AlphaChess

Jason Huang (jasonh1)
Abel Tesfaye (atesfaye)

URL: http://www.tesfayeabel.com/15418FinalProject/

SUMMARY: We are going to parallelize multiple chess AI algorithms using Cilk and our own implementation of a scheduler, comparing the efficiency and the effectiveness of each scheduler.

BACKGROUND: We planning on parallelizing some chess AI algorithms. These algorithms usually try to evaluate all the possible moves at a given board state and then use some heuristic to pick the best move. Some algorithms we are considering are min-max and alpha-beta pruning. Since the moves are considered independently, there are plenty of opportunities for parallelism. The idea is that multiple threads can consider different moves and then the main thread will pick the best move. We are currently looking at implementing this with Cilk and may implement this with other parallel libraries.

THE CHALLENGE: The main issue with making a parallel chess AI application is how to schedule the independent tasks between the threads on a given computer. We cannot use a static assignment because some moves may lead to a longer game of chess, while other moves will lead to a shorter game of chess, so that thread will be occupied for a variable amount of time. This requires us to implement a dynamic scheduler. The dynamic scheduler will minimize the work imbalance between each thread.

By doing this project, we hope to learn how to implement an efficient dynamic scheduler. However, the number of possible moves is much greater than the number of threads we have access to, so efficiently mapping moves to each of the threads will be the biggest challenge of the project.

RESOURCES: We will be using this link (http://supertech.csail.mit.edu/papers/dimacs94.pdf) as guidance for our project. We will also be using the unix clusters to evaluate the performance of our code, and will most likely start from scratch since the rules of chess are not too difficult to implement in code.

GOALS AND DELIVERABLES:

Plan to achieve: We want to be able to implement the sequential chess AI with the min-max and alpha-beta pruning algorithms. We also want to parallelize the sequential version using Cilk and our own scheduler.

Hope to achieve: We hope to beat the performance of Cilk with our own scheduler, since our scheduler will be specifically optimize for chess AI algorithms.

At the poster session, we will display a demo of our chess AI running with our scheduler to prove that it works. We'll bring a laptop to let people play against our chess AI to see if they can beat it. We'll also show the speed-up plots of our scheduler vs. Cilk against the sequential algorithm. We will also include a figure showing how our scheduler would react given a board state, comparing it with how Cilk would respond to a certain board state.

We hope to learn whether the workload of chess AI can be optimize with a custom task scheduler.

PLATFORM CHOICE: As mentioned earlier, we will be using the unix cluster to test performance and correctness of our code. The reason for this is we plan on parallelizing chess algorithms using Cilk, and those run on the CPU. We will be using the C++ language to implement all our algorithms, since C++ supports object-oriented programming, and this is essentially to keep track of the players and the pieces.

SCHEDULE:

Date	Milestone
October 31	Finish the project proposal
November 7	Implement the sequential chess AI algorithms
November 14	Parallelize the chess AI algorithms using Cilk
November 21	Begin implementing our own scheduler

November 28	Finish implementing our own scheduler
December 5	Optimize our scheduler specifically for chess algorithms
December 12	Finish testing our code
December 14	Finish final paper