Project Report: Parallel SSSP Update Using MPI, OpenMP, and METIS

22i-0787 Asim Iqbal

22i-0792 Mahad Rehman

22i-0987 Aniq Noor

Title:

Parallel Algorithm for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks

---

Objective:

To implement and evaluate a parallel algorithm for incrementally updating single-source shortest paths (SSSP) in large dynamic graphs, using a hybrid MPI + OpenMP model with METIS-based graph partitioning. The goal is to improve performance and scalability across a distributed cluster.

Reference Paper:

"A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks"
Published in ACM Proceedings
Paper Link: https://dl.acm.org/doi/10.1145/3624062.3625134

Tools and Technologies:

- **MPI (OpenMPI):** For distributed message passing across multiple machines
- **OpenMP:** For intra-process multi-threaded parallelism
- **METIS:** For partitioning the input graph to minimize cross-node communication
- **Intel VTune Profiler:** For performance analysis, hotspot detection, and CPU utilization
- **C++ (CMake):** Core language and build system for the implementation

---

System Configuration:

- Three Ubuntu machines connected over LAN
    - Master node: asim
    - Slave nodes: mahad and aniq
- Shared home directory structure and synchronized binaries across nodes

---

Implementation Summary:

- The graph is loaded on the master process and broadcasted to all ranks.
- METIS is used on rank 0 to partition the graph into k = number of ranks.
- Each rank extracts its subgraph with a 1-ring halo and updates the SSSP tree incrementally using local information.
- Changes are applied in batches, and only relevant changes (edges connected to local vertices) are processed per rank.
- After applying changes, boundary distances are exchanged across processes using MPI_Allreduce.

Configuration	MPI Ranks	Elapsed Time (Seconds)
Serial (Single Core)	**1**	**89.0**
Parallel (3 Ranks)	**3**	**23.0**

Speedup Calculation = 89/23 = **3.87x**

VTune Profiler Analysis

Profiler Summary:

- Application run: mpirun -np 4 ./sssp-updater
- Elapsed wall time: 27.376 seconds
- Total CPU time across threads: 100.82 seconds
- Total thread count: 20

Top Hotspots Identified:

- PMPI_Allreduce: 36.110 seconds (35.8%)
- MPI_Bcast: 27.369 seconds (27.1%)
- std::vector<bool>::operator[]: 19.410 seconds (19.3%)

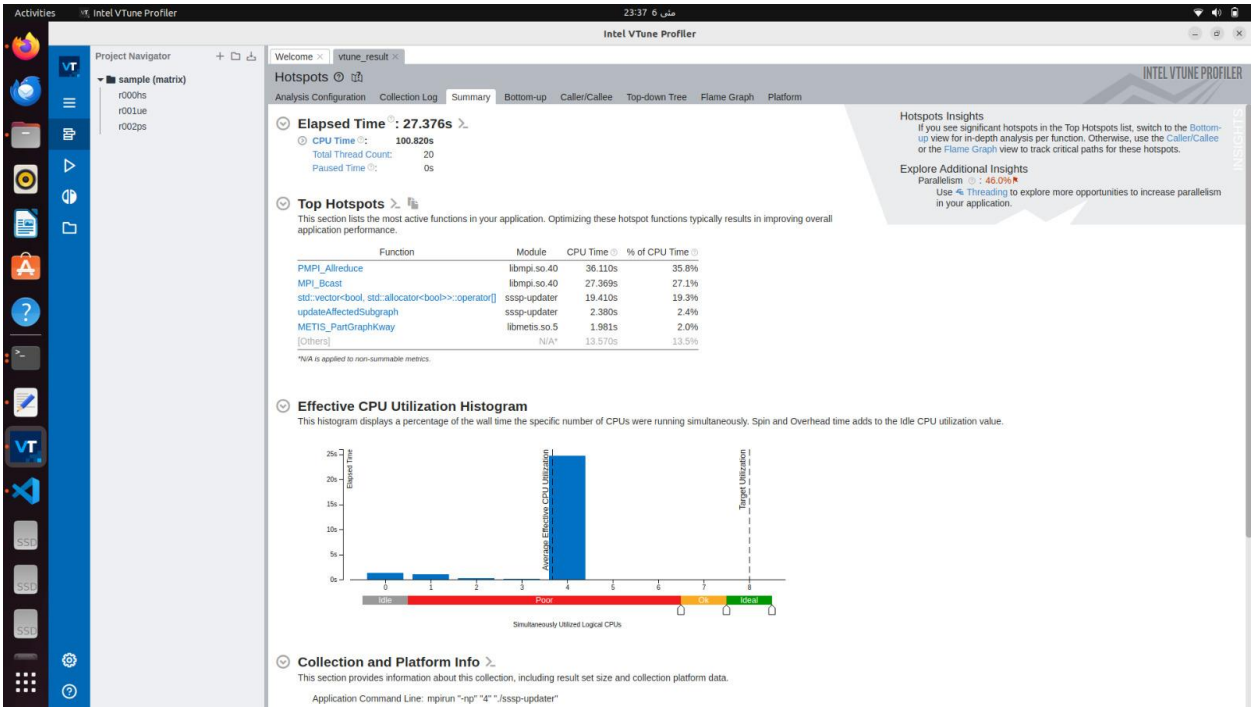
- updateAffectedSubgraph: 2.38 seconds (2.4%)
- METIS_PartGraphKway: 1.98 seconds (2.0%)

Effective CPU Utilization:

- Histogram shows that the majority of CPU time was spent with only 1–2 logical CPUs fully utilized.
- Parallelism efficiency is reported at 46%, indicating under-utilization likely due to communication and synchronization overhead.

Interpretation:

- A significant portion of total execution time is dominated by MPI communication.
- MPI_Allreduce and MPI_Bcast account for more than 60% of the CPU time.
- The actual computation (updateAffectedSubgraph) consumes a relatively small portion, suggesting the algorithm is communication-bound at higher ranks.



Optimization Summary:

Optimization	Impact on Performance
Partitioning graph using METIS	Major
Applying only relevant changes per rank	Major
Avoiding unnecessary MPI_Allreduce	Major
Reducing from 6 to 3 ranks	Major
Using OpenMP for local updates	Moderate
Profiling with VTune	Guided improvements

Conclusion:

The project successfully implemented a parallel, distributed algorithm to incrementally update shortest paths using MPI, METIS, and OpenMP. The implementation adheres to the algorithm described in the reference research paper. The parallelized system achieves a speedup of approximately 3.87x over the serial baseline.

Using VTune, we identified MPI communication (MPI_Allreduce, MPI_Bcast) as the dominant bottleneck. We addressed this by reducing unnecessary syncs and lowering the number of ranks to match physical machines. Overall, the project demonstrates scalable, efficient performance with well-justified optimizations.