1st project of Parallel and Distributed Systems

Giachoudis Christos

20/11/2021

Triangle counting in sparse graphs

SEQUENTIAL AND PARALLEL ALGORITHMS

We attempted to write down some code, to count the number of triangles adjacent with each node in an undirected, unweighted graph.

We did that by using the following formula:

C = A @ (A*A)

Where @ denotes the Hadamard product.

For Hadamard product see: https://en.wikipedia.org/wiki/Hadamard_product_(matrices)

- First: we had to find a way to read/import a coo format matrix and then convert it to the csc format. Such matrices can be found at: https://math.nist.gov/MatrixMarket/

That's what we did at the coo_2_csc.h file, where we encountered some problems:

Firstly, we had to just read the matrix and its parameters

Second, the matrix was given at a lower triangular format. The good news were that the matrix was a symmetric one. So we had to create a symmetric out of a lower triangular.

And then we convert the matrix to the CSC format. For the csc format see:

https://en.wikipedia.org/wiki/Sparse_matrix#Compressed_sparse_column_(CSC_or_CCS)

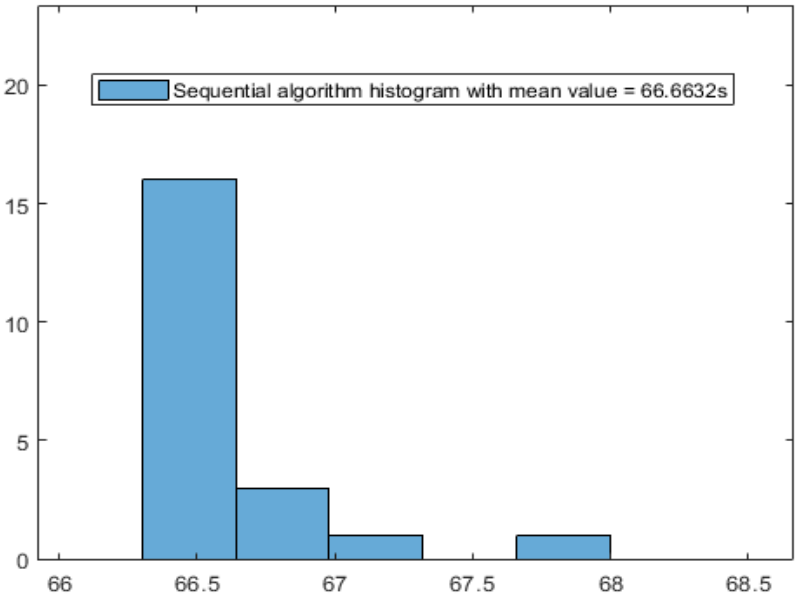
- Second: we had to create the sequential code to achieve the multiplication. At first we tried to create a full multiplication of every row with every column. That was huge and needed 4 for loops only for the A*A multiplication. After that we thought that we do not need to do all those multiplications due to the Hadamard product. So we multiplied only the rows and the columns that were going to be nonzero after the Hadamard multiplication. And that's what seq_mul.h file stands for.

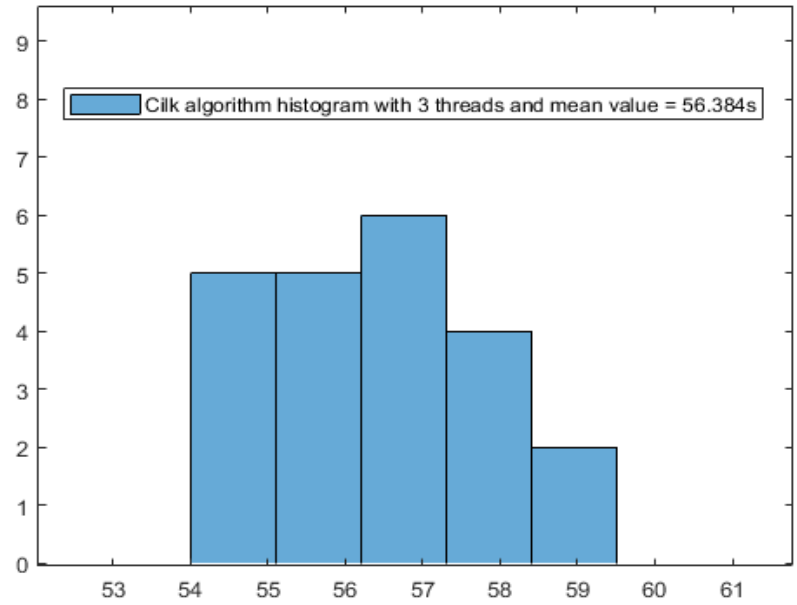
- Third: The parallelization part was a little bit tricky. The code had to be changed, because of racer errors. Each thread was trying to write its results to the same memory location. And that gave us a segmentation fault. So we changed the code and used 3 methods(OpenCilk, OpenMP, PThreads) to parallelize the multiplication algorithm. That was done by using the following files: cilk_mul.h, mp_mul.h, pthreaded.h
- As a final step we measured the algorithms' running times and we imported our samples into MATLAB to run some statistics. The test results are shown below:

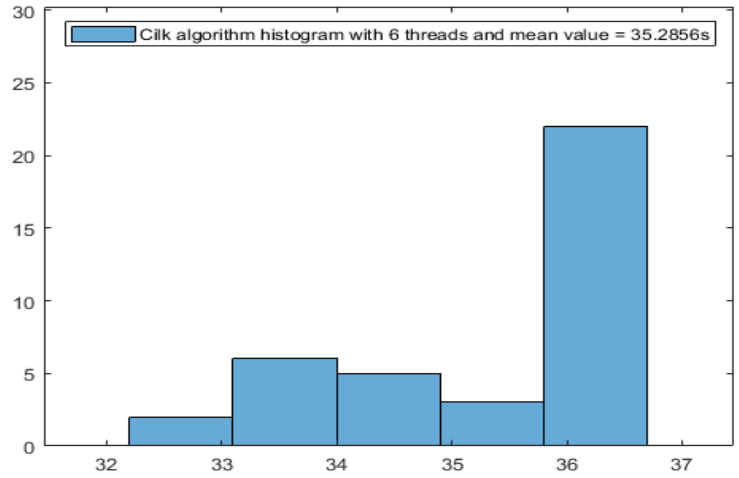
Notes: The matrix we use: https://sparse.tamu.edu/SNAP/com-Youtube

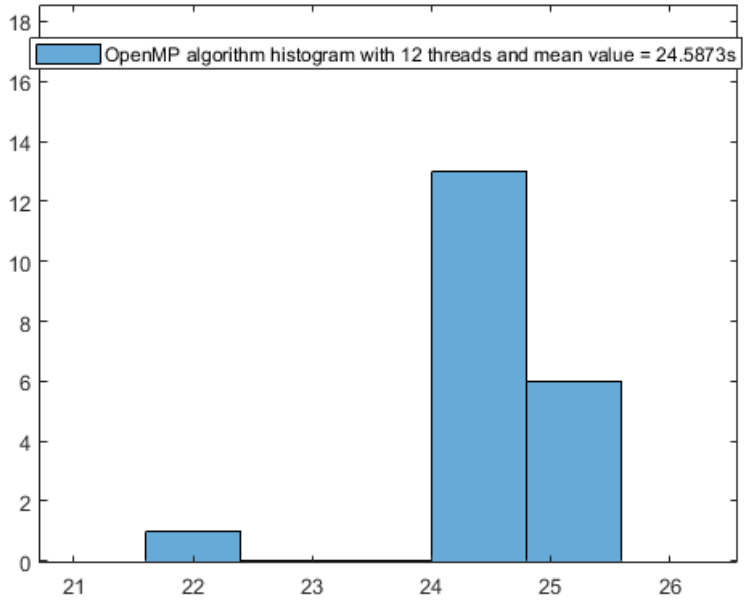
The file that conatins the pthread code exists, gets compiled, runs, but the times that we measure are not better than the sequential algorithm. There may be a logical error, or not suitable code for the specific method. Either way I just note that the matrix above was multiplied in aprpximatelly 1 minute and 40 seconds.

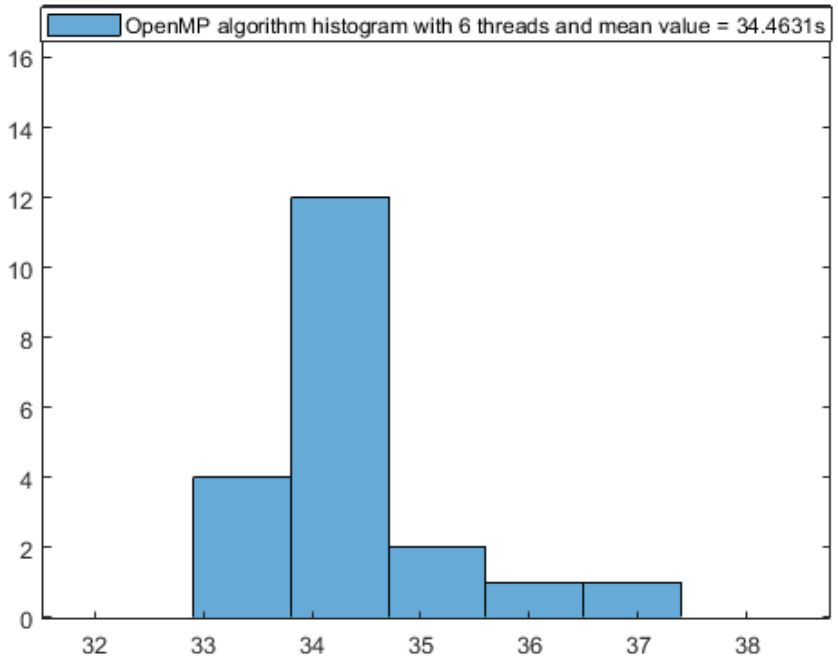
The code can be found here: https://github.com/chrigiach/Parallel-and-Distributed-Systems-1


Sequential algorithm histogram with mean value = 66.6632s
20
15
10
5
0
66
66.5
67
67.5
68
68.5


Cilk algorithm histogram with 3 threads and mean value = 56.384s
9
8
7
6
5
4
3
2
1
0
53
54
55
56
57
58
59
60
61


Cilk algorithm histogram with 6 threads and mean value = 35.2856s
30
25
20
15
10
5
0
32
33
34
35
36
37


OpenMP algorithm histogram with 12 threads and mean value = 24.5873s
18
16
14
12
10
8
6
4
2
0
21
22
23
24
25
26


OpenMP algorithm histogram with 6 threads and mean value = 34.4631s
16
14
12
10
8
6
4
2
0
32
33
34
35
36
37
38