Single-Source Shortest paths in large-scale dynamic networks

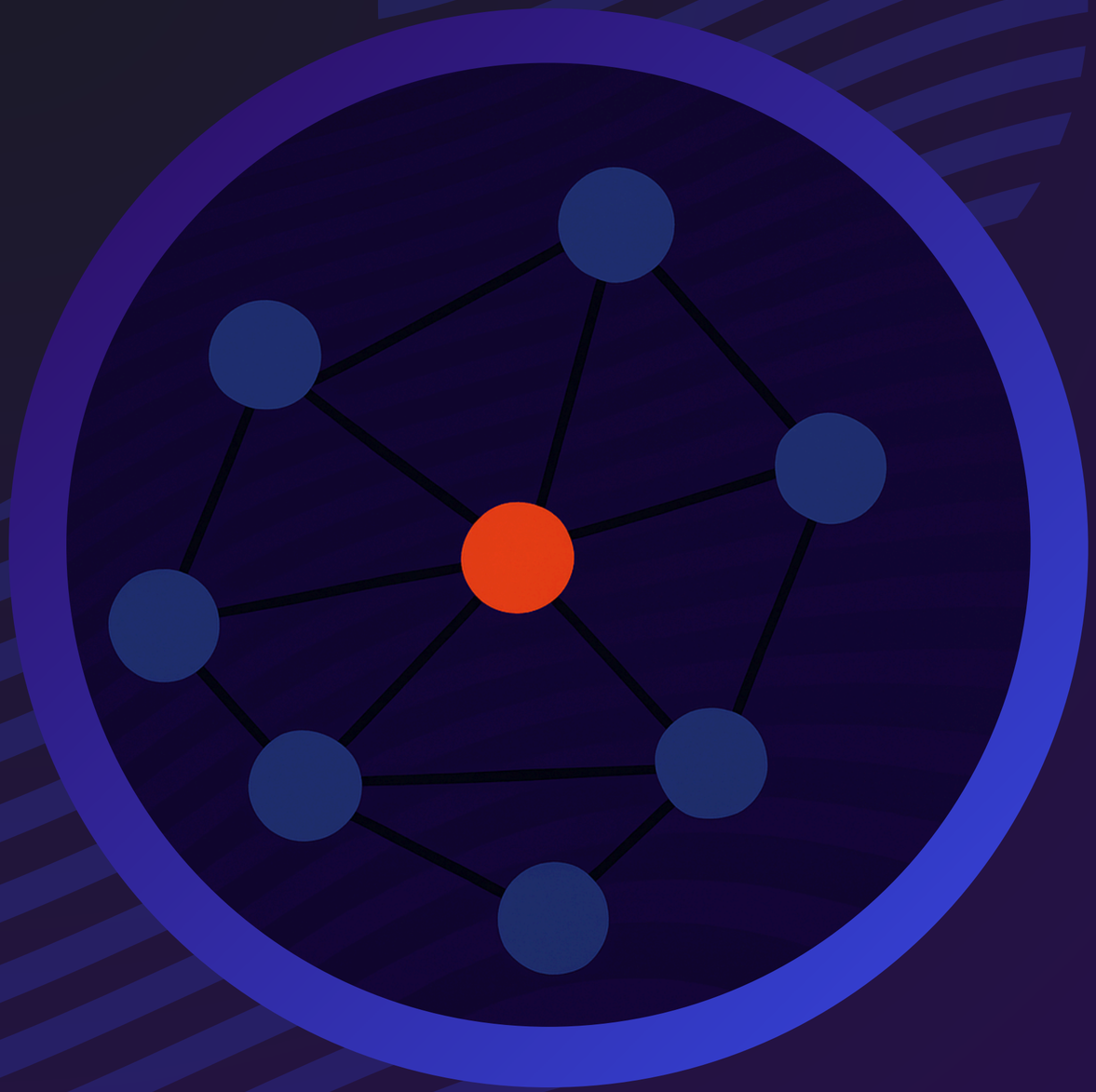
By:

Arrij Fawwad 22i-0755

Hamna Arshad 22i-1098

Maryam Masood 22i-1169

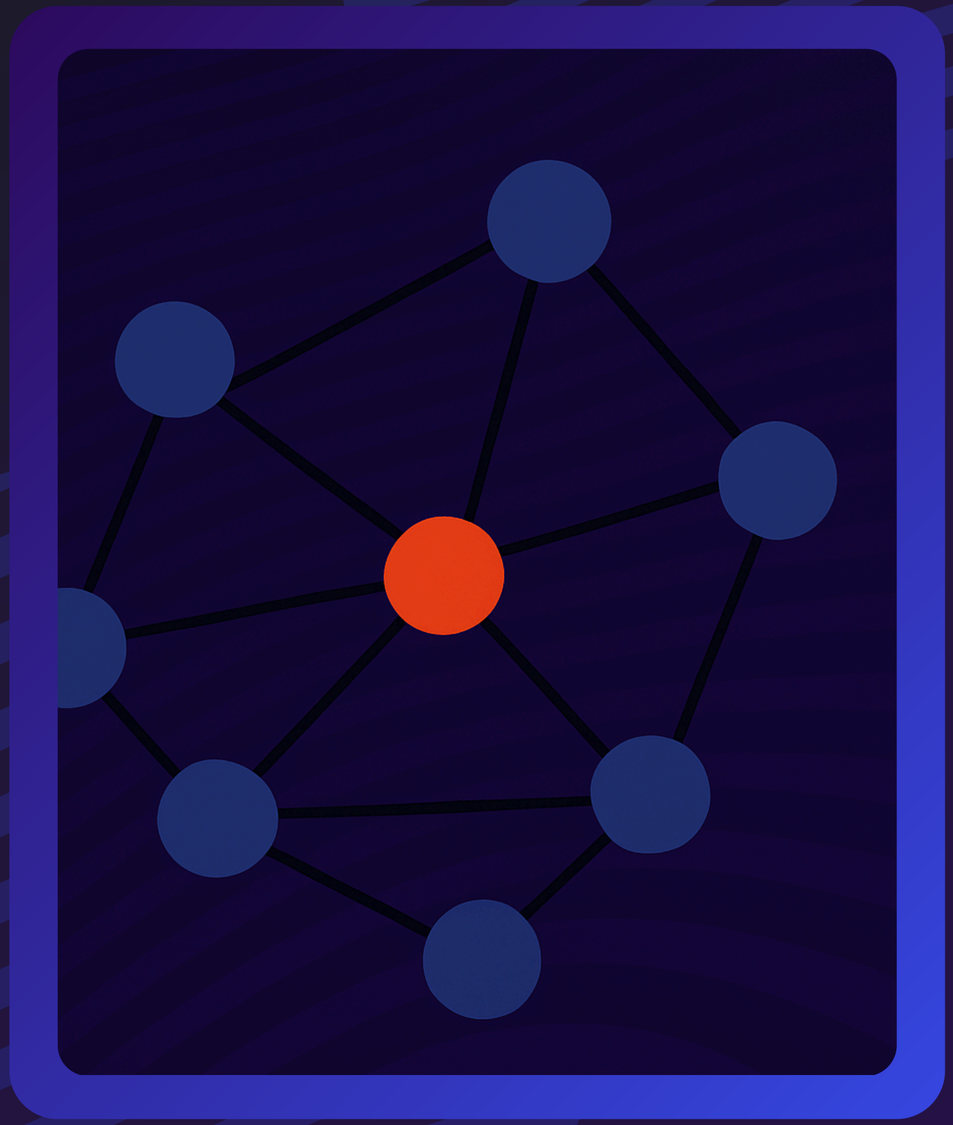
PDC



INTRODUCTION

- SSSP: Shortest paths from one source to all nodes
- Real-world graphs change over time
- Recomputing SSSP is slow and inefficient
- Existing methods often platform-specific
- Proposed: fast, parallel SSSP update framework

PDC



KEY CONTRIBUTION

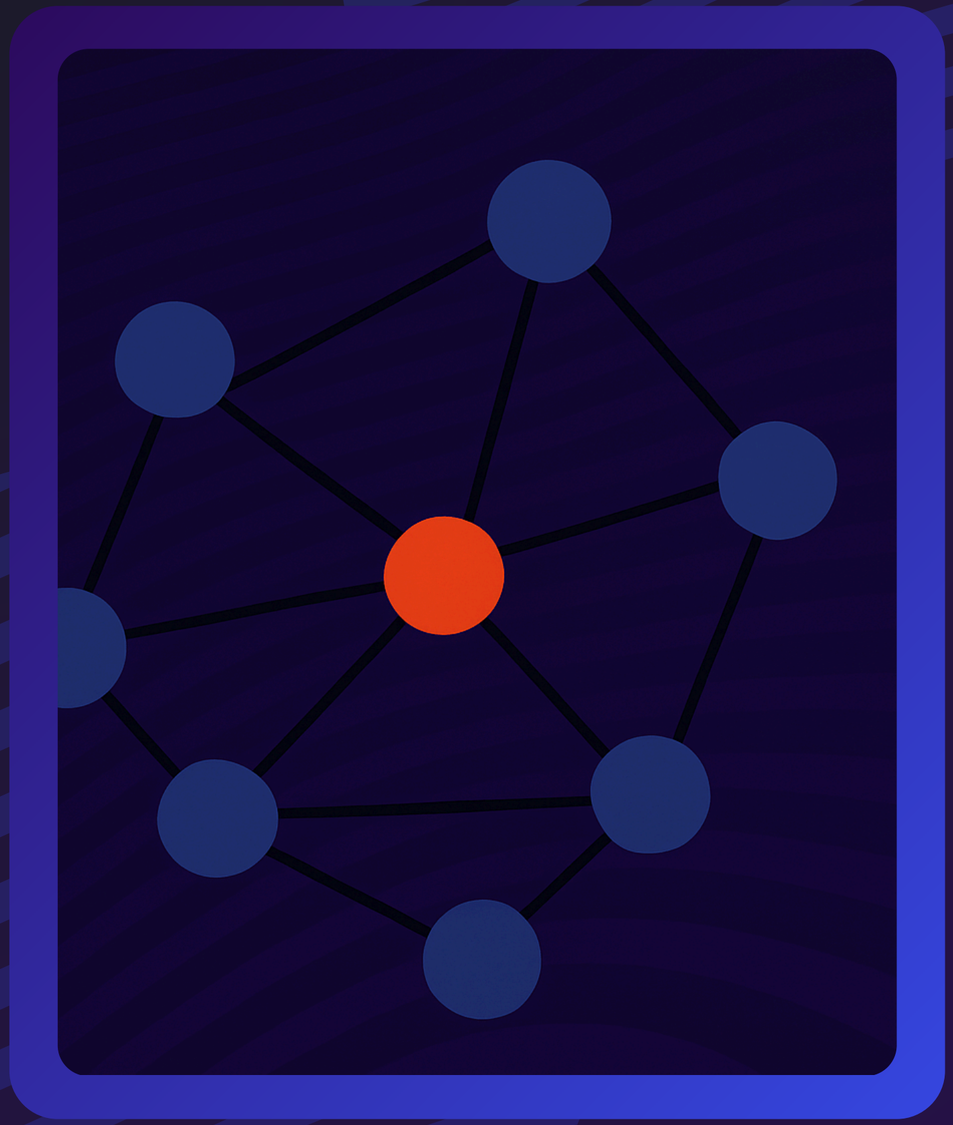
Efficient SSSP Updates via Localized Parallelism

- Generic framework to efficiently update single-source shortest paths in dynamic graphs.
- Avoids full recomputation by updating only affected areas.

Supports both Insertions and Deletions

- Handles edge insertions, deletions, and weight changes.
- Maintains correctness with selective updates.

PDC



KEY CONTRIBUTION

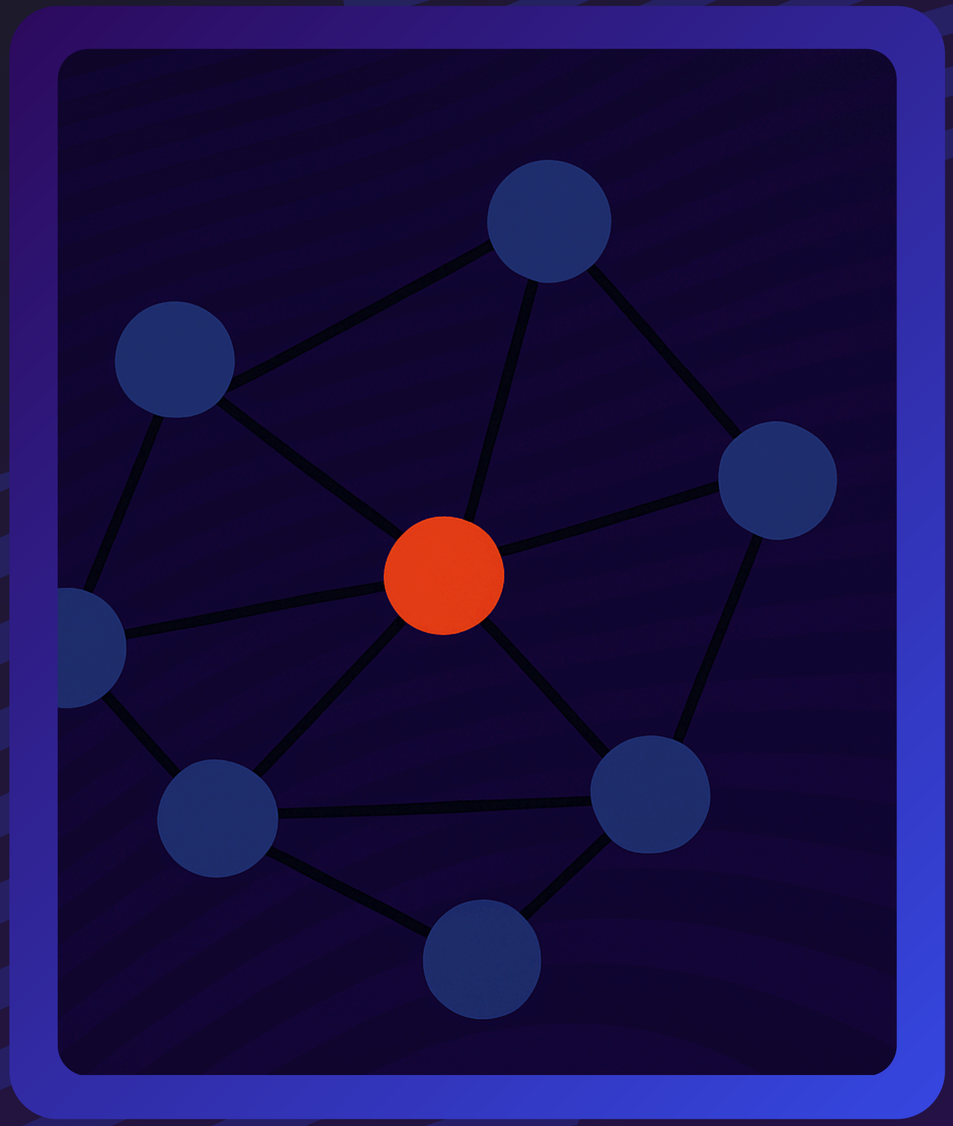
Work-Efficient & Parallel-Friendly

- Minimizes synchronization by using lock-free, iterative updates
- Uses dynamic scheduling to handle load imbalance efficiently

Modular & Flexible Design

- Compatible with multiple SSSP algorithms (e.g., Dijkstra, Bellman-Ford).
- Easily adaptable for different graph types and systems.

PDC



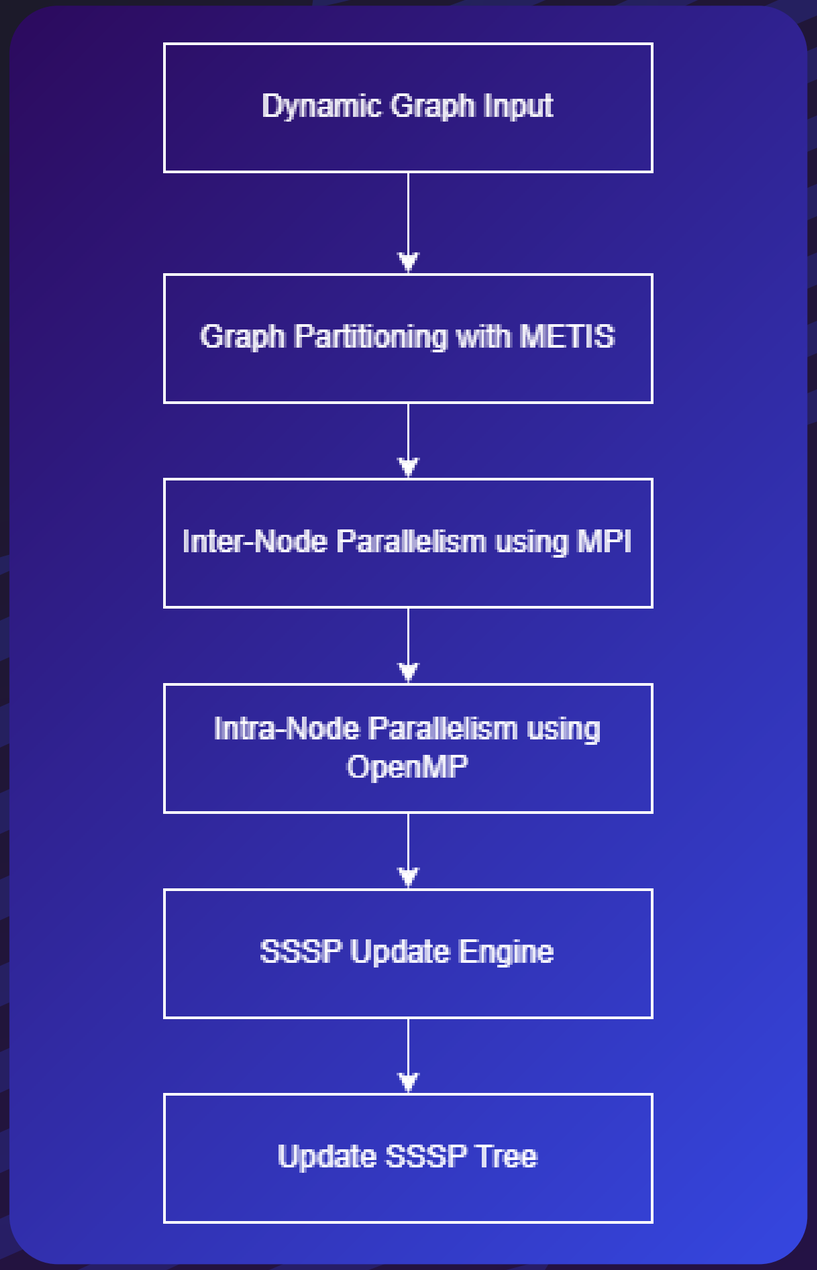
KEY CONTRIBUTION

Strong Empirical Results

- Tested on large real-world networks.
- Achieves 10–20× speedup on multi-core setups.

Scalable for Large Dynamic Graphs

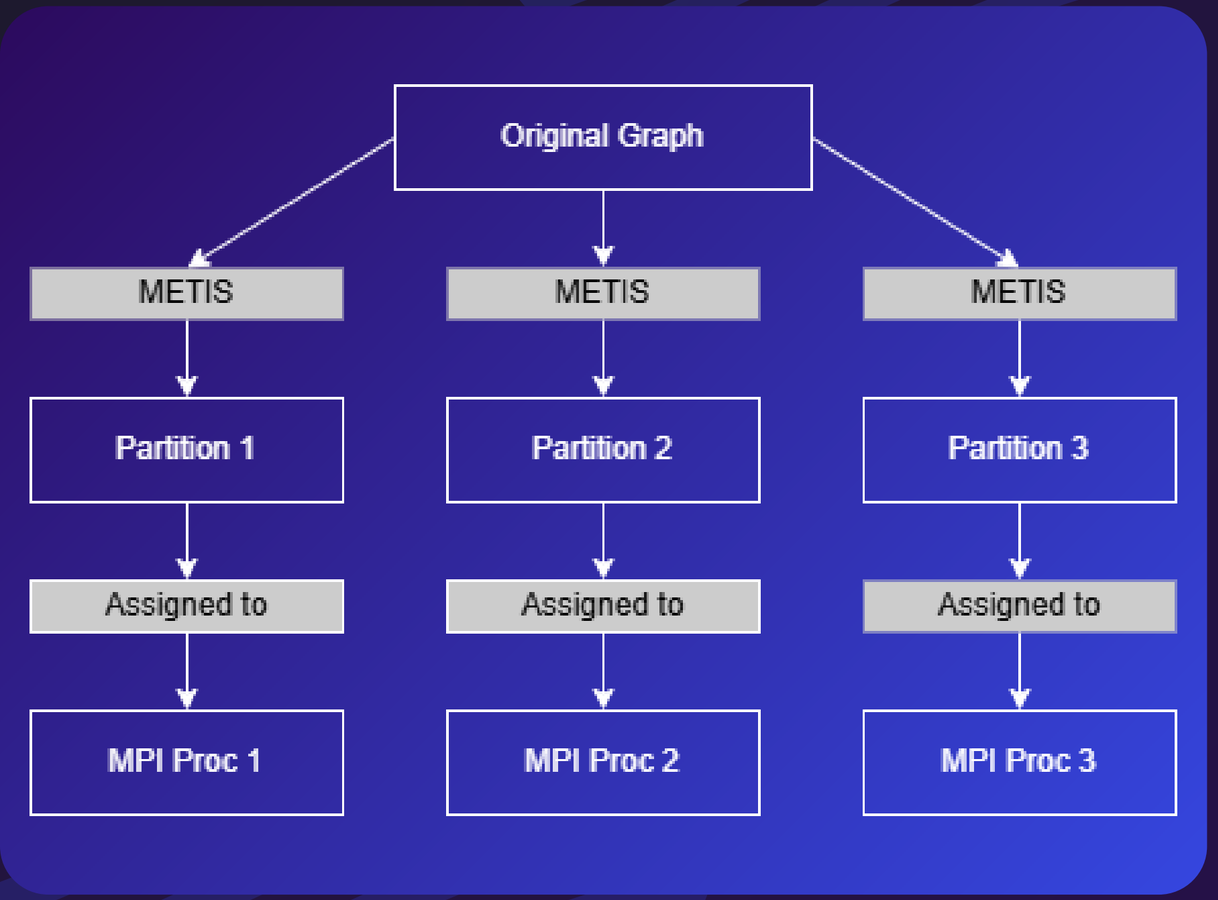
- Efficiently handles large-scale and frequently changing networks.
- Ideal for real-time applications like traffic, social, or web graphs.



PARALLELIZATION STRATEGY

- Graph Partitioning With METIS
- Inter-Node Parallelism with MPI
- Intra-Node Parallelism with OpenMP

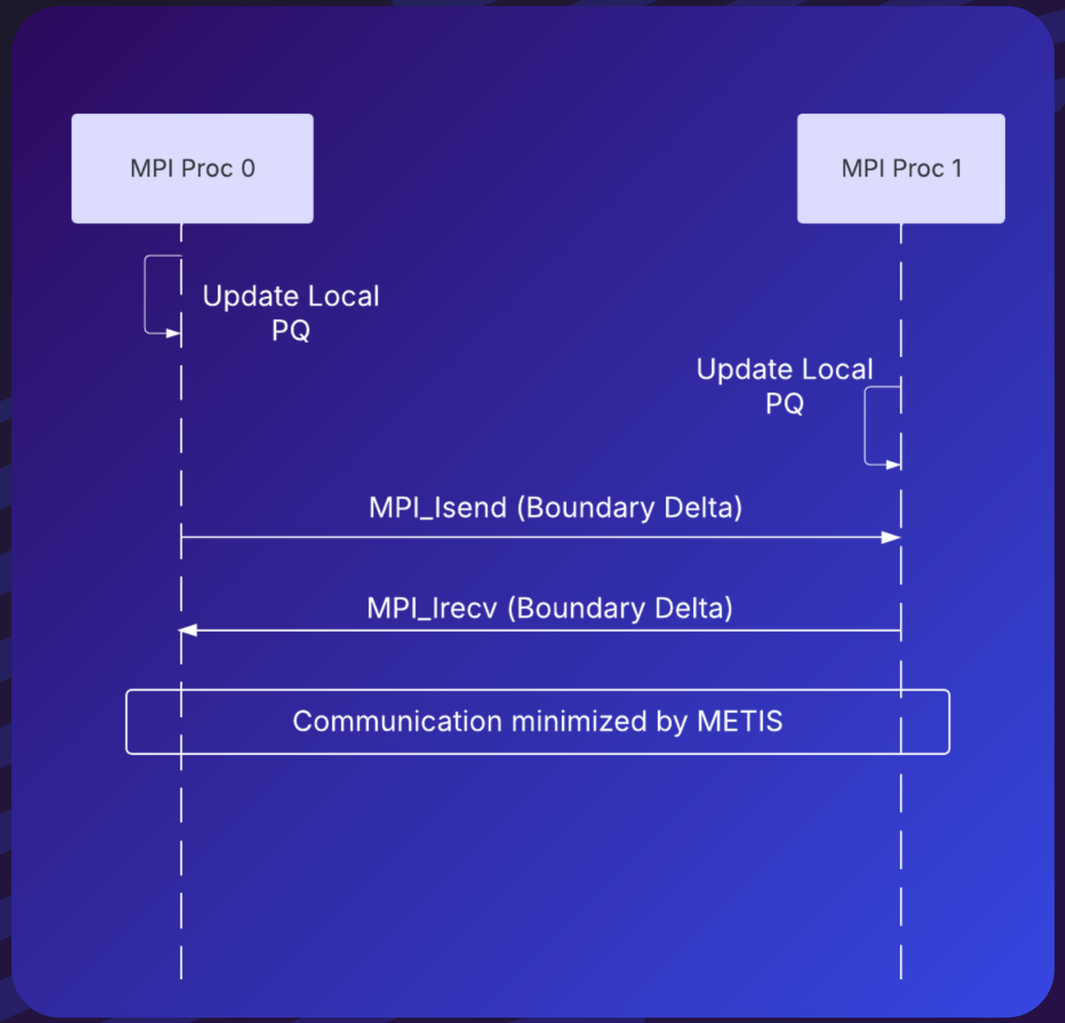
GRAPH PARTITIONING WITH METIS



- Input graph is divided into K balanced subgraphs
- Partitions aim to:
  - Balance workload across processes
  - Minimize edge cuts (inter-process communication)
- Each partition is assigned to a separate MPI process

PDC

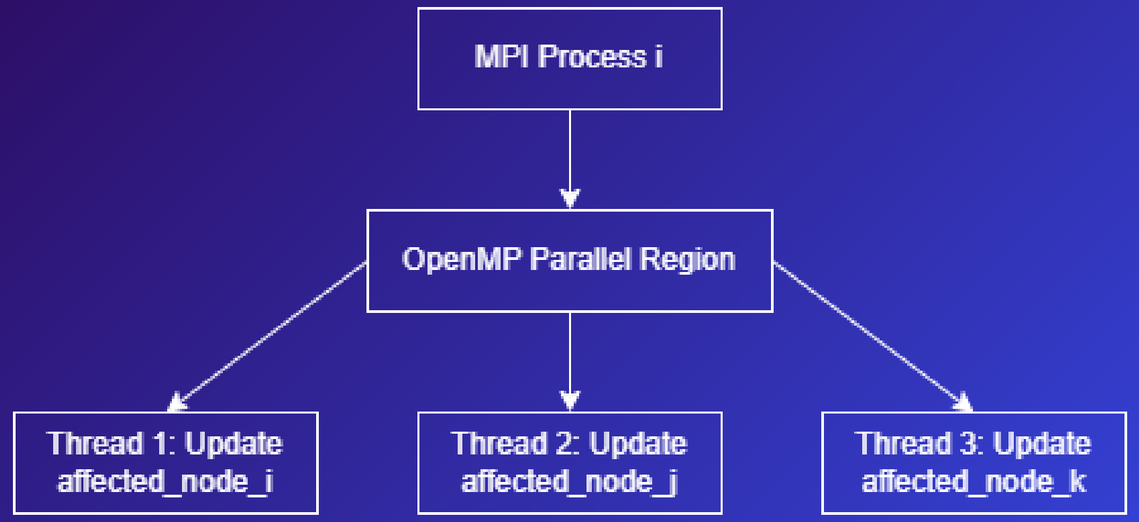
INTER-NODE PARALLELISM-MPI



- Each MPI process operates independently on its subgraph
- Communicates updates at partition boundaries
- Uses non-blocking communication to reduce wait times

PDC

INTRA-NODE PARALLELISM-OPENMP

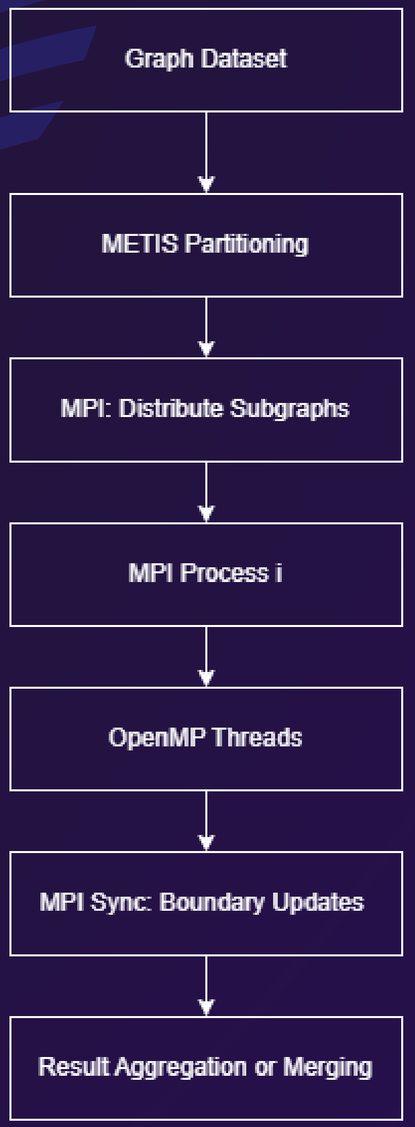


- Each MPI process runs multi-threaded logic

- Applies OpenMP for:
  - Processing affected nodes in parallel
  - Relaxing distances
  - Neighbor updates

PDC

ALGORITHM FLOW



OPTIMIZATION

1. **Reduce Communication:**
   - send only delta updates
2. **Overlap Compute & Comm:**
   - MPI_Isend is non-blocking
3. **Load Balancing:**
   - initial balance by METIS, monitor runtime skew
4. **Thread Scaling:**
   - use OpenMP dynamic scheduling

PERFORMANCE METRICS WE'LL MEASURE

1. Execution Time:

- Sequential vs MPI vs MPI+OpenMP

2. Speedup:

- Speedup=T1/Tp

3. Scalability:

- Strong: fixed graph, more cores
- Weak: graph grows with cores

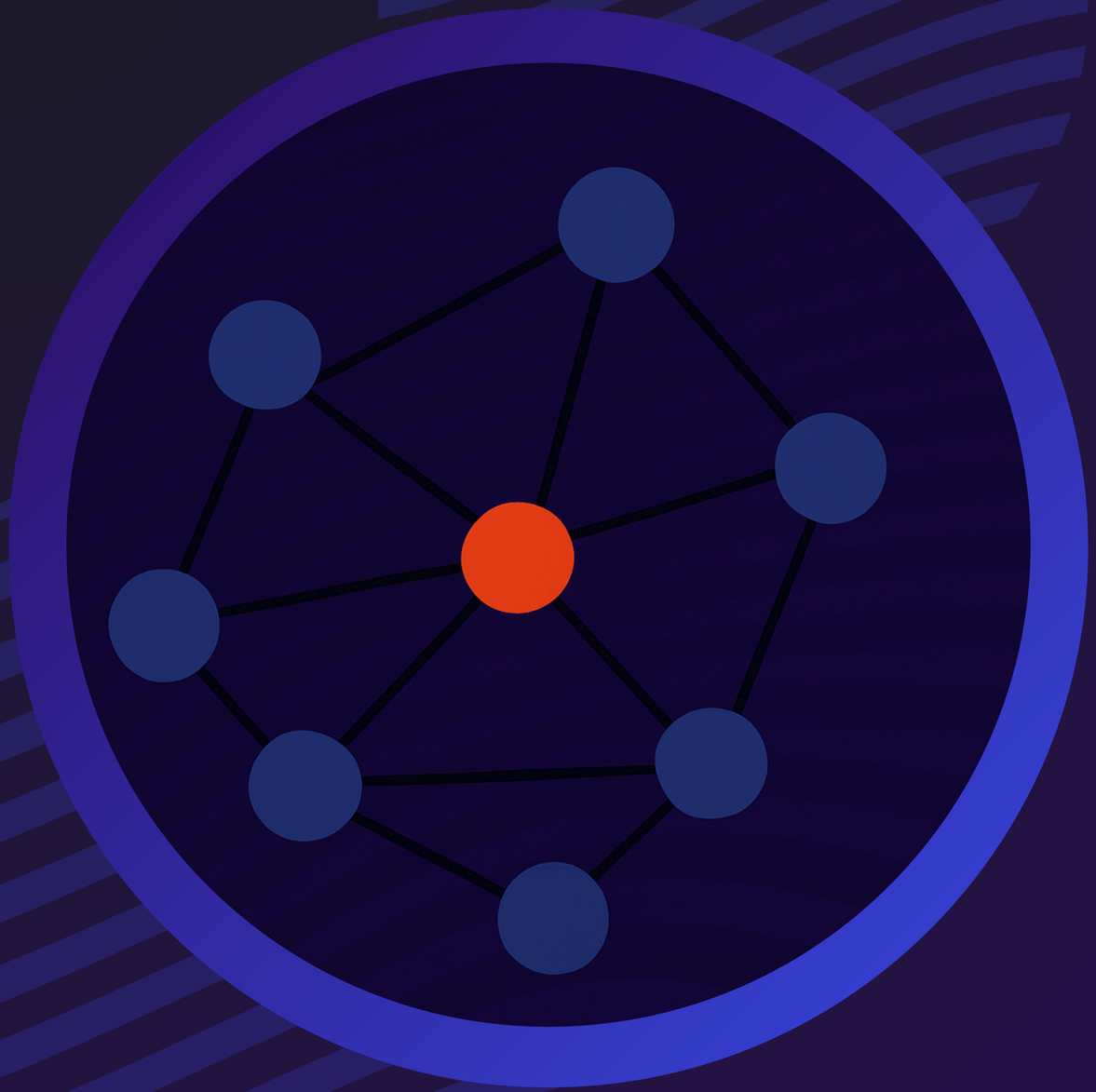
4. **Comm Overhead:**

- Time spent in MPI

5. **Partitioning Quality:**

- Edge cuts and load imbalance

PDC



Conclusion