PROJECT- PHASE 1

Parallel and Distributed Computing

GROUP MEMBERS:

HUZEMA SAIF (20I-0466)

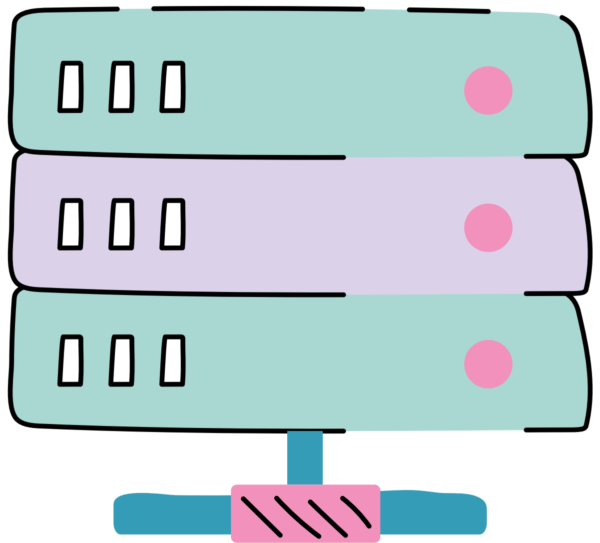
M ZAIN-UL-ABIDEEN (20I-0821)

BILAL MUSTAFA (20I-0901)

Title: Parallel SSSP Updates in Dynamic Graphs

Introduction

- Real-world graphs (e.g., road networks, social networks) are **dynamic**.
- Traditional SSSP algorithms are **inefficient** for frequent updates.
- **Our focus**: Parallel computing (**MPI + OpenMP**) for efficient SSSP updates in dynamic graphs.



Problem Statement

- Graphs change frequently; **recomputing** SSSP each time is too slow.

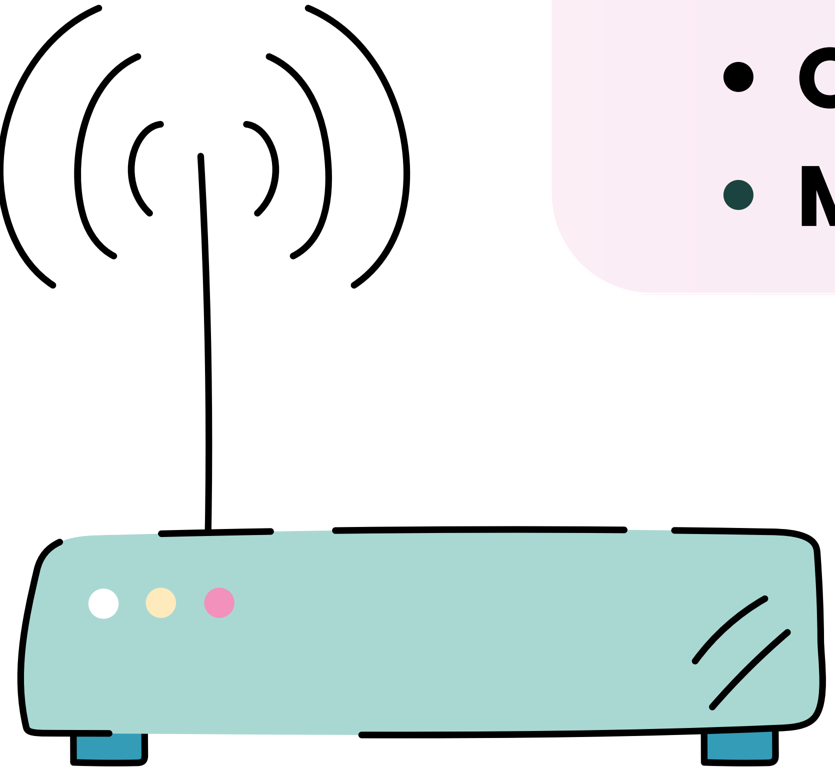
- Need for fast, scalable update methods on large-scale **dynamic graphs**.

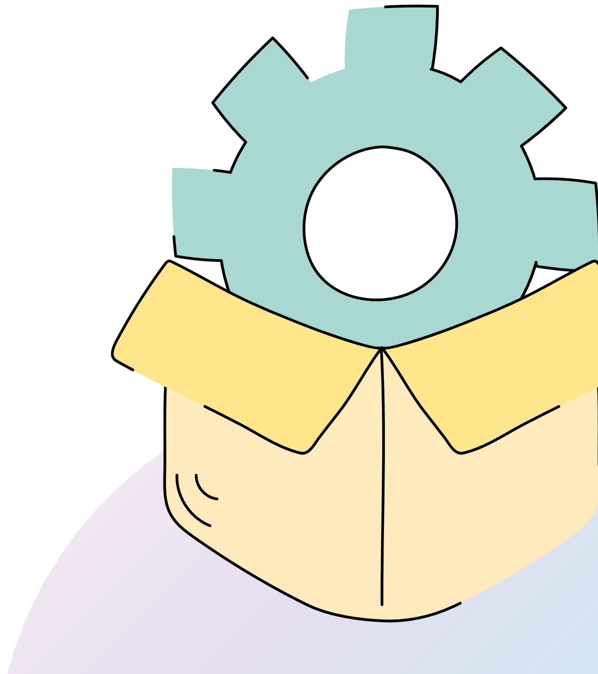
Project Goal

Develop a parallel solution to update SSSP in dynamic graphs.

Combines:

- **MPI** for distributed parallelism.
- **OpenMP** for intra-node threading.
- **METIS** for efficient graph partitioning.



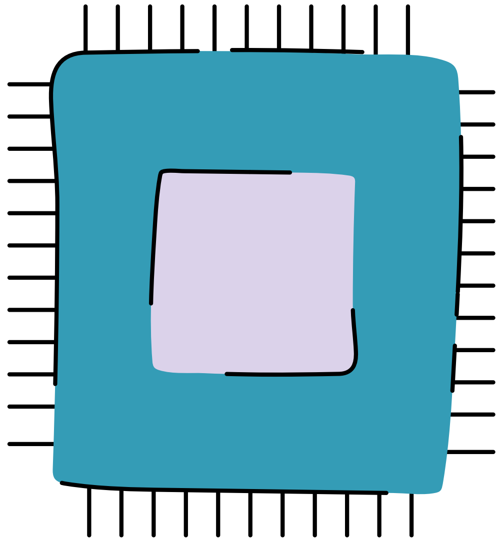


Key Concepts

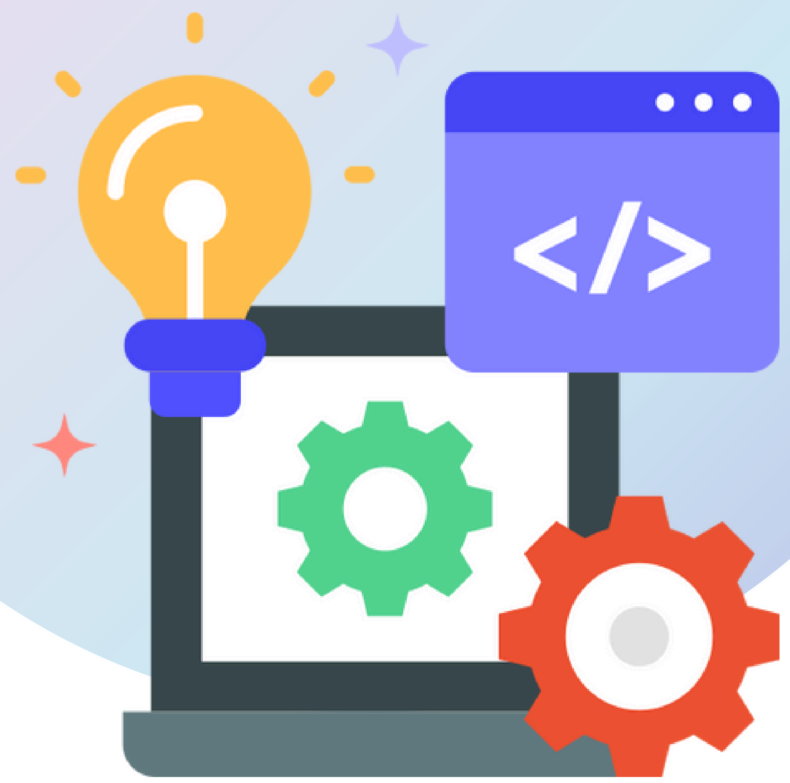
Delta-based updating: updating affected parts.

Highly parallelizable: Minimize redundant work, maximize performance.

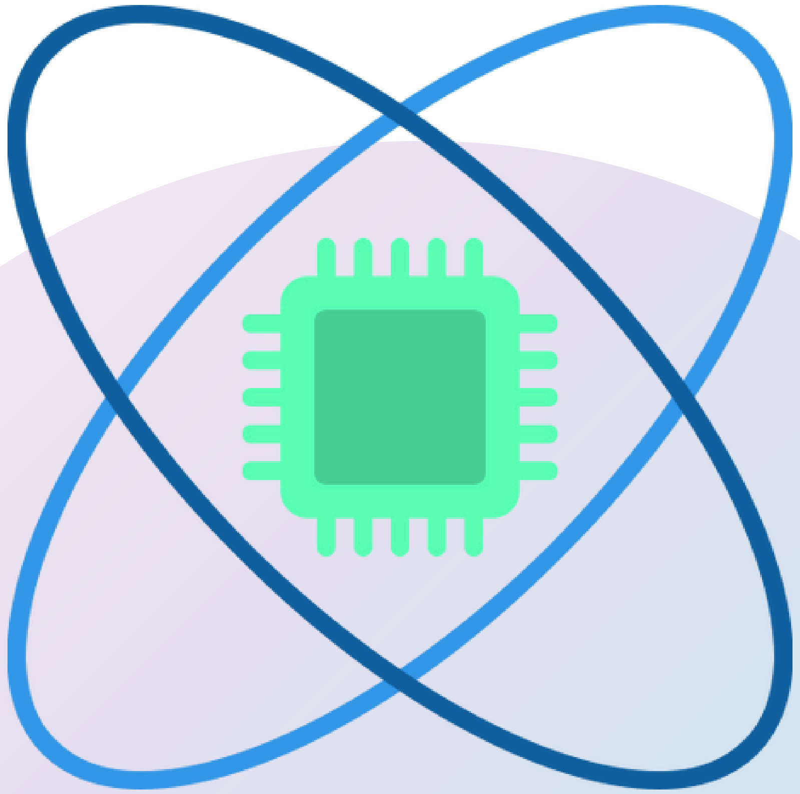
Avoids full recomputation of the entire graph.



Technology Stack



- **MPI:** Distributed memory parallelism.
- **OpenMP:** Shared memory multithreading.



Software

- **METIS:** Graph partitioning tool.
- **Optional Tools:** Benchmarking with MPI_Wtime, Intel Trace Analyzer.

Workflow

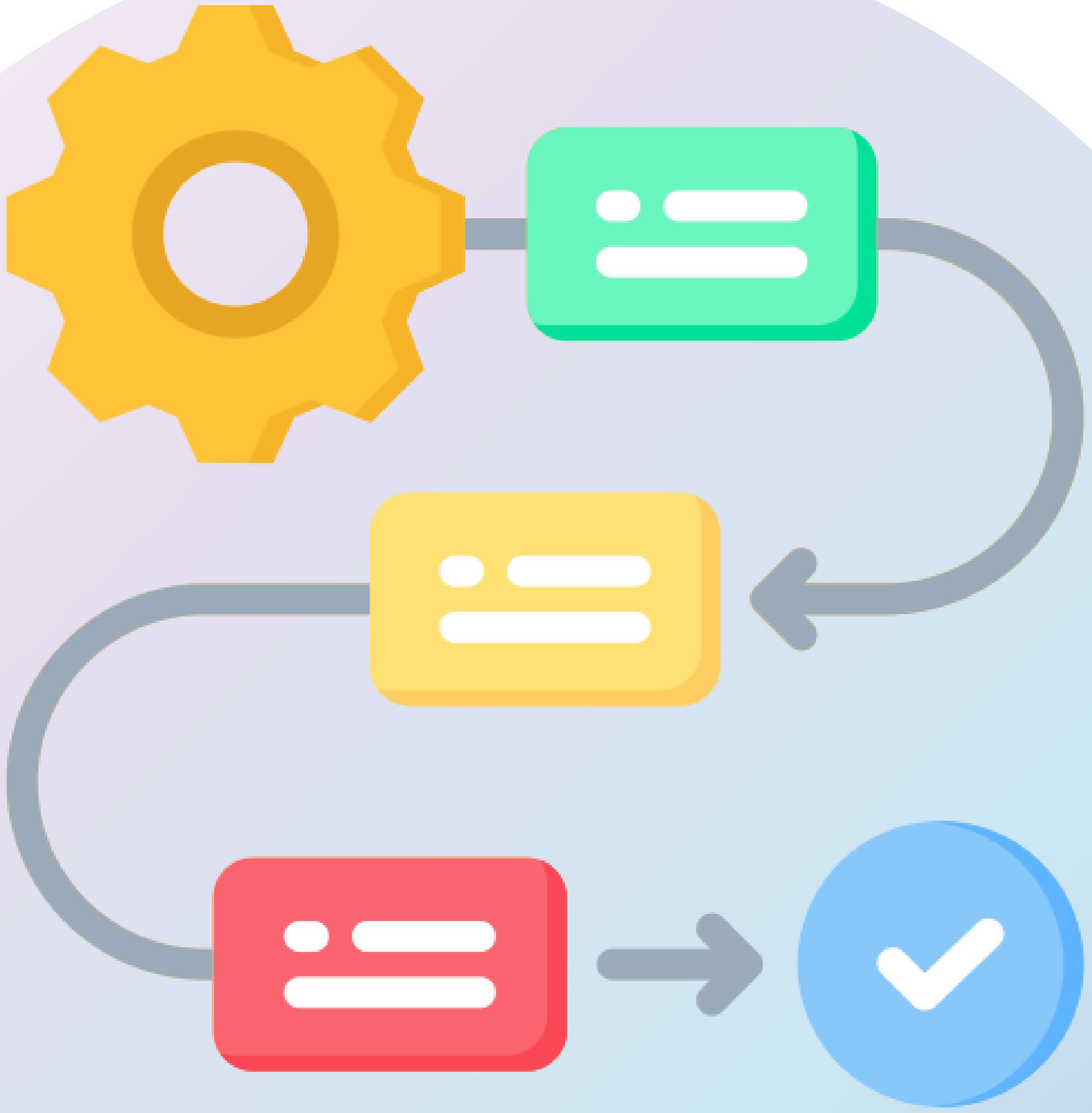
1. Partition input graph using **METIS.**

2. Compute initial SSSP in **parallel**.

3. Inject dynamic updates (**edges**).

4. Update only affected regions using **delta-front strategy**.

5. Synchronize across **MPI** processes.



Algorithm Overview: Two-Step Parallel Update

Step 1: Identify Affected Subgraph

- Each changed edge (insertion/deletion) is processed in parallel.
- Affected vertices are marked without needing synchronization.
- Insertions update distances, deletions disconnect nodes.

Step 2: Update Affected Subgraph

- Affected vertices update their distances iteratively.
- Minimal synchronization needed (no critical sections).
- Ensures convergence to correct shortest paths.

🔹 **Main Idea:**

Only update what's necessary, avoid redundant computations.

Role of METIS in Graph Partitioning

Why METIS?

- Dynamic graphs can become imbalanced after updates.
- METIS divides the graph into balanced partitions.
- Minimizes cross-node communication in MPI.

Impact:

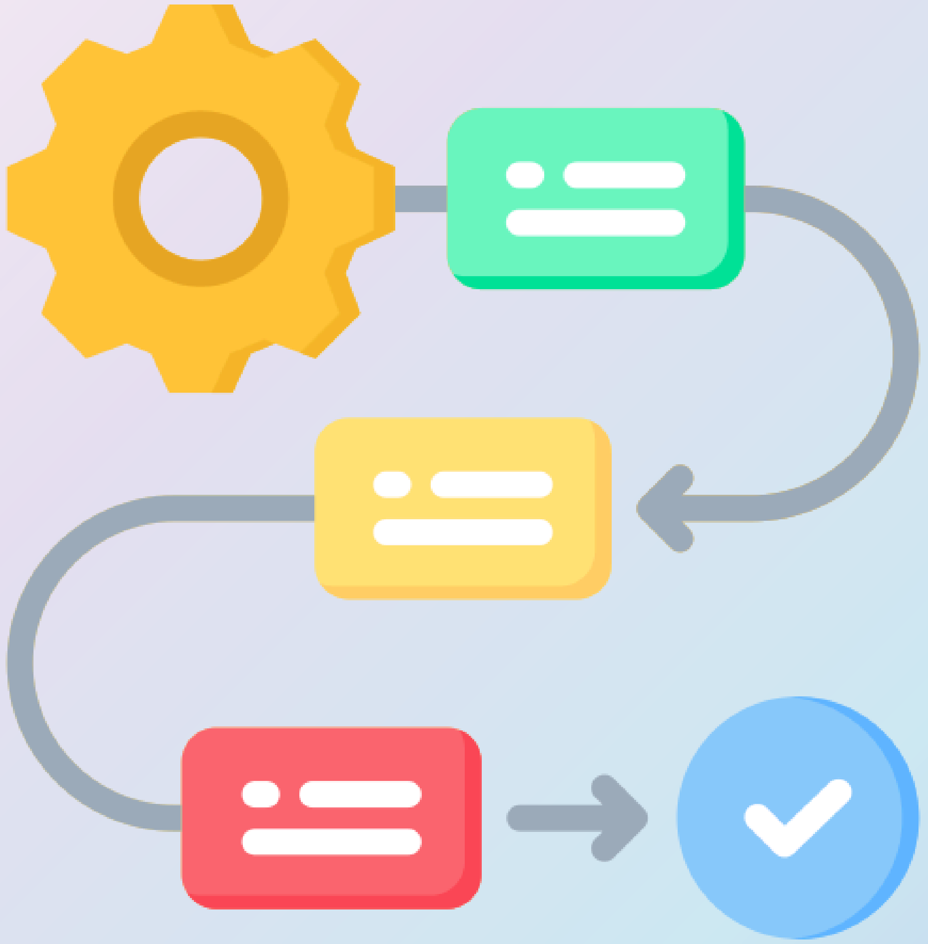
- Better load balancing.
- Faster parallel processing.
- Scalability improvement for large graphs.

🔹 Key Point:

Good partitioning = Less communication + More parallel efficiency!

GitHub Workflow

- **Main branch:** Stable version.
- **Dev branch:** In-progress development.
- **Feature branches:** For isolated modules.
- Well-structured repository with source code, data, and scripts.



Dataset & Metrics

1

Datasets:

- roadNet-CA (**~2M nodes**).
- soc-LiveJournal1 (**~4.8M nodes**).

2

Evaluation Metrics:

- Speedup.
- Efficiency.
- Scalability (Strong Scaling, Weak Scaling).

Progress & Next Steps

Phase 1 Completed:

- Literature Review.
- Architecture Design.
- Environment Setup.

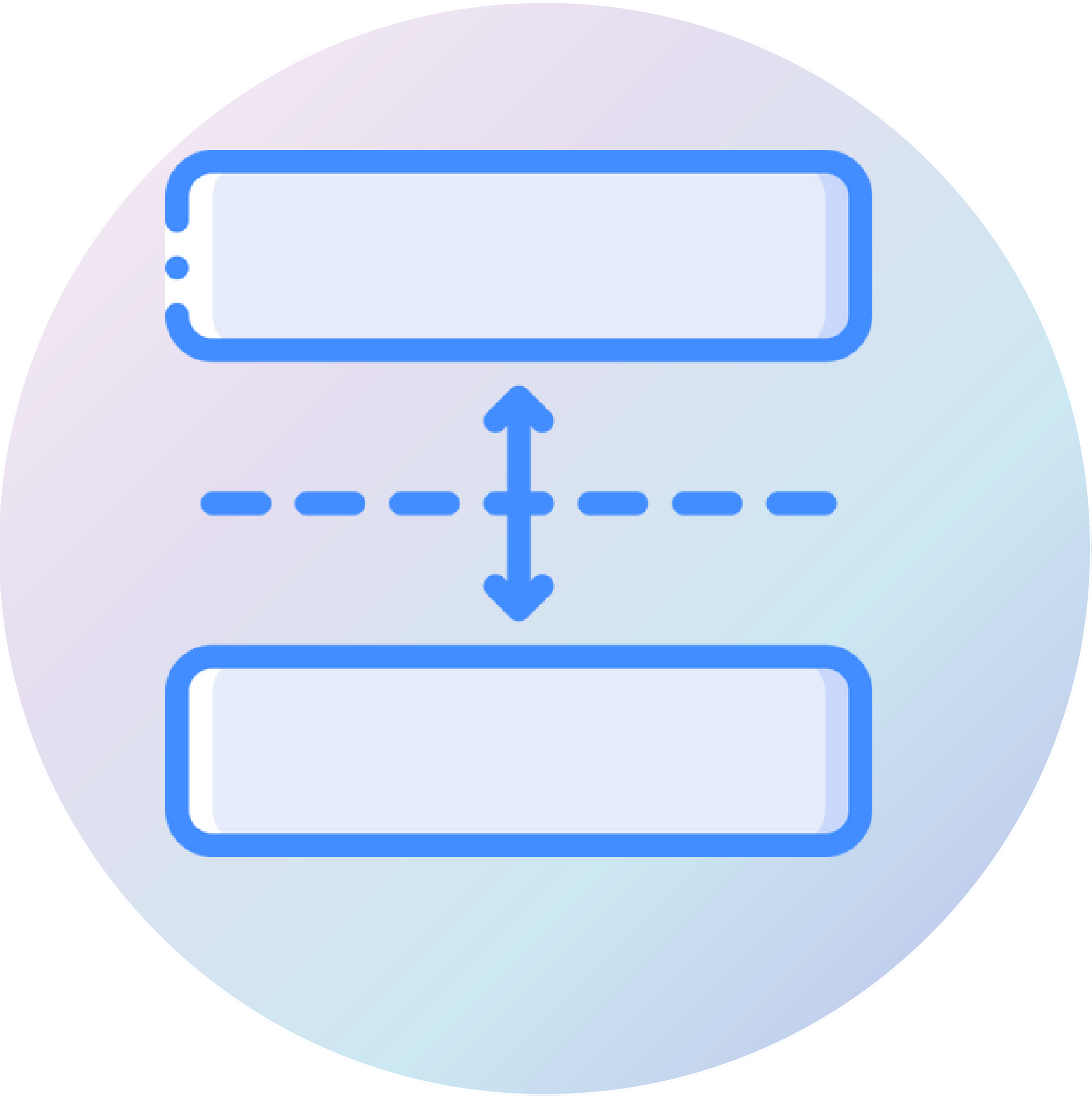
Next:

- Implement METIS Integration.
- Full Parallel Implementation.
- Benchmarking and Scalability Testing.

Experimental Results from Research Paper

Performance Achieved in Research Paper:

- GPU implementation outperforms Gunrock by up to **5.6x** speedup.
- CPU shared-memory implementation outperforms Galois by up to **5x**.
- Updates hundreds of millions of edges efficiently.
- Works well on real-world networks like LiveJournal, Orkut, **Graph500**.



Important Observations:

- Update strategy is better when changes are mostly insertions.
- Recomputation is better when changes are mostly deletions (**>75%**).

Like Delta-Stepping Concept

- Groups nodes into distance-based buckets.
- Processes only affected regions in parallel.
- Minimizes unnecessary computation.

Architecture Overview

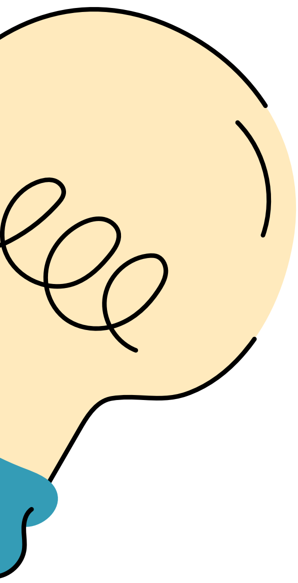
Pipeline:

Input Graph → METIS Partition → MPI Distributed Nodes → OpenMP Threads (per node) → Output.

Result:

- High scalability.
- Balanced workload.
- Fast updates.

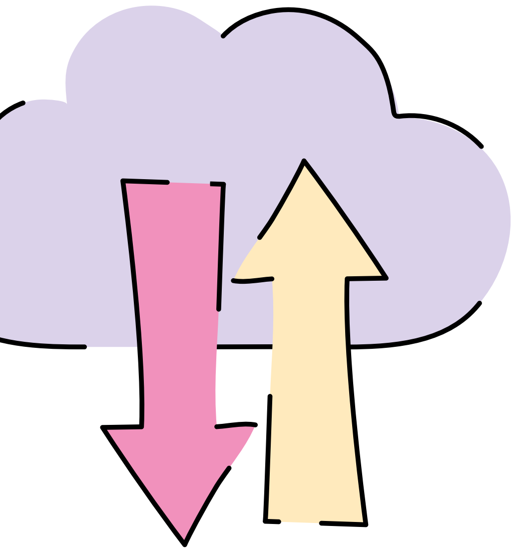
Conclusion



- Dynamic graph updates require efficient SSSP strategies.
- Delta-based parallel updating achieves major speedups.
- Combination of MPI + OpenMP + METIS gives high performance.

Future Work

- Hybrid approaches for selective recomputation vs updating.



Thank You