CENG 478 INTRODUCTION TO PARALLEL COMPUTING
HOMEWORK4 REPORT ON CONVEY'S GAME OF LIFE

1.Theoritical computations

1.1 Communucation Time

In this assignment, since game area is divided by 2D, each processor have a cell amount of [(length of game area(l))/(sqrt(num proccessors(p)))]^2. At each step each processors send their boundary areas of lenth l to at most 4 neighbour processors. Assuming transmitting a shortdata(2 byte) takes a time of tm (I used short for value of cells in the code) , taking number of turn that code runs as nrun, total communication time is then ,

Tcomm = O(4(ts + l/sqrt(p)*tm) * nrun)

1.2 Computation Time

Assuming deciding whether a cell will live , born or die takes a computation amount of tc, we have [(l/sqrt(p))^2] amount of cell for each processor. Thus computation time is then ;

Tcomp = Q((l/sqrt(p))^2 * tc *nrun)

1.3 Speed Up

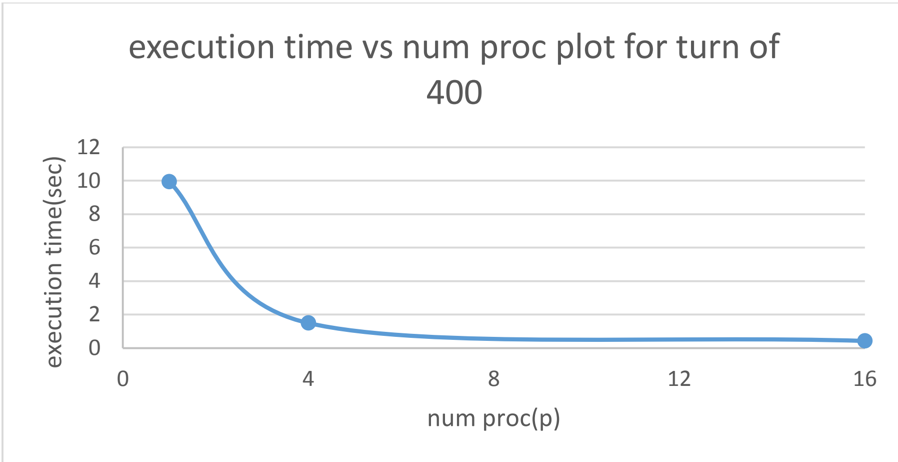
For a single processor computation time is simply number of cell times tc times nrun. Thus Tserial = Q(l*l*nrun*tc), then spped up is;

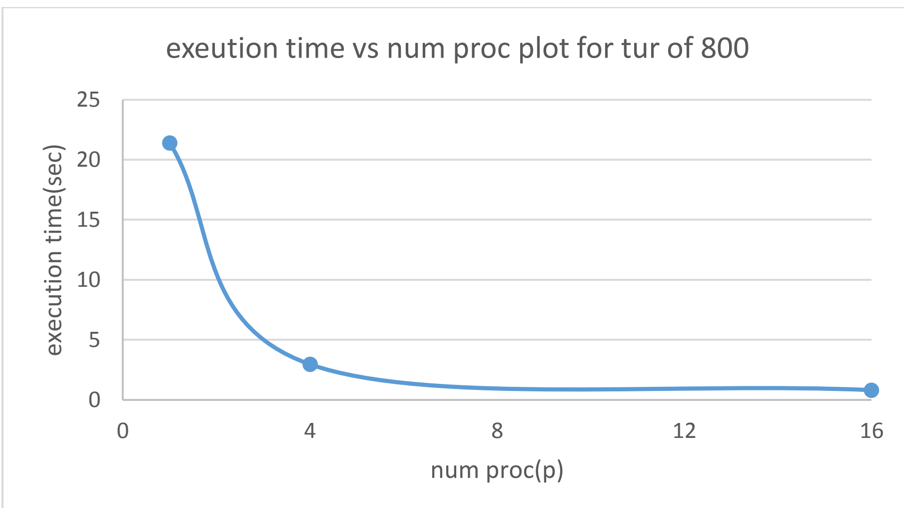
$$S = \frac{O\left(4\left(ts + \frac{l}{sqrt(p)} * tm\right) * nrun\right) + Q\left(\left(\frac{l}{sqrt(p)}\right)^2 * tc * nrun\right)}{Q(l * l * nrun * tc)}$$

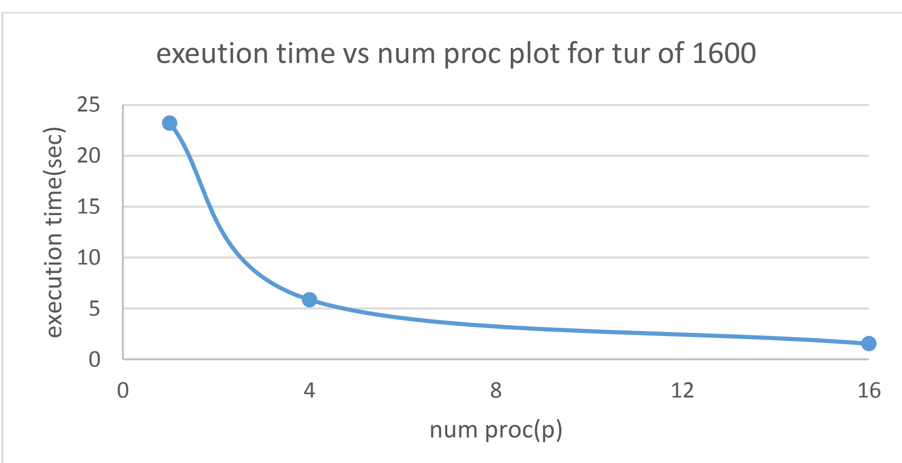
2. Execution Results

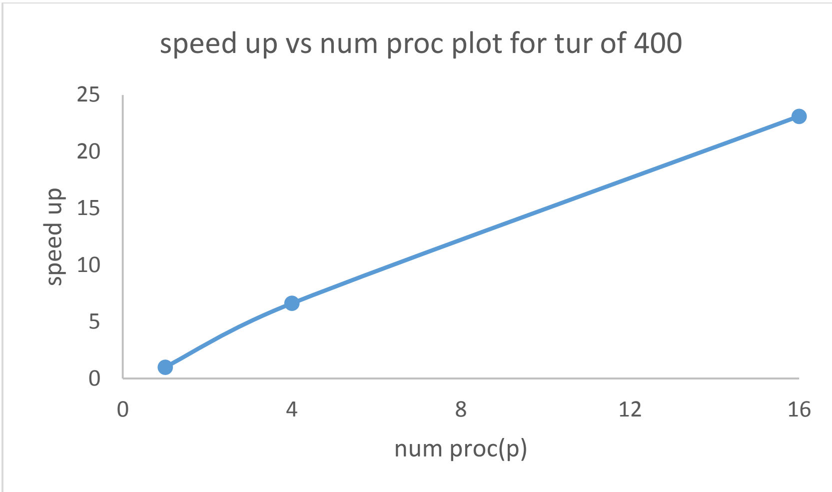
Num proc	Num turn	Execution time	Speed up
1	400	9.940395	**1**
1	800	21.388816	**1**
1	1600	23.199960	**1**
4	400	1.500300	**6.62**
4	800	2.952888	**7.24**
4	1600	5.857924	**3.96**
16	400	0.431866	**23.11**
16	800	0.804355	**26.59**
16	1600	1.542309	**15.05**

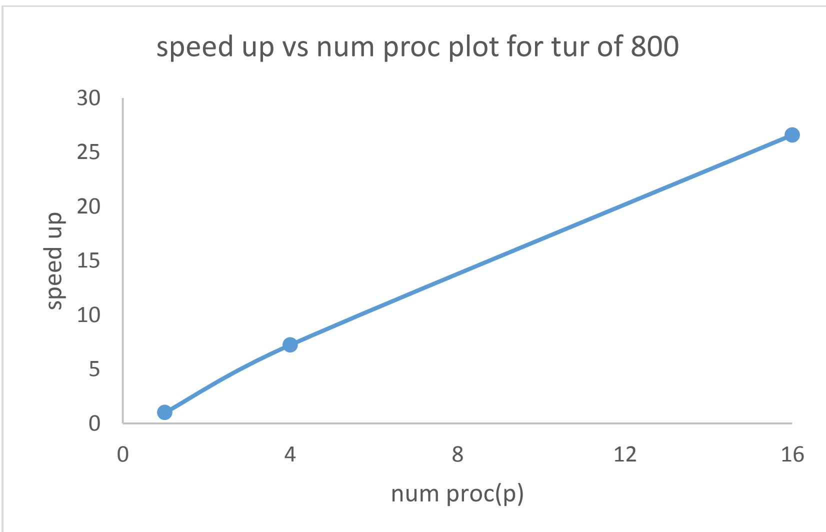
Below graphs shows the time consumed vs number of processors and speed improvement vs number of processors for turn number of 400, 800 and 1600 each.

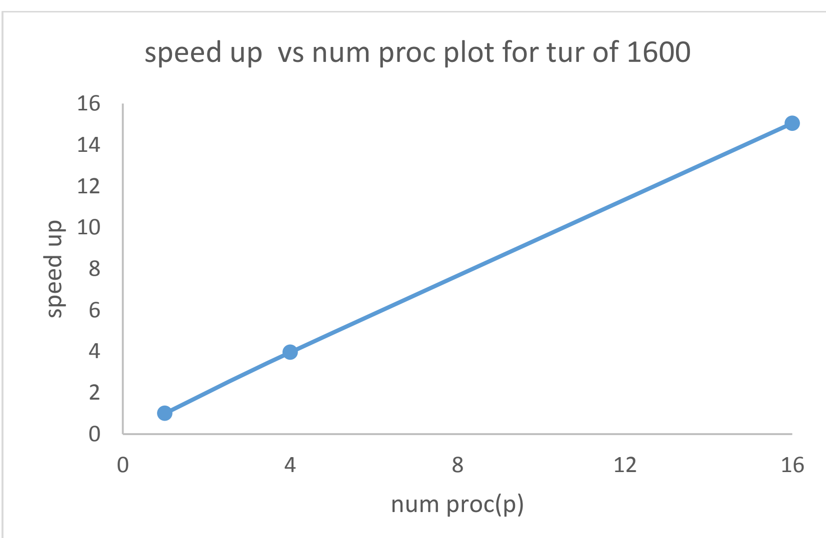

execution time vs num proc plot for turn of 400
execution time(sec)
12
10
8
6
4
2
0
0
4
8
12
16
num proc(p)


exeution time vs num proc plot for tur of 800
execution time(sec)
25
20
15
10
5
0
0
4
8
12
16
num proc(p)


exeution time vs num proc plot for tur of 1600
execution time(sec)
25
20
15
10
5
0
0
4
8
12
16
num proc(p)


speed up vs num proc plot for tur of 400
speed up
num proc(p)
0
5
10
15
20
25
0
4
8
12
16


speed up vs num proc plot for tur of 800
speed up
num proc(p)
0
5
10
15
20
25
30
0
4
8
12
16


speed up vs num proc plot for tur of 1600
speed up
num proc(p)
0
2
4
6
8
10
12
14
16
0
4
8
12
16

2.1 Comments on Results

As the number of processors increases, speed up increases almost linearly in each number of turns of run. This is because, as the number of processors increases , communication overhead does not increase significantly because we send and receive only borders columns or rows. We note here that when number of processor is equal to 16 at turn = 800 , we have a speed improvement of 26.6 which is bigger than number of processors. This may because of increasing total cached data amount as the number of processors increases (due to increase in total cache size). In this case efficiency is bigger then 1. In general, efficiency is nearly one since speed improvement is linear in terms of number of processor (E = S * p)

If we had a more complex initial situation, we had to initialize all subareas of the game area belonging to distinct processors separately. In our case, this is only processor_0. Communication and execution time would not change according to my code. Because I check each cell at each turn and send border rows-columns to neighbor processor without checking its contents.

MUSTAFA ERAL

1676535