PARALLEL AND DISTRIBUTED COMPUTING

DIGITAL ASSIGNMENT 1 AND 2

GROUP MEMBERS: Aditya Chitlangia **(16 BCE 1143)**

Dhruv Garg **(16 BCE 1190)**

QUESTION: Identify the performance scaling with respect to Amdhal's Law, Gustafson's law and Karp-Flatt metric for multiple problems.

THEORY:

Amdahl's law can be formulated in the following way:

Where

$$\boxed{S_{\text{latency}}(s) = \frac{1}{(1-p)+\frac{p}{s}}}$$

- S_{latency} is the theoretical speedup of the execution of the whole task;
- s is the speedup of the part of the task that benefits from improved system resources;
- p is the proportion of execution time that the part benefiting from improved resources originally occupied.

Gustafson Law

Gustafson estimated the speedup S gained by using N processors (instead of just one) for a task with a serial fraction s (which does not benefit from parallelism) as follows:

$$\boxed{S = N + (1-N)s}$$

Using different variables, Gustafson's law can be formulated the following way:

$$\boxed{S_{\text{latency}}(s) = 1 - p + sp,}$$

where

- S_{latency} is the theoretical speedup in latency of the execution of the whole task;
- s is the speedup in latency of the execution of the part of the task that benefits from the improvement of the resources of the system;
- p is the percentage of the execution workload of the whole task concerning the part that benefits from the improvement of the resources of the system *before the improvement*.

Karp–Flatt metric

Given a parallel computation exhibiting **speedup** and **p** processors, where **p** > 1, the experimentally determined serial fraction **e** is defined to be the Karp–Flatt Metric viz:

$$\boxed{e = \frac{\frac{1}{\psi} - \frac{1}{p}}{1 - \frac{1}{p}}}$$

The less the value of **e** the better the parallelization.

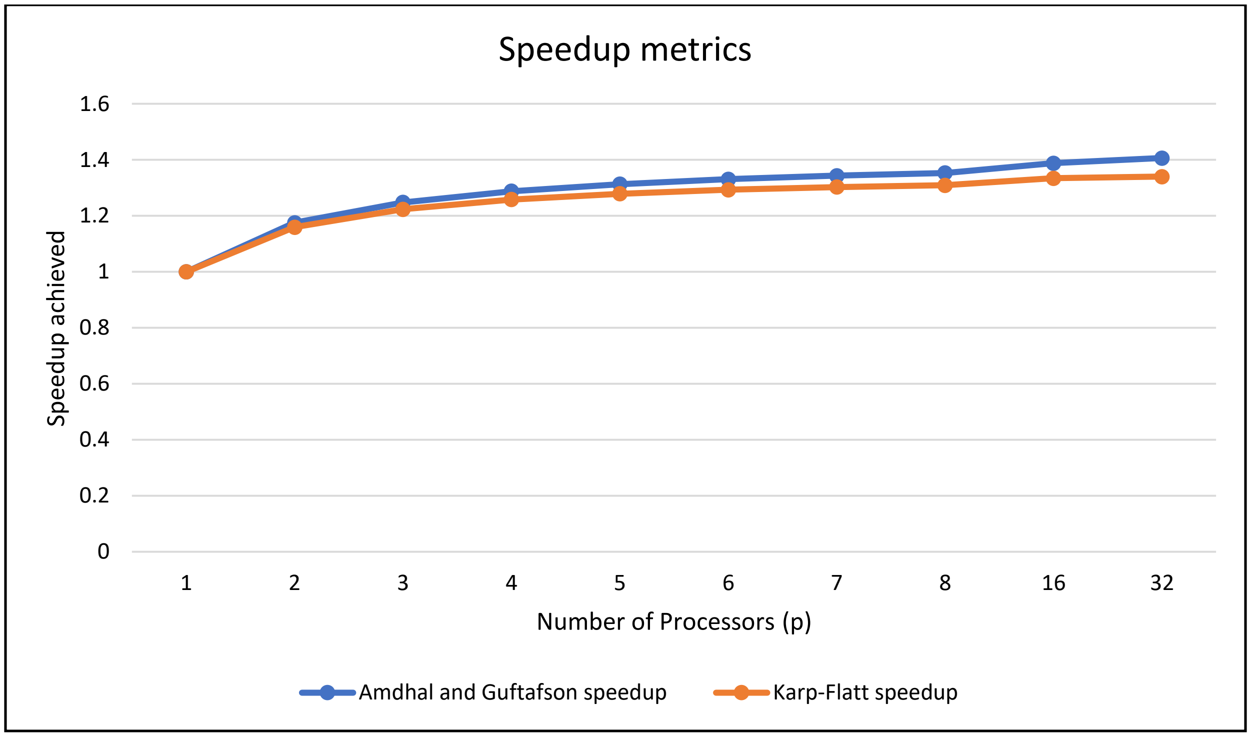
PROBLEM 1: Data decomposition problem

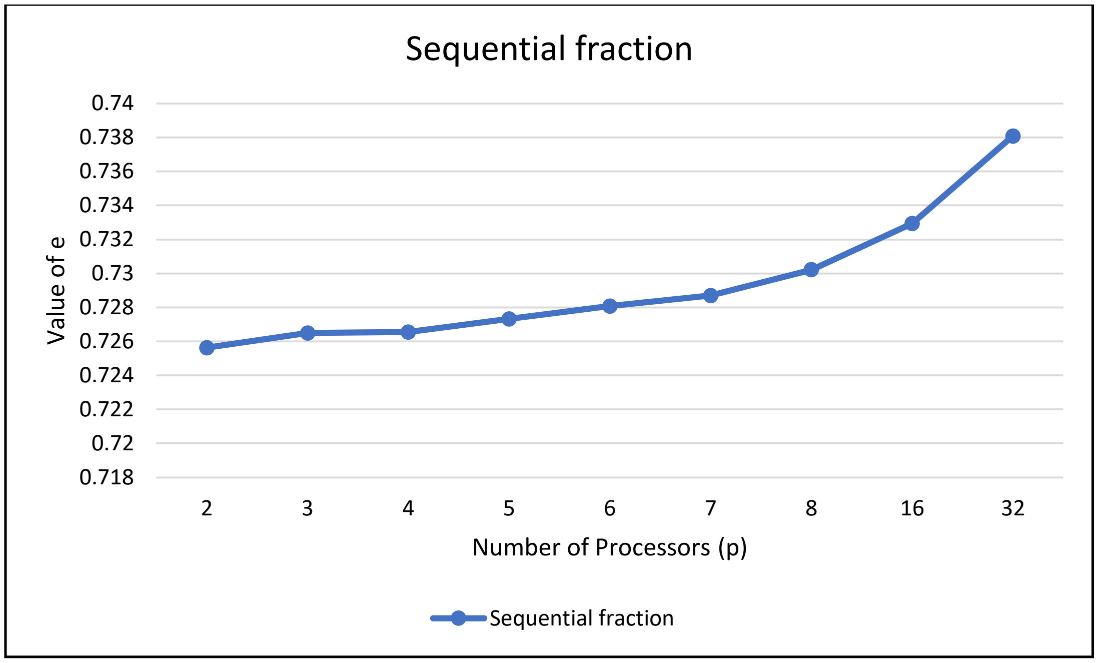
Sieve of Eratosthenes (For N = 50)

The OpenMP Implementation of Sieve of Eratosthenes method in the form of prime number programs. The main idea of performance analysis and comparison depends on number of processors, varying inputs and OpenMP implementation of prime number programs. This work helps us to design suitable sizing of based Sieve of Eratosthenes method over cluster and grid platform and suggest the suitability of the approach in usage of block-based Sieve of Eratosthenes method in production environments.

Serial fraction of code: 153/218=0.701834

Parallel fraction of code: 65/218= 0.298165





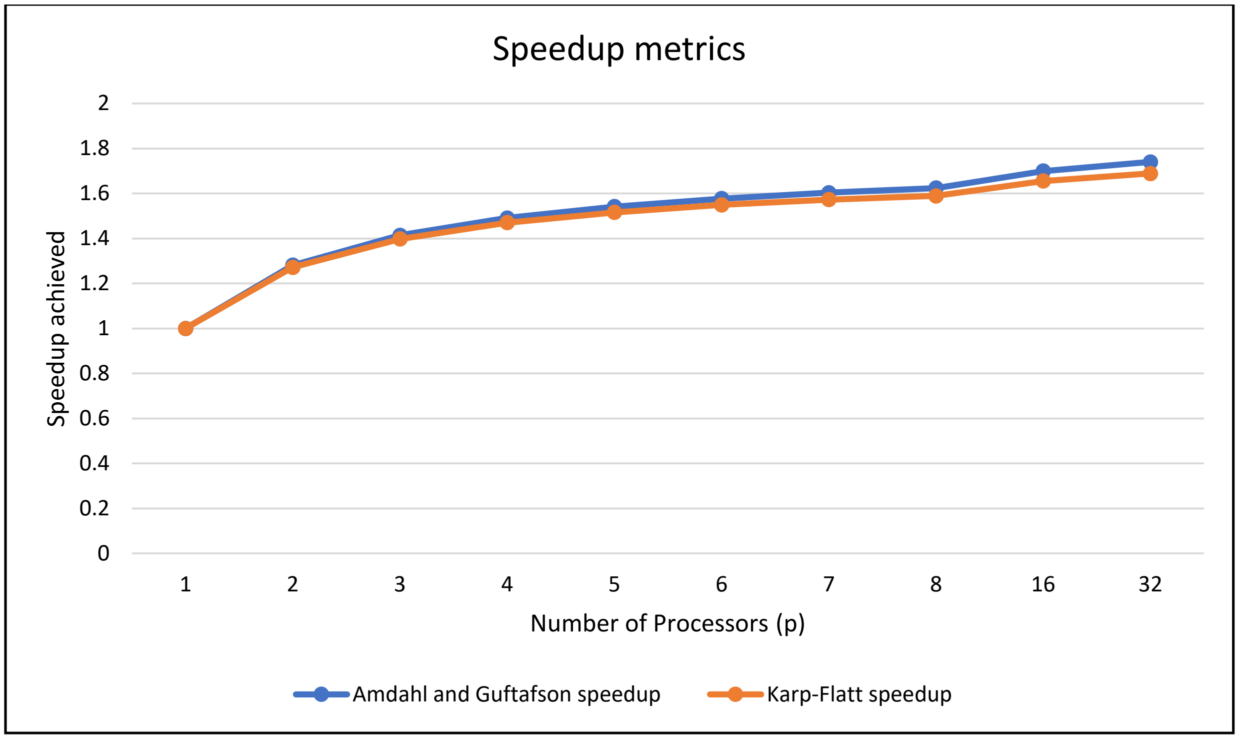
PROBLEM 2: **An exploratory problem**

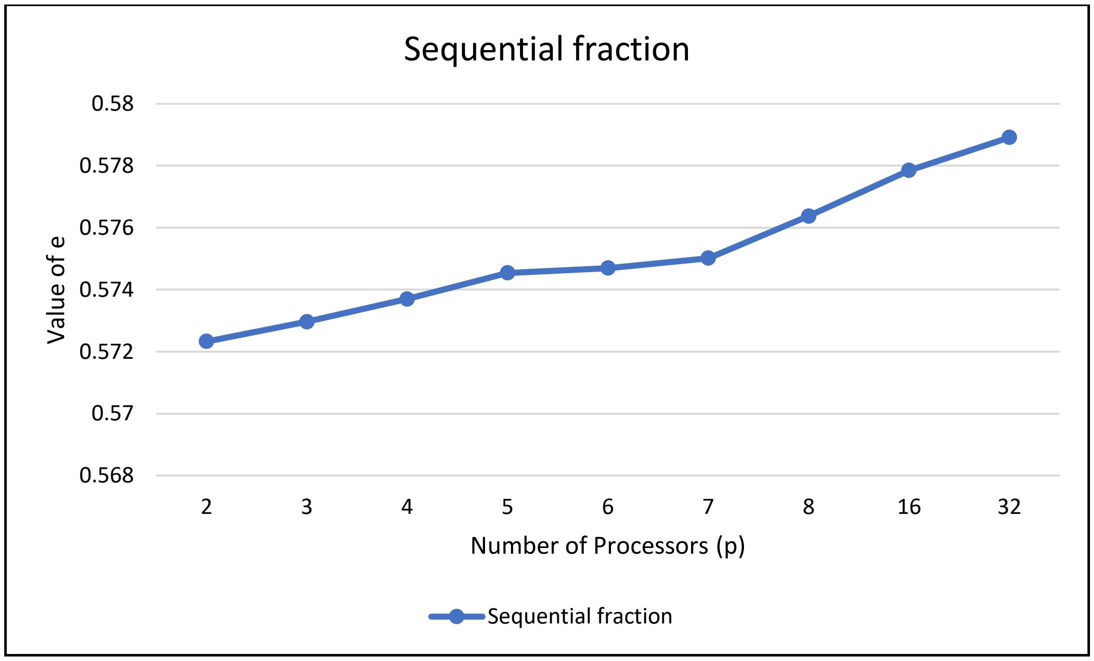
Dijkstra's algorithm (For matrix of dimension 5 x 5)

The problem of parallelization of Dijkstra's algorithm, a well-known algorithm for computing single-source shortest path in a graph. Dijkstra's algorithm can be applied to graphs with a various number of vertices and edges. Dijkstra's shortest path algorithm is implemented and presented, and the performances of its parallel and serial execution are compared. The algorithm implementation was parallelized using OpenMP. Its performances were measured on different configurations based on number of processors. The experimental results prove that the parallel execution of the algorithm has good performances in terms of speed-up ratio, when compared to its serial execution. Finally, the results show that, because of Dijkstra's algorithm in itself is sequential, and difficult to parallelize, average speed-up ratio achieved by parallelization was moderate. This proves to be a huge disadvantage of this algorithm, because its use is widespread, and enhancing its performance would have great effects in its many uses.

Serial fraction of code: 57/89= 0.640449

Parallel fraction of code: 32/89= 0.359550





PROBLEM 3: Boundary value problem

Heated Plate Problem (For matix of dimension 50 x 50)

Heated Plate problem is a C program which illustrates the use of the OpenMP application program interface by employing an iteration that solves the 2D steady state heat equation.

The region is covered with a grid of M by N nodes, and an N by N array W is used to record the temperature. The correspondence between array indices and locations in the region is suggested by giving the indices of the four corners:

```
I = 0
    [0][0] -------------[0][N-1]
           |                |
    J = 0  |                |  J = N-1
           |                |
    [M-1][0]-----------[M-1][N-1]
I = M-1
```

The steady state solution to the discrete heat equation satisfies the following condition at an interior grid point:

W[Central] = (1/4) * (W[North] + W[South] + W[East] + W[West])

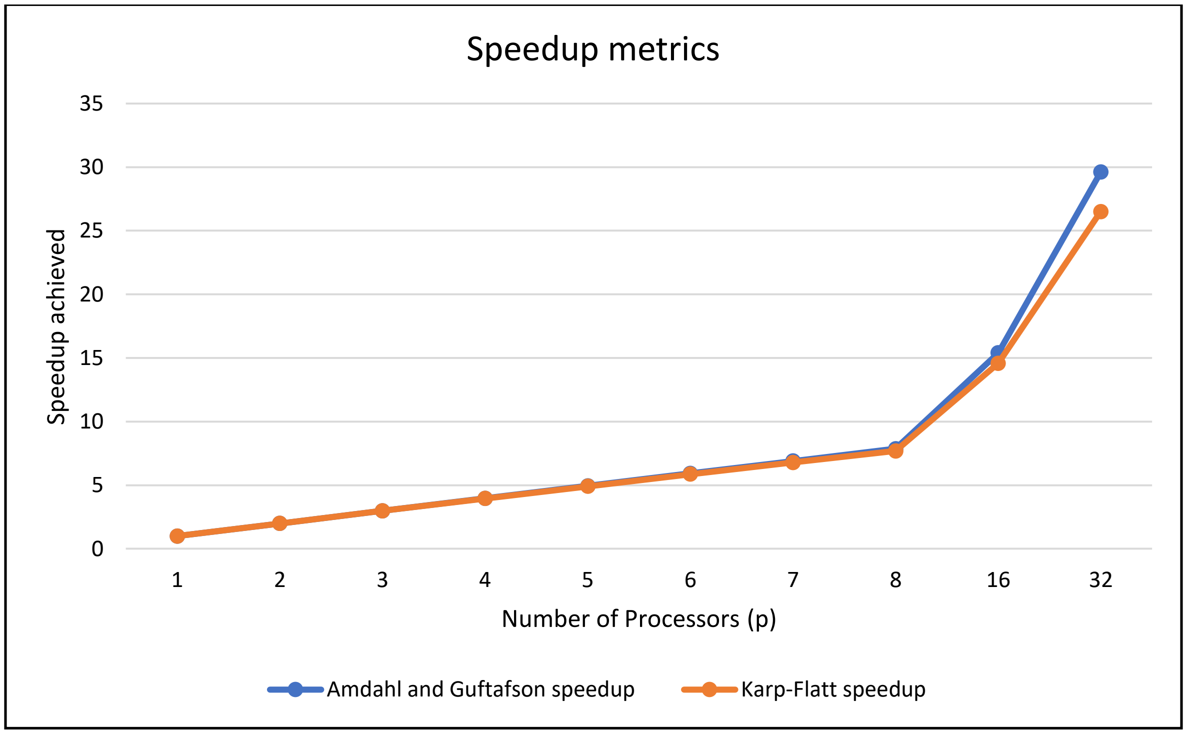
Where "Central" is the index of the grid point," North" is the index of its immediate neighbor to the " north ", and so on. Given an approximate solution of the steady state heat equation, a "better" solution is given by replacing each interior point by the average of its 4 neighbors - in other words, by using the condition as an ASSIGNMENT statement:

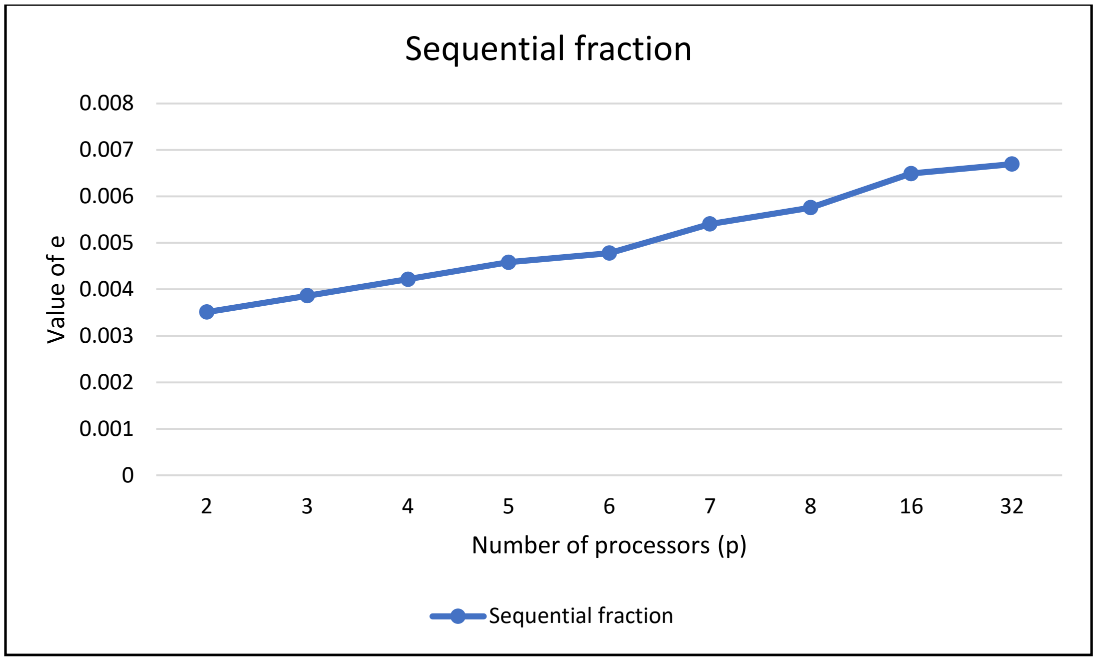
W[Central] <= (1/4) * (W[North] + W[South] + W[East] + W[West])

If this process is repeated often enough ,the difference between successive estimates of the solution will go zero. This program carries out such an iteration, using a tolerance specified by the user, and writes the final estimate of the solution to a file that can be used for graphic processing.

Serial fraction of code: 27/10329= 0.002613

Parallel fraction of code: 10302/10329= 0.997386





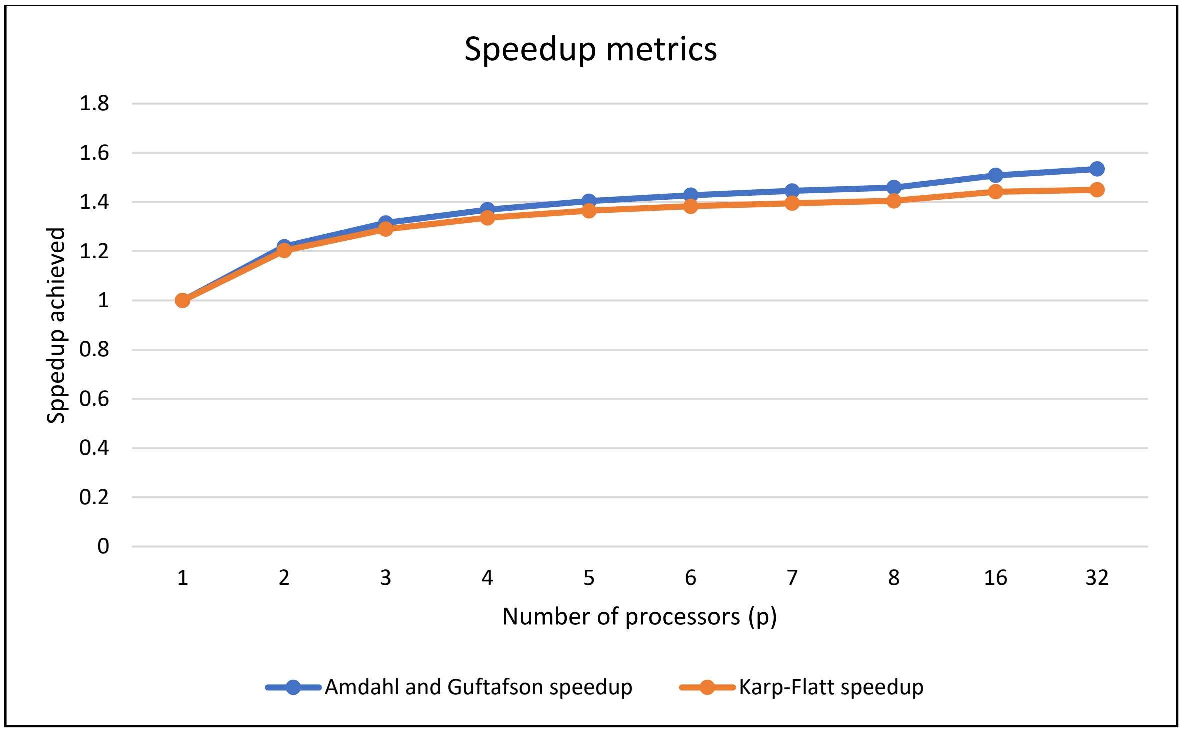
PROBLEM 4: Checkerboard decomposition problem

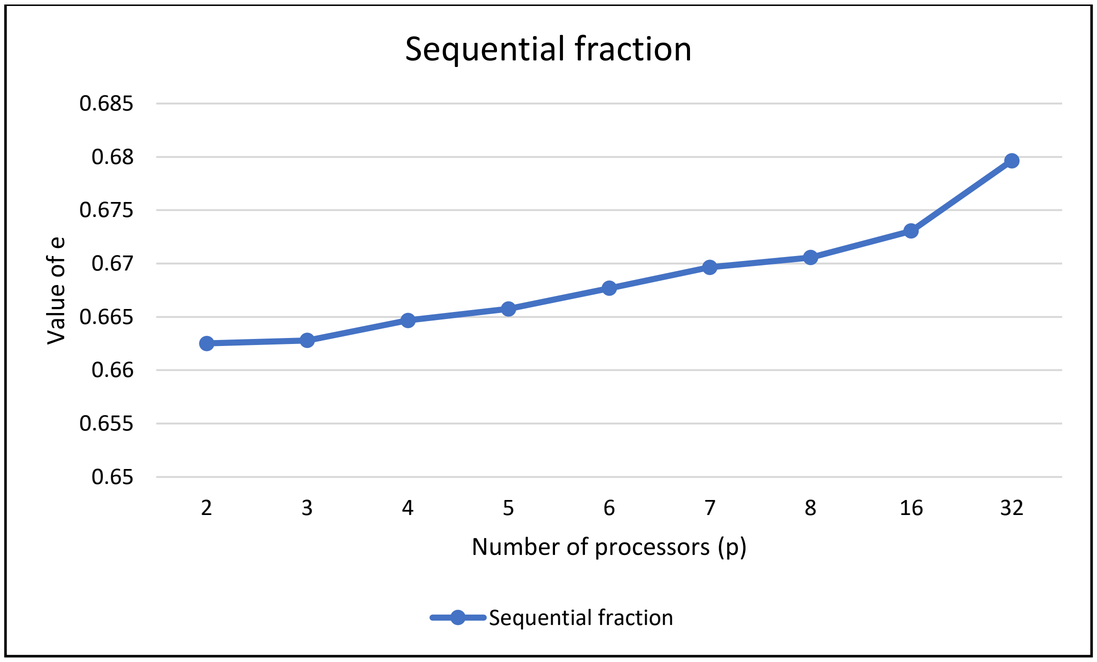
Matrix - vector multiplication (For Vecor: 1 x 5, and Matrix of dimension: 5 x 4)

The matrix - vector multiplication is one of the most fundamental and important problems in science and engineering. We present basic parallel implementation and a variation for matrix – vector multiplication. We evaluated and compared the performance of the implementations on OpenMP shared memory based on different number of processors. Further, we analyzed performance with 3 different performance models. These analyses have identified cost of reading of data from disk and communication cost as the primary factors affecting performance of the basic parallel matrix – vector implementation.

Serial fraction of code: 57/89= 0.640449

Parallel fraction of code: 32/89= 0.359550





NOTE: For further reference for calculations and plots, the Excel file has also been attached.