COMP 429/529: Project 2

Berkay Barlas

April 20, 2019

Date Performed: March 17, 2019
Instructor: Didem Unat

In this assignment I implemented MPI and OpenMP versions on top of given serial version for Cardiac Electrophysiology Simulation and conducted experiments.

In this assignment I have completed

- 1D MPI version of Cardiac Electrophysiology Simulation
- 2D MPI version of Cardiac Electrophysiology Simulation
- 2D MPI+OpenMP version of Cardiac Electrophysiology Simulation
- Performance Studies for Part a, b, c, d, e

1 Implementation

In the first part of this assignment I implemented a parallel version of a simple image blurring algorithm with OpenMP which takes an input image and outputs a blurred image.

I used #pragma omp for collapse() in getGaussian(), loadImage(), saveImage(), applyFilter(), averageRGB() methods since all of have nested for loops that can be parallelized. The biggest nested for loop is in applyFilter() and it can be parallelizable as below. collapse(5) can not be used because last two for loops depends on previous loops.

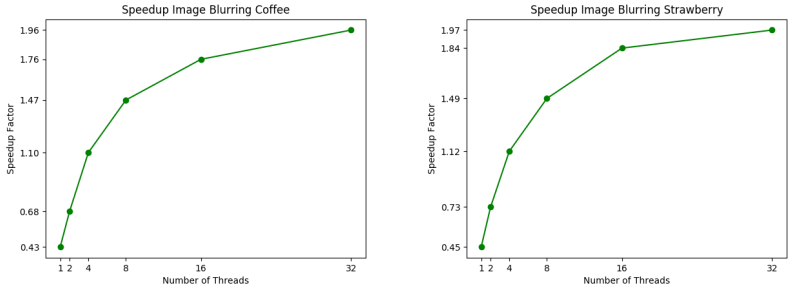
```
1    s
```

2 Studies

2.1 Part A: Strong Scaling

optimal processor geometry:

Results



(a) Speedup results for the blurring on coffee image.

(b) Speedup results for the blurring on strawberry image.

Figure 1: Speedup figures for image blurring application

Explanation

2.2 PART B: Strong Scaling with ntasks-per-node=16

In the second part of this assignment,

```
if(matrix[row][col] != EMPTY) {

}
```

2.3 Part C: Disabling Communication

optimal processor geometry:

2.4 Part D: Single and Dual node Performance with OpenMP

optimal processor geometry:

2.5 Part E: Disabling Communication

optimal processor geometry:

2.6 Tests on Sudoku Problems of Different Grids

Part-B

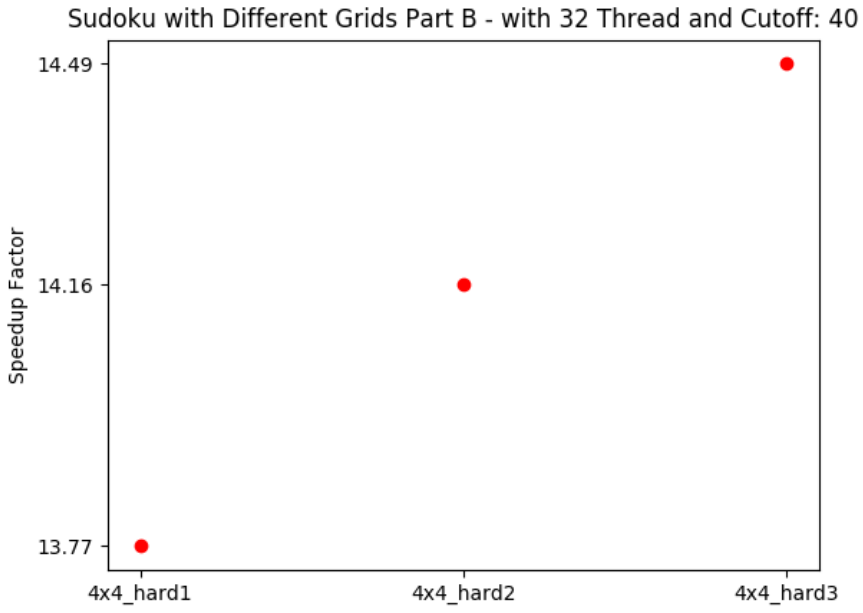


Figure 2: Results for the 32 Thread Parallel Sudoku solver in Part B with different sizes and difficulties.

When the difficulties of sudoku problem increases parallized algorithm in Part B performs better over serial serial version. Effectiveness of parallization

increases beacuse when the task difficulty increased the execution time ratio of parallelizable partion over non-parallelizable partion increases.

3 Formulas Used

a. *Speedup*

$$Speedup = \frac{\mathrm{T1}}{\mathrm{Tp}}$$

b. *Amdahl's Law*

$$Tp \geq Wserial + \frac{\mathrm{Wparallel}}{\mathrm{P}}$$