Gregory Fu
Uni: gf2426

Functional Parallel Programming Final Project Proposal

What I Implemented:

For my final project, I implemented a connect four game played in the console against the computer. The computer utilizes a minimax algorithm to formulate the best move for the specific board given to it. My implementation allows the user to choose if they would like to use the sequential implementation of minimax, or the parallel implementation.

How I Implemented:

- Basic Game:

  After getting which version of the program to run, I used a play loop that switches between the human and the AI until a terminal state is reached to drive the game. The terminal state is checked by checking for a winner, or if the board is filled out. After getting the move from the user, which is just the column number, the AI runs minimax to find the best move. The way that the terminal state is evaluated is by breaking the board into a series of [[char]] that contain all contiguous four positions sequentially, and checking if any of these produces a terminal state.

- Minimax:

  Within this minimax, we first test for terminal conditions on if the depth has been reached or we are in a terminal state. Then, we generate a list of possible moves, which is used to make the next generation of nodes by applying the moves element-wise. Next, we have to evaluate these children to see if there is a terminal state among them, in which we return that move and the value associated with it. Otherwise, we loop through the children with d-1 and the opposite player.

The parallelization also comes into play for the minimax implementation. Every call to minimax among the possible next states is done in its own thread, so each time a node branches, more threads will be created if the resources allow.

Performance Figures:

Through simulating each minimax implementation against a random player, it seems pretty clear that the parallel version is about twice as fast as the sequential version at ~1.06 seconds versus ~2.23 seconds. This is using a max depth of 6 and setting the -N4 flag for 4 maximum concurrent threads. Increasing the max depth should hypothetically yield better comparative results for the parallel version compared to the sequential version considering that the number of nodes grows exponentially. Additionally, against a random agent, the AI won almost every single time, with a winning percentage of across the simulations. I have included the csv of my simulations:

https://docs.google.com/spreadsheets/d/1eqQloJiXqzQuFQZ1KsEvuGQAxHx2qp14v_4LJQH0jqc/edit?usp=sharing

Note: Status here refers to the end status of the game. Here's what they mean:

1. Human/Simulations wins with 4 vertical tokens
2. AI wins with 4 vertical tokens
3. Human/Simulations wins with 4 horizontal tokens
4. AI wins with 4 horizontal tokens
5. Human/Simulations wins with 4 diagonal tokens
6. AI wins with 4 diagonal tokens

Code Written:

I wrote all the code for my project in a main.hs file. I did reference some stackoverflow answers, which I acknowledged by commenting the link next to it. I also referenced the show board function from another repo since I was having trouble getting the squares to line up and it doesn't really concern the crux of the course material anyways.