CSE-4001 PARALLEL AND DISTRIBUTED COMPUTING

SUDOKU SOLVER

Project Component By:

Harshit Mishra (19BCE0799)

Kartik Goel (19BCE2002)

Alokam Nikhitha (19BCE2555)

Submitted to:

Dr. Hiteshwar Kumar Azad

School of Computer Science and Engineering (SCOPE)

Abstract:

Sudoku is a very commonly known game of puzzles. A classical sudoku contains a 9X9 grid and aims at filling range of numbers from 1-9 such that every row, every column and every 3X3 mini-grid has unique set of numbers without any repetitions. Sudoku is a logic-based puzzle where the puzzle setter provides a partially complete grid and one needs to complete the grid with above specified conditions in order to successfully complete the game. Generally, a well-posed sudoku has only a single solution.

The solution to a sudoku falls under the category of NP complete and thus the algorithms that have been proposed for solving it are computationally complex, that is, they have high time complexity. In our project we have implemented a parallel approach in order to solve sudoku using C and OpenMP. We also aim to identify the limitations and advantages in both serial and parallel processing approaches and draw a comparative study of both.

Domain Introduction:

In our project we are going to use Parallel Computing to solve a sudoku. Parallel Computing refers to the process of breaking down larger problems into smaller, independent often similar parts that can be executed simultaneously by multiple processors communicating via shared memory, the results of which are combined upon completion as part of an overall algorithm. The primary goal of parallel computing is to increase available computation power for faster application processing and problem solving.

Parallel computing infrastructure is typically housed within a single datacentre where several processors are installed in a server rack; computation requests are distributed in small chunks by the application server that are then executed simultaneously on each server.

The importance of parallel computing continues to grow with the increasing usage of multicore processors and GPUs. GPUs work together with CPUs to increase the throughput of data and the number of concurrent calculations within an application. Using the power of parallelism, a GPU can complete more work than a CPU in a given amount of time.

Techniques and its related Challenges:

There are quite a few pre-existing solutions to sudoku solving problem. These solutions include Brute Force with Backtracking, Naked Singles, Hidden Singles etc. But there are issues in the existing solutions. The brute force has many such limitations. The main one being increased execution time with increase in input size. In this algorithm, the time needed to find the solution depends upon the number of empty cells in the sudoku board. In various other algorithms, filling the naked singles can eliminate the choices in another empty cell because the number is already used in either the same row, column or mini-grid. The serial code guarantees to find the solution to the sudoku board if a solution exists because of elimination of empty cells by rule-based methods and then applying brute force to the remaining cells. The serial code will significantly slow down as the size of the Sudoku board increases or may not be able to produce the solution due to computational limitations of the machine.

Research Findings:

S. No	Title	Author	Summary	Drawbacks
1	Sudoku Game solving approach through parallel processing	Saxena R., Jain M. and Yaqub S.M.	The paper examines different answers for the issue alongside their advantages and disadvantages. The paper additionally examines the exhibition of a sequential variant arrangement as for execution time taken to explain Sudoku. With certain modifications and suspicions to the sequential form code, we have assessed the conceivable arrangement and bottleneck in the equal methodology. The outcomes have been assessed for the parallelized calculation on an ordinary client machine first, and afterward on a supercomputing arrangement box PARAM SHAVAK[1]	Because of combinatorial nature of the issue, it is intriguing to explore it with certain heuristics or options in contrast to the sequential methodology as the sequential code will significantly back off as the size of the Sudoku board increments or will most likely be unable to create the arrangement because of computational impediments of the machine.

2	A search based sudoku solver	Cazenave, T. and Labo, I.A.	Two search algorithms Forward Checking (FC) and Limited Discrepancy Search (LDS) are evaluated. There is an evident phase transition for 25x25 sudoku puzzles. The min-domain-sum ordering of values is a better approach than the ordering of values lexicographically. LDS is a better alternative for FC. The quasi-group completion problem (QCP) is solving an nxn latin square by unassigning variables. It follows an easy-hard-easy phase transition depending on the number of unassigned values. Sudoku can be modelled as a QCP with additional constraints. Sudoku problem generation is done using LDS by unassigning randomly selected variables till a certain percentage of variables is reached. For a given size and percentage around 50 problemas are possible. For QCP redundant modelling uses n^2 constraints which may be computationally infeasible. In FC every time a variable is assigned, one from the set of free variables from the same row, column and mini grid is removed. LDS traverses leaves in the search tree in increasing order of discrepancy. A discrepancy occurs when the algorithm does not follow the heuristics. [8]	LDS computes the solution for 1000 sudoku problems in less time compared to FC in both lexicographic and vdom+ value ordering heuristic. For FC the phase transition starts at 61% holes whereas it is almost unaffected in LDS. The limitation projected by this paper is the inability to generate puzzles with unique solutions. The time taken by LDS for a 25x25 puzzle in lex and vdom+ modes are 782 and 465 minutes respectively which can be reduced by solving it in parallel. [8]
3	Sudoku Solver by Q'tron Neural Networks	Yue, T.W. and Lee, Z.C.	Q'tron neural network is an energy driven model with its basic processing node being called a q'tron (quantum neuron). The rules of a sudoku puzzle are developed as a function of energy. This neural network can be used for solving sudoku puzzles as well as for generating them. Solving sudoku puzzles is an	This sudoku solver does not function correctly in the noise-free mode. Solving the same puzzle with varying initial conditions on free q'trons did not arrive at a solution for 1000 tests. In

			np-complete problem which means it is at least as hard as any np problem. Usually sudoku puzzles are solved algorithmically by using graph theories. In the q'tron neural network a solution is found when the energy is least in the energy landscape. This paper describes how to build a q'tron neural network to solve and generate sudoku puzzles. [9]	contrast when the solver is run in full mode, it always arrives at a solution. [9]
4	A SAT-based Sudoku Solver	Weber, T.	Since there are 6x10^21+ sudoku grids, a naive backtracking algorithm would be computationally infeasible. Here a sudoku puzzle is transformed into a propositional formula that can be satisfied if and only if the puzzle has a solution. This methodology is implemented using the Isabelle/HOL theorem prover by specifying its constraints. This propositional formula is then fed to a standard SAT solver after which the assignment is transformed into a solution.The propositional formula is a polynomial the size of the grid. The translation is a simple process as an existing framework is made use of. The puzzle is encoded by assigning 9 boolean variables for each of the cells in a 9x9 grid (total = 729 variables). These boolean variables are used to represent a boolean equation for a given constraint in the puzzle. [10]	Since solving a sudoku puzzle is an np-complete problem, no other algorithm offers better complexity. A 9x9 sudoku puzzle is solved in milliseconds. This method can extend to enumerate all possible solutions to any given sudoku problem. [10]

Approach:

We are going to implement a sudoku solver using parallel computing by using a combination of existing algorithms. In the parallel approach, several threads would be spawned to compute parallelly. C programming and OpenMP will be used to perform elimination, lone ranger and twins in the given sequence to get an optimized and faster performance. When the grid sizes reach a higher number like 25, the existing serial algorithm can take a lot of time which will be overcome by implementing it parallelly in this project. We would also be evaluating the improvement projected by parallelizing the code.