Computation Institute

Open Science Grid

BOSCO as service and other considerations for connecting the Campus to OSG

Marco Mambelli
Computation Institute

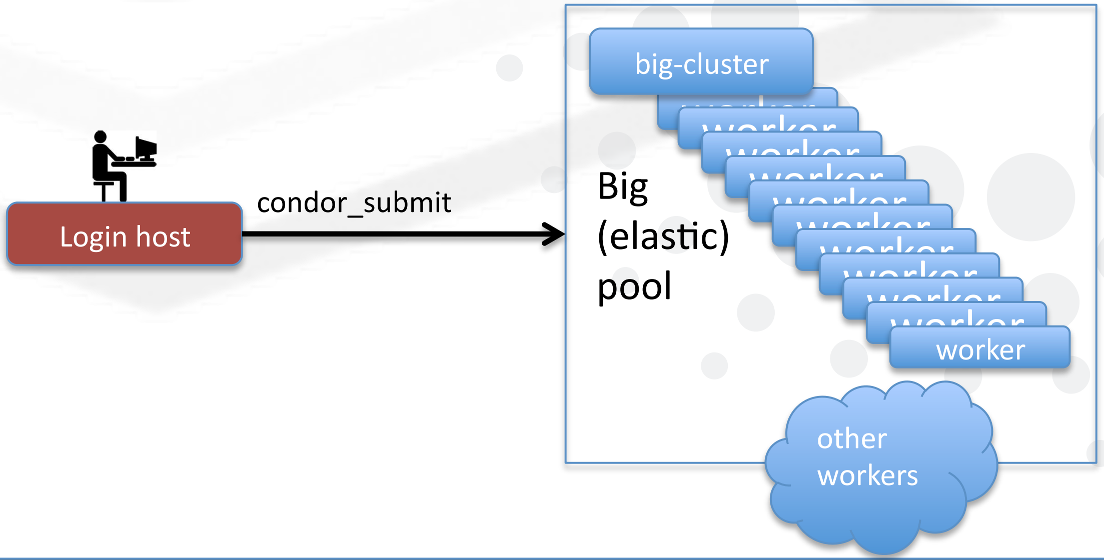
OSG All Hands – CIC day
14-15 March 2013
IUPUI, Indianapolis

THE UNIVERSITY OF CHICAGO

www.opensciencegrid.org
www.ci.uchicago.edu

Less is more

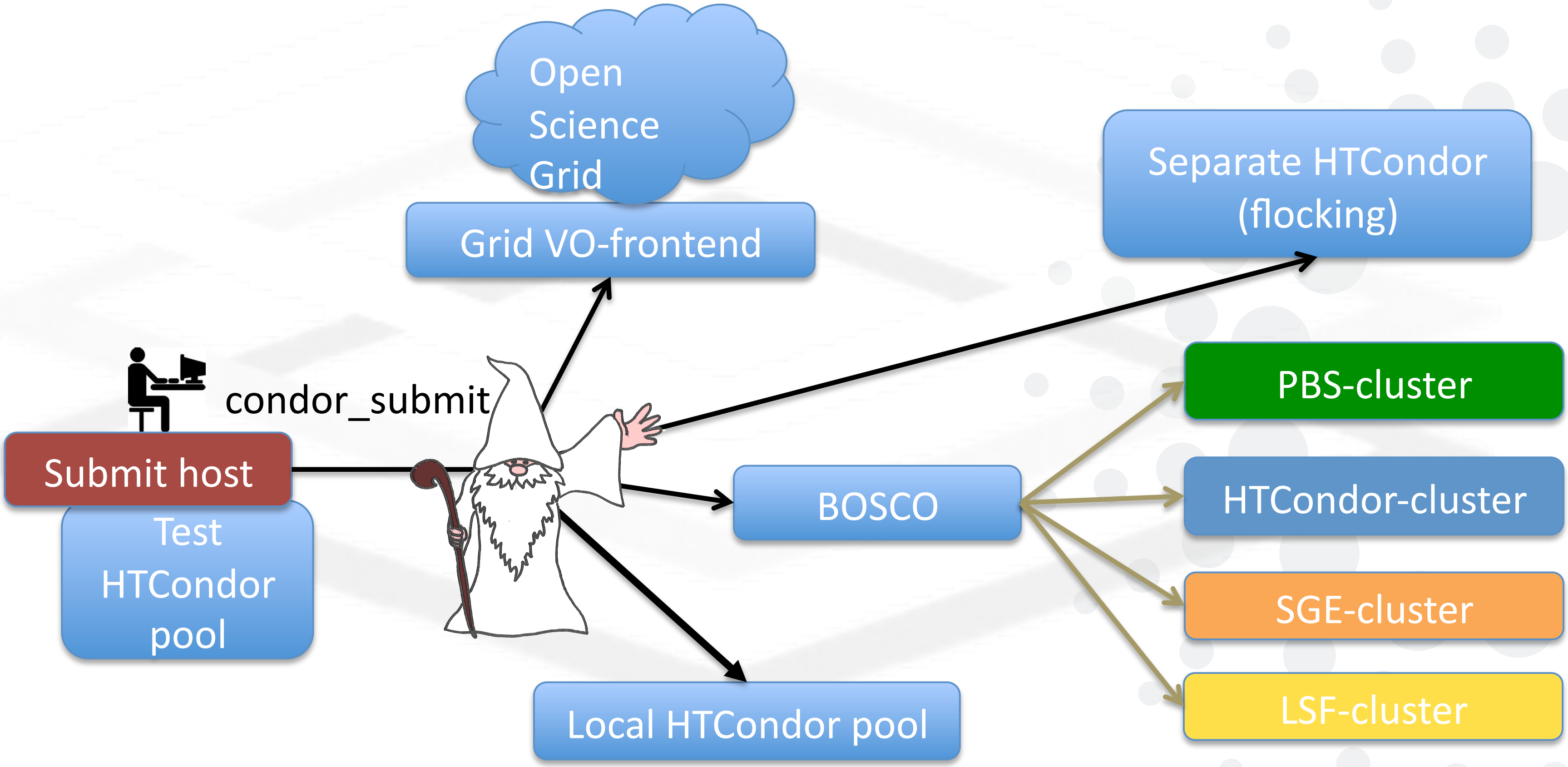
- BOSCO provides an easier way to access HTC
- What if the scientist can simply use HTC resources without worrying about what is available and how to make it available?
- How can this be provided in a simple and efficient way?



High Throughput Computing as Service

- Users
  - Scientists
  - Know how to submit jobs (command line or portal) or are paired with facilitators
- Administrator
  - (Unix) system administrator
  - Pool resources together
  - Provide a single access point and user interface
  - Provide simple instructions to access the resources

Campus Grid with BOSCO Multi User



Open Science Grid

Grid VO-frontend

Separate HTCondor (flocking)

condor_submit

Submit host

Test HTCondor pool

BOSCO

PBS-cluster

HTCondor-cluster

SGE-cluster

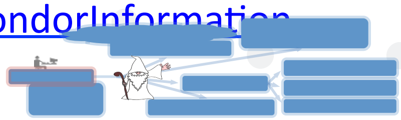
LSF-cluster

Local HTCondor pool

Components used in this Campus Infrastructure

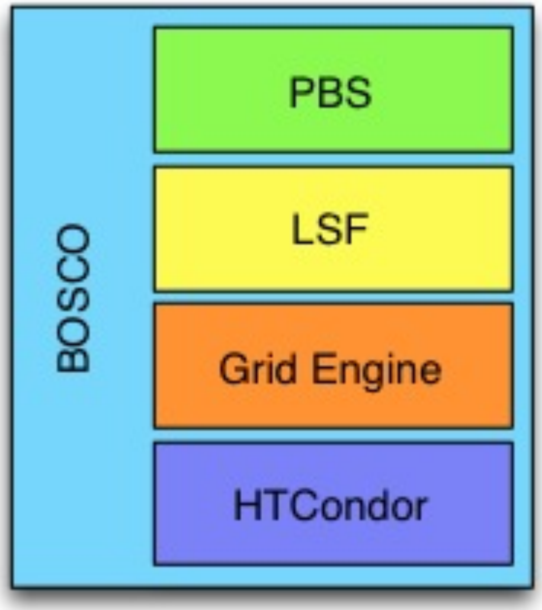
- The infrastructure is based on HTCondor
  - Allows to connect the different pieces (via “flocking”)
  - All HTCondor commands are available (condor_submit, condor_q)
- BOSCO
  - Allows to integrate different LRM
- OSG VO Frontend
  - Allows to submit to OSG
- Recipes
  - Examples of solution used in UC3

- Some references if you are not familiar with HTCondor:
  - https://www.opensciencegrid.org/bin/view/Documentation/Release3/InstallCondor
  - https://www.opensciencegrid.org/bin/view/Documentation/Release3/CondorInformation
  - http://research.cs.wisc.edu/htcondor/manual/v7.9/index.html



Uniform HTCondor interface for different resources “plugged” to the submit host

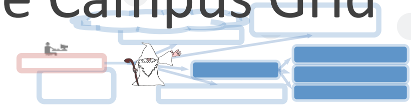
- BOSCO HTCondor Pool via BOSCO multi user
  - PBS
  - LSF
  - Grid Engine
  - HTCondor
- No root access on remote resources



BOSCO SINGLE-user vs. BOSCO MULTI-user

- Installed by user
- User manages BOSCO
  - BOSCO started as User
  - Contributing clusters (BOSCO resources) added by User
- User must have SSH access on all BOSCO resources
- Only User can submit jobs to the HTCondor pool of BOSCO
- No choices because it must be easy to install and run for scientists without system administration experience

- Installed as administrator (root)
- Administrator manages BOSCO
  - BOSCO started as root
  - Contributing clusters added using a single service account
- SSH access via group service account (negotiated by admin)
- All the users on the system can submit jobs to the HTCondor pool of BOSCO
- More flexible because there may be more customization to add BOSCO in the Campus Grid



BOSCO Installation and management

- https://twiki.grid.iu.edu/bin/view/CampusGrids/BoscoMultiUser

```
# As root – add the bosco user and the install direcotry
[root@bosco ~]$ useradd bosco
[root@bosco ~]$ mkdir -p /opt/bosco; chown bosco: /opt/bosco

# As bosco – download and run the installer
[user@bosco ~]$ wget ftp://ftp.cs.wisc.edu/condor/bosco/latest/boscoinstaller
[user@bosco ~]$ ./boscoinstaller --prefix=/opt/bosco
```

```
# As root – start/stop BOSCO
[root@bosco ~]$ source /opt/bosco/bosco_setenv
[root@bosco ~]$ bosco_start

# As bosco – add/remove/list BOSCO resources
[user@bosco ~]$ source /opt/bosco/bosco_setenv
[user@bosco ~]$ bosco_cluster --add username@mycluster-submit.mydomain.org LRMS
```

BOSCO Advanced Setup

- Setup the environment of the system installation

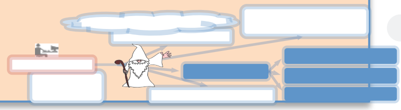
```
[root@bosco ~] cp /opt/bosco/bosco.sh /opt/bosco/bosco.csh /etc/profile.d/

[root@bosco ~] cp /opt/bosco/etc/examples/condor.boot.generic /etc/init.d/condor
# edit the values of CONDOR_CONFIG and CONDOR_CONFIG_VAL
  CONDOR_CONFIG=/opt/bosco/etc/condor_config
  CONDOR_CONFIG_VAL=/opt/bosco/bin/condor_config_val
[root@bosco ~]  chkconfig --level 235 condor on
```

- Tune some parameters for performance and to connect to CI

```
# Submission rate - in /opt/bosco/local.*/condor_config.local
GRIDMANAGER_MAX_SUBMITTED_JOBS_PER_RESOURCE = 100

# Allow flocking (running jobs in the BOSCO pool)  – in /opt/bosco/local.*/config/
condor_config.factory
FLOCK_FROM = yourname.yourdomain.org
```



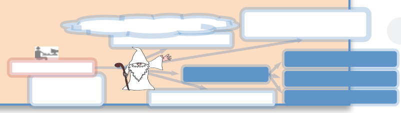
Differences in BOSCO job submission

- “grid” universe job
- Single resource addressed directly

- “vanilla” universe job
- Sent to all available resources

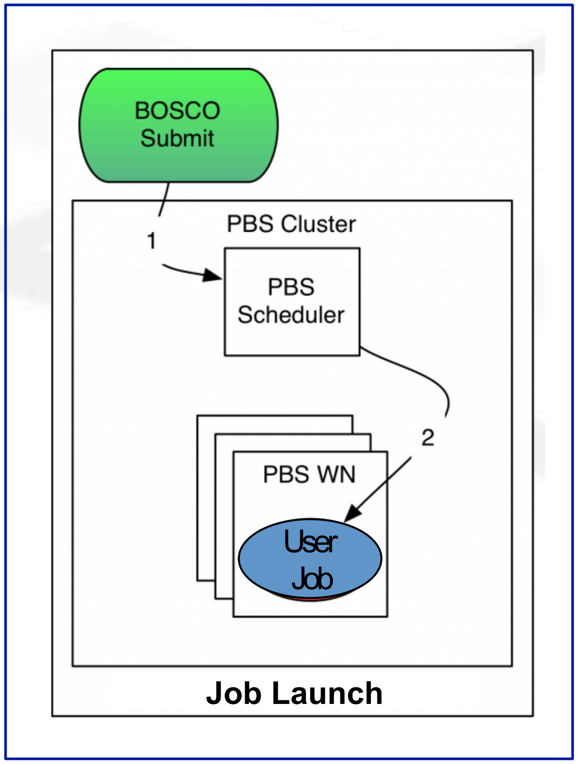
```
universe = grid
grid_resource = batch condor \
   marco@itbv-ce-condor.uchicago.edu
output = gjob.out.$(Cluster)-$(Process)
error = gjob.err.$(Cluster)-$(Process)
Executable     = myjob.sh
arguments = 10
log = gjob.log.$(Cluster)
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```

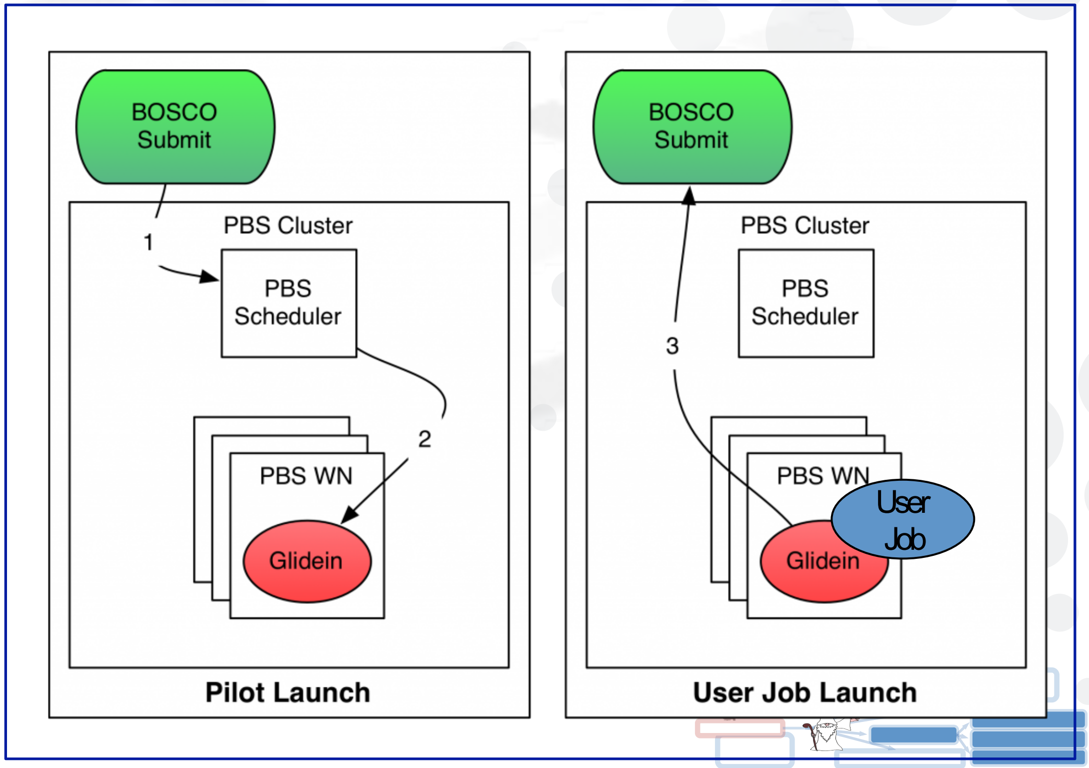
```
universe = vanilla
output = cfjob.out.$(Cluster)-$(Process)
error = cfjob.err.$(Cluster)-$(Process)
Executable     = myjob.sh
arguments = 10
log = cfjob.log.$(Cluster)
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
queue 1
```



Differences in BOSCO job submission (2)

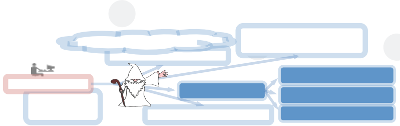
- Direct "grid" universe job batch
- "vanilla" universe job using HTCondor glideins: pilot + job

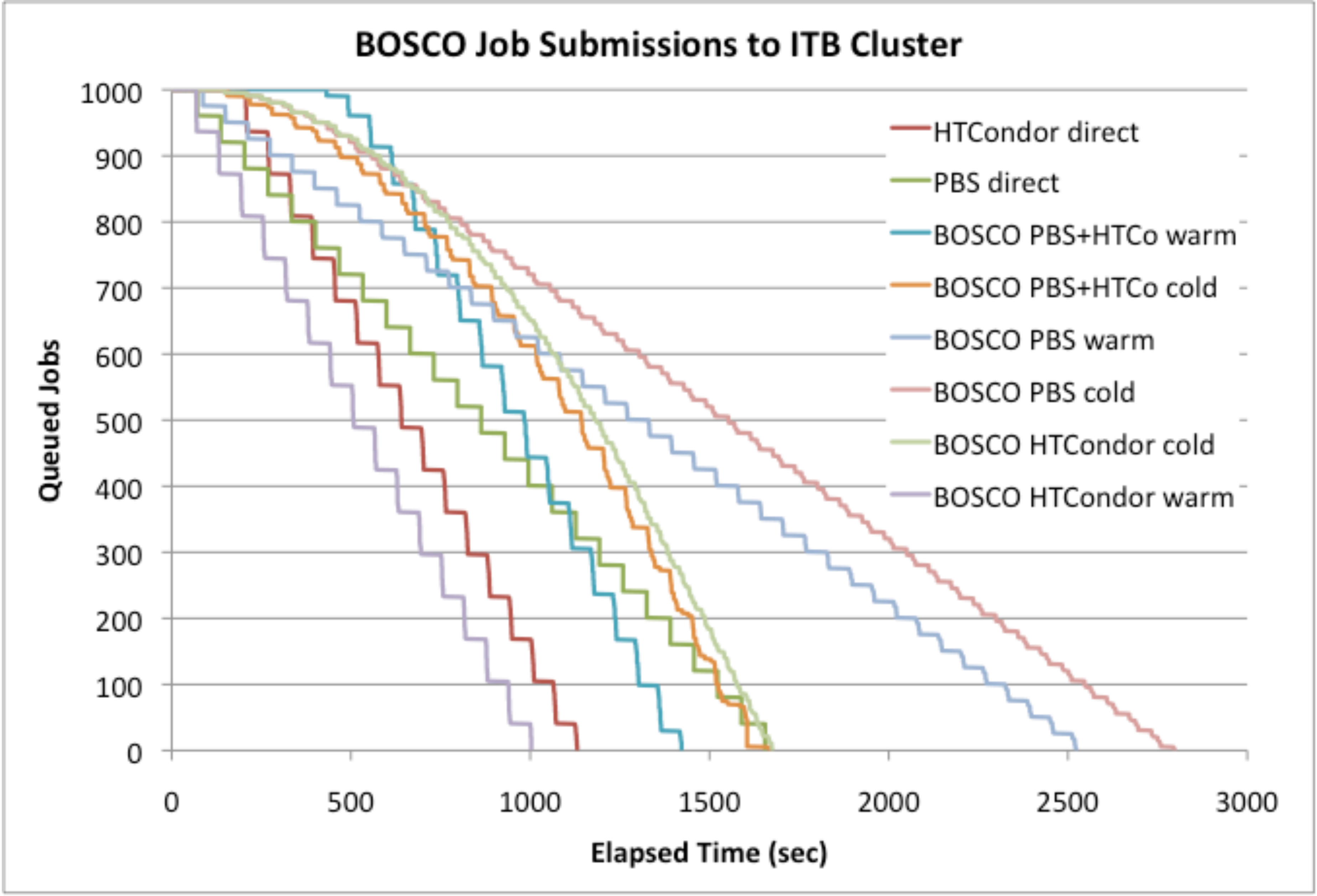




BOSCO performance test on dedicated cluster

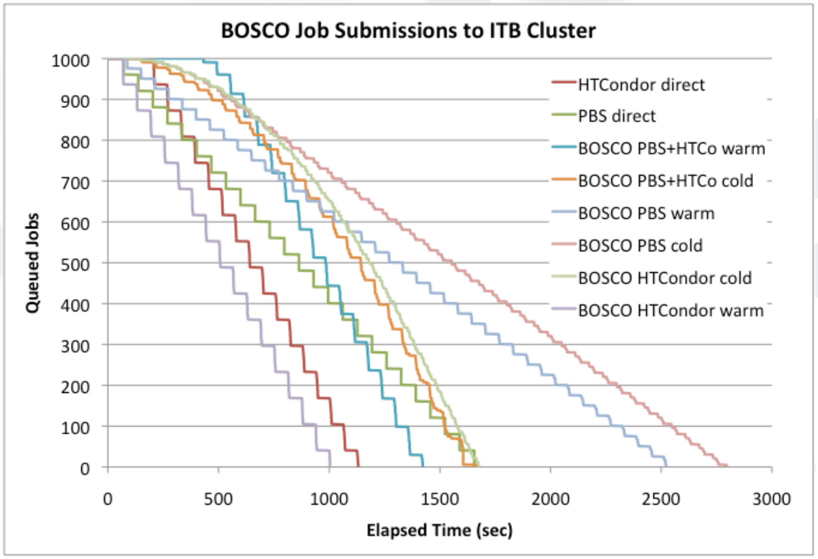
- Comparison of job submission rate with/without BOSCO. Submission of 1000 jobs with single command (most of the time) on an empty cluster (ITB)
  - Simple 1 min sleep job printing some system information (host where it ran, environment, …)
  - Local jobs, BOSCO “vanilla” jobs, BOSCO “grid” jobs
- The ITB cluster has a PBS (Torque/Maui) and a HTCondor queue with separate submit nodes and shared worker nodes (5 dual Intel E5620 hyperthreaded, 16 slots each)
- Measures:
  - Remote completion time (timestamp printed by the job)
  - Local completion time (last status change registered by BOSCO)
  - Almost no difference between the 2 measures




BOSCO Job Submissions to ITB Cluster
Queued Jobs
1000
900
800
700
600
500
400
300
200
100
0
0
500
1000
1500
2000
2500
3000
Elapsed Time (sec)
HTCondor direct
PBS direct
BOSCO PBS+HTCo warm
BOSCO PBS+HTCo cold
BOSCO PBS warm
BOSCO PBS cold
BOSCO HTCondor cold
BOSCO HTCondor warm

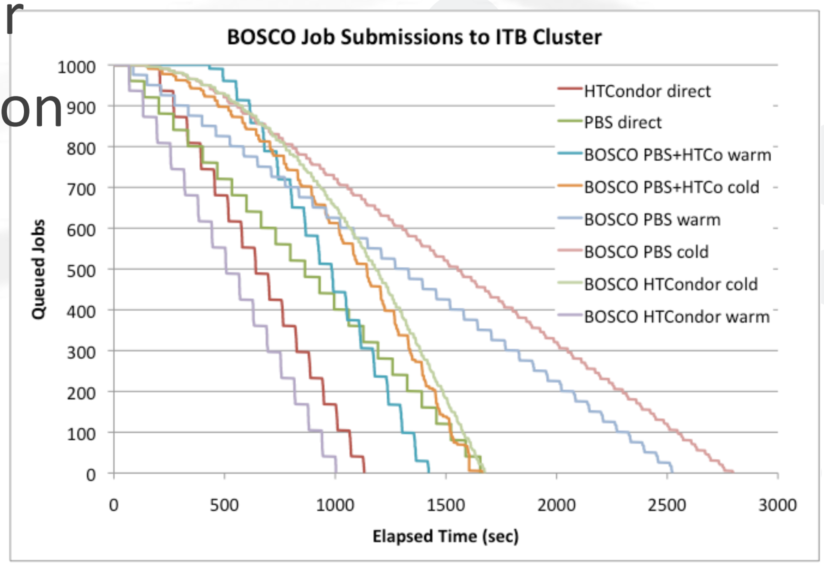
Performance of cluster Job Managers

- HTCondor submission (condor direct)
  - Has 64 job slots
  - 20% overhead
- Direct PBS submission (pbs direct)
  - Runs 40 jobs/time (user limit)
  - 11% overhead
  - Very regular



BOSCO performance

- BOSCO submissions
  - Work better at steady state (when some glidein are already running)
  - Work better with a Condor back-end
    - ○ 50% overhead over PBS, plus 250 overhead
    - ○ 500 sec overhead over Condor
    - ○ HCondor warm start submission outperforms direct HTCondor
  - "grid" universe submission problematic with short jobs

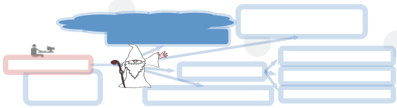


Accessing OSG resources

- It is possible to use opportunistic cycles on OSG
  - Using an existing gateway (OSG_XSEDE and HCC)
  - Using your own VO Frontend

- To access a gateway you need a HTCondor submit node (schedd) authenticating correctly and flocking to one of the gateways

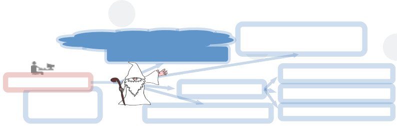
- https://www.opensciencegrid.org/bin/view/CampusGrids/InstallCondorFlockSubmit



OSG VO Frontend

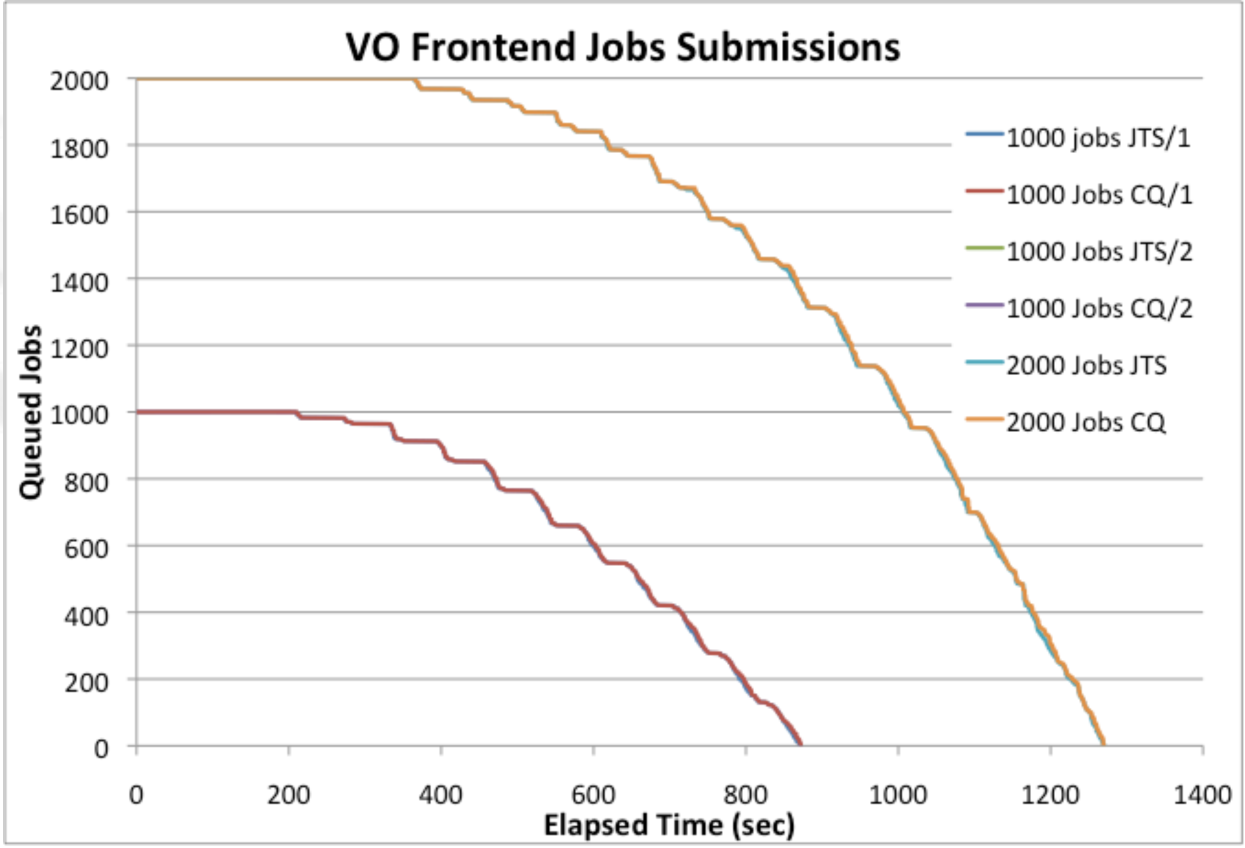
- To be more independent you can install a VO Frontend
  - x509 certificate for host and user (in valid OSG VO)
  - Request access to the GlidinWMS factory (GOC or UCSD)
  - Install the VO frontend
  - Operate the VO frontend (keep the certificate up-to-date)
  - Campaign to provide opportunistic use to your VO
  - You have your own queue
  - Decide on which OSG resources to run on

- References
  - http://ui-gwms.uchicago.edu/vofrontend/monitor/frontendGroupGraphStatusNow.html
  - https://twiki.grid.iu.edu/bin/view/Documentation/Release3/InstallGlideinWMSFrontend



Submission test on VO Frontend

- Higher scalability because the obtained opportunistic resources are more than available local resources (no guarantee)
- Longer startup time



Conclusions

- OSG is providing components and recipes that can be used to put together a Campus Infrastructure based on HTCondor. Other components and recipes come from the CIC.
- BOSCO allows to pool non HTCondor resources
  - Easy to administer
  - Performance overhead is limited (negligible with HTCondor, 50% with PBS)
  - Scales at least to O(1000) jobs and O(100) resources
- VO Frontend allows to reach OSG resources
  - Scales well to at least O(1000) jobs
  - Access to many opportunistic resources

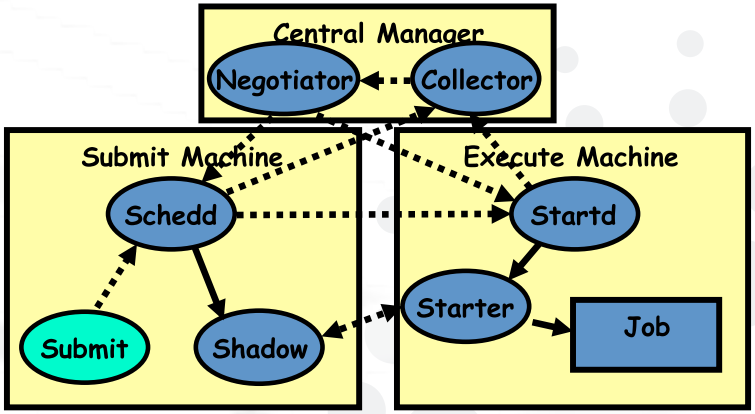
Thank you - Acknowledgments

- BOSCO Team:
  Derek Weitzel, Marco Mambelli (marco@hep.uchicago.edu), Jaime Frey, Dan Fraser, Brooklin Gore, Miha Ahronovitz
- UC3 and OSG ITB teams:
  Lincoln Bryant, David Champion, Rob Gardner, Suchandra Thapa
- Support from the National Science Foundation (Open Science Grid, NSF grant PHY 1148698) and the Computation Institute at the University of Chicago.

- BOSCO 1.1 References:
  - http://tinyurl.com/osgbosco
  - http://bosco.opensciencegrid.org/
  - https://twiki.grid.iu.edu/bin/view/CampusGrids/BoscoMultiUser
- Suggest features for BOSCO 1.3 @:
  - bosco-discuss@opensciencegrid.org

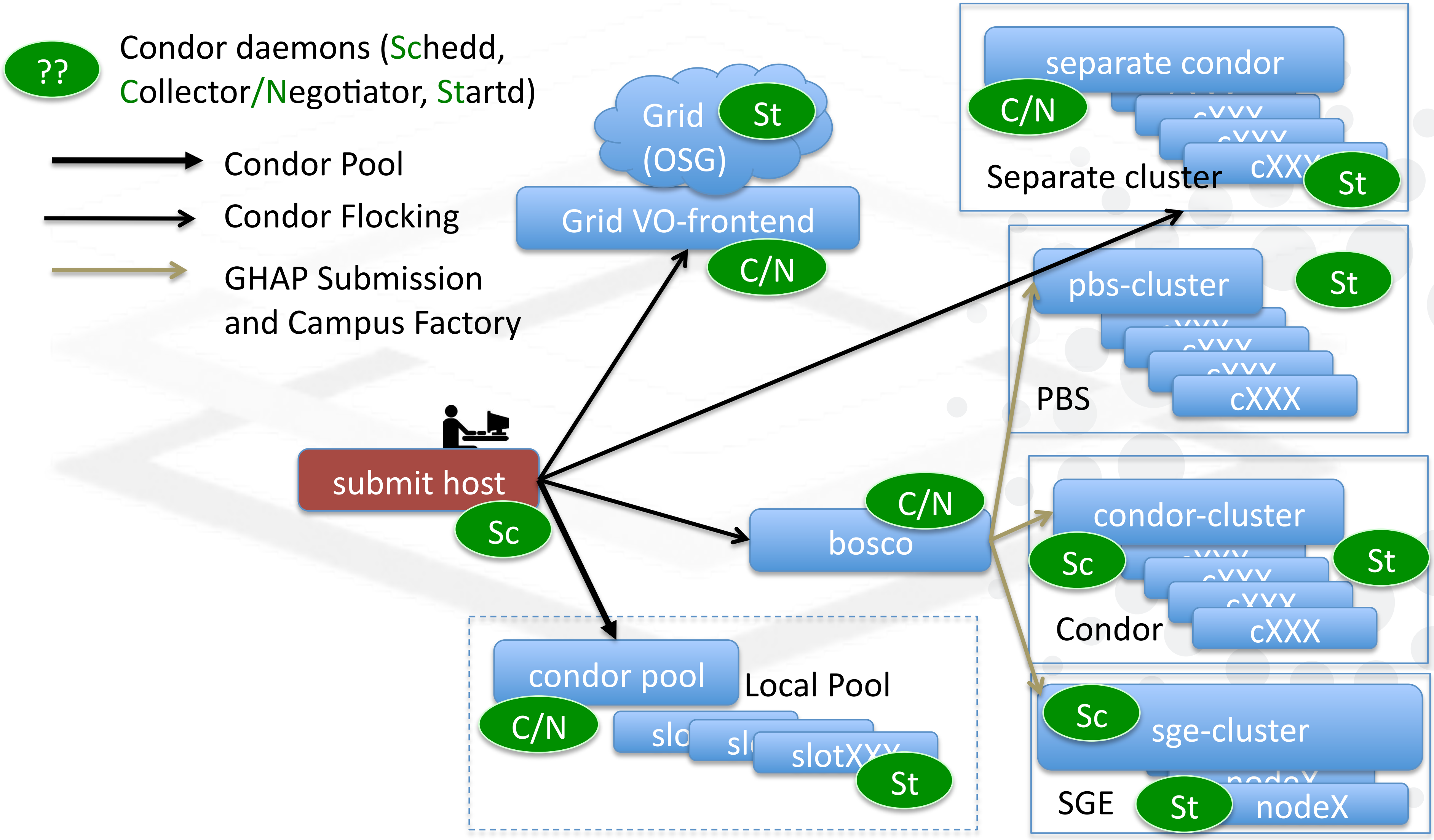
Extra Slides

HTCondor components overview



- Schedd and Shadow
  - represents resource requests (a submit machine) and shadows jobs
- Collector
  - collects all the information about the status of an HTCondor pool. (ClassAd)
- Negotiator
  - responsible for all the match making within the HTCondor system
- Startd and Starter
  - represents a resource, e.g. a machine capable of running jobs

Components in Campus Grid with BOSCO


??
Condor daemons (Schedd, Collector/Negotiator, Startd)
Condor Pool
Condor Flocking
GHAP Submission and Campus Factory
Grid (OSG)
St
Grid VO-frontend
C/N
separate condor
C/N
cXXX
cXXX
cXXX
St
Separate cluster
pbs-cluster
St
cXXX
cXXX
cXXX
cXXX
PBS
submit host
Sc
bosco
C/N
condor-cluster
Sc
St
cXXX
cXXX
cXXX
cXXX
Condor
condor pool
Local Pool
C/N
slotXXX
slotXXX
slotXXX
St
Sc
sge-cluster
nodeX
nodeX
SGE
St