GlideinWMS and CMS Data Analysis

JAMES LETTS
UCSD

Introduction

- I am a physicist (not a computer scientist) who has been involved in computing for HEP analysis for the past 20 years (OPAL, CMS)
  - CMS computing management for Physics Support, previously Analysis Operations.
  - Admin for CMS glidein Front End at UCSD, and our local CMS Tier-2 site.
- High-level overview of how CMS does analysis, and how it interfaces with glideinWMS now and in the future.

How CMS Does Analysis

- CRAB - CMS Remote Analysis Builder
- CRAB2 supports many different modes of operation:
  - "Stand-alone" and CRAB Server
  - Different submission mechanisms: most common are gLite (direct submission) and glidein

How CMS Does Analysis

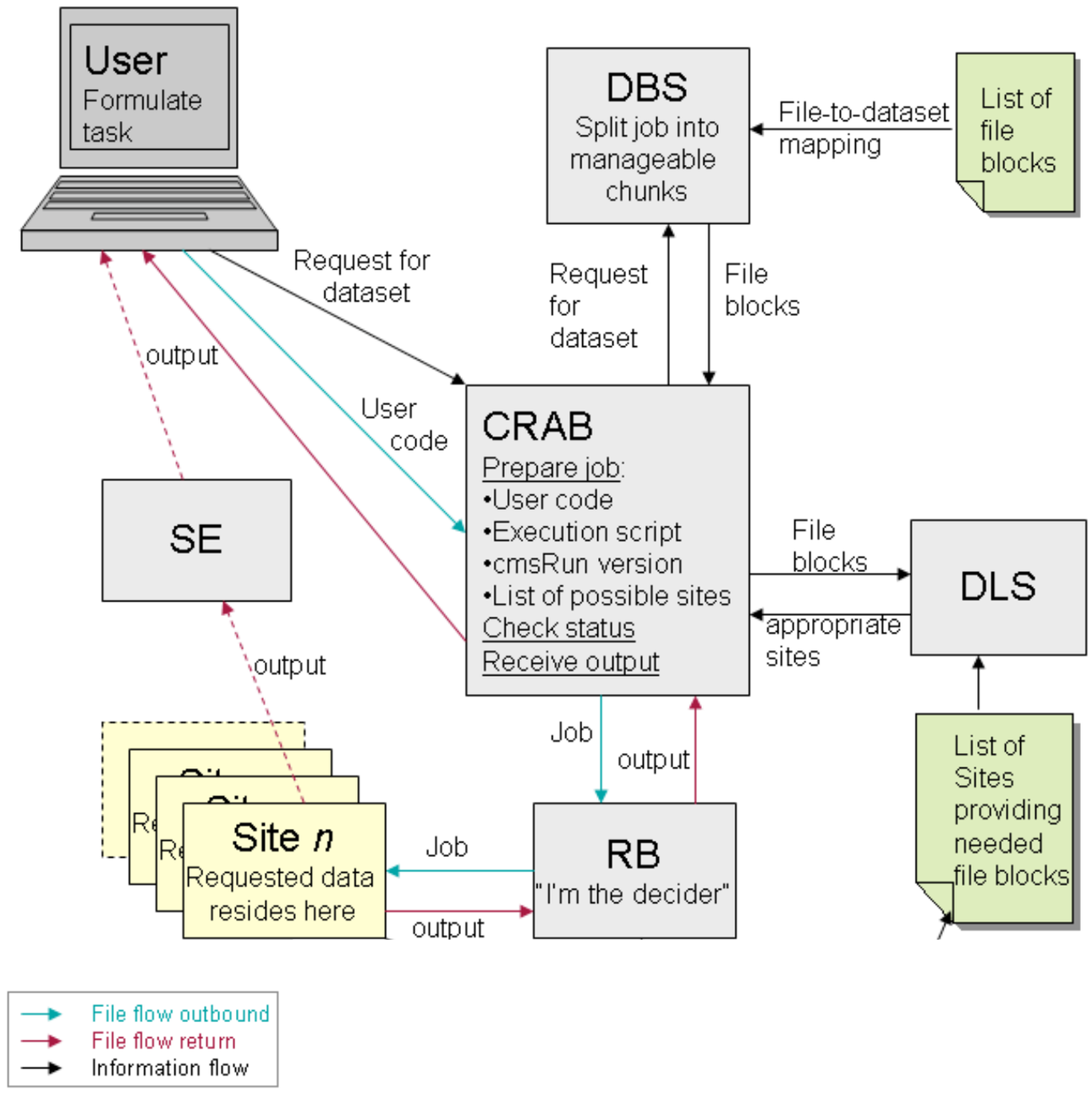
- CRAB2 Server handles the grid submission of jobs for the user, and retrieval of user output.
- Supports glideins, but also gLite etc.
- From the glidein point-of-view, the CRAB Server is a submitter (schedd).
- Authentication by user's grid certificates.

How CMS Does Analysis

- Data are distributed to sites asynchronously using PhEDEx, the CMS data transfer middleware.
- Most data analysis done at Tier-2 sites.
- Generally, run jobs where the data are.
- No site has all the data.
- *CMS Monte Carlo production also uses glideins at Tier-2 sites, but effectively a single-user system, different workflow management, etc.*

CRAB2

- ~400 users weekly; ~600 monthly.
- >100K jobs submitted each day.



How CMS will do Analysis in the Future

- This year CMS is rolling out CRAB3, based on WMAgent, a complete re-design of the workflow management.
- Maintains the client-server model of CRAB2 Server.
- CRAB3 Server will further isolate the user from the grid:
  - Tasks/workflows instead of "jobs"
  - Glidein submission (by the server, not the user), but gLite submission will be supported as well.
  - glideinWMS is a natural partner of CRAB3 in this mission.

Physics Results

- Successfully getting results to the physicist is the primary mission of computing for analysis.
- Single largest failure mode in CRAB2 was writing out of job results to a remote site.
- User analysis jobs ~90% success rate
- ~5% of jobs fail remote stage out.

Asynchronous Stage Out

- CRAB3 includes concept of Asynchronous Stage Out.
- Write job results locally on storage element (SE) at the site where the job ran
- Transfer to destination SE asynchronously
- Should be more fault-tolerant (destination site is temporarily unavailable, e.g.)

Site Validation

- CMS validates sites where jobs can run by periodic SAM tests.
- No active connection between workflow management and site validation in CRAB2.

Legend: NA OK MAINTENANCE ERROR WARNING INFO NOTE CRITICAL

Note: brightest colors: test is 0 - 6 hours old, ... lightest colors: test is more that 24 hours old

Link to the table

Sitename	Service Type	Service Name	nCEjsub	nCEbas	nCEsquid	nCEfront	nCEmc	nCEana	nCEswinst	nSRMdel	nSRMput	nSRMget	nSRMgetPFN
T2_US_UCSD	CE	osg-gw-2.t2.ucsd.edu	ok	ok	ok	ok	ok	ok	ok				
		osg-gw-4.t2.ucsd.edu	ok	ok	ok	ok	ok	ok	ok				
	SRMv2	bsrm-1.t2.ucsd.edu								ok	ok	ok	ok

Site Readiness

- SAM and other tests are aggregated into an overall "Site Readiness" metric

T2_US_UCSD

Site Readiness Status:	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	W
Daily Metric:	E	E	E	E	E	O	O	O	O	O	O	O	O	O	O	O	O	O	O	E	
Maintenance (Topology):	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	Up	
Job Robot:	99%	99%	100%	92%	99%	99%	98%	100%	99%	99%	100%	99%	100%	100%	100%	100%	98%	98%	97%	99%	99%
SAM Availability:	100%	100%	100%	100%	100%	92%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	96%	100%	92%	80%
Good T2 links from T1s:	0/8	0/8	0/8	0/8	1/8	7/8	8/8	7/8	7/8	7/8	7/8	8/8	8/8	7/8	8/8	8/8	8/8	8/8	8/8	6/8	1/8
Good T2 links to T1s:	6/8	7/8	8/8	8/8	7/8	7/8	8/8	8/8	7/8	7/8	7/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8
Active T2 links from T1s:	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
Active T2 links to T1s:	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
	29	30	31	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
	Dec			Jan																		

Glidein Validation

- Glidein validation scripts can provide that link.
- We are currently implementing basic SAM tests as glidein validation step for the pilots.
  - CMSSW version check already implemented
  - Check that local writing to storage works - essential for Asynchronous Stage Out

Legend: NA OK MAINTENANCE ERROR WARNING INFO NOTE CRITICAL

Note: brightest colors: test is 0 - 6 hours old, ... lightest colors: test is more that 24 hours old

Link to the table

Sitename	Service Type	Service Name	nCEjsub	nCEbas	nCEsquid	nCEfront	nCEmc	nCEana	nCEswinst	nSRMdel	nSRMput	nSRMget	nSRMgetPFN
T2_US_UCSD	CE	osg-gw-2.t2.ucsd.edu	ok	ok	ok	ok	ok	ok	ok				
		osg-gw-4.t2.ucsd.edu	ok	ok	ok	ok	ok	ok	ok				
	SRMv2	bsrm-1.t2.ucsd.edu								ok	ok	ok	ok

Experience with Glideins: Common Problems

- Job Matching Issues
- Resource-hungry users - memory and disk space usage on the worker nodes
- glexec failures - like a validation failure, factory admins can act on this info and notify sites.

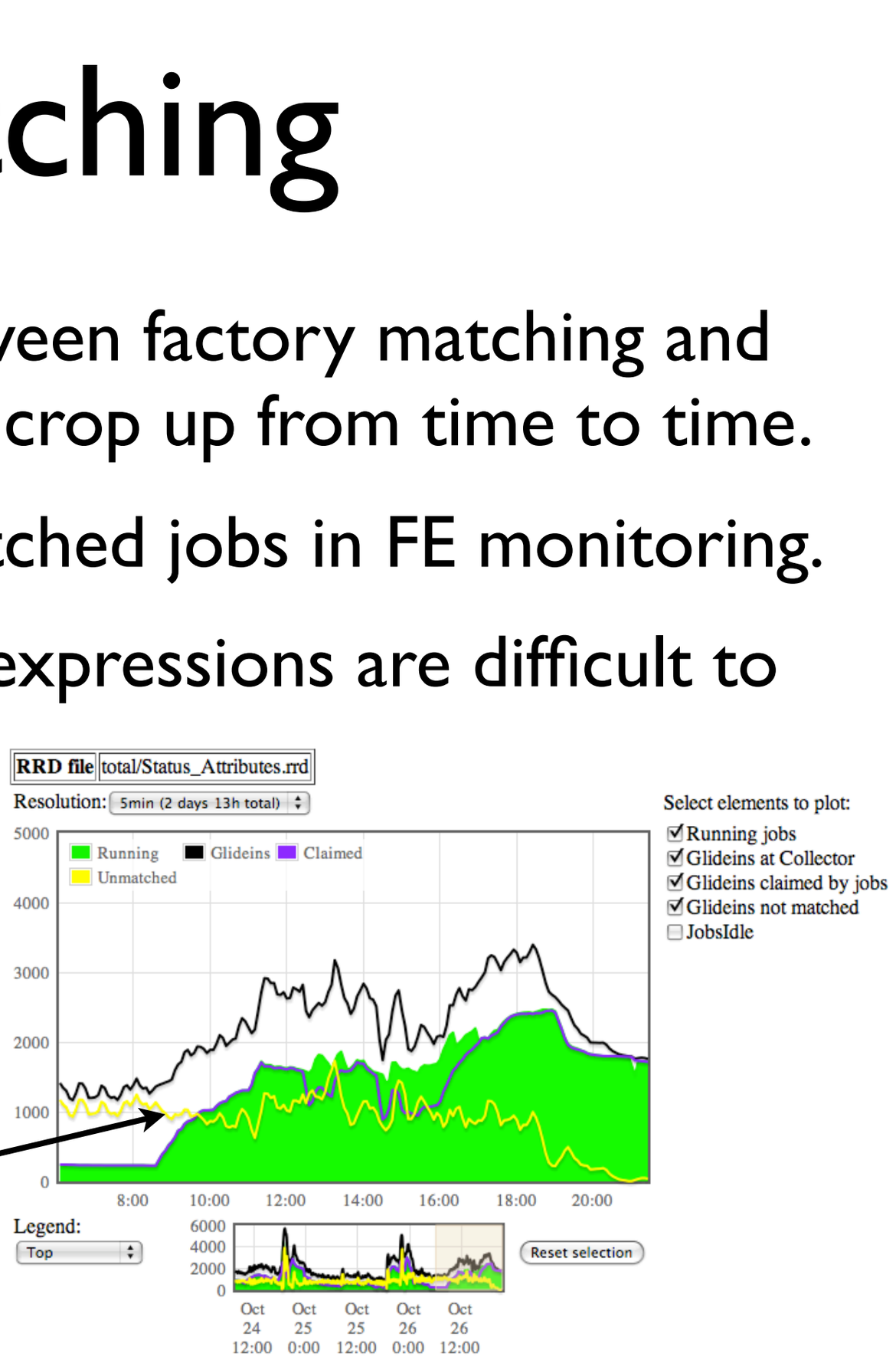
Job Matching

- CMS glidein matching condition expressions are somewhat complex.
- Match mainly by desired_SE (bring jobs to the data)
- Also restrictions on ImageSize for re-submission
- Limitations on maximum running time of jobs; pilots expire after ~24h or after 20m of idle time.
- Single user group - prototype priority user group, but not fully implemented.

Matching

- Inconsistencies between factory matching and job start conditions crop up from time to time.
- Seen as many unmatched jobs in FE monitoring.
- However, matching expressions are difficult to read, and harder to debug.
- Often ImageSize

unmatched jobs



Debugging Matching

- CONDOR has a wealth of information about jobs - almost too much.
- Why does job x (or a set of jobs) not match any running pilots?
- Not easy to check 1000's of jobs and 1000's of pilots against complex matching conditions.
- End up writing my own python scripts to loop through all the jobs and check: glideinWMS utilities library is very helpful!

```
[1516] frontend@glidein-collector ~/letts/scripts$ ./test_monitor.py
Number of Pilots Running                    2916
Number of Claimed Pilots                    2510
Number of Unclaimed Pilots                   415
Efficiency of claimed pilots                86.0 %
```

Matching Conditions

```
match_expr='(job.has_key("DESIRED_Site
s") and (glidein["attrs"]
["GLIDEIN_Site"] in
job["DESIRED_Sites"].split(","))) or
(job.has_key("DESIRED_Gatekeepers")
and (glidein["attrs"]
["GLIDEIN_Gatekeeper"] in
job["DESIRED_Gatekeepers"].split(","))
) or (job.has_key("DESIRED_SEs") and
glidein["attrs"].has_key("GLIDEIN_SEs"
) and (glidein["attrs"]["GLIDEIN_SEs"]
in job["DESIRED_SEs"].split(",")))'
```

Matching Conditions

```
match_expr='(job["JobStatus"]==2) or (((not
glidein["attrs"].has_key("GLIDEIN_Max_Walltime")) or (((not
job.has_key("LastVacateTime")) and ((not
job.has_key("NormMaxWallTimeMins")) or
((job["NormMaxWallTimeMins"]+10)&lt;((glidein["attrs"]
["GLIDEIN_Max_Walltime"]-glidein["attrs"]
["GLIDEIN_Retire_Time_Spread"]-glidein["attrs"]
["GLIDEIN_Job_Max_Time"])/60)))) or
((job.has_key("LastVacateTime")) and ((not
job.has_key("MaxWallTimeMins")) or ((job["MaxWallTimeMins"]
+10)&lt;((glidein["attrs"]["GLIDEIN_Max_Walltime"]-
glidein["attrs"]["GLIDEIN_Retire_Time_Spread"]-
glidein["attrs"]["GLIDEIN_Job_Max_Time"])/60)))))) and
((not job.has_key("ImageSize")) or
(job["ImageSize"]&lt;=(glidein["attrs"]
["GLIDEIN_MaxMemMBs"]*1024))) and (((not
job.has_key("NumJobStarts")) or (job["NumJobStarts"]&lt;5))
or (job.has_key("LastVacateTime") and ((job["ServerTime"]-
job["LastVacateTime"])&gt;3600))))'
```

Job Start Conditions

```
<attr name="GLIDECLIENT_Start" comment="In the first
try, use the NormMaxWallTimeMins limit, in the next
tries, use MaxWallTimeMins limit, if defined"
glidein_publish="False" job_publish="False"
parameter="True" type="string"
value="ifthenelse(LastVacateTime=?
=UNDEFINED,ifthenelse(NormMaxWallTimeMins=!=UNDEFINED,
(NormMaxWallTimeMins*60)&lt;(GLIDEIN_ToRetire
+GLIDEIN_Job_Max_Time-MyCurrentTime),(8*3600)&lt;
(GLIDEIN_ToRetire+GLIDEIN_Job_Max_Time-
MyCurrentTime)),ifthenelse(MaxWallTimeMins=!
=UNDEFINED,(MaxWallTimeMins*60)&lt;(GLIDEIN_ToRetire
+GLIDEIN_Job_Max_Time-MyCurrentTime),(16*3600)&lt;
(GLIDEIN_ToRetire+GLIDEIN_Job_Max_Time-
MyCurrentTime)))&amp;&amp;
(ImageSize&lt;=(GLIDEIN_MaxMemMBs*1024))&amp;&amp;
(JOB_Is_ITB =!= TRUE)"/>
```

Resource-Hungry Users

- No actions taken (yet) on ImageSize, disk space usage of *running* jobs.
- Contemplated, but no consensus yet on what the CMS policy should be.
- Would be easy enough to implement in the PeriodicRemove condition.
- Currently left to sites to protect themselves.

glexec

- We use glexec where installed.
- glexec can fail in certain circumstances, causing pilot to fail validation:
  - Full system partition on worker node.
  - Somewhat difficult for FE admin to monitor: usually noticed first from the Factory side.

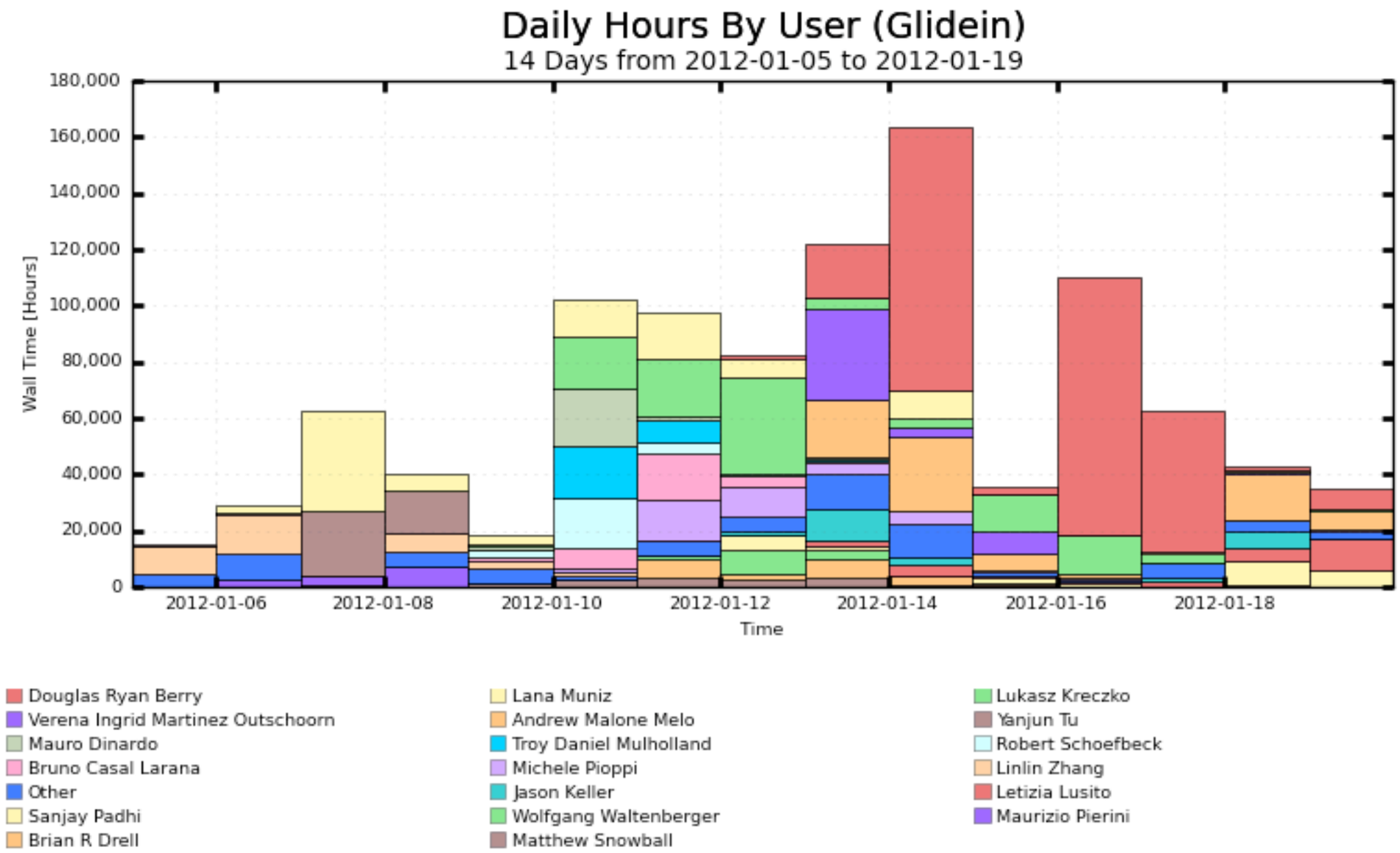
CMS Job Robot

- As part of the Site Readiness, CMS regularly submits "Job Robot" tasks to sites.
- Well-defined job, data set that should work everywhere. Based on a realistic physics analysis workflow.
- Only gLite submission.

HammerCloud

- Last year we implemented a standard grid job like Job Robot for glidein submission through the CRAB2 Server using CERN-IT's HammerCloud service.
- Can schedule regular or on-demand tests.
- Often high-level physics support people do not know how to run a simple but real workflow to validate the health of the system - this is very helpful!

GRATIA Probes on the CRAB2 Submitters



Overflow and Re-direction

- Run a separate Front End at UCSD for sites that have xrootd installed.
- Allows jobs to be sent to another site than the desired one, if over-loaded, and read the data over the WAN with xrootd.
- Effectively treats this group of sites (currently only in the US) to act as a single resource.
- xrootd also provides re-direction: if a file is unavailable at one site, look for it at another.

Conclusions

- glideinWMS is, and will remain, an integral part of the CMS data analysis (and Monte Carlo production) infrastructure.
- At UCSD we administer a front end as part of the CRAB2 Server system.
- Biggest challenge is debugging matching failures - develop our own tools *ad hoc*, but any common solution would be very helpful!
- Development efforts in site validation, and tests using realistic physics workflows.
- UCSD major participant in deploying xrootd for CMS