Cpt S 411 Assignment Cover Sheet

Assignment – Programming Project 03

Submitted by: - Amandeep and Piyush Garewal

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.

Assignment Project Participant(s): Amandeep, Piyush Garewal.

Today's Date:
Nov-09-2018

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

Introduction:

In this programming assignment we have to implement Conway game of life in MPI. It is a cellular automation game where each side is of "n" i.e. it is a square matrix. Each cell has its own state living or dead which is determined by following the provided rules.

Objective:

To calculate the total computation time and communication time and plot the speedup graph with number of process and find the efficiency with respect to number of processors.

Methodology:

To implement the Conway game in MPI first we divide the complete given square into smaller chuck size for each processor. The state of each cell depends upon the number of living neighbors, so each process will have the idea about number of cells for each processor (i.e. chuck size) plus the corresponding neighbor's information to determine the state of each cell.

(NW)	(N)	(NE)
(W)	cell	(E)
(SW)	(S)	(SE)

The above the possible 8 neighbors for each cell.

For each process a submatrix will store the value of total number of live neighbors. So that we can determine the state of cell.

To divide the given square matrix into a submatrix

Chuck size		

For each processor the chuck size + 2 * size of row = submatrix.

Then using MPI_Sent() to distribute the data randomly generated in rank 0.

And each MPI_Received() value to corresponding ranks, then seeds and fills its submatrix with the random values.

Special consideration has been taken for the top row and bottom row.

Data:

Speedup in the concept that a program will run faster if it is implemented in parallel, for this program we have given number of process should be smaller than size of array.

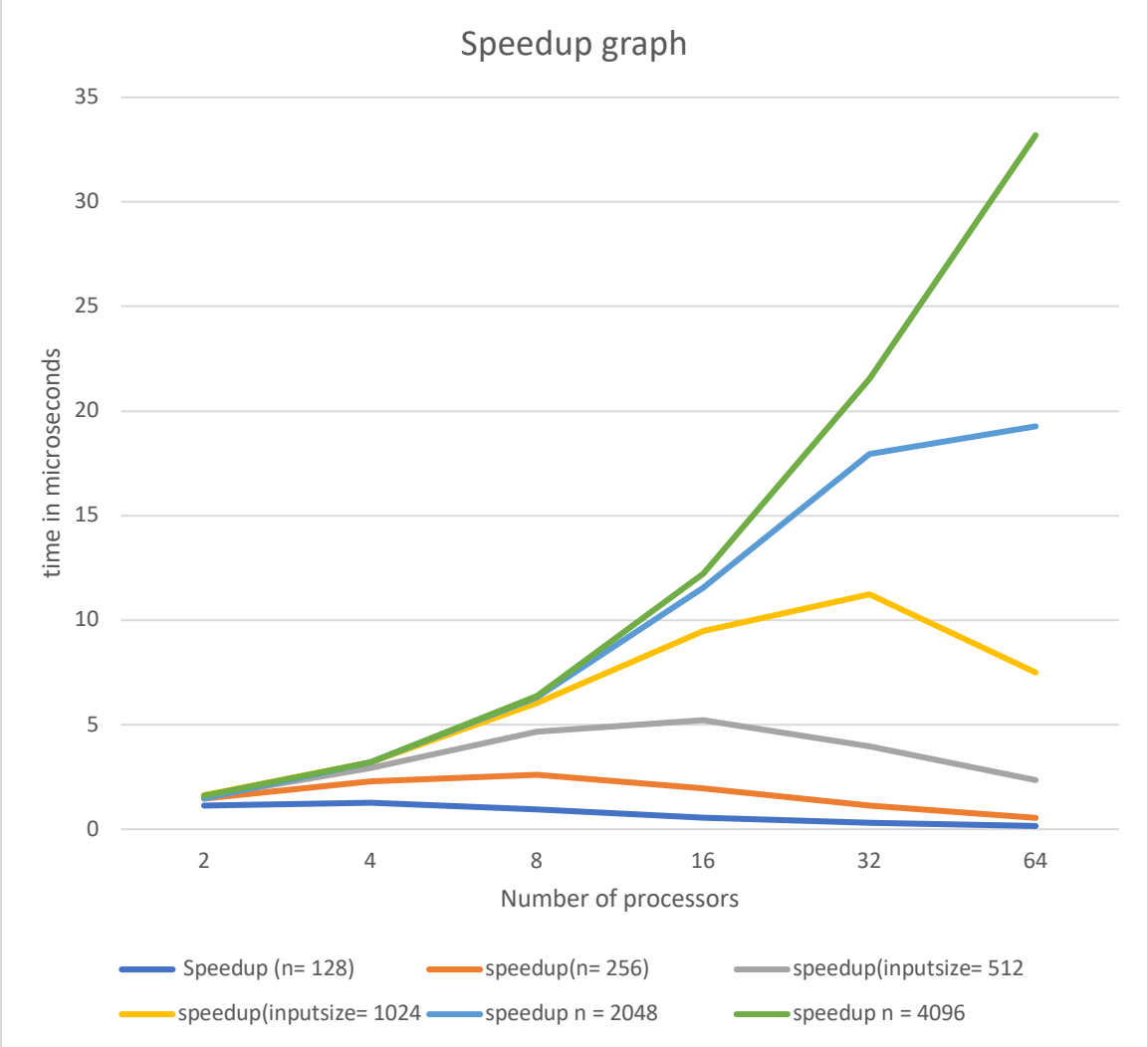


Fig 01: Speedup graph

Observation:

Here the above graph is showing the speedup with increasing number of processors, initially with small value of (n) as the computational time is more the speedup is not ideal for values like 128, 256, 512 but when the value of (n) is increased to 1024 and higher, computational time in less hence the speedup is showing the ideal progress.

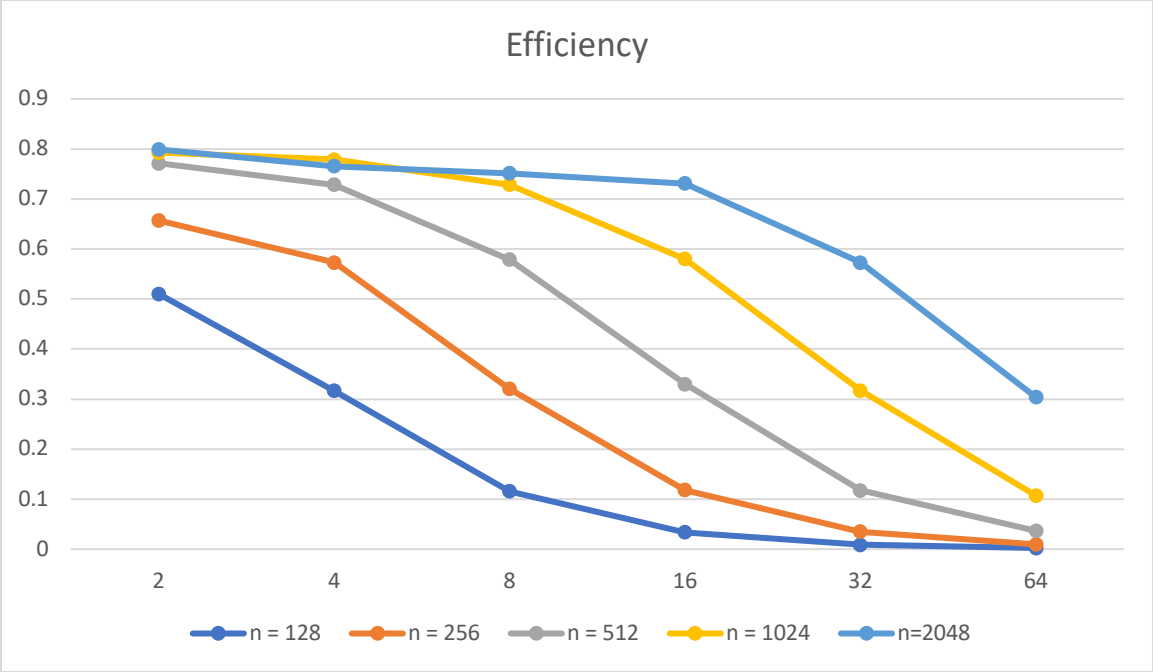
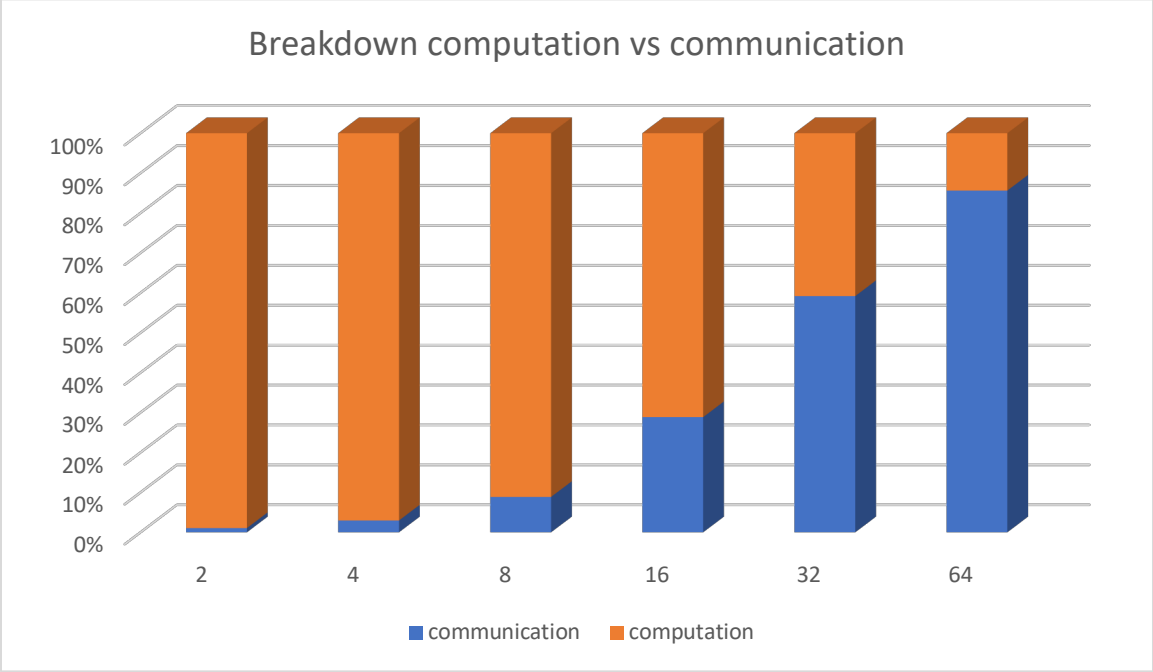


Fig02- efficiency



(Here n 1024).

Observation: From the above graph, keeping the "n" as constant 1024 and with number of processors is increasing the communication time is increasing and the computation time is decreasing as it was supposed to happen. In this program the time taken for the

communication keeps on increasing as with increasing number of processors the task or the work is divided into number of processors to the computation time is decreasing.