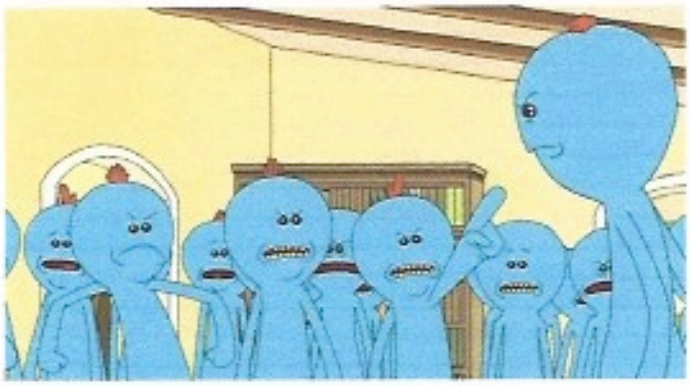
CS302

Assignment 4: Matrix Multiplication in Parallel



Description

Mr. Meeseeks compute the product of two matrices, I'm too busy and really don't care about this problem. I can't trust just one of you to compute the entire result because of what happened last time with Jerry. So I'll spawn the legal amount of Mr. Meeseeks I'm allowed to spawn to compute this problem.

For this assignment, you will need to write a function that is assigned to each thread that computes part of the product of two matrices, and when all the threads are done, the product of two matrices has been computed. You may use a 2D `std::vector< std::vector<int> >` object to maintain the matrices, and they can be global vectors since they will be shared memory among the threads. You might not need to use `std::mutex` locks since each thread might not be writing to the same locations in the matrices at the same time.

Thus in main, it will be your responsibility to not spawn more than the amount of threads that can be run concurrently, and assign each thread to the appropriate part of the matrices to allow the computation to be done in parallel. A warning, passing by reference into a thread function causes issues. Luckily you can have your matrices declared globally, and you can pass value parameters into the thread function that tells the thread which parts of the matrices the thread will access. You would need a `std::vector< std::thread >` object to maintain the list of threads.

Contents of main

In main, you will prompt for an input file, each input file contains the following

n_1 m_1

A set of n_1 lines with m_1 integers on each line separated by a blank space, each line is terminated by an end of line.

n_2 m_2

A set of n_2 lines with m_2 integers on each line separated by a blank space, each line is terminated by an end of line.

Thus you will allocate an $n_1 \times m_1$ and an $n_2 \times m_2$ matrices and store the values read from the file. You can assume $m_1 = n_2$ so a matrix multiplication can actually be done. The resulting matrix would be a $n_1 \times m_2$ matrix, the result will be written into this matrix and the output once the result has been computed.

Specifications

- Your program should be multi-threaded
- Do not more than the legal amount of threads to run concurrently
- Remember you must use the `-pthread` flag to compile and if you are using a mac terminal then you also need to include `-std=c++11`
- If you are using the school remove server to run your code, DO NOT USE BOBBY.CS.UNLV.EDU or SALLY.CS.UNLV.EDU !!!! Please use cardiac.cs.unlv.edu to run your program

Write Up

Along with your code, you also need to provide a write up that addresses the following

1. What operating system/environment did you run your program?
2. How many threads were you able to run concurrently?
3. Describe your parallel algorithm (how you implemented it etc)
4. Is your algorithm completely executed in parallel or is there any parts of your program that must run sequentially? (except for the portion in main that spawns the threads)
5. Is adding more threads going to automatically going to speed up your algorithm? Is there a threshold for the max amount of threads that can be spawned that would not cause a speed up if the amount of threads spawned goes beyond this threshold? (Assuming your system can spawn infinite amount of threads)? Assume you have two matrices $n_1 \times m_1$ and $n_2 \times m_2$, what is the threshold amount? (If any)

Sample Run

```
Cannot fit the results here, I linked the output in a file on canvas
```

Submission

Submit your source code and write up to code grade by the deadline

References

- Link to the top image can be found at *https://www.dailydot.com/parsec/mr-meeseeks/*