Activity: Coarse Grain Synchronization using Parallel Hashtable

The purpose of this assignment is for you to learn more about:

- creating threads and passing parameters to them,
- the overhead associated with thread management and synchronization,
- designing parallel data structures.

As usual all time measurements are to be performed on the cluster.

1 Preliminary

Before anything, we will need to get timing for sequential execution.
Question: On Centaurus, navigate to the `sequential/` directory. Compile the sequential code with `make`, and get sequential time by running `make bench`.

2 Coarse Grain Synchronization

Question: Write a program that uses a hashtable to track the frequency of words in a given text using multi-threading. You can use the sequential code as a reference.

Note: for this assignment you should not need to make changes to `MyHashtable.hpp` or `Dictionary.hpp` (leave them alone). Also note, that the parsing is already done for you.

The easiest way to do this is, after the files have been read and parsed, to create one thread per file. Have each thread read the files and count the words directly into the Hashtable.

Main takes 3 parameters:

- `source`: names a file that lists the texts to count from
- `testword`: (for testing correctness) a word that is looked up at the end of execution
- `threshold`: (for debugging purposes) a `int` that specifies to print words that have a frequencies greater than this number

Once you have written the program, you can verify that it works with `make test` once you pass all the tests you can move on.
Question: Report time and speedup (or slowdown) across a range of executions using various numbers of cores. You can use `make bench` on Centaurus to start this process, as it will create 16 jobs. Once they are complete, plot the charts with `make plot`. Note: you must run these commands from within the `coarse_grain/` directory.
Question: Why is the speedup (or slowdown) for coarse grain the way it is?