Theodore Wu, Charlie Liu, Alexa Daigle, Espie Taylor
Assignment 4
Parallel Programming
03/29/24

<u>Total execution time for each run:</u>

A.) 1 node, 1 GPU, 16Kx16K world size each MPI rank, 128 iterations with 256 CUDA thread block size and pattern 5: 0.643961

B.) 1 node, 2 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 256 CUDA thread block size and pattern 5: 0.656310

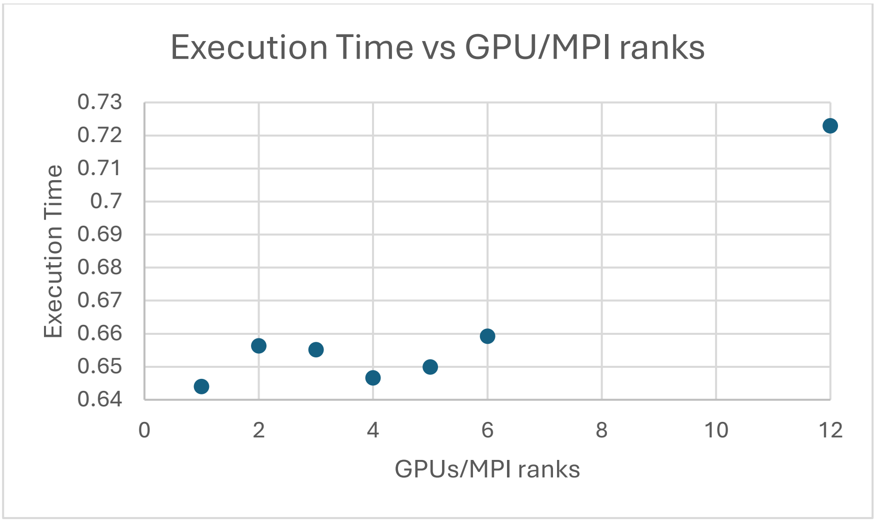
C.) 1 node, 3 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 256 CUDA thread block size and pattern 5: 0.655127

D.) 1 node, 4 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 256 CUDA thread block size and pattern 5: 0.646586

E.) 1 node, 5 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 256 CUDA thread block size and pattern 5: 0.649848

F.) 1 node, 6 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 256 CUDA thread block size and pattern 5: 0.659207

G.) 2 nodes, 12 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 256 CUDA thread block size and pattern 5: 0.722893



<u>Maximum speedup relative to using a single GPU:</u>

Time with single GPU:
Speedup with 2: 0.643961/0.656310 = 0.981x
Speedup with 3: 0.643961/0.655127 = 0.982x
Speedup with 4: 0.643961/0.646586 = 0.996x
Speedup with 5: 0.643961/0.649848 = 0.991x
Speedup with 6: 0.643961/0.659207 = 0.977x
Speedup with 12: 0.643961/0.722893 = 0.891x
Maximum speedup: 0.996x

<u>Configuration that yields the fastest "cells updates per second" rate:</u>

A.) 1 node, 1 GPU: World of 16384^2 that runs for 128 iterations with 0.643961 execution time:
16384^2 * 128 = 34,359,738,368 cell updates
34,359,738,368 / 0.643961 = 53,356,862,244.8 cell updates per second

B.) 1 node, 2 GPUs/MPI ranks: World of 16384^2 that runs for 128 iterations with 0.656310 execution time:
2(16384^2) * 128 = 68,719,476,736 cell updates
68,719,476,736 / 0.656310 = 104,3705,820,018 cell updates per second

C.) 1 node, 3 GPUs/MPI ranks: World of 16384^2 that runs for 128 iterations with 0.655127 execution time:
3(16384^2) * 128 = 103,079,215,104 cell updates
103,079,215,104 / 0.655127 = 157,342,339,888.3 cell updates per second

D.) 1 node, 4 GPUs/MPI ranks: World of 16384^2 that runs for 128 iterations with 0.646586 execution time:
4(16384^2) * 128 = 137,438,953,472 cell updates
137,438,953,472 / 0.646586 = 212,560,979,470.6 cell updates per second

E.) 1 node, 5 GPUs/MPI ranks: World of 16384^2 that runs for 128 iterations with 0.649848 execution time:
5(16384^2) * 128 = 171,798,691,840 cell updates
171,798,691,840 / 0.649848 = 264,367,501,077.2 cell updates per second

F.) 1 node, 6 GPUs/MPI ranks: World of 16384^2 that runs for 128 iterations with 0.659207 execution time:
6(16384^2) * 128 = 206,158,430,208 cell updates
206,158,430,208 / 0.659207 = 312,737,016,154.3 cell updates per second

G.) 2 nodes, 12 GPUs/MPI ranks: World of 16384^2 that runs for 128 iterations with 0.722893 execution time:
12(16384^2) * 128 = 412,316,860,416 cell updates
412,316,860,416 / 0.722893 = 570,370,525,673.9 cell updates per second

Fastest "cells updates per second" rate:
2 nodes, 12 GPUs/MPI ranks, 16Kx16K world size each MPI rank, 128 iterations with 256 CUDA thread block size and pattern 5 with 570,370,525,673.9 cell updates per second

Why this configuration was faster than others:

Contributions:
All group members engaged in a collaborative effort to review and test the C and CUDA code to ensure correctness. Additionally, all group members effectively communicated with one another in regard to dividing up work and completing various tasks in a timely manner.
Other individual contributions:
Theodore Wu: Corrections and bug fixes in code
Charlie Liu: C and CUDA code
Alexa Daigle: Report and calculations
Espie Taylor: AIMOS runs