PARALLEL ALGORITHM FOR UPDATING

SINGLE-SOURCE SHORTEST PATHS IN LARGE-SCALE DYNAMIC NETWORKS

PRESENTED BY:

Raima Tariq , Rimsha Azam , Javeria Waseem

PROBLEM CONTEXT

WHY DYNAMIC SSSP?

- Real-world networks (social, transport) evolve over time.
- Traditional SSSP algorithms (e.g., Dijkstra) assume static graphs → inefficiency.

CHALLENGES:

- Load balancing with uneven subgraph sizes.
- Synchronization overhead in parallel updates.

KEY CONTRIBUTIONS

1.UNIFIED FRAMEWORK

- Parallel SSSP updates for dynamic networks on CPUs/GPUs.
- Rooted tree data structure for efficient updates.

2. GPU OPTIMIZATION

- Vertex-Marking Functional Block (VMFB) reduces redundant computation.
- 5.6x speedup over Gunrock (recomputation).

3. SHARED-MEMORY SCALABILITY

- Batch processing (up to 100M changes).
- 5x speedup over Galois.

4. ASYNCHRONOUS UPDATES

- Iterative convergence avoids expensive synchronization.

PROPOSED FRAMEWORK

A TWO STEP APPROACH:

1. IDENTIFY AFFECTED SUBGRAPHS

- Parallel edge processing (insertions/deletions).

2. UPDATE SSSP TREE

- Iterative distance relaxation (no locks).

PROPOSED PARALLELIZATION STRATEGY
HYBRID MPI + OPENMP + METIS

1. Graph Partitioning (METIS)
    - Minimise communication overhead

2. Inter-Node Parallelism (MPI)
    - Each node handles a subdomain.
    - Scale across distributed systems

3. Intra-Node Parallelism (OpenMP)
    - Multithreaded updates within each subdomain.
    - Dynamic scheduling for load balancing.

GRAPH PARTITIONING (METIS)

BALANCED SUBDOMAINS

- Partition graph into k subdomains of equal vertex/edge weight.
- Example: Use METIS's k-way partitioning for sparse graphs.

MINIMIZE EDGE CUTS

- Fewer edges between partitions → less MPI communication.
- Metric: Edge-cut ratio (lower = better).

INTER-NODE PARALLELISM (MPI)

SUBDOMAIN ASSIGNMENT

- Each MPI process manages one subdomain.
- Data: Local vertices + halo (boundary) vertices.

COMMUNICATION PROTOCOL

- Halo Updates: Sync boundary vertices post-update.
- Non-blocking MPI: Overlap computation/communication.

INTRA-NODE PARALLELISM (OPENMP)

THREAD-LEVEL PARALLELISM

- Each MPI process spawns OpenMP threads.

WORKFLOW:

- Threads process local vertices in parallel.
- Dynamic scheduling for load imbalance (e.g., #pragma omp schedule(dynamic, chunk_size)).

AFFECTED SUBGRAPH HANDLING

- Thread teams focus on disconnected subtrees.

WHY IS THIS EFFECTIVE?

MPI

ROLE :

Cross-node coordination

ADV :

Scales to 1000s of nodes

METIS

ROLE:

Partitioning

ADV:

Balanced loads + minimal communication

OPENMP

ROLE:

Intra-node parallelism

ADV:

Efficient use of multicore resources

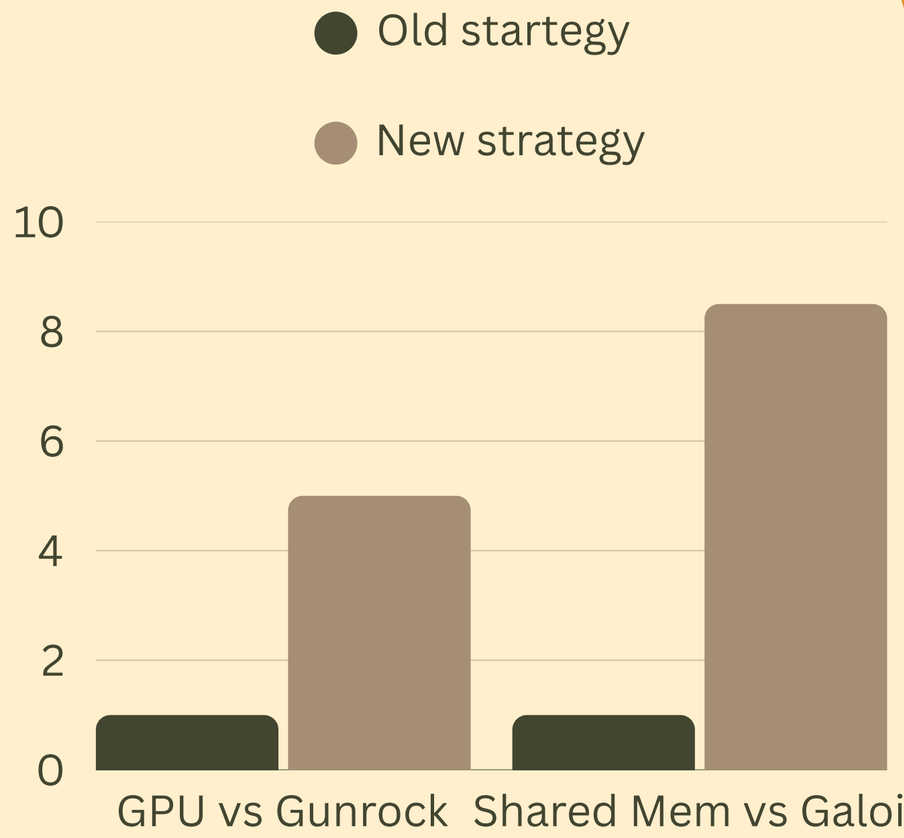
WRAPPING IT UP...

CONCLUSION:

- Framework outperforms recomputation for insert-heavy changes.
- Portable across CPU/GPU architectures.

FUTURE WORK:

- Hybrid recomputation/update strategy.
- Predictive algorithms for change batches.



THANK YOU :)