CS341 Cache Lab Part B

Ryan Scherbarth

December 2024

1 Case 1: 32×32 Matrix

The primary optimization for the 32×32 matrix relied on an 8×8 block size. Each block fits within a single cache line, so we have fewer cache misses due to row-major ordering. Additionally, diagonal elements were saved to the end of the block's processing to avoid unnecessary cache evictions. These two ideas together were enough to get below the threshold for full credit on this test case.

2 Case 2: 64×64 Matrix

The 64×64 matrix posed a more difficult challenge due to the larger size and the potential for increased cache conflicts. An initial implementation with a 4×4 blocking size still had too many cache misses. I tried a number of different ideas

- Increasing the block size to 8×8 and 16×16 to align better with cache lines.
- Reordering loop structure to improve spatial locality.
- Skipping diagonal elements and processing them after the block to minimize evictions.
- Using Asymmetrical block sizes, 8×4, for example

Regardless, I was still not able to get the full credit for this question. My final implementation for my best score is a little under 1,800 misses.

3 Case 3: 61×67 Matrix

For the irregular 61×67 matrix, I started by just using the same implementation from the first case where I iterate over a squared block size and save the diagonal values until the end. I went through a few other block sizes and found using the exact same implementation with a 16×16 block size met the full criteria for this problem, so I just left it at that.