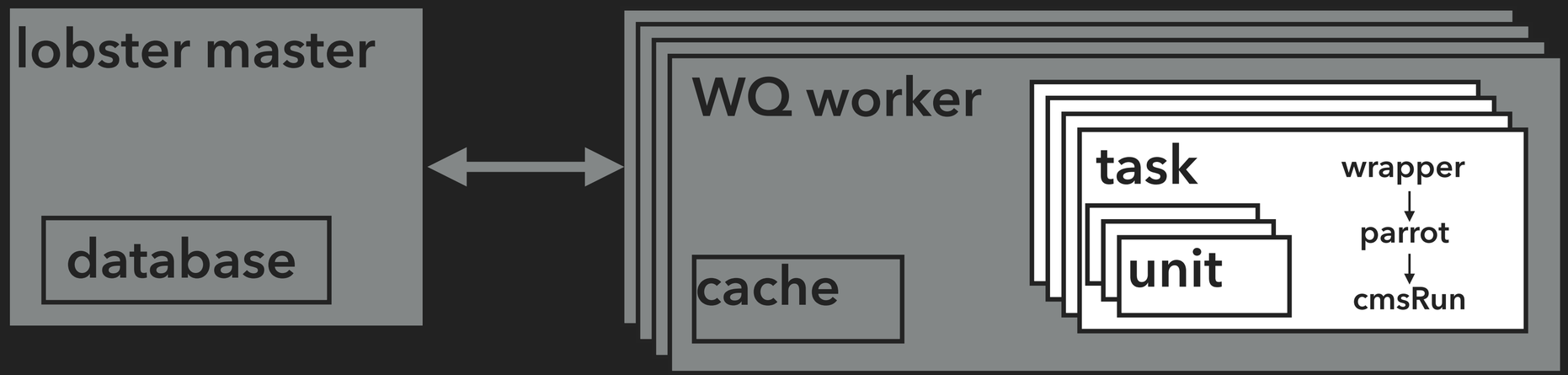
WQ worker

- user starts a factory to submit workers
- runs tasks
- provides cache (shared and reused by tasks)

task

- runs a wrapper which starts Parrot if needed
- sets up working environment
- executes cmsRun

LOBSTER ANATOMY



master

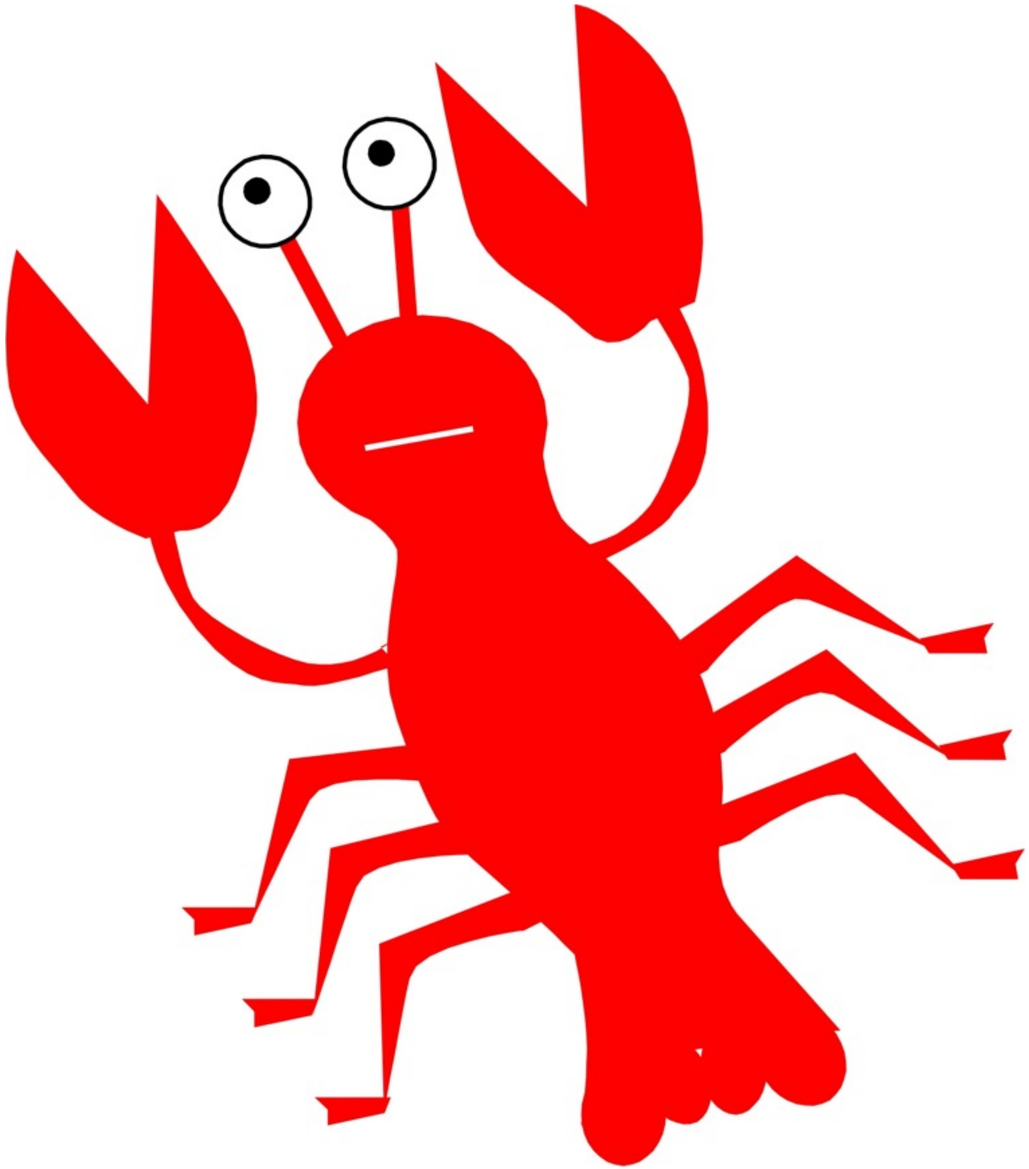
- started by user
- tracks workers
- performs unit accounting
- assembles tasks on-the-fly (allows for dynamic task sizes, reconfigurable parameters, isolating problematic units…)

WQ worker

- user starts a factory to submit workers
- runs tasks
- provides cache (shared and reused by tasks)

task

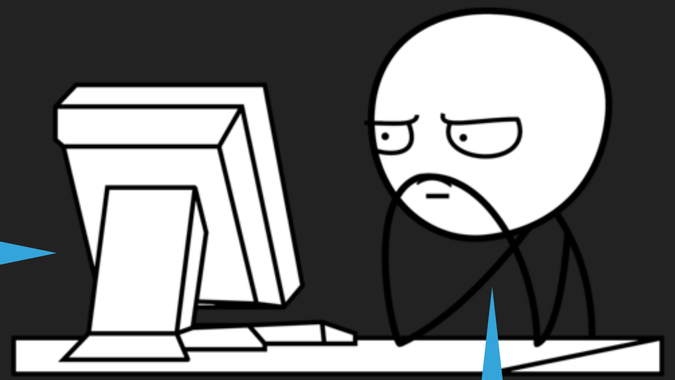
- runs a wrapper which starts Parrot if needed
- sets up working environment
- executes cmsRun



WHAT'S NEW?

WHAT DO USERS KNOW?

YOU ALMOST BROUGHT DOWN THE CLUSTER BY COMPLETELY FILLING UP ALL OF THE LOCAL DISKS!!!!!!!!! HOW MUCH {DISK, CPUS, MEMORY} DOES EACH JOB NEED, ANYWAYS?



SYSADMIN

DO YOU KNOW ANYTHING ABOUT YOUR JOBS?

PHYSICIST

UHM...

I KNOW THAT THESE ARE GEN JOBS, AND THOSE ARE RECO JOBS!

WHAT KIND OF JOB THEY ARE RUNNING!

CATEGORIES

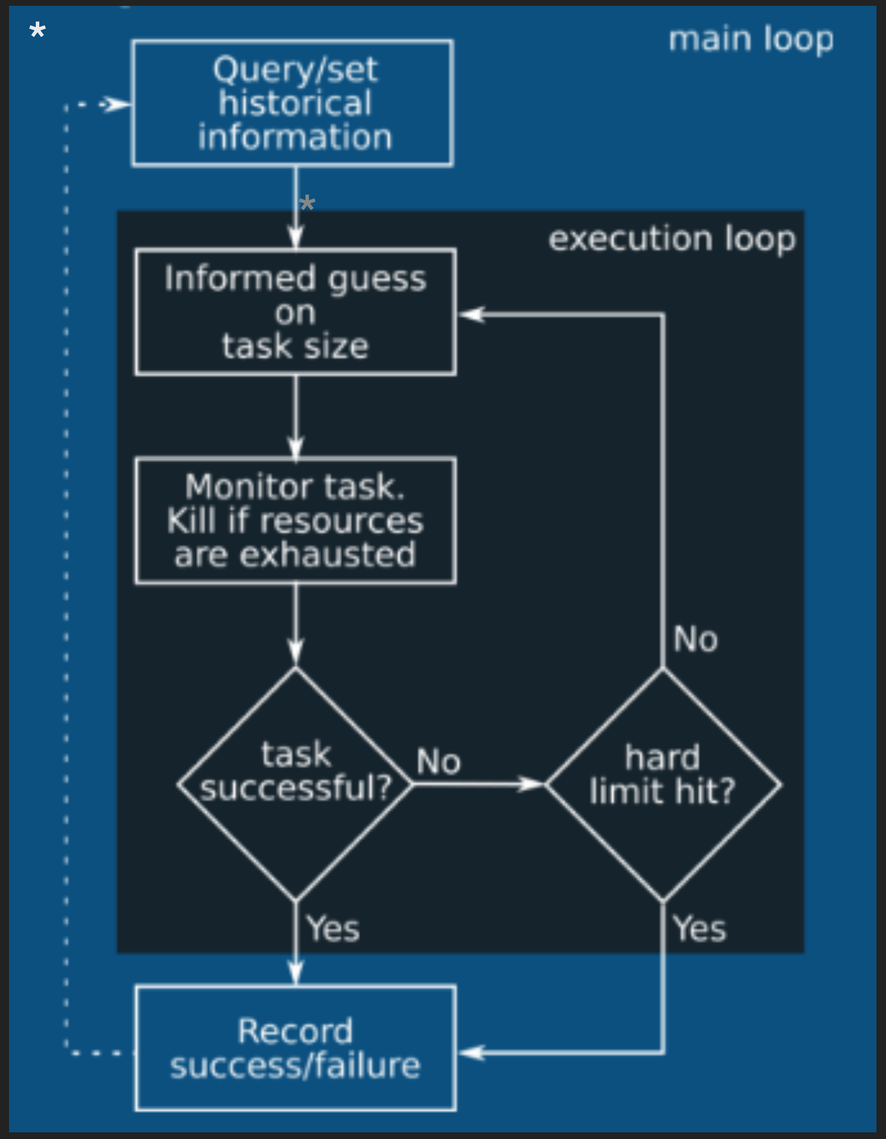
- Workflows can now be divided into categories based on the resources they need– optimizes packing!
- Per-category limits on concurrent jobs running to control utilization of communication bandwidth
- Runtime limits on each task category to voluntarily terminate tasks: better resilience against eviction!

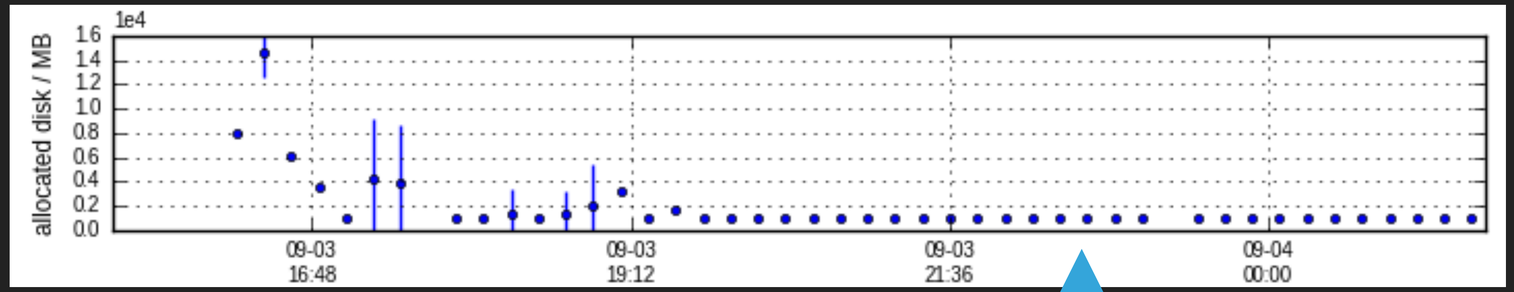
```
category=Category(
    name='lhe',
    cores=1,
    memory=2000
)
```

```
category=Category(
    name='digi',
    cores=1,
    memory=2600,
    runtime=45 * 60,
    tasks_max=100
)
```

```
category=Category(
    name='reco',
    cores=4,
    memory=2800,
    runtime=45 * 60,
    tasks_min=10
)
```

RESOURCE MONITORING FEEDBACK LOOP

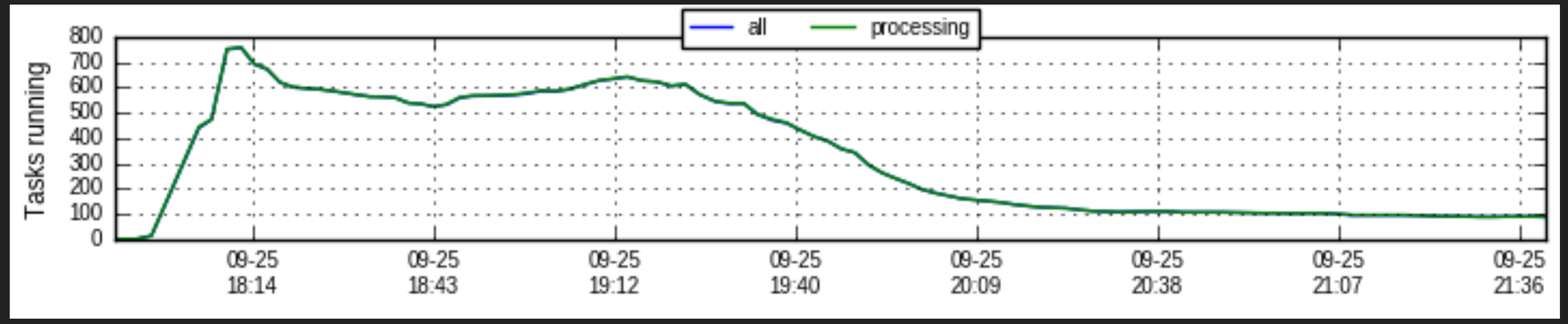




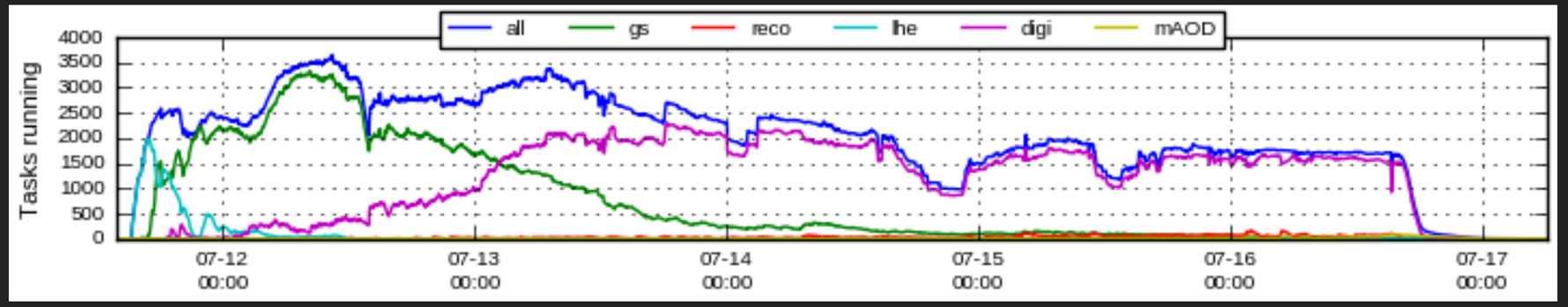
WQ isn't sure how much disk the task needs, so it starts out by taking up the whole worker. As tasks return, the disk allocated settles to the real requirement.

* From Ben Tovar's Condor Week talk: http://tinyurl.com/jcajbnd

WORKFLOW DEPENDENCIES



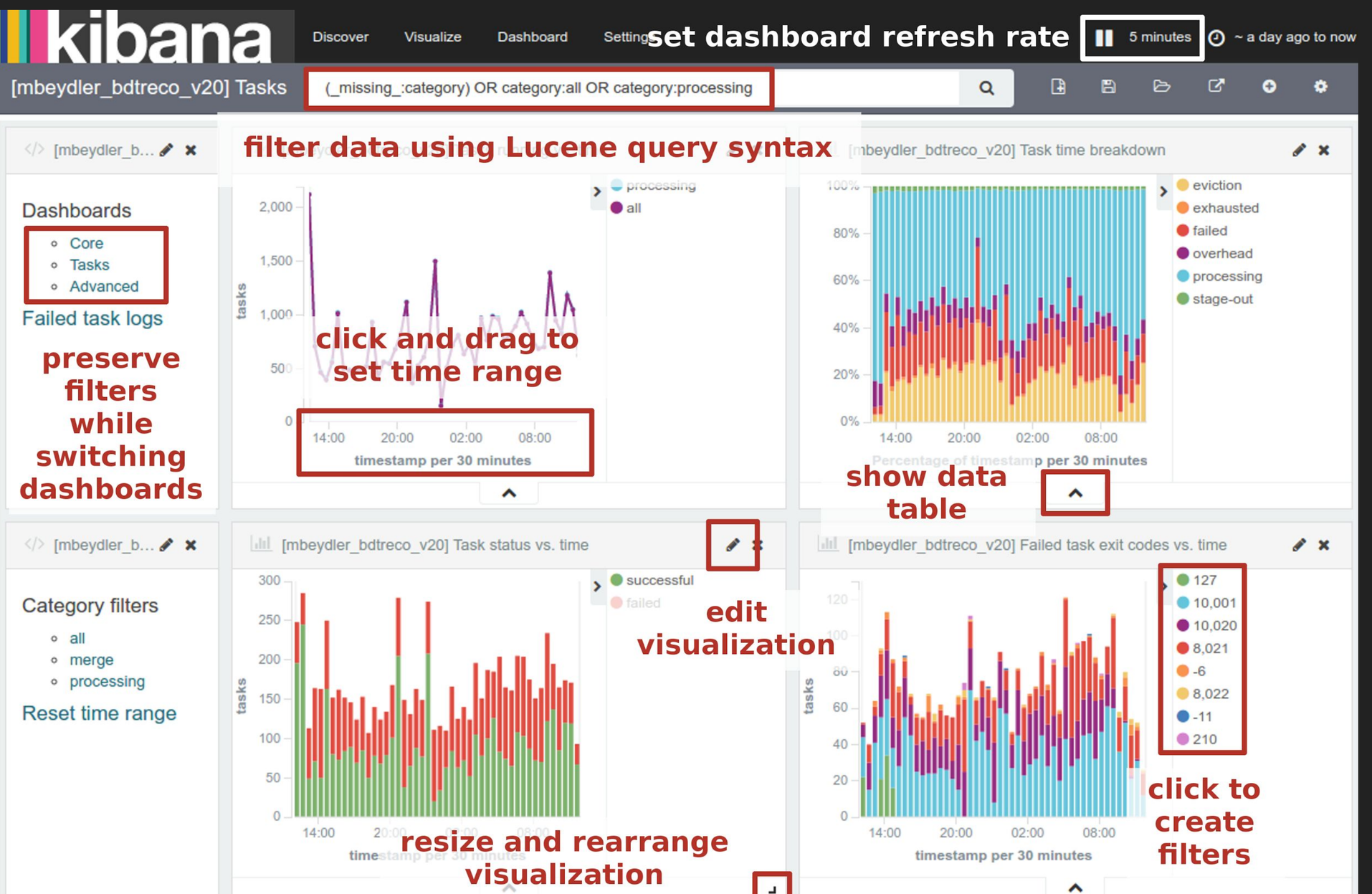
PROBLEM: Processing runs have tails. Consider a project with N workflows, where W_N depends on W_{N-1}. For N * (tail length) time, processing is running far below capacity!



SOLUTION: Allow user to specify workflow dependencies. Submit a task for the next workflow as soon as input data comes in!

ELASTICSEARCH! LOGSTASH! KIBANA!

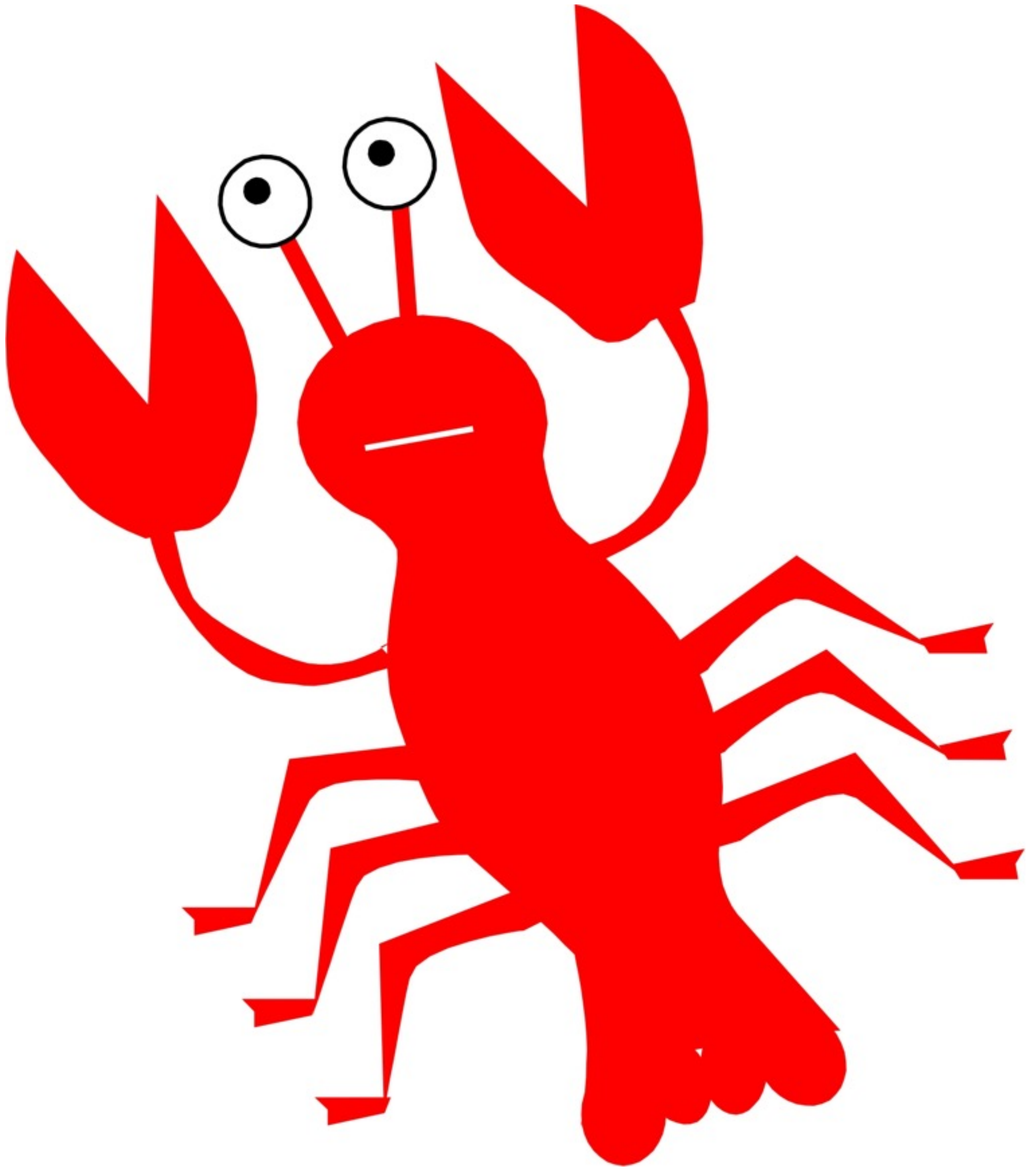
SUMMER DISC REU BY ANNA YANNAKOPOULOS.
SEND US YOUR UNDERGRADS!



Monitoring has been implemented in ELK... Huge improvement!

CONCLUSION

- The capability of Lobster has been significantly increased– a few highlights have been presented:
  - Task categories
  - Resource monitoring feedback loop
  - Workflow dependencies
  - Improved monitoring with ELK
- Lobster is helping to make progress on general CS research problems in the context of real, complex, and demanding physics needs
- For more Lobster fun, see our poster: “Scaling Up a CMS Tier-3 Site with Campus Resources and a 100 Gb/s Network Connection: What Could Go Wrong?”



BACKUP

LOBSTER ANATOMY