TITLE: Parallel Minesweeper Solver, by Daniel Kim and Yang Zhou

URL: https://ddkim1.github.io/15418-Final-Project/

SUMMARY:

We are going to implement a solver for the game Minesweeper using several parallel methods. We will be analyzing the speedup from utilizing CUDA, OpenMP, and if time persists, MPI, compared to a sequential implementation.

BACKGROUND:

Minesweeper is a popular logic puzzle computer video game in which the player's objective is to clear the grid while not clicking on any of the hidden mines. To determine the location of the mines, players will be given clues in the form of numbers on spaces, where the number indicates the number of adjacent squares that contain a mine. Despite its simple nature, it has been proved as a co-NP complete problem. [1] As a result, efficient sequential implementations currently do not exist, and performance gains will be found through parallelization.

In this project, our goal will be to first implement a sequential program that is able to solve a board by clearing the board while not exposing any mines. Afterwards, we aim to build upon our work by parallelizing this program using several different parallel frameworks. The general approach towards solving a board is that we will look at every possible square, and denote the probability for each square that it contains a mine. From there, the program will mark off any squares that for sure have mines, and then click on squares with the lowest probability of containing a mine. This process continues until the board is cleared. However, especially for large grids, this becomes a very slow process to perform sequentially, as at each step, the program would have to analyze every square at each step to determine which square to click next, especially if there are numerous squares that are guaranteed to contain mines. Instead, if we were to parallelize the program by, for instance, dividing the squares of the grid across different parallel workers, it could lead to substantial speedup.

THE CHALLENGE:

The workload is difficult to perfectly balance largely due to there being less mines or potential squares to click in certain areas of the board. Ideally, if we were to perfectly divide the workload across the parallel workers, then each worker would have an equal number of mines in their assigned squares; however, that is not necessarily the case, as Minesweeper often has large areas of squares with no mines nearby. Attempting to find a good workload split that balances the workload across the workers will be challenging as a result.

In addition, communication also makes workload balance quite difficult. If a worker manages to mark a square as a mine, or if a worker manages to clear a square as not a mine, then it needs to communicate this information to other workers that may use this information to be able to clear their squares. This then becomes a challenge of determining which workers we need to communicate with whenever we clear a square. Designing such algorithms to divide the workload as evenly as possible might need us to research existing literature to delve into heuristics or stochastic nature of the bomb nature.

Another challenge originates from the nature of the game. Sometimes, minesweeper requires the user to guess on which step to proceed. In this case, it favors the recursive parallelization methods like OpenMP. However, we want to still achieve good results using data parallelism tools like OpenMPI or CUDA. In addition, given that we have a powerful GPU or a powerful CPU with many cores, we would also like to try to parallelize our solver to perform multiple guesses at once. All of these thoughts are interesting and also challenging to implement.

RESOURCES:

We have a few resources to start with: first we have an extended study of the math behind Minesweeper [1], which looks at the problem of solving the minesweeper game from a probabilistic lens. Also, we found a self-proclaimed efficient sequential implementation [2], from which we can study and base our initial sequential algorithm from.

Code-wise, we will be starting from scratch. In addition to our current research, we will be doing some further research to determine what Minesweeper solving algorithms exist currently first, and see if there are any sequential algorithms that have been developed that are faster than the algorithm proposed earlier. We will then code a sequential algorithm that we determine is best, and then code the parallel algorithms afterwards.

In terms of computers, we would be using the GHC machines and the Bridges-2 machines. GHC machines have both CPU and GPU resources, besides we have access to some additional GPU resources. On the CPU side, we will be using the GHC machines to code our initial parallel implementations, and we will then use the Bridges-2 machines to determine how our parallel algorithms scale with a large number of workers and a large amount of computing power. On the GPU side, we will run experiments on either the GPU on GHC and, if we don’t have available machines, on extra GPUs we can get our hands on.

GOALS AND DELIVERABLES:

Our end goal is to achieve speedup relative to the sequential version of the solver.

Deliverables are summarized in the following list (1-3 are what we plan to achieve, while 4 is what we hope to achieve):
1). We will develop an efficient working version of the sequential solver of the game, which is capable of running on both CPU and GPU platforms.
2). After that, we would build upon the sequential version and finish our initial effort of parallelization using OpenMP and CUDA.
3). For each parallel implementation, after we successfully finish our programs we will attempt to determine better workload distributions in hopes of improving speedup.
4). We hope to build a visualization tool to showcase the speed of our parallelized solver and also implement a parallel solver using MPI if time permits.

In summary, we plan to achieve a working sequential, CUDA, and OpenMP implementation, in addition to a script that will check for both correctness of our implementation as well as the speedup we achieve in comparison to a sequential implementation. We also hope to achieve an implementation using MPI, and in addition to the speedup graphs for the parallel implementations, we also hope to have a form of live demonstration of the parallel solvers at the poster session. Through the graphs and the live demonstrations, we hope to showcase the speedup we obtained from parallelizing our program, in addition to which parallel framework we found to work the best for this particular problem.

Through our analysis, we are hoping to learn how fast a minesweeper solver can be made through parallelization, and we are also hoping to learn the applicability of parallelization in improving the performance of algorithms for NP-complete problems such as minesweeper.

PLATFORM CHOICE:

We will be using C++ and CUDA for our programming languages. This is because since we are aiming to use CUDA, OpenMP, and potentially MPI as our parallel frameworks, we need to use C++ and CUDA to utilize these frameworks. We will also be using the GHC and Bridges-2 machines to determine our programs' performance, as the GHC machines will give us an indication as to whether our algorithms are able to perform work in parallel, and the Bridges-2 machines will be able to indicate to us how well our programs scale due to the amount of computing power on the machines.

SCHEDULE:

We will aim to stick to the following schedule:

<u>November 19, 2023</u>

We will complete our preliminary research on minesweeper solvers, finalize the sequential algorithm we will be implementing, and also complete the design for the board data structure.

November 26, 2023

We will complete the implementation of the sequential algorithm for the minesweeper board solver. In addition, we will also code a checker script that will tell us the correctness of our algorithm, in addition to any speedup we have against the sequential algorithm.

December 1, 2023

We will finish the parallel implementation of our algorithm using OpenMP, and we will begin writing our milestone report.

December 3, 2023

We will finish our milestone report by this point and submit by 11:59 pm.

December 10, 2023

We will finish the parallel implementation of our algorithm using CUDA, and we will begin preparing for the final report and the poster session.

December 14, 2023

We will finalize our final report and submit it by 11:59 pm. We will also finalize our poster session presentation materials.

If time allows, we will also aim to hit these deadlines:

December 12, 2023

We will finish the parallel implementation of our algorithm using MPI, and incorporate the results of this algorithm into our final report.

December 15, 2023

We will have a live demo displaying our parallel solvers in action.

Citations:

[1] Becerra, David J. *Algorithmic approaches to playing minesweeper*. Diss. 2015.
[2] https://dev.to/krlove/creating-advanced-minesweeper-solver-using-logic-programming-2ppd