Sudoku Solver Project Report

Mateusz Polis, 327250

December 19, 2024

Abstract

This report presents a comparative analysis of GPU-based and CPU-based Sudoku solvers implemented using parallel and sequential backtracking algorithms, respectively.

1 Introduction

Sudoku, a widely recognized logic-based number-placement puzzle, serves as an excellent benchmark for evaluating algorithmic efficiency and computational performance. Efficiently solving Sudoku puzzles, especially those of varying difficulty levels, is crucial for applications requiring rapid problem-solving capabilities. This report compares two implementations of Sudoku solvers:

- **GPU-Based Solver:** Utilizes parallel computations to expedite the backtracking process.
- **CPU-Based Solver:** Implements a traditional sequential backtracking algorithm without parallelism.

The comparison focuses on execution times across various Sudoku board difficulties, providing insights into the scalability and efficiency of parallel versus sequential computing approaches.

2 Methodology

2.1 GPU-Based Sudoku Solver

The GPU-based solver leverages parallel processing capabilities to explore multiple Sudoku board states simultaneously. The implementation employs a breadth-first search (BFS) strategy, where numerous board states are expanded in parallel. Key aspects of the GPU methodology include:

- **Parallel Board Expansion:** Each parallel unit handles the expansion of a distinct board state, identifying empty cells and attempting valid number placements concurrently.
- **Atomic Operations:** Ensures thread-safe increments when generating new boards and recording solutions, maintaining data integrity during parallel operations.

- **Memory Management:** Allocates separate memory buffers for current and next board states, facilitating efficient memory access patterns and minimizing latency.
- **Iteration Control:** Implements a maximum iteration limit to prevent infinite loops, ensuring termination even for unsolvable or highly complex boards.

This parallel approach significantly reduces the time required to explore multiple board states, especially beneficial for solving complex Sudoku puzzles with numerous possibilities.

2.2 CPU-Based Sudoku Solver

The CPU-based solver employs a sequential backtracking algorithm optimized for efficient execution. Unlike the GPU counterpart, the CPU implementation processes one board state at a time without parallel expansion. Key characteristics of the CPU methodology include:

- **Sequential Processing:** Each Sudoku board is processed individually, systematically exploring valid number placements through recursive backtracking.
- **Single-Threaded Execution:** The solver operates in a single-threaded context to maintain algorithmic consistency and avoid complexities associated with parallelism.
- **Memory Efficiency:** Optimized for cache locality and reduced memory overhead, the CPU solver efficiently manages board states within the hierarchical memory architecture of traditional CPUs.

This sequential approach, while straightforward, can be time-consuming for puzzles with high complexity due to the extensive recursive exploration required.

3 Results

The following table presents the execution times for both GPU-based and CPU-based Sudoku solvers across different board difficulties.

4 Results

The following table presents the execution times for both GPU-based and CPU-based Sudoku solvers across different board difficulties. All benchmark boards were taken from `https://sudoku.com` and are listed in the file `benchmark_boards.txt`.

Table 1: Execution Time Comparison of GPU-Based and CPU-Based Sudoku Solvers (9x9 Boards)

Difficulty	Board ID	CPU Time (s)	GPU Time (s)	Speedup Factor
Easy	1	0.0039848220	0.0022750960	1.75x
Easy	2	0.0001666222	0.0027870100	0.06x
Medium	3	0.0069525800	0.0032596880	2.13x
Medium	4	0.0014566100	0.0034554980	0.42x
Hard	5	0.0021670080	0.0039656080	0.55x
Hard	6	0.0005131876	0.0033522840	0.15x
Expert	7	0.0156623400	0.0059919620	2.61x
Expert	8	0.0027246140	0.0041217540	0.66x
Master	9	0.1090768800	0.0082719580	13.18x
Master	10	0.0017557680	0.0041979360	0.42x
Extreme	11	0.1211764000	0.0106921200	11.33x
Extreme	12	0.0377153000	0.0075583660	4.99x

4.1 Hardware Configuration

CPU: Dual Intel Xeon processors with a total of 72 threads. **GPU:** NVIDIA Tesla P100 GPUs with 16GB memory each.

These hardware resources were utilized under consistent conditions for benchmarking.

5 Discussion

When evaluating relatively small Sudoku boards (9x9) sourced from `https://sudoku.com`, the GPU performance compared to the CPU was not consistently superior across all difficulty levels. On easier boards, the GPU sometimes outperformed the CPU and sometimes did not. For example, with some Easy-level boards, the GPU was faster, while with others, the CPU was quicker. This variability can be attributed to the overhead of parallelization not always paying off for simpler or smaller problems, as well as inherent differences in how the CPU and GPU handle computational tasks per thread. Direct comparisons of absolute performance are challenging because the CPU and GPU have fundamentally different architectures, thread capabilities, and clock speeds. All the boards used for these benchmarks are listed in `benchmark_boards.txt`, ensuring reproducibility and transparency.

However, at the highest difficulty levels (e.g., Extreme), the GPU consistently demonstrated a clear advantage, offering an order-of-magnitude speedup compared to the CPU. This suggests that as the complexity of the puzzle increases, the benefits of parallel computation on the GPU become more pronounced. It is reasonable to expect that for larger board sizes (e.g., 16x16 Sudoku grids), where the search space and complexity are substantially greater, the GPU would likely offer even more significant performance gains.

6 Multiple Board Computations

The advantage of GPU parallelization becomes especially apparent when solving a large number of puzzles concurrently. To explore this, we generated 50,000 puzzles at the highest hardness level (5) using the `sudoku-puzzle` tool `https://github.com/DhanushNehru/sudoku-puzzle`, with the generation code provided in `generator.js`. For this large batch of complex puzzles, the results were as follows:

- **CPU:** 80.5698 seconds total
- **GPU:** 1.35919 seconds total

Here, the GPU shines by leveraging massive parallelization to handle tens of thousands of boards simultaneously. While per-board improvements might vary at smaller scales or easier difficulty levels, when the workload expands to large batches of challenging puzzles, the GPU delivers a dramatic reduction in total processing time.

7 Conclusion

This comparative study demonstrates that while GPU-based parallelization does not consistently outperform CPU-based approaches on single, small Sudoku boards—particularly at lower difficulty levels—it shows remarkable advantages under certain conditions. When individual boards are simple or have limited search spaces, the overhead associated with parallelization may offset any theoretical gains in speed. However, as the difficulty and complexity of the puzzles increase, the GPU-based approach increasingly outperforms the CPU. This is especially evident at the highest difficulty levels (such as "Extreme"), where the GPU not only rivaled but often surpassed the CPU by orders of magnitude in terms of execution time.

More importantly, the true strength of the GPU-based solution emerges when scaling to large numbers of boards. By solving tens of thousands of difficult puzzles in parallel, the GPU dramatically reduced the total computation time to a fraction of what the CPU required. In these large-scale scenarios, the GPU's massive parallelism shines, effectively amortizing the overhead and leveraging its concurrent execution model to deliver significant performance gains.

These findings suggest that while the immediate benefit of GPU acceleration may not always be evident when solving a single small puzzle, the GPU's advantages become increasingly pronounced as the problem scope broadens—either through more challenging puzzles or simply by processing many boards concurrently. For applications where large-scale puzzle solving or complex Sudoku variants are required, GPU-based methods offer a powerful and scalable solution.