Interfacing HTCondor-CE with OpenStack

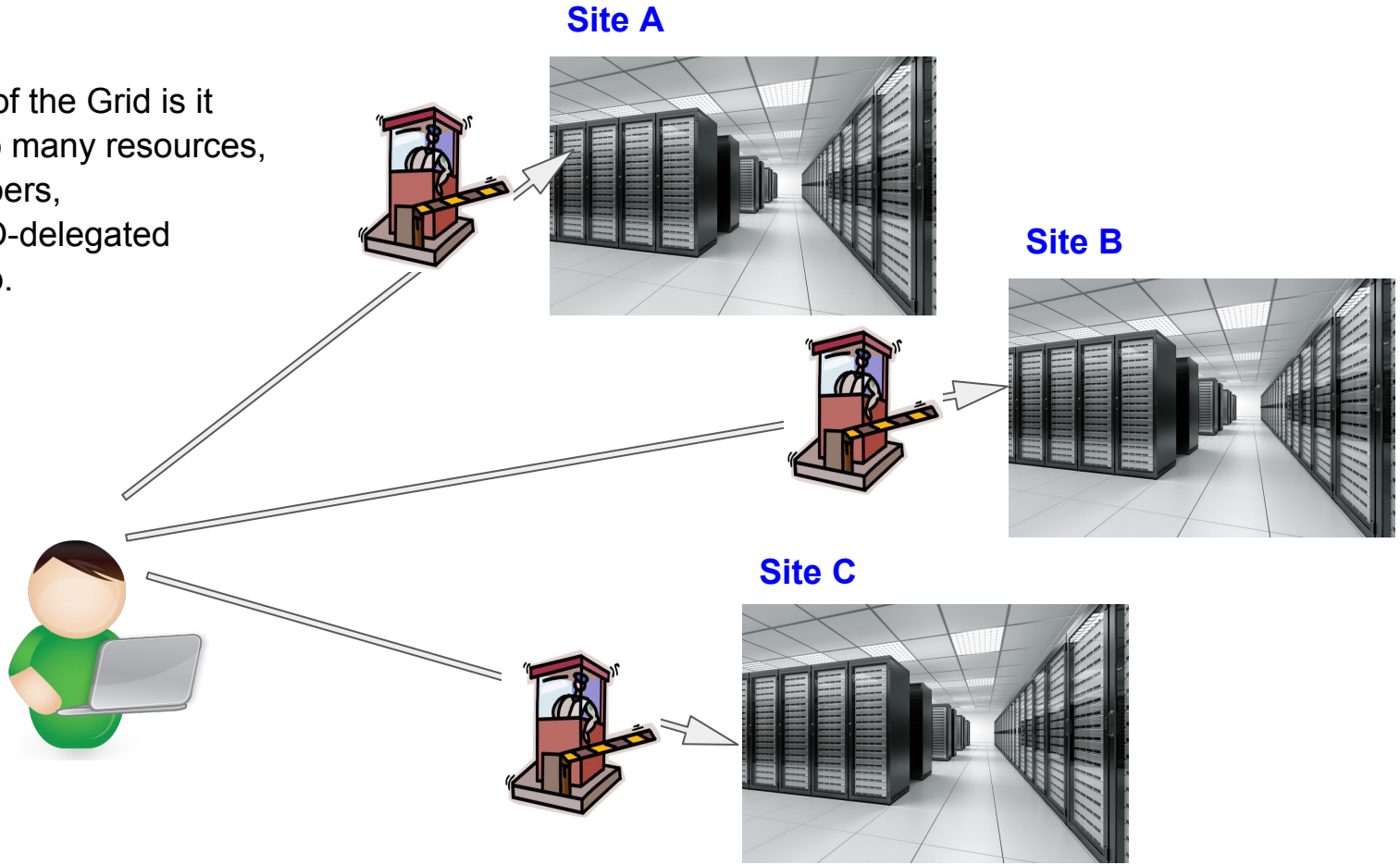
Brian Bockelman <bbockelm@cse.unl.edu>
Jose Caballero <jcaballero@bnl.gov>
John Hover <jhover@bnl.gov>

as part of the Open Science Grid Technology Investigation team

CHEP 2016, San Francisco

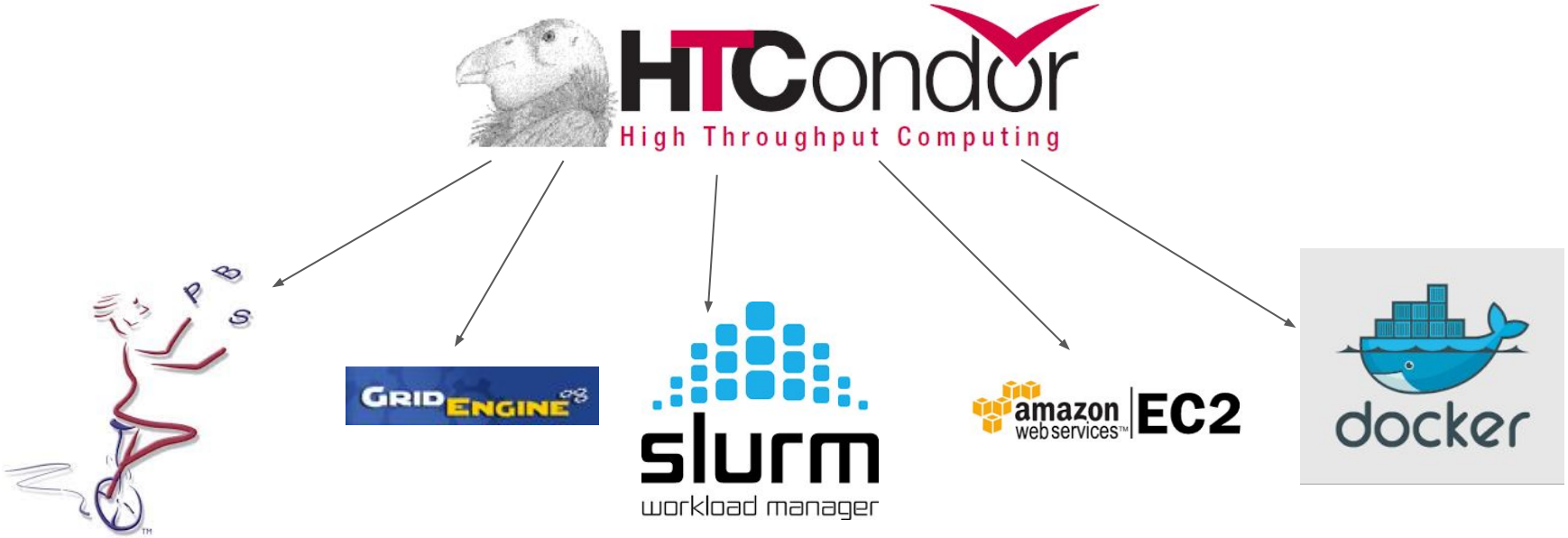
Motivation

The advantage of the Grid is it allows access to many resources, via the gatekeepers, thanks to the VO-delegated trust relationship.



Motivation

One of the most common CEs is the HTCondor-CE.
It leverages the [many] capabilities from HTCondor, including the ability to talk or route the jobs to a different batch system/platform.

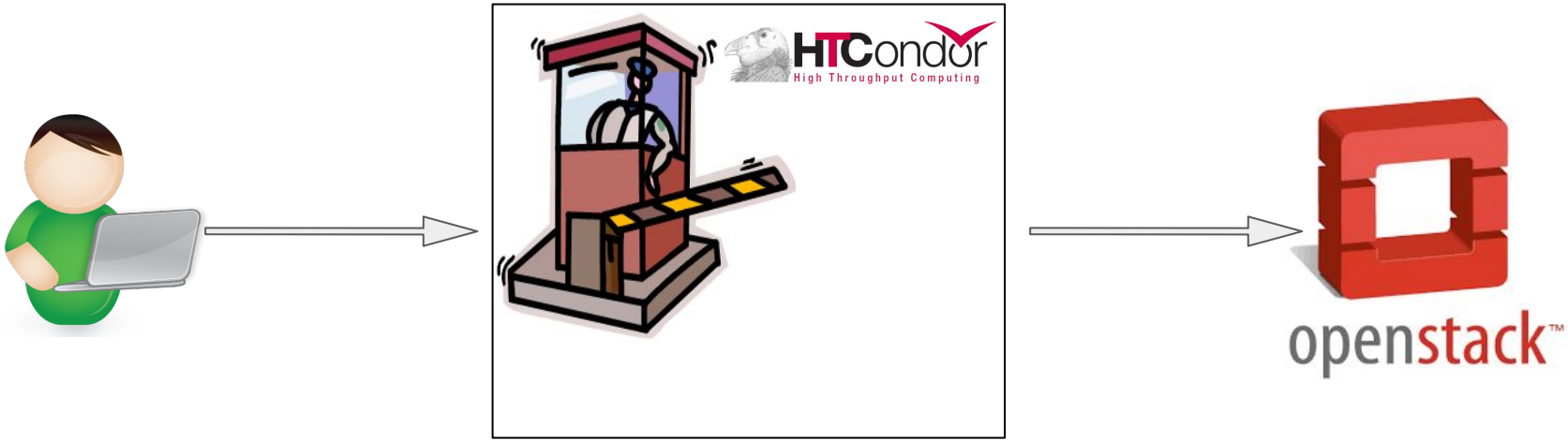


Motivation

Separate from the GRID world, a lot of alternatives solutions have been developed in the land of virtualization:

Motivation

In this work we try to join both worlds, Grid and Cloud, and allow users to interact with OpenStack via the HTCondor-CE.



Scenarios

- The user does not know it. Jobs are routed to the OpenStack cluster when they have requirements that can not be satisfied by the regular farm behind the CE.
  Common use case: job requires a different version of the Operative System.

- The user knows there is an OpenStack cluster, and desires to boot a VM and get ssh access into it.

In both cases, there are different ways to pick the VM image to be instantiated:

- Programmatically, based on job requirements.

- The user requests directly the image/flavor to be booted.

- The user provides an URL with the image to be booted.

And all of that without requiring the users to have OpenStack credentials, OpenStack client installed or to know how to interact with the services.

HTCondor Job Router

Def: The HTCondor Job Router is an add-on to the *condor_schedd* that transforms jobs from one type into another according to a configurable policy.

Job Router Hooks (*): arbitrary code that can be executed at some points of the job life cycle.

Translate	responsible for doing the transformation of the job and configuring any resources that are external to HTCondor if applicable.
Update	invoked to provide status on the specified routed job when the Job Router polls the status of routed jobs
Exit	invoked when the job has completed
Cleanup	invoked when the Job Router finishes managing the job

http://research.cs.wisc.edu/htcondor/manual/v8.4/4_4Hooks.html#SECTION005420000000000000000

(*) Example of layout in the backup slides.

Prototype setup

For this prototype, in order to make things simpler, we just use the following setup:

- grid ids mapped to a single UNIX account, which is also the unique OpenStack tenant account
- backend batch system is HTCondor as well
- VMs have HTCondor startd setup pre-configured to join the same pool
- the startd is set to stop after the first job has finished, so it does not pick more jobs

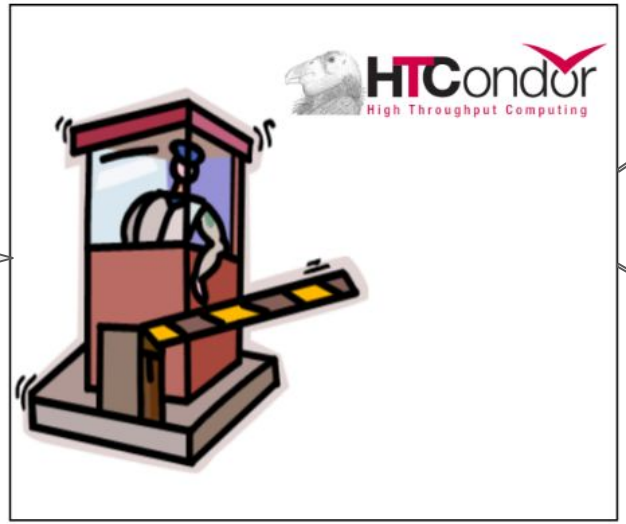
Also:

- tests have been performed one job at a time.
  Probably some improvements are needed to manage multiple jobs at the same time.

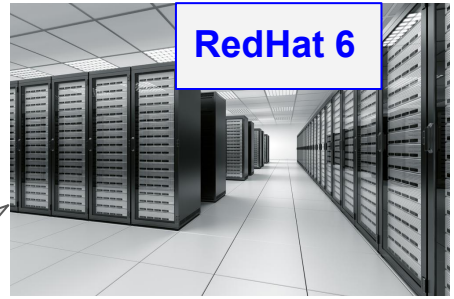
Scenario 1

Submit a job

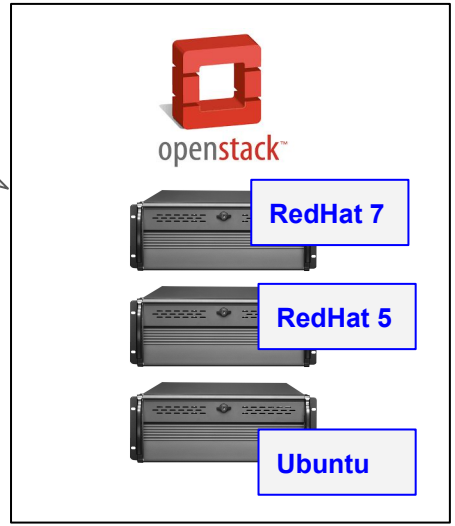
GRID Site



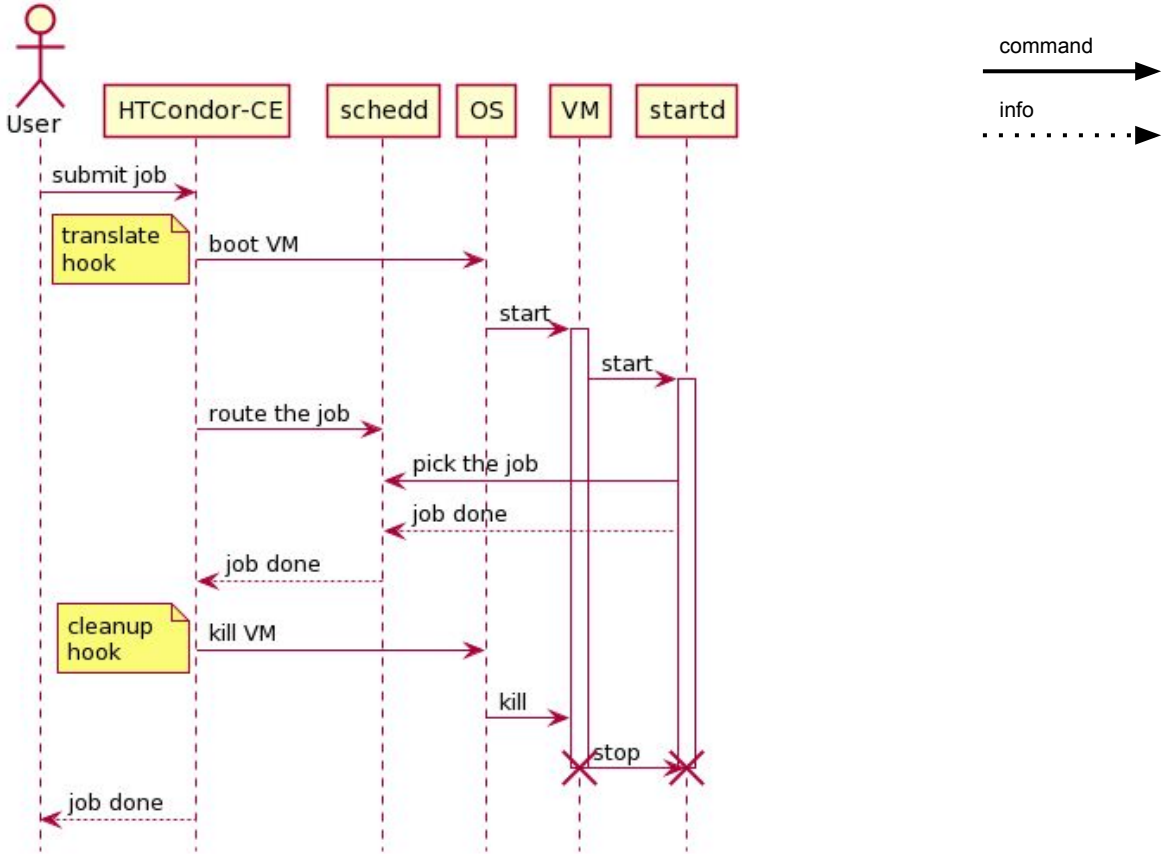
Run the job



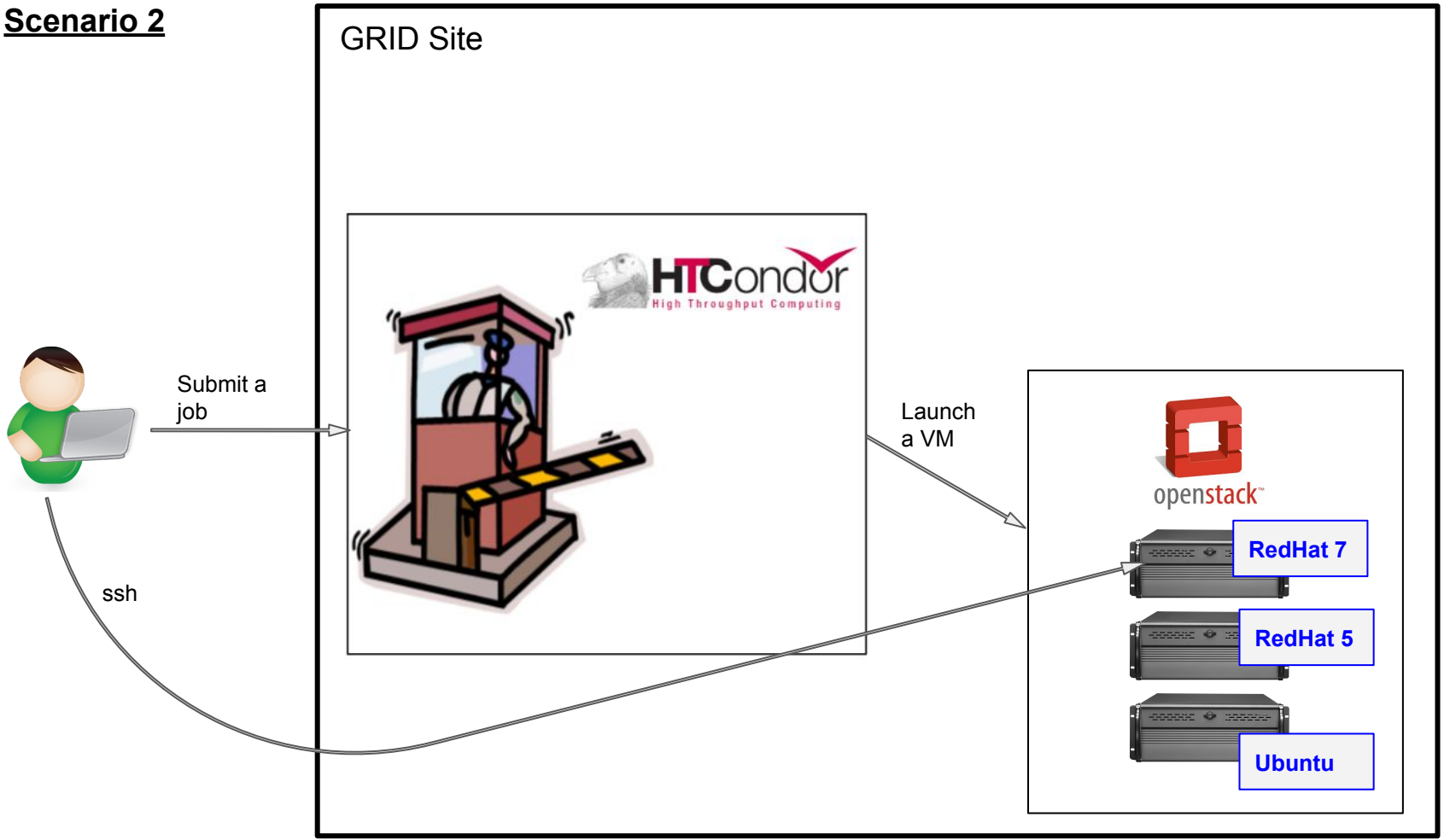
Launch VM and run the job



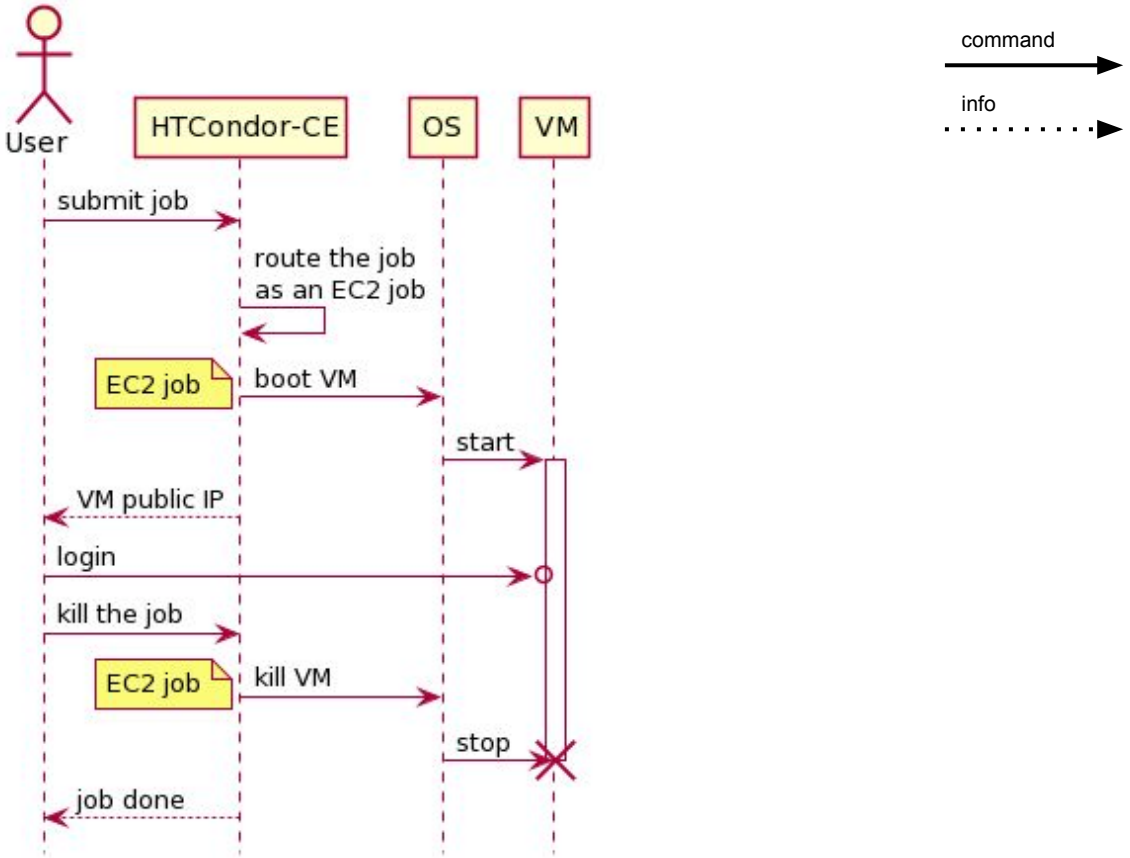
Scenario 1: implementation



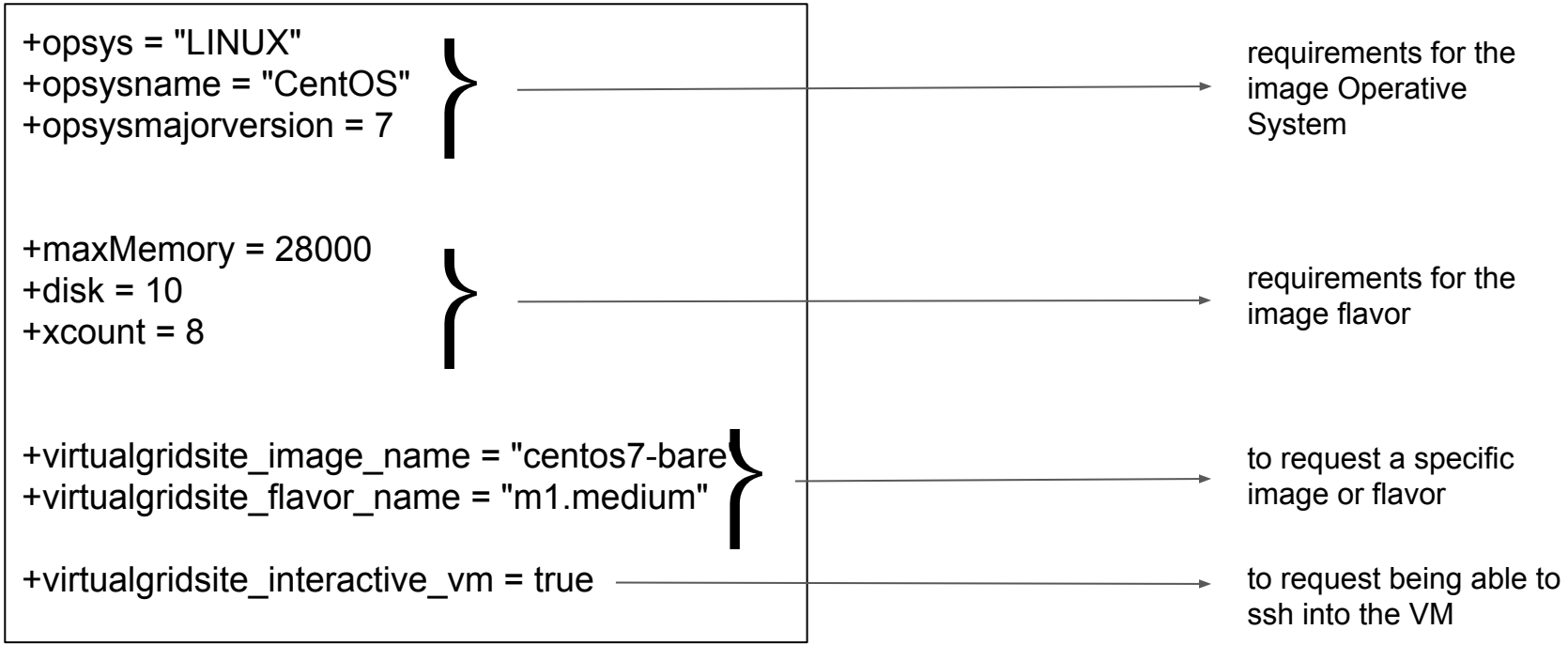
Scenario 2


GRID Site
HTCondor
High Throughput Computing
Submit a job
Launch a VM
ssh
openstack
RedHat 7
RedHat 5
Ubuntu

Scenario 2: implementation



ClassAds in the HTCondor submit file (*)



(*) some names are provisional.

ClassAds in the HTCondor submit file: matchmaking

Farm nodes conf file:

```
[type_1]
name = rh5
opsys = LINUX
opsysname = RedHat
opsysmajorversion = 5

[type_2]
name = rh6
opsys = LINUX
opsysname = RedHat
opsysmajorversion = 6
```

Available VM images conf file:

```
[centos7]
name = centos7-osg-condor
ami = ami-0000004b
opsys = LINUX
opsysname = CentOS
opsysmajorversion = 7
mode = batch, interactive
```

Available VM flavors conf file:

```
[m1.medium]
name = m1.medium
memory = 4096
xcount = 22
disksize = 40

[m1.xlarge.5000]
name = m1.xlarge.5000
memory = 16384
xcount = 8
disksize = 10
```

ClassAds in the HTCondor submit file: matchmaking

Requirements expressions for the HTCondor JobRouter are created combining a static file and programmatically based on the content of those 3 configuration files. Examples:

- for interactive jobs (static configuration):

```
Requirements = TARGET.virtualgridsite_interactive_vm isnt undefined
```

- for jobs that cannot be run (created by script):

```
Requirements = TARGET.virtualgridsite_interactive_vm is undefined &&
               TARGET.virtualgridsite_image_name is undefined &&
               TARGET.noreroute is undefined &&
               ( TARGET.opsysname != "RedHat" || TARGET.opsysmajorversion != "5" ) &&
               ( TARGET.opsysname != "RedHat" || TARGET.opsysmajorversion != "6" ) &&
               ( TARGET.opsysname != "CentOS" || TARGET.opsysmajorversion != "7" );
```

Custom image

The user can specify in the HTCondor submit file an URL with the VM image they want to run:

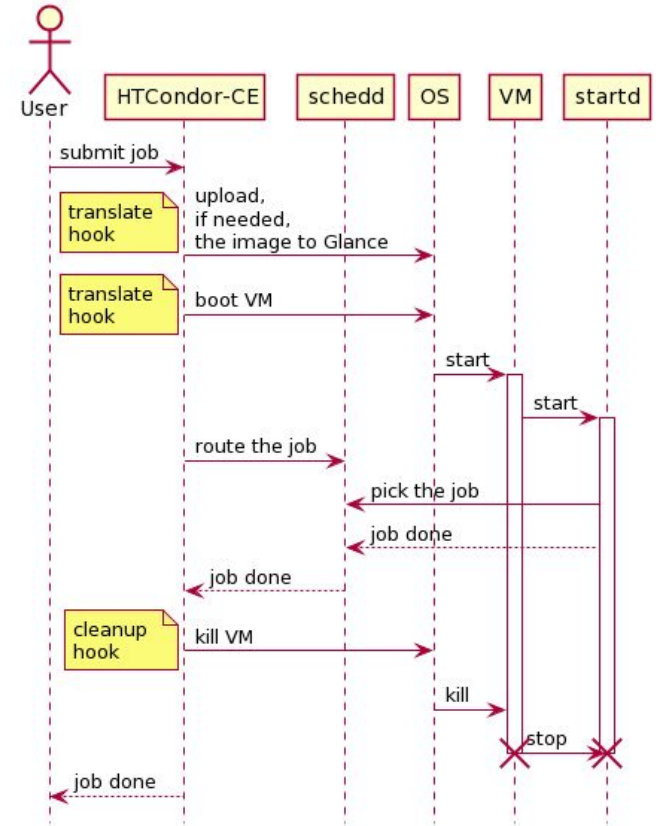
+virtualgridsite_url = <URL>

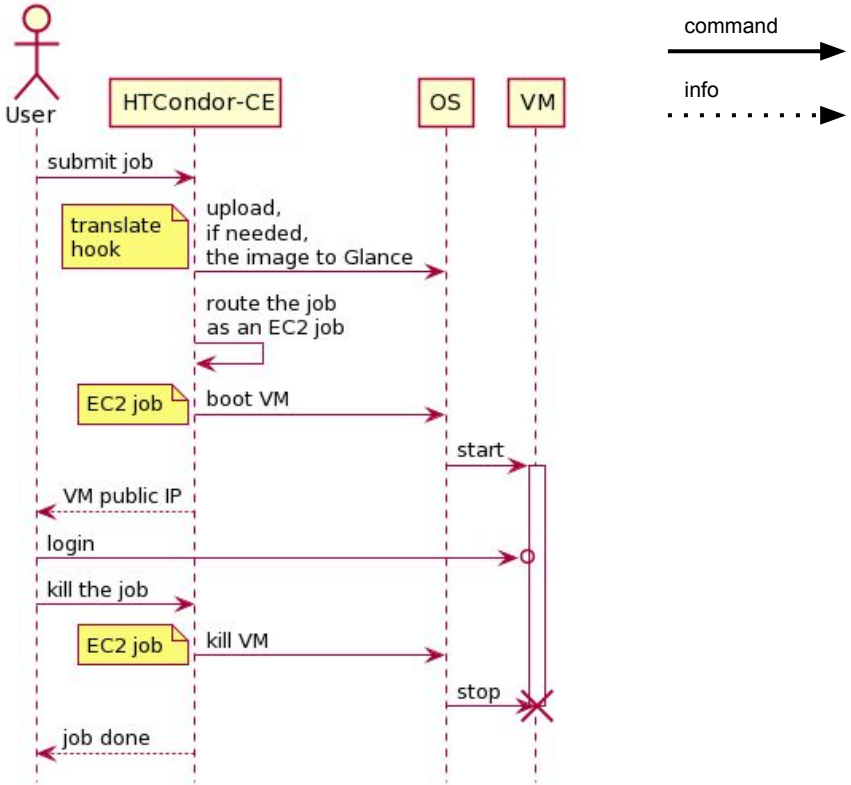
The first time, the file is downloaded from that URL, and then uploaded into Glance (the OpenStack images service).

http://docs.openstack.org/developer/glance/

The image is assigned an unique name based on the URL's hash and timestamp, so no need to repeat the process next time the user has the same request.

Custom image





Security

- The HTCondor-CE is configured to not allow jobs being forked at the CE host (*).
  Therefore, the user's job cannot read configuration files -including the one with OpenStack auth data-.

  The configuration files are read by the hooks code.
  Idea to improve the security is to make the hooks to call a root-owned daemon, which would be the only entity authorized to read them.

- With the current implementation users don't need OpenStack credentials, but they must have valid a X509 proxy.
  How to allow interaction with the HTCondor-CE with a security mechanism other that GSI is under investigation.

(*) See the backup slides.

Problems found

- If the VM instantiation fails, there is no clean way to terminate the job. A failed JobRouter hook makes the job to be re-routed again.

- Some of the ideas may require the cleanup hook to print out the job classad. We have found that that triggers a couple of hidden bugs.

- If there is no host or VM that can run the job, it is complicated to handle it. There is no clear way to "not route" it [(*)]

Working with the developers of the HTCondor team to improve the job life cycle when managed by the JobRouter.

- The standard HTCondor-CE installation comes with some restrictive setup in its configuration that needs to be overridden.

[(*)]Current solution in the backup slides.

Potential improvements and new features

- Instantiating VMs at EC2 (and/or other clouds), not only OpenStack.
- HTCondor startd configuration passed via "user data" instead of hardcoded.
- Integration with BOSCO-CE project.
- More complex job lifecycles:
    - elastic expansion of the cluster: instantiating a VM if a job has been idle too long
    - reusing the same VM for more than one job
- More than one OpenStack user (tenant), and therefore user quotas, share mechanisms, accounting and billing.
- Non GSI security requirements to submit jobs to the HTCondor-CE.
- The process calling the hooks being a system daemon:
    - configuration files are read only once
    - only root can read the configuration files

Backup
and
Technical slides

OpenStack cluster specs

- Openstack (Icehouse) instance
- 120 compute nodes (16 cores, 32GB RAM, 2 to 5 TB disks).
- 200TB Swift (Amazon S3-equivalent) *object store.*
  - Upload/download data from anywhere (BNL, external)
  - Good bandwidth from VMs.
- Usable for data analysis/processing, prototyping, software testing.
- EC2 API enabled

HTCondor-CE configuration:

Hooks:

JOB_ROUTER_HOOK_KEYWORD = NOVA

NOVA_HOOK_TRANSLATE_JOB = /usr/share/virtualgridsite/nova_hook_translate_job

NOVA_HOOK_UPDATE_JOB_INFO = /usr/libexec/nova_hook_update_job_info

NOVA_HOOK_JOB_EXIT = /usr/libexec/nova_hook_job_exit

NOVA_HOOK_JOB_CLEANUP = /usr/libexec/nova_hook_cleanup_job

HTCondor-CE configuration:

To prevent the end user from forking jobs on the CE host, where files with passwords and keys may be stored:

START_LOCAL_UNIVERSE = False

START_SCHEDULER_UNIVERSE = $(START_LOCAL_UNIVERSE)

HTCondor-CE configuration:

To generate the routing tables by code:

JOB_ROUTER_ENTRIES_CMD = /usr/lib/python2.6/site-packages/virtualgridsite/routes.py
JOB_ROUTER_ENTRIES_REFRESH = 600

HTCondor-CE configuration:

To get the value of the EC2 public IP mirrored into the source job classad:

NOVA_ATTRS_TO_COPY = EC2ElasticIP

HTCondor-CE configuration:

To override default configuration that enforces JobUniverse 1 or 5:

JOB_ROUTER_SOURCE_JOB_CONSTRAINT = (target.x509userproxysubject =!= UNDEFINED) && (target.x509UserProxyExpiration =!= UNDEFINED) && (time() < target.x509UserProxyExpiration)

HTCondor-CE configuration:

To prevent a job from running on the CE host:

START_LOCAL_UNIVERSE = False
START_SCHEDULER_UNIVERSE = $(START_LOCAL_UNIVERSE)

Routing table for jobs that cannot be routed:

```
[
  Name = "No_route";
  EditJobInPlace = true;
  set_noreroute = "True";
  Requirements = TARGET.noreroute is undefined && <node requirements>
  set_PeriodicRemove = ( JobStatus == 1 && ( time() - EnteredCurrentStatus ) > 10 );
  TargetUniverse = 5
]
```

Routing table for jobs that cannot be routed (ii):