CS 385 Project 1: Matrix multiplication in OpenMP

Due: Wednesday, March 11 at 11:59 PM

Summary

You will parallelize and optimize a matrix multiplication program using OpenMP.

Deliverables

You will turn in:

- Your final, parallel optimized code (as *yourlastname*P1.cpp)
- Your writeup (as *yourlastname*P1 followed by the extension for your file format).

You may submit these assignments by copying them to `~srivoire/cs385/submit`. You can verify your submission by checking http://rivoire.cs.sonoma.edu/cs385/proj1sub.txt.

Getting started

Copy these two files into your project directory on cwolf:

- `~srivoire/cs385/pickup/p1-stub.h`
- `~srivoire/cs385/pickup/p1-stub.cpp`

You are responsible for defining the `matmult` function prototyped in `p1-stub.h` in your project file. This function, and any functions it calls, should be the only functions in your project file. I will test your code with my own copies of `p1-stub.h` and `p1-stub.cpp`, so your code needs to work with these files as written in order to get credit.

The main program in `p1-stub.cpp` does the following:

- Initializes two `NxN` matrices `A` and `B` with random data and a matrix `C` with zeroes
- Multiplies `A` and `B` using your `matmult` function and checks for correctness
- Multiplies `A` and `B` repeatedly without reinitializing `C` in order to time the performance of your function
- Computes and prints timing information for your function.

To get started, you can base your code on the sequential correctness-checking code from `p1-stub.cpp` into your project file.

To compile your code, you may create your own makefile or just type
`g++ -fopenmp p1-stub.cpp` *`yourlastname`*`P1.cpp`

Correctness

You must not sacrifice correctness for parallel performance. Your code must work correctly for:

- Any value of `N` that does not cause the array initializing functions in `p1-stub.h` to segfault
- Any value of `P` <= `N` (for very large N, OpenMP will probably refuse to allocate that many threads)

Optimizations

You may optimize your code either for cwolf or for a multicore machine of your choosing. If you choose a machine other than cwolf, you **must** add the processor manufacturer and model to your writeup.
On Linux, including on cwolf, you can get all kinds of information about the processor by typing

```
cat /proc/cpuinfo
```

If you use a machine that you do not have exclusive access to (including cwolf), you must run all performance numbers three times and report the individual measurements and the average.

Hints for optimizing your code:

- You have a choice of three loops to parallelize – which one is best?
- You can control OpenMP's assignment of iterations to threads with the schedule directive.
- You can conceptually subdivide your matrix into smaller pieces in order to maximize the percentage of memory accesses that are fulfilled by the cache. This will require some research about the cache size of your machine, as well as some algorithmic restructuring. A good online resource is: http://mathworld.wolfram.com/BlockMatrix.html
- You can play with g++'s compiler flags to see if they provide any added benefit.

You must attempt at least 5 optimizations. The correctness of your code and the quality of your optimizations are worth 35% of your assignment grade.

Writeup

Your writeup is worth 65% of your assignment grade and must contain the following:

- [20%] Performance numbers for each of the 5 optimizations:
  - Single-threaded and double-threaded (or more if you get better results that way)
  - Results for at least 3 different values of N that must be separated by at least a factor of 10.

  That's at least 6 data points per iteration, and remember that you must report 3 measurements *per data point* if you are using a machine to which you don't have exclusive access.
- [25%] Analysis of your optimizations: why did you perform them? Why do you think you got the results you did?

- [15%] A diary of your time spent in the design and coding stages (not performance measurement). In particular, keep track of your time:
    - To get the first correct parallel implementation
    - To get the best-performing implementation

  I would prefer that your diary be accurate rather than perfectly precise. If, for example, you forget to precisely account for about an hour of coding time, please note that rather than omitting it out of fear that you will get points off.
- [5%] Your qualitative thoughts about this exercise and/or about OpenMP. For example, what was easy? What was hard? What surprised you? What did you learn?
- Citations for any sources of help you received

You may submit your writeup in any commonly used format. PDF is preferred, but Microsoft Word documents, HTML files (including gzipped HTML files with images), and text files will be accepted. Clear organization and formatting can only help your grade.

Collaboration and Citation

You are welcome to discuss optimization strategies with each other and others, but the collaboration policy about not sharing your code with others is in full effect, and your writeup must also be your own work. You should feel free to do research on optimizing matrix multiplication and on OpenMP; however, you should cite your sources in your writeup in order to avoid committing academic misconduct.