Final Project Proposal

Home

Title

Parallel 2048 Solver (Wei-Ting Tang)

URL

https://white123.github.io

Summary

I am going to create a parallel solver of the 2048 game using OpenMP, and perform a deep analysis of the performance characteristics.

Background

2048 is a sliding tile puzzle game, and the objective of the game is to combine the numbered tiles to create higher tiles, as large as possible.

A 2048 solver simulates a few "future steps" to determine the best next move. Given the current board, it will try to slide all the directions, namely right, left, up, and down, and generate a new tile at every empty tile to get all of the possible next states. Then, repeat the above steps a few times (e.g., depth = 8), and calculate the "heuristics" score of each move. The various heuristic scores, such as open squares and large values on the edge, are weighted and combined into a comprehensive score, determining how "good" the board position is.

The simulation steps are independent, so it has a huge potential to be parallelized.

Challenge

The solver creates many tasks; however, parallelism is not intuitive. The eval function will be called recursively, creating a tremendous amount of calculation. The workload of each step grows exponentially, depending on the number of empty tiles on a given board. This causes the workload to be highly imbalanced, and the tremendous number of tasks makes the context switch costs significantly large.

My goal is to tradeoff between parallel computing and overhead cost, and take advantage of the high number of cores on the bridge computer as much as possible.

Resources

This post discusses the mechanism details of the 2048 solver, and this repo implemented the 2048 solver in C++. I will start at this repo and optimize the core algorithm to run on the high-count core computer.

Deliverables

Plan to achieve

- Finish 300 steps in 8 seconds on average
- Graph of performance versus different thread count
- Live or video demo of solver with UI

Hope to achieve

- Finish 300 steps in 4 seconds on average
- Break the overhead barrier of high thread count

Platform

OpenMP is suitable for the project since we can easily distribute the tasks to different cores (threads) on bridge computers.

Schedule

Week	**Plan**
Week 1 (11/6 - 11/12)	Submit the proposal and research on the topic
Week 2 (11/13 - 11/19)	Build up the sequential implementation that runs on bridge computers
Week 3 (11/20 - 11/26)	Develop a naive parallel implementation and submit the milestone report
Week 4 (11/27 - 12/3)	Optimize parallel implementations to utilize high-count cores and analyze the bottleneck
Week 5 (12/4 - 12/10)	Finalize the parallel version and apply different experiments to it
Week 6 (12/11 - 12/17)	Finish and submit the final report