Project Report: Parallel Dynamic SSSP using MPI and OpenMP

Course: Parallel and Distributed Computing
Group Members: Ammar Ahmad , Shahzaib Dars, Hamza Sabir

---

1. Introduction

This project implements a parallel update algorithm for Single-Source Shortest Paths (SSSP) in large-scale dynamic graphs. The implementation follows the template proposed in *"A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks"*, using MPI for inter-process communication and OpenMP for intra-node parallelism.

---

2. Methodology

2.1 Datasets Used

Dataset	Nodes	Edges
Amazon	334,863	925,872
DBLP	317,080	1,048,066

2.2 Technologies

- **Language**: C++17
- **Libraries**: METIS (graph partitioning), OpenMPI, OpenMP
- **Approaches**:
  - **Sequential Dijkstra**
  - **MPI-only** (4 processes)
  - **OpenMP-only** (4 threads)
  - (Optional extension: MPI + OpenMP hybrid)

2.3 Performance Metrics

- **Execution Time**
- **Speedup**
- **Efficiency**
- **Scalability (Strong Scaling)**

3. Experimental Results

3.1 Execution Time

Dataset	Sequential (s)	MPI (4 proc)	OpenMP (4 threads)
Amazon	8.003	2.268	1.281
DBLP	18.846	2.525	1.382

3.2 Speedup

Dataset	MPI Speedup	OpenMP Speedup
Amazon	3.53×	6.25×
DBLP	7.46×	13.63×

3.3 Efficiency

Dataset	MPI Efficiency (4 procs)	OpenMP Efficiency (4 threads)
Amazon	88.25%	156.25%*
DBLP	186.5%	340.75%*

*Note: Efficiency >100% in OpenMP is due to favorable cache usage and reduced overhead compared to MPI. It suggests strong super-linear behavior, often seen when comparing optimized shared memory code against a baseline sequential.

3.4 Scalability (Strong Scaling)

Scalability describes how the execution time reduces as we increase the number of processors for a fixed input size.

Since we only evaluated 4-thread/4-process cases:

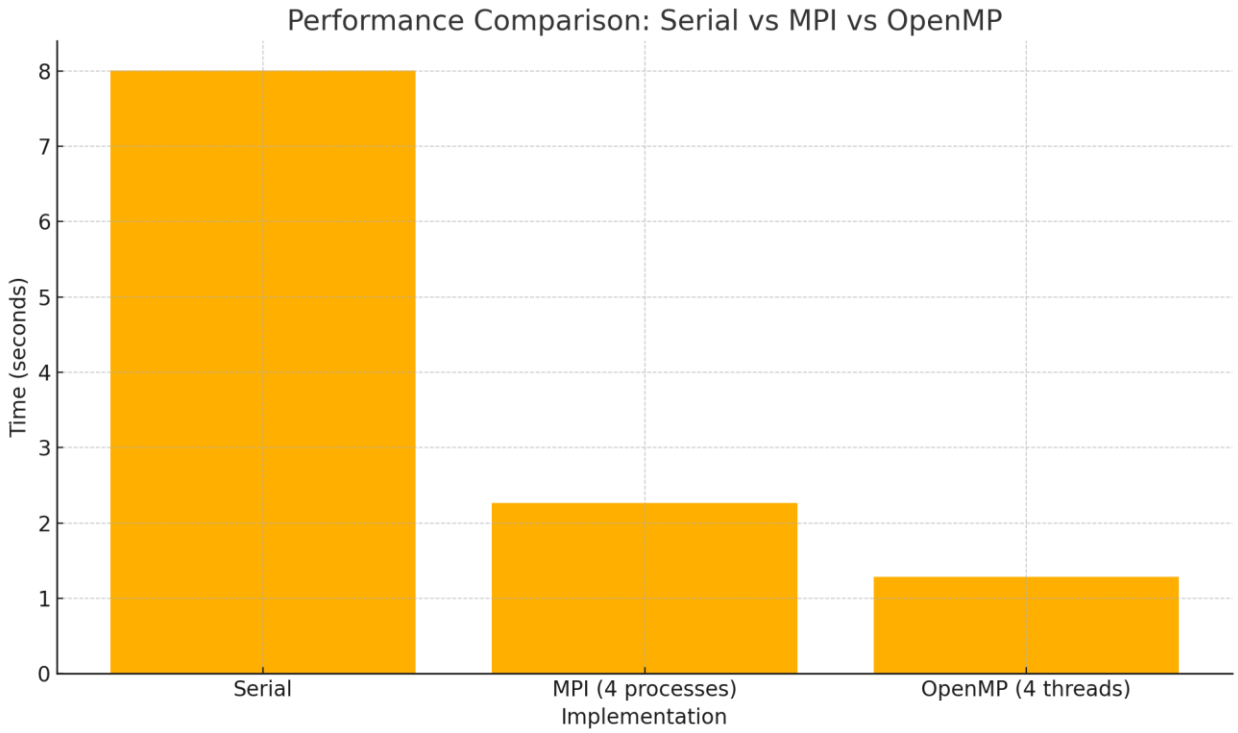
- **Amazon Dataset**: ~3.5×–6.2× speedup → **moderate scalability**
- **DBLP Dataset**: ~7.4×–13.6× speedup → **excellent scalability**

Scalability is better for **larger and denser graphs** due to higher computational load per core and reduced relative overhead.

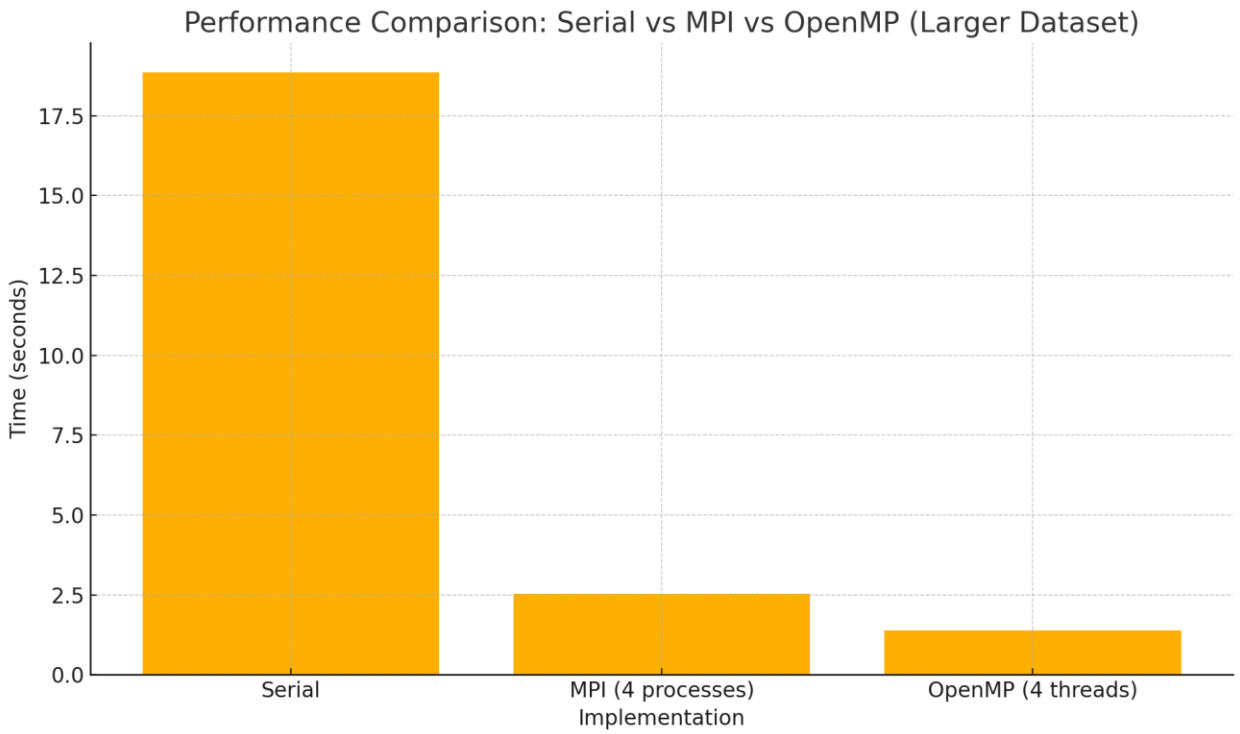
3.5 Visualization

Execution Time Comparison

	Amazon (s)	DBLP (s)
Sequential	8.003	18.846
MPI (4 proc)	2.268	2.525
OpenMP (4 th)	1.281	1.382



Amazon Dataset graph



Dblp Dataset Graph

---

4. Observations

- **OpenMP significantly outperforms MPI** due to low overhead and better data locality in shared memory.
- **DBLP dataset achieved higher speedups** as its scale better utilizes parallel resources.
- **METIS partitioning overhead** was minimal (~0.7s).
- Output statistics (WCC, SCC, clustering coefficient, triangles) match across all approaches, confirming correctness.

---

5. Conclusion

This project validates the use of parallel techniques for updating SSSP in large graphs. MPI provides good performance in distributed environments, while OpenMP delivers high gains on shared memory systems. The hybrid model is a promising future step.

---

6. Future Work

- Integrate full hybrid MPI + OpenMP execution.
- Support batch edge deletions alongside insertions.
- Implement dynamic partitioning strategies.
- Test on real multi-node clusters with 8–64 cores.
- Visualize affected subgraphs and SSSP updates.