CorralWMS: Integrating glideinWMS and Corral

A collaboration between Fermilab, UCSD, and USC ISI

Krista Larson
Fermi National Laboratory
klarson1@fnal.gov

Mats Rynge
USC Information Sciences Institute
rynge@isi.edu

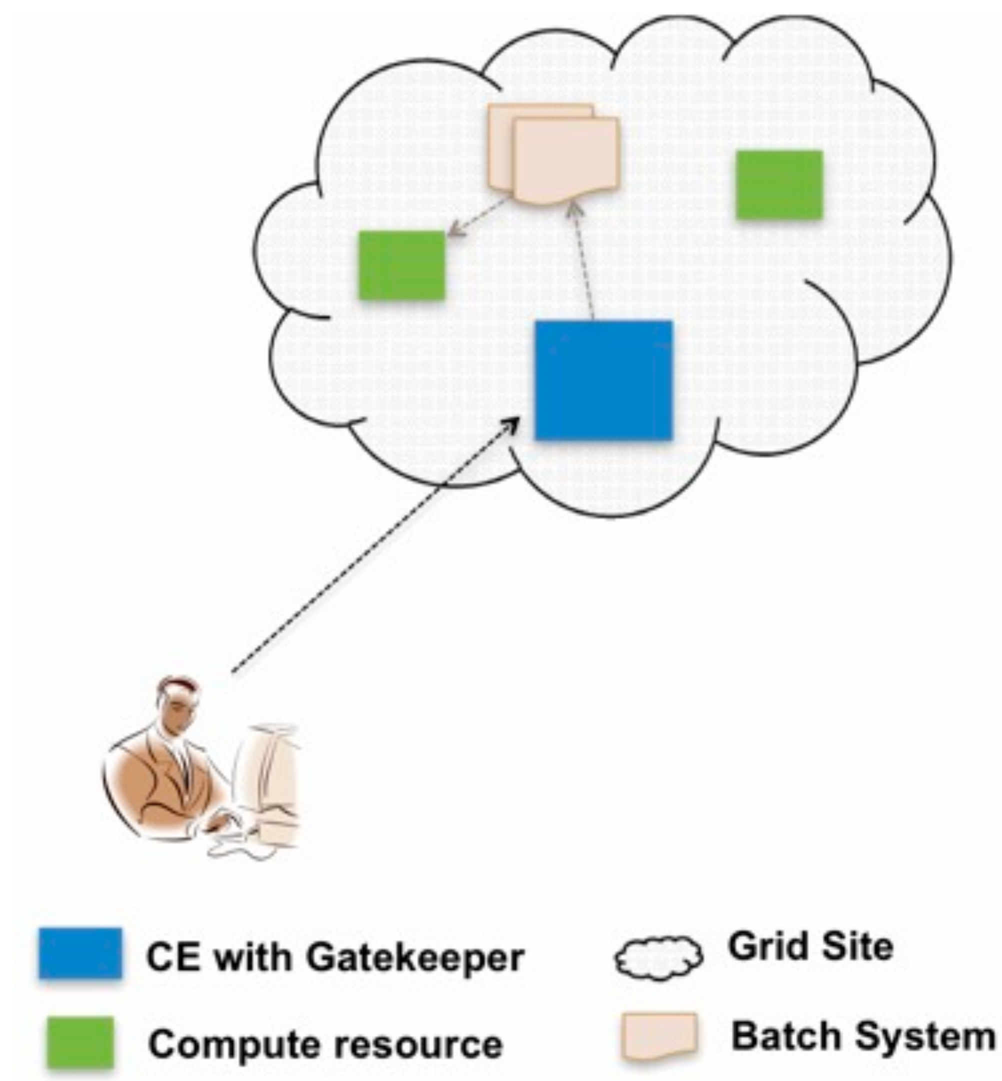
Condor Week 2010

Overview

- Grid computing
- Pilot based workload management systems
- glideinWMS
  - On-demand glidein provisioner used by CMS (and others)
- Corral
  - Static glidein provisioner for Pegasus workflows
- CorralWMS

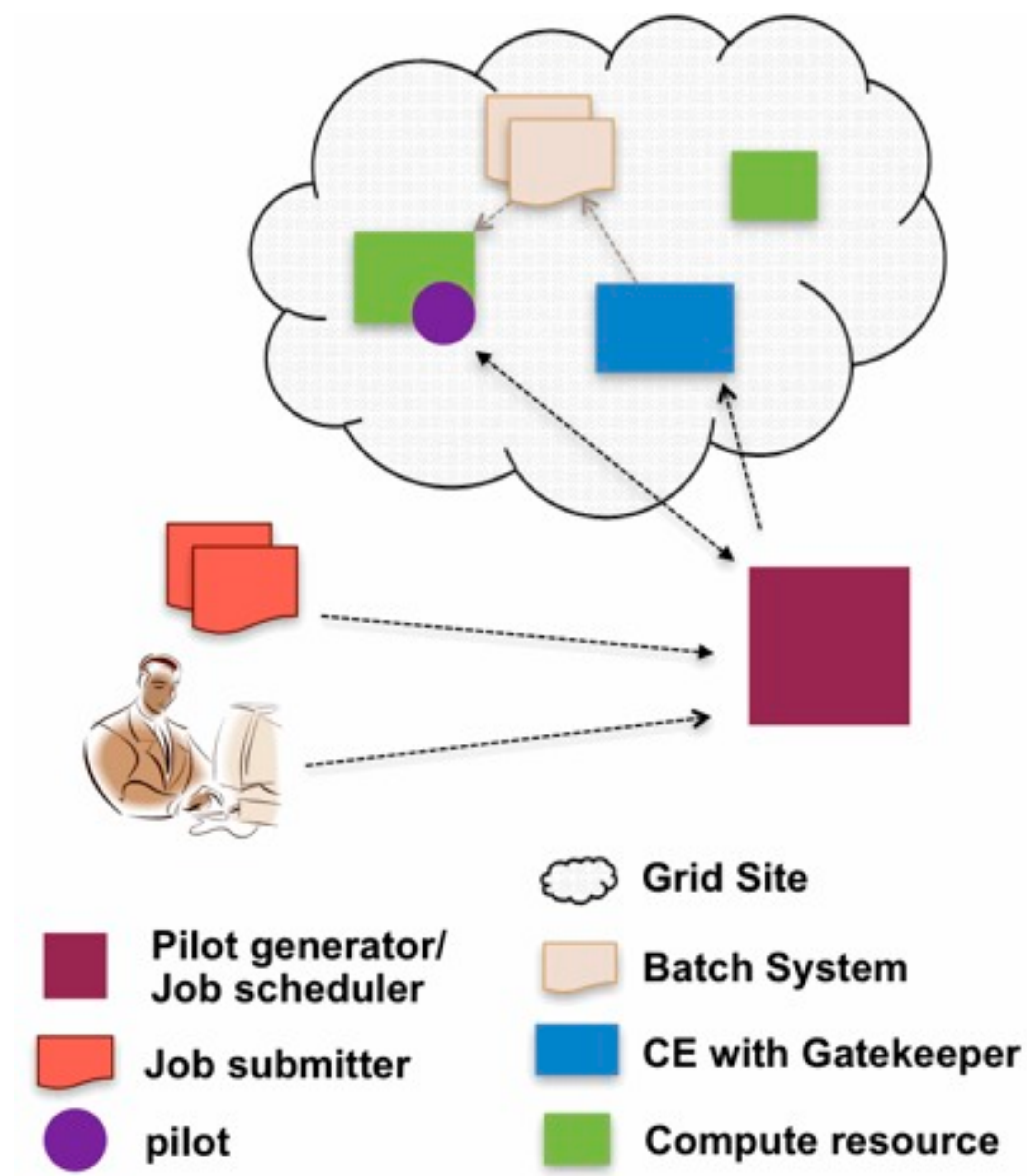
Grid Computing

- Combines distributed computing resources from multiple administrative domains
- User has access to a large pool of resources, but
  - Middleware has problems managing jobs
  - Monitoring jobs is complicated
  - Heterogeneous grid resources can cause issues
  - Queueing and scheduling delays
  - Software overheads and scheduling policies



Pilot Based Workload Management Systems

- Pilot generator submits pilots to the grid sites
- Pilots start running on the compute resources
  - Pilot can run several checks
  - Hides some diversity of grid resources
  - Overlays personal cluster on top of the grid
- Pilots fetch user jobs from a scheduler and execute
- Issues with scalability
  - Central queue can be resource intensive
  - Security handshake can be expensive



glideinWMS

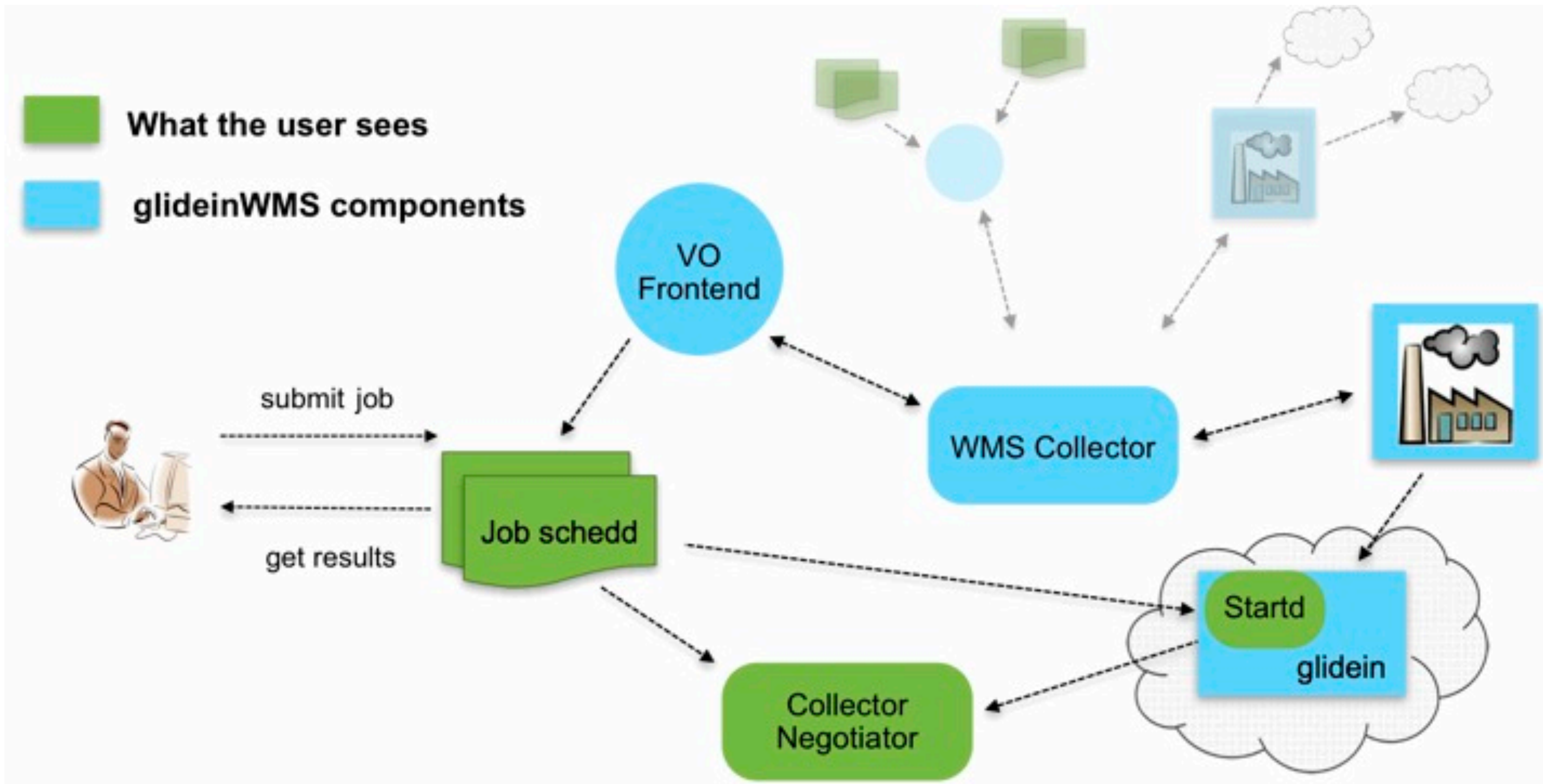
- glideinWMS a thin layer on top of Condor
- Uses glideins (i.e. pilot jobs)
  - a glidein is a Condor Startd submitted as a grid job
- All network traffic authenticated and integrity checked
- Pseudo-interactive job monitoring is included
- Addresses scalability considerations
  - Multiple user queues can spread the load
  - Increase memory of the machine hosting the schedd service
  - Multiple slave collectors can reduce communication issues

glideinWMS

Scalability achieved with 1 master collector and 70 slave collectors (on a single machine), machine with 16GB memory for hosting the schedd service:

Criteria	Design goal	Achieved so far
Total number of user jobs in the queue at any given time	100k	200k
Number of glideins in the system at any given time	10k	~26k
Number of running jobs per schedd at any given time	10k	~23k
Grid sites handled	~100	~100

glideinWMS



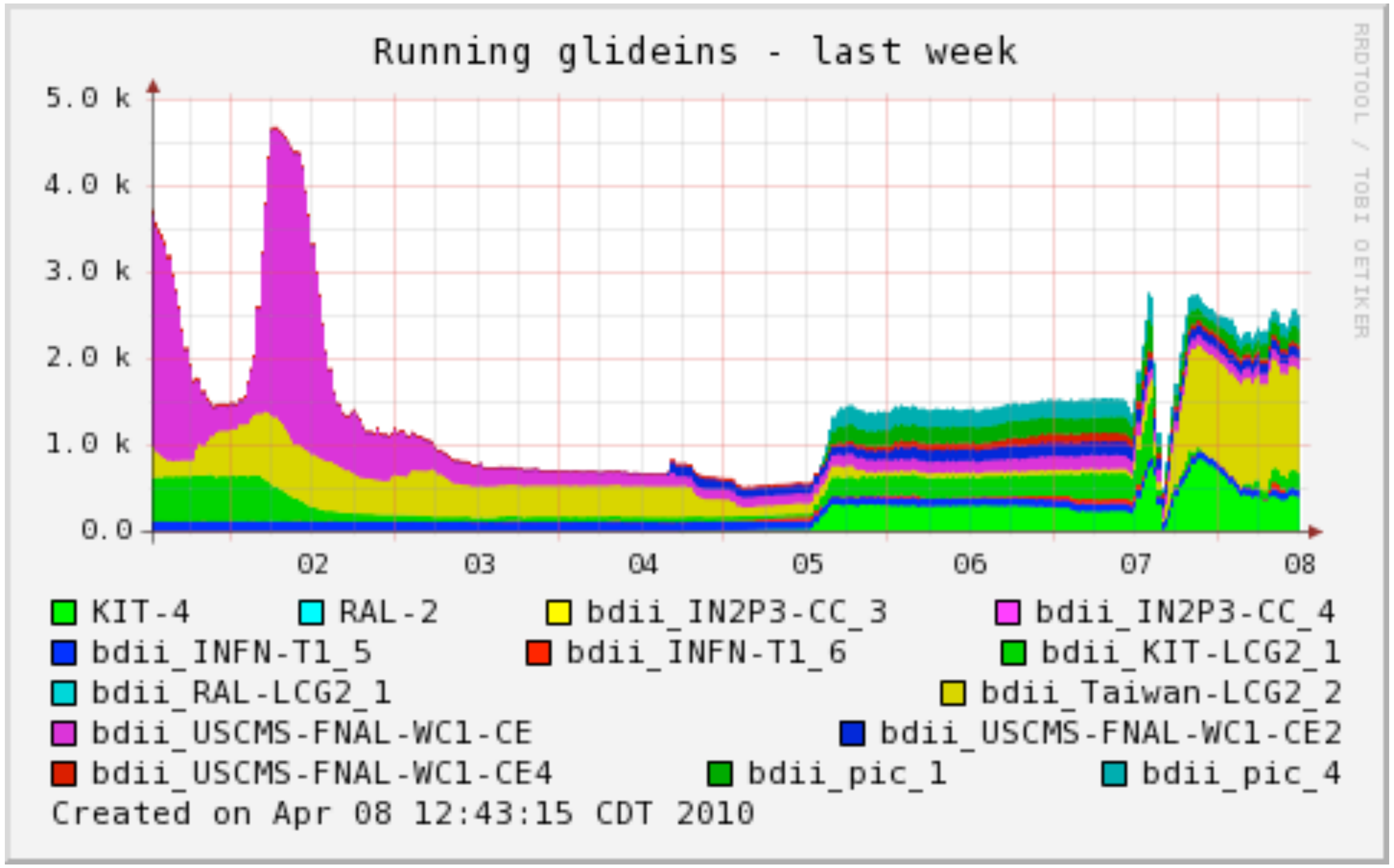
- Glidein Factories know about grid sites, how to submit glideins
- VO Frontends know about job details, number and kind of glideins needed
- Factories and VO Frontends communicate through common (Condor) WMS Collector

glideinWMS:
at Fermilab

- Gains for CMS through glideinWMS:
  - Jobs are not submitted to bad resources because pilots check and preconfigure environment
  - Eliminates steep turn on curve for allocating resources since pool already contains preconfigured slots
  - Workflows can be accommodated using a scheduler on the user queue
    - ➡ Ability to prioritize jobs for the local queue
  - Interactive debugging
  - Light on CEs because user jobs come directly from the submitter instead of CE gatekeeper

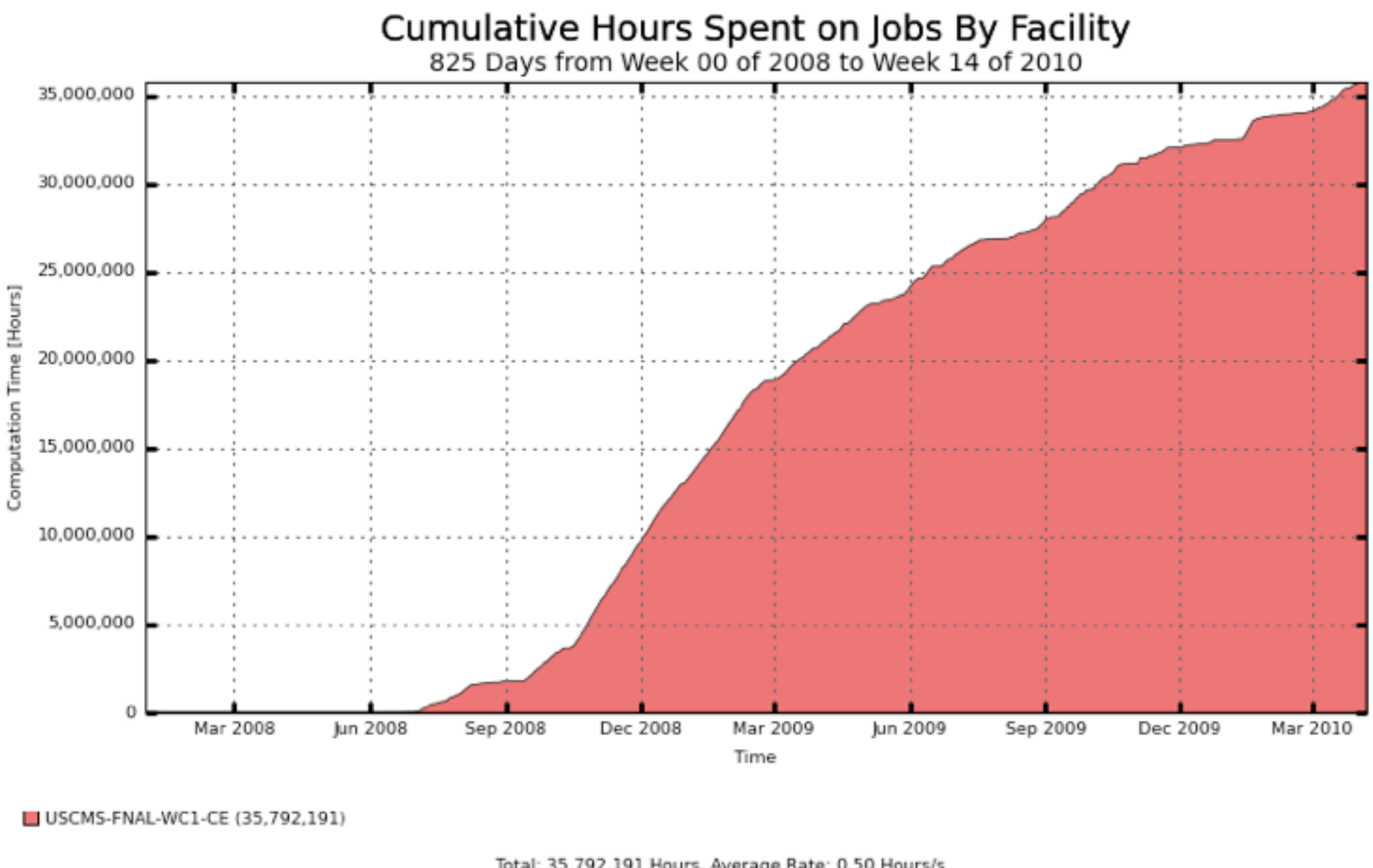
glideinWMS: at Fermilab

- Tier 1 processing center for data collected at the LHC CMS experiment
  - skims to reduce the data size
  - data reconstruction



glideinWMS: at Fermilab

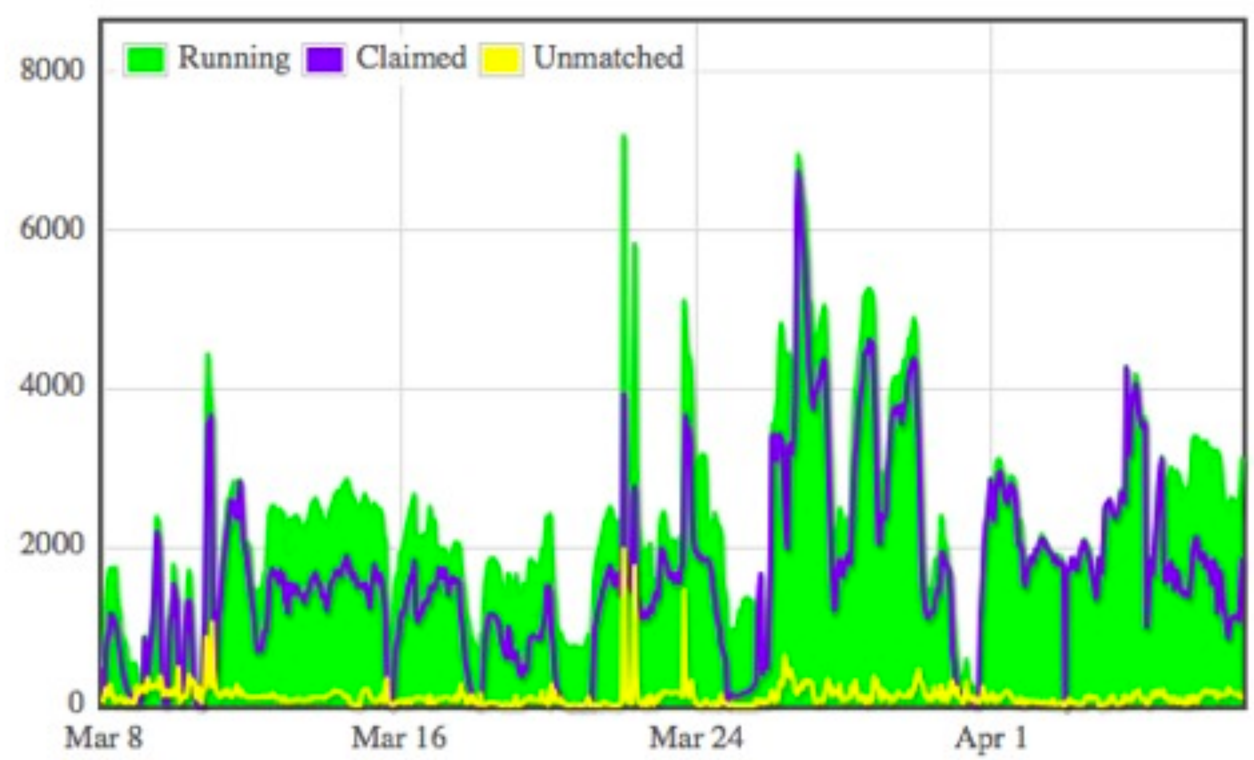
- Currently running glideinWMS v1.6
- In production since July 2008



glideinWMS: at UCSD

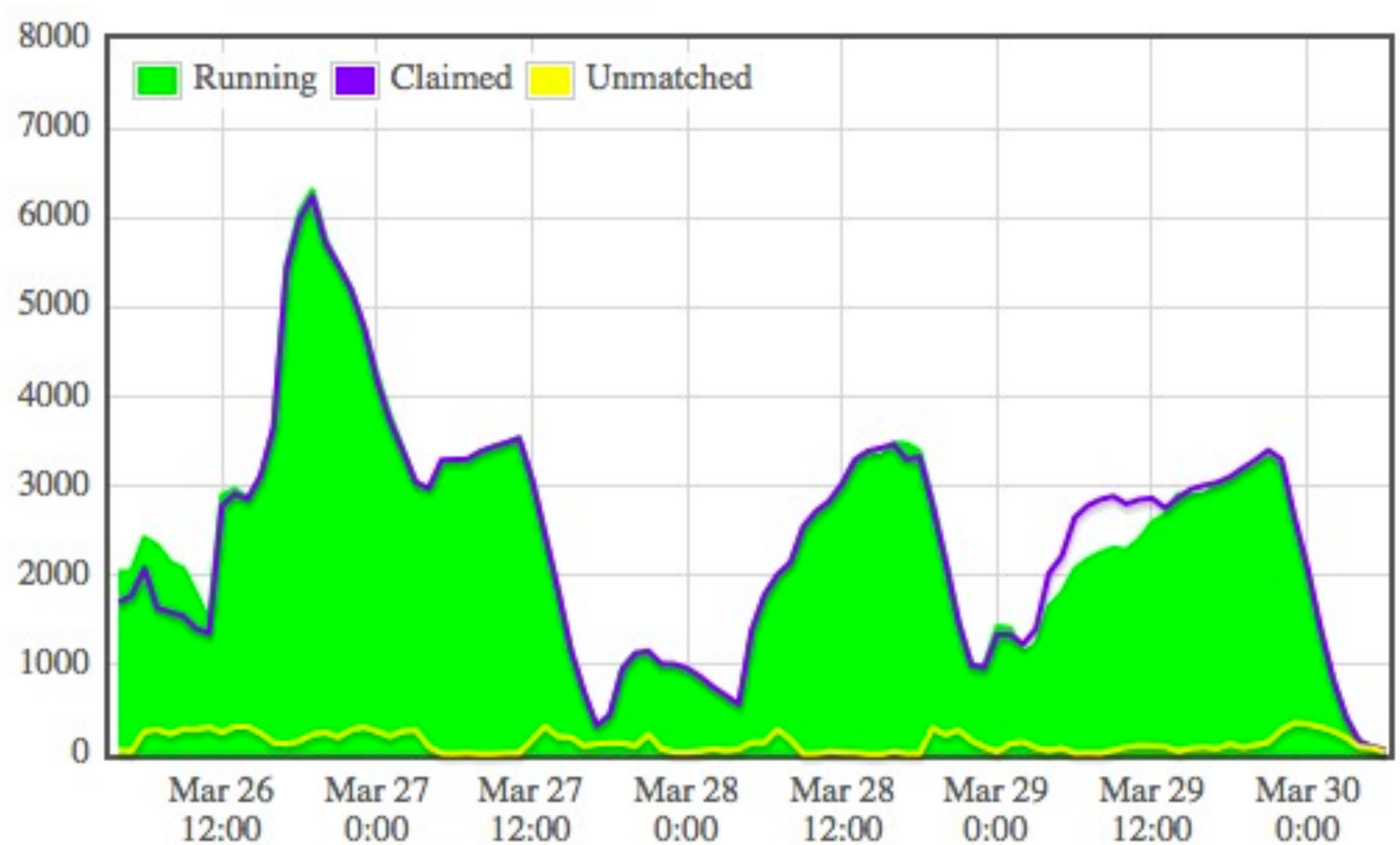
- Running glideinWMS v2.4
- Sends glideins to grid sites in both the Open Science Grid (OSG) and Enabling Grids for E-sciencE (EGEE)
- Currently has 3 clients: CMS, GLOW/IceCube, and HCC
- Each VO Frontend is running its own collector, submitter, and frontend

Glidein Factory Status



glideinWMS: at UCSD

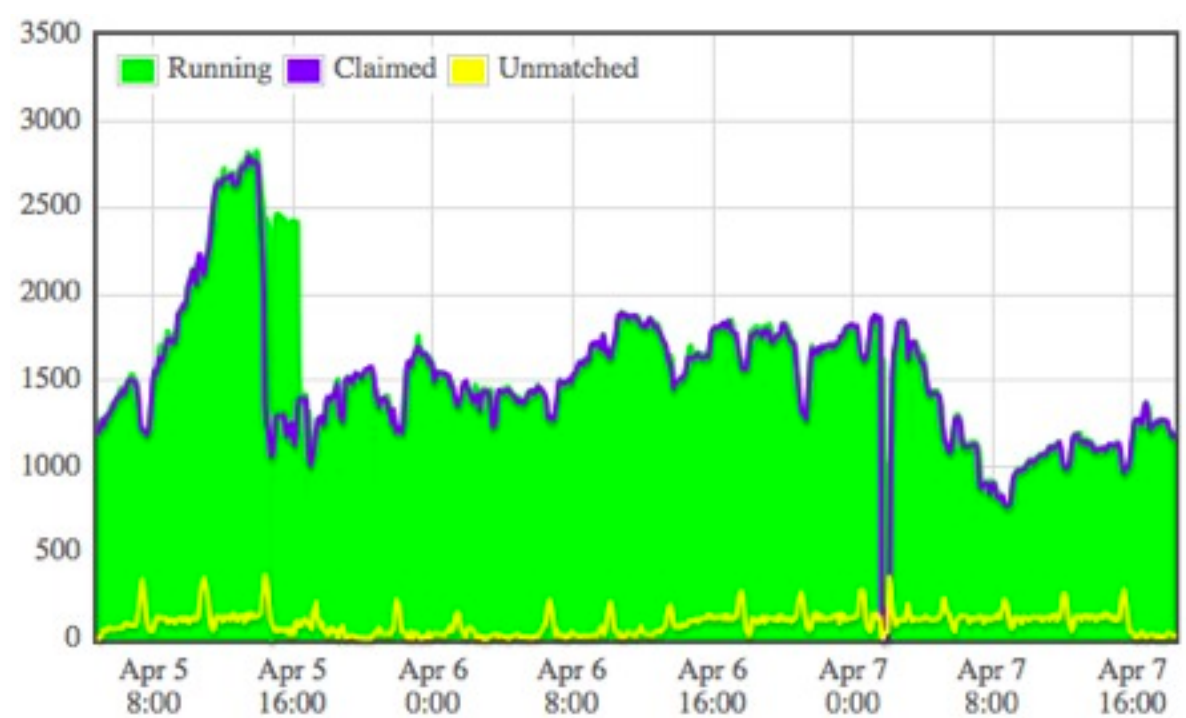
UCSD is running the VO Frontend for user analysis in the LHC CMS experiment:



glideinWMS: at Nebraska

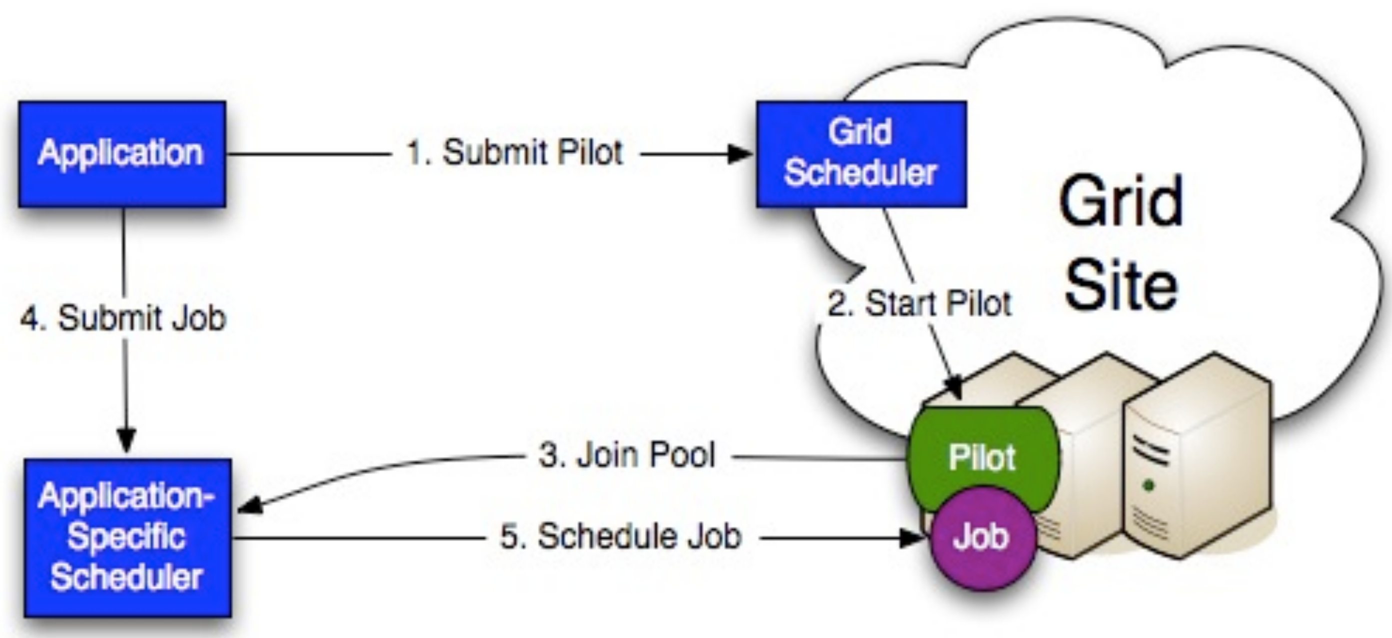
- Running VO Frontend, Submitter, and Collector that uses the Factory at UCSD
- Used for combinatorics, biology, and bioinformatics applications
- Average 25,000 hours / day running jobs on glideinWMS
- 2.1 million CPU hours since beginning of 2010
- Currently limited by the memory of submit machine
  - Working on flocking local Condor Schedds to glideinWMS

VO Frontend Status



Corral

- Resource provisioning system
  - Allocate resources explicitly rather than implicitly
  - Pay to allocate resources once and reuse them
  - Effectively minimizes grid overheads
  - Requires resource specification



Pegasus:
Planning for Execution in Grids

- Abstract Workflows - Pegasus input workflow description
  - workflow "high-level language"
  - only identifies the computations that a user wants to do
  - devoid of resource descriptions
  - devoid of data locations
- Pegasus
  - a workflow "compiler"
  - target language - DAGMan's DAG and Condor submit files
  - transforms the workflow for performance and reliability
  - automatically locates physical locations for both workflow components and data
  - finds appropriate resources to execute the components
  - provides runtime provenance
- DAGMan
  - a workflow executor
  - scalable and reliable execution of an executable workflow

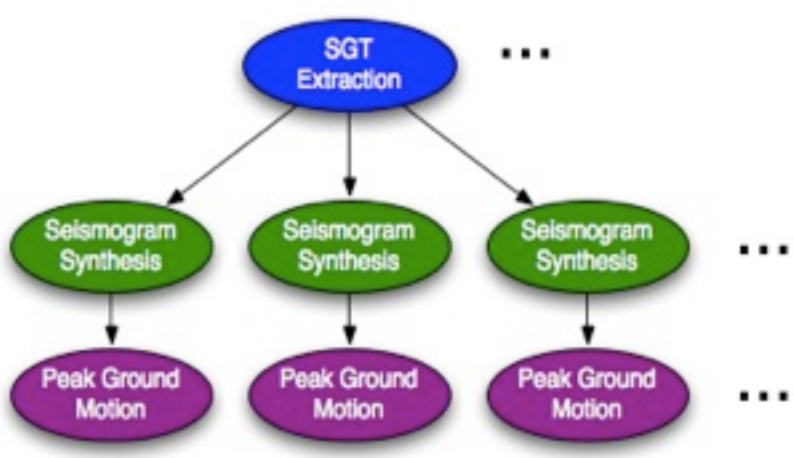
Corral Features

- Auto-configuration
  - Detect architecture, OS, glibc => Condor package
  - Determine public IP (if any)
  - Generates Condor configuration file
- Large requests
  - 1 glidein job = N slots
- Multiple interfaces
  - Command-line, REST, Java API
- Automatic resubmission
  - Indefinitely, N times, until date/time
- Notifications
  - Asynchronous API for receiving glidein status

Southern California Earthquake Center

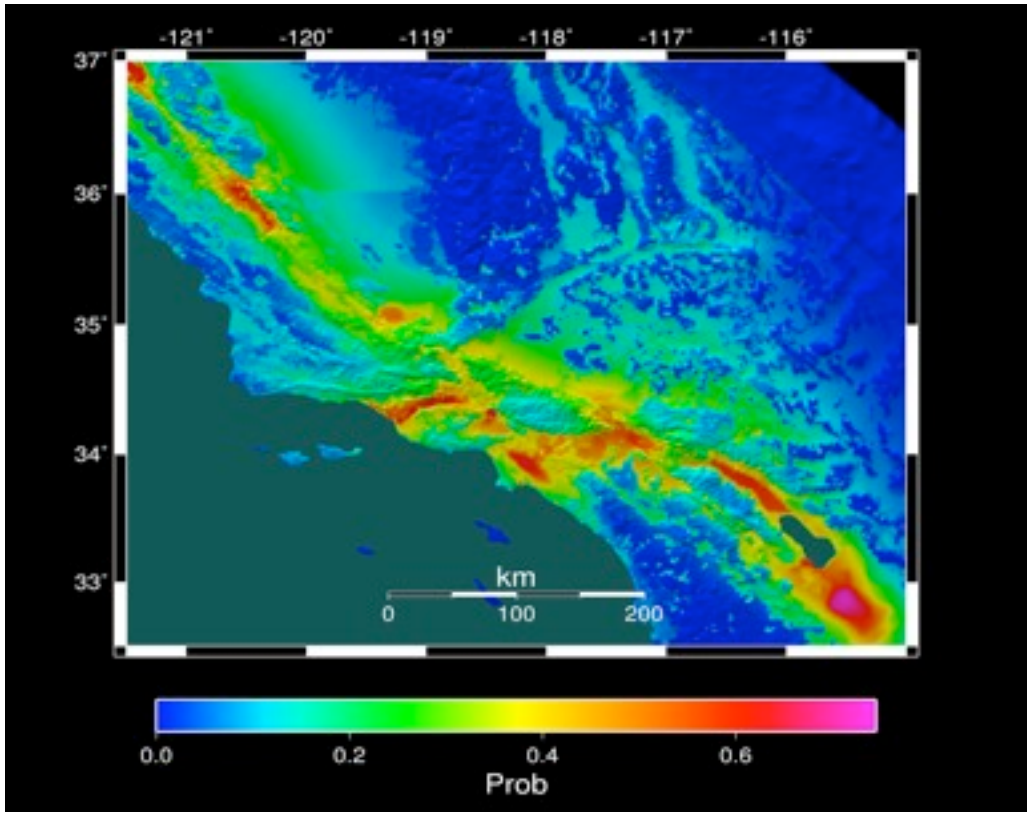
an NSF+USGS center

- Probabilistic seismic hazard analysis workflow
  - How hard will the ground shake in the future?
- Uses Pegasus and DAGMan for workflow management



Transformation	Tasks
SGT Extraction	7,000
Seismogram Synthesis	420,000
Peak Ground Motion	420,000
Tasks:	847,000

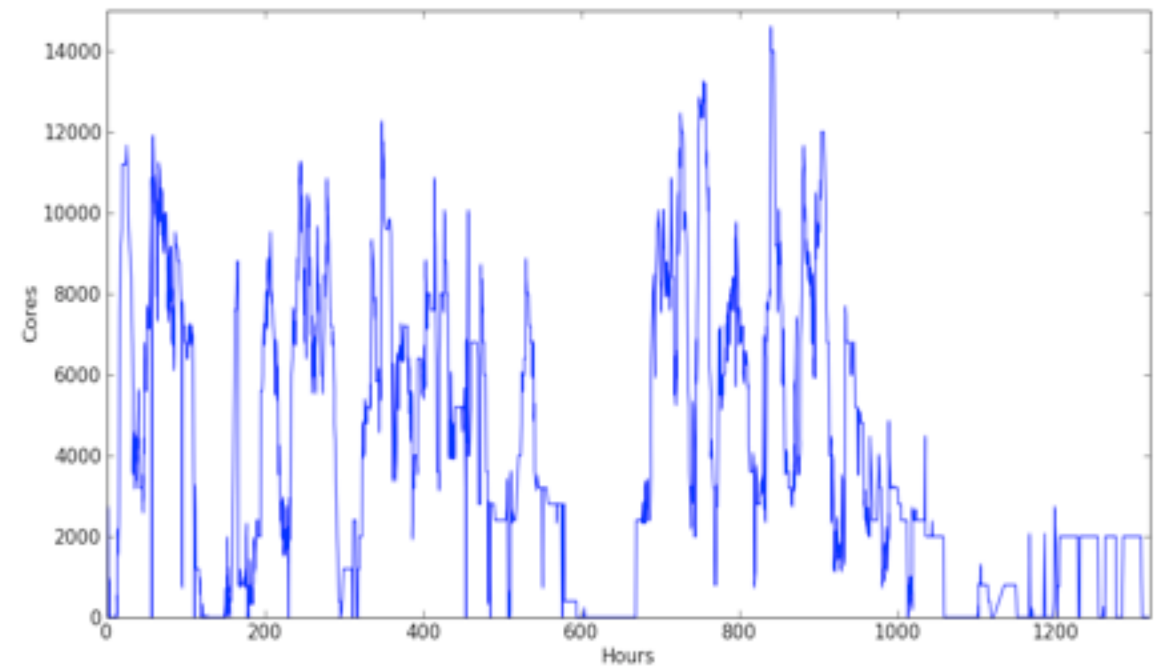
223 sites -> 189 million tasks

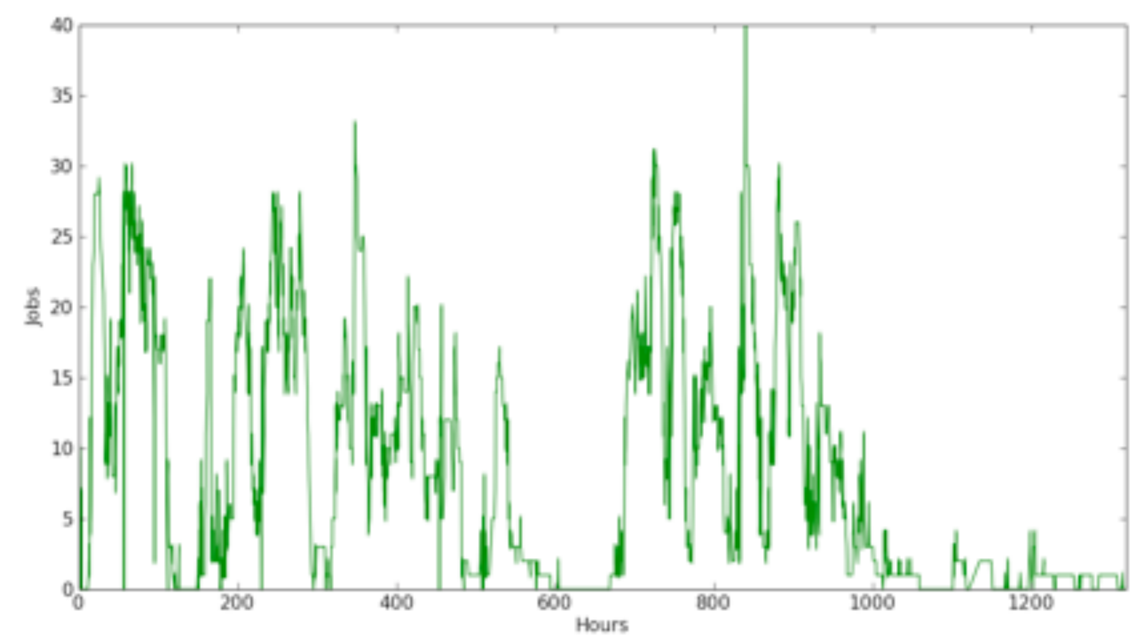


Slide courtesy: Scott Callaghan, SCEC

CyberShake 2009

- Run from 4/16/09 - 6/10/09
- TACC's Ranger
- 223 sites
  - Curve produced every 5.4 hrs
- 1207 wallclock hrs
  - 4,420 cores on average
  - 14,540 peak (23% of Ranger)
- 189 million tasks
  - 43 tasks/sec
  - 3.8 million Condor jobs
    - 289 failures
  - 3952 Ranger queue jobs

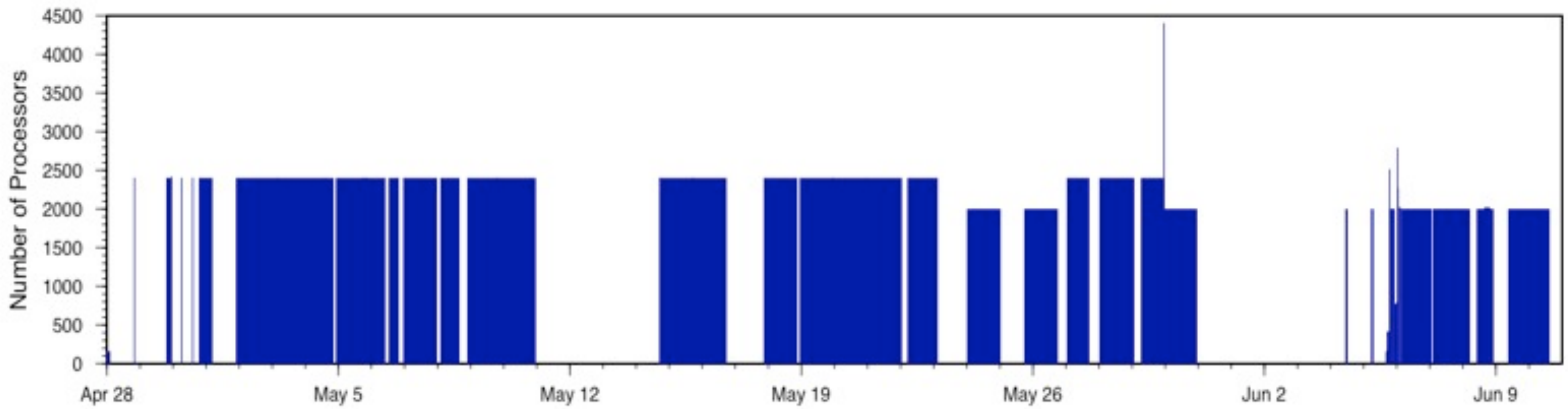


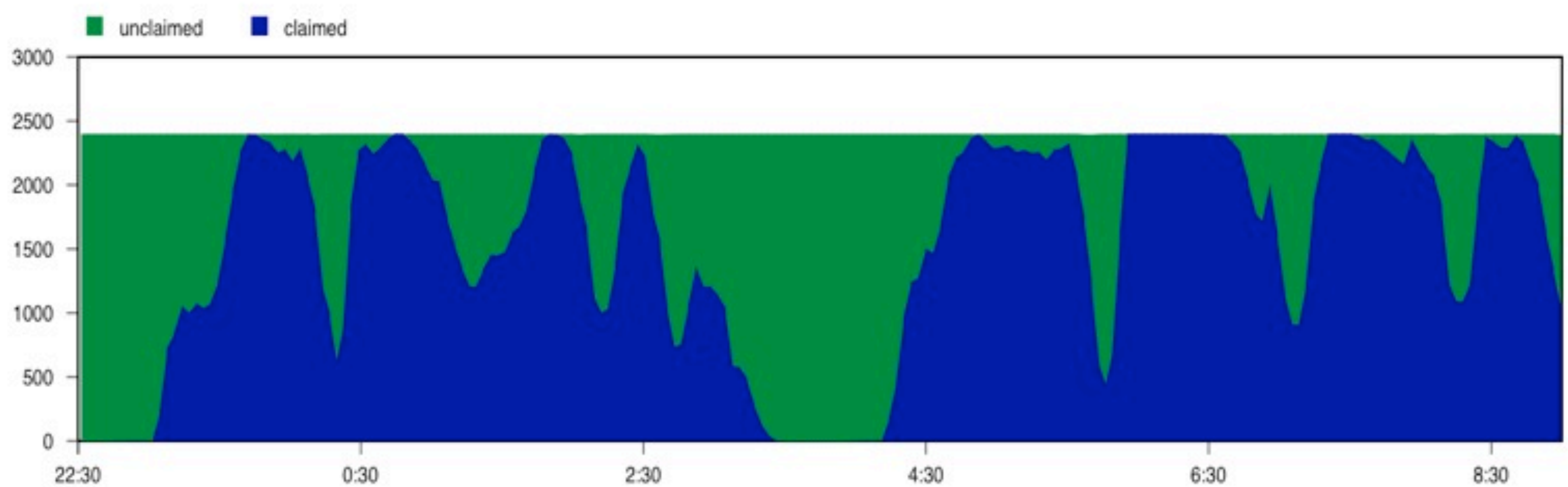


Slide courtesy: Scott Callaghan, SCEC

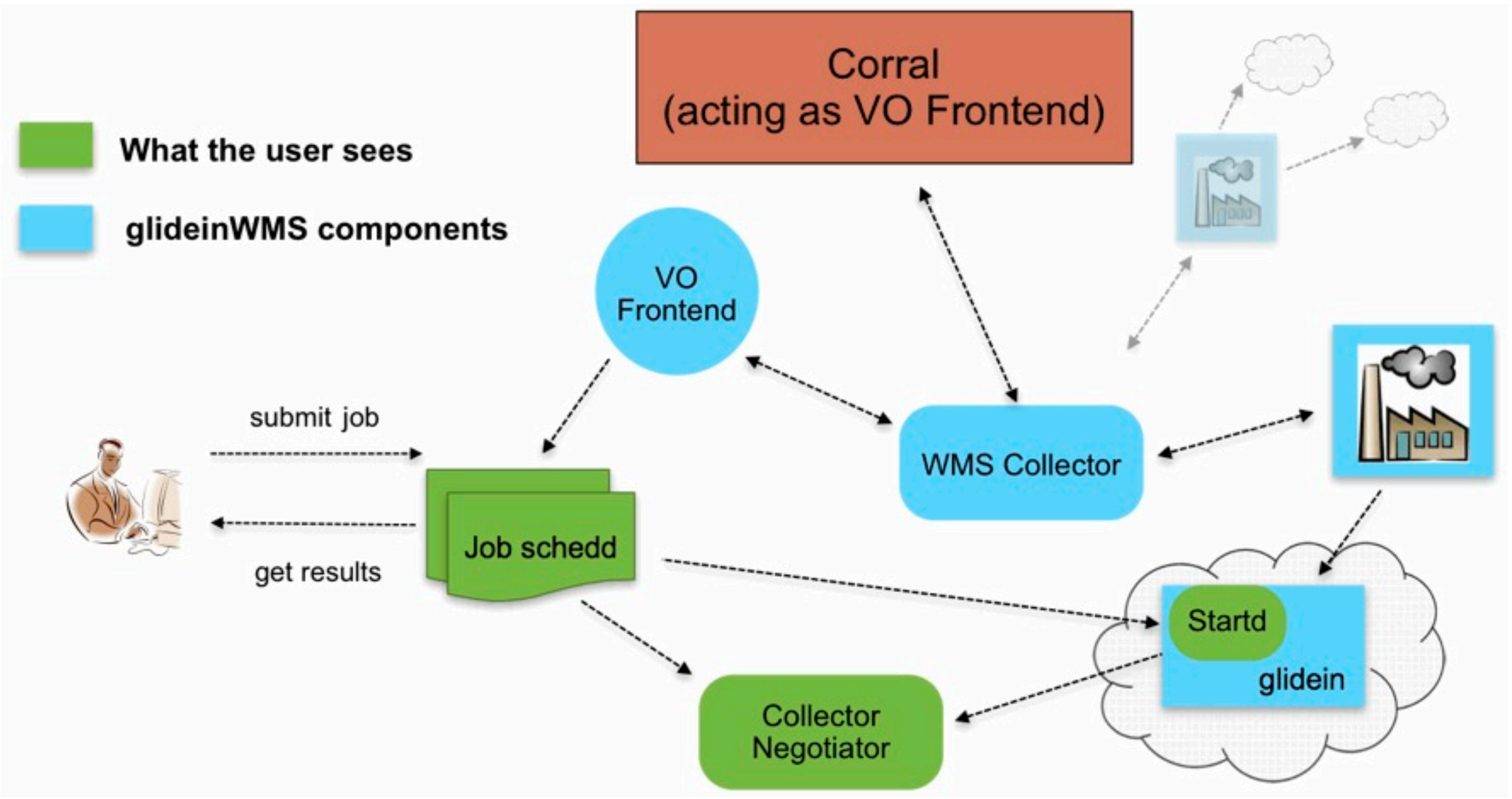
CyberShake 2009 Corral

Requests: 82
CPUs: Up to 2400 at a time
CPU Hours: 1.19M
Condor jobs: 4.17M





CorralWMS



What the user sees
glideinWMS components
Corral
(acting as VO Frontend)
VO
Frontend
submit job
get results
Job schedd
WMS Collector
Startd
glidein
Collector
Negotiator

CorralWMS Goals

- Domain application workflows as driving force
- Support a broad set of workload execution environments
  - Local, major national CI providers (Open Science Grid / TeraGrid), commercial and science clouds
- Retain identity of both systems to maintain backwards compatibility for existing users

More Information

- This material is based upon work supported by the National Science Foundation under Grant No. 0943725

- glideinWMS:

  http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/

- Pegasus/Corral:

  http://pegasus.isi.edu/