Daniel Nakashima => cs61c-hi

Chung Fai Tsim => cs61c-eh

Matrix Multiply - Project 3 - Writeup Part 2

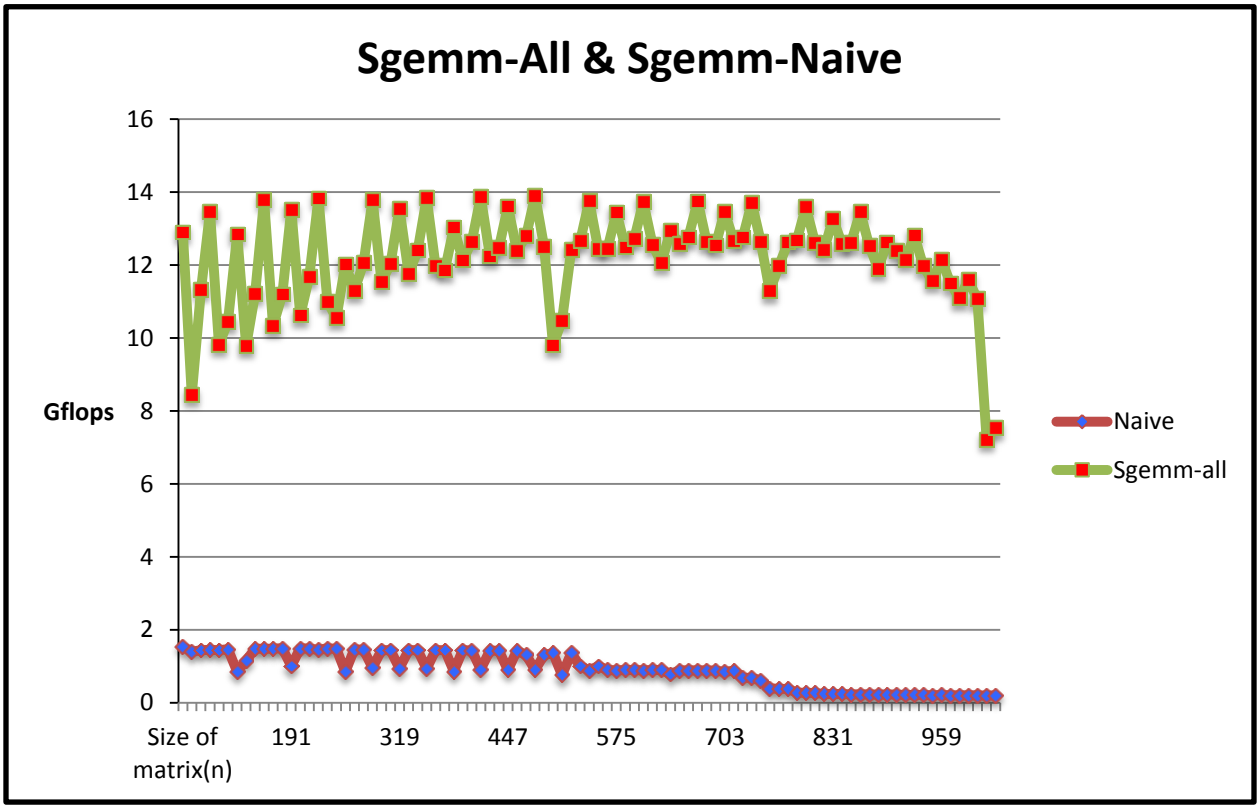
1. **A brief description of any changes you made to your code from part 1 in order to get it to run well on a range of matrix sizes (max 150 words)**

   In our first submission, in the event that input size of the matrix was not a multiple of 16, we handled this naively with a single intrinsic and purely naively for the case for the fringes. In our sgemm-all submission, our approach was to "pad" our input matrix size N that was not divisible by 16 to the ideal size. IE, N=N/8*8 + 8. To do this, we iteratively pad the each of the A ,B and C matrices with zeros and from there perform the same operations that we did for sgemm-small i.e. ideal sizes. This especially improved the performance for the matrix sizes that are not divisible by 8.

2. **A brief description of how you used OpenMP pragmas to parallelize your code in sgemm-openmp.c (max 150 words).**

   In sgemm-all.c , the nested loop are in the order of k, j, i. We stored the result of C= C+A*B back in the most inner loop i according to C = k+j*n. Since we switched to use openmp, we need to split each of the work equivalently into 8 cores. So what we need to do is switch the loop order of k and j. In our sgemm-openmp.c, we have the loop order j, k, i. For example, if N=64 and we have 8 cores, then cpu0 is calculating the result and store in the column 0-7 in Matrix C, and cpu7 is calculating the result and store in the column 56-63 in Matrix C. We set every variable we used inside the nested loop to be private except matrix size N.

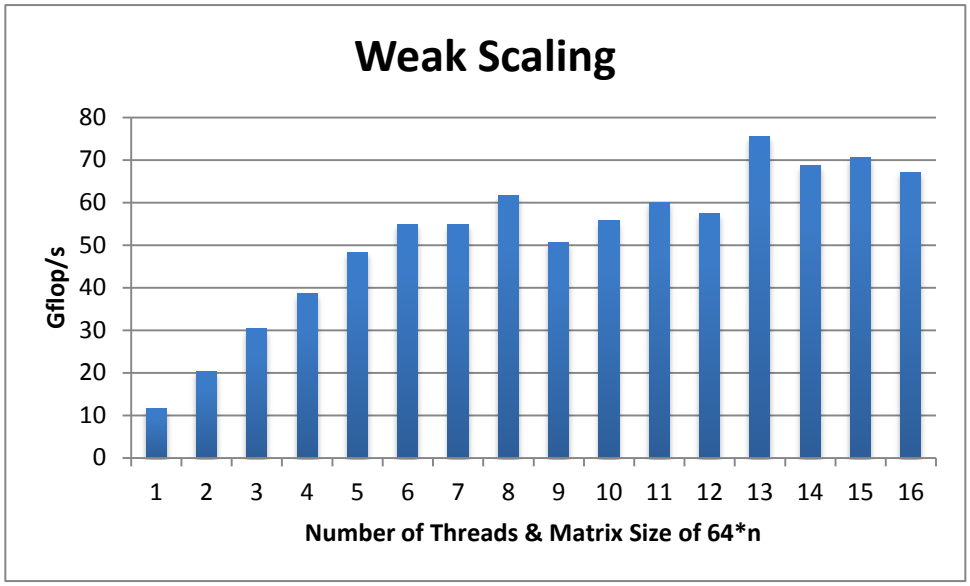
3. **A plot showing the speedup of sgemm-all.c over sgemm-naive.c for values of N between 64 and 1024**



4. **A weak scaling plot of the performance of your sgemm-openmp.c code (use your sgemm-all.c code as the baseline for the single threaded case)**

   Number of threads range = 1-16

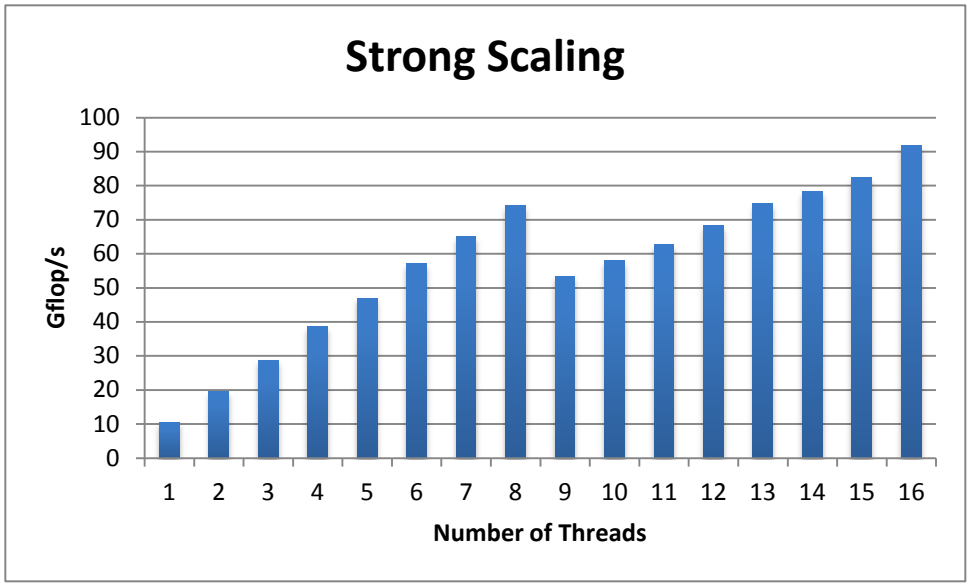
   Matrix Size => multiples of 64 i.e. 2 threads ran a matrix size of 2*64 = 128



5. **A strong scaling plot of the performance of your sgemm-openmp.c code**

   Used fixed matrix size n = 256;

   Number of threads range = 1 - 16;


Strong Scaling
Gflop/s
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Number of Threads