Class 10 - Running Jobs on Distributed Systems

[**ssh keys revisited:**](#_hozdy5ehtpj) **1**

[**Writable shared file system:**](#_enc4upe27t53) **2**

[Mount on Puppet Server:](#_nutclqj83snc) 2

[Mount on Puppet Client:](#_6k54j617p3cm) 2

[Create working directory on /nfs/htc-data:](#_maybo5bn358u) 3

[**Running remote jobs with ssh:**](#_y0wpnr64nyvg) **3**

[Without using a shared file system:](#_9foaxm8ivd0e) 3

[With shared file system:](#_t1oe3qdy563y) 4

[Problems and difficulties:](#_fw8aye1n4lnn) 4

[**Using a Batch System:**](#_hkx3txkjtlxl) **5**

[Installing HTCondor:](#_qi0dlz6i5b3i) 6

[Installing HTCondor on Puppet Server:](#_abs4ti3jmfz) 6

[HTCondor Firewall:](#_azz1akwru2ha) 8

[Installing HTCondor on Puppet Client:](#_j38g7b8n5o15) 8

[Submitting to HTCondor:](#_x7odbnr43dr9) 10

[Example Simple Submit Jobs:](#_qkv3ru1wb4yz) 10

# ssh keys revisited:

It is important the ssh keys that were generated in Class 9 are installed correctly.

Working from the **Puppet Server**, you should be able to:

* From the student account login into the student account on the Puppet Client without a password.
* From the root account login into the root account on the Puppet Client without a password.

|  |
| --- |
| **Only do the following if you are unable to perform the two operations above.** |

Login into the student account:

|  |
| --- |
| ssh-keygen -t rsa 2048  ssh-copy-id student@htcXXX |

Login into the root account:

|  |
| --- |
| ssh-keygen -t rsa 2048  ssh-copy-id root@htcXXX |

# 

# Writable shared file system:

Another **shared file system** has been added to the **class server**. This one allows you to **write** to from either your **Puppet Server** or **Puppet Client**.

## Mount on Puppet Server:

Modify **/etc/auto.nfs** and add the following line:

|  |
| --- |
| htc-data htc180:/nfs/htc-data |

|  |
| --- |
| cd /etc  vi /etc/auto.nfs  systemctl reload autofs |

You now be able to see the shared file system:

|  |
| --- |
| ls -al /nfs/htc-data/ |

## Mount on Puppet Client:

To mount the directory on the **Puppet Client** modify the **configuration** used by **autofs** module for the /**nfs** file systems. Add the same parameters you used for **/etc/auto.nfs** above:

|  |
| --- |
| cd /etc/puppetlabs/code/environments/productiont/modules/autofs/files/  vi auto.nfs |

## Create working directory on /nfs/htc-data:

We have **not** **created** unique **user accounts** yet, so you will need to create a **unique** **working directory** on /**nfs/htc-data**. I recommend:

|  |
| --- |
| /nfs/htc-data/htcXXX |

where **XXX** is the number of your **Puppet Server**.

|  |
| --- |
| It is important that these directories are unique so that you are not  overwriting other class member’s files. |

You now have a unique directory on a shared file system. These files are visible from all machines in the room.

# Running remote jobs with ssh:

|  |
| --- |
| **You should be working from the student account.** |

Before we start to talk about batch systems, let’s try to run some remote jobs without a batch system. Remember you are able to **execute remote commands** on your **Puppet Client** **without** a **password**.

## Without using a shared file system:

Create a directory in your home directory called **RemoteJobs**:

|  |
| --- |
| cd  mkdir RemoteJobs  cd RemoteJobs |

Now copy the **Prime** executable from the **class server** to this directory This is a simple **PERL** script that calculates **prime numbers**. It is **not** meant to be **efficient**, in fact it is simply meant to **use CPU time**.

|  |
| --- |
| cp /nfs/htc180/pub/condor/user-files/Prime Prime |

Now run the program and it should write out the first **10000 prime numbers**.

|  |
| --- |
| /home/student/RemoteJobs/Prime |

and if you want to **save** the numbers, you can use:

|  |
| --- |
| /home/student/RemoteJobs/Prime > Prime-Numbers-001.lst |

Now wouldn’t it be nice if we had to run **Prime** (or perhaps some other more useful program) **many times** to be able to **run it on the Puppet Client** host? Try:

|  |
| --- |
| ssh student@htcXXX /home/student/RemoteJobs/Prime |

|  |
| --- |
| **Oops! That didn’t work. Why not?** |

## With shared file system:

Let’s try this again using the **shared file system**. **Instead** of putting and running the program **Prime** from the **home directory**, install it and **run it from the shared directory**:

|  |
| --- |
| cd /nfs/htc-data/htcXXX  cp /nfs/htc180/pub/condor/user-files/Prime Prime  /nfs/htc-data/htcXXX/Prime |

and now try:

|  |
| --- |
| ssh student@htcXXX /nfs/htc-data/htcXXX/Prime |

|  |
| --- |
| **Well that’s better. We can now begin to use the remote resources.** |

## Problems and difficulties:

But this method has problems:

* It doesn’t utilize the resources efficiently.
* There is no easy way to run many instances of the program.
* If too many people are reading and writing to the shared file system, its performance will degrade.

|  |
| --- |
| It would be much better if we had a system to queue jobs and utilize the resources more efficiently. After all that’s what cluster computing is! |

# Using a Batch System:

**Batch processing** is the **execution** of a series of **jobs** on a computer or computing system **without manual intervention**. Batch systems or batch processing has a number of benefits:

* It minimizes computer idle time.
* Shares resources among many users.
* It allows the system to use different priorities for different jobs and/or users.
* The batch system can be tuned to a maximal number of jobs run at a time to maximize CPU efficiency.

|  |
| --- |
| **NOTE:** There is an **optimum** number of **simultaneous** jobs to run on a machine to **maximize** **CPU** **utilization** and therefore **throughput**. It is **generally** one **CPU intensive process per core** on a machine. |

The batch system will use is called [**HTCondor**](http://research.cs.wisc.edu/htcondor/)**.**

****

|  |
| --- |
| Our goal is to develop, implement, deploy, and evaluate mechanisms and policies that support **High Throughput Computing (HTC)** on large collections of distributively owned computing resources. Guided by both the technological and sociological challenges of such a computing environment, the **Center for High Throughput Computing** at UW-Madison has been building the **open source** HTCondor distributed computing software (pronounced "aitch-tee-condor") and related technologies to enable scientists and engineers to increase their computing throughput.  Note: The **HTCondor** software was known as **'Condor'** from **1988** until its name changed in 2012. |

|  |
| --- |
| **HTCondor** has an enormous number of options when configuring the system and when running jobs. It is **used for everything** from a **small cluster of computers** to being the **underlying technology** used for **GRID computing**. The entire **HTCondor manual** is over **1000 pages.** We will be doing a **simple**, but yet **powerful**, installation. The installation that we do **can** easily **scale** to hundreds or even thousands of computers. |

## Installing HTCondor:

**HTcondor** must be install on every machine that will run jobs as well as every machine that submits jobs. **HTCondor** relies on a number of services. These services can be all on one machine or spread across multiple servers. We will use the following setup:

* **Puppet Server** - Will server as the **HTCondor master node**. You will be able to submit jobs and run jobs on it. It will be the central server for the two node system.
* **Puppet Client** - Will only serve as a machine to **run jobs.** You will not be able to submit jobs from the **Puppet Client**.

|  |
| --- |
| Each on the machines, **Puppet Serve**r and **Puppet Client,** have 4 cores. So the two node batch system will have a **total of 8 batch slots**. |

We will install **HTCondor** on the **Puppet Server** from the **command line** and then install **HTCondor** on the **Puppet Client** using **puppet**. You will be asked to write the **puppet module** to install the software and the configuration files. Of course, if you get stuck, please ask for help.

### Installing HTCondor on Puppet Server:

|  |
| --- |
| **Please note:** At this time, 99% of the **HTCondor** **configuration** is being provided for you. We will try to return to this subject later, **but**:  **You could easily take what we do here and install it on a**  **much larger system without any changes.** |

Make sure you are logged into the root account on the **Puppet Server**. To install **HTCondor** use:

|  |
| --- |
| yum install condor |

This most likely installed a large number of packages.

Now configure the HTCondor master. The configuration files that you need are on the class server. Use the following to install them:

|  |
| --- |
| cp /nfs/htc180/pub/condor/condor-server/9\* /etc/condor/config.d/ |

Now we need to make one change to the file:

|  |
| --- |
| /etc/condor/config.d/95-cluster.config |

Edit this file and change the values of:

|  |
| --- |
| CONDOR\_HOST = IP\_PUPPET\_SERVER |

for example:

|  |
| --- |
| CONDOR\_HOST = 172.16.9.180 |

Now start and enable HTCondor using:

|  |
| --- |
| systemctl start condor  systemctl enable condor |

### HTCondor Firewall:

The HTCondor Server (machine running the Collector) needs to have a port opened in the firewall.

|  |
| --- |
| firewall-cmd --permanent --add-port=9618/tcp |

### Installing HTCondor on Puppet Client:

To install HTCondor on the Puppet Client you want to let puppet do the work. You need to create a module, **condor**, modify **init.pp** to install the **configuration files, install condor and add a service for condor**. This is very similar to what was done for autofs.

Recall that **modules** are **stored** in:

|  |
| --- |
| /etc/puppetlabs/code/environment/production/modules |

and you **create** a **puppet** module **skeleton** using:

|  |
| --- |
| cd /etc/puppetlabs/code/environment/production/modules  puppet module generate htcclass-condor  mkdir condor/files condor/templates |

The configuration files for condor go in:

|  |
| --- |
| /etc/puppetlabs/code/environment/production/modules/condor/files |

use the commands:

|  |
| --- |
| cd /etc/puppetlabs/code/environment/production/modules/condor/files  cp /nfs/htc180/pub/condor/condor-client/9\* files/ |

Then make the same change you made above for:

|  |
| --- |
| /etc/condor/config.d/95-cluster.config |

|  |
| --- |
| **NOTE:** The configuration files for Puppet Server and Puppet Client are identical except for the value of the parameter:  DAEMON\_LIST  in the file **/etc/condor/config.d/99-local.config**  Puppet Master:  DAEMON\_LIST = MASTER, COLLECTOR, NEGOTIATOR, SCHEDD, STARTD  Puppet Client:  DAEMON\_LIST = MASTER, STARTD |

### WARNING: Puppet Client OSG GPG Key Issue:

|  |
| --- |
| **WARNING**: You may encounter a problem when running puppet agent on the Puppet Client while installing HTCondor. HTCondor is installed from the osg.repo repository and the public for this repository might not be installed. If you have encounte an error  please try copying the key from the class server to the Puppet Client using:  ssh root@htcXXX  cp /nfs/htc180/oub/osg/3.3/RPM-GPG-KEY-OSG XXX.GNU /etc/pki/rpm-gpg/ |

## Submitting to HTCondor:

Before you submit you first jobs, check to make sure the system is working. Issue the commands:

|  |
| --- |
| condor\_q  condor\_status |

These should produce output and no errors.

Remember when you needed to use a shared file system to run Prime on Puppet Client? HTCondor not only doesn’t need a shared file system, it will collect the output from the job and send it back to the machine you submitted from. You can also specify data or job output files that will be transferred back to you machine.

### Example Simple Submit Jobs:

|  |
| --- |
| **Login to the student account**  **You should never try to run HTCondor jobs as root.** |

Create a subdirectory in you home directory and get a copy your first HTCondor submit files:

|  |
| --- |
| cd  mkdir HTCondor  cd HTCondor  cp /nfs/htc180/pub/condor/user-files/\* . |

You should now have 3 files in your working directory:

* **CondorTestJob** - This is the **HTCondor** configuration file that you will submit to the batch system.
* **TestJob** - A **simple** bash **shell script** to execute in batch system.
* **Prime** - An executable that is run from TestJob. It will be sent to the remote machine and its output returned to your machine.

**CondorTestJob** - Condor configuration file

Instructs **HTCondor** to transfer the files to execution host:

* TestJob
* Prime

Then run **TestJob** on the execution host and **copy** back the **Prime-Numbers.lst**

The STDOUT, STDERR and LOG information are written to the files in the current directory:

* Output = CondorTestJob.$(Cluster).$(Process).out
* Error = CondorTestJob.$(Cluster).$(Process).err
* Log = CondorTestJob.$(Cluster).$(Process).log

**HTCondor** **assigns** the unique values of **Cluster** and **Process**.

|  |
| --- |
| ####################  #  # submit description file  # Example 1: queuing multiple jobs with differing  # command line arguments and output files.  #  ####################  Executable = **TestJob**  Universe = vanilla  #Notification = Never  Output = CondorTestJob.$(Cluster).$(Process).out  Error = CondorTestJob.$(Cluster).$(Process).err  Log = CondorTestJob.$(Cluster).$(Process).log  ###################  ShouldTransferFiles = YES  WhenToTransferOutput = ON\_EXIT  TransferInputFiles = **Prime**  TransferOutputFiles = **Prime-Numbers.lst**  ##################  #Notification = Never  # Optional resource requests  #+cpus = 2  #+memory = 2048 # Mbytes  #+disk = 17384 # KB  #+maxWallTime = 120  Queue |

**TestJob**:

**Runs** a few standard **shell commands** and runs **Prime** writing the output to **Prime-Number.lst**

|  |
| --- |
| #!/bin/bash  echo " >>>>>>>>>> Hello World. I am running on <<<<<<<<<< "`hostname`  /bin/ls -alF  /usr/bin/printenv  /bin/pwd  /bin/df -h  /bin/ls -al Prime  ./Prime > Prime-Numbers.lst 2>&1  /bin/ls -al Prime-Numbers.lst  /bin/sleep 120  exit |

To submit this job to the batch system use:

|  |
| --- |
| condor\_submit CondorTestJob |

To the status of jobs in the queue:

|  |
| --- |
| condor\_q |

When the job finishes, you will have the first **10000** prime numbers in **Prime-Numbers.lst** and the various outputs from the job in the files mentioned above.

|  |
| --- |
| **Next time we will start developing a true workflow.** |