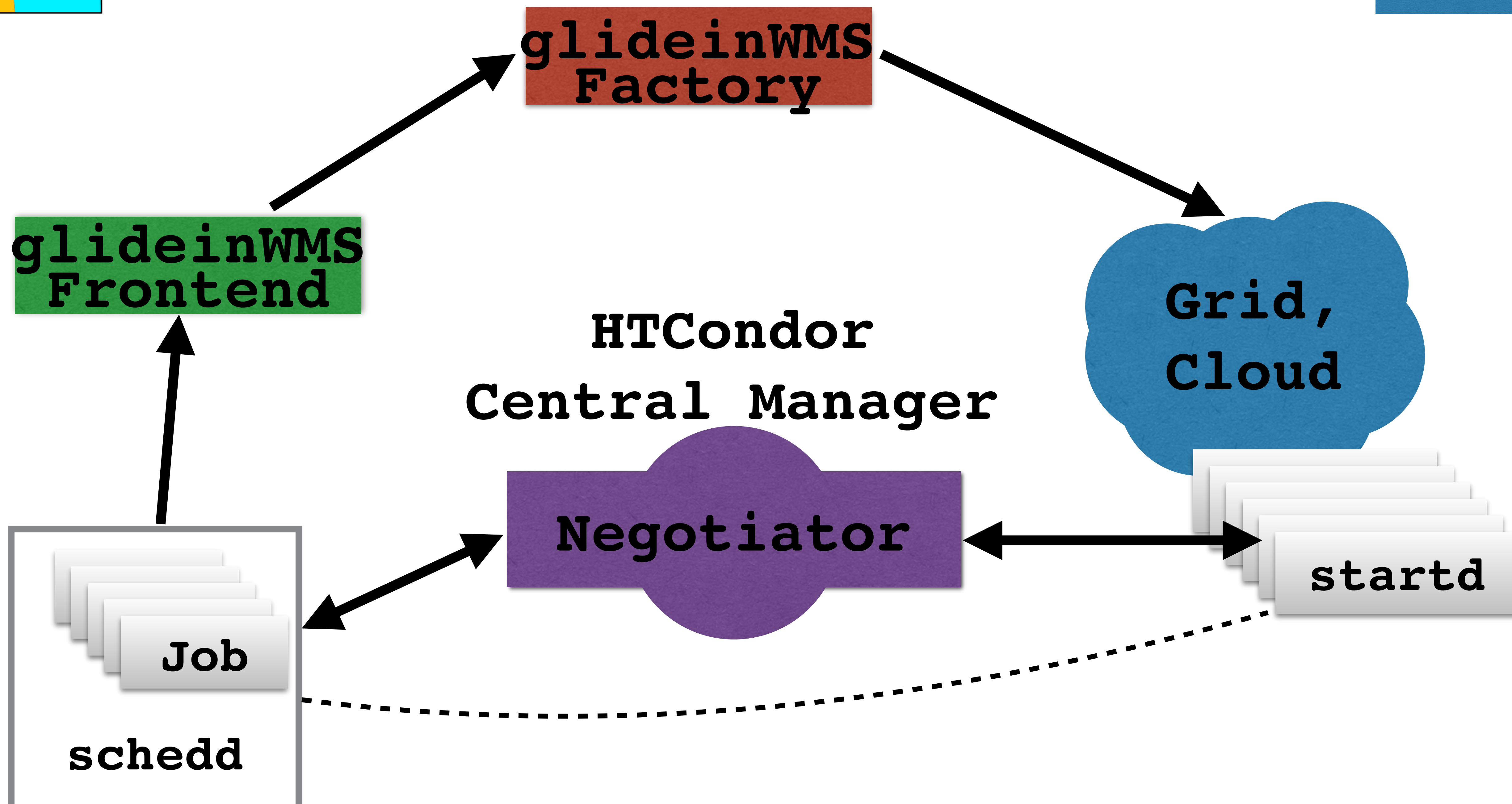
Reaching New Scales with the CMS HTCondor Global Pool

JAMES LETTS & ANTONIO PÉREZ-CALERO
on behalf of the Submission Infrastructure Group
of the CMS Experiment

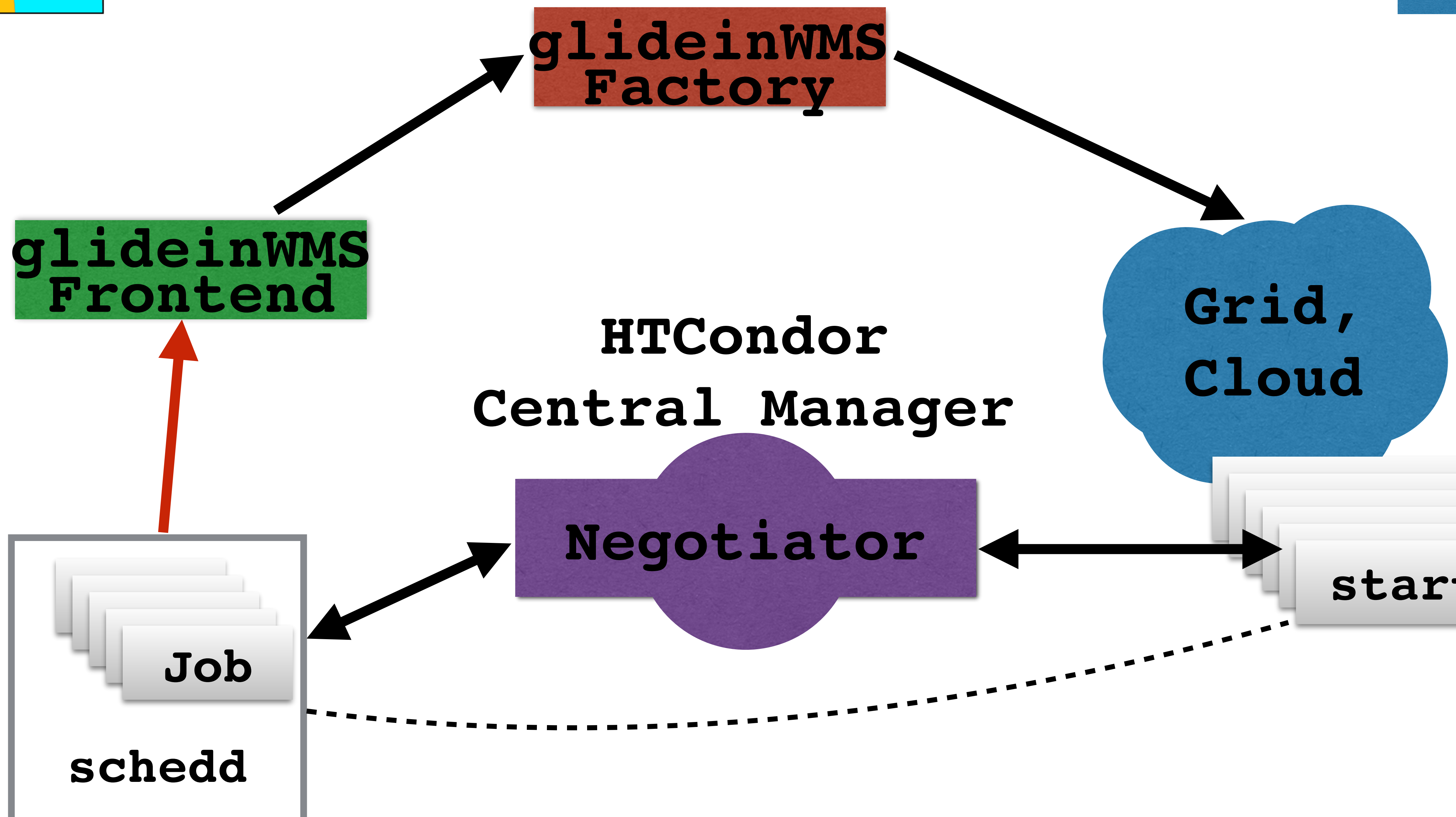
May 3, 2017

Global Pool



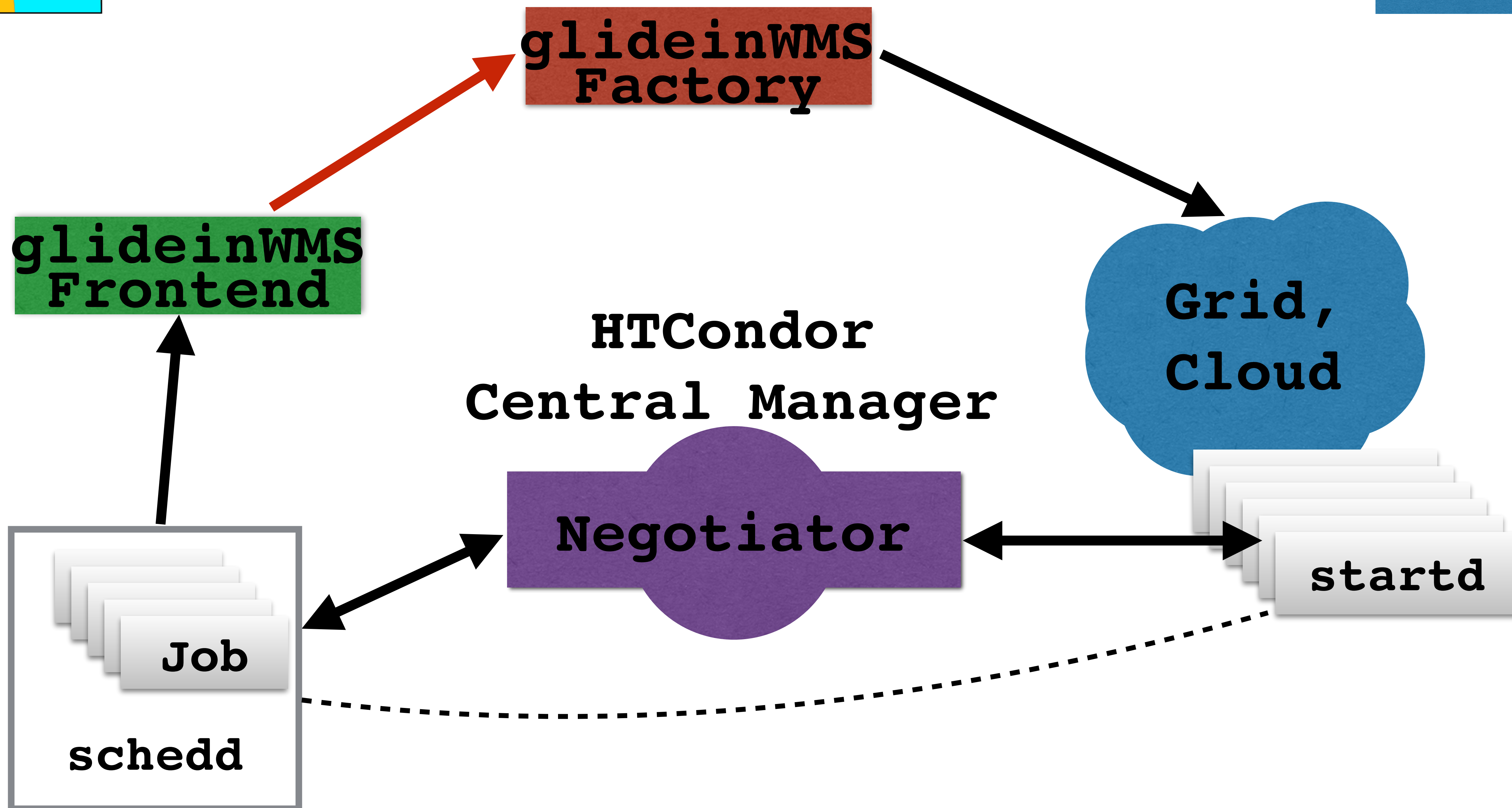
A global HTCondor pool provisioned by glideinWMS.

Job Pressure



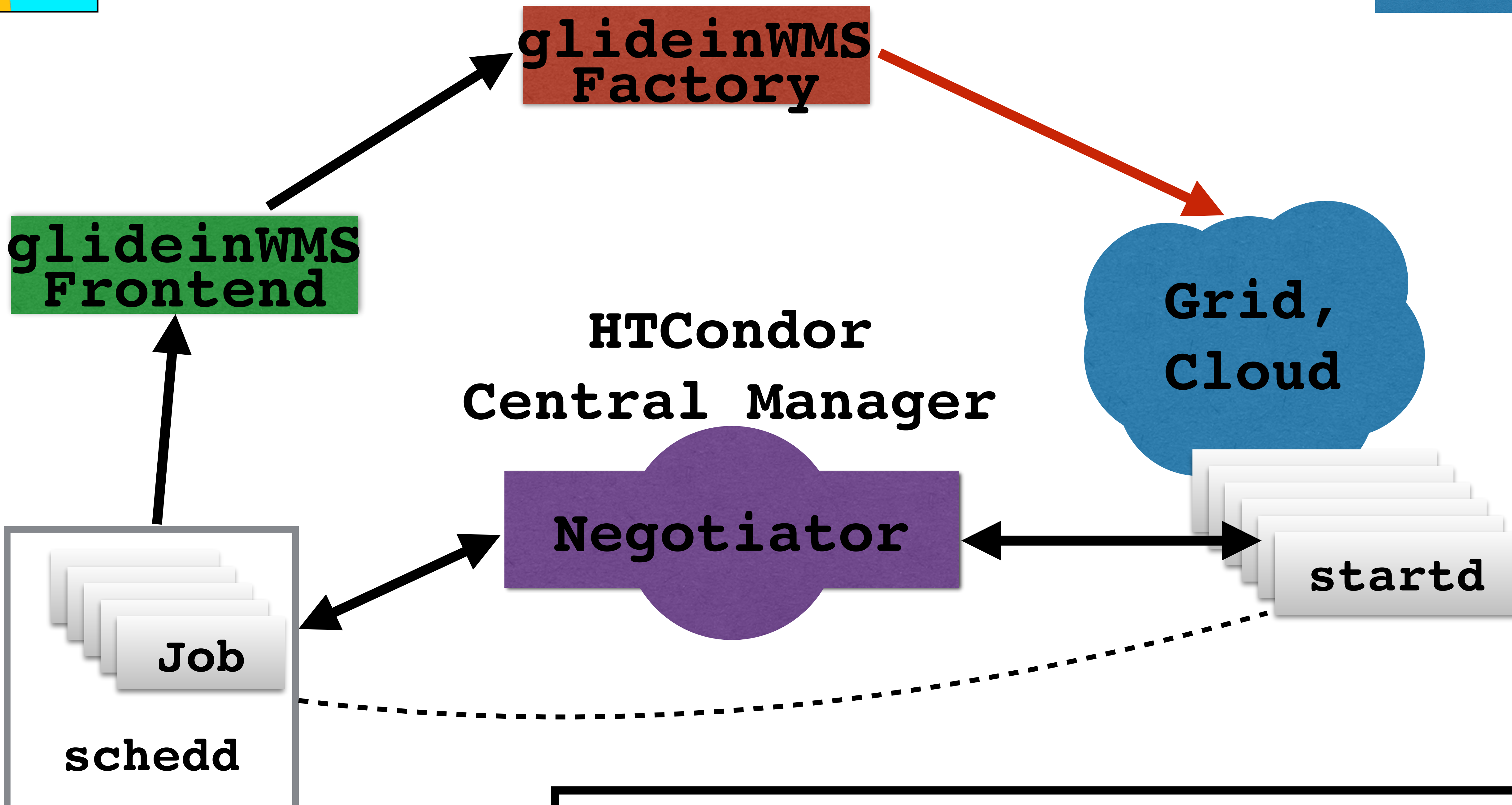
The Frontend reads the job queues on the schedd's.

Resource Request



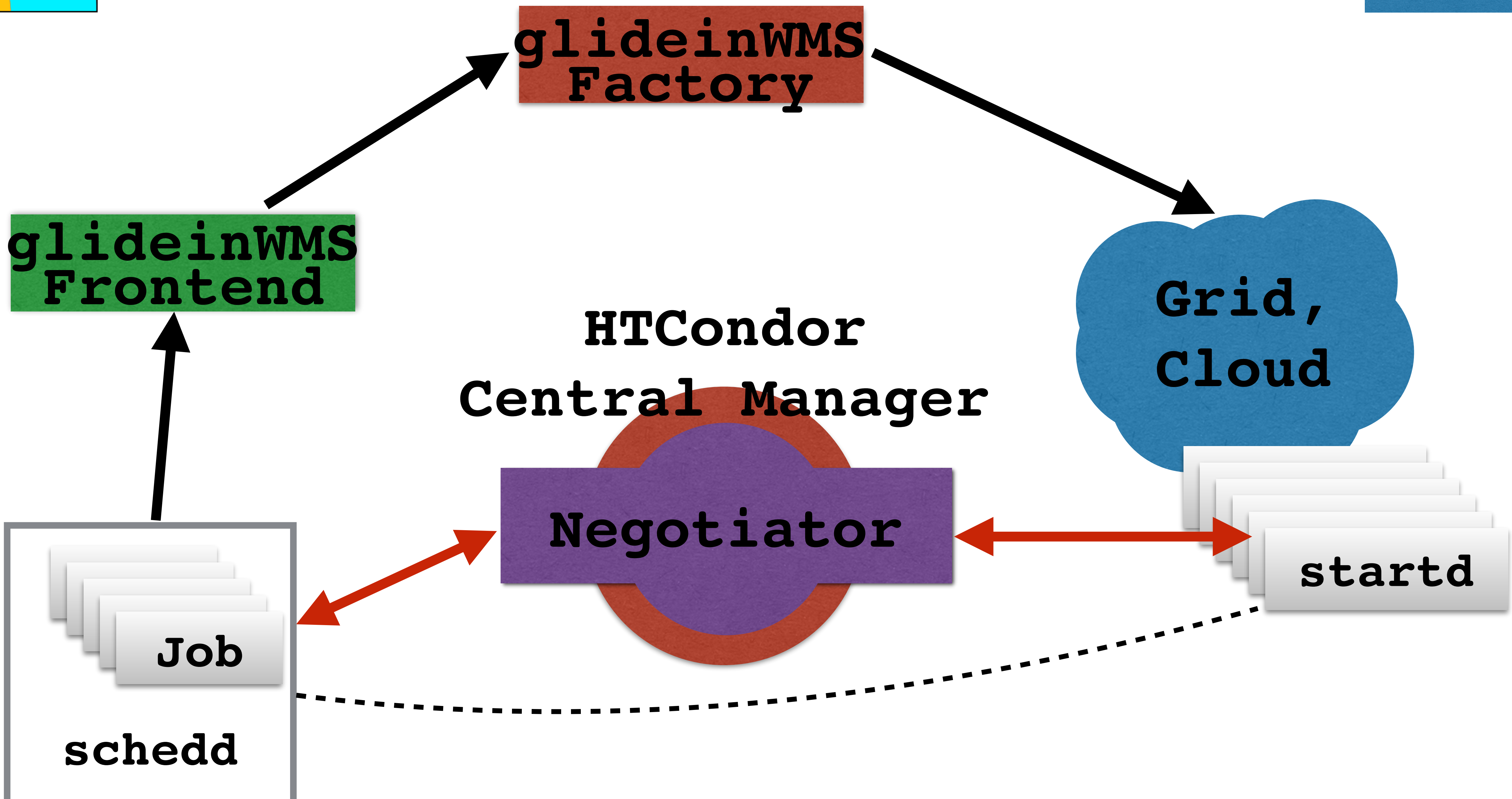
The Frontend then requests the factory for startd's on the target sites.

Pilot Submission



The Factory then submits (with HTCondor) pilot jobs to the target sites which launch startd's registered in the Global Pool.

Negotiation



The Negotiator then matches jobs in the schedd queues with startd's.

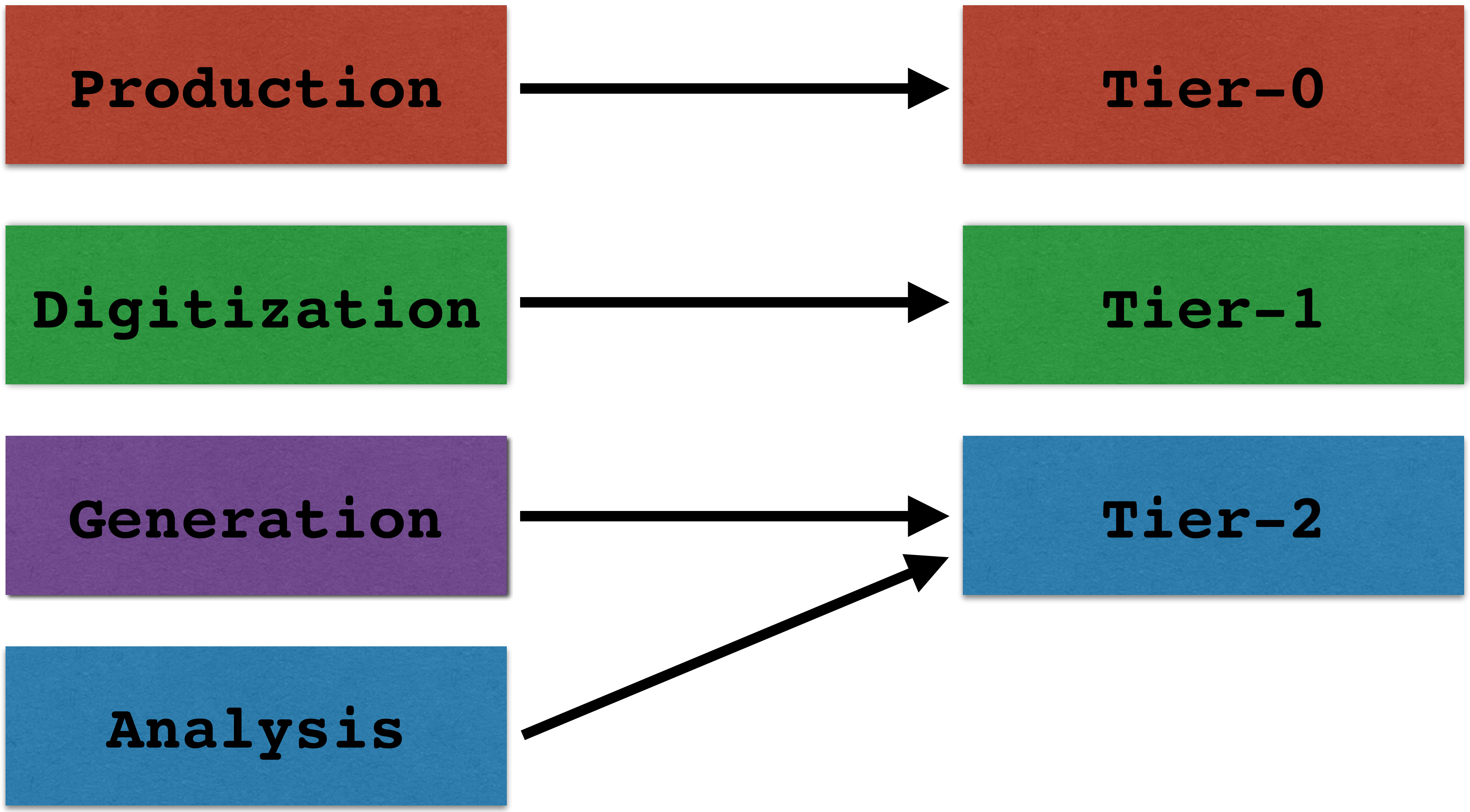
Why a Global Pool?

- Before the Global Pool, we had different HTCondor pools for data analysis and production activities.

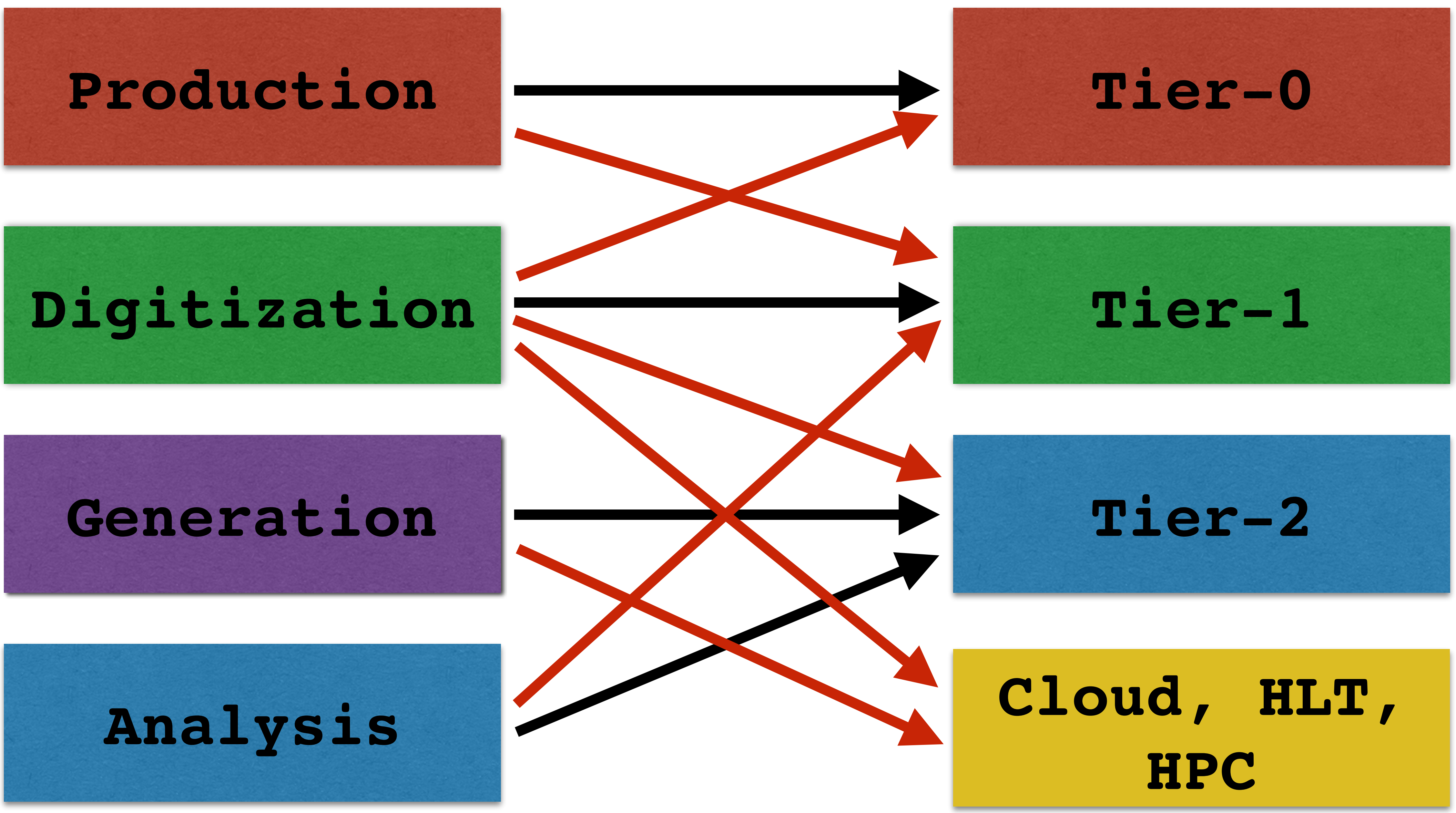
- As we moved away from the original tiered LHC Computing Model, we wanted to flexibly run different types of workflows across tiers, as well as integrating new types of Cloud and allocation-based resources.

Tiered (MONARC) Model

when networking was a scarcer resource



Global Model



Why a Global Pool?

- We now have a unified, multi-core, HTCondor Global Pool with re-usable pilots.

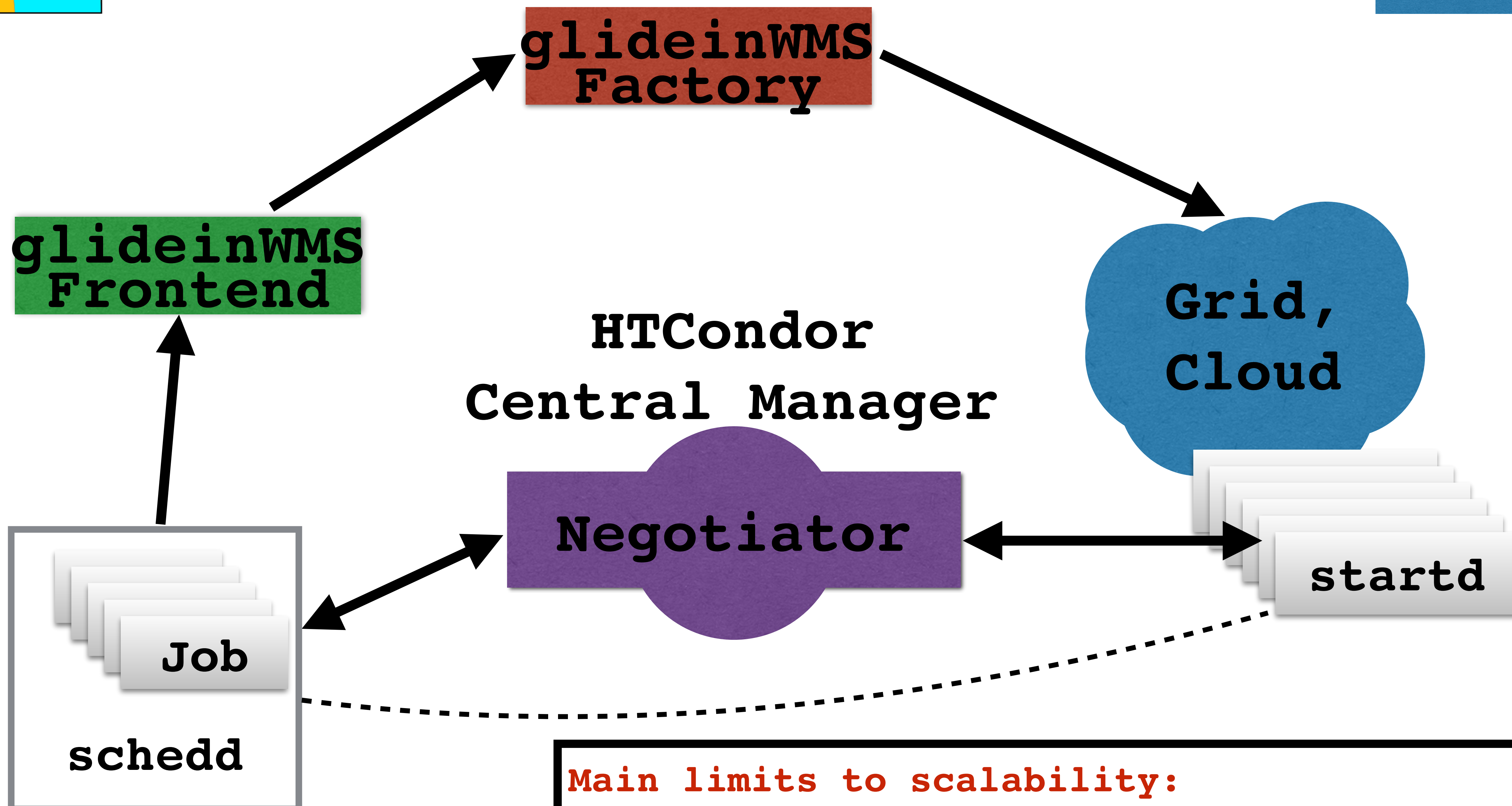
- For reasons of stability for CMS data taking, the Tier-0 has its own separate pool.

- CMS is also moving away from data locality: AAA, xrootd, caching, etc.

- Makes resource scheduling much more complicated! CPU, Memory, Disk, and now I/O. Network is still scarce in places.

- In 2016, CMS moved into a resource-constrained environment for the first time.

Scalability



Main limits to scalability:

- I/O between and within these components
- Speed of individual components

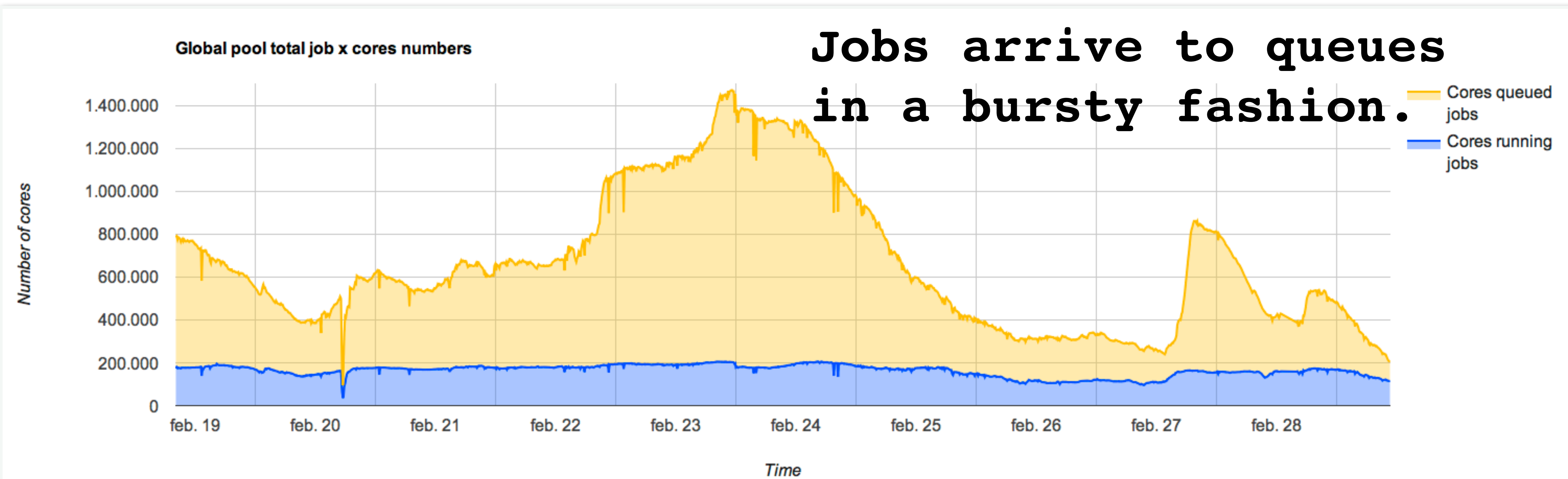
Driven by combinatorics of Matchmaking: (RRLs x startd's)

Auto-clusters

- Multi-core and longer-running jobs ease the scale limitations: fewer jobs in the system.
- Unfortunately, arrival of analysis and production jobs is very chaotic:

Auto-clusters

- Multi-core and longer-running jobs ease the scale limitations: fewer jobs in the system.
- Unfortunately, arrival of analysis and production jobs is very chaotic:



Challenges

- Most recent scalability limits over the past few years have been found in the Central Manager.

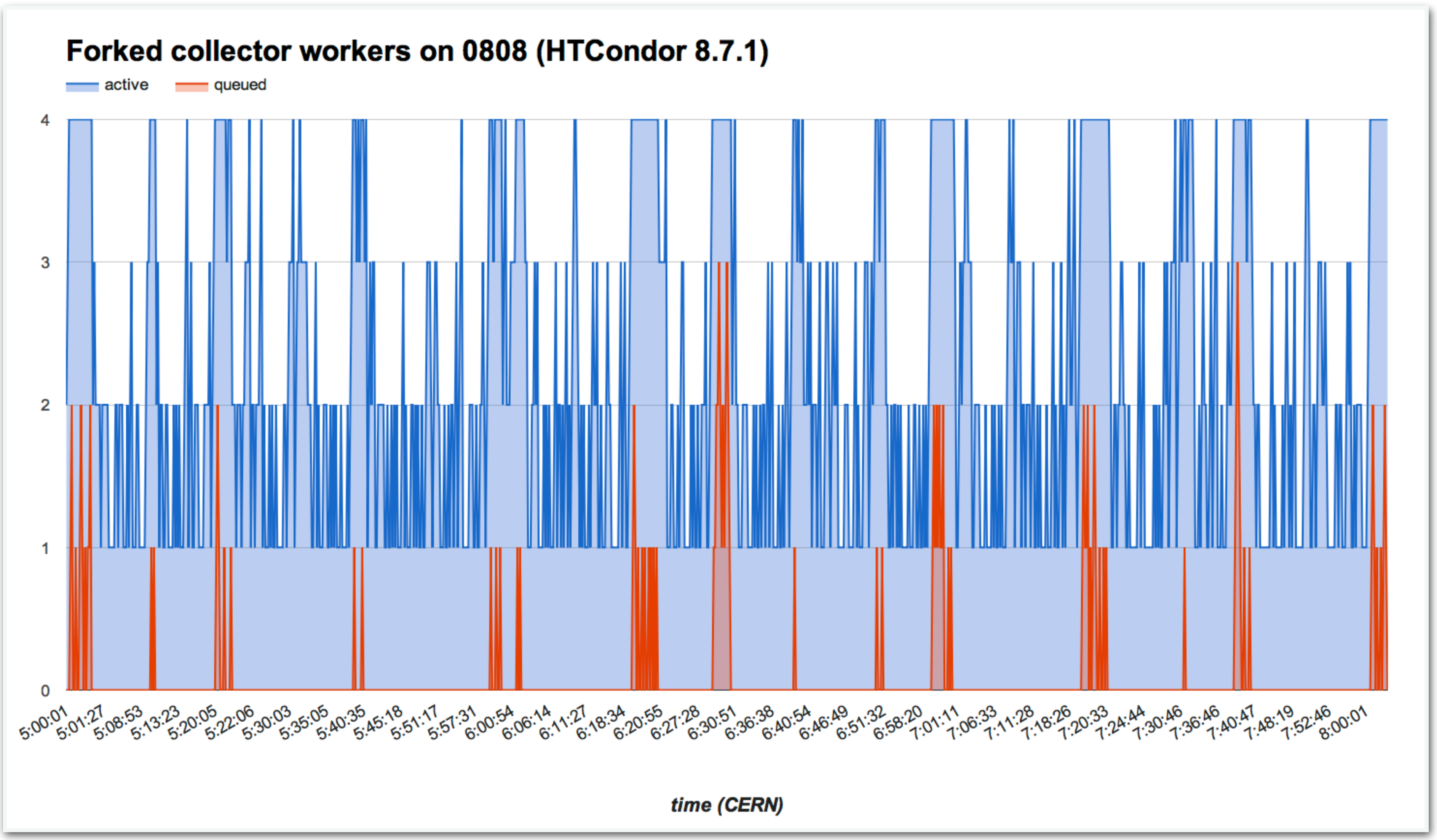
- Scale tests with single-core jobs and conducted with the OSG in 2014 at 200,000 static slots found that separation of the CCB's onto hardware separate from the Central Manager was essential to go beyond 150,000 CPUs (in the lab).

Challenges

- However, in the wild, a new blocker arose in 2016 at ~155,000 CPUs in a multi-core environment.
- Symptom: Central Manager machine dropping UDP updates.
- CMS worked closely with the HTCondor developers to study the problem. Finally found out that the Top Collector was being blocked by **queries.**
- Limited number of forked query workers exhausted, remaining queries were blocking the Top Collector.

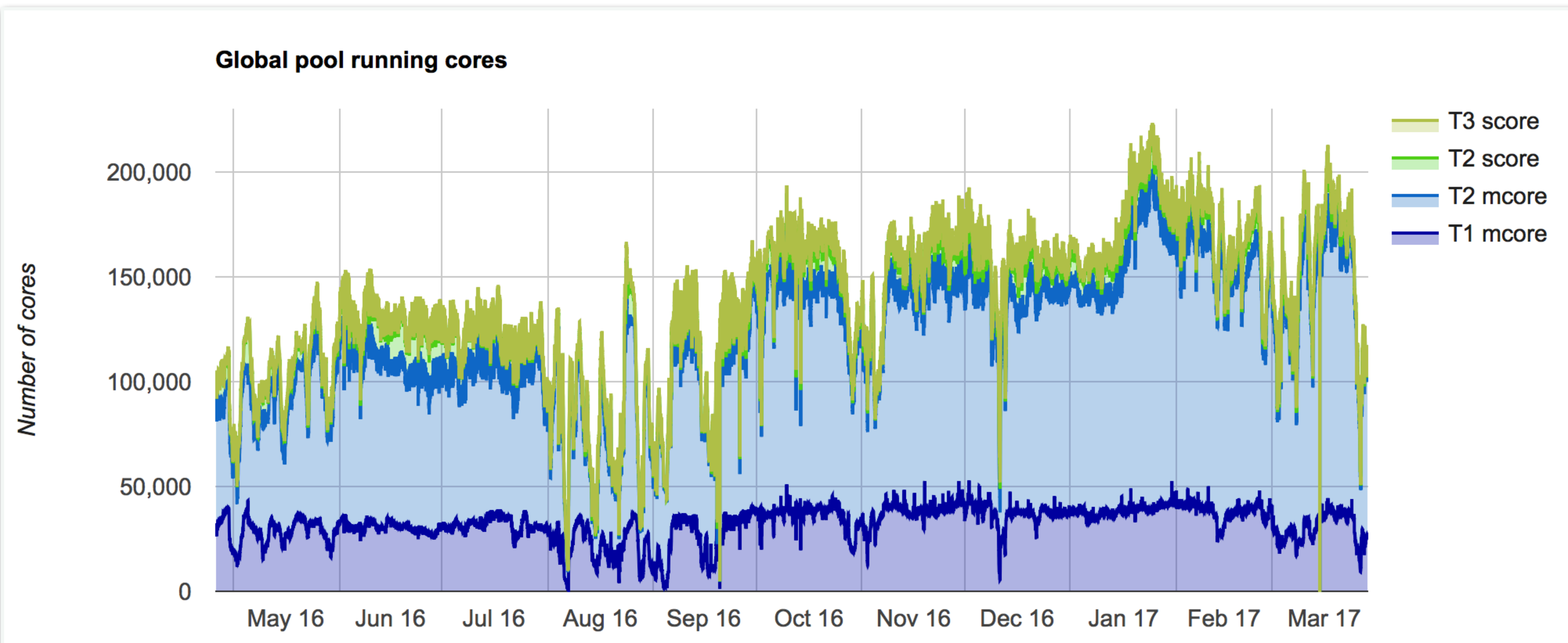
Solution

- Developers' solution was to queue updates, not let them go to the Top Collector, and prioritize queries from the Negotiator.



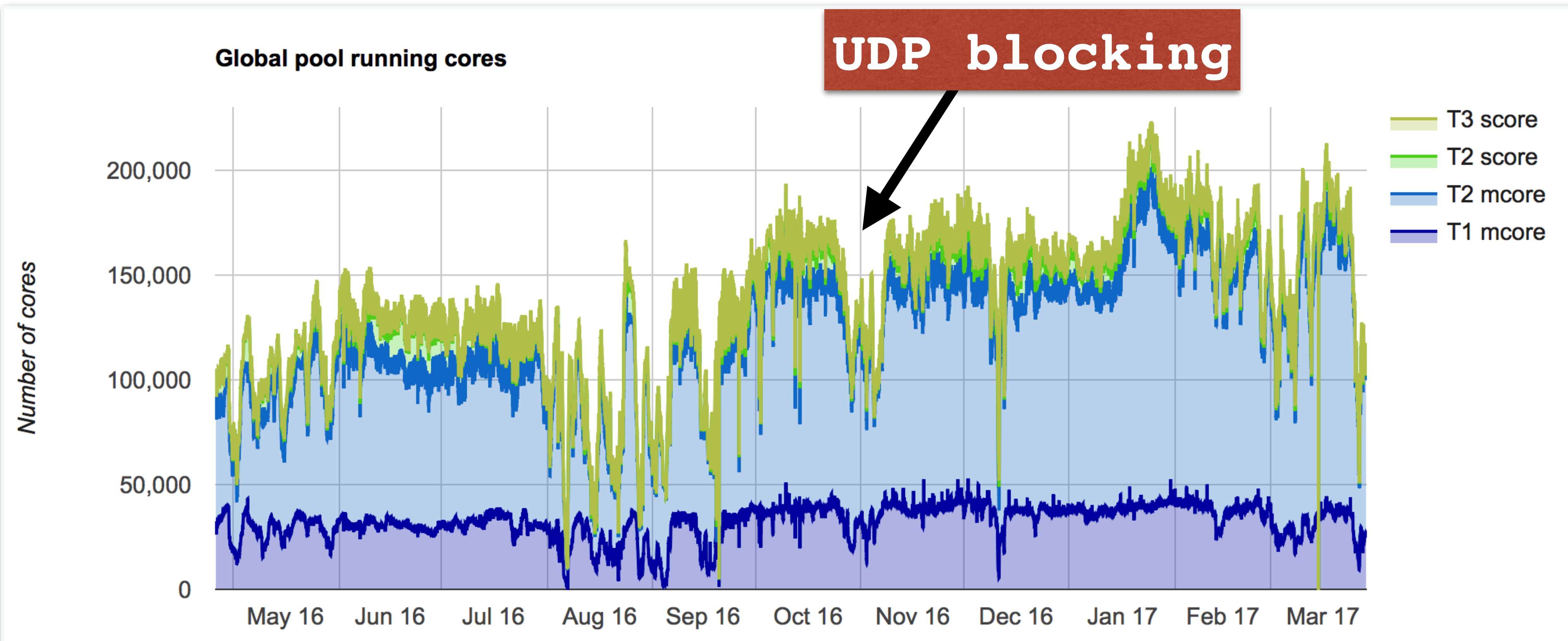
Evolution

- In the past year we have gone from 100K CPU cores to a peak of 240K.



Evolution

- In the past year we have gone from 100K CPU cores to a peak of 240K.



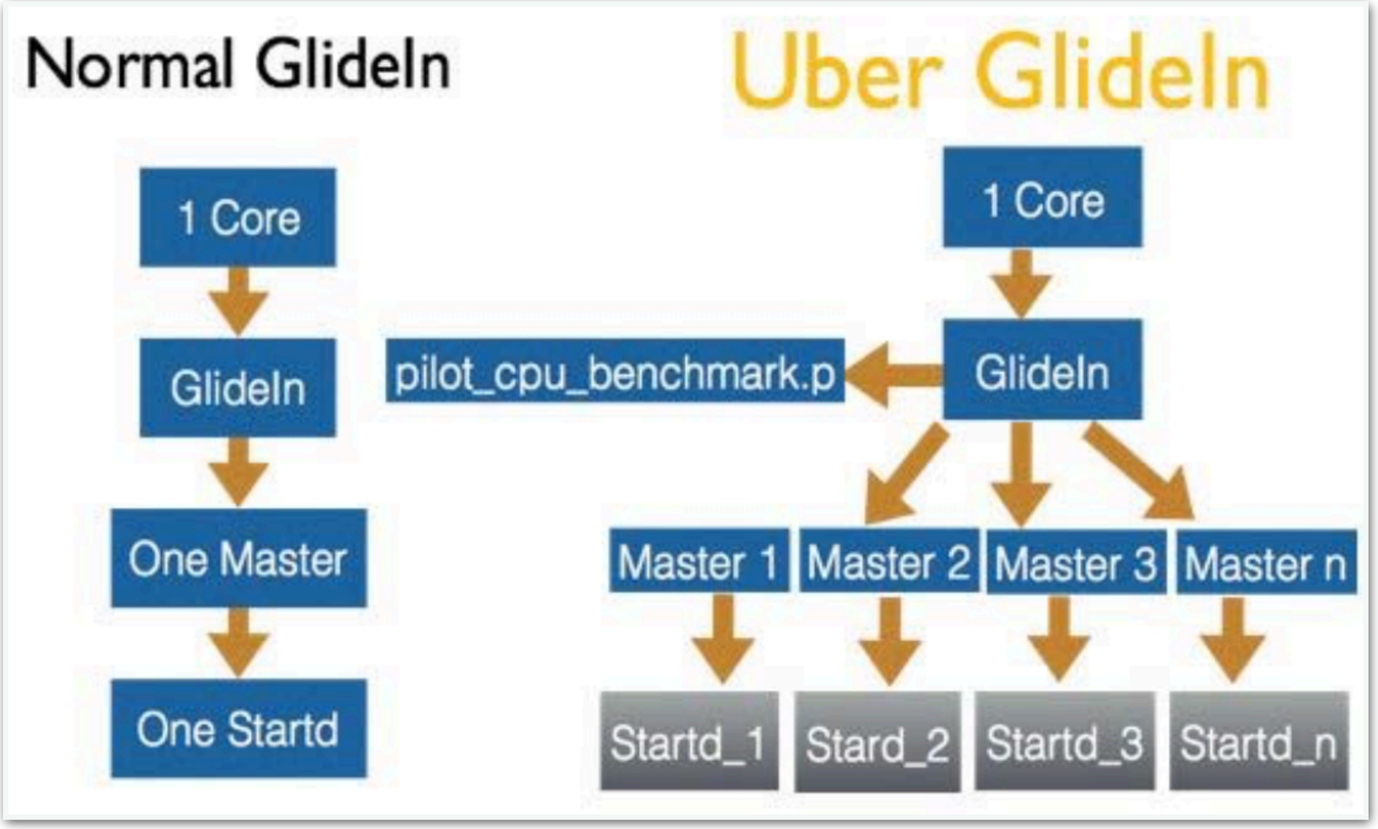
Earlier Issues

- Negotiator scalability has been an issue we have encountered several times so far.
- Since the beginning of 2015 we run **3 separate Negotiators** in parallel based on resource type: Tier-1, U.S. Tier-2, and the rest. Each group ~80K CPU cores.
- Allows us to do resource-based fair share: production gets 95% of the Tier-1 sites, while rest are 50% physics analysis.
- HTCondor developers have been parallelizing the Negotiator even more since.

Scale Tests

- In principle, CMS wants to find and fix blockers in scale tests, not in a production system.

- Last round of scale tests with OSG in 2014 used the concept of "über-glideins": one pilot launches multiple startd's, thus achieving the I/O of a much larger pool:



- Using a factor of 32, can reach the I/O of 500K startd's using only 15,625 physical cores.

2017 Scale Tests

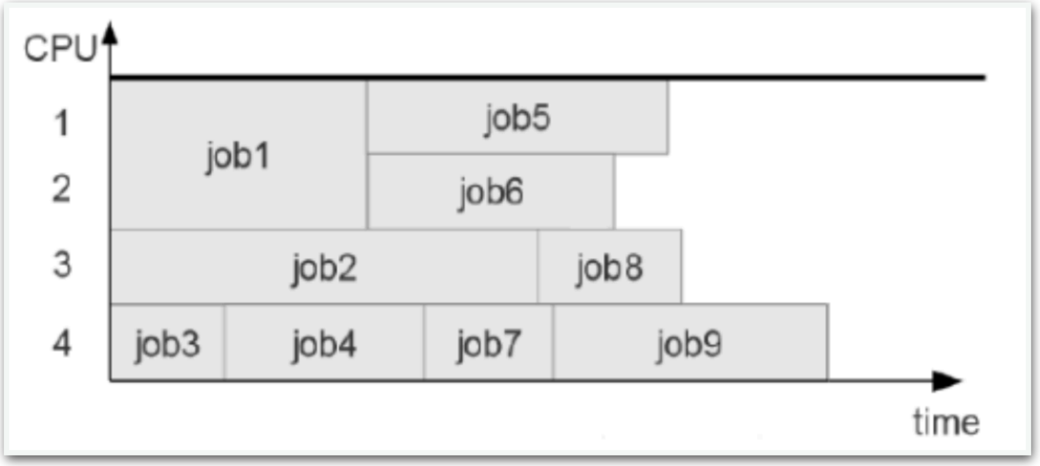
- In principle CMS can do something similar within the existing Global Pool, since we are not worried about the scalability of glideinWMS so much as HTCondor.
- Planning the next round of tests for August, which is historically a low period of usage after the major summer conferences, on new, beefier VMs provided by CERN/IT (~96GB RAM).
- Hope to find the next blockers in a test environment rather than in the wild!

Future

- Future challenges besides global scalability that we are facing for 2017-2018:
- Scheduling I/O: Now that individual sites are approaching 50-100K CPU cores, how can we not kill the network or the storage at the site? Or even across groups of sites? (CMS is moving away from data locality)
- Improving scheduling efficiency (next slide):

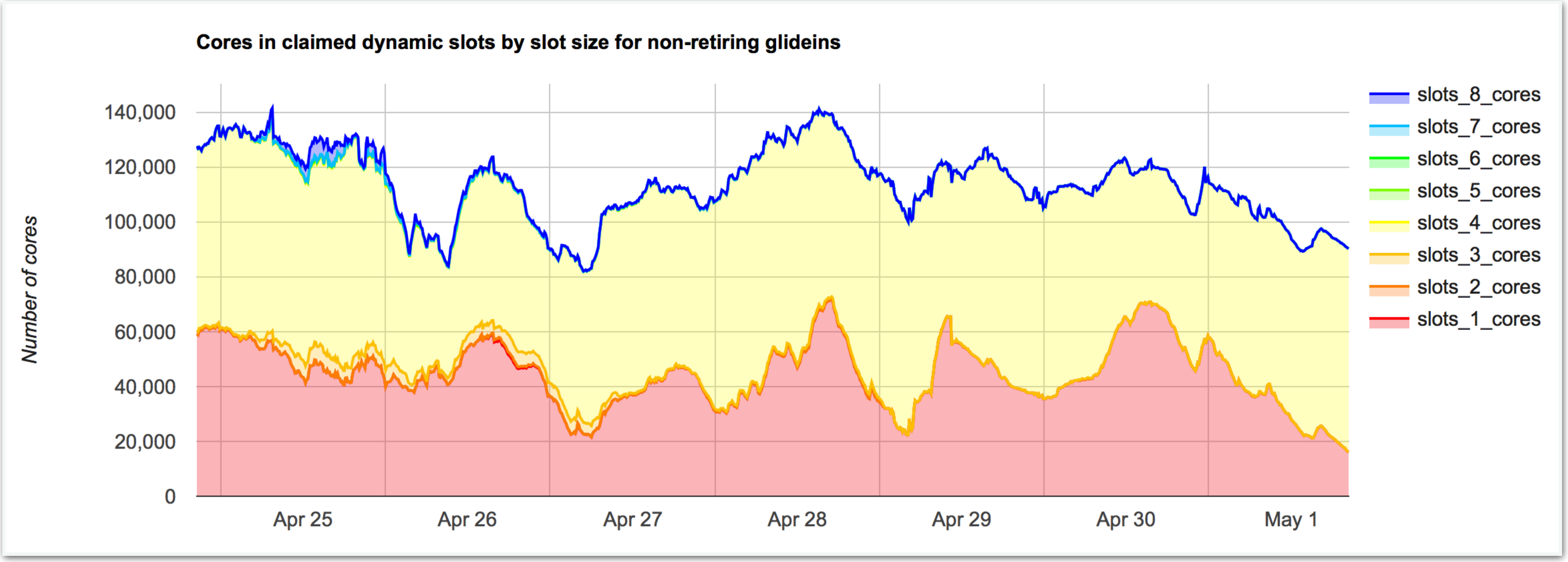
Improving Scheduling Efficiency

- Filling multi-core p-slots is a multi-dimensional problem:
  - Job requirements (e.g. time, CPUs, memory, resizable jobs)
  - Bursty nature of job arrival (time)
  - Resource constraints (CPUs, Memory)
  - Fair-share and priority (ranking)
  - Pilot lifetime (time)



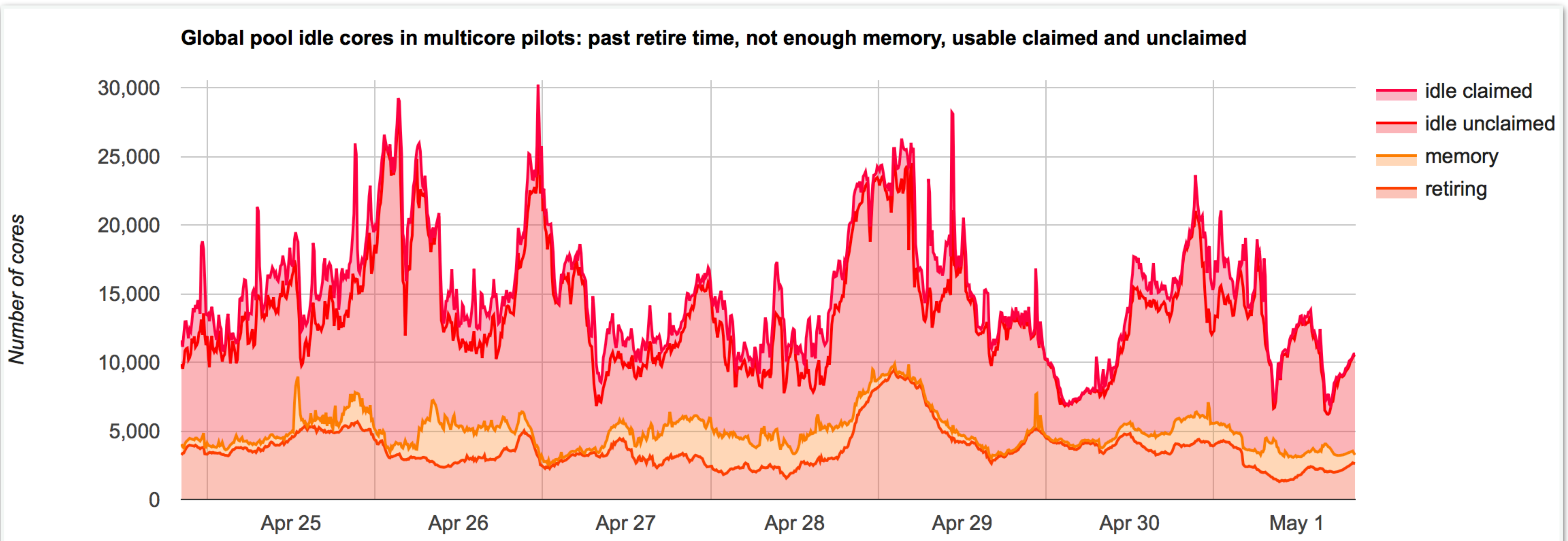
Improving Scheduling Efficiency

- Pool partitioning evolves over time to serve the demand.
- The challenge is that sometimes more CPU cores than desired are left unscheduled.



Improving Scheduling Efficiency

- There is a ~5% irreducible amount of wasted CPU from retiring glideins (p-slots). Tunable?
- High-memory jobs can take all of the RAM of a pilot, so that (justifiably) some CPU will be left unused.



Conclusions

- We thank the HTCondor development team for their close collaboration.

- We have met some interesting scalability and stability challenges over the past couple of years and look forward to reaching even greater heights in the years to come.