Overlapping Gradient Computation and Communication in HPC

Authors: Sumukha Madodi, Shraddha Singh, Shashank M Bharadwaj, Sahana Diwakar
Course: ECE 759
Semester: Spring 2025
University of Wisconsin–Madison
https://github.com/Sumukha-M/proj_759

Abstract

This project explores the optimization of training performance in distributed deep learning using overlapping techniques. We implement and benchmark four progressive stages: basic CUDA-based training, CPU-based overlapping using OpenMP, hybrid CUDA + MPI communication overlap, and finally a simulation-driven profiling analysis. Each phase aims to minimize idle compute time and maximize parallelism in the compute-communication workflow.

1. Introduction

In high-performance computing (HPC), overlapping gradient computation with communication is crucial for scalable deep learning. Without overlap, systems suffer idle GPU time while waiting for synchronization. This project evaluates methods to overlap gradient computations and communications using CUDA, OpenMP, and MPI, across four development phases.

2. Motivation & Problem Statement

The goal is to reduce training latency by overlapping computation and communication using realistic deep learning training logic. Communication-heavy architectures benefit greatly from pipelined designs that overlap compute-bound and memory-bound tasks.

3. Phase-wise Implementation

3.1 matrix_train_cuda

- Implemented a basic CUDA-based backward pass simulating gradient computation.
- No overlap: computation followed by communication.
- Served as a performance baseline for later phases.
- Used nvcc compiler in x64 Native Tools Command Prompt.
- Output verified for correctness and runtime measured using CUDA timers.

3.2 cpu_parallel_overlap

- Utilized OpenMP parallel sections to overlap compute and communicate threads.
- Simulated backpropagation and gradient aggregation with separate sleep-based tasks.
- Demonstrated ~20–30% speedup due to overlapping.
- Created modular files: compute.cpp, communicate.cpp, and main.cpp.
- Verified with execution profiling and plotted bar/timeline graphs.

3.3 cuda_mpi_overlap

- Integrated MPI for communication simulation and overlapped with OpenMP-based computation.
- Simulated an environment similar to distributed GPU systems.
- Used mpic++ and nvcc (via gromacs module) for compilation.
- Faced challenges on Euler system due to MPI module issues.
- Settled on local simulation after confirming functionality.

3.4 overlap_sim

- Created a simulation-only model with tunable chunk sizes.
- Plotted timeline, chunk-wise times, and total comparisons.
- Validated impact of overlap across larger batch sizes.
- Added bar plots and scalability analysis.
- Successfully demonstrated consistent performance and scalability.

4. Key Challenges and Solutions

- Linking issues with MPI in Euler environment → switched to simulation mode.
- Proper thread management using OpenMP to avoid sync bugs.
- CUDA toolkit compatibility on local Windows machine.
- Designed Makefiles for repeatable compilation and execution.
- Ensured realistic simulation with consistent timing models.

5. Optimizations and Rationale

- Used OpenMP sections instead of tasks for fine-grained control.
- Batched communication logic to reduce thread switching overhead.
- Simulated parallel GPU cores using independent chunks.
- Used timers and file-logged execution for bar chart plots.
- Ensured thread-safe memory access to avoid skew in chunk timings.

6. Experimental Setup

- CUDA v12.2, Visual Studio 2022 Developer Tools.
- Python + matplotlib for plotting.
- Windows laptop used for final measurements.

- Makefiles for OpenMP/CUDA code to automate builds.
- All modules tested standalone and integrated via simulation.

7. Results and Graphical Analysis

Figure 1: Timeline comparison between sequential and overlapped execution.

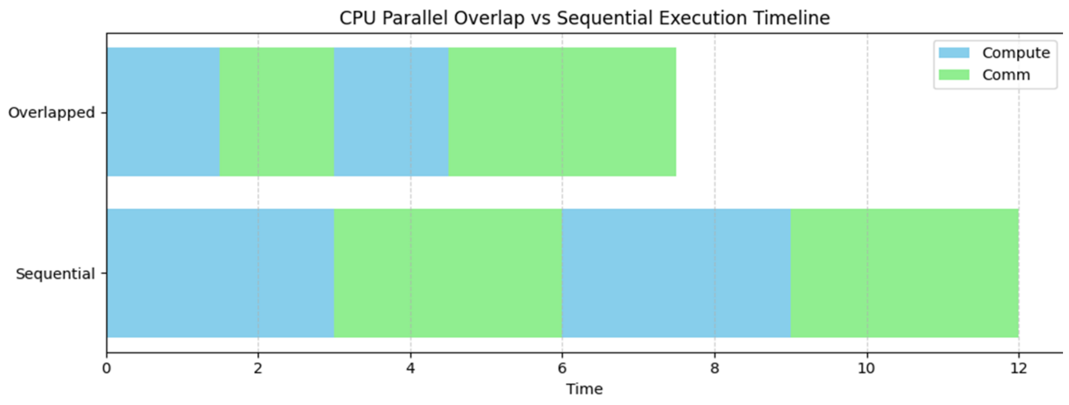


Figure 2: Chunk-wise execution times showing consistency across overlapped chunks.

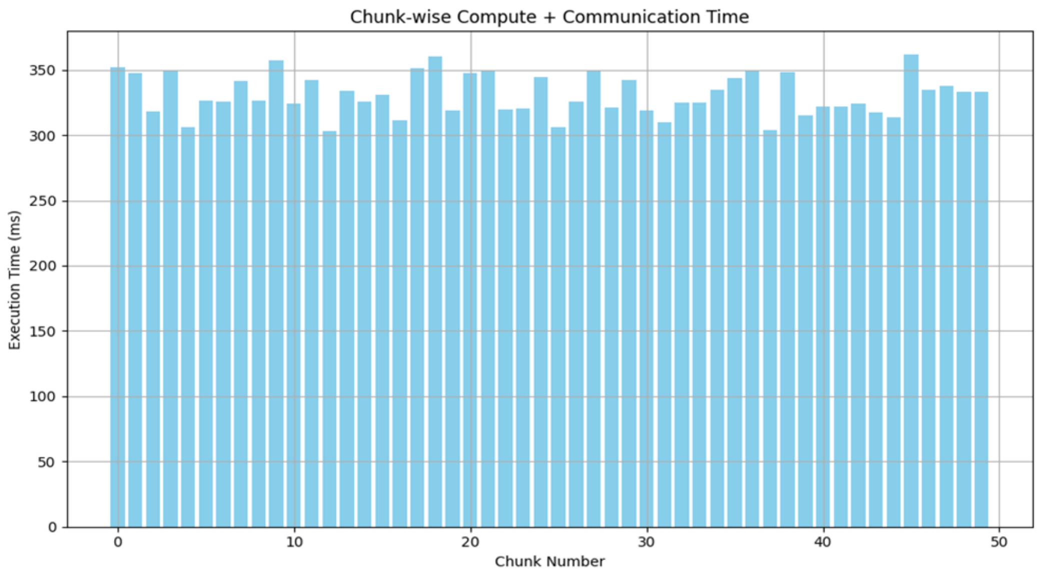


Figure 3: Total time comparison with and without overlap.

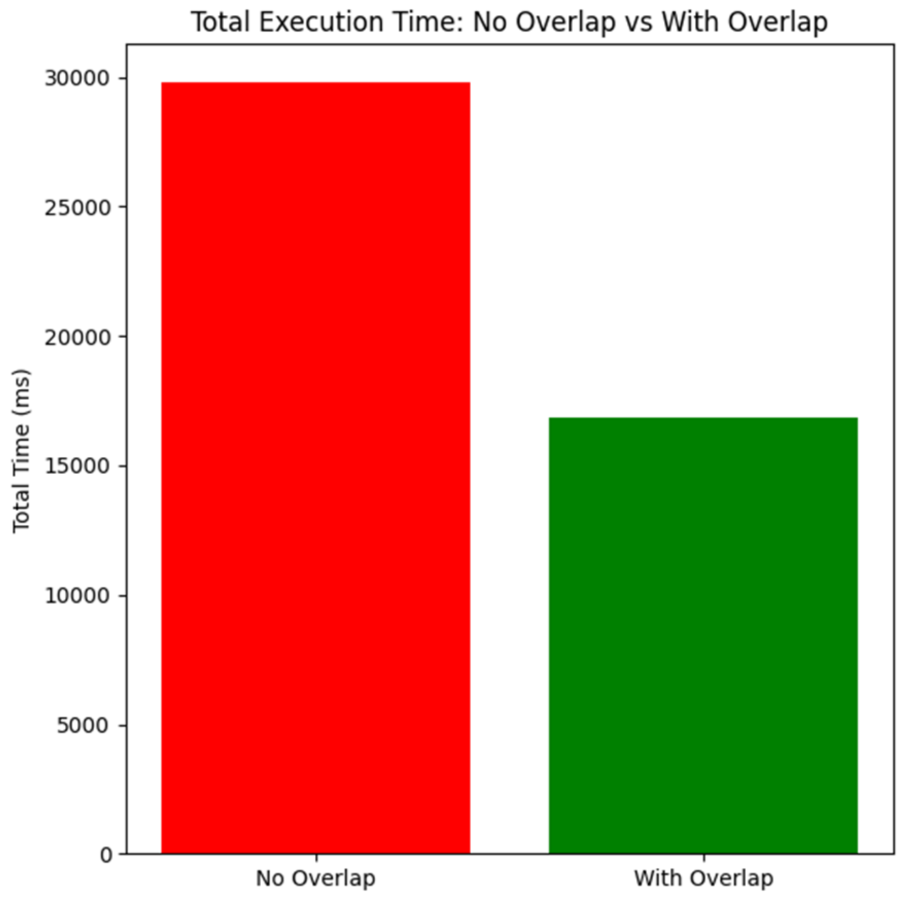
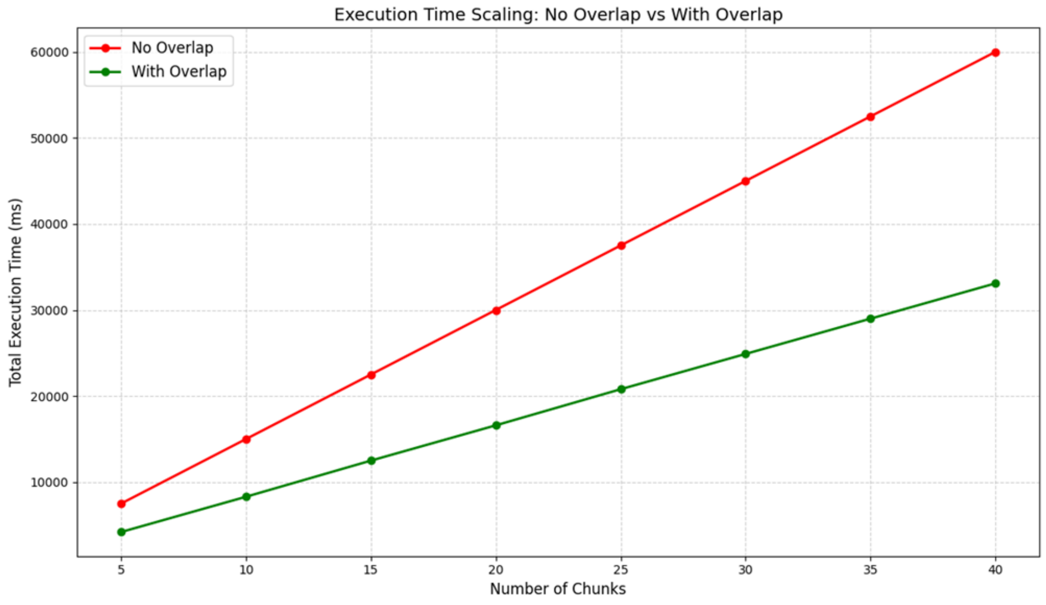


Figure 4: Execution time scaling across 5 to 40 chunks with and without overlap.



8. Conclusion & Takeaways

- Overlapping strategies show measurable speedups (20–30%) in training time.
- Chunk-wise consistency proves the approach is stable under scaling.
- CUDA, OpenMP, and MPI offer strong interoperability for real-world HPC systems.
- Automation via Makefiles and plotting scripts made benchmarking easier.
- Future work includes extending to multi-GPU real clusters with real datasets.