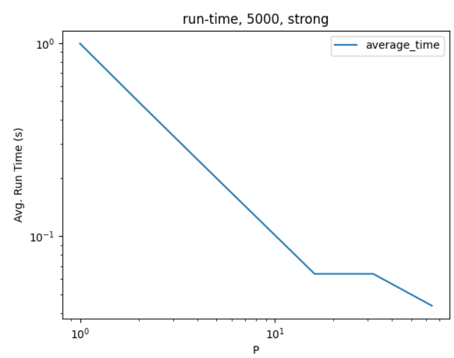
Project 3 Analysis

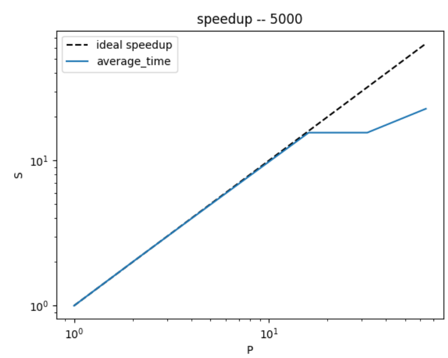
Brian Lee

5 November, 2021

1 Strong Scalbility Study



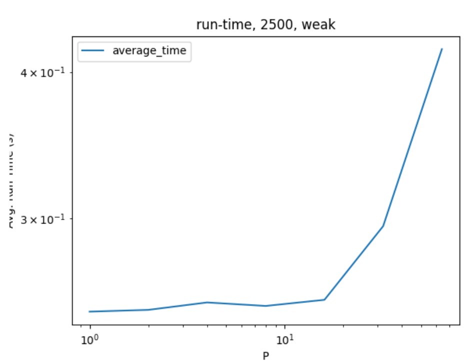
(a) Run-time plot



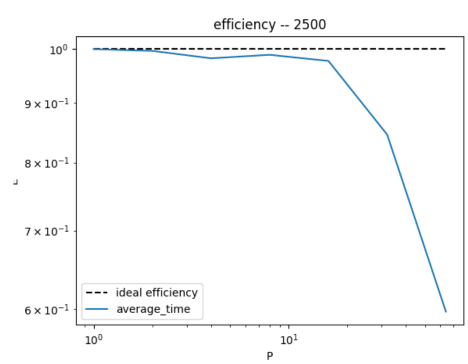
(b) Plot on speedup

This problem exhibits good strong scalability because as we increased the number of processors, the run-times were roughly cut in half.

2 Weak Scalability Study



(a) Run-time plot



(b) Plot of efficiency

This problem exhibits poor weak scalability because as we increased the number of processors, the average run-time was close to constant or even increased.

3 Average Run-time by Grid Size

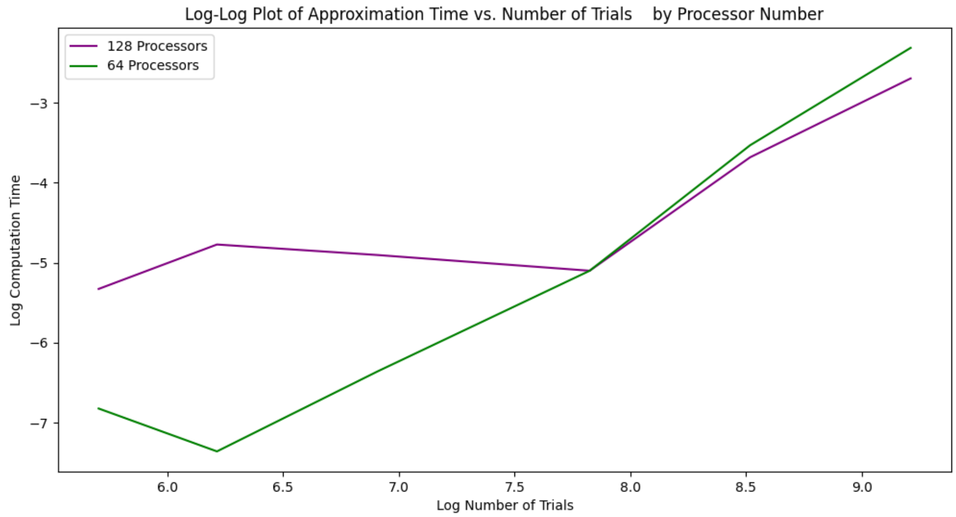


Figure 3: Time of computation plotted against grid dimension.

- Seeing as though the relationship between number grid size and computation time is roughly linear,
  - For P = 64 processors on a 100,000 by 100,000 grid:

$$35.98sec \times \frac{100,000}{20,000} = 179.9sec \tag{1}$$

  - For P = 64 processors on a 100,000 by 100,000 grid:

$$23.57sec \times \frac{100,000}{20,000} = 117.9sec \tag{2}$$

The following are tables of times of execution.

Grid size (n x n)	Average run-time (seconds per iteration)
n = 300	0.0010819sec
n = 500	0.000637sec
n = 1000	0.001731sec
n = 2500	0.006096sec
n = 5000	0.02929sec
n = 10000	0.098967sec
n = 20000	0.359857sec

Table 1: A table of average time of executions based on a given size, performed in parallelized C. 64 PROCESSORS USED.

Grid size (n x n)	Average run-time (seconds per iteration)
n = 300	0.004958sec
n = 500	0.008465sec
n = 1000	0.007422sec
n = 2500	0.006098sec
n = 5000	0.025192sec
n = 10000	0.067529sec
n = 20000	0.235673sec

Table 2: A table of average time of executions based on a given size, performed in parallelized C. 128 PROCESSORS USED.

We can compare these to the times achieved in sequential C and sequential Python.

Grid size (n x n)	Average run-time (seconds per iteration)
n = 10	0.0sec
n = 25	0.0sec
n = 50	0.0001sec
n = 100	0.0007sec
n = 200	0.003sec
n = 300	0.0068sec
n = 500	0.0185sec
n = 1000	0.0747sec
n = 2500	0.4654sec

Table 3: Table of average run-times based on a given size, in sequential C

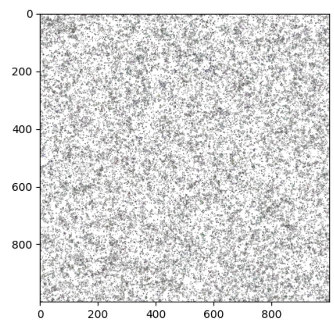
Grid Size (n x n)	Time (seconds per iteration)
n = 10	0.043sec
n = 25	0.078sec
n = 50	0.12sec
n = 100	0.1855sec
n = 200	0.3511sec
n = 300	0.5493sec

Table 4: Table of average run-times based on a given size, in sequential Python

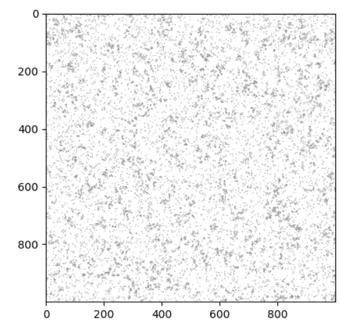
As can be seen, runtimes of the same probability and size can have drastically reduced run-times if parallelized properly.

4 States, 1000 × 1000

- For this problem, I chose to use 64 OpenMP threads because for a small-enough grid size, the average run-times were actually greater than using 128 threads. Perhaps this could be because each core of 64 processors must communicate the updated shared data with each other, lengthening run-time.
- This problem does not converge to a steady state by 1000 iterations.



(a) 100th iteration



(b) 500th iteration

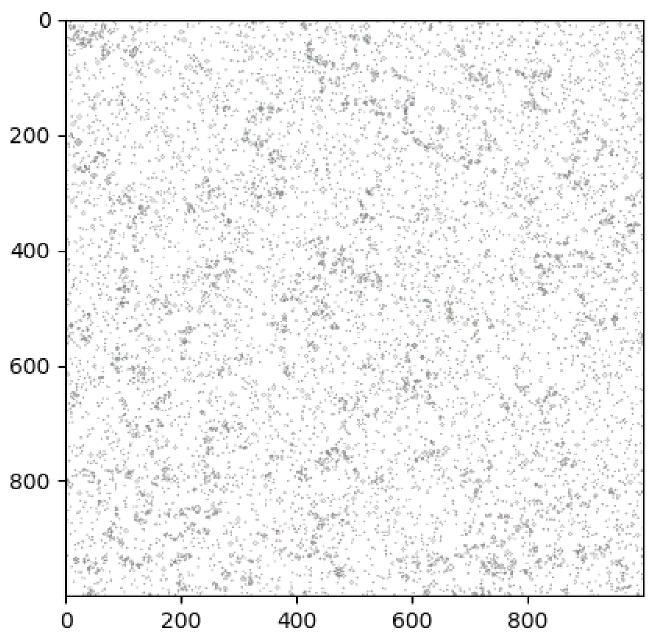
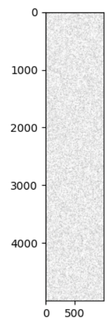


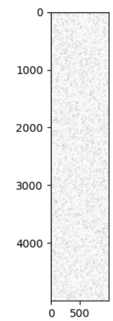
Figure 5: 1000th iteration

5 States, 5000 $\times$ 1000

- For this problem, I chose to use 128 OpenMP threads because for a larger grid size, the average run-times were improved from using twice as many threads. Perhaps this could be because the communication times between the two cores is smaller relative to the added computation time of a larger grid.
- This problem does not converge to a steady state by 1000 iterations.



(a) 100th iteration



(b) 500th iteration

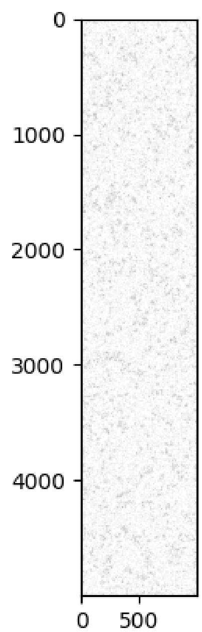


Figure 7: 1000th iteration

Collaboration

Classmates given assistance to:

- Stephanie Balarezo
- Carmin Berberich
- Loveish Sarolia
- Sreekar Singnam

Classmates assistance received from:

- Stephanie Balarezo
- Carmin Berberich
- Loveish Sarolia
- Sreekar Singnam