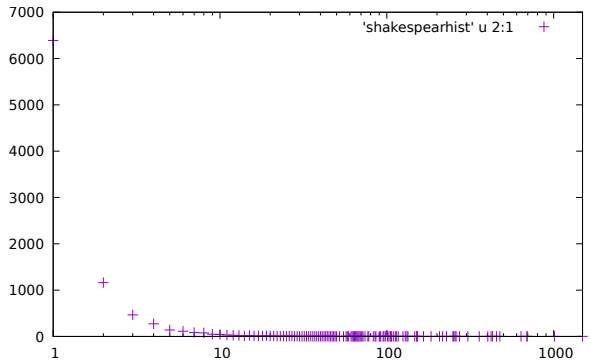
Assignment Map Reduce MPI: Word Histogram

As usual all time measurements are to be performed on the cluster. The application using MapReduce MPI need to be linked against the mrmpi library. In the assignment, the Makefile accounts for it.

1 Word Count Frequencies

We are given a set of text files. We want to generate for each textfile a histogram that present how many words appear a particular number of times. Look for instance at the results for Shakespeare's Hamlet. There are 6390 words that appeared exactly once, 1162 words who appeared exactly twice and 466 words who appeared exactly three times.



The problem is to write a MapReduceMPI application that can perform that analysis on many files at once. That is to say for each file, the application should generate the histogram of how many words appear how many times.

The application should be designed to be a single pipeline. And the application should be designed so that all map and reduce tasks require a constant amount of memory (read: $O(1)$). However, there is no restriction on how many key-value pairs they can generate or ingested by a task. The program should generate the histogram for each file before completing.

For the purpose of the assignment, you can count as a word any string of characters that does not contain a white character (space, line break, ...). Feel free to ignore punctuation.

Question: Go into the `wordhist/` directory and write the code in `wordhist.cpp` using MapReduce MPI. There is a sequential implementation of the problem in `wordhist_seq.cpp`. Although the MapReduce implementation will looks fairly different from the sequential implementation.

The code should take any number of parameters which are the files being processed.

The program should:

- Print on `stdout` for each file the data necessary to make an histogram.
- Print on `stderr` the time it took for the application to make that computation (IO included).

You can test your code with `make test`. Though you will notice that the test isn't as exhaustive as in previous assignments. You'll have to check the result by hand.

Question: Run the code on Centuarus using `make bench`. And once it is over, run `make plot`. Note that in this assignment `make bench` may queue the jobs one at a time and wait for each job to complete.