Course Project #2: Matrix-Matrix Multiplication

Due date: Oct. 15

1. Introduction

The objective of this design project is to implement a C/C++ module that carries out high-speed matrix-matrix multiplication by explicitly utilizing (i) multiple threads, (ii) x86 SIMD instructions, and/or (iii) techniques to minimize cache miss rate via restructuring data access patterns (as discussed in class). Matrix-matrix multiplication is one of the most important data processing kernels in numerous real-life applications, e.g., machine learning, computer vision, signal processing, and scientific computing. This project aims to help you gain hands-on experience of multi-thread programming, SIMD programming, and cache access optimization, which all could very much help your job hunting in the near future. It will help you develop a deeper understanding of the importance of exploiting task/data-level parallelism and minimizing cache miss rate. Have fun!

2. Requirement

Your implementation should be able to support (1) configurable matrix size that can be much larger than the on-chip cache capacity, and (2) both fixed-point and floating-point data. Moreover, your implement should allow users to individually turn on/off the three optimization techniques (i.e., multi-threading, SIMD, and cache miss minimization) and configure the thread number so that users could easily observe the effect of any combination of these three optimization techniques. Other than the source code, your Github site should contain

(1) Readme that clearly explains the structure/installation/usage of your code
(2) Experimental results that show the performance of your code under different matrix size (at least including 1,000x1,000 and 10,000x10,000) and different data precision (4-byte floating-point, 2-byte fixed-point)
(3) Present and compare the performance of (i) native implementation of matrix-matrix multiplication without any optimization, (ii) using multi-threading only, (iii) using SMID only, (iv) using cache miss optimization only, (v) using all the three techniques together, and (vi) three 1+1 combinations of the three optimization techniques
(4) Thorough analysis and conclusion

3. Additional Information

C does not have built-in support for multithreaded application, hence must rely on the underlying operating systems. Linux provides the pthread library to support multithreaded programming. You can find a very nice tutorial on pthread at https://computing.llnl.gov/tutorials/pthreads/. Microsoft also provides support for multi-thread programming (e.g., see https://docs.microsoft.com/en-us/windows/win32/procthread/multiple-threads). Since C++11, C++ has built-in support of multithreading programming, feel free to utilize it. You are highly encouraged to program on Linux since Linux-based programming experience will help you most on the job market.

The easiest way to use SIMD instructions is to call the intrinsic functions in your C/C++ code. The complete reference of the intrinsic functions can be found at https://software.intel.com/sites/landingpage/IntrinsicsGuide/, and you can find many on-line materials about their usage.

Moreover, matrix-matrix multiplication has been well studied in the industry, and one well-known library is the Intel Math Kernel Library (MKL), which can be a good reference for you.