Rosetta & OSG Engagement VO

Resource Selection on the Grid

Mats Rynge

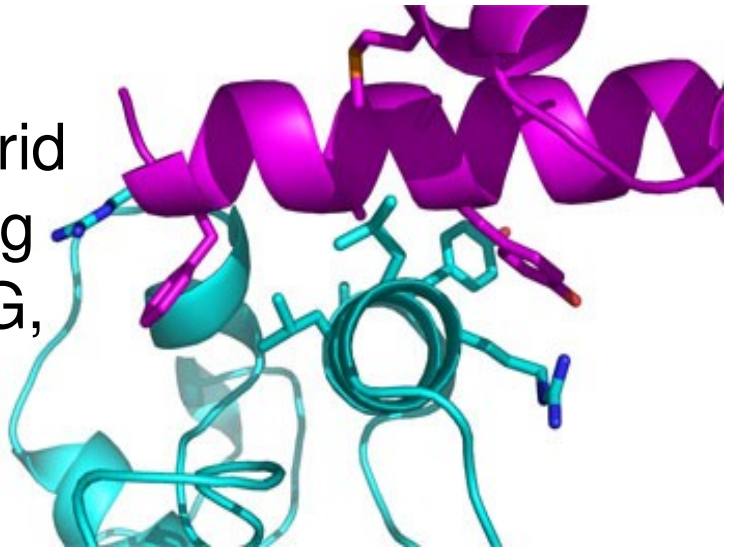
Renaissance Computing Institute

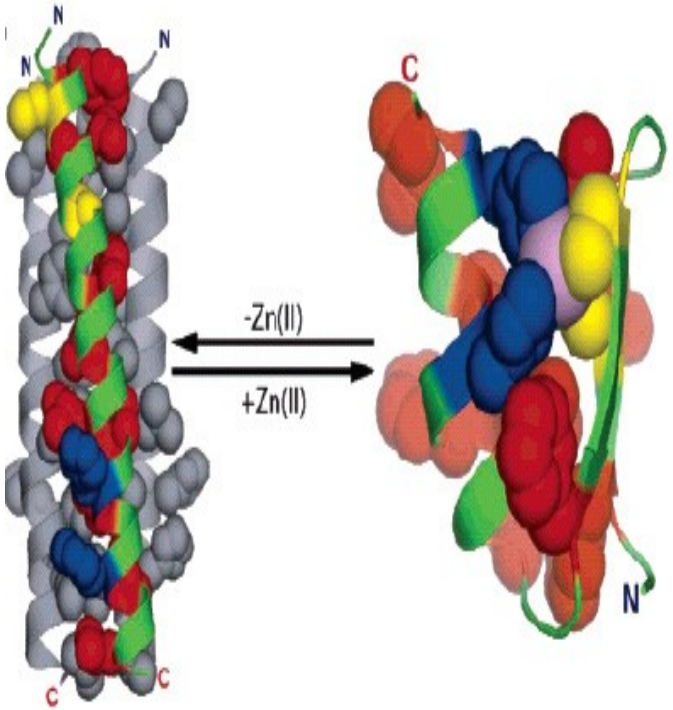
Engagement Mission

- Help new user communities from diverse scientific domains adapt their computational systems to leverage OSG
- Facilitate University Campus CI deployment, and interconnect it with the national organizations

Sample Engagement: Kuhlman Lab

Sr. Rosetta Researcher and his team, little CS/IT expertise, no grid expertise. Quickly up and running with large scale jobs across OSG, **>250k CPU hours**





Using OSG to design proteins that adopt specific three dimensional structures and bind and regulate target proteins important in cell biology and pathogenesis. These designed proteins are used in experiments with living cells to detect when and where the target proteins are activated in the cells.

Why resource selection?

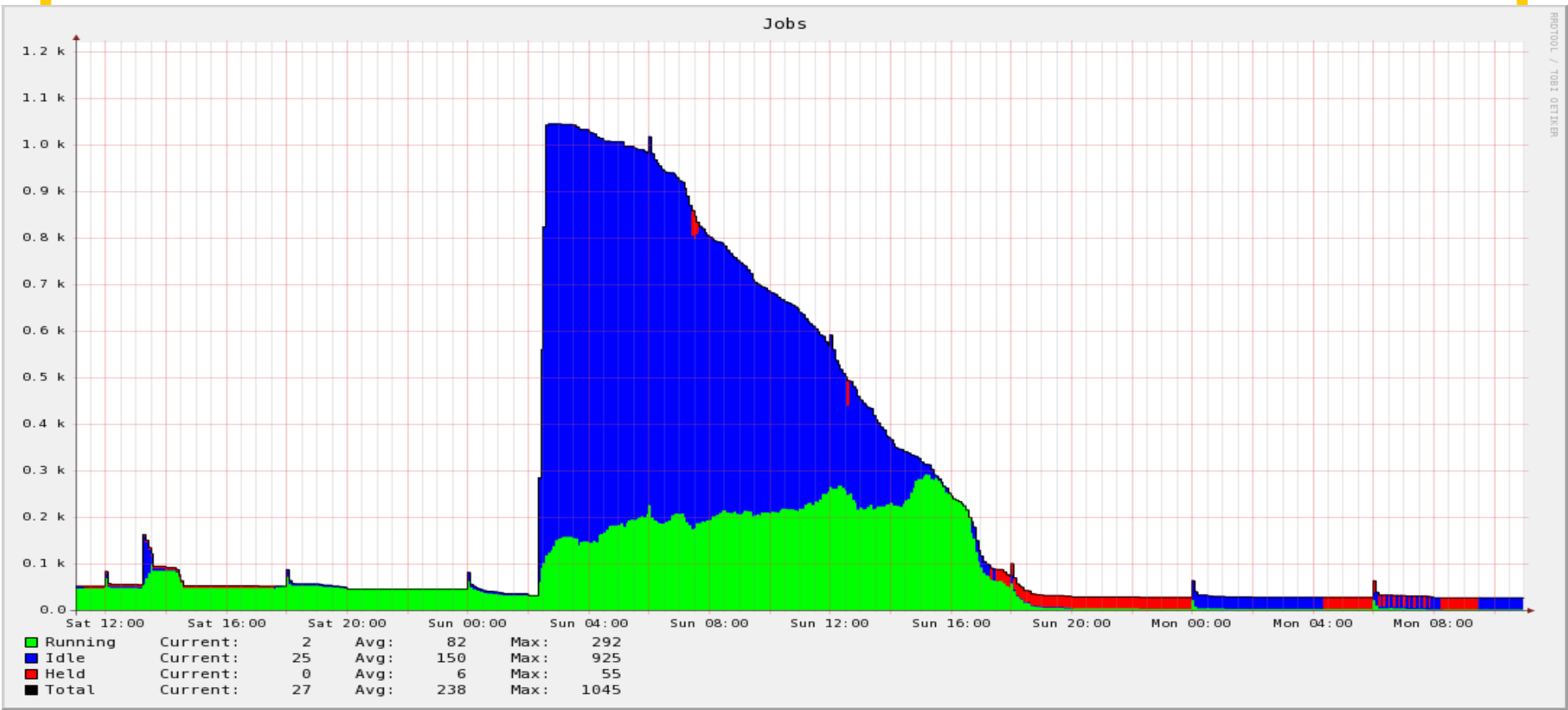
- Engagement users
  - Small labs/experiments
  - Little or no CS/IT resources
- Example: Kuhlman lab at UNC
  - Protein folding
  - 1 job run (~ 3 days)
  - 4 weeks in wet lab, using results from run
- No knowledge / time / resources / need for a complex job handling system

What is Resource Selection?

- Well described jobs and resources
- **Automatically** match the jobs up against resources
- Additional features include
  - automatic retries of failed jobs
  - site verification

Rosetta on OSG

- Good example on why resources selection is important
  - Each protein design requires about 5,000 CPU hours, distributed across 1,000 individual compute jobs



Engagement VO Jobs

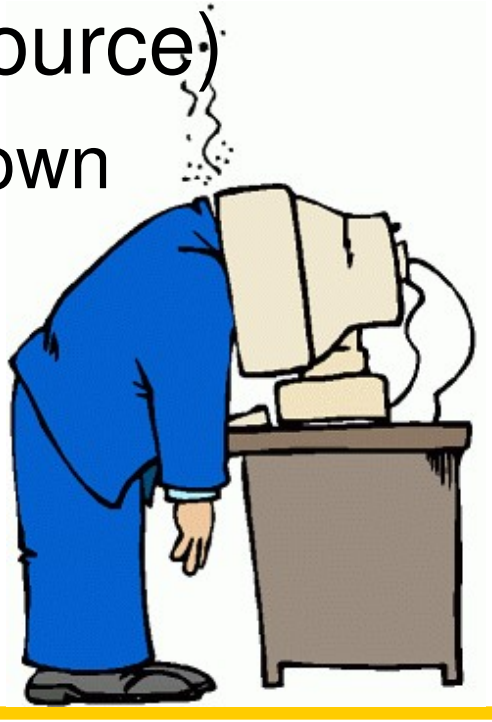
- Mostly very simple jobs
  - Embarrassingly parallel
  - No inter-job dependencies
  - Simple staging requirements
    - Inputs/outputs staged with job
- Job independence makes it easy to spread a run across many sites
- Great candidates for matchmaking

Engagements – Initial Interactions with new Users

- Users describes executables, needed inputs and example on how to run the model
- Every user is different, but in general, Engagement team creates:
  - submit tool (creates jobs / dags)
  - job-wrapper (wraps model remotely)
  - job-success-check (checks stdout)

Biggest Challenges?

- Used to be security
- Big step to go from 1 job to 1000s
  - job / data management
- The black box (aka remote resource)
  - small differences hard to track down
  - environment configuration
  - file system configuration
  - available system utilities
  - network setup



Condor

- Condor is the base system in the Engagement Resource Selection system
- Quick introduction to Condor
  - Representation of resources and jobs
    - Class ads
  - Match making
    - Requirements
    - Ranks

Condor
High Throughput Computing

Condor ClassAds

- “Condor's ClassAds are analogous to the classified advertising section of the newspaper. Sellers advertise specifics about what they have to sell, hoping to attract a buyer. Buyers may advertise specifics about what they wish to purchase. Both buyers and sellers list constraints that need to be satisfied. For instance, a buyer has a maximum spending limit, and a seller requires a minimum purchase price. Furthermore, both want to rank requests to their own advantage. Certainly a seller would rank one offer of $50 dollars higher than a different offer of $25. In Condor, users submitting jobs can be thought of as buyers of compute resources and machine owners are sellers.”

(source: Condor documentation)

ClassAds (cont.)

- Both resources and jobs uses the same format to describe themselves
- ASCII
- Key/value pairs
- Simplicity of the ClassAds is good news, because it means they are easy to create/manipulate

Resources

Resource ClassAd

```
MyType = "Machine"
GlueSubClusterLogicalCPUs = 2
GlueCEPolicyAssignedJobSlots = 0
GlueCEInfoHostName = "antaeus.hpcc.ttu.edu"
GlueHostNetworkAdapterOutboundIP = TRUE
GlueHostArchitectureSMPSize = 2
EngageSoftware_Rosetta_v3 = TRUE
EngageMemPerCPU = 1010460
GlueSubClusterWNTmpDir = "/state/partition1"
EngageOSGAPPWriteWorkNode = TRUE
GlueCEInfoContactString =
  "antaeus.hpcc.ttu.edu:2119/jobmanager-lsf"
GlueHostOperatingSystemName = "CentOS"
```

Job / Resource Ranks

- How is the “best” site chosen from a set of sites matching a job’s requirements?
- Ranks can be an expression
  - CPU speed, job success rate
- Engagement is using a custom resource rank (explained later)

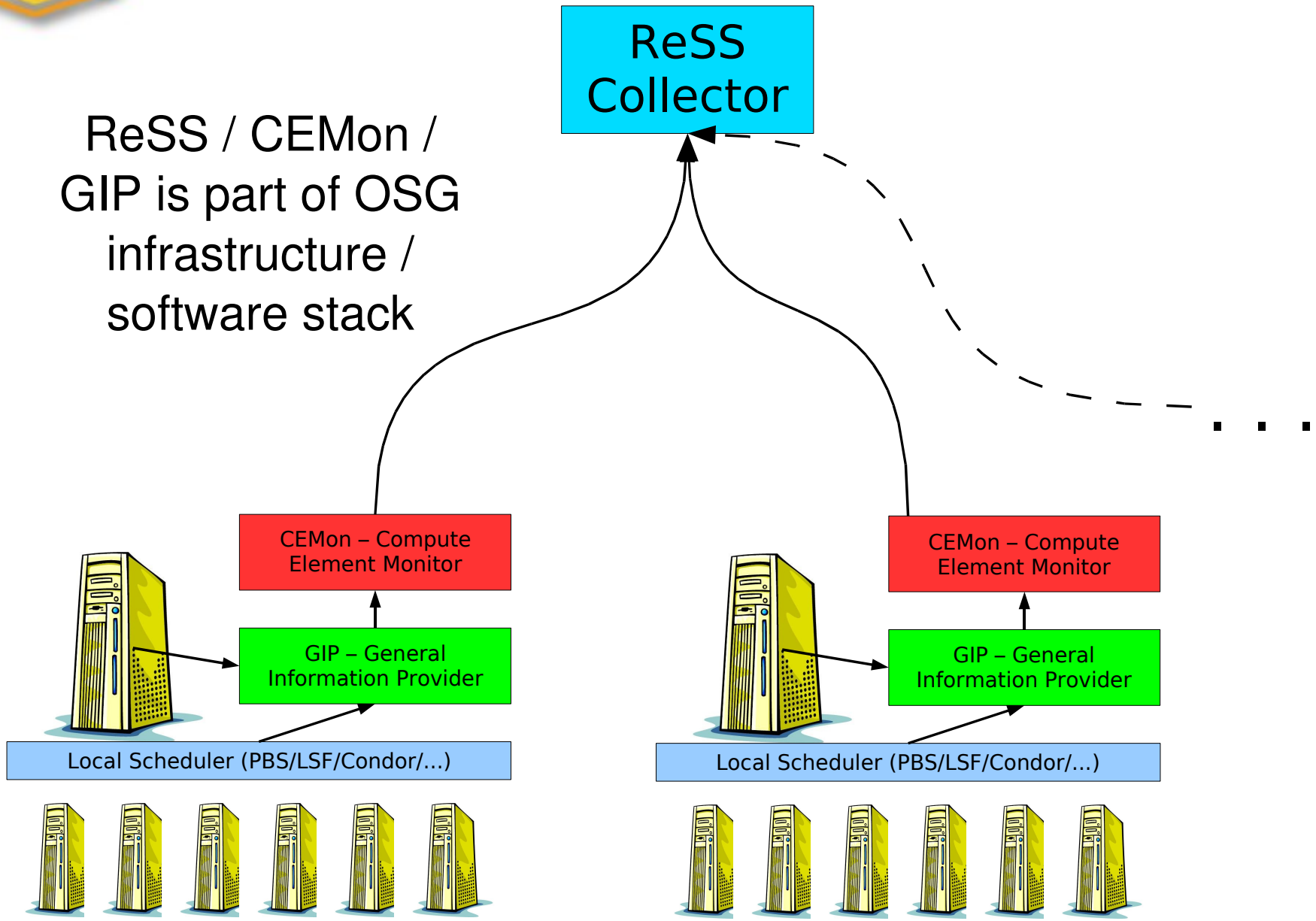
ReSS

- Resource Selection Service
  - but is only really an information provider
- Developed at Fermi Lab and is part of OSG infrastructure
- Collects data from compute elements (clusters) and publishes the data in Condor ClassAd format

Information in ReSS

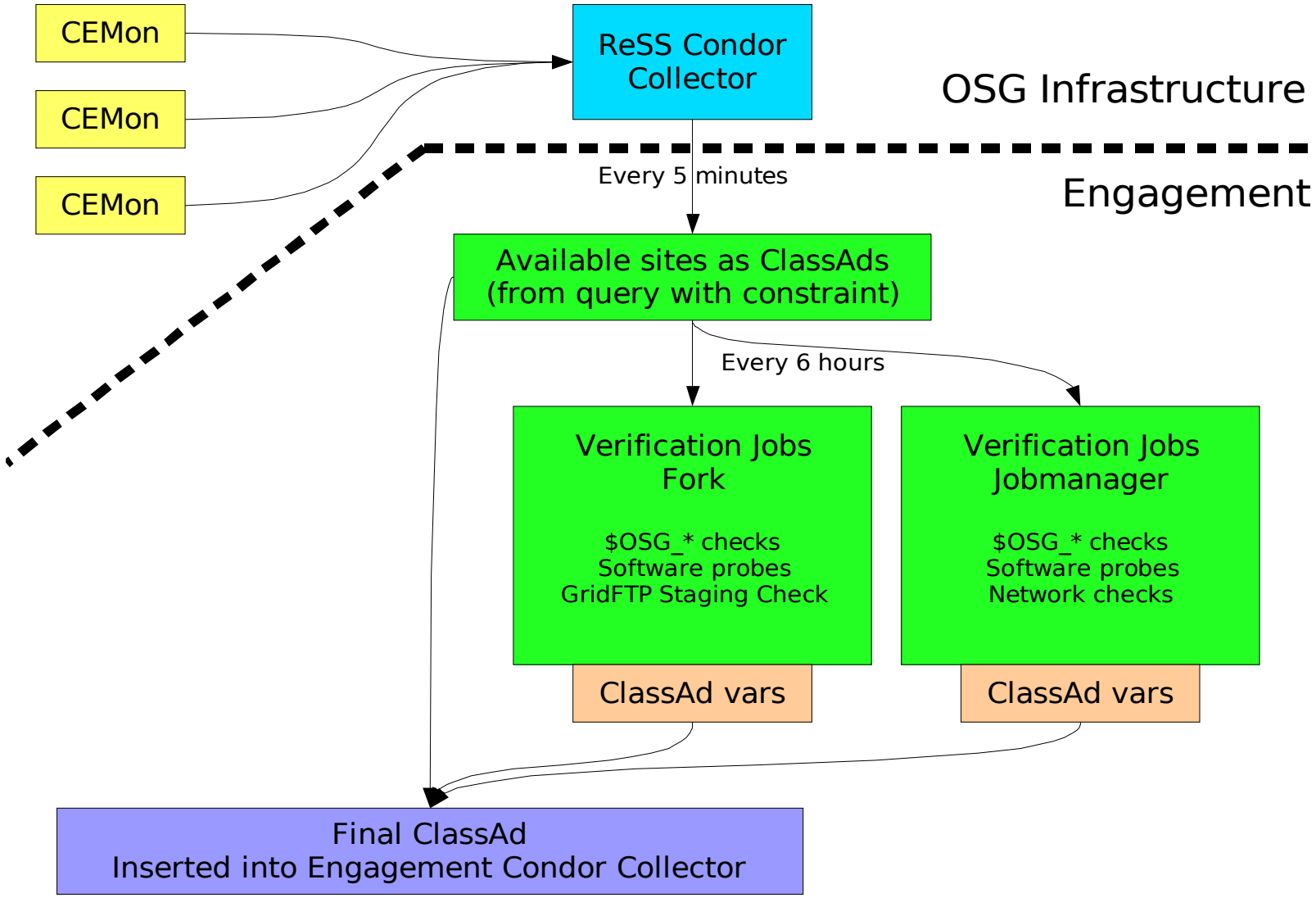
- OS name / version
- LRM information
  - Total number of job slots
  - Assigned slots
  - Open job slots
- Memory / CPU / Disk
- Network setup
- Storage configuration

ReSS / CEMon / GIP is part of OSG infrastructure / software stack



Engagement ReSS Layer

- Retrieve base ClassAds from ReSS
- Validate the sites with probe jobs
- Determine the current state of the system by looking at current job states and success rates (continuous system feedback)
- Merge the information, and insert into local Condor system


CEMon
CEMon
CEMon
ReSS Condor Collector
OSG Infrastructure
Every 5 minutes
Engagement
Available sites as ClassAds (from query with constraint)
Every 6 hours
Verification Jobs Fork
$OSG_* checks
Software probes
GridFTP Staging Check
Verification Jobs Jobmanager
$OSG_* checks
Software probes
Network checks
ClassAd vars
ClassAd vars
Final ClassAd
Inserted into Engagement Condor Collector

Verification: File Systems

- $OSG_WN_TMP
  - most of the time local, similar to /tmp
    Exist? Write permissions?
- $OSG_DATA
  - shared, read/write from worknodes
    Is it mounted? Permissions? Does it have the data we staged earlier?
- $OSG_APP
  - shared, read-only from worknodes
    Is it mounted? Permissions? Does it have the applications we expect?

ReSS + Site Verification

- ReSS information + site verification:
  - The result is a set of resource classads
    - Well formed
    - Verified information
  - Users' can trust the information
  - Increases job success rates

Site Rank

- Simple – but works!
- Integer between 0 and 1000
- Calculated every minute from current state
- Factors:
  - Jobs submitting/staging/pending/running provides the baseline
  - Job success rate for the site over the last 6 hours
  - Ratio between matched jobs, and the max number we want on that site

Jobs

Job Requirements

- Can you list all the requirements for your jobs?
  - Memory usage?
  - Disk usage?
  - Dependencies?
- Most users have a hard time describing their models

Additional Job Requirements from the Resource Selection

- Job fails...
- Job is in the queue for too long...
- Job is running for too long...

} resubmit to another site

- When submitting to another site, do not submit to a site which we have already failed on

Condor Submit File

```
globusscheduler = $$(GlueCEInfoContactString)

requirements = (
   (TARGET.GlueCEInfoContactString =!= UNDEFINED) &&
   (TARGET.Rank > 300) &&
   (EngageSoftwareWget == True) &&
   (TARGET.EngageCENetworkOutbound == True))
```

```
# when retrying, remember the last 4 resources tried
match_list_length = 4
Rank               = (TARGET.Rank) -
   ((TARGET.Name =?= LastMatchName0) * 1000) -
   ((TARGET.Name =?= LastMatchName1) * 1000) -
   ((TARGET.Name =?= LastMatchName2) * 1000) -
   ((TARGET.Name =?= LastMatchName3) * 1000)
```

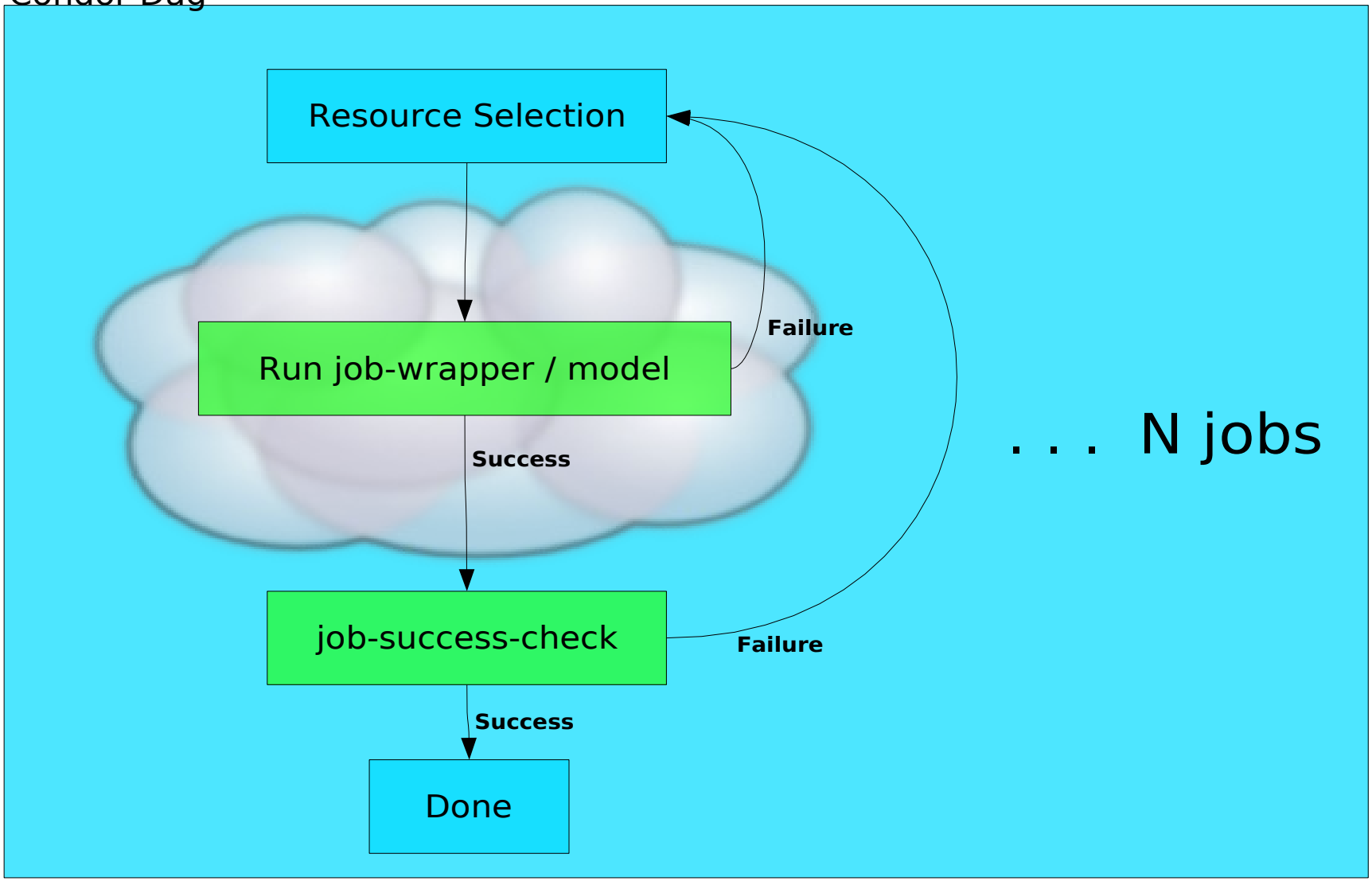
Condor Submit File (cont.)

```
# make sure the job is being retried and rematched
periodic_release = (NumGlobusSubmits < 10)
globusresubmit = (NumSystemHolds >= NumJobMatches)
rematch = True
globus_rematch = True
```

```
# only allow for the job to be queued or running for a while
# then try to move it
#  JobStatus==1 is pending
#  JobStatus==2 is running
periodic_hold = (
  ((JobStatus==1) && ((CurrentTime - EnteredCurrentStatus) >
   (5*60*60))) ||
  ((JobStatus==2) && ((CurrentTime - EnteredCurrentStatus) >
   (24*60*60))) )
```

Job Life Cycle

Condor Dag



Rosetta

- Rosetta database is pre-installed on compute elements
  - availability is advertised in resource ClassAds and can be used job requirement expressions
- Scripts:
  - generate-run
  - rosetta-wrapper
  - job-success-check
  - extract-results

rosetta-wrapper

- Bigger than most other Engagement wrappers
- Rosetta has a complicated command line and many different modes of operation

job-success-wrapper

- Checks stdout of finished jobs to determine if the execution was successful or not
- Runs locally, part of the DAG
- Why?
  - Some job managers do not return real exit codes
  - Part of the automatic job re-submit

makeself

- Generates a self-extractable tar.gz
- Similar to software installers
  - one executable
  - includes payload
- Rosetta jobs are makeself executables containing the job wrapper, the Rosetta executable, and a set of inputs

CLI: condor_grid_overview

```
ID      Owner   Resource        Status    Time Sta Sub
======= ====== ============== ======== ======== ===
46381   rynge   (DAGMan)                  1:58:54
46382   rynge   GLOW            Running   1:55:43   1
46384   rynge   UWMilwaukee     Pending   1:57:04   1
46387   rynge   Nebraska        Running   1:00:43   1


Site          Jobs   Subm   Pend   Run   Stage   Fail   Rank
============ ===== ===== ===== ===== ===== ===== =====
ASGC_OSG        17      0      0    15      2      0    155
FNAL_GPFARM     14      4      0    10      0      0    720
GLOW            36      6      5    22      3      0    372
Nebraska        17      0      5    12      0      0    288
Purdue-Lear     15      4      0    10      1      0    372
TTU-ANTAEUS     15      2      0    11      2      0    372
Vanderbilt      45      4      4    37      0      0    350
```

DEMO

Outline

- A closer look at condor_grid_overview
- Submit a set of Rosetta jobs
- Monitoring

Match Maker Visualization

- Not useful in real life, but looks cool
- Created for a SC'07 demo
- Shows jobs a small discs flying around

generate-run

- Script to build Condor submit files
- Number of jobs
- Number of PDBs per job
- Example:

- ./generate-run --number-jobs=100 --number-pdbs-per-job=2

Submit a Run

```
voms-proxy-info

./generate-run –number-jobs=20 \
                --number-pdbs-per-job=1
```

Monitor the Run

- condor_grid_overview
- Match Maker visualization
- Graph

More Information

- ReSS (Resource Selection Service):
  https://twiki.grid.iu.edu/twiki/bin/view/ResourceSelection/WebHome
- OSG Engagement VO
  https://twiki.grid.iu.edu/twiki/bin/view/Engagement/WebHome
- Questions?
  - Email: osg@renci.org