COMP 429/529: Project 1

Berkay Barlas

March 19, 2019

Date Performed:	March 17, 2019
Instructor:	Didem Unat

In this assignment I developed my parallel implementations on top of given serial version for two different applications;
an image blurring algorithm and sudoku solver using OpenMP.
While the first application in data parallelism, the second application in task parallelism.

In this assignment I have completed

- Parallel Version of Image Blurring
- Performance Study for Part I
- Parallel Version of Sudoku Part A, Part B, Part C
- Performance Study for Part II

1 Part I: Image Blurring

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.

The image is represented as a 2-dimensional grid with three components. After reading the image to be filtered, the program generates an n by n filter. The filter is then applied to blur every pixel in the image. For pixels which are located along/near the edges of the image, zero padding technique is used to add additional zero-valued pixels beyond the edges of the image.

1.1 Stability Test

Serial version execution time:

Coffee Image: 13.64
Strawberry Image: 27.56

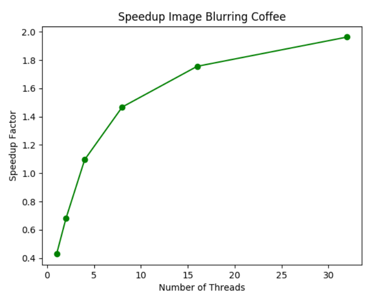
Paralel version with single thread execution time:

Coffee Image: 30.88
Strawberry Image: 55.93

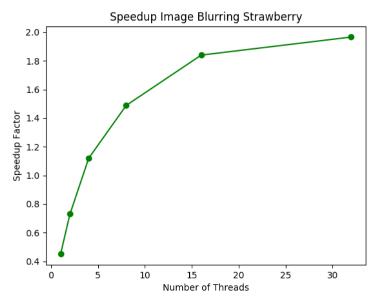
Which thread number gives best performance?

32 thread count gives best performance for both blurring applications. The reason of serial version performs better than parallel version with 1 thread is Parallization overhead. The difference between them caused by the execution time of parallization.

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 of Speedup Curve

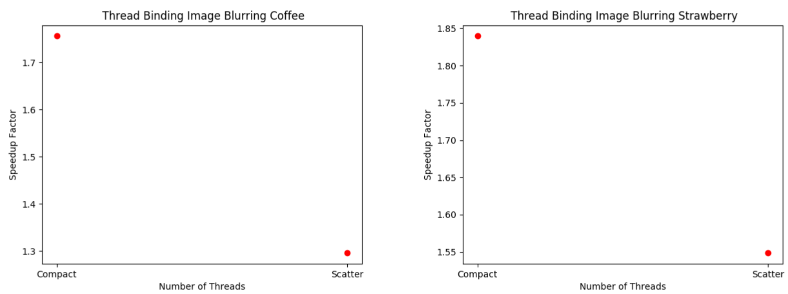
Even though, a linear/perfect speedup is not expected the results are actually worse than what is expected.
The reasons of that is some part of the code is non-parallelizable such as ...
The 32 thread parallel version on 32 core cluster gives only around 2x speedup.

1.2 Thread Binding Test

In the compact mapping, multiple threads are mapped as close as possible to each other, while in the scatter mapping, the threads are distributed as evenly as possible across all cores.

Different mapping strategies; Compact and Scatter

Results



(a) Results for the blurring on coffee image.

(b) Results for the blurring on strawberry image.

Figure 2: Speedup figures for image blurring application

Which Mapping Gives Better Performance, Why?

Compact gives better performance for both images because when neighbouring threads are accessing the same or nearby data; the data which is brought into the cache by one thread can be used by the other, avoiding a costly memory access.
Talk about Caching.

2 PART II: Parallel Sudoku Solver

In the second part of this assignment, I parallelized a serial sudoku solver with OpenMP which takes a sudoku problem as an input and finds all possible solutions from it using a brute force search for searching by all possible solutions to the problem.

2.1 Scalability Test

2.1.1 Part A

Serial version execution time: 48.10

Paralel version with single thread execution time: 78.69

Which thread number gives best performance?
32 thread count gives best performance.

Results

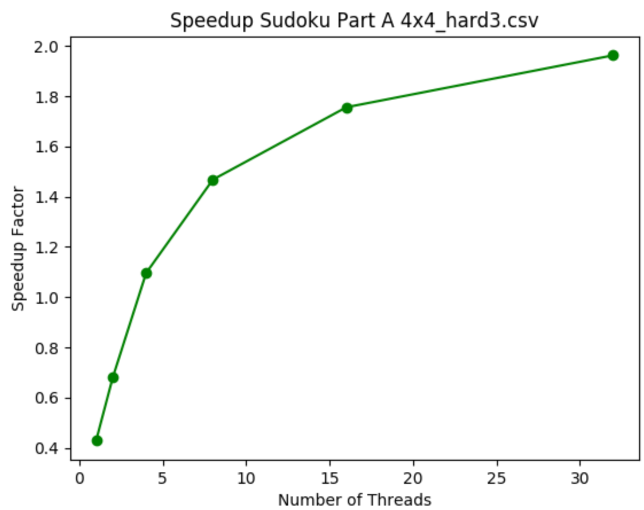


Figure 3: Results for the Sudoku 4x4hard3 using algorithm in.

Explanation of Speedup Curve
The task Parallization of serial version results in the creation of too many different task which causes to a great overhead. Therefore, the speedup results are lower than expected. Even with 32 thread speedup is just around 2.
WHY IS NOT LINEAR

2.1.2 Part B

Serial version execution time: 48.10

Paralel version with single thread execution time: 74.50

Which thread number gives best performance?
32 thread count gives best performance.

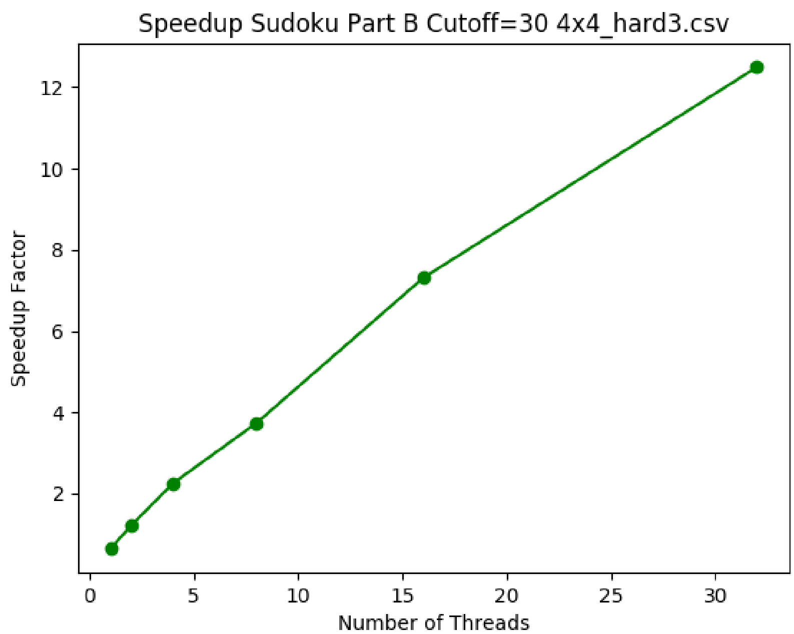


Figure 4: Results for the blurring on strawberry image.

Explanation of Speedup Curve
In order to improve the performance of the previous parallel version, a cutoff parameter to limit the number of parallel tasks is needed.

I defined a variable called depth and passed it as parameter to recursive method to prevent task creation after certain depth in the call-path tree. After that depth switch to the serial execution and do not generate more tasks. To determine that cut off parameter, I executed parallel program with several different values.

The speedup curve is linear which is expected.

2.1.3 Part C

Stopping the execution after finding a solution is very easy for serial version which can be done by returning a different value inside of for loops when a solution is found.

In order to guraantee single solution in parallel version a shared variable 'found' which will stop further task creation and execution can be defined.

Serial version execution time: 0.33

Paralel version with single thread execution time: 0.57

Which thread number gives best performance?
32 thread count gives best performance.

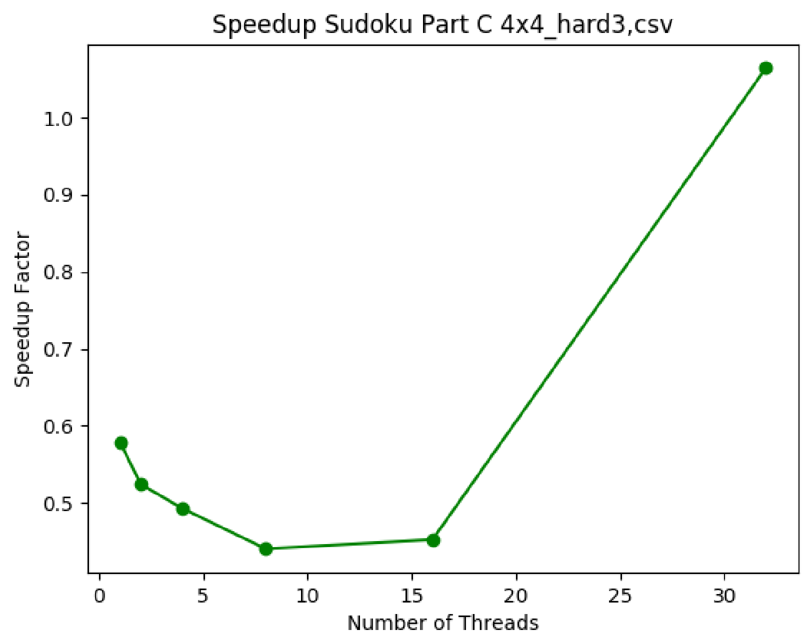


Figure 5: Results for the blurring on strawberry image.

Explanation of Speedup Curve

2.2 Thread Binding Test

2.2.1 Part A

Different mapping strategies; Compact and Scatter

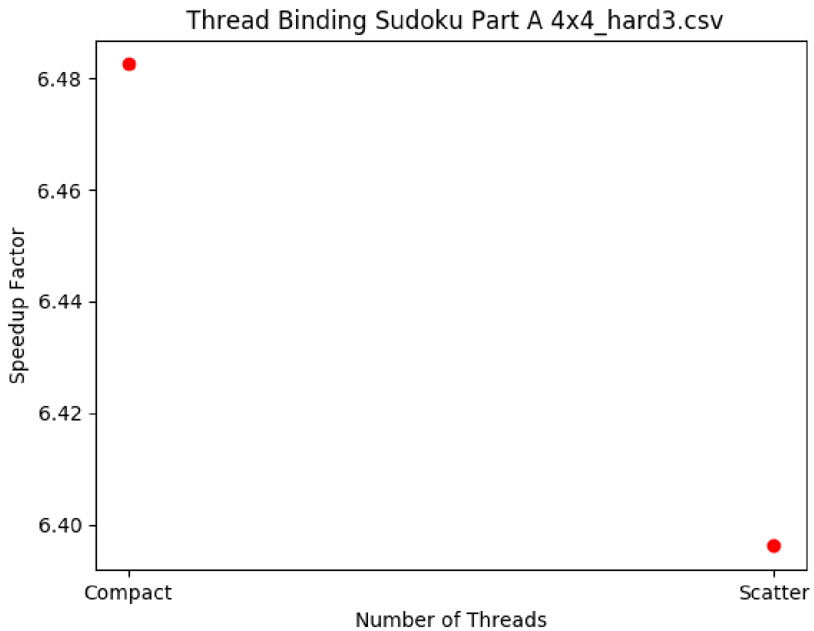


Figure 6: Results for the blurring on strawberry image.

Which Mapping Gives Better Performance, Why?

Compact gives better performance for both images because when neighbouring threads are accessing the same or nearby data; the data which is brought into the cache by one thread can be used by the other, avoiding a costly memory access.
Talk about Caching.

2.2.2 Part B

Different mapping strategies; Compact and Scatter

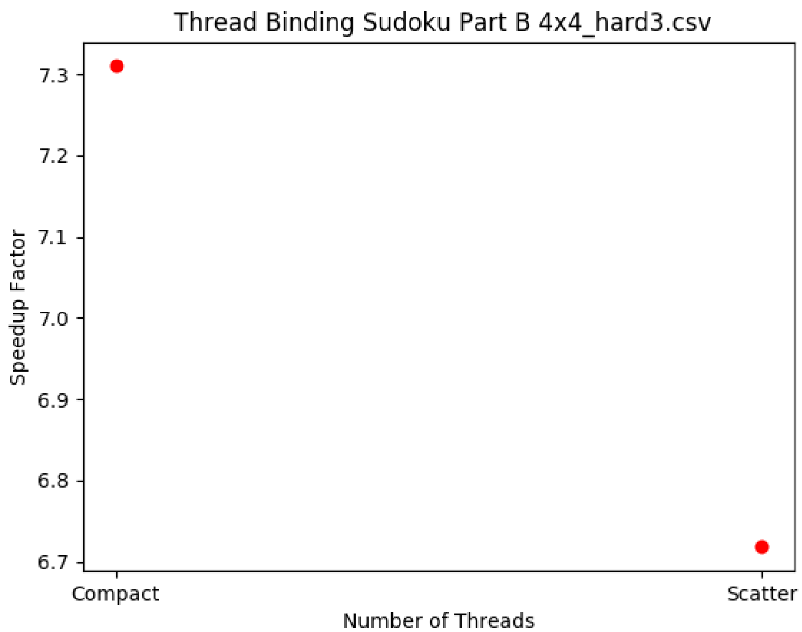


Figure 7: Results for the blurring on strawberry image.

Which Mapping Gives Better Performance, Why?

Compact gives better performance for both images because when neighbouring threads are accessing the same or nearby data; the data which is brought into the cache by one thread can be used by the other, avoiding a costly memory access.
Talk about Caching.

2.2.3 Part C

Different mapping strategies; Compact and Scatter

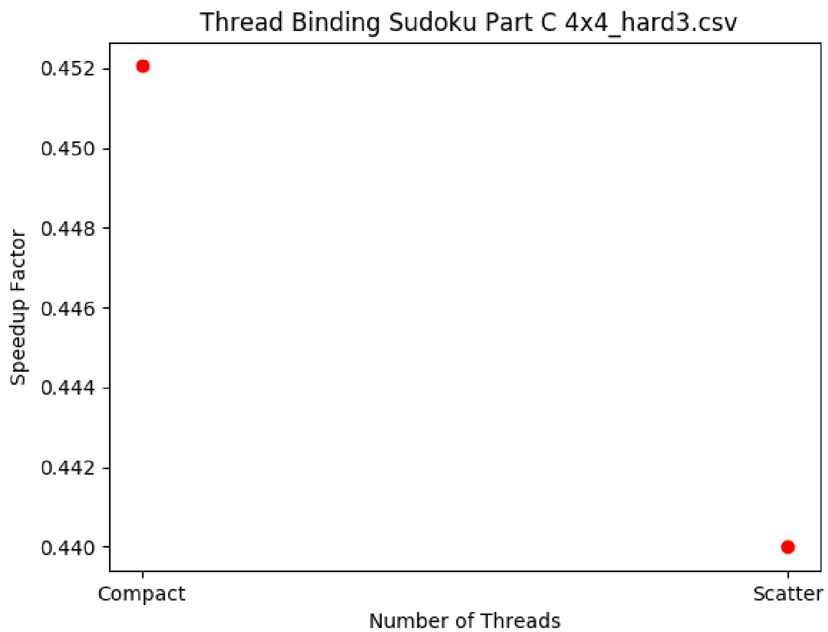


Figure 8: Results for the blurring on strawberry image.

Which Mapping Gives Better Performance, Why?

Compact gives better performance for both images because when neighbouring threads are accessing the same or nearby data; the data which is brought into the cache by one thread can be used by the other, avoiding a costly memory access.
Talk about Caching.

2.3 Tests on Sudoku Problems of Different Grids

Part-B

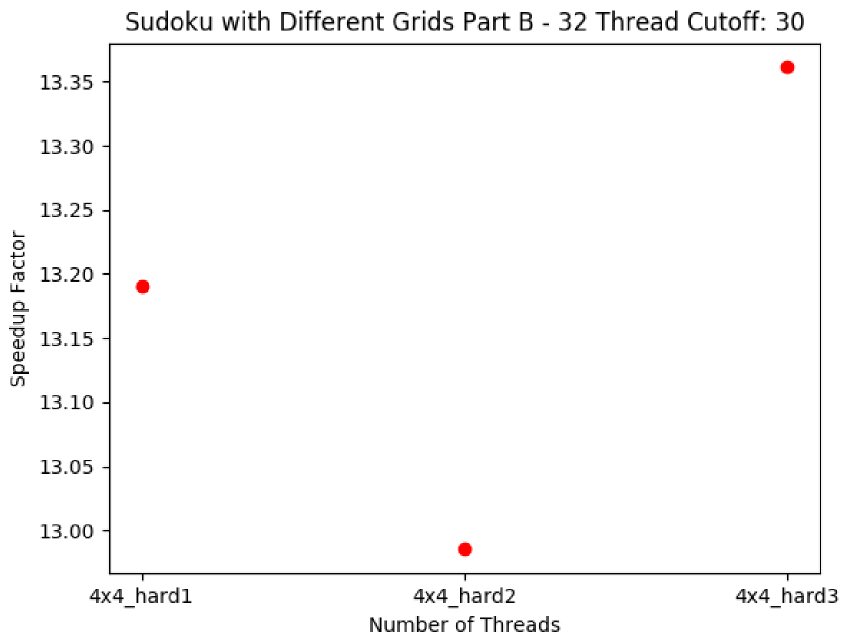


Figure 9: Results for the 32 Thread Parallel Sudoku solver in Part B with different sizes and difficulties.

3 Formulas Used

a. *Speedup*

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

$$\mathrm{Speedup}=\mathrm{T}_1/\mathrm{Tp}$$