R.V. COLLEGE OF ENGINEERING
BENGALURU – 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

"COMPARISON OF PERFORMANCE OF FAST-FOURIER TRANSFORM BETWEEN SERIAL AND PARALLEL (MPI) "

PADP (16CS71) ASSIGNMENT REPORT

Submitted by

Name	USN
ADITYA PATI	1RV17CS009
AISHRITH RAO	1RV17CS011

Under the Guidance of

Dr. Minal Moharir
Professor, PADP

Department of Computer Science & Engineering

RV COLLEGE OF ENGINEERING®, BENGALURU - 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the **Open-Ended Experiment** titled "**COMPARISON OF PERFORMANCE OF FAST-FOURIER TRANSFORM BETWEEN SERIAL AND PARALLEL (MPI)**" has been carried out by **Aditya Pati (1RV17CS009)**, **Aishrith Rao (1RV17CS011),** bonafide students of RV College of Engineering, Bengaluru, have submitted in partial fulfillment for the **Internal Assessment of Course: PADP (16CS71)** during the year 2020-2021. It is certified that all corrections/suggestions indicated for the internal Assessment have been incorporated in the report.

Dr. Minal Moharir	Dr. Ramakanth Kumar P
Faculty Incharge,	Professor & Head
Department of CSE,	Department of CSE,
R.V.C.E., Bengaluru–59	R.V.C.E., Bengaluru–59

RV COLLEGE OF ENGINEERING® , BENGALURU - 560059

(Autonomous Institution Affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **Aditya Pati(1RV17CS009), Aishrith P Rao (1RV17CS011)** the students of Seventh Semester B.E., Computer Science and Engineering, R.V. College of Engineering, Bengaluru hereby declare that the mini-project titled **"COMPARISON OF PERFORMANCE OF FAST-FOURIER TRANSFORM BETWEEN SERIAL AND PARALLEL (MPI)"** has been carried out by us and submitted in partial fulfillment for the **Internal Assessment of Course: PADP (16CS71) - Open-Ended Experiment** during the year 2020-2021. We do declare that matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Place: Bengaluru **Aditya Pati, Aishrith P Rao**

Date: 17/01/2020

INTRODUCTION

A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

The DFT is obtained by decomposing a sequence of values into components of different frequencies.This operation is useful in many fields, but computing it directly from the definition is often too slow to be practical. An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors.

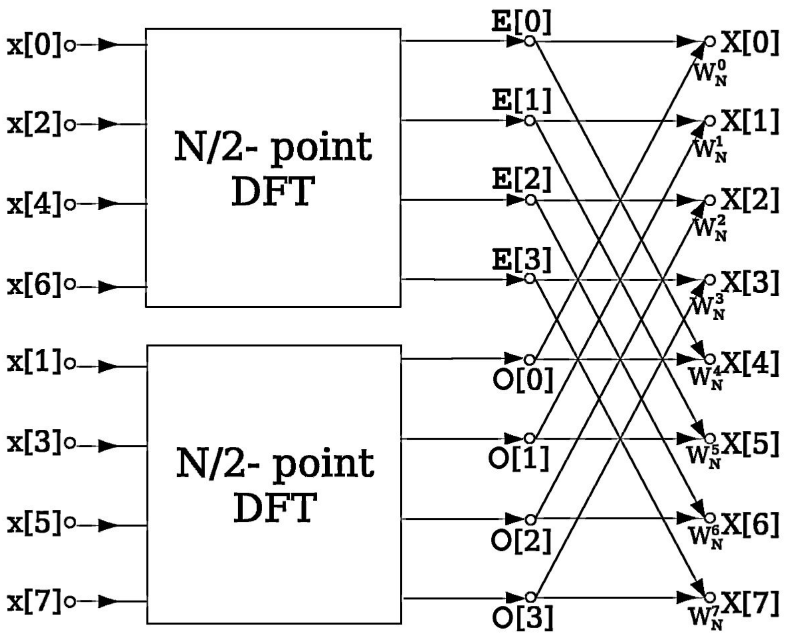
As a result, it manages to reduce the complexity of computing the DFT from N^{2} which arises if one simply applies the definition of DFT, to O(Nlog N), where N is the data size.

As a result, it manages to reduce the complexity of computing the DFT from its data size. The difference in speed can be enormous, especially for long data sets where *N* may be in the thousands or millions. In the presence of round-off error, many FFT algorithms are much more accurate than evaluating the DFT definition directly or indirectly. There are many different FFT algorithms based on a wide range of published theories, from simple complex-number arithmetic to group theory and number theory.

Fast Fourier transforms are widely used for applications in engineering, music, science, and mathematics. The basic ideas were popularized in 1965, but some algorithms had been derived as early as 1805. In 1994, Gilbert Strang described the FFT as "the most important numerical algorithm of our lifetime"

METHODOLOGY

Contains Serial and Parallel Version (openMPI) of the Fast Fourier Transform algorithm. Used to convert a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. A performance analysis was performed to show display execution time, speed-up, and efficiency.



SERIAL CODE

Sequentially performs the FFT algorithm and records time. Execution time was recorded using the TimeSource.c and timer.h files.

PARALLEL CODE

Parallel implementation of the FFT written in C using openMPI. Execution time was recorded using MPI's time function MPI_Wtime().

MASTER CODE

The master process creates a table of real and imaginary numbers. The master process then distributes a portion of the table to each individual process. The master process then gathers all the calculated information from the slave processes. Then calculates the last part of the FFT which requires all the imaginary and real numbers to be summed. The master process is in charge of recording the execution time.

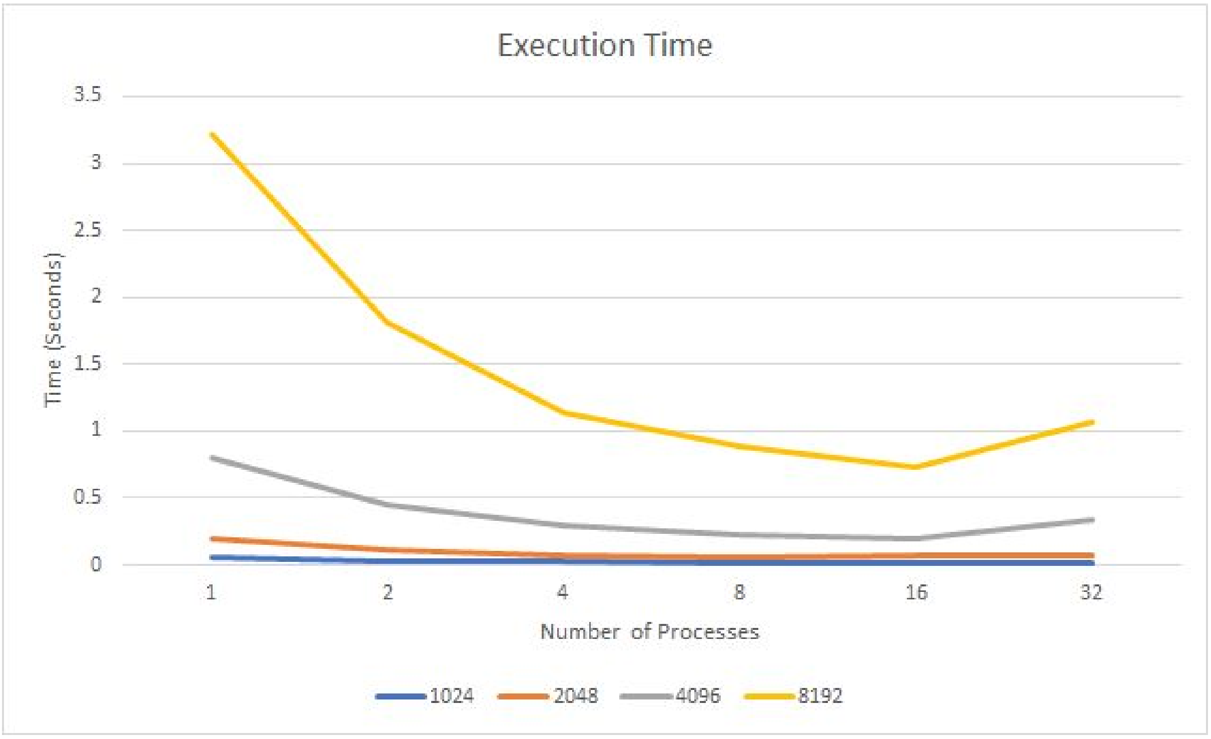
SLAVE PROCESSES

The slaves processes recieves a portion of the table that hold imaginary and real values. The slave processes then perform the FFT algorithm on thier portion. This portion is later gathered by the master process.

PERFORMANCE AND ANALYTICS

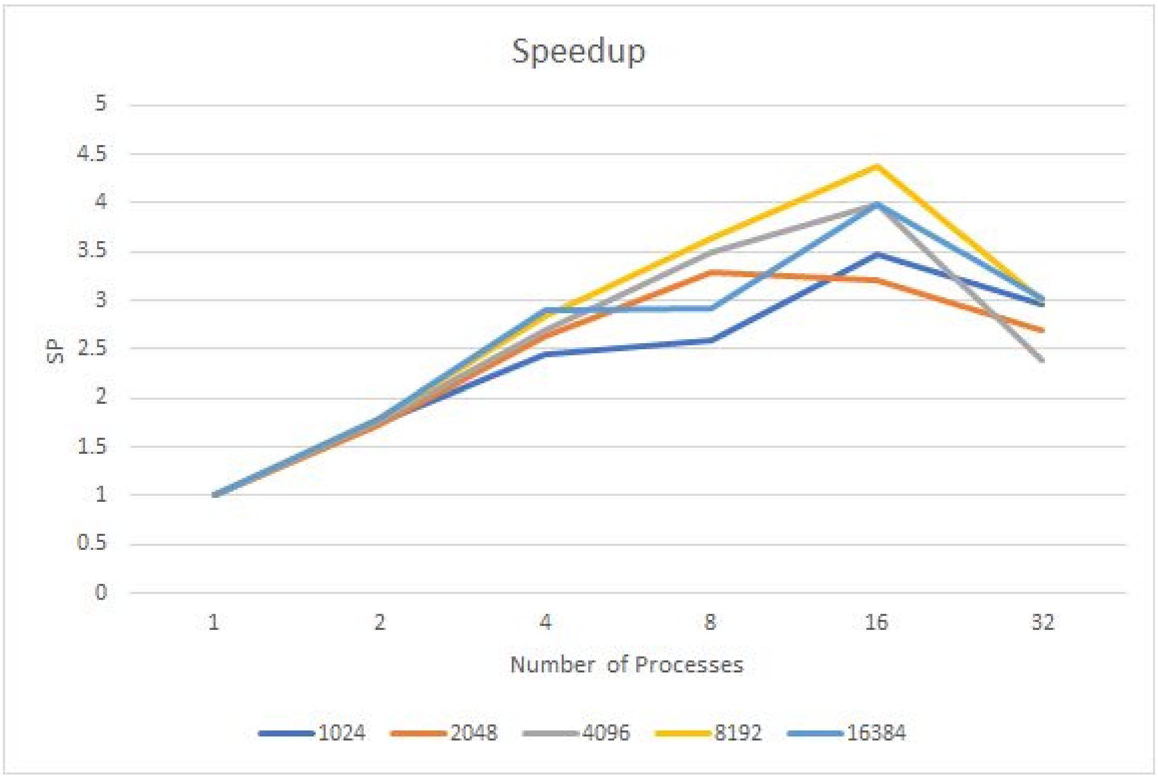
EXECUTION TIME

Execution time goes down with the more processors that are being utilized. The exception is when the number of processors is > 16. This is due to each node in the cluster that I used has 16 possible cores. When there are more than 16 processors the overhead of communication between nodes slows then execution down.



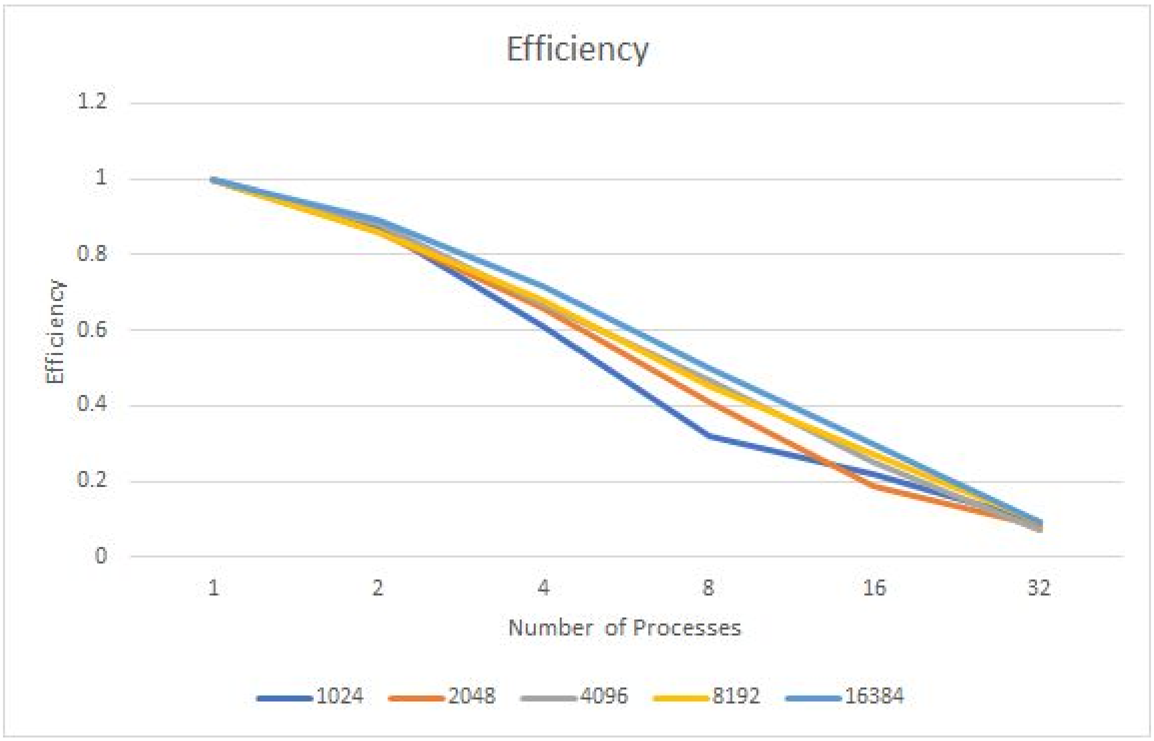
SPEED-UP

Speedup is greatest when using the max amount of processes in a single node. In this case speedup is greatest using 16 processes



EFFICIENCY

Efficiency decreases in these examples due to the relativly small size of the problem. In this case problem size never goes beyond 16384. Efficiency decreases due to each process spending more time waiting than computing. If the problem size was bigger then efficiency would increase due to each process spening more time computing than waiting.



APPLICATIONS

Fast large-integer and polynomial multiplication

- Efficient matrix-vector multiplication for Toeplitz, circulant and other structured matrices
- Filtering algorithms (see overlap-add and overlap-save methods)
- Fast algorithms for discrete cosine or sine transforms (e.g. fast DCT used for JPEG and MPEG/MP3 encoding and decoding.