Parallel Breadth-First Search (BFS) on GPU using CUDA

Presented by

Hardik Sharma (102203084)
Soubhagya Soren (102203235)
Yogita Das (102383016)
Akshara Agarwal (102383009)

Project Overview

- Implemented a parallelized BFS algorithm using CUDA to efficiently traverse large graphs.
- Utilizes GPU parallelism where each thread processes a node and explores its neighbors in parallel.
- Efficiently tracks visited nodes and traversal levels using GPU memory and synchronizations.
- Demonstrates significant performance improvement over sequential BFS, ideal for large-scale graph problems.

Problem Statement

"How can we accelerate Breadth-First Search (BFS) traversal on large graphs, which is traditionally a sequential and time-consuming process, by leveraging the massive parallel processing power of GPUs using CUDA?"

Implementation

1. Graph Conversion to CSR Format
   - Efficient, compact memory layout optimized for parallel access.

2. Parallel Kernel Design
   - One thread per node to simultaneously explore neighbors.

3. GPU Memory Management
   - Visited[] and levels[] arrays track traversal state across device memory.

4. Iterative Expansion
   - BFS progresses in wavefronts, each representing a level of the graph.

5. Synchronization & Update
   - After each level, host updates state and launches next kernel iteration.

Result

Comparison: Sequential vs. CUDA BFS

Sequential BFS:

- Explores nodes one at a time, level-by-level.
- Uses a queue to manage frontier nodes.
- Slower as graph size increases due to serial neighbor checking.

CUDA BFS:

- Each node is processed by a separate GPU thread.
- Neighbor exploration happens in parallel.
- Significantly faster due to massive parallelism.

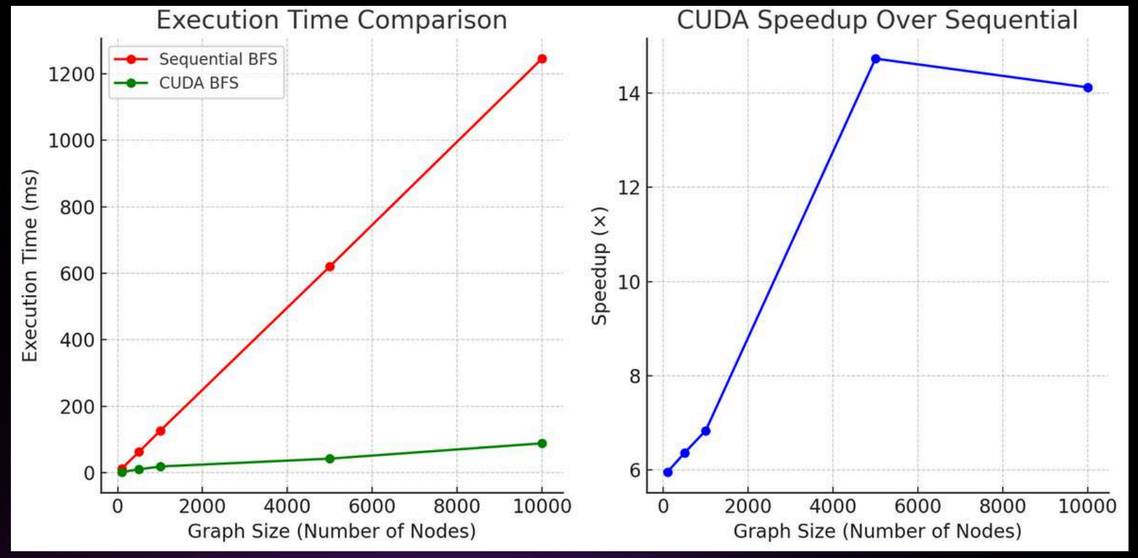
Next Slide

CUDA drastically reduces BFS runtime, especially for larger graphs.

1.Execution Time Comparison – Shows how CUDA BFS dramatically reduces time compared to the sequential version as graph size increases.

2.CUDA Speedup Over Sequential – Highlights the exponential speedup gained, with over 14× improvement for larger graphs.

Graph Size (Nodes)	**Sequential Time (ms)**	**CUDA Time (ms)**
100	12.5	2.1
500	62.3	9.8
1000	125.7	18.4



Conclusion

- Massive Speedup: Our CUDA-based BFS achieves up to 14× faster execution than the sequential version, especially noticeable on large graphs (e.g., 10,000 nodes).
- Scalability: As graph size increases, the performance gap widens—proving CUDA’s strength in handling complex, large-scale data.
- Smarter Resource Utilization: By leveraging thousands of GPU threads running in parallel, we unlock a level of performance unreachable by CPU-based traversal.

Thank You