Sudoku Validation using Multi-threading

INTRODUCTION -

In this assignment the given task is to check the sudoku and output if it was a Valid Sudoku or not.
Sudoku is a popular puzzle game which requires validation of its N*N grid of numbers. It is required to validate its rows, columns and subgrids of size root N.

Three approaches implemented in main.c file are :

- Sequential Method
- Chunk Method
- Mixed Method

Chunk and Mixed method use multi-threading.

IMPLEMENTATION -

The logic for the implementation of all methods is written in main.c file. I used POSIX threads library to created threads for chunk and mixed method. Time.h library to capture the time taken for the process to run.

I have written methods -

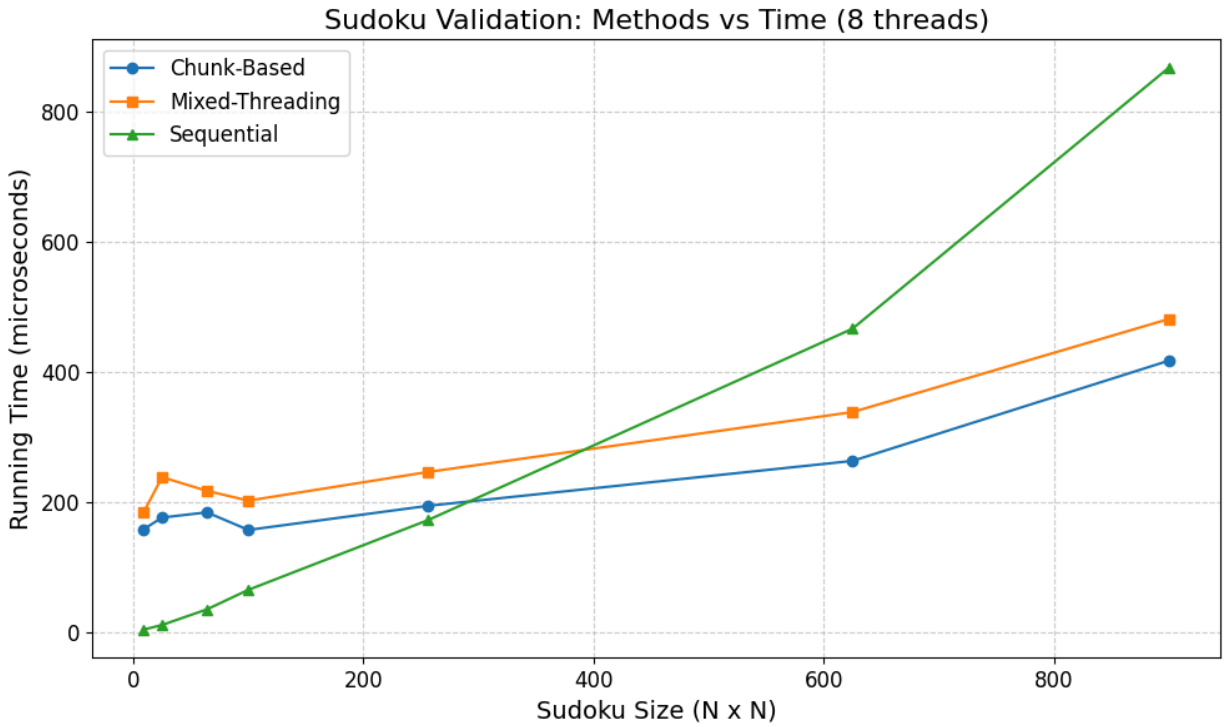
- Read_input_from_file - Reads input from input.txt file and creates a dynamic sudoku array of N*N
- Check_row - Takes input i and checks if row i is valid
- Check_col - Takes input i and checks if column i is valid
- Check_Sub_Grid - Takes input i and checks if row i is valid
- Validate_chunk - Handles threads by Chunk method
- Validate_mixed - Handles threads for Mixed method
- Sequential_method - Normal implementation of sudoku validation
- Chunk_method - Chunk based implementation
- Mixed_method - Mixed based implementation

3 output file - output_chunk.txt, output_mixed.txt, time_record.txt

EXPERIMENTAL SETUP (Laptop used : MacBook Air M2 (4 cores))-

Experiment-1

Effect of Sudoku Size keeping threads constant



1. Sequential shows exponential growth indicating inefficiency for large size sudoku

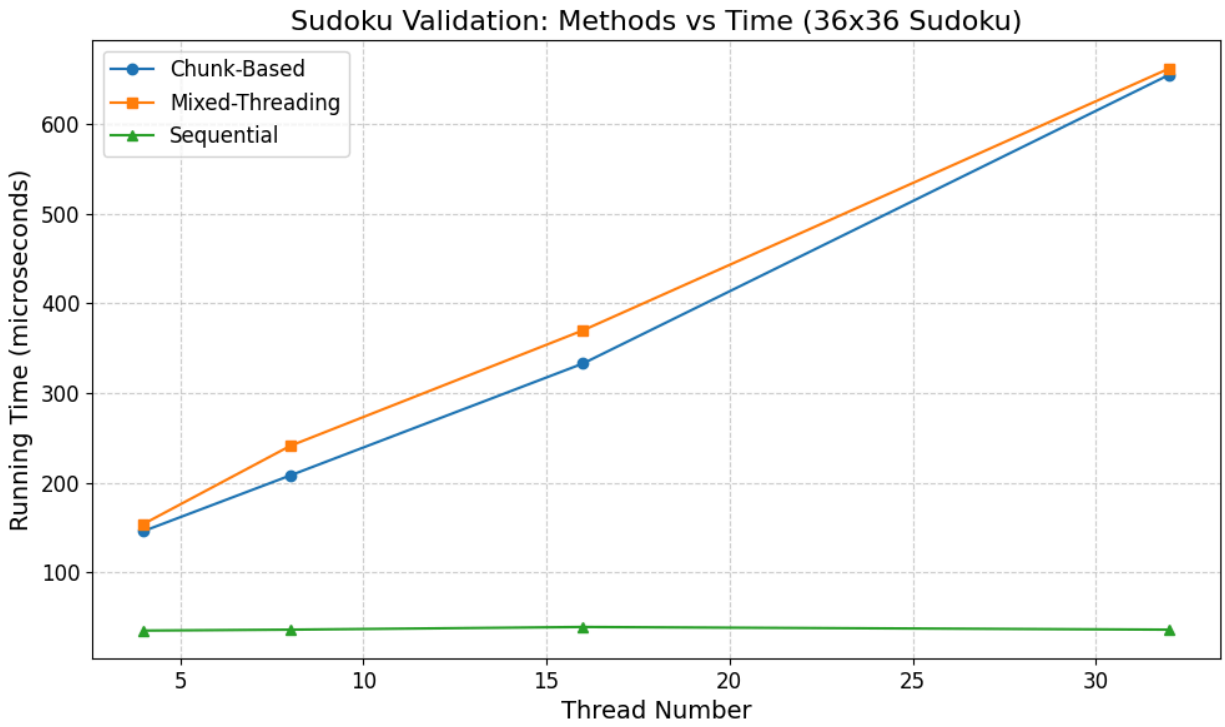
2. Methods using Multithreading that is chunk and mixed outperform sequential as the sudoku size increases above 256, but for smaller size sudoku Sequential method outperforms multithreading as thread creating overhead dominates

3. Both Chunk and Mixed methods behave in identical manner i.e they have same shape they are only shifted along by Y-axis.

4. It is visible from graph that chunk method is faster than mixed method. (Possible reasons discussed Page 4).

Experiment-2

Effect of increasing threads keeping sudoku size constant



1. As it is 36x36 sudoku due to thread creation overhead dominating sequential is faster than multithreading methods.

2. As threads are not used in Sequential method. Time taken is constant as number of threads increases.

3. My laptop has 4 cores so creating 4 threads is fastest as they can run simultaneously on 4 cores thus making maximum use of parallelism. As the thread number increases number of context switches also increases thus causing overhead and making code slower.

4. Both the mixed and chunk methods almost take the same time but here also chunk is slightly faster than mixed. (Possible reasons discussed in Page 4)

Probable reasons why Chunk based method is faster :

1. **Better Cache Utilization** - The M2 chip has unified memory architecture that benefits from accessing contiguous block of memory, reducing cache misses

2. **Better Thread Scheduling** - The mac OS scheduler efficiently handles chunk-based parallelism as it reduces context switches.

3. The Chunk-Based method likely allows the OS to allocate resources more efficiently since threads perform similar, predictable workloads.

Conclusion

The results indicate that multi-threading significantly improves Sudoku validation performance over sequential execution, especially for larger grids. Among the methods tested, the **Chunk-Based approach** proves to be the more efficient than mixed.