CONWAY'S GAME OF LIFE

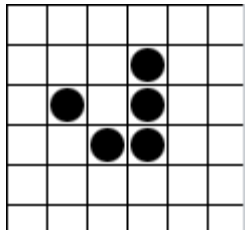
Varun Sudarshan

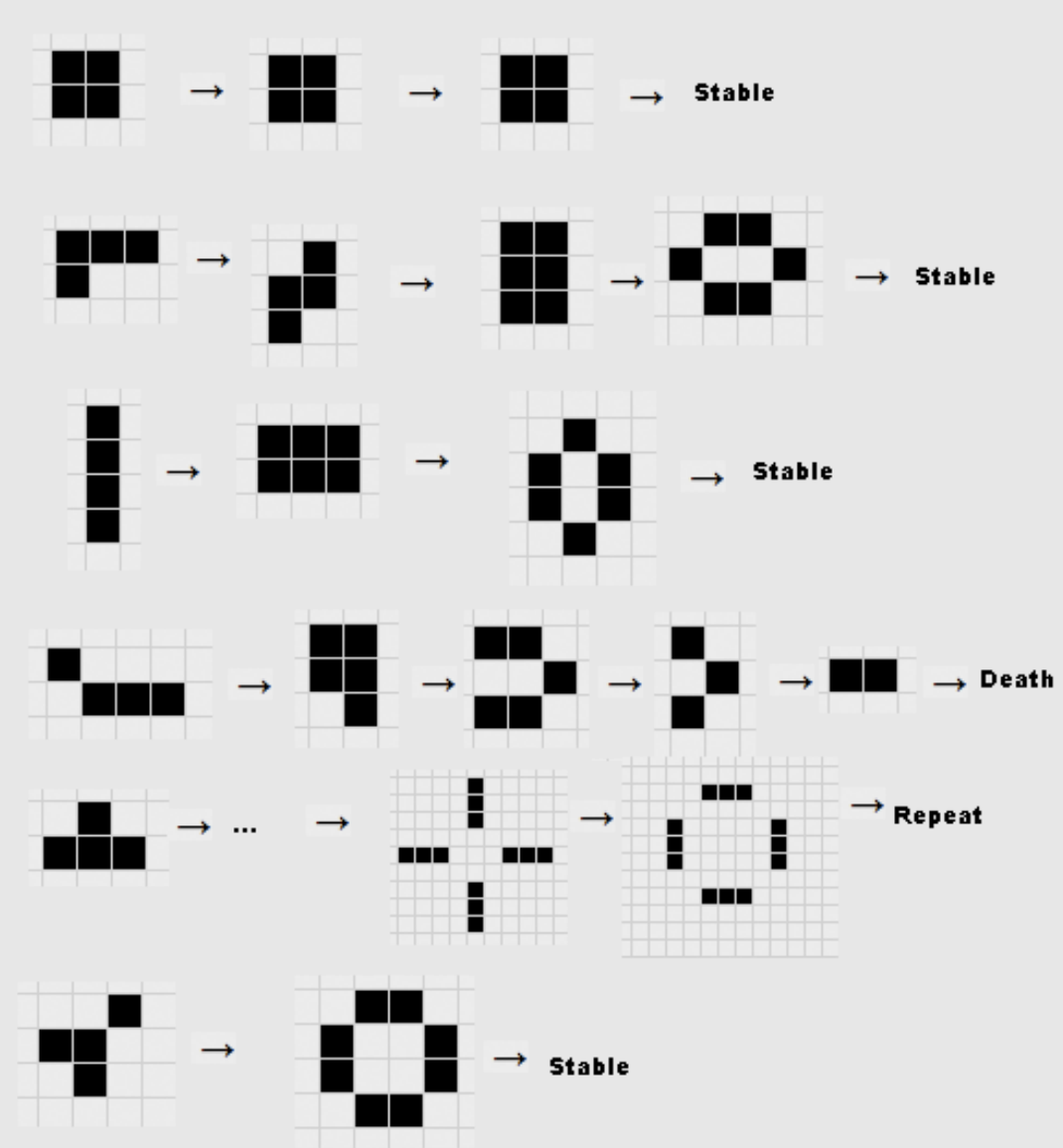
CSE 633 Parallel Algorithms

University at Buffalo The State University of New York

Game of Life

- You start with a pre-set pattern
- There are certain rules that define how the pattern evolves
  - Check for Overcrowding
  - Check for Loneliness
  - Check for New life





http://pi.math.cornell.edu/~lipa/mec/lesson6.html

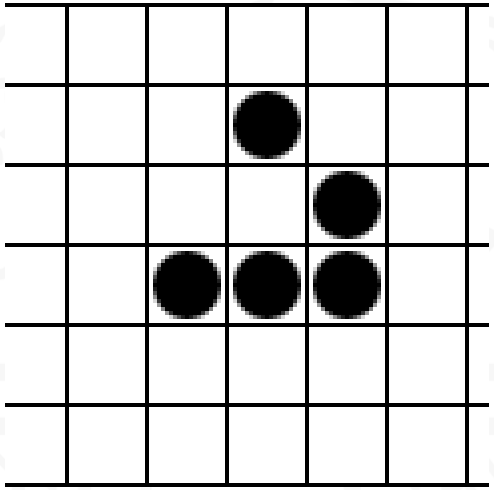
Let and

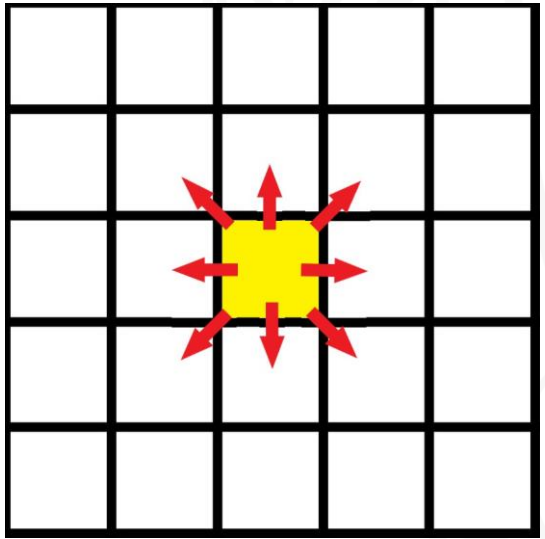
Algorithm

1. Any live cell with two or three live neighbors survives.
2. Any dead cell with three live neighbors becomes a live cell.
3. All other live cells die in the next generation. Similarly, all other dead cells stay dead.

On a sequential processor, we would traverse across the grid, look at the neighbours of each cell and apply the 3 rules, one by one, to each cell

Each cell of the matrix is dependent on its 8 immediate neighboring cells.



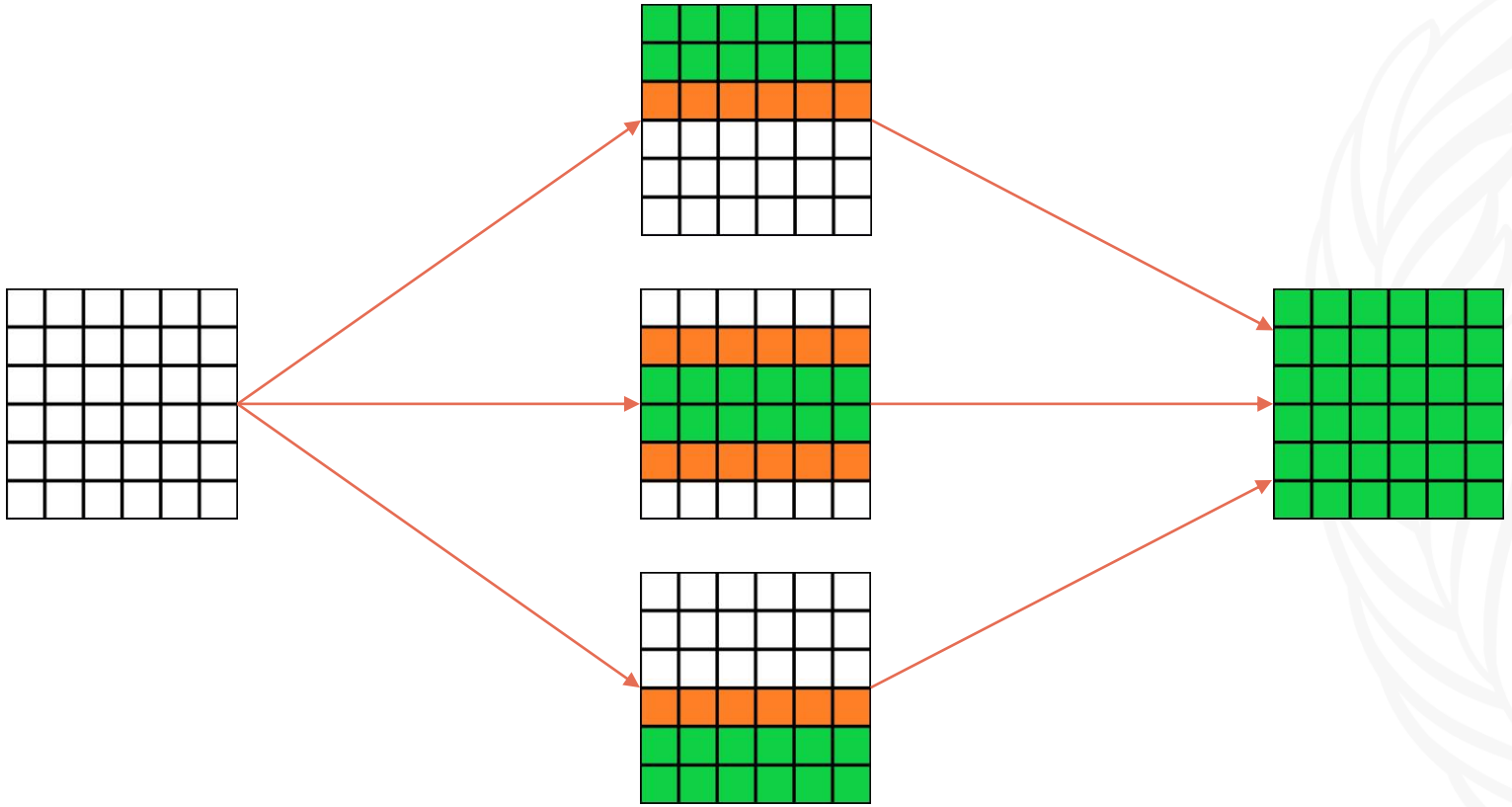


Parallel Implementation

- We decide how much data each processor takes based on the number of nodes available
- We divide the grid into smaller chunks
- To equally divide the data among all the processors, we divide the grid into (grid size/No. of processors) sized sub-grids
- For each sub-grid, we run the algorithm sequentially and pass the data back to the root node

Parallel Implementation

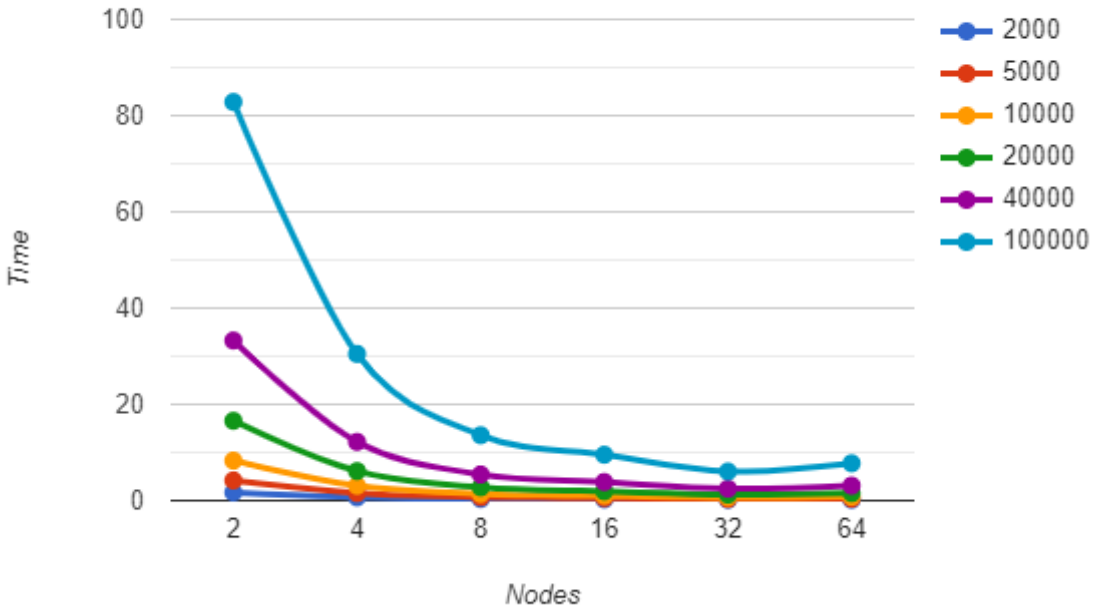
Since the state of a cell is dependent on the immediate neighbors,
we need to send the first and the last rows of a sub-grid to its previous and next node respectively



Results

Nodes

Generations: 128x128 Grid	2	4	8	16	32	64
2000	1.69229	0.671716	0.33108	0.256694	0.159455	0.171978
5000	4.18826	1.55547	0.718589	0.529803	0.331439	0.404831
10000	8.31188	3.08018	1.38794	1.00523	0.63995	0.784497
20000	16.5857	6.13125	2.76894	1.94491	1.23622	1.55557
40000	33.2702	12.2025	5.42279	3.86277	2.47809	3.10878
100000	82.8432	30.4821	13.5879	9.53127	6.05105	7.75544

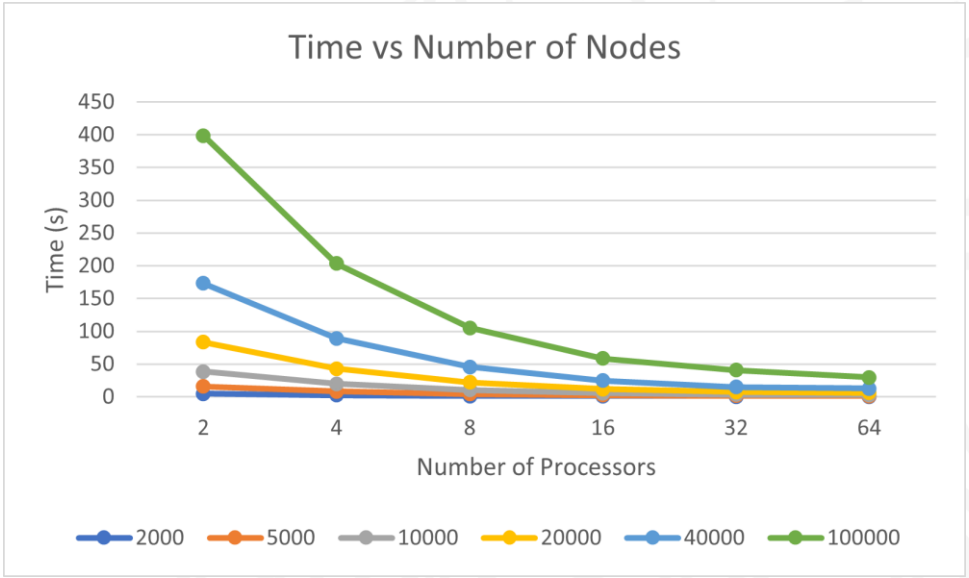


Results

Nodes

Generations

256x256 Grid	2	4	8	16	32	64
2000	4.63722	2.36425	1.25399	0.717133	0.471025	0.414976
5000	11.3749	5.85455	3.00758	1.65677	1.06862	0.884227
10000	22.4962	11.6196	6.06067	3.3433	1.95591	1.69642
20000	44.9724	23.0864	11.8386	6.3225	3.85505	3.29994
40000	89.7803	46.0753	23.713	12.7974	7.6951	6.52095
100000	225.479	114.506	59.466	33.7538	25.6513	16.6905

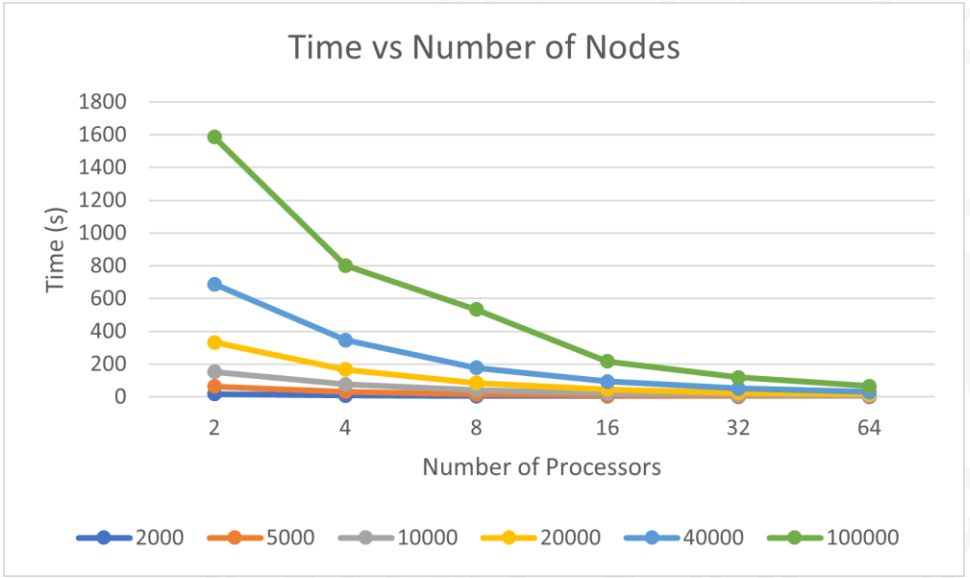


Results

Nodes

Generations

512x512 Grid	2	4	8	16	32	64
2000	17.8652	9.03771	4.72184	2.43581	1.54537	1.03963
5000	44.7803	22.4419	11.5091	6.30291	3.49551	2.10276
10000	90.2277	44.8646	23.1275	12.2398	6.68308	3.75925
20000	178.947	89.9282	45.825	24.5663	13.1133	7.89644
40000	356.691	180.413	92.2154	48.2092	26.4426	15.6172
100000	896.876	454.511	355.482	122.025	66.6963	34.5268

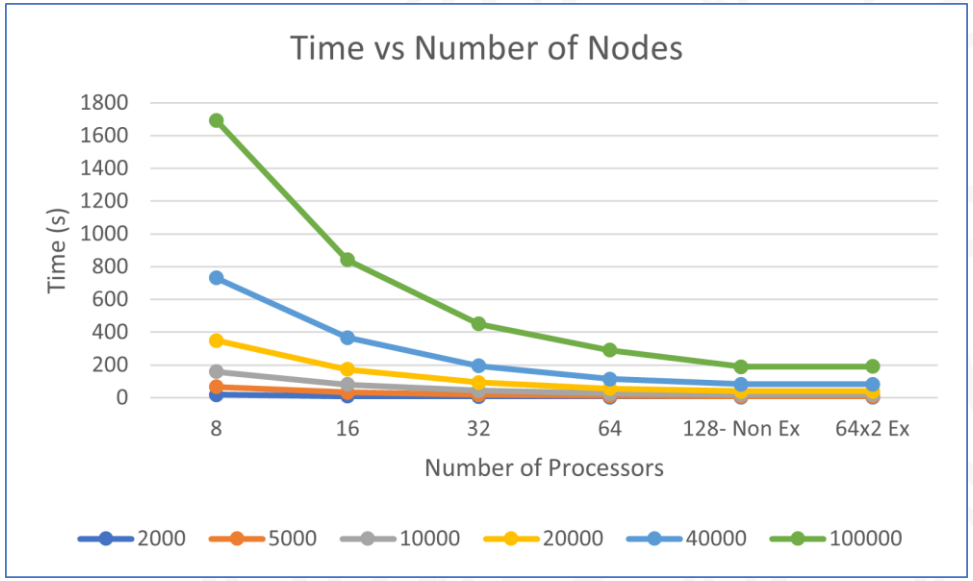


Results

Nodes

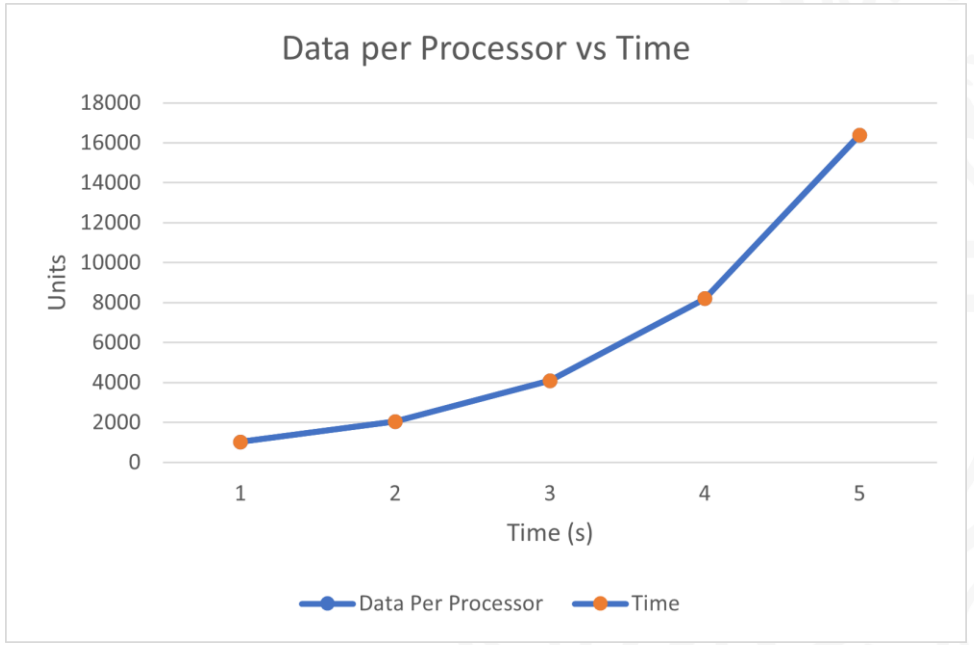
Generations

1024x1024 Grid	8	16	32	64	128- Non Ex	64x2 Ex
2000	18.7678	9.34687	5.0574	3.18806	2.1716	2.23913
5000	46.601	23.4859	12.9617	7.34906	5.45177	5.44231
10000	93.2208	46.7425	25.5975	14.8146	10.7789	10.9067
20000	190.405	93.0172	49.9057	29.3662	21.5298	21.7252
40000	381.589	194.962	99.8908	59.4824	42.191	42.7178
100000	962.487	473.983	256.136	175.659	106.927	108.071

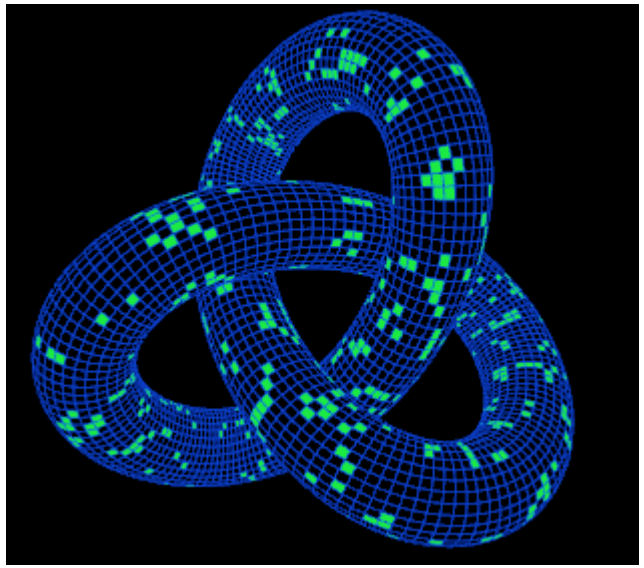


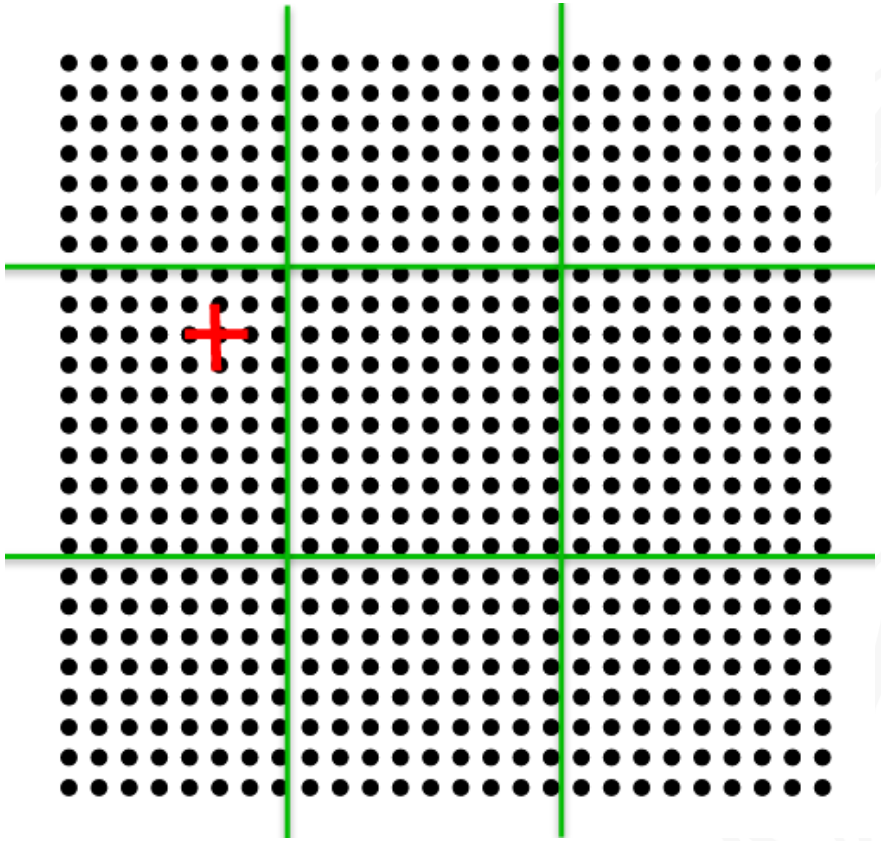
Scaling

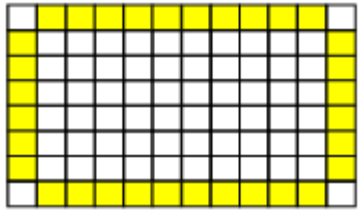
Data Per Processor	Time
1024	1.32992
2048	1.38794
4096	3.3433
8192	6.68308
16384	14.8146

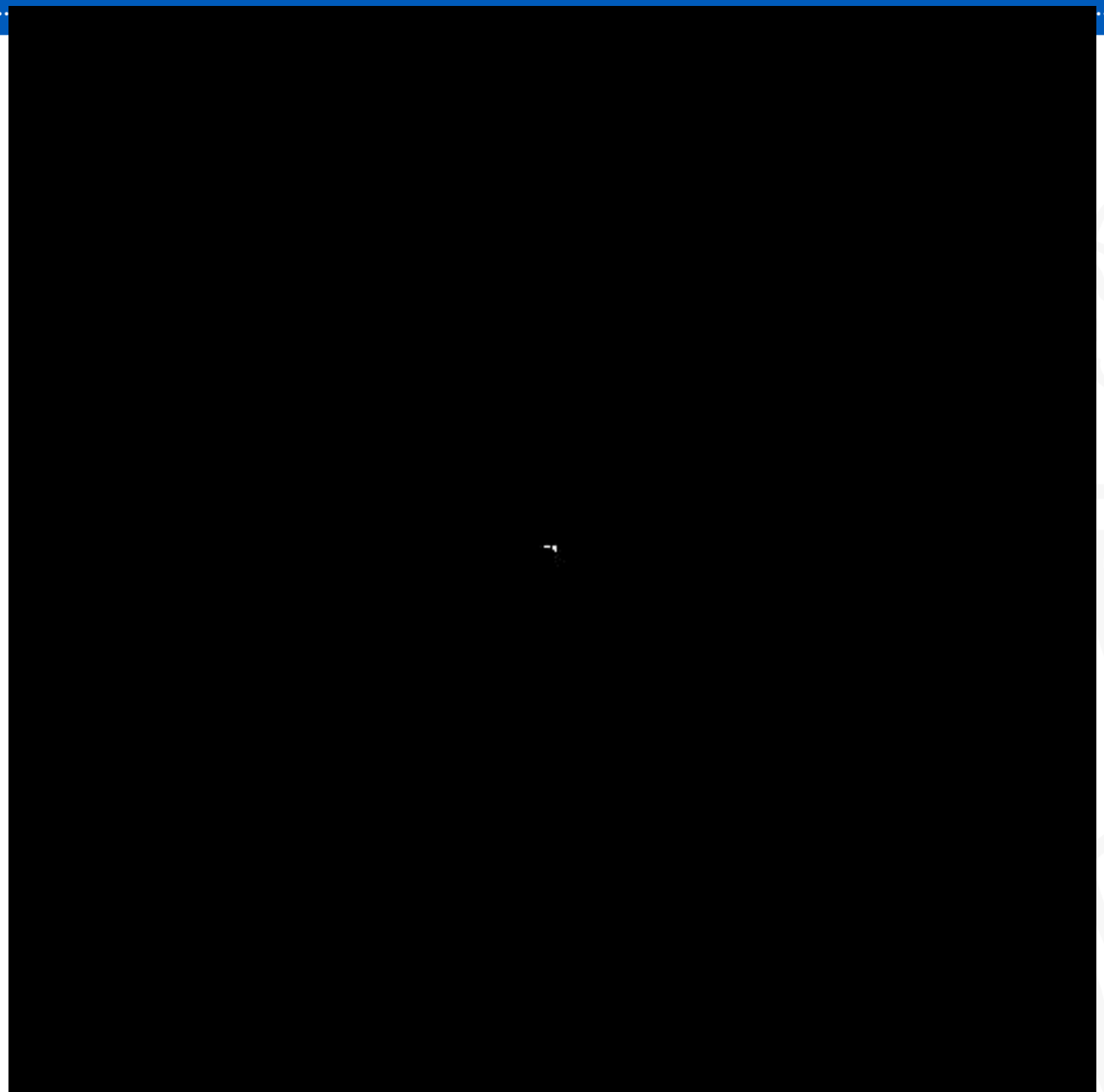


Alternative approach









Thank You