

## Introduction

The Fast Transform (FFT) is an essential algorithm for signal processing, numerical analysis, and engineering applications.

This project compares execution times of different FFT implementations:

- Sequential (CPU)
- Parallel OpenMP (CPU)
- CUDA GPU

This study aims to analyze the performance differences between CPU multithreading and GPU acceleration across various FFT sizes.

## Results & Analysis

Average execution times from 5 test runs

FFT Size	Sequential (CPU)	Parallel OpenMP (CPU)	CUDA GPU
4096	0.000688	0.151379	0.339362
65536	0.010447	0.196264	0.000996
1048576	0.229312	0.382001	0.022231
16777216	4.534188	0.801931	0.058288

Speedup Average Trends


FFT SIZE	Speedup (OpenMP vs. Sequential)	Speedup (CUDA vs. Sequential)	Speedup (CUDA vs. OpenMP)
4096	~4.5x slower	~2x slower	~2.2x slower
65536	~1.9x slower	~10.5x faster	~197x faster
1048576	~1.6x faster	~17.2x faster	~6.3x faster
16777216	~5.6x faster	~77.8x faster	~13.7x faster

## Hardware Setup

 CPU: AMD EPYC 7453 (112 Cores)

 GPU: NVIDIA A100-SXM4-80GB

 RAM: 2.0 TiB

 Software: Ubuntu 22.04, CUDA 12.4, GCC 12.3, OpenMP 4.5

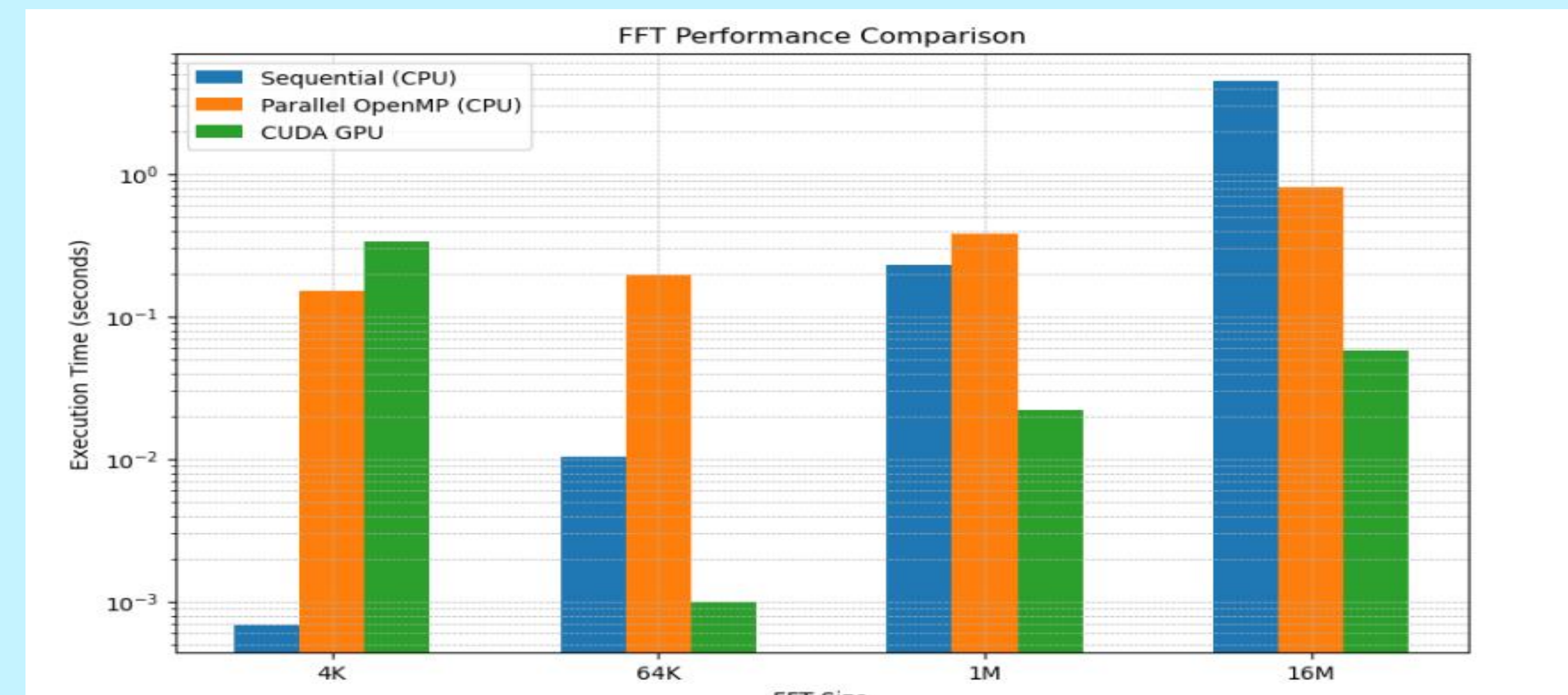
## Methods

Three FFT implementations were tested:

- Sequential FFT (Baseline CPU)
- Parallel FFT (OpenMP Multi-threading)
- CUDA GPU FFT (High-Performance Acceleration)

Tested on different FFT sizes: 4K, 64K, 1M, 16M.

Measured execution time and calculated speedups.



### Key Observations:

- For small FFTs (4K, 64K): OpenMP is faster than GPU due to CUDA overhead.
- For large FFTs (1M, 16M): GPU significantly outperforms CPU methods.
- Speedup: CUDA reaches ~100x acceleration for large problem sizes.

## Conclusion & Future Work

- CUDA is best for large FFT sizes, while OpenMP is efficient for mid-range sizes.
- Explore hybrid CPU-GPU execution for better performance.