HTCondor Directed Acyclic Graph Manager

[**References:**](#_k0t3afnvcbgw) **1**

[**Introduction:**](#_ipjiyw8an37z) **1**

[**First DAG Use:**](#_8oqmusmn6atm) **3**

[**Second DAG Use:**](#_yo71nlc1mmh0) **6**

[Let’s look at a few files:](#_nwcvtev3k2ri) 6

[CondorCountWords:](#_t0b578detce1) 7

[Executable, Universe and Arguments:](#_2gw7wx1u19j9) 7

[The section for “transfer\_output\_files” is new:](#_5640cwnqsvgu) 7

[Queue Command:](#_23kvctqbw4im) 7

[InputData.lst:](#_1zx9096nvr25) 8

[CondorReport:](#_9couyomc6lt8) 9

[OutputData.lst](#_cuhcsm1w9itn) 9

[Run DAG Example:](#_sgbvfvf5dv6c) 10

[**Take It To XSEDE**](#_kth82kvjmlo) **10**

# References:

|  |
| --- |
| <http://research.cs.wisc.edu/htcondor/manual/v8.7/2_10DAGMan_Applications.html>  <http://www.iac.es/sieinvens/siepedia/pmwiki.php?n=HOWTOs.CondorSubmitFile> |

# Introduction:

**DAGMan** (Directed Acyclic Graph Manager) is a meta-scheduler for **HTCondor**. It manages dependencies between jobs at a higher level than the **HTCondor** **Scheduler**.

A directed acyclic graph (**DAG**) can be used to represent a set of programs where the input, output, or execution of one or more programs is dependent on one or more other programs. The programs are nodes (vertices) in the graph, and the edges (arcs) identify the dependencies. **DAGMan** submits jobs to HTCondor in the order defined in the DAG configuration file. The DAG configuration file is submitted to HTCondor using:

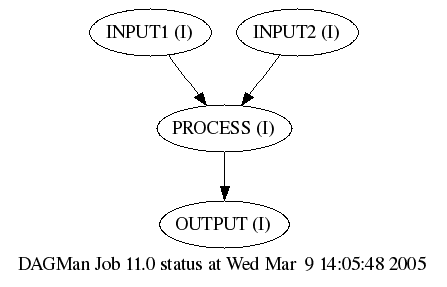
|  |
| --- |
| condor\_submit\_dag DAQ\_Configuration\_File |

The **DAG** configuration defines the **HTCondor** submit files that should be run and their **interdependency**. An example of a **DAG** **configuration** is:

|  |
| --- |
| JOB J01 INPUT1  JOB J02 INPUT2  JOB J03 PROCESS  JOB J04 OUTPUT  PARENT J01 J03  PARENT J02 J03  PARENT J03 J04 |

This would run J01 (INPUT1) and J02 (INPUT2), then J03(PROCESS) and finally J04(OUTPUT). In the configuration file INPUT1, INPUT2, PROCESS and OUTPUT are the names of **HTCondor configurations**. J01, J02, J03 and J04 **are alises** for the **HTCondor** **configuration** files. The above configuration is illustrated in the image below.

.



**HTCondor** **DAG** has been used for very complicated analysis as shown below. The image is the pictorial representation of the **DAG** used by the **LIGO** experiment. **LIGO** is an experiment to detect gravitational waves and was **awarded the 2017 Nobel Prize in Physics**.

## 

# First DAG Use:

The first example, uses an HTCondor DAG to count the number of words in 5 books and then list the top five used words in each book. All the processes that count words finish before the final process runs.

You can download the code for the class tutorials from GITHUB:

|  |
| --- |
| cd  cd Condor  git clone https://github.com/taktse/htc-class |

This will create a subdirectory htc-class. In that directory you will find:

|  |
| --- |
| condor-wn01> ls -al  total 8  drwxr-xr-x 7 drjohn man 91 Apr 25 16:01 .  drwxr-xr-x 3 drjohn man 23 Apr 25 16:01 ..  drwxr-xr-x 8 drjohn man 163 Apr 25 16:01 .git  drwxr-xr-x 2 drjohn man 116 Apr 25 16:01 CalcPi  drwxr-xr-x 4 drjohn man 234 Apr 25 16:01 DAG  drwxr-xr-x 6 drjohn man 4096 Apr 25 16:07 DAG01  drwxr-xr-x 2 drjohn man 145 Apr 25 16:01 HelloWorld  -rw-r--r-- 1 drjohn man 1120 Apr 25 16:01 README.md |

We want to use DAG01:

|  |
| --- |
| condor-wn01> di  /batch/condor-wn01/test/htc-class/DAG01  total 48K  drwxr-xr-x 3 drjohn man 214 Apr 25 16:12 ./  drwxr-xr-x 7 drjohn man 91 Apr 25 16:01 ../  -rw-r--r-- 1 drjohn man 1.4K Apr 25 16:01 01.condor  -rw-r--r-- 1 drjohn man 1.4K Apr 25 16:01 02.condor  -rw-r--r-- 1 drjohn man 1.4K Apr 25 16:01 03.condor  -rw-r--r-- 1 drjohn man 1.4K Apr 25 16:01 04.condor  -rw-r--r-- 1 drjohn man 1.4K Apr 25 16:01 05.condor  -rwxr-xr-x 1 drjohn man 81 Apr 25 16:01 Clean\*  -rw-r--r-- 1 drjohn man 924 Apr 25 16:01 CondorReport  -rwxr-xr-x 1 drjohn man 191 Apr 25 16:01 CountReport\*  -rw-r--r-- 1 drjohn man 588 Apr 25 16:01 CountWords.dag  -rwxr-xr-x 1 drjohn man 54 Apr 25 16:01 Initialize\*  -rwxr-xr-x 1 drjohn man 609 Apr 25 16:01 WordCount\*  drwxr-xr-x 2 drjohn man 4.0K Apr 25 16:01 inputs/ |

Let’s looks at these files:

* WordCount - python script that counts the words in a file. You should try WordCount -h and reading the code. This script does command line parsing, opens a file for reading, uses a standard python to count words and writes the output to STDOUT.
* 01.count - Condor submit file to count words in a hardwired book . Please read this file. Make sure you understand what it does.
* Clean - Bash script to clean out files from a previous run.,
* CountReport - Bash script to combine the results at the end. This is the last script run by HTCondor
* CondorReport - HTCondor submit script for the CountReport script. Again, please read this and understand what it does.
* Initialize - Set up the directory to run. It creates the output, error and log directories.
* CountWords.dag - The HTCondor DAG configuration for this workflow.

|  |
| --- |
| #--------------------------------------------------------------  #  # HTCondor DAG file WordCount  #  #--------------------------------------------------------------  # Define the parent jobs  # Each HTCondor job analyzes one book  #  JOB J01 01.condor  JOB J02 02.condor  JOB J03 03.condor  JOB J04 04.condor  JOB J05 05.condor  #  # Define the HTCondor job that will combine the results  #  JOB REPORT CondorReport  #  # Define the relationships and dependency  #  PARENT J01 CHILD REPORT  PARENT J02 CHILD REPORT  PARENT J03 CHILD REPORT  PARENT J04 CHILD REPORT  PARENT J05 CHILD REPORT |

This configuration file defines 6 jobs: J01, J02, J03, J04, J05 and REPORT. J01, J02, J03, J04, and J05 are the parents of the job REPORT. This means J01, J02, J03, J04, and J05 run and finish before REPORT is run. To define a job (or node in DAG lingo), we have a line beginning with the keyword JOB followed by a unique identifier for that job, for example, J00 for the first job and, the HTCondor submit file to be used, i.e. 01.condor. A line starting with pound (#) character is a comment. Dependencies are formed with the keywords PARENT/CHILD as in PARENT J01 CHILD REPORT.

Here is how you run it:

|  |
| --- |
| ./Clean  ./Initialize  condor\_submit\_dag CountWords.dag |

If you watch condor\_q, you will see:

* DAGman job start
* 5 WordCount jobs will run
* The WordCount jobs will finish at different times
* CountReport will run and produce the final result report.txt. It will be similar to:

|  |
| --- |
| The top 5 words with the highest word frequency for each book:  Alices\_Adventures\_in\_Wonderland\_by\_Lewis\_Carroll.txt  said 462  alice 403  that 330  with 227  this 181  Dracula\_by\_Bram\_Stoker.txt  that 2480  with 1323  have 1057  when 771  there 769 |

# Second DAG Use:

Now use the code in the DAQ subdirectory:

|  |
| --- |
| -rwxr-xr-x 1 drjohn man 81 Apr 25 16:17 Clean\*  -rw-r--r-- 1 drjohn man 914 Apr 25 16:42 CondorCountWords  -rw-r--r-- 1 drjohn man 867 Apr 25 16:17 CondorReport  -rwxr-xr-x 1 drjohn man 191 Apr 25 16:17 CountReport\*  -rw-r--r-- 1 drjohn man 81 Apr 25 16:17 CountWords.dag  -rwxr-xr-x 1 drjohn man 54 Apr 25 16:17 Initialize\*  -rw-r--r-- 1 drjohn man 665 Apr 25 16:17 InputData.lst  drwxr-xr-x 2 drjohn man 49 Apr 25 16:17 ListFiles/  -rwxr-xr-x 1 drjohn man 911 Apr 25 16:17 MakeListFiles\*  -rw-r--r-- 1 drjohn man 99 Apr 25 16:17 OutputData.lst  -rwxr-xr-x 1 drjohn man 884 Apr 25 16:18 WordCount\*  drwxr-xr-x 2 drjohn man 4.0K Apr 25 16:17 inputs/ |

## Let’s look at a few files:

Before you run this example, you should work on **understanding** what this **workflow** does. Once you understand this workflow, you will begin to understand how to build more complicated workflows and data analysis than simply one or many independent jobs. This workflow:

* Uses a list of i**nput files** and **command arguments in a file**.
* **Uses the line number** from the **InputData.lst** file to defined the **output file** in the **HTCondor** submit file **CondorCountWords**.
* Both **WordCount** and **CountReport** rely on the **file transfer feature of HTCondo**r to send and receive data.
* The files **InputData.lst** and **OutputData**.lst are derived from the book files in the subdirectory **inputs**. Please look at **MakeListFile**.

### 

### **CondorCountWords**:

**HTCondor** submit file for **WordCount**. This submit file uses some **new features** of **HTCondor**. Let’s look at a few different pieces of this file.

#### Executable, Universe and Arguments:

The file contains the standard definitions for **Executable** and **Universe**. **Arguments** is defined as **$(file)**. This tells **HTCondor** that the **Arguments** will be defined in a file given on the **Queue** line.

|  |
| --- |
| Arguments = $(file) |

#### The section for “transfer\_output\_files” is new:

|  |
| --- |
| transfer\_output\_files = output.dat  line = $(Row)+1  transfer\_output\_remaps = "output.dat=output/count$INT(line).dat" |

This code defines the output to be “**output.dat**”, but then remaps (renames) it to “**output/count$INT(line).dat**”. The variable “line” is the **line number** from the **input Arguments file.** It starts counting at 0. So when the files appear back on the submit machine they are of the form:

|  |
| --- |
| output/count1.dat  output/count2.dat  .... |

This gives us a way to keep the **output** files **associated** **with** the **input** **arguments**.

#### Queue Command:

|  |
| --- |
| Queue transfer\_input\_files,arguments from InputData.lst |

This submits one job for each line in the file **InputData.lst.** The first field is the **input** file that is **transferred** to the execute machine and the second field are the **arguments** that are passed to **Executable**.

#### InputData.lst:

Please be sure to look closely at the file **InputData.lst**. It has lines like:

|  |
| --- |
| inputs/Alices\_Adventures\_in\_Wonderland\_by\_Lewis\_Carroll.txt, -f Alices\_Adventures\_in\_Wonderland\_by\_Lewis\_Carroll.txt -o output.dat |

Where:

* **inputs/Alices\_Adventures\_in\_Wonderland\_by\_Lewis\_Carroll.txt**    
  Input file transferred to execute host.
* **-f Alices\_Adventures\_in\_Wonderland\_by\_Lewis\_Carroll.txt -o output.dat**   
  Command line arguments passed to Executable.

### Putting it all together:

|  |
| --- |
| ####################  #  # submit description file  # Example 1: queuing multiple jobs with differing  # command line arguments and output files.  #  ####################  +ProjectName = "TG-CIE170062"  Executable = WordCount  Arguments = $(file)  Universe = vanilla  transfer\_output\_files = output.dat  line = $(Row)+1  transfer\_output\_remaps = "output.dat=output/count$INT(line).dat"  # Notification = Never  Error = error/CondorCountWords.$(Cluster).$(Process).err  Log = logs/CondorCountWords.$(Cluster).$(Process).log  # Send the job to Held state on failure.  on\_exit\_hold = (ExitBySignal == True) || (ExitCode != 0)  # Periodically retry the jobs every 1 hour, up to a maximum of 5 retries.  periodic\_release = (NumJobStarts < 5) && ((CurrentTime - EnteredCurrentStatus) > 60)  ShouldTransferFiles = YES  Queue transfer\_input\_files,arguments from InputData.lst |

### CondorReport:

**HTCondor** submit file for **CountReport**. Notice that it also **transfers** files to the **execute** node. These files will be read by **CountReport** to do the final processing.

### OutputData.lst

List of output files from the previous step that **CountReport** will process:

|  |
| --- |
| output/count1.dat, output/count2.dat, output/count3.dat, \ output/count4.dat, output/count5.dat, |

These are the files that are defined in **CondorCountWords**.

### Run DAG Example:

**Please try to run this on your own**. You should be able to **generate** the final **report.txt** file. Once you do that look at the **MakeListFiles** and create new versions of **InputData.lst** and **OutputData.lst**. Run the workflow again.

|  |
| --- |
| **Now you have a workflow that can analysis any number of books in text format.** |

Since your workflow is expandable, let’s do it. Go to the website:

|  |
| --- |
| https://www.gutenberg.org/ |

You can download many books in text format from this site. You have over 50,000 to choose from. When you save the books to you the inputs subdirectory make sure they do not have space or any other special characters.

# Take It To XSEDE

Now that you have run the DAG example on the local cluster, try running it on XSEDE. Once you login to XSEDE, use git to get the code and try the examples:

|  |
| --- |
| cd  cd Condor  git clone https://github.com/taktse/htc-class |

|  |
| --- |
| Please note that I **have not tried this** and do not know that it will run without any changes on **XSEDE**. If it doesn’t work on **XSEDE**, please use this an opportunity to make it work there. After all you are the **Cluster Computer Experts** at **An Najah University**. |