**Project 7: Writing Cache Friendly Code**

CSCE 312: Computer Organization

Ryley Priddy

Section 504

April 15, 2021

In this project, we were asked to implement different forms of matrix multiplication by using 3 forms of loop ordering and cache blocking. By doing this, we were able to analyze the benefits of each form of loop and from cache blocking.

There are 6 types of loop orderings that can be implemented in matrix multiplication. These loops allow the user to vary the order of operations (either row-wise, column-wise, or location-wise) and decide how matrix elements are calculated to reach a more efficient algorithm or to avoid any possible data misses within an iteration. Within these 6 types, there are pairs that essentially offer the same method of operation, reducing the overall number of combinations to 3. Each of these combinations deal with two inputted matrices (A and B) and a result/product matrix (C). The first pair or iteration is the ijk/jik mode of operation. This pair iterates through j and i to go through a specific row (i) for A and a specific column (j) for B in order to find a fixed location to calculate the product for C (i,j). The difference between these two is that ijk iterates through i first and the jik iterates through j first. This pair is the most efficient and performs the fastest but has a high rate of misses at 1.25 per iteration. The next pairing is kij and ikj. This pairing uses a fixed location for A (i,k) and uses row-wise operations for B (row k) and C(row i) in order to find the product. To do this, two external for loops are formed for i and k, which are used to find the exact location for A. Then, another for loop is added underneath to iterate through j to calculate the product of the matrices and store it in C. This pairing has the lowest number of misses per inner loop. The final pairing is the jki/kji pairing. This pair iterates through j and k to go through a specific location (k,j) for B and a specific column (k) for A in order to complete a column-wise operation for C (j). This pairing is the slowest and has about 2 misses per inner loop iteration.

To help avoid these misses and to speed up our program, we were asked to implement cache blocking on the most efficient pairing (ijk/jik). To do this, we would need to parse out matrices into smaller submatrices to load directly into the cache. Once this submatrix is in the cache, normal multiplication operations are performed on it to implement both spatial and temporal locality. In Figure 1, an overall comparison can be seen between each looping method and the cache blocking method with a block size of 100 (the most efficient size tested). Due to segmentation faults in the code used to calculate this data, the largest input accepted are two 500x500 matrices.

*Figure 1: Comparison of Different Matrix Multiplication Methods to Size and Time*

The ijk/jik loop ordering method is clearly the most efficient based on the diagram. If larger sample sizes were able to be tested, the differences between the methods would have been much more distinct. However, the cache blocking method consistently had the lowest run time compared to all the other methods. A comparison between the operation times of different block sizes can be seen in Figure 2.

*Figure 2: Block Size Impact on Overall Run Time*

Based on the data collected for the different matrix sizes, the difference between the block size of 100 and 50 appear to be very close. However, as the size of the matrices increases, the block size of 100 will be more efficient due to the ratio between this value and the overall size. This gap can be seen with a block size of 5, which has a very low ratio in comparison to the other two sizes.

This project was a great way to get hands on experience writing cache friendly code. It allowed us to take an operation that we normally would not optimize and showed us all the possible methods to create a more efficient program. We also were challenged to trouble shoot on our own, research errors, and dive deep into the concepts taught in this project and in our course.