Combining Resources to Make One Cluster

[**Introduction:**](#_jqpt7zn2izli) **1**

[**HTCondor Changes:**](#_4itvhojr3jl) **1**

[Modifications for Puppet Server:](#_xs9pmw78ihlu) 2

[Modifications for Puppet Client:](#_okoo09srvmaa) 3

[**Create Temporary User Accounts:**](#_lgwofhlgjj2r) **3**

[The script /nfs/htc180/pub/AnNajah-Files/AddTemporaryUsers](#_9gp6gm9lh8z9) 4

[**Running a test using compiled C code:**](#_q8ke7eo6dscn) **5**

[Installing C compiler (gcc):](#_zmi5pmchpvk) 5

[HelloWorld HTCondor Tutorial:](#_5vn2ra8c98iv) 5

[HellWorld.c](#_ncodx1ds0j6e) 6

[Compile HelloWorld.c](#_flergfbsrnqh) 6

[Submit HelloWorld to HTCondor:](#_rmbyiykh3n4) 6

[Running Many Tests and Observing the queue:](#_ozzcpivm10pi) 7

# Introduction:

We are now at the point where we can run a **single HTCondor Master** on the class server and have all the **other machines** run as **HTCondor Submit** and **HTCondor Execute** node. Recall that to run as **Submit** node, you have to run **SCHEDD** and to run as an **Execute** node, you have to run **STARTD**.

# HTCondor Changes:

The changes to HTCondor are fairly straight forward:

* Every machine will run:
  + MASTER, SCHEDD, STARTD
  + Every machine will use 172.16.9.180 for CONDOR\_HOST
* This requires changing the files (in two different places):
  + 95-cluster.config
  + 99-local.config

Recall that we have not yet moved all of the machines to **Puppet** and we have not created a single **Puppet Master** yet. Perhaps having to change these files in more than one place will **exemplify** the **value** of **Puppet**. If we have a central **Puppet Master,** we would only have to make these change in one place and let Puppet Agent do the work.

## Modifications for Puppet Server:

The files for your **Puppet Server** are still configured by hand, so you have to edit them. On the **Puppet Server**, you will be doing the following:

|  |
| --- |
| cd /etc/condor/config.d  vi 95-cluster.config  vi 99-local.config |

In **95-cluster.config,** set the value for **CONDOR\_HOST**:

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

In **99-local.config**, set the value for **DAEMON\_LIST**:

|  |
| --- |
| DAEMON\_LIST = MASTER, SCHEDD, STARTD |

Then issue the commands:

|  |
| --- |
| condor\_reconfig  sleep 30  condor\_status |

**condor\_status** should starting looking like:

|  |
| --- |
| student@htc188 HellloWorld]$ condor\_status  Name OpSys Arch State Activity LoadAv Mem ActvtyTime  slot1@htc180.najah LINUX X86\_64 Unclaimed Idle 0.300 63930 0+01:00:04  slot1@htc185.najah LINUX X86\_64 Unclaimed Idle 0.000 11842 0+00:33:12  slot1@htc188.najah LINUX X86\_64 Unclaimed Idle 0.000 11843 0+00:19:43 |

Notice the additional hosts that are available for you to run on. As more groups add their machines to the htc180 pool, there will be more resources available. When everyone has re-configured their machines, we will have about 10 machines and about 50 cores to run on.

|  |
| --- |
| **BUT**: We are getting ahead of ourselves, we need to add the Puppet Client to the pool. |

## Modifications for Puppet Client:

For the **Puppet Client**, we have to make the same changes except the **Puppet Client** is managed by **Puppet** so the files are in:

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

Go to this directory and make the same changes to:

* 95-cluster.config
* 99-local.config

that you made above:

|  |
| --- |
| cd /etc/puppetlabs/code/environments/production/modules/condor/files  vi 95-cluster.config  vi 99-local.config |

In **95-cluster.config,** set the value for **CONDOR\_HOST**:

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

In **99-local.config**, set the value for **DAEMON\_LIST**:

|  |
| --- |
| DAEMON\_LIST = MASTER, SCHEDD, STARTD |

To speed the process, you can login to the Puppet Client and issue the commands:

|  |
| --- |
| puppet agent -t  condor\_reconfig  sleep 30  condor\_status |

Now condor\_status should show even more hosts available.

# Create Temporary User Accounts:

To make this test more meaningful, we are going to **create 20 accounts** on **every** **machine**. This just a quick way to create accounts that we can use now, we will delete these in the future when we have a better way to create and manage accounts. The script below will create **20** new **studentXXX** accounts. They all have the same password, the one we use for everything and they all can run sudo. The command is adduser and the options are:

**adduser:**

* -m - Create home directory with default files
* -p encrypted-password - Notice the encrypted password is in single quotes
* -r - Create an administer account, one that can run sudo
* -s /bin/bash - The login shell
* -K UMASK=002 - Set the protections on the home directory to 0755. I prefer this so that people can more easily share files.
* -C “Student Account XXX’ - The comment field in the password file
* studentXXX - The account that is created.

## The script /nfs/htc180/pub/AnNajah-Files/AddTemporaryUsers

|  |
| --- |
| #!/bin/bash  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 180" student180  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 181" student181  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 182" student182  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 183" student183  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 184" student184  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 185" student185  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 186" student186  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 187" student187  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 188" student188  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 189" student189  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 190" student190  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 191" student191  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 192" student192  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 193" student193  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 194" student194  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 195" student195  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 196" student196  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 197" student197  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 198" student198  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 199" student199  adduser -m -p '$1$KaNnjW8M$1RmtNZozm.rIj8HAC1DRS/' -r -s /bin/bash -K UMASK=002 -c "Student Account 200" student200  exit |

You can just run the script on both the Puppet Server and Puppet Client:

|  |
| --- |
| sudo /nfs/htc180/pub/AnNajah-Files/AddTemporaryUsers |

Now logout and log back in as the studentXXX for the GUI you are sitting in front of.

# Running a test using compiled C code:

Now that you are all individuals, it is time to use all the resources and share them. I was asked if we can run compiled C code with HTCondor? Of course, we can.

This tutorial contains 3 files and 3 subdirectories:

* HelloWorld.c
* CondorHelloWorld - Condor Resource File for 1 job
* CondorHelloWorldMany - Condor Resource File for 500 jobs
* output - Where the output goes.
* error- Where the errors go.
* logs- Where the logs go.

## 

## Installing C compiler (gcc):

Before we can compile C code, we have to install the C compiler. In Linux the C compiler is called gcc. To install it, use:

|  |
| --- |
| sudo yum install gcc |

|  |
| --- |
| Notice we are bypassing Puppet for now. We will fix all of this when we configure the central Puppet Master and Cobbler Master. Notice how we are moving toward one cluster, one room, one world? |

## HelloWorld HTCondor Tutorial:

This HTCondor **tutorial** is in **/nfs/htc-data/tutorials/HelloWorld.** Install it in your brand new home directory using:

|  |
| --- |
| cd  mkdir Condor  cd Condor  mkdir HellloWorld  cd HellloWorld/  cp -R /nfs/htc-data/tutorial/HelloWorld/\* . |

It is generally easier to manage the input and output from a shell script, but here is a simple C program that we will run with HTCondor. Guess what we call it?

## HellWorld.c

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  int main(void) {  char hostname[1024];  gethostname(hostname, 1024);  printf(" Hello World, I am running on %s\n", hostname);  return EXIT\_SUCCESS;  } |

## Compile HelloWorld.c

To compile the code use the following:

|  |
| --- |
| gcc HelloWorld.c -o HelloWorld |

|  |
| --- |
| **WARNING**: Make sure the last parameter above is **HelloWorld** not **HelloWorld.c**  If you use **HelloWorld.c** you will **overwrite** the source code file. |

Now if you run HelloWorld, you should get:

|  |
| --- |
| [student@htc188 HellloWorld]$ ./HelloWorld  Hello World, I am running on htc188.najah.edu |

## Submit HelloWorld to HTCondor:

Now you can submit HelloWorld to HTCondor using the Condor Resource File CondorHelloWorld. It is included with the tutorial files.

|  |
| --- |
| Executable = HelloWorld  Universe = vanilla  Output = output/CondorTestJob.$(Cluster).$(Process).out  Error = error/CondorTestJob.$(Cluster).$(Process).err  Log = logs/CondorTestJob.$(Cluster).$(Process).log  ShouldTransferFiles = YES  WhenToTransferOutput = ON\_EXIT  Queue |

Use the command:

|  |
| --- |
| condor\_submit CondorHelloWorld |

Check your job using (look for your username):

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

Then check the output using::

|  |
| --- |
| cat output/\* |

## Running Many Tests and Observing the queue:

Things get a lot more interesting when everyone submits many jobs at the same time to HTCondor. We can then see how resources are both shared and utilized. You can submit 500 jobs with one command:

|  |
| --- |
| condor\_submit CondorHelloWorldMany |

|  |  |  |
| --- | --- | --- |
| Look at the difference between **CondorHelloWorld** and **CondorHelloWorldMany**   |  | | --- | | diff CondorHelloWorld CondorHelloWorldMany |  |  | | --- | | It is a very small change for such a big effect. | |

Now look at the queue and the contents of your output directory.

You can look at the various places your jobs have run by examining their output. Again issue the command:

|  |
| --- |
| cat output/\* |

Or you can count the number of jobs the have run per machine using the script provided with the tutorial CountHosts:

|  |
| --- |
| ./CountHosts |

|  |
| --- |
| Now we are really doing  cluster computing!!! |