Project Proposal: Playing Connect-4 with Parallelism

Howard Chen, Elizabeth Ji
hhchen@andrew.cmu.edu, emji@andrew.cmu.edu
April 26, 2017

1 Progress

In our original schedule, we detailed the following implementations for our state-searcher:

- Sequential (baseline)
- OpenMP
- Pthread
- CUDA

At this point, we have completed the sequential and OpenMP implementations, as well as a benchmarking harness for measuring performance. Our testing harness currently tests on empty initial boards and allows different runs of multiple depths to compare results. In the process of optimizing our OpenMP implementation, we also implemented a lock-free hashmap. For testing purposes, we have also implemented a simple text-based interface for playing against the AI.

2 Projected Deliverables

We still believe it is feasible to achieve all of the deliverables detailed in our proposal. We will also extend our testing harness to allow inputs of different starting boards to examine if the affects performance. Additionally, we will very likely have time to implement a program that allows different AI's to play against each other. We will also create a set of graphs to illustrate the relative performances of different implementations on different inputs, varying on the initial board as well as the search depth.

Due to the relative simplicity of Connect4 boards, we have decided to not pursue the direction of finding/parallelizing more complex board-scoring heuristics. The remaining goals we wish to hit are as follows:

- Pthread and CUDA implementations
- Performance Graphs
- AI vs AI program

At the parallelism competition, we will likely be presenting the performance graphs.

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

3 Preliminary results

The table below summarizes the relative performance of the OpenMP implementation versus the baseline. These results were obtained by starting the search on the empty board.

Search depth	States	Sequential time (ms)	OpenMP time (ms)
3	294	6.7	22.7
5	6577	101.8	64.2
7	76965	875.1	474.2
9	825432	8252.0	4642.3

4 Concerns

The primary difficulty with implementing an OpenMP-based solutioon was coordination between threads. In particular, there are many different sequences of moves that result in the same boards, and we do not need to score a board more than once. However, de-duplication of boards in a parallel setting can be fairly difficult to do in a performant manner. For this reason, we implemented a lock-free hash table that can be used for CPU-based implementations. However, we don't currently have a good way of working around this problem for our CUDA-based solution. In the worst case, any CUDA implementation may simply be slower than the baseline.