CS250B: Modern Computer Systems

Lab 1: Matrix Multiplication Derby

Due: 2021-05-13

Overview

In this lab, you will write a best-effort multithreaded implementation of a dense matrix-matrix multiply in C/C++, and compare it against other implementations with various levels of optimizations.

Files Involved

In the lab directory, you will find four files, `guide.pdf`, `main.cpp`, `mult.cpp`, and `Makefile`. You will only edit mult.cpp. guide.pdf is this document

Implementation guide

Try to use all ideas discussed in class, including AVX, cache-obliviousness and loop unrolling. Some ideas may not work well together. Achieve the best you can!

The sole function in the `mult.cpp` file, `mult`, takes in five arguments: `a, b, c, matrix_size` and `thread_count`. `a` and `b` are to be multiplied and stored in `c`. `matrix_size` defines the size of the matrix input, where there is $\texttt{matrix\_size}^2$ elements in the square matrix. `thread_count` define the number of threads to be spawned.

The `mult` function is currently implemented using a naive triple-loop method.

In order to build and run the project run make, and then `./matrix T M`. T is the number of threads argument that will be passed to your mult function. N is the size of the matrix, where the number of elements in the matrix is N*N. The default value of N is 2048 unless specified.

Evaluation environment

Your code will be tested on various (but reasonable, desktop-class) machines, with different cache, processor and memory characteristics, using a different thread values. The matrix size will vary between 2048 to 16384.

All machines will have AVX2 and FMA support at least.

The derby

Each submission will be given a code name for anonymity, and I will create charts and leaderboards to compare various implementations, as well as some implementations I will make with various levels of optimizations.

Grading will not be on a curve, so relax, and do your best