Report: Dijkstra's Algorithm with GraphX

Implementation Overview

This project uses Apache Spark GraphX to implement Dijkstra's algorithm for computing the shortest path distances from a source node across a large graph. The input graph is ingested in edge list format, and GraphX's Pregel API is utilized for iterative message passing to update and propagate minimum distances efficiently.

Initially, vertices are assigned a distance of 0 (for the source node) or Infinity (for all other nodes). During each Pregel iteration, vertices update their distances based on neighboring vertices' values plus edge weights.

Performance Analysis

Running on an Azure VM setup with Spark Standalone Mode:

- Dataset: 10,000 nodes and 100,000 edges.
- Observed runtime: ~1-2 minutes, depending on VM resources (Standard_D2s_v3 recommended).
- Bottlenecks primarily observed in network transmission and disk I/O if resources were limited.

GraphX effectively distributed the computations, minimizing driver-side overhead and fully leveraging the worker nodes for parallel distance calculation.

Challenges and Lessons Learned

- Cluster Setup: Setting up and correctly configuring Spark (Master and Workers) on Azure VMs was essential. Advertising the correct external IP in spark-env.sh was crucial to prevent connection failures.
- Pregel Abstraction: Understanding GraphX's Pregel abstraction simplified the Dijkstra logic but required careful initialization and message function definitions.
- Deployment Errors: Common issues such as Connection Refused errors were solved by configuring the correct master IP address and ensuring firewall ports (7077, 8080) were open.

Through this project, we reinforced best practices in distributed graph processing, cluster configuration, and debugging Spark application deployment issues.