APMA 2822b Homework 2

Ryan Greenblatt

February 2019

1

Code is attached in the email.

2

The total number of FLOPS for Matrix-Matrix multiplication with matrices of size n is $2n^3$. The total data in bytes that must be transfered (assuming the use of doubles) is $8 * 3n^2 = 24n^2$. The arithmetic intensity for this task is $\frac{2n^3}{24n^2} = \frac{1}{12}n = \frac{4096}{12} = 341.3$. This is well above the ratio of peak FLOPS to peak memory bandwidth for this system ($\frac{300}{55} = 5.5$). Thus, the computation can be expected to ideally be FLOP bounded. However, poor cache utilization or other bottlenecks could result in the computation not being FLOP bounded.

3

I implemented an algorithm which utilizes 3 levels of blocking. The outer most level of blocking is split between threads and loops over the first dimension of the A and C matrices. The middle level loops over the last dimension of the A matrix and the first dimension of the B matrix. The inner level loops over the last dimensions of the C and B matrices. In addition to tuning the block size parameters, I tested aligning threads side by side on the outer level instead of the 3rd level of blocking. This greatly reduced performance. I also tested transposing the blocks of the A matrix and copying the blocks from the B matrix to allow for the data access to be more sequential during the final operation; however, this reduced performance. The Eigen linear algebra library was also benchmarked for comparison (by default the code is run without benchmarking Eigen). I tested with and without the '-march=native' compiler flag to determine how much of a speed up each method is obtaining from the use of optimized SIMD instructions. The naive approach appears to obtain no benefit from SIMD, the optimized algorithm obtains a sizeable benefit from SIMD, and Eigen gains an even larger benefit from SIMD.

Implementation	n	Seconds	GLOPS
Naive	512	0.0249	10.5
Optimized	512	0.00427	60.0
Eigen	512	0.00333	75.1
Naive	1024	0.181	11.0
Optimized	1024	0.0335	59.6
Eigen	1024	0.0127	158.1
Naive	2048	11.9	1.3
Optimized	2048	0.338	47.4
Eigen	2048	0.0805	198.7

Table 1: Performance with the '-march=native' compiler argument

Implementation	n	Seconds	GLOPS
Naive	512	0.0244	10.2
Optimized	512	0.00797	31.4
Eigen	512	0.00403	62.0
Naive	1024	0.221	9.04
Optimized	1024	0.0642	31.1
Eigen	1024	0.0199	100.6
Naive	2048	12.2	1.3
Optimized	2048	0.675	23.7
Eigen	2048	0.243	65.9

Table 2: Performance without '-march=native'