Parallel SSSP Update Framework for Dynamic Networks

Abstract

This project presents a scalable, platform-independent parallel implementation of the Single-Source Shortest Path (SSSP) update algorithm tailored for large-scale dynamic networks. By leveraging a hybrid parallelization approach—integrating MPI for inter-node communication, OpenMP for intra-node parallelism, OpenCL for GPU acceleration, and METIS for graph partitioning—the framework efficiently handles dynamic graph updates without necessitating full recomputation. The implementation is deployed on a Docker-based cluster environment, facilitating reproducibility and scalability.

1. Introduction

Dynamic networks, such as social graphs, communication systems, and biological networks, are characterized by frequent structural changes, including edge insertions and deletions. Traditional SSSP algorithms, like Dijkstra's, are inefficient in such settings due to the overhead of recomputing paths from scratch after each update. To address this challenge, we implement a two-phase parallel algorithm inspired by Khanda et al. (2022), which incrementally updates the SSSP tree, thereby optimizing performance in dynamic environments.

2. System Architecture

2.1 Docker-Based Cluster Deployment

The framework is deployed on a cluster of Docker containers, each simulating a compute node. This setup ensures a controlled and reproducible environment, facilitating testing and scalability.

2.2 Hybrid Parallelization Strategy

The implementation employs a hybrid parallelization approach:

- MPI (Message Passing Interface): For distributed communication
- OpenMP: For intra-node multithreading
- OpenCL: For GPU acceleration
- METIS: For graph partitioning and load balancing

3. Algorithmic Workflow

3.1 Phase 1: Identify Affected Subgraph

Processes edge insertions/deletions in parallel and flags vertices that require updates.

3.2 Phase 2: Update Affected Subgraph
Performs lock-free, asynchronous relaxations using hybrid MPI + OpenMP + OpenCL parallelism.

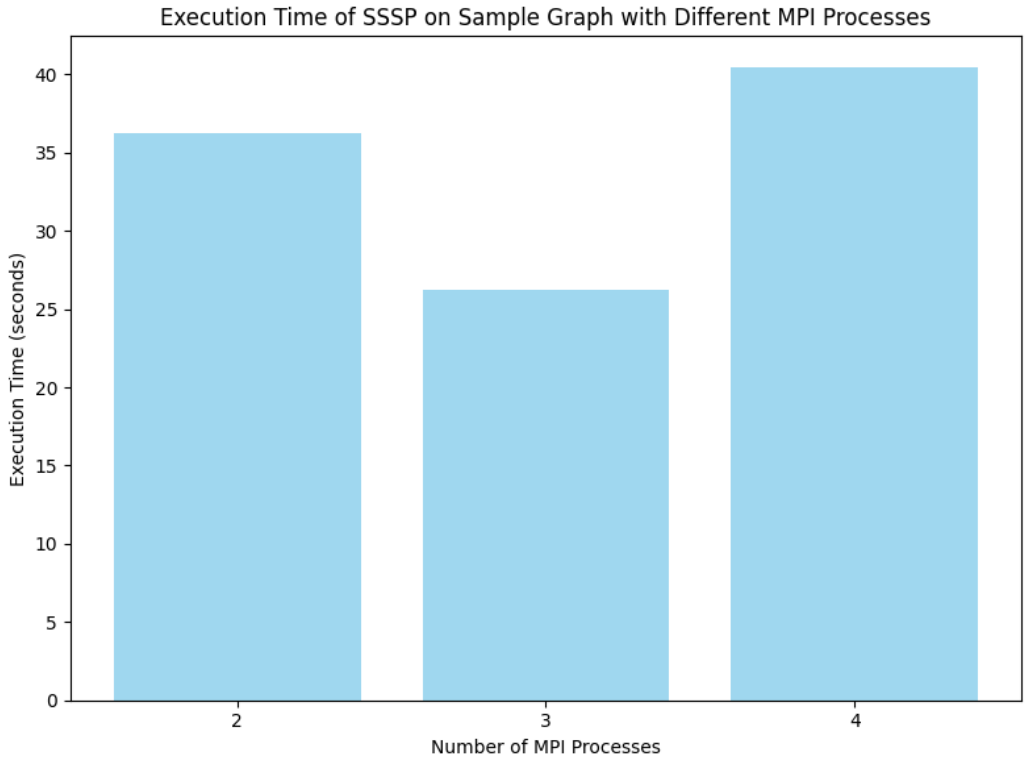
4. Implementation Details

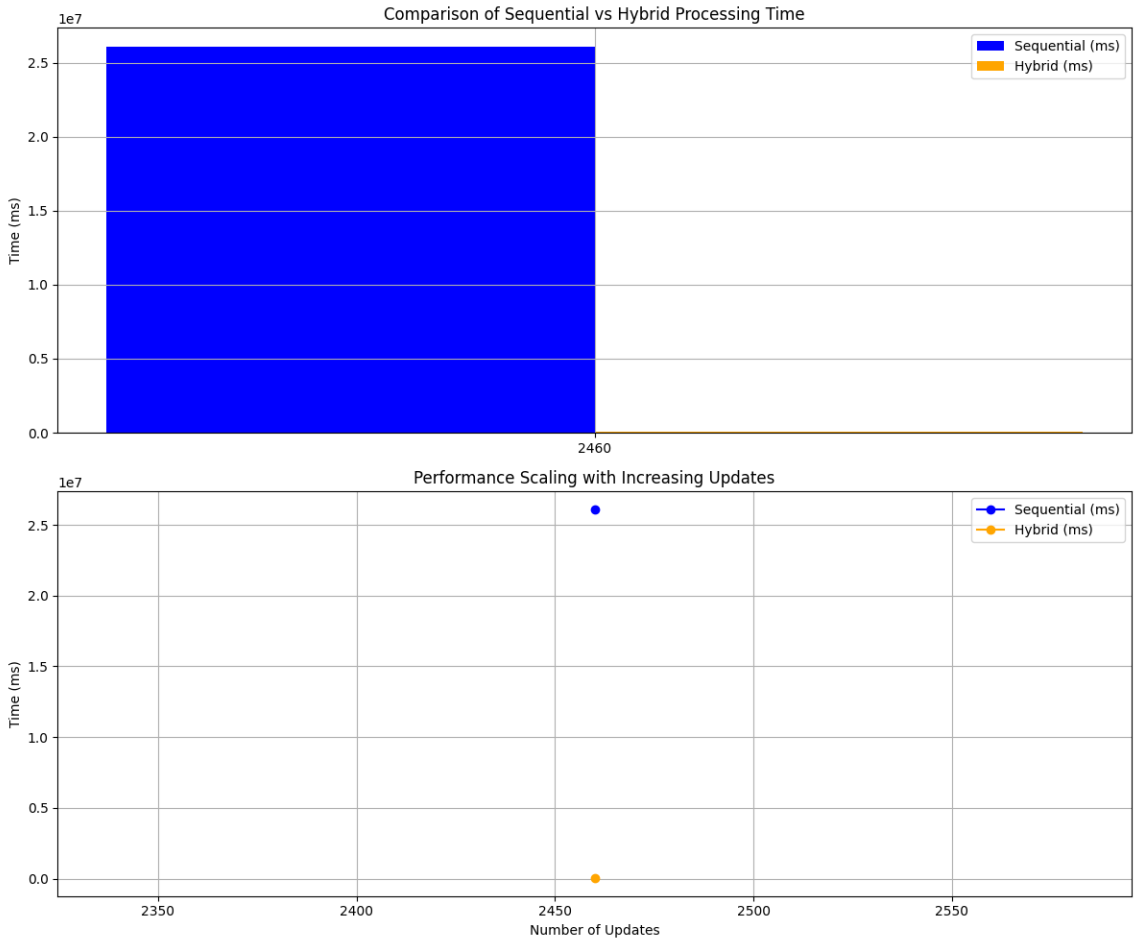
4.1 Serial Implementation
Implements Dijkstra's algorithm in serial_execution.cpp.
Command:
g++ -std=c++11 serial_execution.cpp -o serial_sssp
./serial_sssp sample_graph.txt sample_updates.txt 10000 output.txt

4.2 Parallel Implementation
Integrates MPI, OpenMP, OpenCL, METIS.
Command:
mpicxx -O3 -march=native -funroll-loops -fopenmp -DCL_TARGET_OPENCL_VERSION=200 -o sssp main.cpp graph.cpp utils.cpp sssp.cpp opencl_utils.cpp -I. -L/usr/local/lib -lOpenCL -lmetis

5. Performance Evaluation

Uses Python script plotGraph.py to visualize execution time across different process counts.
Outputs sssp_performance.png to show performance scalability.





6. Challenges and Solutions

6.1 Executable Accessibility Across Nodes
Ensured the sssp executable is present in all container volumes.

6.2 Hostfile Configuration Errors
Corrected the path and ensured format.

6.3 Serial Executable Not Found
Compiled serial_sssp using provided instructions.

7. Conclusion

This project demonstrates a robust hybrid parallel framework for SSSP updates in dynamic graphs. Using MPI, OpenMP, OpenCL, and METIS within Docker-based clusters enables performance, scalability, and reproducibility.

References

Khanda, A., Srinivasan, S., Bhowmick, S., & Norris, B. (2022). A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks. IEEE Transactions on Parallel and Distributed Systems, 33(4), 929–940.