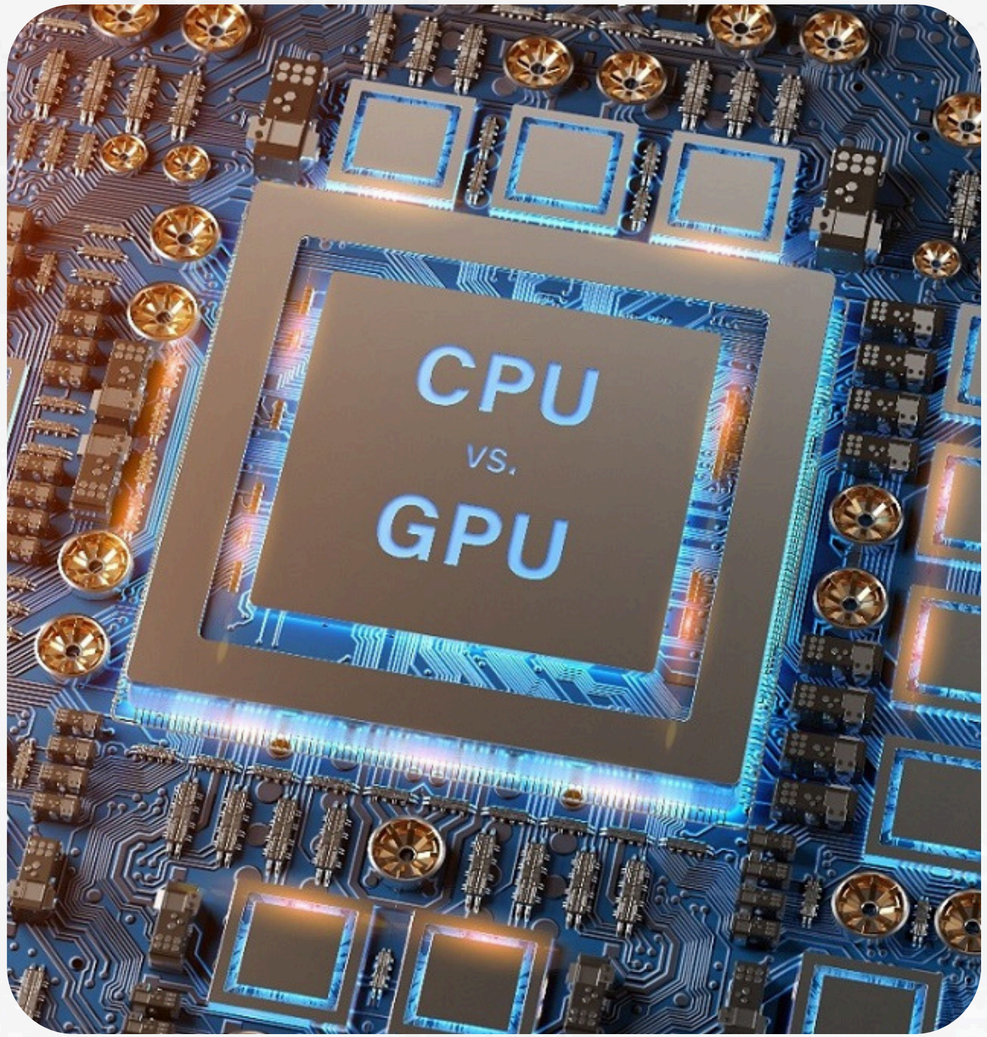
Accelerating Image Convolution: A CPU vs. GPU Performance Analysis

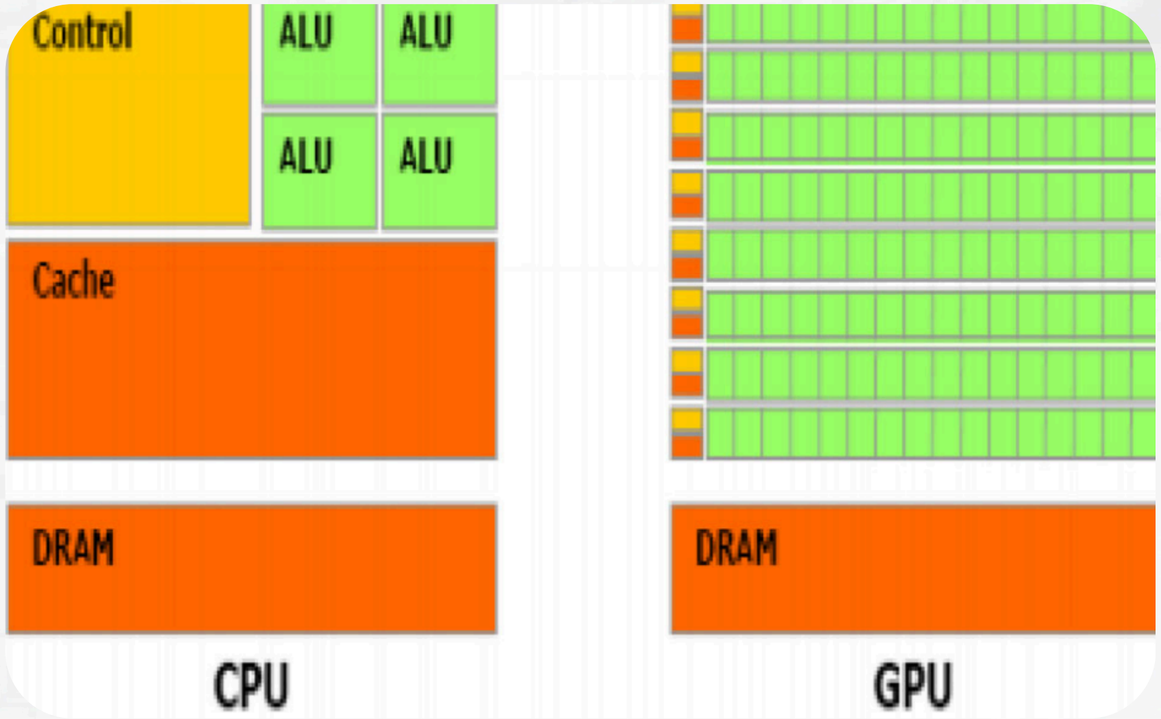
This presentation introduces research as a systematic process of inquiry and exploration. It outlines fundamental principles, diverse methodologies, and innovative practices that collectively advance academic investigation.

Presented by Faiyaz Bin Yousuf
ID: 22101845



Contents

Motivation & Background



Clarity

Image processing relies on computationally complex and memory-intensive operations like 2D convolution

Bottleneck

With the rise of high-resolution images (4K, 8K), traditional single-threaded CPU performance has become a significant bottleneck

Insufficiency

Sequential processing is often insufficient for the performance demands of modern applications, which drives the need for parallel computing solutions.

Solution

Graphics Processing Units (GPUs), with their massively parallel architecture, have emerged as a powerful solution for accelerating these demanding tasks.

Research Question & Objectives

Main Question:

How do different parallel computing architectures compare in performance and scalability when executing an intensive image convolution task?

Objective - 1

To implement a Gaussian blur filter using the modern Python and Numba toolchain to leverage a high-level language for GPU computing.

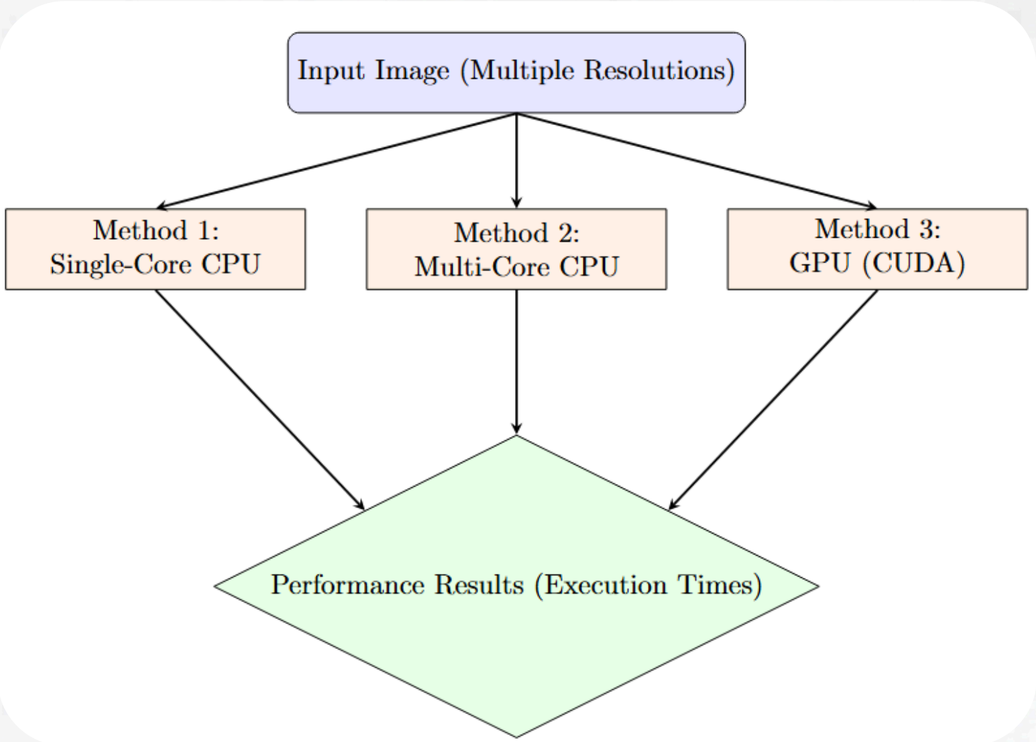
Objective - 2

To benchmark the implementation across three distinct architectures: a sequential Single-Core CPU, a parallel Multi-Core CPU, and a Graphics Processing Unit (GPU).

Objective - 3

To analyze the scalability of each architecture by measuring performance across a range of image resolutions, as the complexity of image processing increases with image size.

Methodology



01

Hardware:

- CPU: AMD Ryzen 7 3700X (8 Cores, 16 Threads)
- GPU: NVIDIA GeForce RTX 3070 (5,888 CUDA Cores)

02

Framework:

- Language: Python 3.11
- GPU Acceleration: Numba library for Just-In-Time (JIT) compilation of CUDA kernels

03

Architectures Compared:

- Sequential Single-Core CPU
- Parallel Multi-Core CPU (Python multiprocessing)
- Massively Parallel GPU (Python Numba/CUDA)

04

Benchmark:

- A 5x5 Gaussian Blur filter was tested across four image resolutions (SD, HD, FHD, QHD) to analyze scalability

Results - Execution Times

Performance was measured for a 5x5 Gaussian Blur on a Quad HD (2560x1440) image

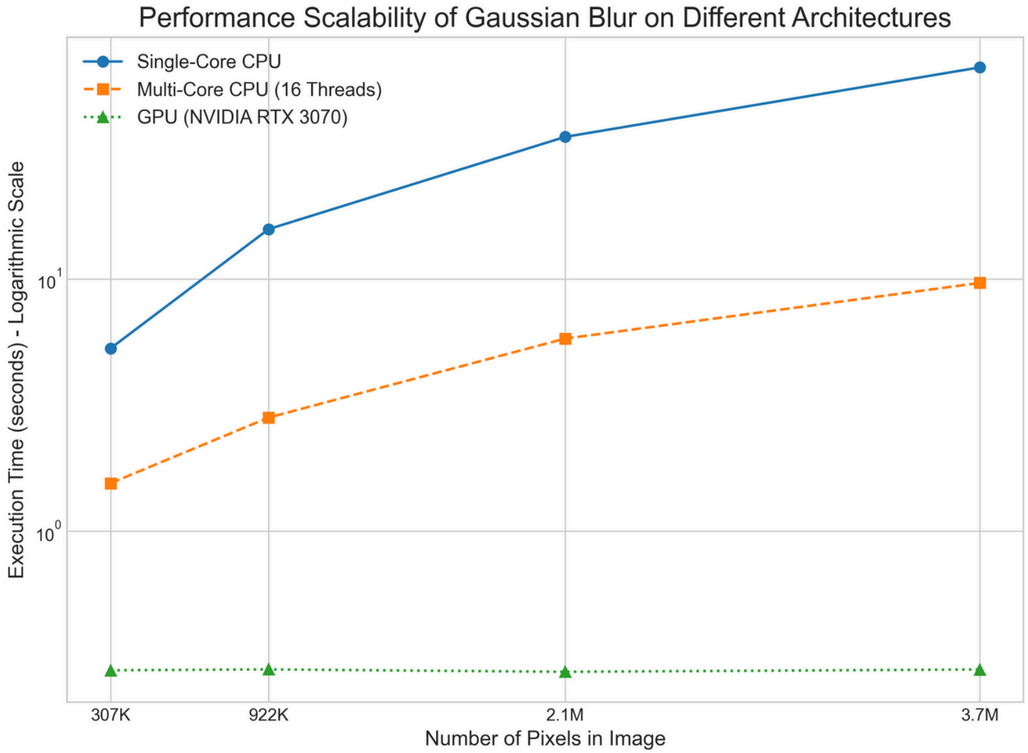
Architecture	Execution Time (seconds)
Single-Core CPU	68.92 s
Multi-Core CPU (16 Threads)	9.64 s
GPU (NVIDIA RTX 3070)	0.28 s

Key Findings:

- The parallel Multi-Core CPU was ~5.4x faster than the Single-Core CPU.
- The GPU was ~24x faster than the fully utilized Multi-Core CPU.
- The GPU achieved a massive
- ~130x speedup over the sequential Single-Core CPU, demonstrating the significant performance gains of parallel architectures for image convolution.

Results - Performance Scalability

Performance was measured for a 5x5 Gaussian Blur on a Quad HD (2560x1440) image



Key Findings:

- Execution times for both single-core and multi-core CPU implementations increase significantly with the number of pixels.
- In contrast, the GPU's execution time remains nearly flat and consistently low across all tested resolutions, appearing almost zero in comparison to the CPU processing time.
- This trend demonstrates the superior scalability of the GPU's data-parallel architecture, as the performance benefits become even more pronounced as the image size increases

Discussion & Analysis

01

The Architectural Advantage:

The GPU's massive performance advantage stems from its specialized architecture for parallel processing. CPUs use a few powerful cores optimized for complex, sequential tasks, while GPUs use thousands of simpler cores designed for data parallelism—executing the same operation on many data points simultaneously

02

Interpreting the Speedups:

The ~24x speedup over a fully utilized 16-thread CPU highlights that even a parallel general-purpose processor cannot match the GPU's specialized throughput for this task. The CPU's sub-linear scaling (~5.4x speedup) is expected due to overheads from process management and data handling.

03

Performance Context:

The GPU's flat performance curve suggests the task is limited by fixed overheads (like kernel launch time), not the computation itself. Furthermore, while Python/Numba offers great convenience, its performance may be lower than native C-CUDA due to interpreter overhead and the Just-In-Time (JIT) compilation process.

Conclusion and Future Work

Conclusion

In conclusion, this project successfully benchmarked a Gaussian blur filter across single-core, multi-core, and GPU architectures using Python and Numba. The findings were definitive: the GPU demonstrated a profound performance advantage, achieving a maximum speedup of approximately 130x over the single-core CPU and 24x over the fully-utilized multi-core CPU. This study confirms that for data-parallel tasks like image convolution, the GPU's specialized architecture is the most critical factor for achieving high performance

Future Work

Looking ahead, several avenues for future research exist. Future work could involve implementing advanced GPU kernel optimizations, such as using shared memory, to further reduce memory latency and increase performance. Additionally, a direct comparison between this Python/Numba implementation and a traditional C++/CUDA version would provide valuable quantitative data on the trade-offs between development ease and raw performance, building on the findings of Oden (2020).

A special thanks to Annajit Alim Rasel sir

FEEL FREE TO ASK ANY QUESTIONS