Sudoku Solution Validator: Single-Threaded vs. Multithreaded Performance

Suramya Raj Angdembay

May 7, 2025

1 Introduction

This report compares two implementations of a Sudoku solution validator:

- A **single-threaded** version (naive approach).
- A **multithreaded** version (27 threads for rows, columns, and subgrids).

2 Methodology

Both validators check:

- All rows contain digits 1–9 exactly once.
- All columns contain digits 1–9 exactly once.
- All 3×3 subgrids contain digits 1–9 exactly once.

Execution time is measured using `clock_gettime()` in milliseconds.

3 Code Implementation

3.1 Single-Threaded Validator

Key functions from the single-threaded validator:

Listing 1: Validation Logic

```c
// Check if a set of 9 numbers is valid
int is_valid_set(int *set) {
    int seen[10] = {0}; // 1-based indexing
    for (int i = 0; i < 9; i++) {
        int num = set[i];
        if (num < 1 || num > 9 || seen[num]++)
            return 0;
    }
    return 1;
```

```
}

// Validate all rows sequentially
int validate_rows() {
    for (int i = 0; i < 9; i++) {
        if (!is_valid_set(sudoku[i]))
            return 0;
    }
    return 1;
}
```

3.2 Multithreaded Validator

Key functions from the multithreaded validator:

Listing 2: Thread Workload

```
// Thread function for subgrid validation
void *validate_subgrid(void *param) {
    parameters *p = (parameters *)param;
    int validity[9] = {0};
    for (int i = p->row; i < p->row + 3; i++) {
        for (int j = p->column; j < p->column + 3; j++) {
            int num = sudoku[i][j];
            if (num < 1 || num > 9 || validity[num - 1]++) {
                free(param);
                pthread_exit(NULL);
            }
        }
    }
    valid[SUBGRID_OFFSET + (p->row / 3) * 3 + (p->column / 3)] = 1;
    free(param);
    pthread_exit(NULL);
}
```

4 Performance Analysis

4.1 Results

Validator Type	Time (ms)
Single-Threaded	0.013
Multithreaded (27 threads)	2.005

Table 1: Execution Time for 9×9 Sudoku

4.2 Why Multithreading is Slower

- **Thread Overhead**: Creating 27 threads dominates runtime.

- **Small Problem Size**: Validating 81 cells is too fast to benefit from parallelism.
- **False Sharing**: Threads contend for the global `valid` array.

5 Optimization Recommendations

- Use fewer threads (e.g., 11 instead of 27).
- Batch work (e.g., assign multiple rows to one thread).
- Switch to multithreading only for larger grids (e.g., 16×16).

6 Conclusion

For small Sudoku puzzles (9×9), a single-threaded validator is more efficient due to low overhead. Multithreading becomes advantageous for larger grids where parallelization offsets coordination costs.