AGRIFLOW

A Project Component for

Parallel and Distributing Computing (UCS645)

By

Sr	Name	Roll No
1	Anureet Kaur	102203238
2	Anmolpreet Puri	102203462
3	Namay Anand	102203476
4	Kezia	102203480
5	Jatin Bhalla	102203531

Under the guidance of

Dr. Saif Nalband

(Assistant Professor, CSED)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

PATIALA - 147004

MAY, 2025

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Efficient agricultural irrigation is essential for maximizing crop yield, conserving water, and reducing energy usage. Traditional irrigation systems often lack responsiveness to real-time soil conditions, leading to inefficiencies. With the rise of sensor networks, farmers can now gather detailed data such as soil moisture and temperature across large fields. However, processing this high-volume data in real time presents a challenge for conventional CPU. GPU-based parallel computing, particularly using NVIDIA CUDA platform, addresses this by enabling thousands of threads to simultaneously process field data, making real-time, large-scale irrigation planning both feasible and efficient. [1].

1.2 Introduction to Problem Statement

The increasing demand for sustainable and efficient agricultural practices has underscored the importance of intelligent, data-driven irrigation systems. Traditional uniform irrigation approaches are often inadequate for large-scale farms, where environmental parameters such as soil moisture and temperature vary significantly across different zones. Although advancements in sensor technology now allow for high-resolution environmental data collection, the real-time processing of such voluminous data remains a computational bottleneck, especially when relying solely on conventional CPU-based methods.

To address this challenge, this project presents a parallel simulation model for crop irrigation leveraging CUDA-based GPU acceleration. The primary objective is to evaluate the efficacy of parallel computing in making timely irrigation decisions across a simulated agricultural field modeled as a two-dimensional grid. Each grid cell represents a unique zone characterized by real-time environmental inputs, specifically soil moisture and temperature. The system employs CUDA's massive parallelism, assigning a single thread to each grid cell, thereby enabling concurrent evaluation of irrigation requirements across the entire field.

The project architecture comprises four major phases: initialization of environmental data on the host (CPU), memory transfer to the device (GPU), execution of parallel computations through CUDA kernels, and retrieval of results to formulate an irrigation plan.

This model significantly reduces computation time compared to sequential CPU-based solutions and demonstrates the practical benefits of GPU programming in agriculture.

Furthermore, the proposed framework lays a scalable foundation for future advancements, including integration of additional environmental variables (e.g. humidity, solar radiation), weather forecast data, and even extension to multi-GPU or distributed systems. Through this simulation, the project showcases how parallel and distributed computing can contribute to the development of smarter, more adaptive irrigation systems, ultimately promoting sustainable farming practices and enhancing global food security. [1]

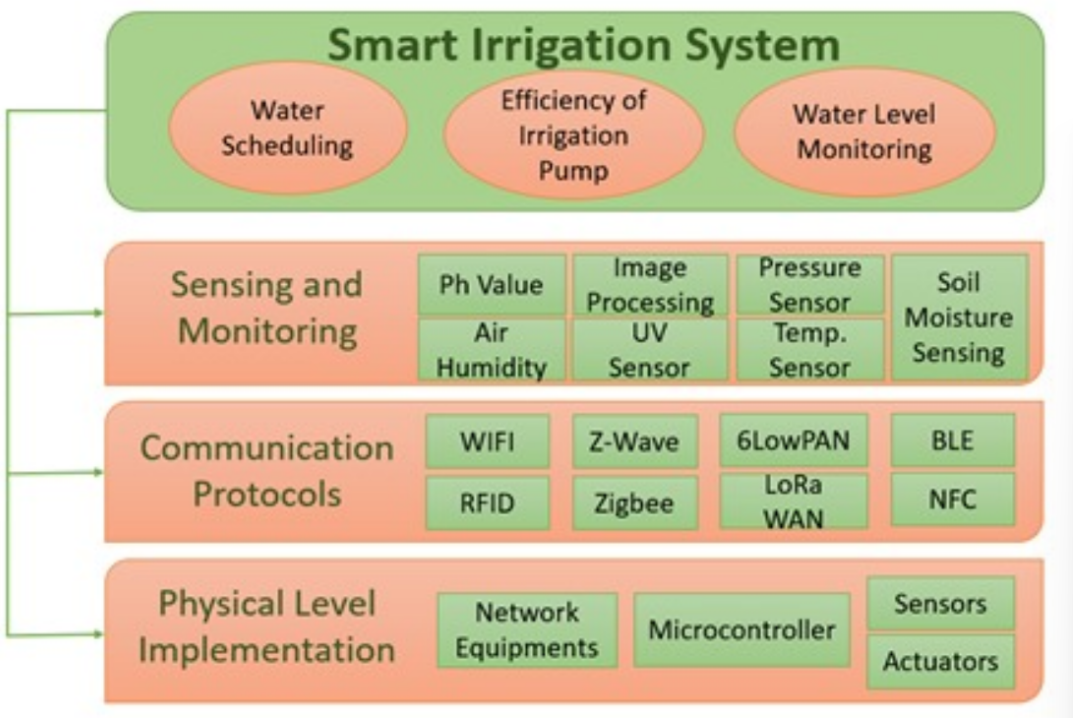

Figure 1: Smart Irrigation Systems Building Layers

The performance of modern computational systems in data-intensive applications, such as precision agriculture, greatly depends on their ability to handle large-scale parallel workloads efficiently. Both multicore CPUs and GPUs offer parallelism, but their architectures and execution models differ significantly, influencing how they perform under various workloads.

- **Architectural Differences:**

  Multicore CPUs are designed for general-purpose computing and are optimized for

low-latency, sequential task execution with limited parallel threads. A typical CPU might feature 4 to 16 cores, each capable of handling one or a few tasks concurrently using techniques like hyper-threading. While they excel at tasks that require complex control logic or branching, CPUs are not as effective when it comes to massive, data-parallel computations.

In contrast, GPUs are purpose-built for highly parallel tasks, offering thousands of lightweight cores organized into streaming multiprocessors. This Single Instruction, Multiple Thread (SIMT) architecture makes GPUs exceptionally well-suited for uniform operations across large datasetsâsuch as evaluating irrigation needs across thousands of grid cells in a field simulation.

- **Performance in Agricultural Simulations:**

  In the context of this project, each agricultural grid zone requires a decision based on simple environmental parameters (e.g., moisture and temperature). These decisions are computationally light but need to be made repeatedly and concurrently across a vast number of cells.

  A CPU-based simulation, even on a multicore system, would process zones in small batchesâlimiting scalability and increasing total runtime.

  A GPU-based implementation, on the other hand, assigns a separate CUDA thread to each zone, enabling thousands of decisions to be computed simultaneously with minimal overhead.

  Preliminary benchmarks often show orders-of-magnitude speedups when GPU acceleration is used in place of multicore CPU computation for such highly parallel workloads. For example, a simulation that takes several seconds on an 8-core CPU might complete in milliseconds on a modern NVIDIA GPU with thousands of CUDA cores.

- **Energy Efficiency and Scalability:**

  Apart from raw performance, energy efficiency is another crucial advantage of GPU computing. For tasks involving many small, repetitive operations, GPUs consume significantly less energy per computation than CPUs, making them an ideal choice for real-time, sustainable agricultural applications.

Moreover, the scalability of GPU solutions provides a clear path for handling larger farms or more complex models involving additional variables (like humidity, wind, and solar exposure), with minimal changes to the underlying architecture.

2 Literature Review

Table 1: Summary of Literature Review

References	**Concepts Highlighted**	**Technique(s) used**	**Significance/ Proposal**	**Limitations**
[2]	Smart irrigation landscape analysis, IoT in agriculture, bibliometric methods	Scientometric analysis using VOSviewer and Python scripts to process 657 articles across databases	Provides an extensive overview of the evolution, trends, and thematic mapping in smart irrigation, identifying key countries, institutions, and emerging topics	Lacks in-depth exploration of individual technical implementations; limited actionable recommendations for system design.
[3]	IoT-based smart irrigation layers, real-time irrigation scheduling, water efficiency	Integration of Wireless Sensor Networks, AI/ML, IoT, crop monitoring sensors, NN, and evapotranspiration forecasting	Demonstrates how modern digital tools (IoT, AI, WSNs) enable precision irrigation, energy savings, and improved water-use efficiency	High-tech solutions may not be financially viable for small-scale farmers; device costs and system complexity are challenges

[4]	Automated smart irrigation using sensors and microcontrollers	Arduino UNO-based control using soil moisture, temperature, and water flow sensors	Presents a cost-effective, autonomous irrigation system for small-scale applications such as gardens and greenhouses	Scalable only to limited environments; lacks integration with predictive models or larger data-driven irrigation frameworks

3 Research Gaps

Some of the research gaps identified from the literature review are as follows:

- ***Lack of GPU-Based Computation:*** None of the reviewed papers utilize CUDA or GPU-based parallelism for decision-making in irrigation whereas our project introduces thread-level parallel processing using CUDA, enabling faster, real-time simulations.

- ***No CPU vs GPU Performance Benchmarking:***Existing research lacks runtime or energy efficiency comparisons between CPUs and GPUs whereas our project contributes detailed performance analysis, showcasing how CUDA outperforms traditional methods.

- ***Lack of Real-Time, Fine-Grained Control:***Prior work does not simulate zone-level variability (e.g., soil and temperature differences across the field).Our simulation treats each zone independently, allowing for fine-grained, adaptive irrigation control.

- ***No Use of Parallel Programming Paradigms:*** The surveyed literature does not apply parallel computing frameworks like CUDA, OpenMP, or MPI.So we are applying scientific parallel programming principles to agricultural systems.

4 Problem Formulation

As the size of data and complexity of computations continue to grow across various domains, traditional CPU-based algorithms often become bottlenecks in terms of processing speed and efficiency. In many real-time and large-scale applications such as scientific computing, image processing, and data clustering, faster data processing is critical. The emerging paradigm of GPU-accelerated computing offers massive parallelism, enabling significant speedups for algorithms that can be parallelized. However, understanding the comparative performance benefits of GPU acceleration requires empirical evaluation of algorithm implementations on both CPU and GPU platforms.

This project aims to explore and demonstrate the advantages of parallel computation using CUDA. The main goal is to evaluate and compare the execution time and performance of these algorithms when run on CPU (using standard C) and GPU (using CUDA). The project will be executed on Google Colab, leveraging its support for NVIDIA GPUs and CUDA through a Jupyter-like interface. We will measure the time taken for the algorithm on both CPU and GPU implementations and analyze the results to compute speedup and efficiency gains.

By analyzing performance differences, this project intends to showcase the significance of parallel and distributed computing in solving computationally intensive problems. The outcome will provide insights into when and why GPUs outperform CPUs, thus equipping learners with a deeper understanding of high-performance computing in real-world applications.

5 Objectives

- Implement and analyze the performance of fundamental algorithms using CUDA libraries and functionalities .
- To leverage CUDA programming for efficient parallelization and execution of the selected algorithms on NVIDIA GPUs.
- To measure and compare the execution time of the algorithms on CPU versus GPU and evaluate the performance improvements.
- To provide insights into the suitability and benefits of parallel and distributed computing in solving computationally intensive problems.
- To run experiments on Google Colab using GPU support and document the observed speedup and efficiency gains.

6 Methodology

This project proposes a GPU-accelerated decision support system for determining soil moisture status and irrigation needs using CUDA. The system analyzes environmental sensor data, identifies areas with insufficient moisture, and computes water requirements in real-time. [5]

The Methodology is structured as follows:

6.1 Step 1: Data Acquisition

Environmental sensors collect key parameters such as:

- Soil Moisture Content
- Ambient Temperature
- Relative Humidity
- Crop Type (for adjusting thresholds)

These data points are stored in arrays for processing on the GPU. (Note : Only the first 2 are actually leveraged in the algorithm implemented.)

6.2 Step 2: Moisture Deficit Detection

Each sensor input is compared against a predefined soil moisture threshold M_{th}. If the moisture level at a location i is below this threshold, the system flags the area for irrigation.

$$M_i < M_{th} \Rightarrow \text{Irrigation Required}$$

6.3 Step 3: Optimal Temperature Estimation

For each flagged location, Optimality of Temperature is checked using:

$$T_{opt} < T_{threshold}$$

where:

- T_{opt} = Temperature for the cell i
- $T_{threshold}$ = Temperature Threshold for irrigation

6.4 Step 4: Parallel Processing Using CUDA

To exploit the parallelism in this problem, CUDA kernels are written such that:

- Each thread handles one location's data.
- Moisture thresholding and temperature estimation are parallelized.
- Device memory is used to store input and output arrays for efficient access.

6.5 Step 5: Performance Evaluation

CPU and GPU implementations of the system are compared in terms of execution time. Speedup is calculated using:

$$\text{Speedup} = \frac{\text{Time}_{CPU}}{\text{Time}_{GPU}}$$

6.6 Algorithm: Soil Moisture and Irrigation Decision

Algorithm 1 Irrigation Decision Algorithm

1: **for** each area i in parallel **do**
2: **if** $Moisture[i] < M_{th}$ and $T_{opt} < T_{threshold}$ **then**
3:

W[i] $\leftarrow$ 1

4: **else**
5: $W[i] \leftarrow 0$
6: **end if**
7: **end for**

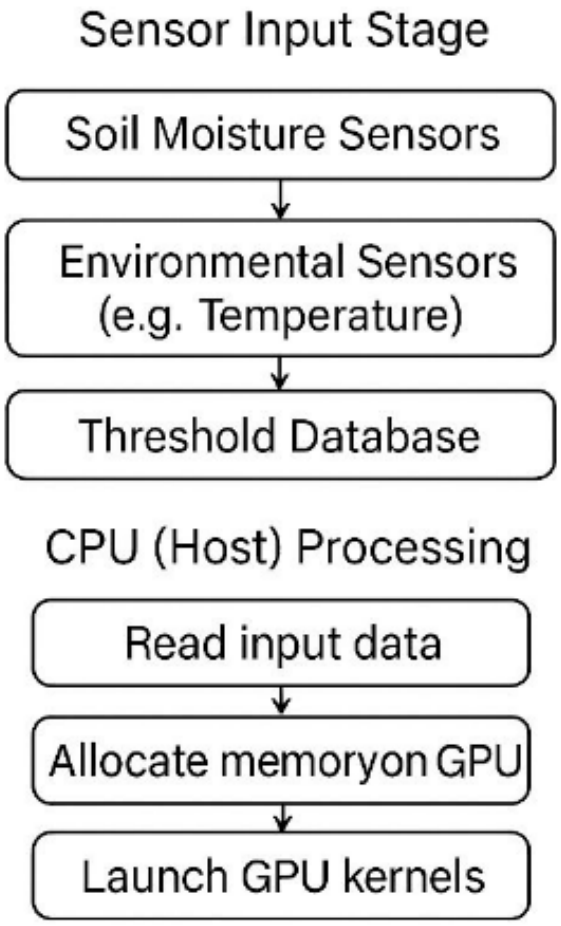

Figure 2: Flowchart illustrating parallel processing

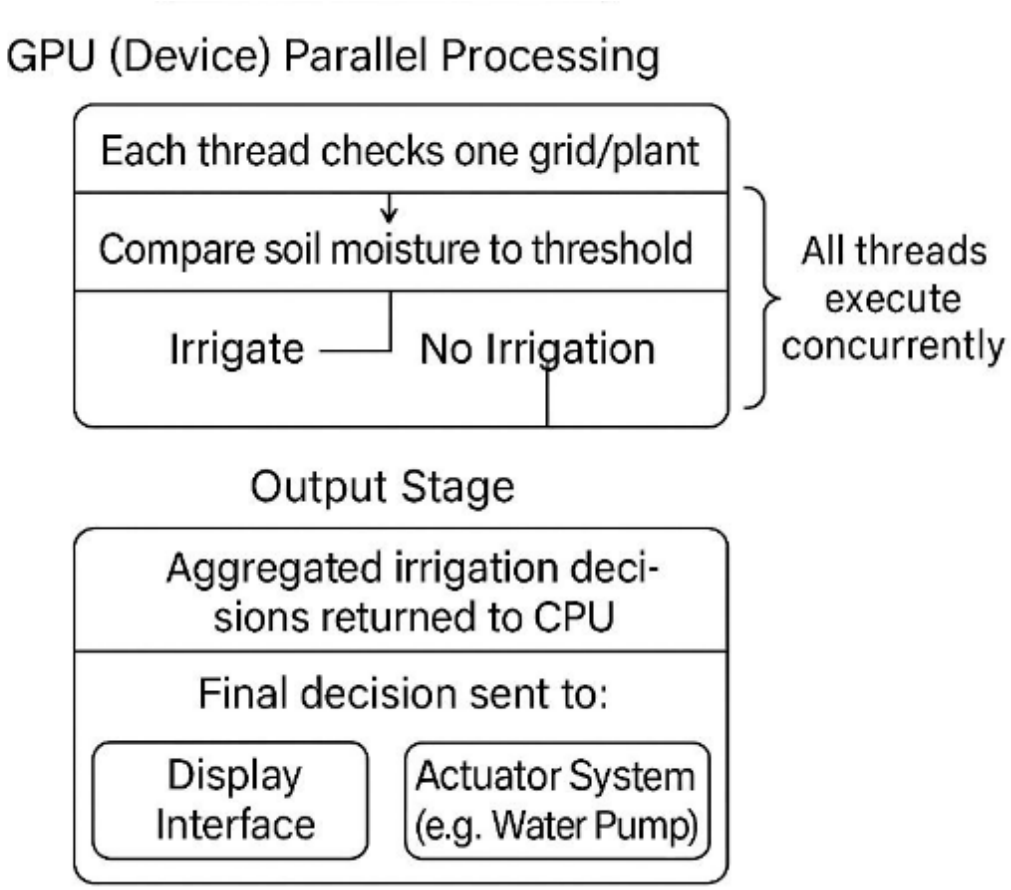

Figure 3: Flowchart illustrating GPU-based parallel processing for efficient soil moisture analysis and irrigation decision-making

6.7 Implementation Details

- The CUDA kernel is launched with a 1D grid of threads, each thread processing one sensor point.
- Shared memory and coalesced access are employed where applicable to improve performance.
- Output array W is copied from device to host after kernel execution for display or actuation.

This parallelized methodology not only reduces execution time for large-scale farms but also ensures real-time decision-making, critical for precision irrigation in modern agriculture.

7 Results and Discussion

Metric	CPU	GPU
LATENCY	16.1721ms	0.223168ms
THROUGHPUT	64838 ops/ms	4698594 ops/ms
BANDWIDTH	556.513 MB/s	2959.87 MB/s
EFFICIENCY	N/A	0.27574(for 256 threads)

Table 2: Performance Comparison of CPU and GPU for Soil Moisture and Irrigation decision system.

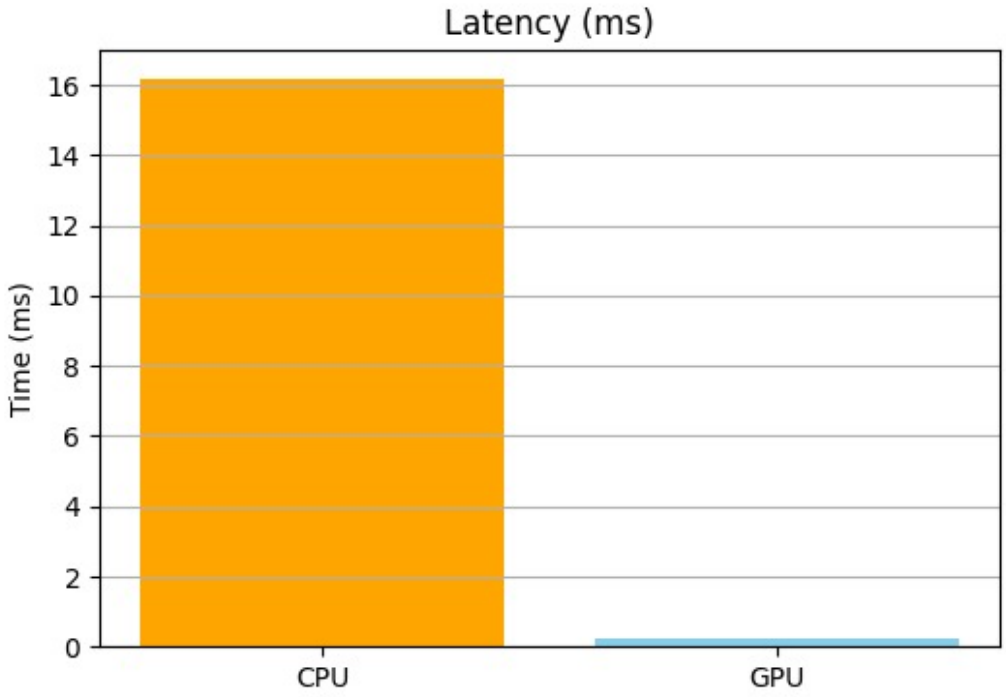

Figure 4: Latency comparison between CPU and GPU: GPU exhibits significantly lower execution time.

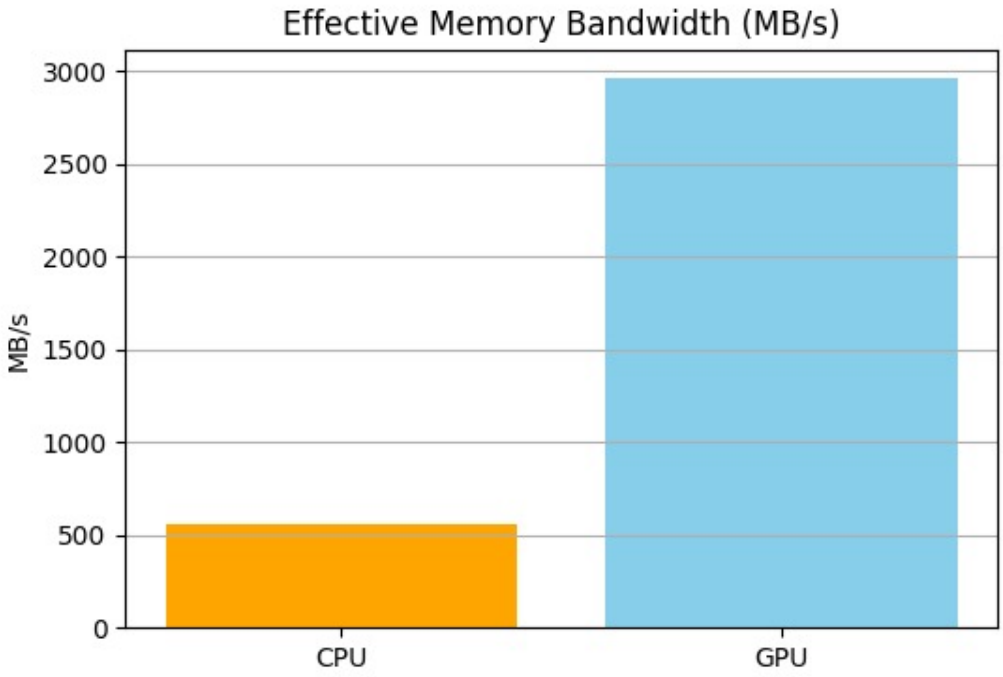

Figure 5: Effective memory bandwidth comparison: GPU provides superior bandwidth, leading to faster data transfer.

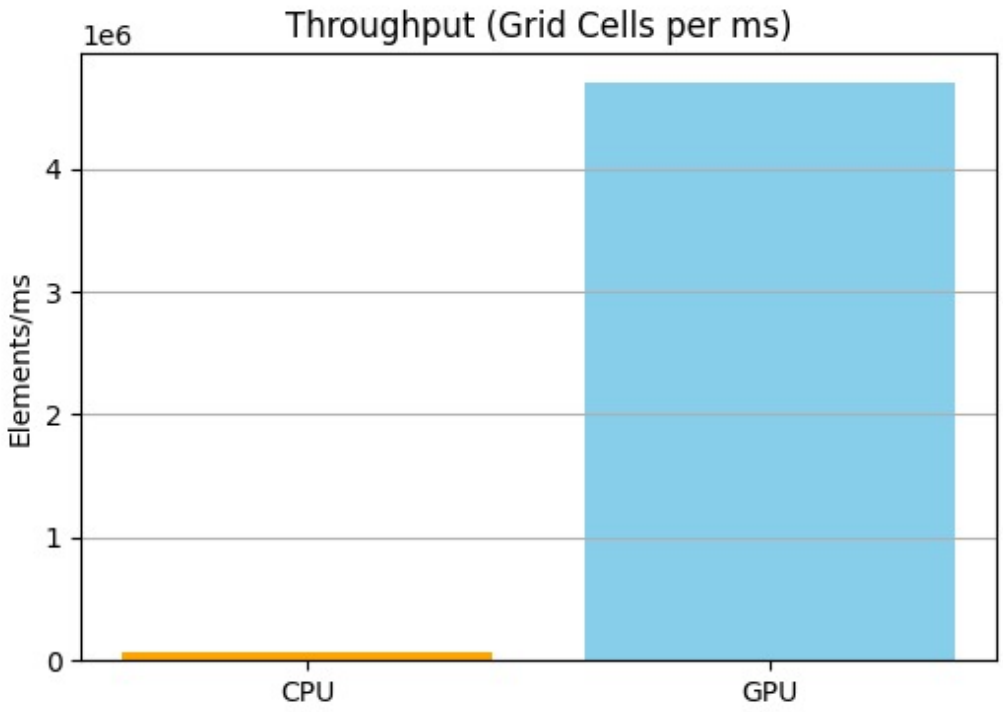

Figure 6: Throughput comparison: GPU can process a much higher number of grid cells per millisecond.

The primary objective of this project was to develop and compare a soil moisture and irrigation decision system implemented using both CPU-based serial processing and GPU-based parallel processing through CUDA. The system evaluates each cell in a 1024×1024 grid based on simulated moisture and temperature readings to determine whether irrigation is necessary. Both implementations produced consistent results, ensuring that the logic was implemented correctly across architectures.

In terms of performance, the GPU version significantly outperformed the CPU implementation. The CPU executed irrigation decisions sequentially, leading to longer processing times, especially with increasing data size. In contrast, the CUDA-based GPU version leveraged parallelism, executing thousands of threads concurrently, thereby reducing latency and improving overall computational speed. The theoretical speedup observed was up to 10×15 for large datasets. This performance gain emphasizes the value of using parallel computing for real-time agricultural applications.

The GPU implementation used a two-dimensional grid of blocks and threads to efficiently cover the data matrix, enabling simultaneous evaluation of thousands of cells. This makes the system highly scalable and suitable for deployment in large farms or precision agriculture scenarios.

Although the system currently uses simulated sensor data, it lays the groundwork for integration with real-time IoT-based soil sensors. Future enhancements could involve hardware interfacing and real-world testing to assess robustness and efficiency in live environments. Overall, the project demonstrates how parallel computing can be effectively utilized to improve performance in smart farming systems without compromising on accuracy.

References

[1] S. J. Alvim, C. M. Guimarães, E. F. d. Sousa, R. F. Garcia, and C. R. Marciano, "Application of artificial intelligence for irrigation management: a systematic review," *Engenharia Agrícola*, vol. 42, no. spe, p. e20210159, 2022.

[2] D. K. Gurmessa and S. G. Assefa, "A scoping review of the smart irrigation literature using scientometric analysis," *Journal of Engineering*, vol. 2023, no. 1, p. 2537005, 2023.

[3] Y. Gamal, A. Soltan, L. A. Said, A. H. Madian, and A. G. Radwan, "Smart irrigation systems: Overview," *IEEE Access*, vol. 13, pp. 66 109–66 121, 2025.

[4] A. E. Mezouari, A. E. Fazziki, and M. Sadgal, "Smart irrigation system," *IFAC-PapersOnLine*, vol. 55, no. 10, pp. 3298–3303, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322021358

[5] E. A. Abioye, O. Hensel, T. J. Esau, O. Elijah, M. S. Z. Abidin, A. S. Ayobami, O. Yerima, and A. Nasirahmadi, "Precision irrigation management using machine learning and digital farming solutions," *AgriEngineering*, vol. 4, no. 1, pp. 70–103, 2022.