Nick Fireman

Project 2

CSI 5360
February 10, 2020

1 Table of results

Mat. size	Opt. level	Algorithm	Trial 1	Trial 2	Trial 3	Average time
1024	O0	Naive	10.666	10.296	10.188	10.383
		Optimized	3.582	3.702	3.517	3.600
1024	O3	Naive	2.686	2.681	2.666	2.678
		Optimized	0.525	0.518	0.501	0.515
2048	O0	Naive	88.579	84.823	84.662	86.021
		Optimized	30.158	28.232	28.418	28.936
2048	O3	Naive	22.786	22.892	23.628	23.102
		Optimized	4.487	4.482	4.886	4.618

Table 1: Results of timed trials

2 Explanation of optimization

Three optimizations were used. The most effective optimization was to permute the two inner for loops. That is, instead of iterating in the order i, j, k, iteration is done in the order i, k, j. In this way, the innermost loop's memory accesses $A[i * N + k]$ on the matrix A increase consecutively with k, and so are contiguous in memory, which dramatically improves performance by reducing the frequency of cache misses. I found that this optimization alone was enough to get under the 6 second performance goal.

The second optimization used was to use a local array *temp_row* to store the temporary results for each row of the result matrix as it is being computed. Once the row is fully computed, *temp_row* is then copied to C all at once. Writing to C, which is non-local, is costlier than writing to *temp_row*, so only writing the final result of each row into C is an improvement.

Finally, the third optimization used was the pre-computation of offsets $a\_offset = i * N$ and $b\_offset = k * N$ into the arrays A and B, respectively. Instead of computing $i * N$ and $k * N$ at each iteration of the innermost loop, they are computed once at the beginning of the i and k loops by adding i and k to a_offset and b_offset, respectively. Additionally, b_offset must be reset to 0 at the start of a new loop over k to realign with $k = 0$. This saves some redundant integer arithmetic, although it did not have a significant impact relative to the other two optimizations.