## Distributed Computing Exercise Using HTCondor on CHTC Machines

Note that in this exercise we are using small data for which the parallel step slows us down! Working in parallel will help when we have more data than fit in memory on one machine. The goal here is to learn HTCondor scripting to prepare for a large search for Lyman-break galaxies.

- Write a HTCondor DAGMan script, words.dag, to make a sorted list of the words and their counts in the works of Shakespeare from shakespeare.tar (which extracts to a shakespeare directory). Include the following steps:
  - Write a PRE ".sh" script:
    - \* Write all the plays to one large file.
      - wget can download shakespeare.tar from
        - https://pages.stat.wisc.edu/~jgillett/DSCP/CHTC/wordCounting/shakespeare.tar
      - $\cdot$  tar can extract it.
      - $\cdot\,$  cat with wildcards can concatenate all the plays into one file.
    - $\ast\,$  Break the large file into 5 smaller files.
      - split can do it. Search for CHUNKS in the split man page for a one-line way to do this without splitting lines.
  - Write a HTCondor ".sub" script to process the 5 files in parallel by calling a ".sh" script on each file. The ".sh" script should use a pipeline to reformat and then sort the current file:
    - \* Make all letters lowercase, e.g. with tr '[:upper:]' '[:lower:]' (see man tr).
    - \* Remove punctuation. e.g. sed -e 's/PATTERN/REPLACEMENT/g' ("stream editor: substitute PATTERN with REPLACEMENT globally") can do it using the [:punct:] character class (inside another set of []).
    - \* Make it have one word per line, e.g. by replacing each tab or space with a newline e.g. Use sed again.
    - \* Remove blank lines (and those containing only spaces). e.g. grep -v "PATTERN" can do it, since -v selects non-matching lines. Write a PATTERN for "start-of-line zero-or-more-spaces end-of-line".
    - \* Sort the file of words with **sort**.
  - Write a POST ".sh" script:
    - \* Merge the sorted small files into one large sorted file, e.g. with sort -m. (Don't just run sort, which would waste the parallel sorting of the small files. As you can see in the sort man page, sort -m *merges* several sorted files *without* sorting. See also the Wikipedia page for k-way merge.)

Note: sort can produce output that looks unsorted because it relies on your computer's locale settings. You can ensure familiar English sort order by running export LC\_ALL=C in your script before running sort.

\* Convert the merged sorted file to one called countsOfWords with lines of the form count word

e.g. uniq can do it (see man uniq to include counts). Then sort by descending count.

Hint: My solution found "the" to be the most frequent word.

Submit one tar file per group, words.tar, that extracts to make a directory words containing:

- your words.dag file and any other code you write
- your countsOfWords file
- a plain-text file README that includes information on your group members (3 to 5 students) in the line format NetID,LastName,FirstName. For example, if Wilma Flintstone (NetID: wflint3) and Charlie Brown (NetID: cbrown71) worked together, their REAMDME file would be:

wflint3,Flinstone,Wilma cbrown71,Brown,Charlie

• no input data