Project Proposal: Playing Connect-4 with Parallelism

Howard Chen, Elizabeth Ji
hhchen@andrew.cmu.edu, emji@andrew.cmu.edu

April 10, 2017

1 Summary

We will implement part of an AI for playing Connect-4 that leverages parallel hardware to search the problem space quickly. We will develop implementations that utilize CUDA, OpenMP, etc. and compare their performance using various metrics.

2 Background

Connect four is a turn-based strategy game that allows for up to 7 possible moves on a given turn. Each game lasts up to 49 turns, resulting in a fairly sizable space of possible positions. One way for an AI player to select moves is to search down to some fixed depth, use a heuristic, and then apply the MiniMax algorithm to score states. Our goal is to write a program that can perform this estimation without using a pre-computed lookup table. In theory, the more states our program can search within the given time, the better estimate it can provide on the desirability of the states. However, in a situation where such an AI is used to play humans, we wish to make decisions within a few seconds. Thus, we will be optimizing to search as many states as possible within a fixed window of time.
The specific part we will implement will take a game state as input, and search/score as many states as possible within the allowed time frame. Another potential direction is to measure the time taken to search down to a given depth. Ultimately, this program can be easily extended to handle all sorts of deterministic turn-based games, but we have chosen Connect-4 as a simple use case that allows us to focus on the parallelizable portion of the program.

3 The Challenge

Coordinating the parallel search is fairly non-trivial. We can view the state space as a graph with states as nodes and moves as edges. There are many ways to schedule the search, each permitting different amounts of parallelism. Furthermore, there are often different paths that converge at the same states, so we want to minimize redundant computation without incurring high communication costs. This may involve designing a performant thread-safe hash table or other data structure.

4 Resources

We will be running this on machines with Nvidia GPU's and some multi-core machines (either Xeon Phi's or latedays machines).

5 Goals and Deliverables

We plan to achieve the following:

Howard Chen, Elizabeth Ji
hhchen@andrew.cmu.edu, emji@andrew.cmu.edu

- Develop the parallel search program with multiple frameworks (CUDA, OpenMP, pthreads)
- Develop benchmarking harnesses to measure the performance of each implementation on different time frames or search depths

If time allows, we can also try to set up the full game framework so that AI's can play against each other. Additionally, we can also try more complex state-scoring heuristics, and potentially parallelize those computations as well.

6 Platform

This problem has no real hardware constraints, but we wish to harness the compute power of parallel systems such as GPU's and multi-core processors.

7 Schedule

Completion Date	Task
April 14	Sequential implementation (baseline)
April 18	Test Harness
April 21	OpenMP Implementation & tuning
April 28	pthread Implementation & tuning
May 5	CUDA Implementation & tuning