Cpt S 411 Assignment Cover Sheet

(To be turned in along with each homework and program project submission)

Assignment # 2

For individual assignments:

Student name (Last, First): Barik, Reet

For team projects:

List of all students (Last, First): Barik, Reet

List of collaborative personnel (excluding team participants):

I[1] certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received or given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment. I also certify that I have not referred to online solutions that may be available on the web or sought the help of other students outside the class, in preparing my solution. I attest that the solution is my own and if evidence is found to the contrary, I understand that I will be subject to the academic dishonesty policy as outlined in the course syllabus.

Please print your names.

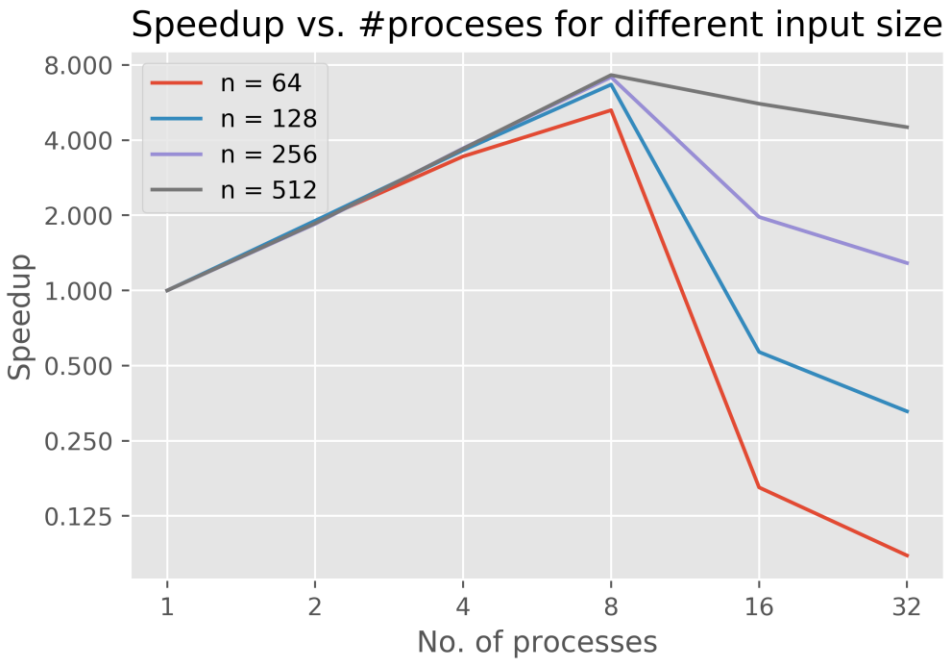
Assignment Project Participant(s): Barik, Reet

Today's Date: 8 October 2020

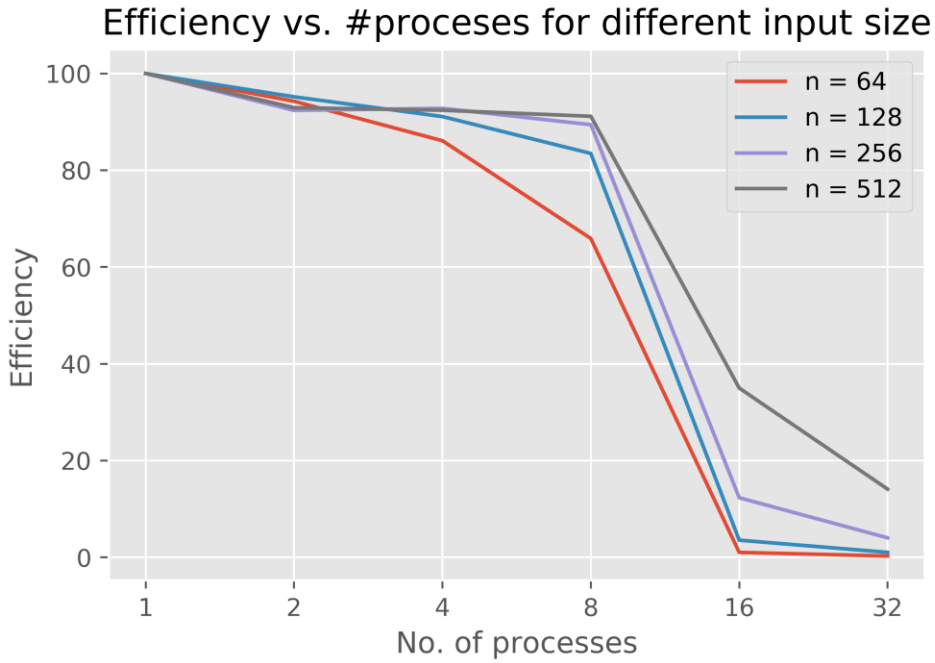
[1] If you worked as a team, then the word "I" includes yourself and your team members.

The required plots are as follows:

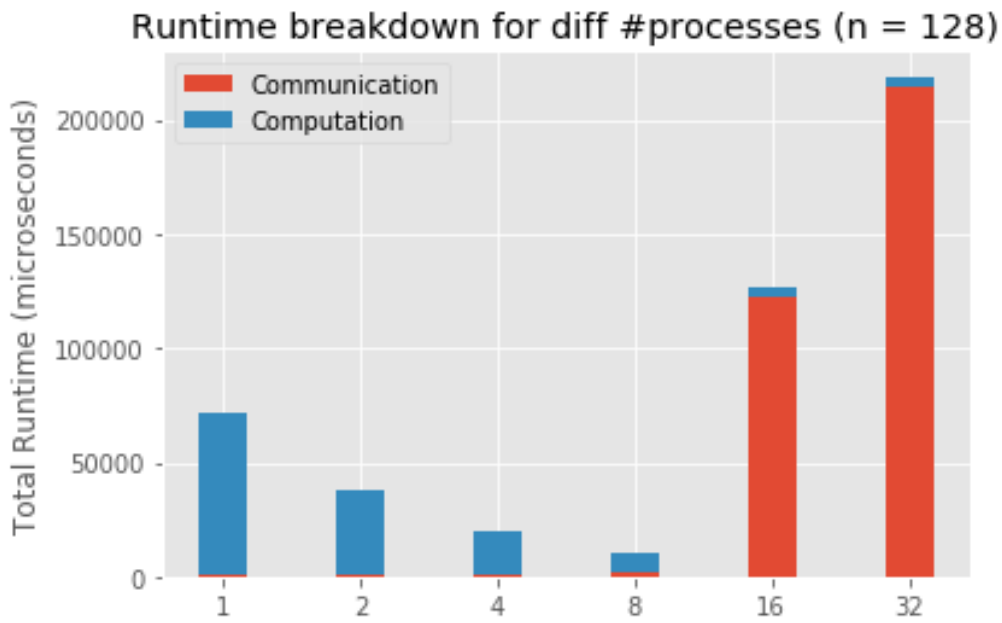
1. Speedup:



2. Efficiency:



3. Breakup of Communication and Computation time in the total runtime



Interpretation:

I ran the code using the command '*sbatch -n <#processes> sub.sh'*.
This results in my code scaling well till #processes = 8. This is because each node has 8 processors and hence can run at most 8 processes without incurring significant slowdown due to communication. As soon as #processes go above 8, it requires at least 2 nodes. Which is where communication becomes a bottleneck (as can be seen from the chart showing the breakup between communication and computation time).

If I had run my code using the command '*sbatch -N <#nodes> -n <#processes> sub.sh'*, then I definitely would have had the communication bottleneck from #processes = 2 onwards. And I would not have seen any speedup.

Possible fix:

1. I am using 'int' as the datatype to store the matrices. I should use 'bool' instead and hence the communication load should ideally be reduced to 25% of what it is right now.
2. Also, the way my code is set up right now, I am adding up the values of all the neighboring cells and then checking whether the current cell should live or die based on the sum. The 4 byte integer additions might be done away with if we are using bool and bring down the computation time further: have an 8 bit array for each cell we are considering (each position corresponding to the 8 neighbors). The number of ones in the array (Hamming Weight) can be calculated by bitwise operations (a combination of bitwise AND and right shift). This is significantly cheaper than integer additions.