Docker with HTCondor at FNAL

Anthony Tiradani

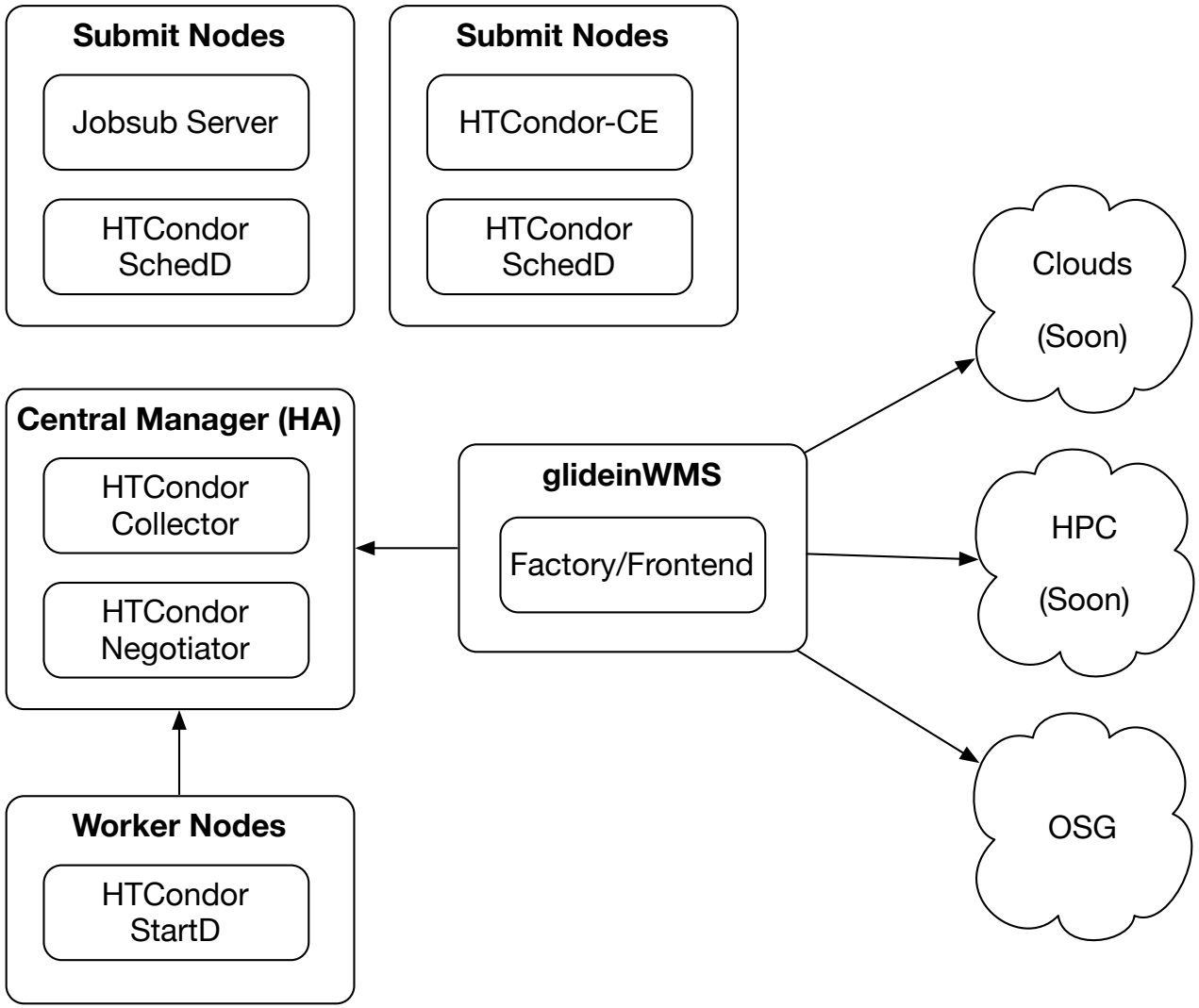
HTCondor Week 2018

May 2018

Fermi National Accelerator Lab

- Located in Batavia, IL (~45 miles/~60 km west of Chicago)
- Home of Scientific Linux
- Support 30+ experiments and projects, largest experiment (CMS) ~3000 users, largest project (DUNE) ~1000 users, both international
- Host the US Tier-1 facility for the CMS collaboration
- Computing Scale
  - 3 HTCondor clusters totaling ~45k cores
  - ~100PB Tape storage
  - ~58PB Disk

Cluster Diagram



Fermilab

Why Docker?

- Job Isolation
- OS portability
  - Remove linkage between worker node and job requirements
  - Allow experiments to pick supported OS's
- Environment management and reproducibility
  - Enable custom cgroup policies (without having to manage a "menu" of permutations)
  - Enable custom environments

Docker with HTCondor

A very basic setup of HTCondor with Docker is as follows:

- On the worker node set the following knob to the Docker executable on the machine:

```
# Setting the path to docker binary
DOCKER = /usr/bin/docker
```

- On the schedd side, a vanilla universe job has to add the following two classAds to the job submission file:

```
# To indicate that the job wants to run in Docker
+WantDocker = True
# To indicate the container image it wants to run
+DockerImage = "scientificlinux/sl:7"
```

Requirements and constraints at FNAL

List of constraints and requirements we had when we decided to migrate to Docker:

- The migration to Docker has to be transparent to the users
- We should be able to enforce container environments from the worker node side
  - OS
  - Environments variables
  - Images
  - Capabilities
  - Networking, etc.

Docker with HTCondor

To determine which startds are Docker capable (remember, we have pilots from OSG joining the pool), we add the following knob to the schedd config:

```
# To fetch FERMIHTC_DOCKER_CAPABLE attribute from the
# STARTD into the job
SYSTEM_JOB_MACHINE_ATTRS = $(SYSTEM_JOB_MACHINE_ATTRS) \
                           FERMIHTC_DOCKER_CAPABLE
```

We use the job machine attributes feature of HTCondor to fetch the classAd `FERMIHTC_DOCKER_CAPABLE` into the job classAds. This was necessary to make sure we do not try to invoke Docker on pilots requested at other grid sites

NOTE: **JOBS** control Docker functions in HTCondor (currently)

Docker with HTCondor

Fetching `FERMIHTC_DOCKER_CAPABLE` from the startd results in job classAd '`MachineAttrFERMIHTC_DOCKER_CAPABLE0`' for the most recent match the job had. We use this classAd to then determine the value of '`WantDocker`' by using job transforms:

```
# The transform policy to set relevant Docker classAds
JOB_TRANSFORM_NAMES = FERMI_HTC_Docker
JOB_TRANSFORM_FERMI_HTC_Docker = \
    [ \
    set_WantDocker = \
        isUndefined(MachineAttrFERMIHTC_DOCKER_CAPABLE0) \
        ? FALSE \
        : MachineAttrFERMIHTC_DOCKER_CAPABLE0; \
    set_DockerImage = "gpgrid-sl7:production"; \
    ]
```

Docker with HTCondor

So to summarize;

- A vanilla job is transparently converted into a “Docker” job using the schedd transforms when a user submits it
- ‘`WantDocker`’ is set as an expression that relies on the startd to tell the job if it supports Docker
- We explicitly set `FERMIHTC_DOCKER_CAPABLE` to False for pilots to prevent the use of Docker offsite
- The image of our choice is appended to the job classAds when the transform is applied
- Using the transform allows us to select multiple images based on, for example, the OS. An `ifthenelse()` block can be added to achieve this

Docker with HTCondor

As an extra measure, job submit requirements are added to the schedds to prevent users from overwriting the image in their job submission files:

```
# Preventing jobs from specifying their own docker images
SUBMIT_REQUIREMENT_NAMES = CustomDocker
SUBMIT_REQUIREMENT_CustomDocker = \
    (DockerImage=?="gpgrid-sl7:production")
SUBMIT_REQUIREMENT_CustomDocker_REASON = \
    "Custom Docker images are not supported. Please unset
DockerImage classAd"
```

Docker classAd attributes are also marked as ‘PROTECTED’ to prevent condor_qedits:

```
# Preventing anyone from editing the jobs for docker
# classAds
PROTECTED_JOB_ATTRS = WantDocker DockerImage
```

Docker with HTCondor

On the worker nodes, `FERMIHTC_DOCKER_CAPABLE` is added to indicate whether HTCondor should attempt to run the job inside Docker (as discussed on the previous slides):

```
# Set the attribute to 'True' if the worker is intended to
# run jobs inside Docker. Otherwise, 'False'.
# The attribute is fetched by the job when claim is
# activated to determine whether to set 'WantDocker' to
# true or false
FERMIHTC_DOCKER_CAPABLE = True
STARTD_ATTRS = $(STARTD_ATTRS) FERMIHTC_DOCKER_CAPABLE
```

Next, we set '`DOCKER`' in the configuration file such it doesn't point to a docker binary, but to our custom wrapper script:

```
# The default path is commented out to run Docker from
# a wrapper script. This allows us to make custom changes
# to the container if/when needed
DOCKER = /usr/local/libexec/condor-docker
```

Docker with HTCondor

The idea is to intercept the default arguments passed by HTCondor to Docker and make the necessary changes by appending Docker CLI flags to these arguments. For example*:

```
Apr 23 11:26:29 wnitb020 condor-docker: [condor-docker]: INFO - Received
condor arguments:/usr/local/libexec/condor-docker create --cpu-shares=100
--memory=512m --hostname fkhan-8347.4-wnitb020.fnal.gov --name
HTCJob8347_4_slot1_2_PID2231 -e BATCH_SYSTEM=HTCondor -e EXPERIMENT=mu2e -e
GRID_USER=fkhan --volume /cvmfs:/cvmfs:ro,slave gpgrid-sl7:production
./condor_exec.exe 750 750

Apr 23 11:26:29 wnitb020 condor-docker: [condor-docker]: INFO - Script
executed command:/usr/bin/docker create --cpu-shares=100 --name
HTCJob8347_4_slot1_2_PID2231 --hostname fkhan-8347-4-wnitb020.fnal.gov
--volume /cvmfs:/cvmfs:ro,slave --env BATCH_SYSTEM=HTCondor --env
EXPERIMENT=mu2e --env GRID_USER=fkhan --memory 512m --memory-swap 512m
--env FIFE_GLIDEIN_ToDie=$((`date +%s` + 346200)) --env UPS_OVERRIDE="-H
Linux64bit+2.6-2.12" --cap-add=DAC_OVERRIDE --cap-add=SETUID --cap-
add=SETGID --cap-add=SYS_ADMIN --cap-add=SYS_CHROOT --cap-add=SYS_PTRACE
--network host gpgrid-sl7:production ./condor_exec.exe 750 750
```

(*) The arguments are truncated for sanity

Docker with HTCondor

Using the HTCondor python bindings, we were able to avoid hardcoding values inside the python wrapper and instead created a few auxiliary parameters to dictate the behavior of the container on the worker node. For example,

```
# List of trusted Docker images
# Full name format: (<registry>/<user>/<repository>:<tag>)
# Privileged capabilities will only be passed to trusted
# images. This can be a space or comma separated list
FERMIHTC_DOCKER_TRUSTED_IMAGES = \
   'gpgrid-sl7:production', 'centos:7'
```

Docker with HTCondor

Similarly, another example of an auxiliary knob:

```
# Comma or space separated list of capabilities to be
# enabled inside the containers listed under
# FERMIHTC_DOCKER_TRUSTED_IMAGES
# List of all kernel capabilities can be seen here:
# http://man7.org/linux/man-pages/man7/capabilities.7.html
# If undefined, no capabilities are added to the
# containers
FERMIHTC_DOCKER_CAPABILITIES = DAC_OVERRIDE, SETUID, \
          SETGID, SYS_ADMIN, \
          SYS_CHROOT, SYS_PTRACE
```

Docker with HTCondor

A simple snippet from inside the wrapper to read `FERMIHTC_DOCKER_CAPABILITIES` knob:

```
def getDockerCapabilities():
    """
    To fetch docker capabilities defined in the HTCondor
    configuration files and to append --cap-add to the
    capabilities to pass it to docker
    """
    # Fetching the capabilities from HTCondor configuration file
    condorDockerCaps = htcondor.param.get("FERMIHTC_DOCKER_CAPABILITIES")
    # Converting the fetched string into a list
    # and appending '--cap-add='
    try:
        finalDockerCaps = ["--cap-add="+val for val in re.findall(r'[\w]+',
            condorDockerCaps)]
    except TypeError:
        syslog.syslog('[condor-docker]: WARNING - ' \
            'FERMIHTC_DOCKER_CAPABILITIES is not defined in HTCondor' \
            'configuration. Not adding any capabilities to the container.')
        finalDockerCaps = ['']
    return finalDockerCaps
```

Docker with HTCondor

Some of the use cases we have so far been able to address by using a wrapper are as follows:

- We disabled swap for the containers. The default value per container adds up to be more than we have on our workers
- We were able to implement a ‘docker pull’ before ‘docker run’ to always try and run the newest available image. By default, docker only runs the latest from the local cache
- We were able to pass various privileged kernel capabilities to our containers in order to run experiment Singularity images
- We were able to set custom environment variables to facilitate FIFE experiments
- We were able to tweak network configuration of the containers to address performance and security concerns

Conclusions and future work

- It is our understanding that the HTCondor team would like to minimize the use of wrapper scripts.
- We are working with condor developers on getting some of these features into HTCondor itself that might benefit others
- Would be nice if HTCondor exposed a way to modify/remove/add CLI flags or arguments DOCKER_EXT_ARGS, maybe?
- Much of what we do is slightly convoluted because the features are job driven whereas we'd like to maintain administrative control. Would be nice to have features that allow administrative control natively.

Acknowledgements

University of Nebraska-Lincoln (UNL) invests heavily in Docker and containerization solutions for their users and compute clusters.

Thanks to Brian Bockelman and Nebraska site admins for their help in getting us started with Docker!

Thanks to the HTCondor team for implementing Docker support and continuing support for our use cases.