CONWAY’S GAME OF LIFE

CSE708

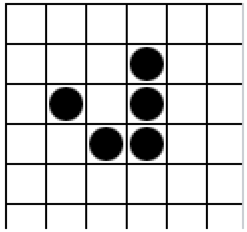
Varun Sudarshan

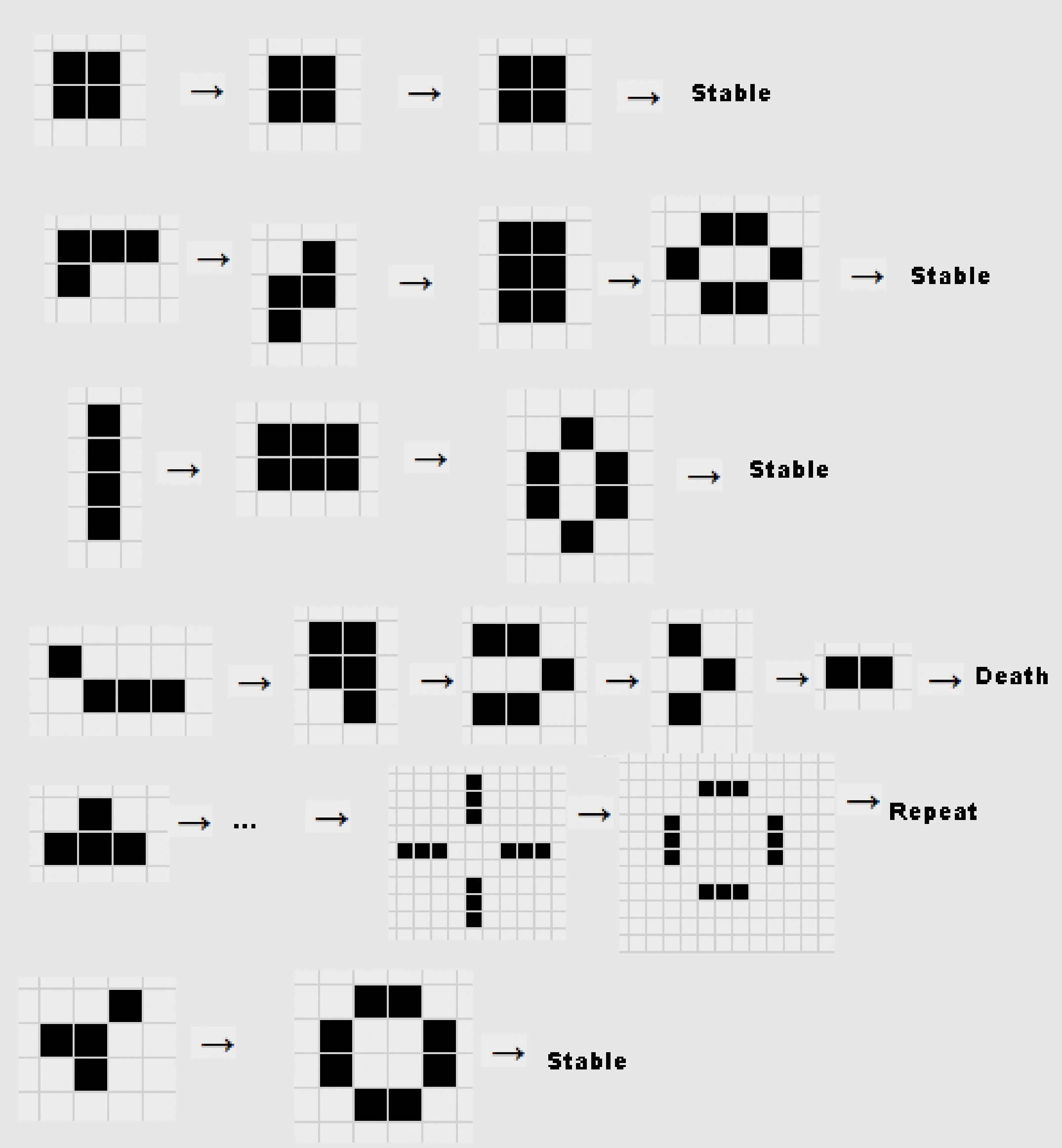
University at Buffalo The State University of New York

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

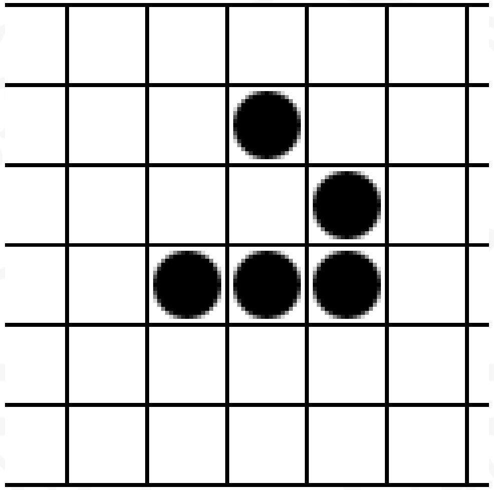




http://pi.math.cornell.edu/~lipa/mec/lesson6.html

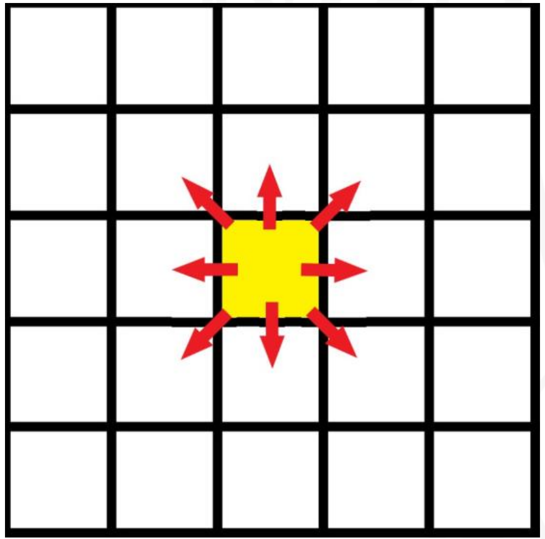
Algorithm

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.

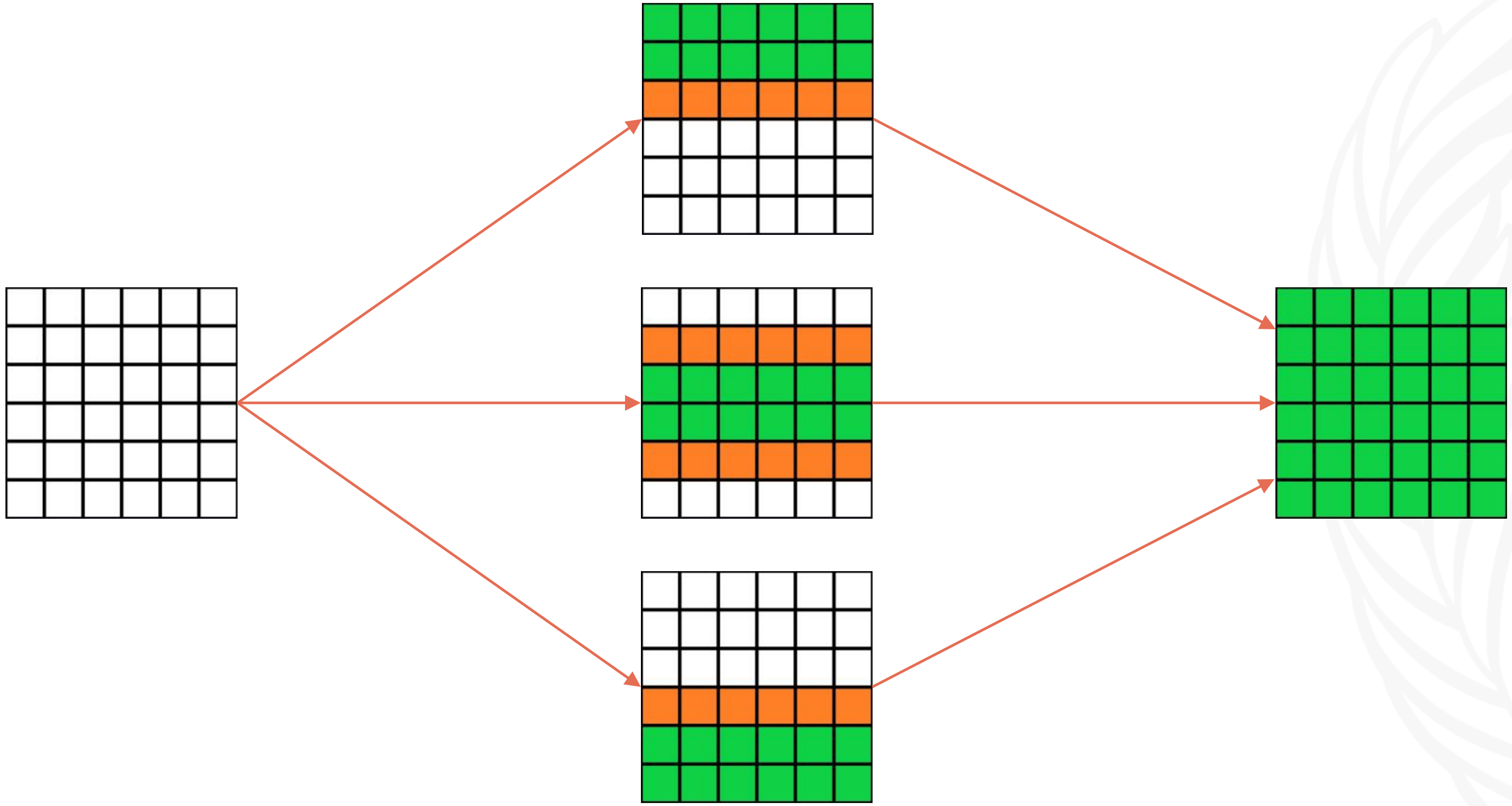


Parallel Implementation

- 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



Threads and Thread Blocks- CUDA

- Conceptually, the division of data and the computation of each check sequentially remains the same
- To implement this in CUDA, we use threads and a kernel function.
- This Kernel function is executed in each thread.
- A group of threads is known as a Thread Block in the CUDA world
- The distinction between which part of the data you are operating on is made based on a combination of Block ID and Thread ID which is accessible by each instance of the Kernel function (ie. Thread)

Low level execution

- Streaming Multiprocessors : General purpose processors that picks up a new thread block when the previous block's execution is complete.
- Warps : a thread block is composed of 'warps'. A warp is a set of 32 threads within a thread block such that all the threads in a warp execute the same instruction.
- Compared to a general purpose instruction computation, this method is more efficient as the overhead for changing out the instruction is removed and the only change that happens is which memory the instruction acts upon.

Speed-up

MPI

- For one 64x64 grid, 10000 Generations : ~40 seconds

CUDA

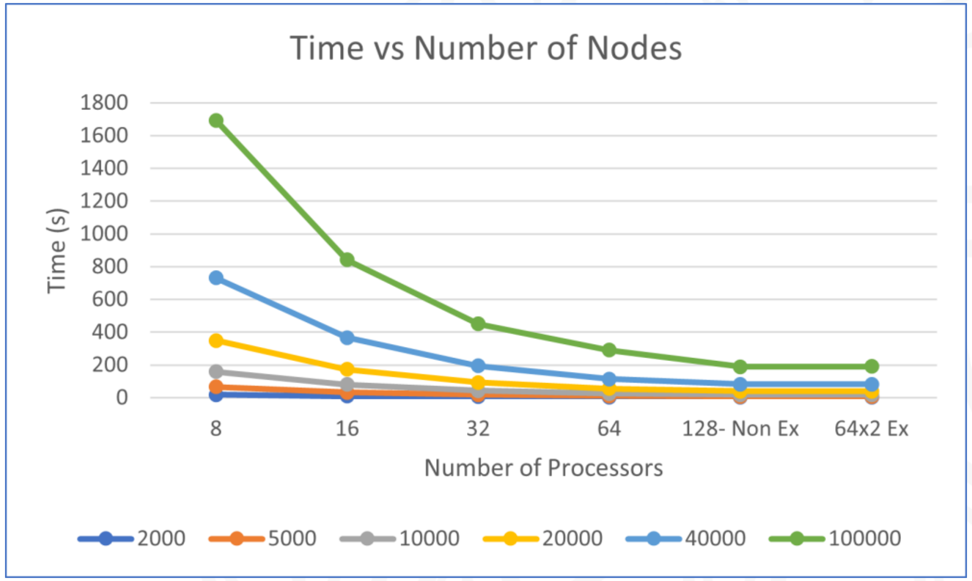
- For one 64x64 grid, 1000000 Generations : 71 milliseconds (0.071 s)

Results (old)

Nodes

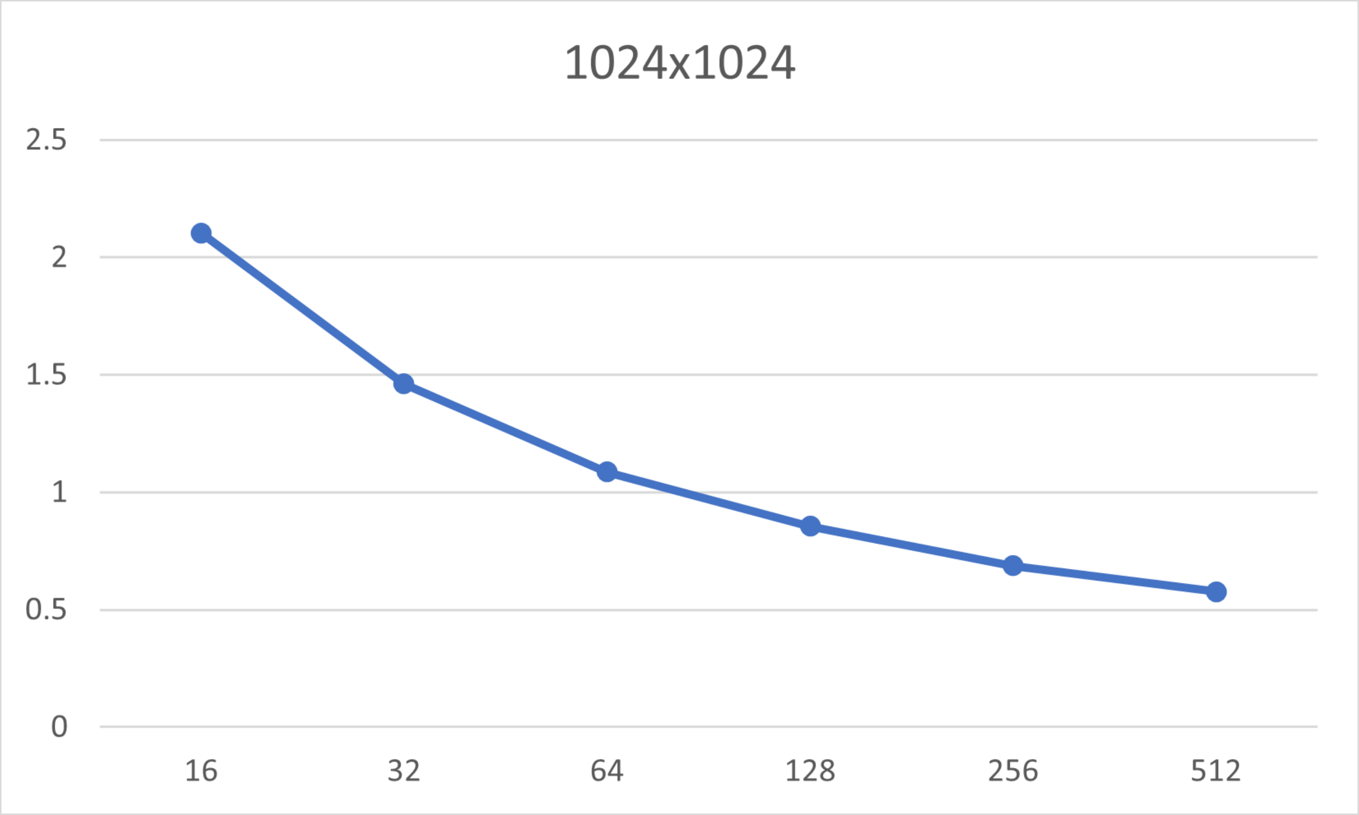
Generations

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



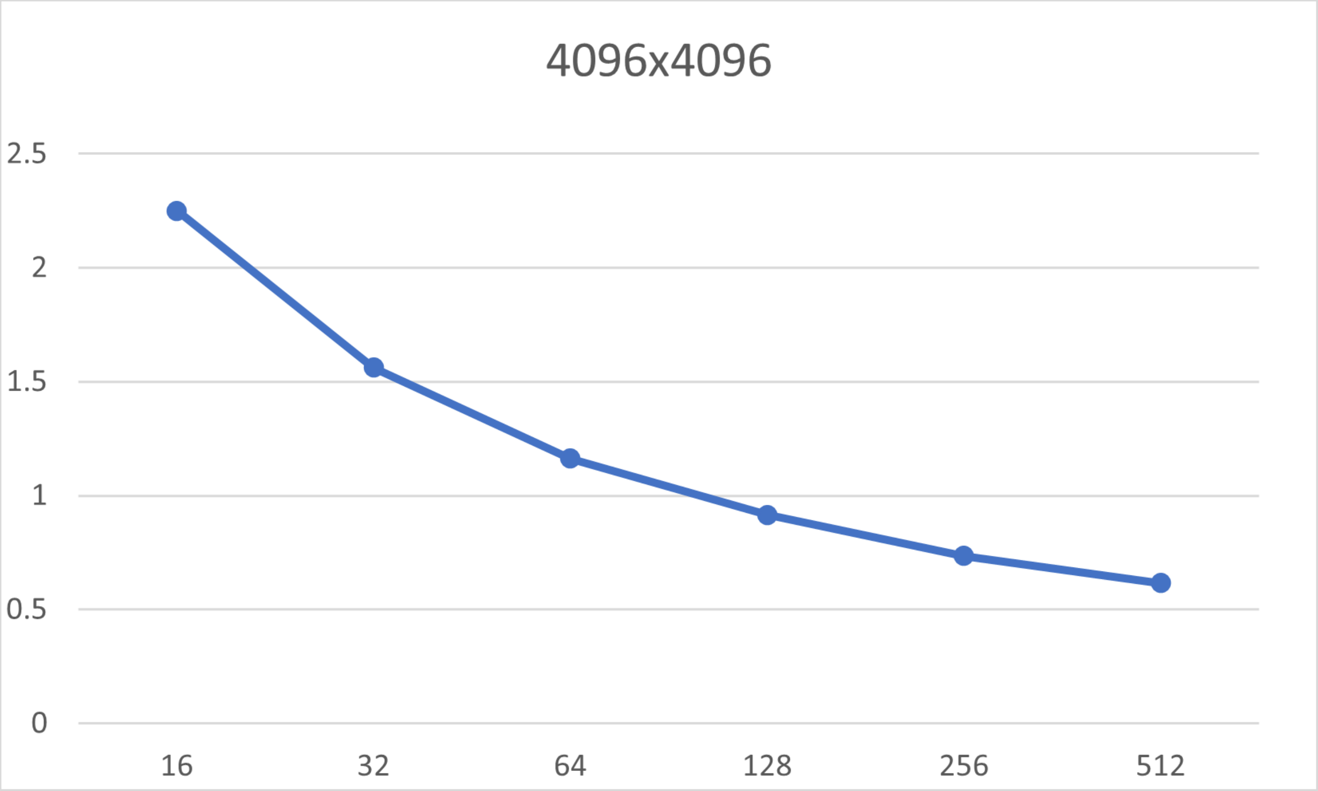
1024 x 1024 Grid

Threads	Time (s)
16	2.103825
32	1.461213
64	1.087408
128	0.856166
256	0.686833
512	0.576121



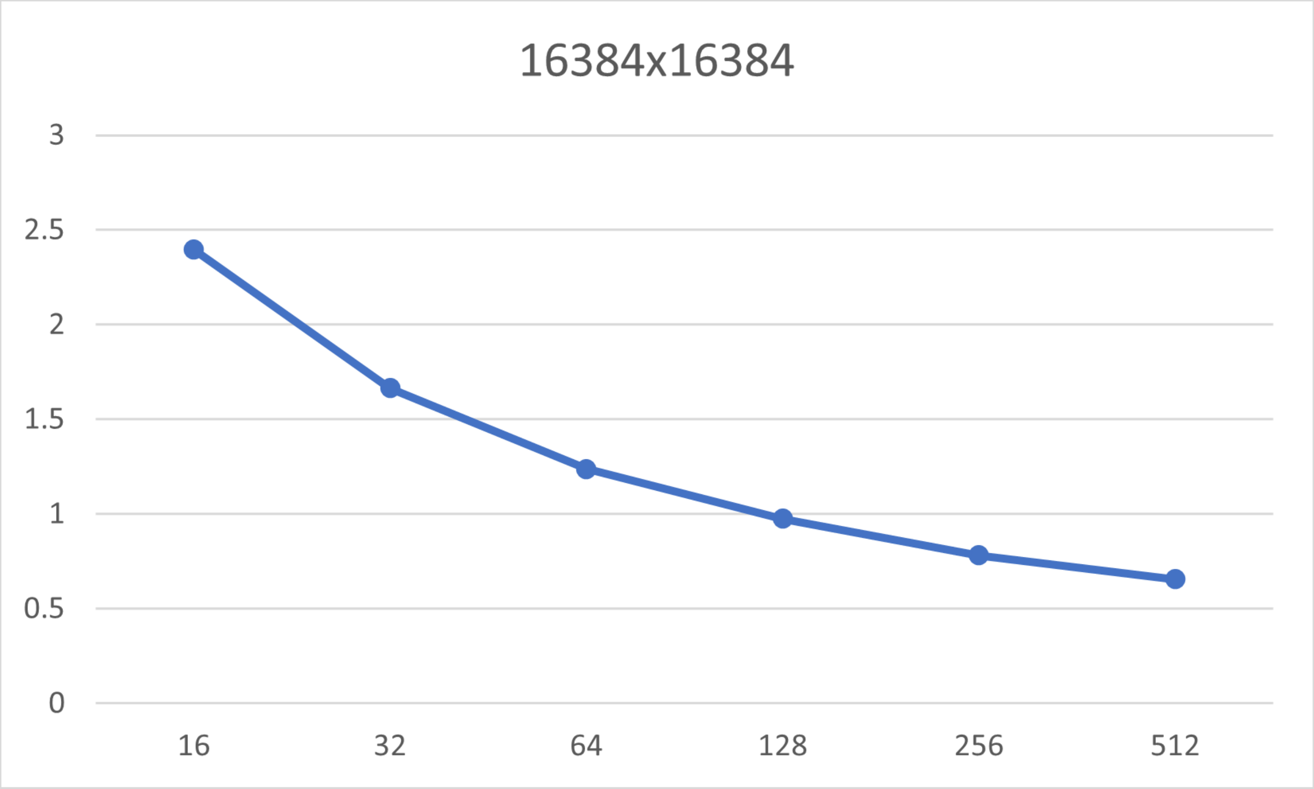
4096 x 4096Grid

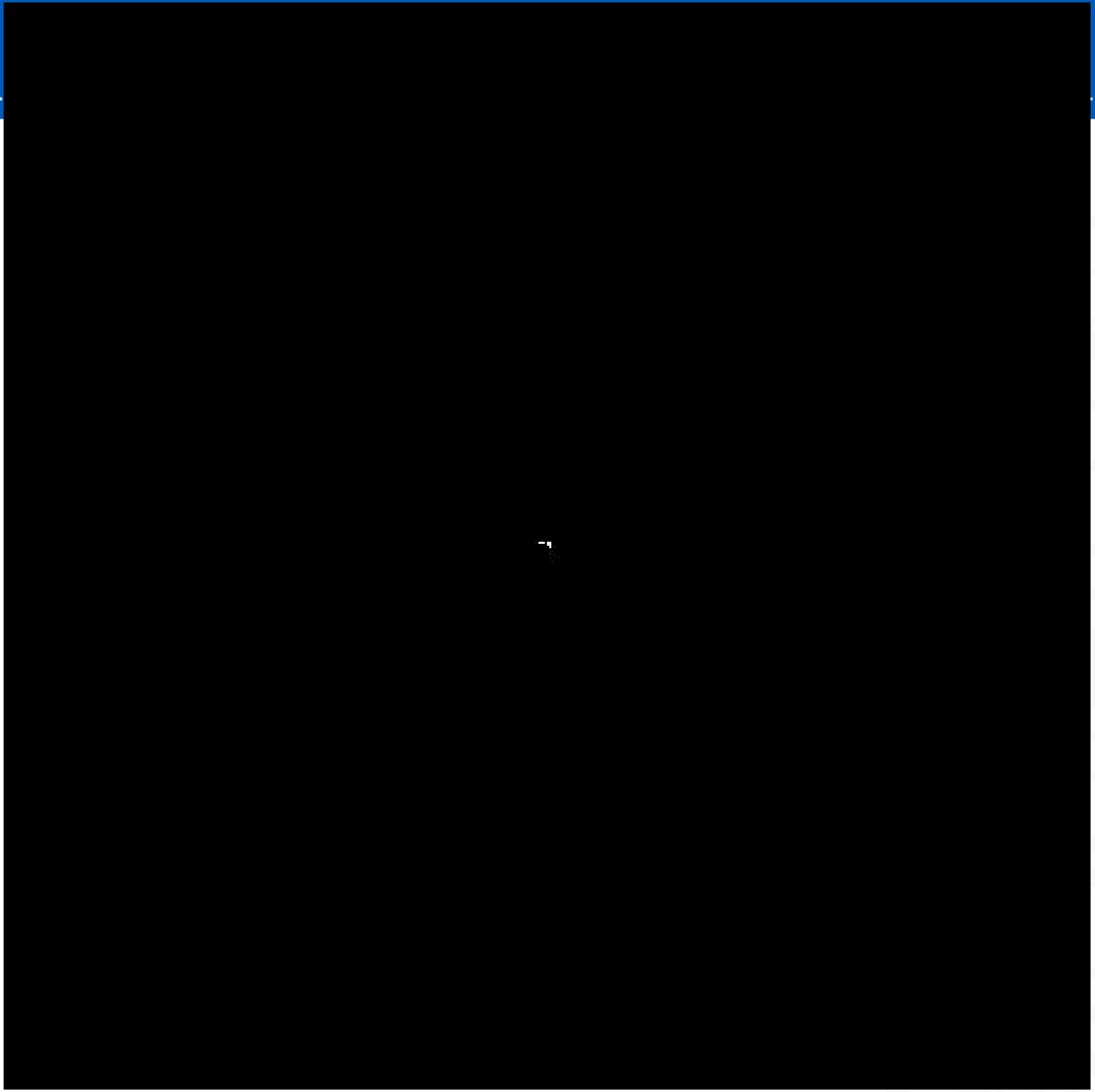
Threads	Time (s)
16	2.249718
32	1.562543
64	1.162816
128	0.915539
256	0.734462
512	0.616073



16384 x 16384 Grid

Threads	Time (s)
16	2.395611
32	1.663873
64	1.238224
128	0.974911
256	0.782092
512	0.656025





Thank You