Activity: Fine Grain Synchronization using Parallel Hashtable

1 Fine Grain Synchronization

The coarse grain synchronization of the previous activity yielded little speedup because the synchronization locked the entire hash table.

Question: Write a program that uses a hashtable to track the frequency of words in a given text using multi-threading and a fine grain synchronization scheme. Use your coarse grain code as a starting point. And write the code in the `fine_grain/` directory.

Note: for this assignment you will need to make changes to `MyHashtable.hpp` and to the `main.cpp` you wrote in the previous activity.

Think of what is the semantic that you need to make this application work: the fundamental operation that counting word needs. You may need to introduce new functions in the hash table to provide this semantic. You may want to disable standard features of hash tables to make the fine grain synchronization easier to write. (For instance, you may want to set an initial number of buckets and disable the bucket resizing feature.)

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 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 `fine_grain/` directory.

Question: Did you achieve a better speedup with fine grain synchronization than with coarse grain?

Note: A higher score will be given for better speedup. We expect speedup to be between 3 and 4.