Releasing the HTCondor-CE into the Wild

Brian Bockelman
HEPiX Fall 2014 Workshop

Trouble in CE land?

- In 2012, the OSG Executive Team requested we do a risk analysis of the components of the software stack. For each piece of software,
    - What is the health of the project? Is it actively being developed & new features added?
    - Who else uses the software?
    - Does an equivalent piece of software exist?
- One finding was that OSG is unique in its use of GRAM at scale and there was relatively few GRAM experts available.
    - We were asked to investigate alternates.

Actions Considered

- **Do nothing**: It takes significant effort to switch platforms; we could conclude that the disadvantages of GRAM didn't outweigh the costs.

- **Adopt CREAM**: Obvious advantages in synchronizing with EGI.

  - *In retrospect*, we failed examine a few other possible gatekeepers closely enough (ARC, Unicore).

- **Adopt HTCondor**: Several features were coming online to make this a viable alternative.

What does a gatekeeper do?

- **Remote access**: Provide a network-exposed service that remote clients can interact with.

- **Authentication and authorization**: Provides mechanism whereby clients can be identified and mapped to appropriate actions.

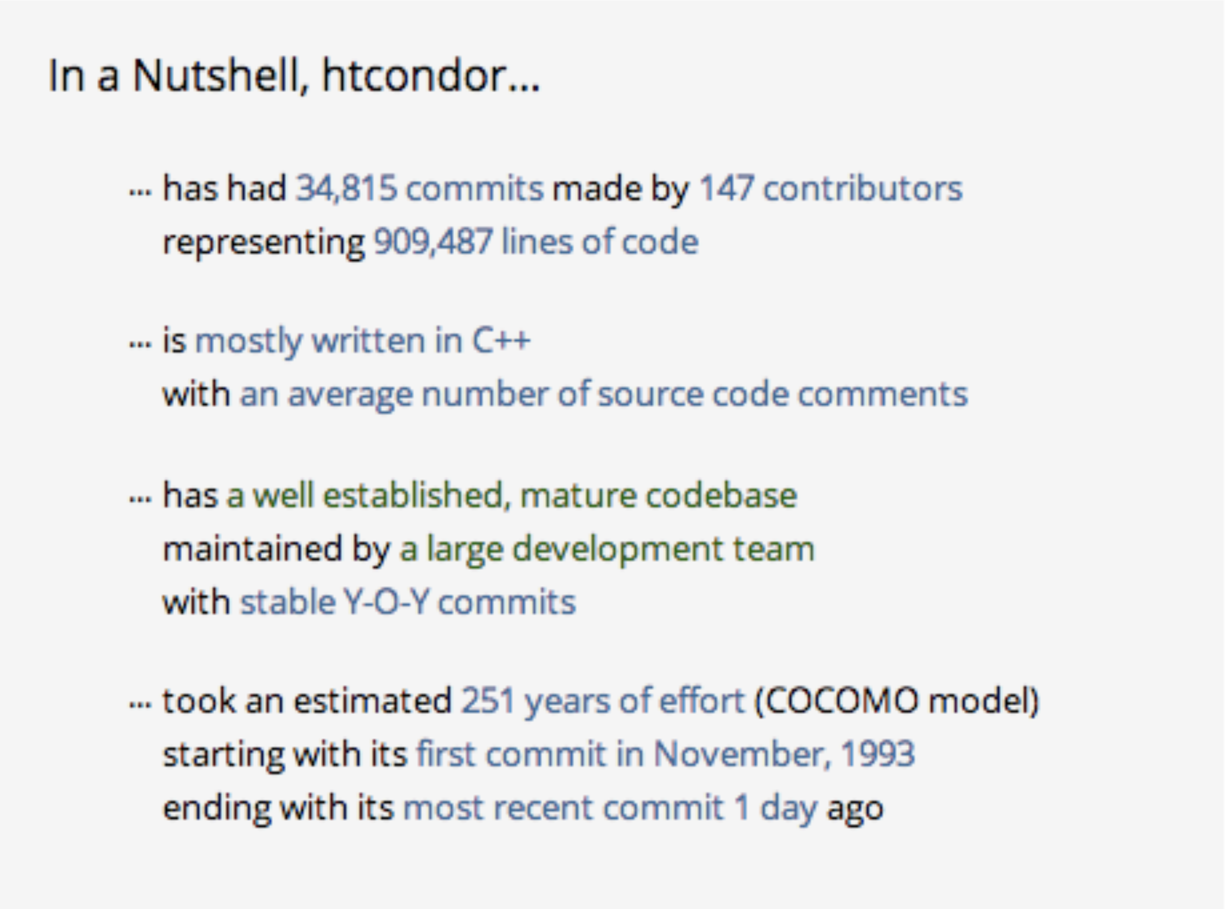
- **Resource allocation**: The gatekeeper accepts an abstract description of a resource to allocate and actualizes the resource request within the local environment.

  - Note I tweak this definition to fit the “pilot-based” world we live in.

Why HTCondor?

- Close working relationship between the OSG and HTCondor teams. We already use HTCondor throughout OSG Software, so it was the only choice that allowed us to ***reduce*** our number of external software providers.

- It's software with a long, long track record. It's 30th birthday was celebrated this year. However, despite its age, it still has a vibrant development community. Statistics from openhub.net:



In a Nutshell, htcondor...

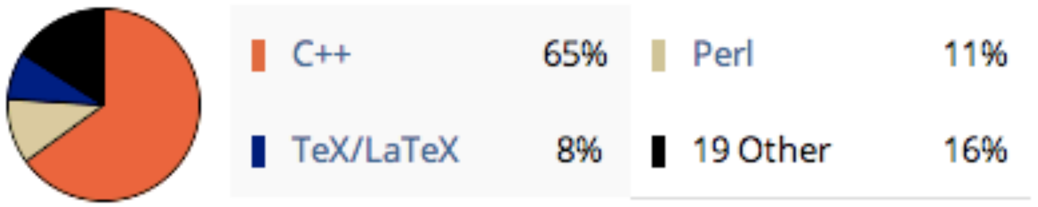
... has had 34,815 commits made by 147 contributors
representing 909,487 lines of code

... is mostly written in C++
with an average number of source code comments

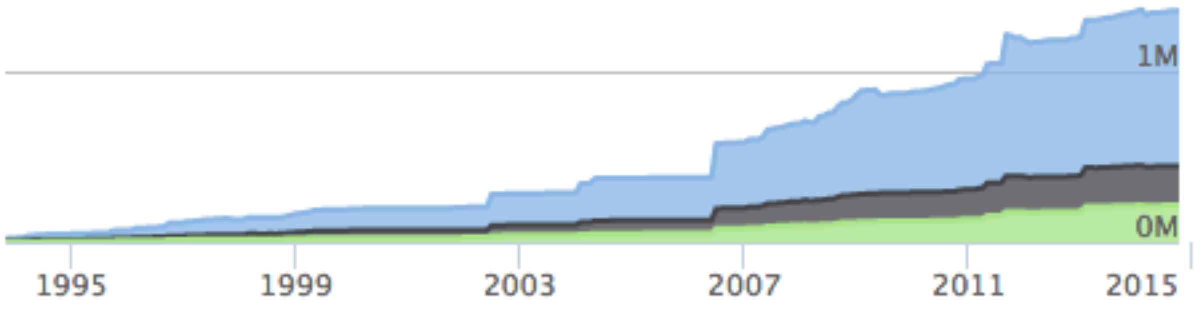
... has a well established, mature codebase
maintained by a large development team
with stable Y-O-Y commits

... took an estimated 251 years of effort (COCOMO model)
starting with its first commit in November, 1993
ending with its most recent commit 1 day ago

Languages



Lines of Code



HTCondor through the ages

1998

2014

Why HTCondor?

- HTCondor provided nearly all required gatekeeper functionality.
  - We put the pieces together, but follow the rule that the HTCondor-CE is ***just a special configuration of HTCondor***.
- In parallel, OSG and HTCondor teams were already working on another gatekeeper technology - BOSCO - that only requires SSH access.
  - HTCondor itself initiates the SSH connection and pipes commands to the local batch system.
  - The BOSCO wrapper helps with the staging of the HTCondor executables and configuration details.

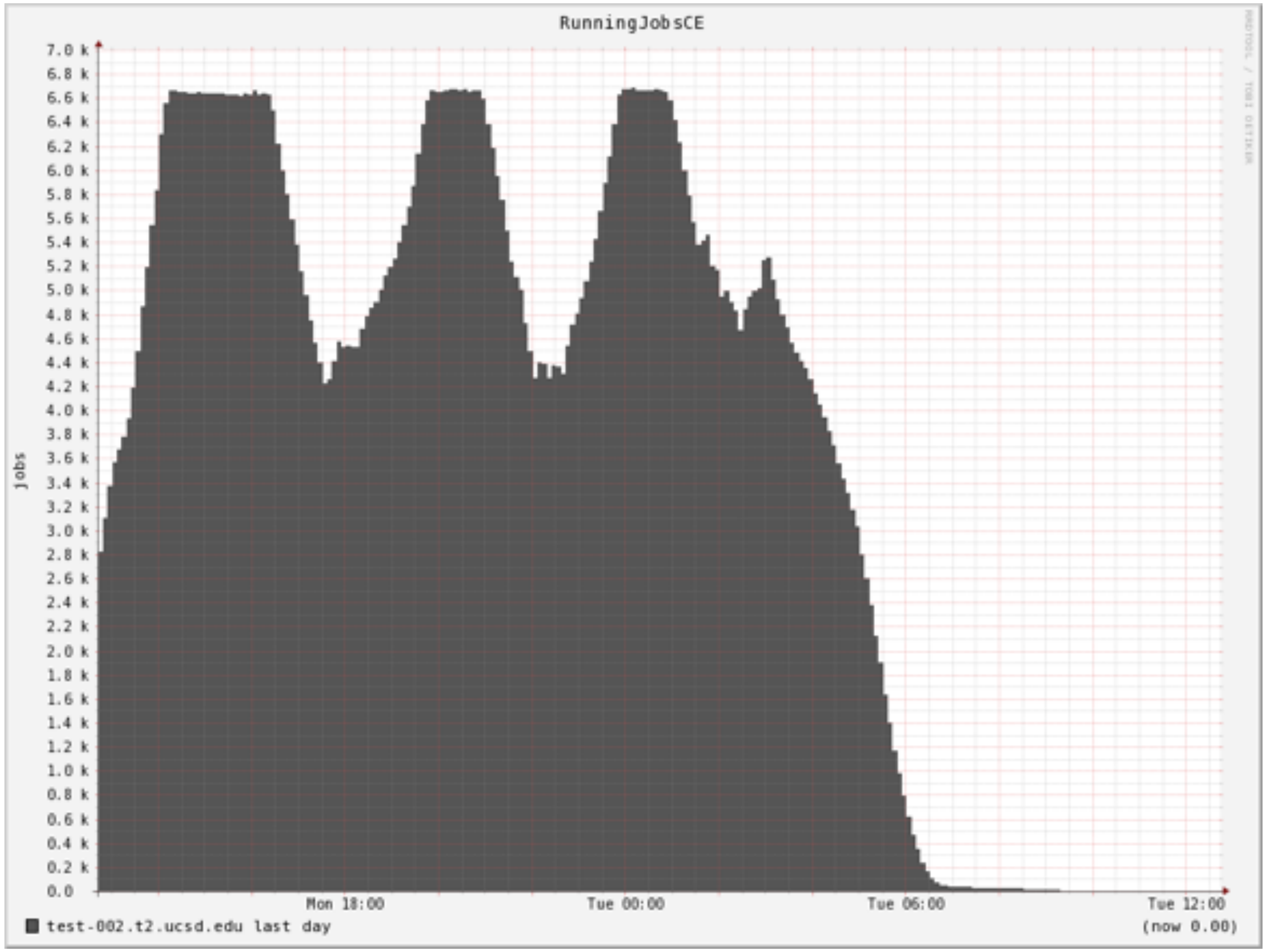
Introducing HTCondor-CE

- HTCondor-CE provides:
  - **Remote access**: Based on the internal CEDAR protocol.
  - **Authentication and Authorization**: Based on Globus libraries for GSI and authorization callout.
  - **Resource allocation**: Grid jobs are taken and transformed to local jobs using the JobRouter component.
    - Any software HTCondor can interact with is a potential backend. This includes EC2, OpenStack, or even another HTCondor-CE!

Tiny Details

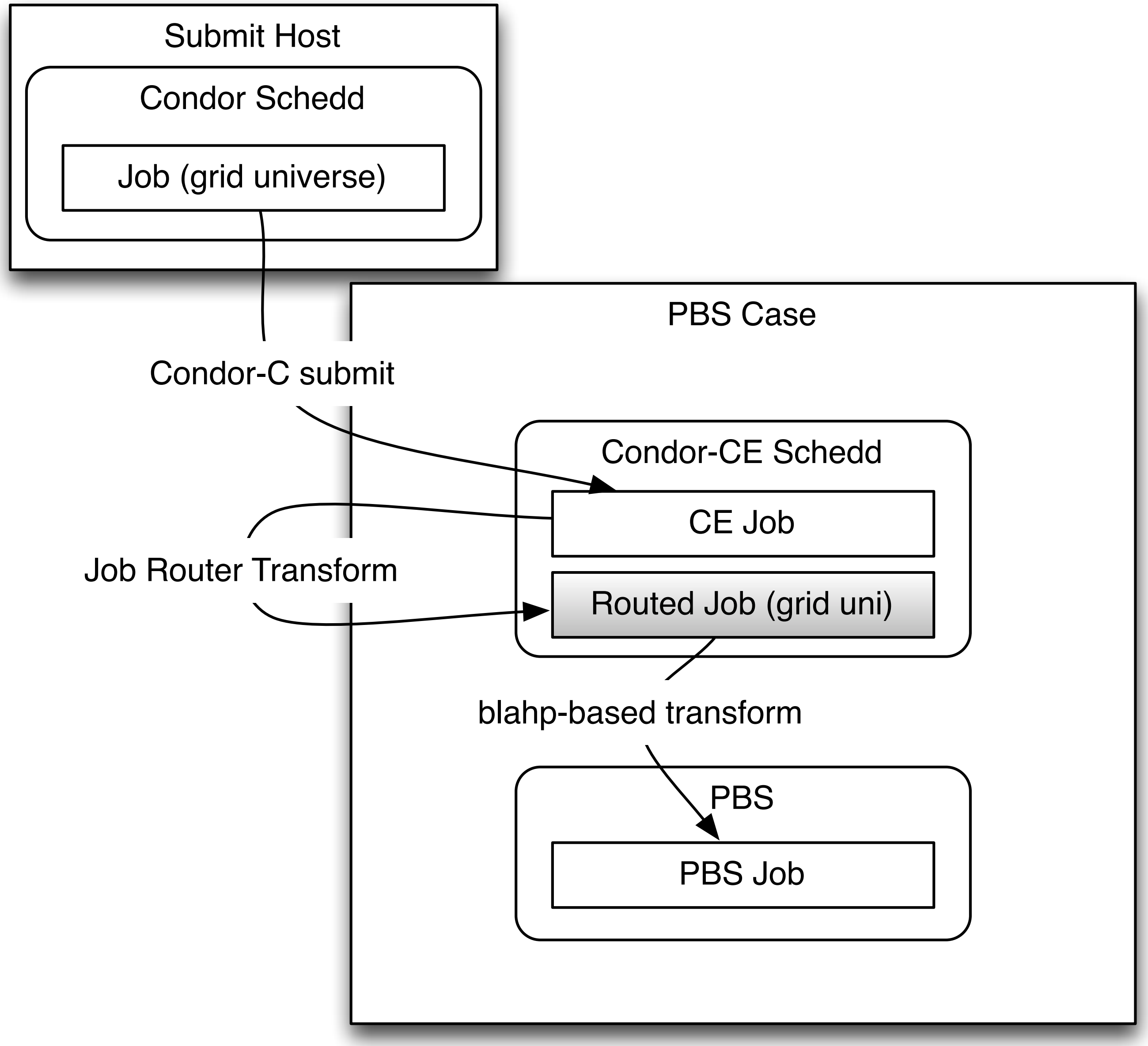
- We use the condor_shared_port to simplify firewall configuration - the CE needs two incoming ports (future versions will reduce this to one port).
    - Submitters no longer need an open port!
- Scalability - original tests done in 2012; just started a new round of tests on the latest HTCondor:
    - Sustained submit rates are about 40% better than GRAM (about 1.4 jobs / s); peak submit rates are about 5 jobs / s.
    - Currently peaks at about 7K running jobs - we think this is an HTCondor configuration issue and we should be fine up to 20K jobs / CE.
    - We're only about 3 days into the tests; more firm numbers will be presented at next HEPiX.
- There is much more visibility into the internals of the system - unlike GRAM, we can do "condor_ce_q" to see the grid jobs! All the other "condor_*" tools are still useful.

Example scale test run

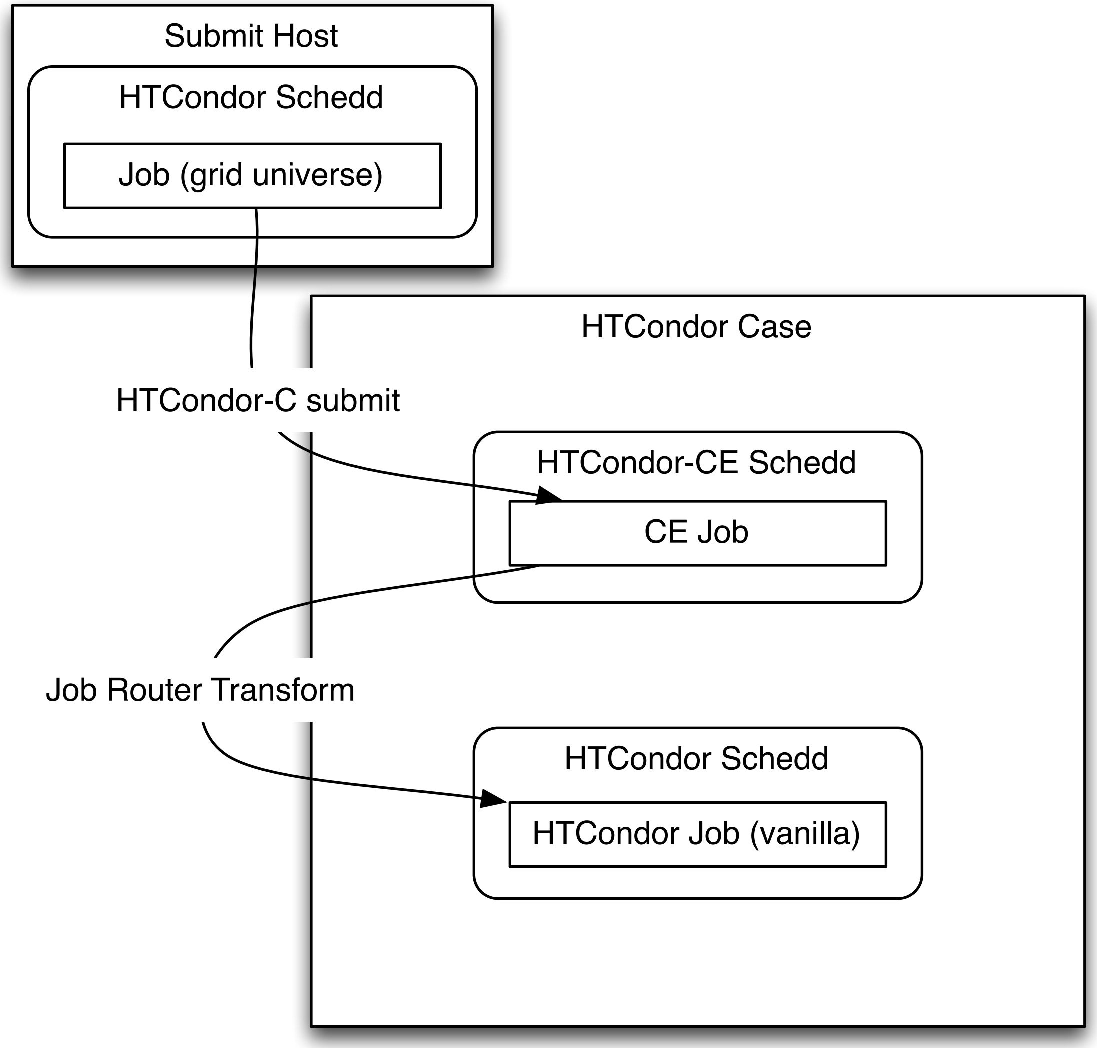


Example only - final numbers will come later.

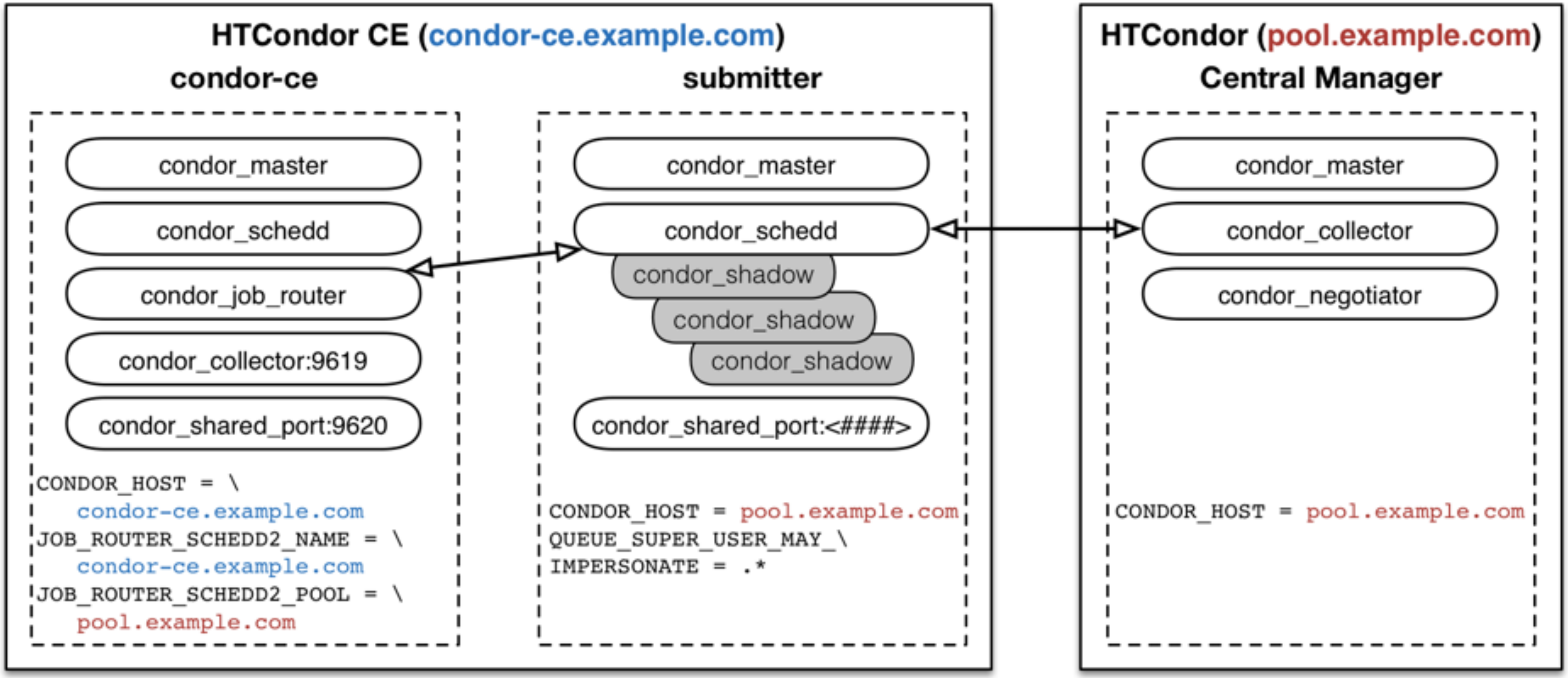
Routing Jobs



Special Case: HTCondor site



The Gory Details - HTCondor site



HTCondor CE (condor-ce.example.com)

condor-ce

- condor_master
- condor_schedd
- condor_job_router
- condor_collector:9619
- condor_shared_port:9620

```
CONDOR_HOST = \
   condor-ce.example.com
JOB_ROUTER_SCHEDD2_NAME = \
   condor-ce.example.com
JOB_ROUTER_SCHEDD2_POOL = \
   pool.example.com
```

submitter

- condor_master
- condor_schedd
- condor_shadow
- condor_shadow
- condor_shadow
- condor_shared_port:<####>

```
CONDOR_HOST = pool.example.com
QUEUE_SUPER_USER_MAY_\
IMPERSONATE = .*
```

HTCondor (pool.example.com)

Central Manager

- condor_master
- condor_collector
- condor_negotiator

```
CONDOR_HOST = pool.example.com
```

The Job Router

- A key technology is the **Job Router**, which creates a copy of the job and transform it according to a set of rules.
    - Each set of rules, or route, is specified as a declarative ClassAd.
    - Previously (GRAM), job transformations were specified in an imperative language (perl). The Job Router includes an “hook” which allows the sysadmin to specify a script in any language.
- **NEW PHILOSOPHY**: The pilot describes the resources it needs and the site implementation details are hidden by the JobRouter.
    - Sites have the *option* of exposing internal configurations, but we’d like to encourage VOs to get to “site-independent pilot submission” - only the endpoint name is different!

Example Route

```
JOB_ROUTER_ENTRIES = \
  [ \
    GridResource = "batch pbs"; \
    TargetUniverse = 9; \
    name = "Local_PBS_cms"; \
    set_remote_queue = "cms"; \
    Requirements = target.x509UserProxyVOName =?= "cms"; \
  ] \
  [ \
    GridResource = "batch pbs"; \
    TargetUniverse = 9; \
    name = "Local_PBS_other"; \
    set_remote_queue = "other"; \
    Requirements = target.x509UserProxyVOName =!= "cms"; \
  ]
```

More recipes available at:
https://twiki.grid.iu.edu/bin/view/Documentation/Release3/JobRouterRecipes

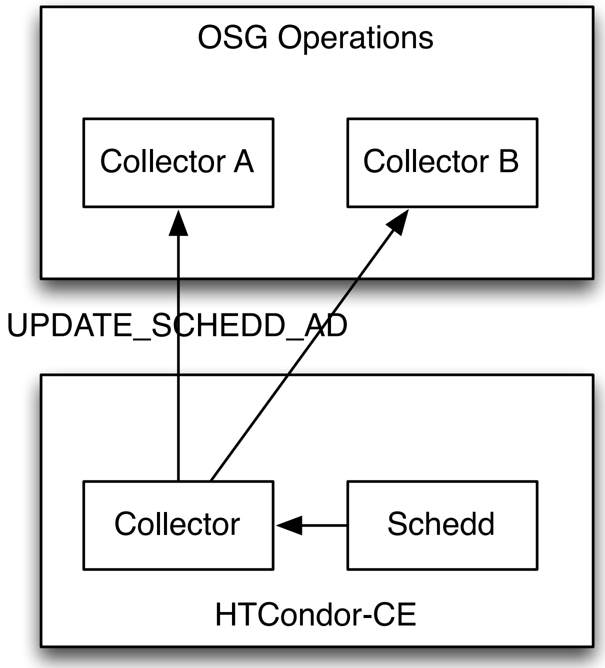
No More Site Details!

- What we *don’t* want is the following route and the instructions to the pilot factory of “please set CMS analysis pilots to queue ‘cms’”.

```
JOB_ROUTER_ENTRIES = \
  [ \
    GridResource = "batch pbs"; \
    TargetUniverse = 9; \
    name = "Local_PBS_cms"; \
 ]
```

Information Services

- For information services, we had two goals:
  - No new software - again, just a special configuration of HTCondor.
  - Fundamentally rethink what information is advertised - only provide the *minimal* amount needed for provisioning.
    - Don’t advertise 5 attributes if 4 would do!
- We aim for a *provisioning information system* - the only client is the pilot factory, not accounting, not storage, not monitoring.



Information Services

- Each daemon in HTCondor keeps an *ad* in the collector. This contains all the pertinent information about the daemon in the system.
    - We complement the HTCondor schedd ad (which is the core component of the CE) with CE-specific information.
    - The ad is then forwarded from the CE collector to the GOC, authenticated by GSI.
    - Currently, the query tool is condor_status or the python bindings.
- First version of the information system is released today! Target was to only provide enough information for a factory to find the CE.
    - In fact, we're considering banning querying by un-authenticated users to prevent other use cases from developing.
- Second phase will allow sites to advertise relevant policies for custom pilots (multicore, high-memory, VO-specific transforms, etc). Target is December 2014.

CE Deploy

- There are currently about 10-15 CEs deployed.
  - WLCG sites are still waiting on SAMv3 improvements for to turn off GRAM.
- We hope to default all new installs to HTCondor-CE in the next OSG release.
- The CE is already integrated with the accounting system, monitoring system, etc - although encountered a few bugs along the rollout.

Where are we going?

- **Sandbox management** - HTCondor has rudimentary support to limit the volume of file transferred per job. We're looking into how HTCondor might aggregate sandbox limits.
- **Alternate security mechanisms** - The CEDAR protocol allows the client/server to negotiate the security protocol; this gives sites the freedom to use different mechanisms (kerberos, shared password, etc).
- **Cloud / VM provisioning** - As the CE can route to any HTCondor backend, we believe this will be one mechanism to wrap some "grid-like" mechanisms (auth, queueing) in front of EC2-like resources.
- **Complex policies** - The sky is the limit! We are curious to see how sites will chain together routes, multiple CEs, implement complex routing policies.
    - Example: "overflow pilots to another cluster if the local cluster has been under-pledge for 24 hours".
    - Example: the Fermigrid load-balancing across several internal clusters.
- **Monitoring** - We're looking to find reasonable "out of the box" monitoring that is *not* specific to the CE - just HTCondor.

Lessons Learned

- After a decade of Globus GRAM, we significantly underestimated how poor the OSG's GRAM documentation was.
    - Basically, sysadmins had memorized all the pieces and steps.
    - This caused major trouble when we replaced GRAM with HTCondor-CE but didn't do major improvements to the docs.
- Even OSG's support staff didn't know what documentation they needed — they had basically memorized all the GRAM failure modes!
- **Lessons learned**: Documentation is key and the development team often doesn't know what is missing.

Lessons Learned

- JobRouter: Many sysadmins struggled with this component.
    - While we firmly believe it's a better model, changing from perl (imperative) to ClassAds (declarative) for transforms is a huge mental change for the admins.
        - Debugging is tricky - you need to pull information from several log files, logging lines may be missing.
    - We gave this feedback to the HTCondor team and they have been working hard to remove sharp lessons.
        - Several of the changes help prevent silly configuration errors.
- **LESSON**: Almost none of these issues were predicted by the developers; get the product in the hands of friendly testers ASAP. You need friends who are willing to eat the dog food.

Lessons Learned

- BLAHP:
  - blahp is used by HTCondor to talk to other batch systems.
  - Shared component with CREAM, but we worry about diverging use cases. Fundamentally, we don’t believe in (only) tailing log files!
  - The blahp component must cover a large diversity of site configurations. Validation has been very slow outside PBS.
- **Lesson learned**: Even “common components” require care and feeding. OSG needs to grow expertise in LSF and SGE.

Lessons Learned

- Collaborations:
  - We had several meetings with our stakeholders about requirements.
  - However, several new required features were requested *after* the initial releases.
- **Lesson learned**: Talk, talk, talk to your users. Unfortunately, the users don't know what features they need - and you probably aren't talking to the right users!
- **Lesson learned**: External dependencies can play havoc with the release schedule, especially if there are systems managed by non-stakeholders.

A vision of the future

- HTCondor-CE is just one of several technologies OSG is investing in. However, it fits into an overall vision.
- OSG will provide an increasingly **homogeneous execution environment** built from increasingly **heterogeneous resources**.
    - Homogeneous execution environment: software distribution (CVMFS), remote data access (HTTP, Xrootd), and job execution (PanDA, HTCondor).
    - Heterogeneous resource acquisition: HTCondor-CE, GRAM, SSH+local submit (BOSCO), EC2-like.

Questions?