Final Project Report: Parallel SSSP Update Using MPI, OpenMP, and METIS

Members: Muaz Ahmed (22i-1125), Ayan Asif (22i-1097), Muhammad Abbas (22i-1409)

Paper: A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks

1. Introduction

This project is based on the paper titled *"A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks"*. The paper proposes a highly parallel template for updating SSSP paths efficiently after dynamic changes in the network, enabling scalable, real-time updates in large graphs.

The selected paper proposes a framework that combines work-efficient, theoretically optimal techniques with practical considerations to deliver scalable performance. Our implementation leverages this template using MPI for inter-node communication, OpenMP for intra-node shared-memory parallelism, and METIS for efficient graph partitioning.

2. Tools and Technologies

- **MPI (Message Passing Interface):** For inter-node communication across a distributed memory system.
- **OpenMP (Open Multi-Processing):** For parallelism within individual nodes.
- **METIS:** For graph partitioning to optimize load distribution across MPI processes.
- **Dataset Used:** `mdual.graph` – a real-world graph used to test performance and scalability.

3. Problem Statement

The goal is to efficiently update the shortest paths from a source node in a large-scale graph upon dynamic changes (e.g., edge insertions or deletions). Sequential algorithms often re-compute the SSSP from scratch, which is inefficient. This project addresses the need for an incremental and scalable parallel approach.

4. Parallelization Strategy

- **Graph Partitioning (METIS):** The graph was partitioned using METIS to distribute subgraphs across MPI processes. This reduced inter-process communication and ensured load balance.
- **Inter-node Parallelism (MPI):** Each MPI process handled updates on a partition of the graph, communicating frontier changes with neighboring processes.
- **Intra-node Parallelism (OpenMP):** Within each MPI process, OpenMP was used to parallelize relaxations and updates of vertex distances.

5. Implementation Overview

- Read the partitioned graph using METIS.
- Each MPI process loads its subgraph and initializes local SSSP data structures.
- Upon an edge update, only affected nodes are propagated using a frontier-based model.
- OpenMP parallelizes the inner loop of vertex relaxations.
- Frontier updates are communicated via MPI.

6. Experimental Setup

- **Cluster:** 4-node MPI cluster with shared memory support.
- **Graph:** `mdual.graph`
- **Metrics:** Execution time, speedup, scalability (weak and strong), partition balance.

7. Challenges and Solutions

- **Challenge:** Communication overhead among MPI processes.
  - *Solution:* Optimized graph partitioning and minimized inter-process frontier exchanges.
- **Challenge:** Synchronization between MPI and OpenMP regions.

- *Solution:* Carefully structured parallel regions and used barriers where necessary.

8. GitHub Repository

The project code, presentation, datasets, and documentation are maintained at:
GitHub: https://github.com/molarmuaz/PDC_PROJECT

Features:

- Source code for MPI, OpenMP, METIS integration.
- Experimental data and plots.
- Slides and report documentation.

9. Conclusion

We successfully implemented and demonstrated a parallel algorithm for updating SSSP paths using MPI, OpenMP, and METIS. The approach is scalable, efficient, and adaptable to real-time dynamic graphs. This project validates the paper’s claim that theoretically efficient parallel graph algorithms can be practical and scalable in real-world settings.