One Pool To Rule Them All

The CMS HTCondor/glideinWMS Global Pool

D. Mason for CMS Software & Computing

- Going to try to give you a picture of the CMS HTCondor/ glideinWMS global pool
- What's the use case — what problem are we trying to solve
- How we're solving it — i.e. the global pool
- Building the thing, how well its working
- Obligatory prognostication

CMS is a particle physics experiment at the CERN LHC

Alps

Gran Sasso

Lake Geneva

Geneva

CNGS ν's

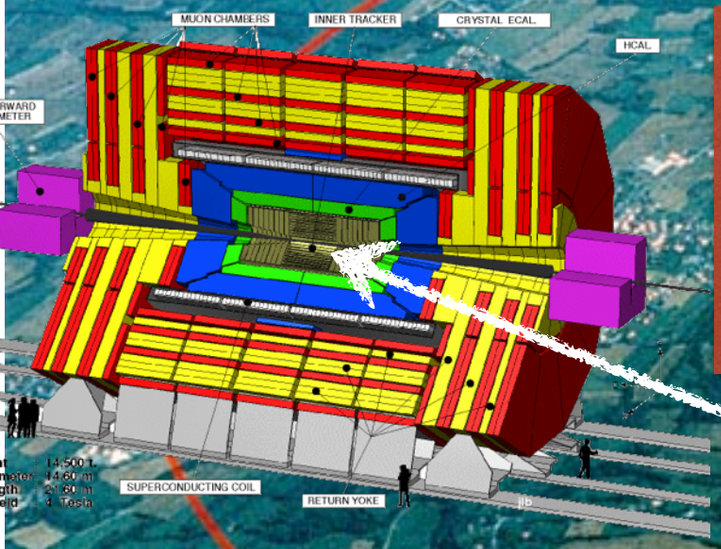
Geneva Airport

CMS

LHCb

CERN Meyrin

ATLAS



Millions of protons at very high energy collide every second

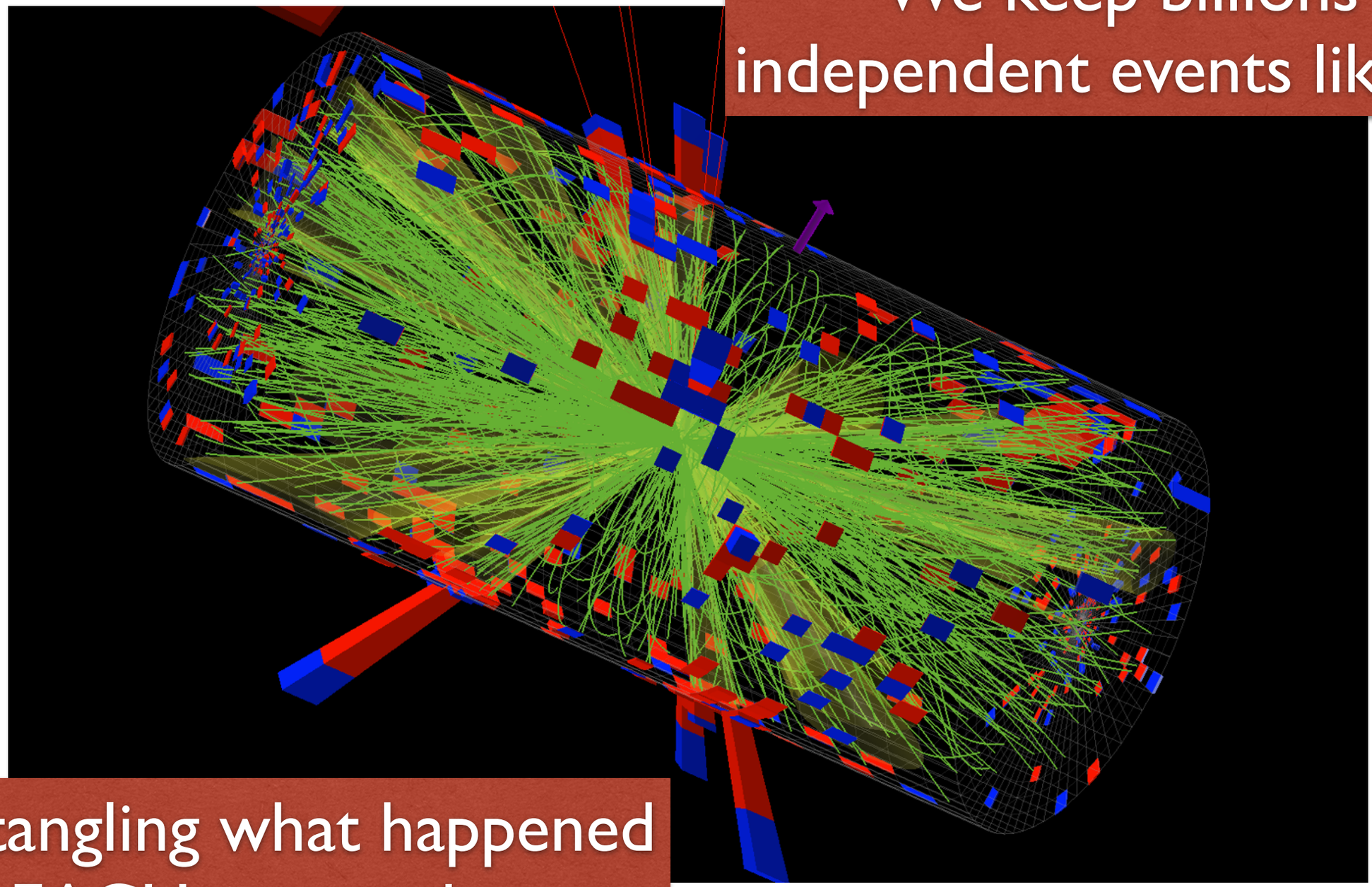
ALICE

Here.

Our detector records collisions we think are interesting
These are "events"

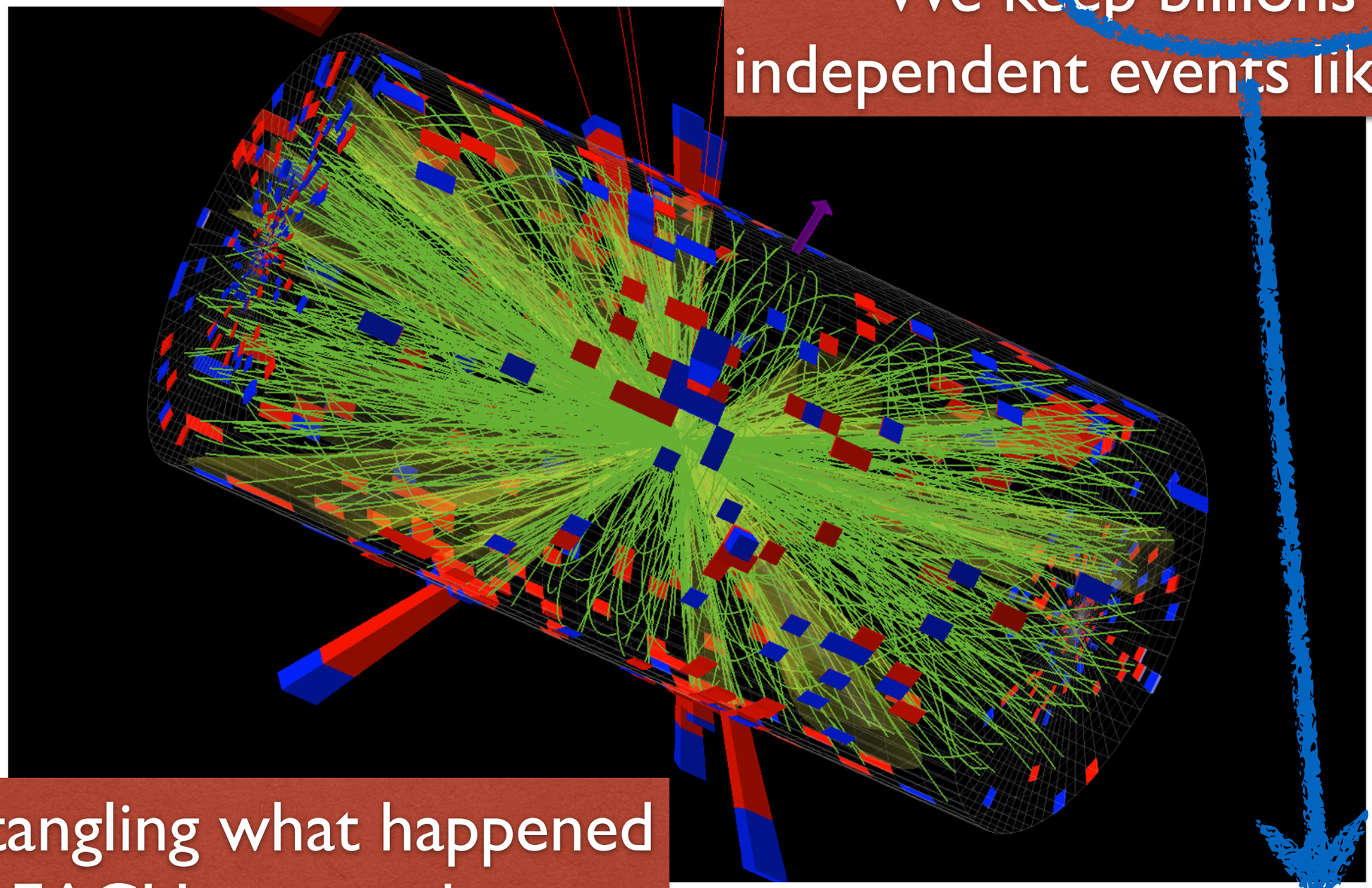
LHC

We keep billions of independent events like these



Disentangling what happened in EACH event takes ~minutes of CPU

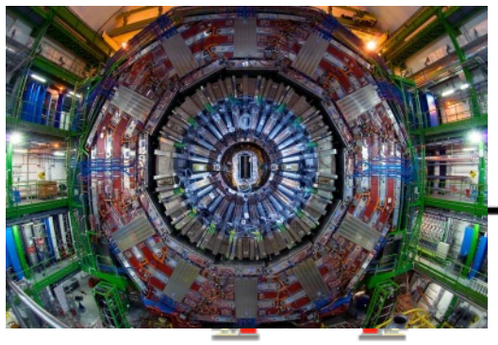
We keep billions of independent events like these



Disentangling what happened in EACH event takes ~minutes of CPU

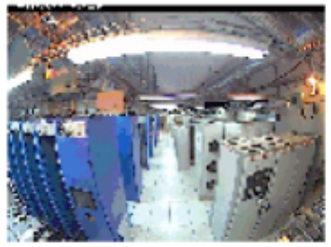
HT

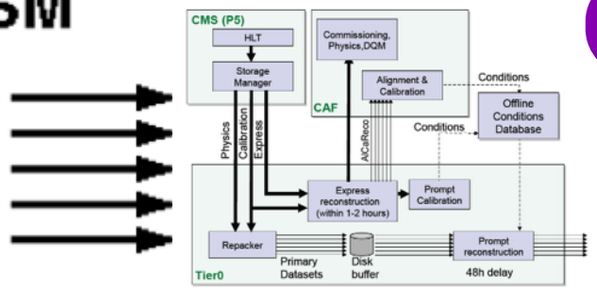
How We Handled This in the First LHC Run



CMS Detector

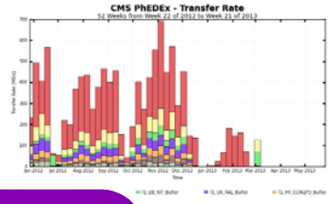
Trigger/DAQ/SM





CERN T0:
Repacking
Archive ("cold")
Prompt Reco
Release Validation
AOD/AlcaRECO/CAF

Data Transfer



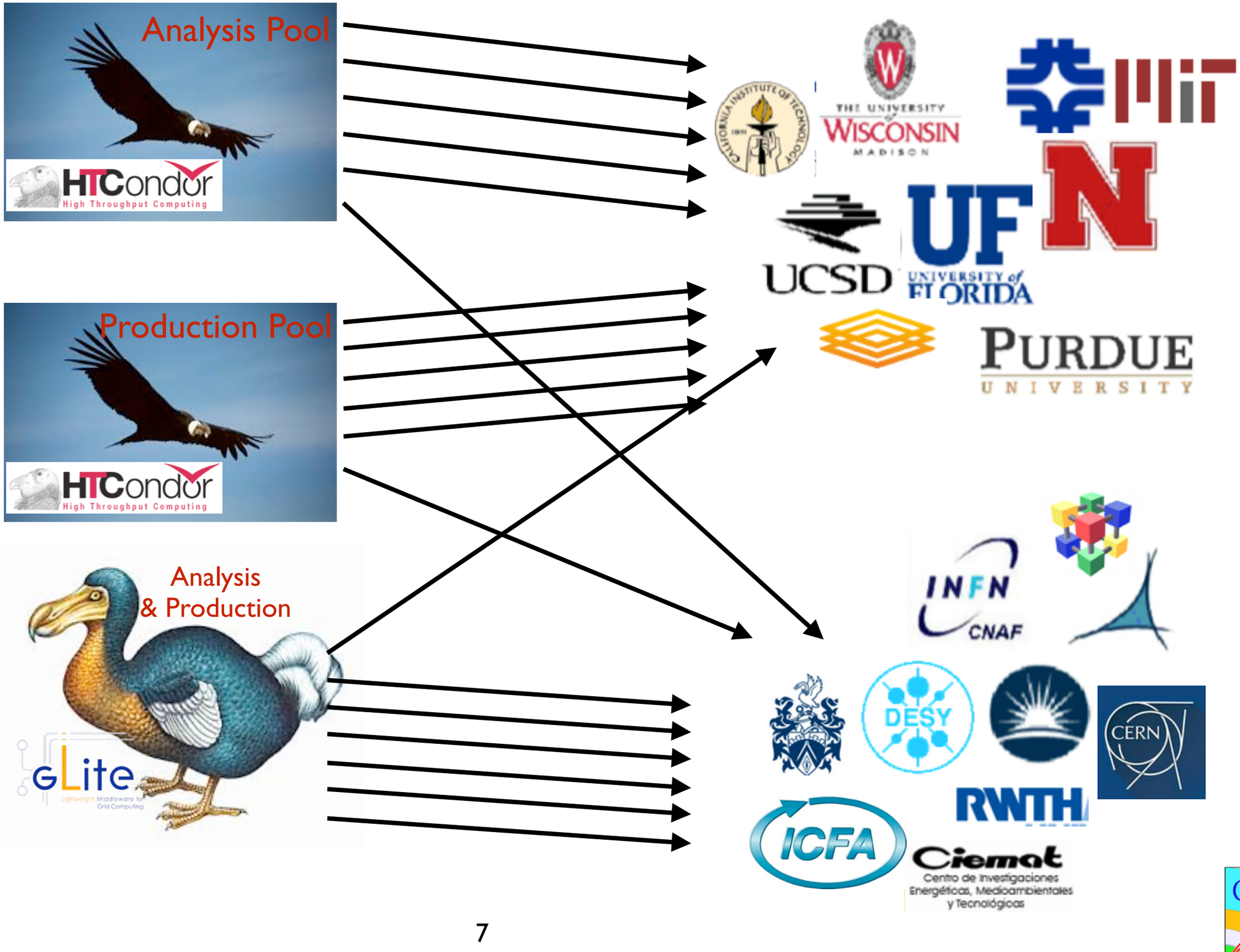
T1 Centers:
Archive ("hot")
Reconstruc
Skimming
Release Val

Data Transfer

T2 Centers: MC Production User Analysis

Strictly Tiered Infrastructure. Tasks Constrained To Class of Resource

And then subdivided depending on grid



LHC Run 1: Hard partitioned resources with multiple submission methods

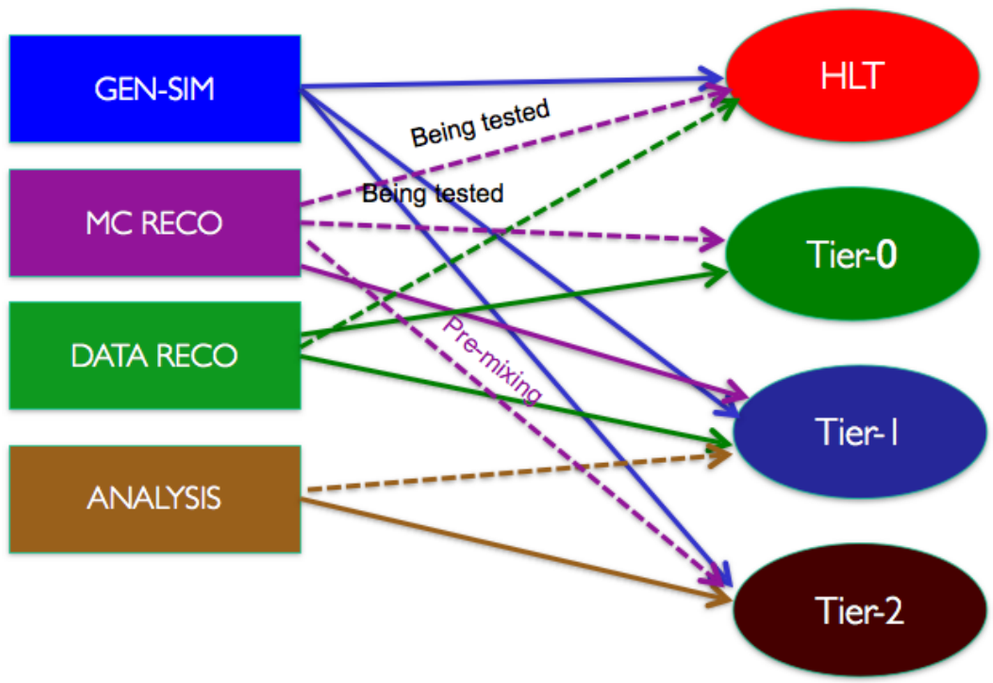
- **But in this upcoming run**
  - Beam energy is two times higher
  - We'll record two times the rate of data
  - The machine will collide many more protons together at the same time

➡ Need many more resources than we did for Run 1 O(100k's cores)
(E. Fajardo's talk tomorrow AM)

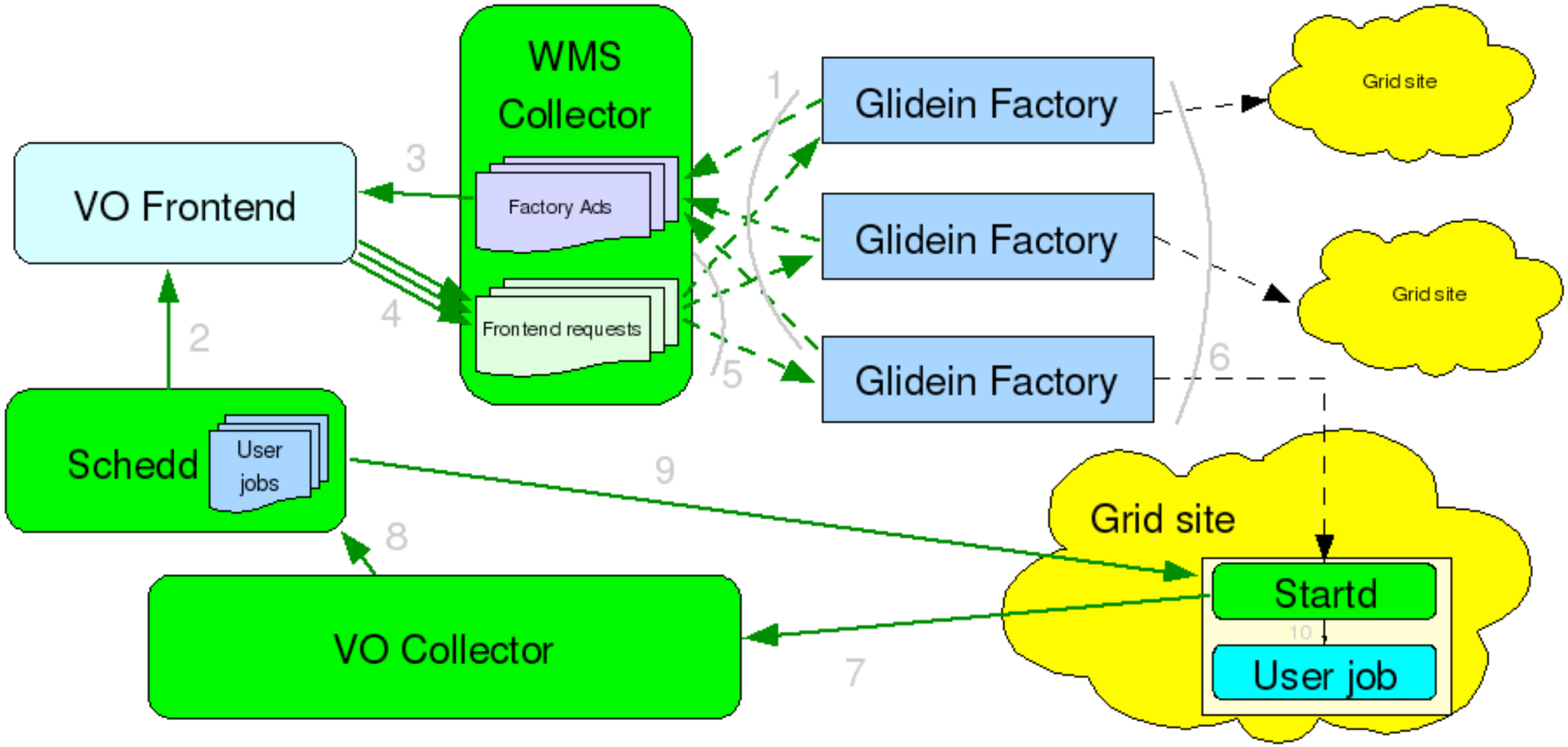
✓ Choose a submission method: HTCondor High Throughput Computing +glideinWMS

✓ Need to pool resources — be flexible where to run

✓ Need to be able to rebalance priorities globally



The Unifier — glideIn WMS



- Independent of the underlying batch system of a site, from the VO perspective glideinWMS constructs a uniform HTCondor pool — essential for making a global pool.

CMS Drivers & Implementation

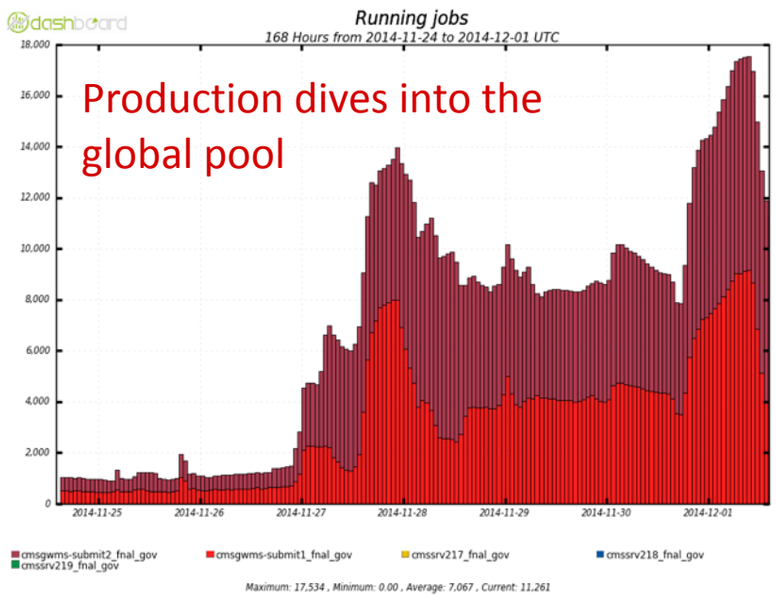
- The analysis and central production use cases rely on agents (ultimately daemons written in python) collecting, building and submitting jobs.
  - CRAB3 collects user jobs and handles job submission, retries, data stage out
  - WMAgent handles requests from physics groups for simulation or data reprocessing.
- Agents sit with the schedd's, all schedd's talk to the common global pool Frontend
- In the absence of other requirements, site whitelists, memory # core requirements, a global priority determines who runs first.
- Frontend calls up factories to send pilots to the requisite sites — pilots have a "pilot" or "production" role.
- Use gLExec to take on the central production or analysis user's credentials.

Overall Design Characteristics

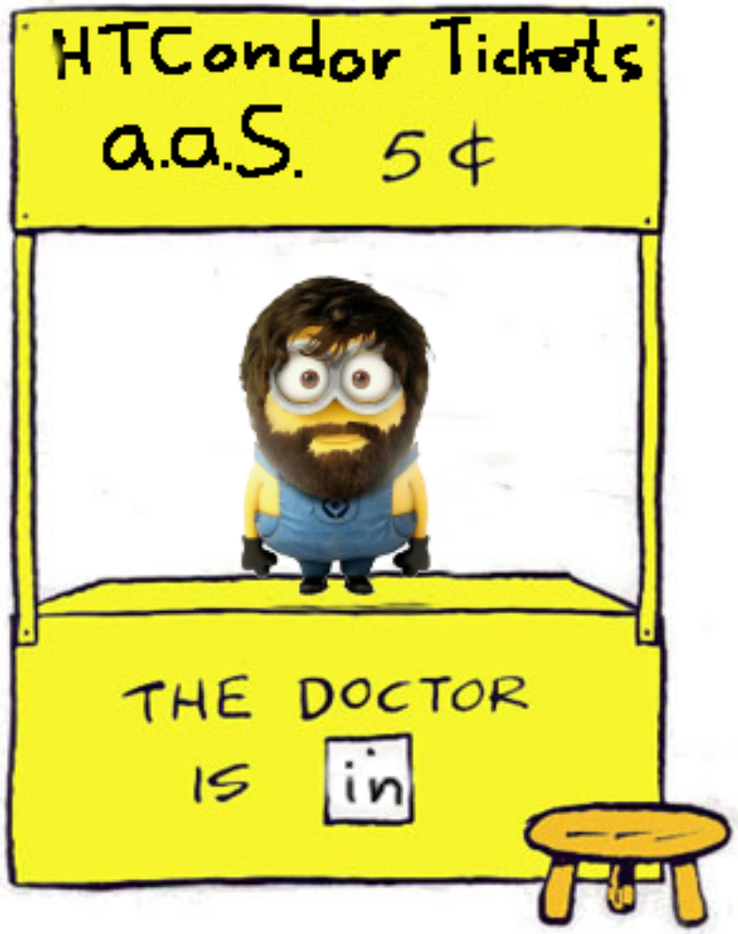
- We have agents/schedd's distributed at CERN, FNAL, UCSD and UNL.
  - About a dozen active at any given time
- With the help of OSG Factory Ops we use factories at CERN, FNAL, UCSD, GOC
- We've as much as possible tried to configure the glideinWMS components as HA between CERN and FNAL.
  - With the latest glideinWMS now all components, Frontend, Collectors, can run HA
  - Important when you don't have a 24 hour team at your disposal (FNAL does, used as a last resort) but need 24 hour availability.
- Worked hard to move from patched up custom built infrastructure of a year ago to all release RPM's, deployed via puppet.
  - Much easier to scale up when needed, replace failed nodes, make test instances !

Global Pool Timeline

~May 2014	Global Pool begins with analysis use case
June-July 2014:	Analysis pool scale testing in "CSA14" exercise
Aug. 2014:	Begin adding test production jobs to the mix
Sep 2014:	50k test production jobs reached in global pool
Nov 2014:	Production officially joins analysis in global pool
Jan 2015:	>100k analysis and production jobs reached
Mar 2015:	CMS Tier 0 begins flocking to global pool

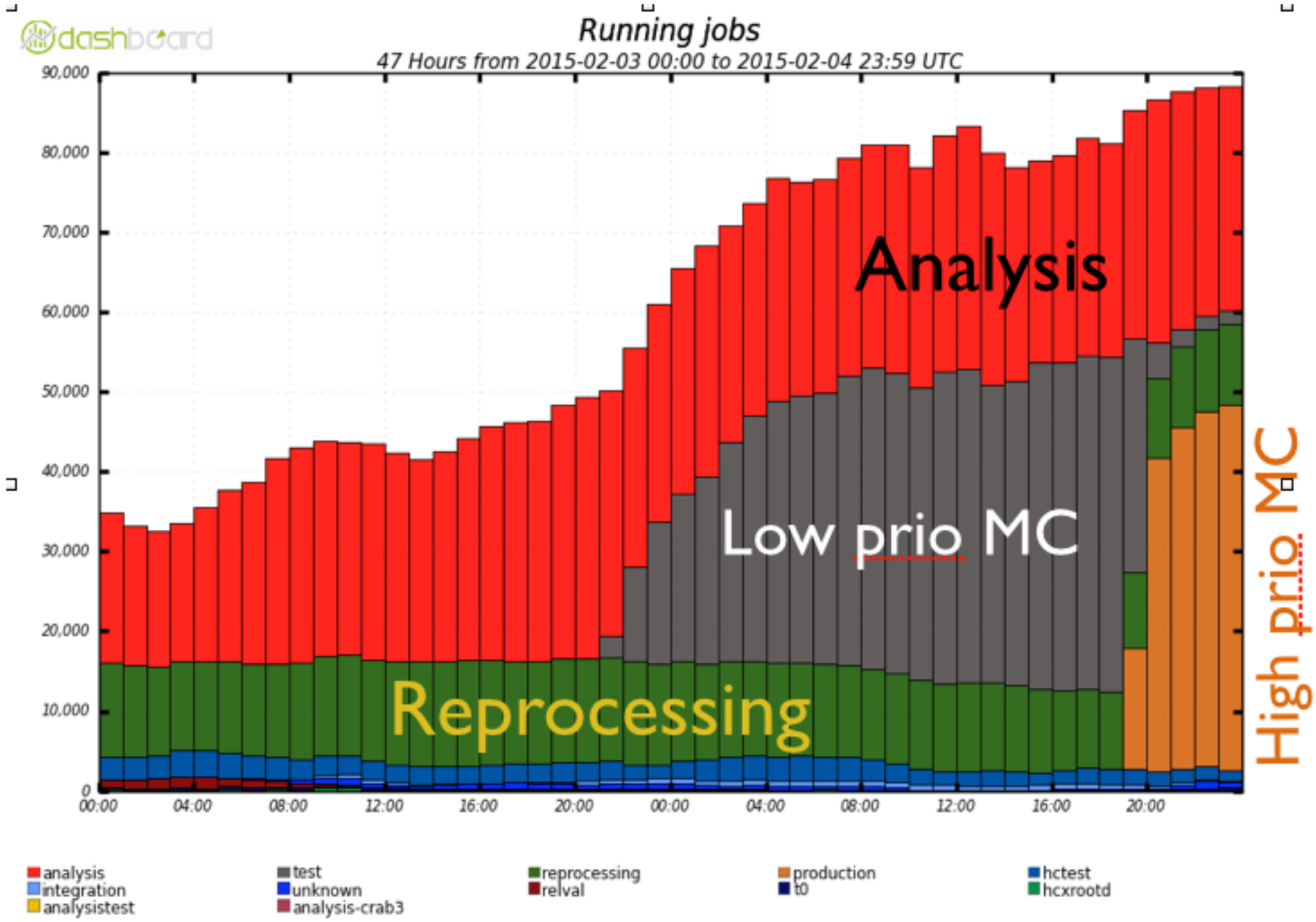


- A year ago we suffered from long negotiation cycles, schedd crashes, Frontends shedding jobs.
- Certainly thanks to close cooperation with the HTCondor and glideInWMS developers, the global pool now reliably handles ~100k jobs.



Global Prioritization

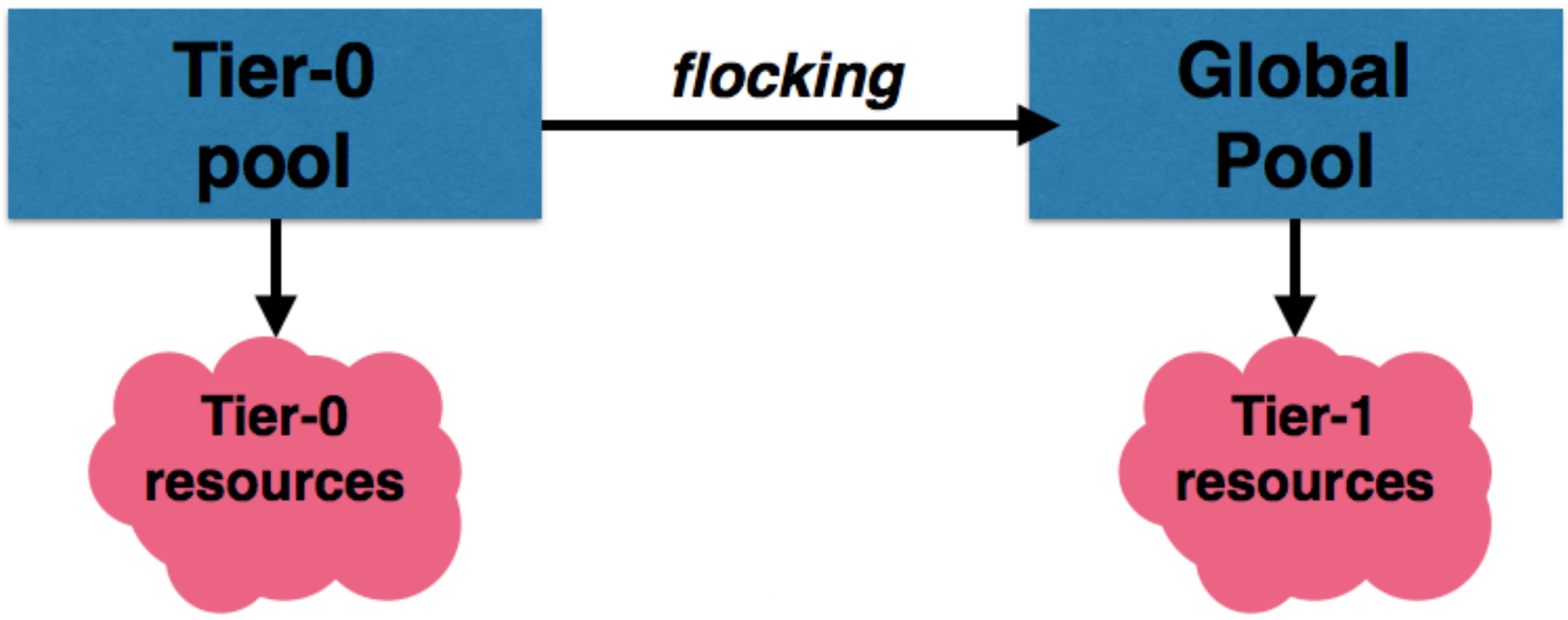
- The global priority now allows us to better control use of the resources depending on the need



Tier 0 and the global pool

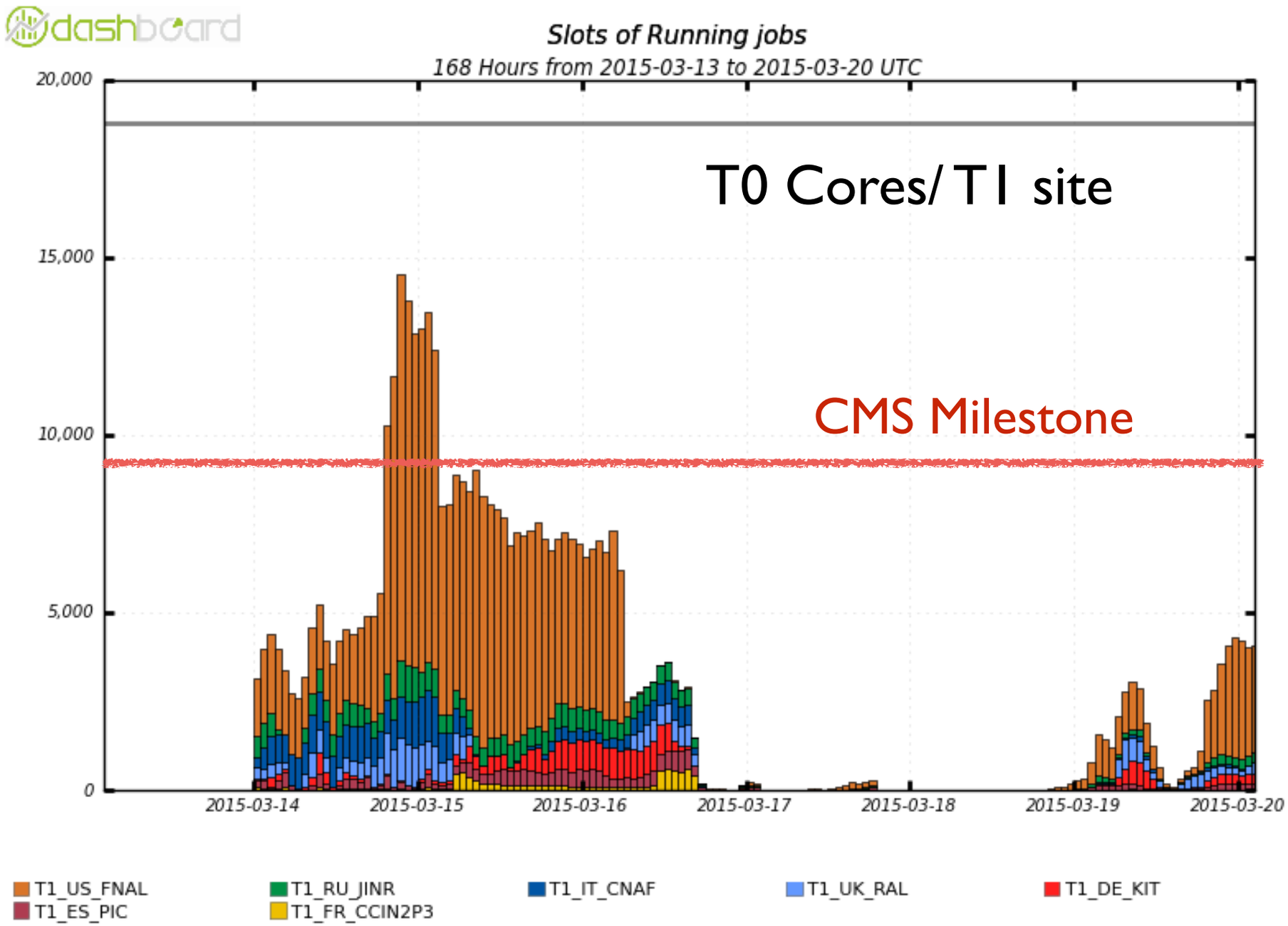
- In Run 1 the Tier 0, the first line of prompt processing after data leaves the detector ran in the local LSF queue at CERN.
- This time around, with the move to the Agile Infrastructure cloud CMS decided to build a dedicated condor pool for the Tier 0 at CERN.
  - The Tier 0 is tied to the data-taking stream, so reliability is key
  - Because of length of time to provision OpenStack AI resources, we carve off VM's running month long pilots.
  - There were concerns during machine down times, lower activity, etc. that having large numbers of long lived idle pilots would have adverse effects on the global pool
- But also given the high data rate * more complicated events we know the CERN resources will not be sufficient to run all the Tier 0 workflows.
  - Overflow processing to Tier 1's
  - I.e. flock to the global pool

Tier 0 pool to Global Pool Flocking



- Allows the Tier 0 to expand out to take advantage of the Tier 1 resources
- Inherently set a high priority for the flocked jobs, there is a ~few day requirement for T0 job completion.
- Input data already distributed to Tier 1 sites by the time the jobs needing it as input run.
  - Though can use xrootd as a fallback to read direct from CERN

Successful Flocking of T0 Jobs

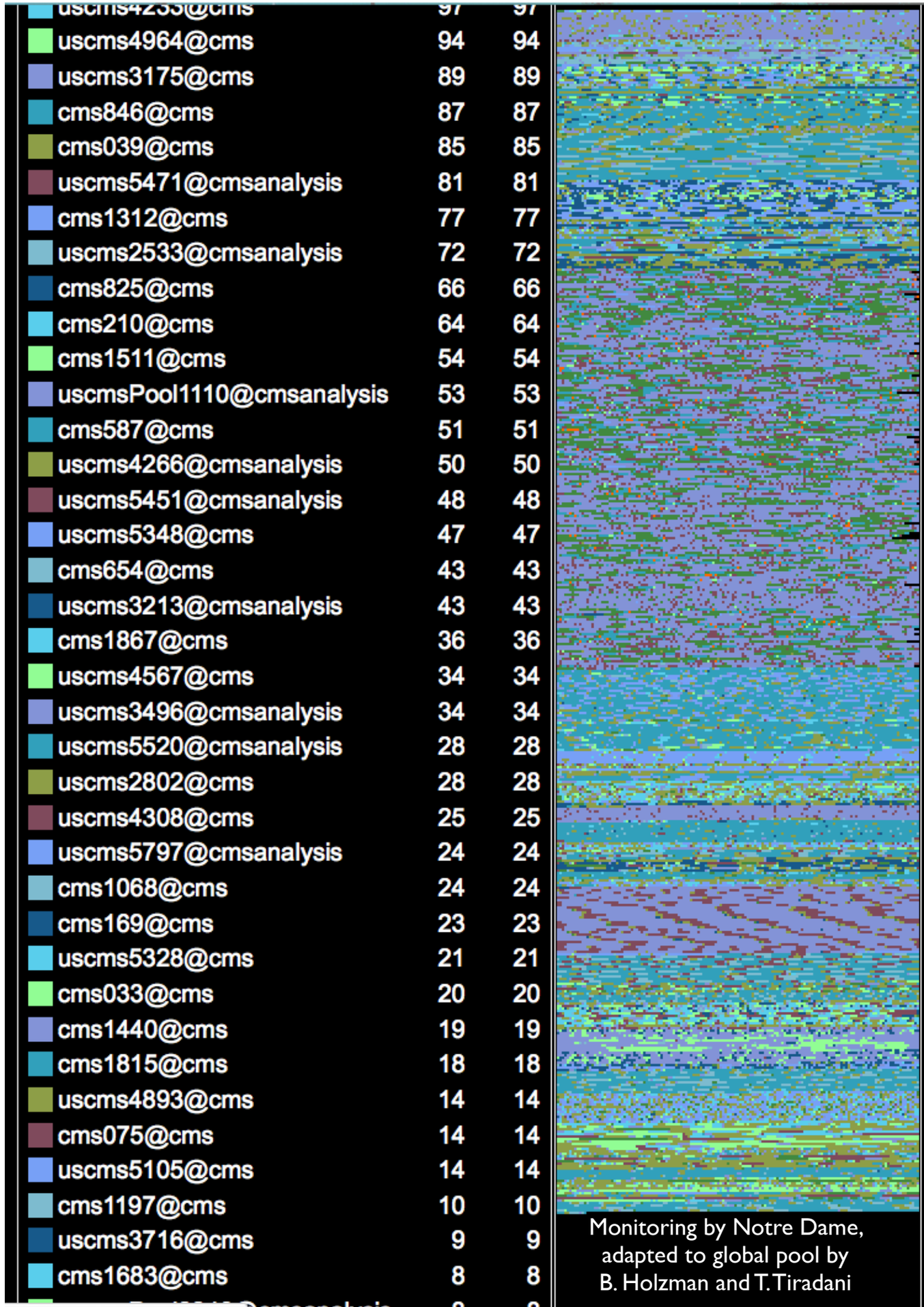


What Next?

- As the intensity of the LHC increases the focus will be gaining access to and utilizing additional resources.
  - Opportunistic use of spare cycles in OSG
  - Making use of allocations at HPC centers — accessing via Bosco and Parrot
  - CMS High Level Trigger farm between LHC fills
  - Gambling on getting cheap resources on commercial clouds
  - All of this of course you saw in Tony T's talk this morning… Sanjay's talk just a bit ago…
- Providing easy and appropriate access to local resources, "my campus cluster"

A picture is worth 100k jobs…

- Fuzzy thing at the left is what the global pool really looks like!
  http://condor.cse.nd.edu/condor_matrix.cgi
- In time for the new physics run CMS has converted its submission infrastructure into a single coherent global HTCondor/glideinWMS pool.
- It will allow us to be more flexible, use resources more efficiently, and be better able to exploit the science of LHC Run 2!
- As we move forward, as the machine intensity increases, so will the need for more and more varied resources
- HTCondor & glideinWMS has more than met the challenge so far! We look forward to continue working with the HTCondor & glideInWMS teams to meet the challenges to come!



Monitoring by Notre Dame, adapted to global pool by B. Holzman and T. Tiradani

Acknowledgements

- James Letts (UCSD) co CMS Submission Infrastructure lead
  Brian Bockelman (UNL) as Bockelman-at-Large
- FNAL and CERN operations teams —K. Larson, T. Tiradani, A. Malta, F. Khan, A. McCrea, M. Mascheroni, B. Holzman, M. Saiz-Santos, S. Belaforte, D. Hufnagel, V. Verguilov, J. Silva, J. Barcas
- Jeff Dost (UCSD) & the OSG Factory Ops Team
- HTCondor development team
- glideInWMS development team