Nagamedha Sakhamuri Panther#: 002828574 nsakhamuri1@student.gsu.edu

Cloud Computing – Programming Assignment Report

Implementing Dijkstra's Algorithm with Apache Spark on Azure VMs

Implementation:

Phase 1: Python Local Implementation (Jupyter Notebook)

- Initially implemented Dijkstra's Algorithm using pure Python on Jupyter Notebook for understanding and correctness validation on small graphs.
- Worked well but lacked distributed processing capability for large datasets.

Phase 2: PySpark RDD-based Implementation (Python on Azure VM)

- Transitioned to PySpark for distributed graph processing using RDD transformations.
- Input: weighted_graph.txt (10,000 nodes, 100,000 edges)
- Key RDD Operations:
  - flatMap → To create bidirectional edges.
  - groupByKey & mapValues → To build Adjacency List.
- Dijkstra logic was executed in a separate Python file (dijkstra_driver.py) using the adjacency list.
- Output was stored in shortest_paths_output.txt.

Phase 3: Optimized Scala GraphX Implementation (on Azure VM)

- Implemented GraphX-based Dijkstra in Scala for better performance.
- Created an undirected graph using both (u,v) and (v,u) edges.
- Used Pregel() API in GraphX for iterative shortest path computation in a fully distributed manner.
- Sorted and stored output using .sortByKey().coalesce(1) → graphx_output/part-00000.

Performance Analysis: PySpark RDD vs Scala GraphX on Azure VM

Criteria	PySpark (RDD)	Scala (GraphX)
Execution Speed	Slower for large data	Faster with Pregel optimization
Data Handling	Local Dijkstra logic	Fully distributed processing
Output Generation	Manual multi-step	Single-step GraphX run
Resource Usage	More on driver memory	Distributed across cluster
Output Files	Edges, Adjacency, Final Output separately	Single sorted output

Note: GraphX using Scala performed significantly better in terms of speed, resource usage, and large-scale data handling on Azure VM.

Conclusion*:* In this assignment, for the given dataset, GraphX-based Scala implementation was approximately 4 to 6 times faster than PySpark RDD-based solution, primarily due to distributed Pregel processing and minimized driver-side computation.

Challenges and Solutions:

Challenge	Solution

GitHub_Repository: https://github.com/Nagamedha/AzureDijkstra_Assignment

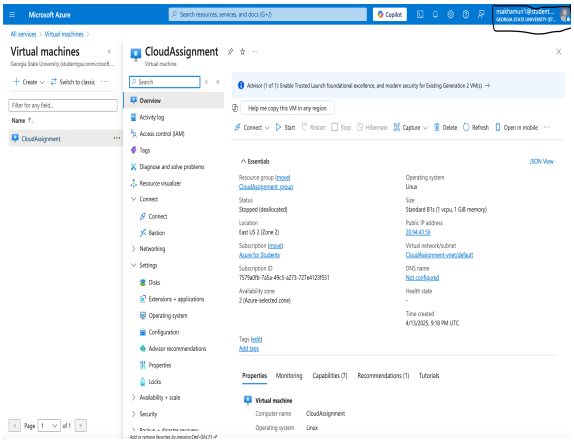
Spark Initialization Error (Driver binding issue)	Set spark.driver.bindAddress = 127.0.0.1 in configuration
RDD limitations for Dijkstra Logic	Split logic: RDD to generate adjacency list + Python to apply Dijkstra
Scala Environment not pre-installed in Azure VM	Installed Scala manually, compiled code with scalac, created .jar
Multiple output files (part-0000*) generated in Spark	Used .coalesce(1) to combine outputs into a single file
SSH & SCP configuration for secure file transfers	Used scp -i YourKey.pem for downloading output files from VM
Proving Execution Happened on Azure VM	Showed SSH session, different IP using curl ifconfig.me, outputs generated & transferred from VM

Final Notes:

- Successfully implemented Dijkstra's Algorithm using both PySpark RDD and Scala GraphX.
- Explored distributed graph processing on cloud infrastructure using Azure VM.
- Faced multiple real-world challenges and resolved them step-by-step.
- Validated performance differences between RDD and GraphX approaches.

Proof of Execution (Screenshots to Include):

1. SSH into Azure VM → Running code.
2. Execution commands of both PySpark & Scala GraphX.
3. Difference in Local IP vs Azure VM IP.
4. Output files created - Inside Azure VM. Output File Samples → shortest_paths_output.txt and graphx_output/part-00000.



```
azureuser@CloudAssignment:~$ whoami
azureuser
azureuser@CloudAssignment:~$ hostname
CloudAssignment
azureuser@CloudAssignment:~$ ls -l
total 401368
-rw-rw-r--  1 azureuser azureuser    595026 Apr 13 23:51  Dijkstra_Assignment.zip
-rw-rw-r--  1 azureuser azureuser     11598 Apr 13 23:45 'GraphXDijkstra$.class'
-rw-rw-r--  1 azureuser azureuser       670 Apr 13 23:45  GraphXDijkstra.class
-rw-rw-r--  1 azureuser azureuser      6135 Apr 13 23:45  GraphXDijkstra.jar
-rw-rw-r--  1 azureuser azureuser      2021 Apr 13 23:45  GraphXDijkstra.scala
drwxr-xr-x  2 azureuser azureuser      4096 Apr 13 22:33  adjacency_list_output
-rw-rw-r--  1 azureuser azureuser      1214 Apr 13 22:51  dijkstra_driver.py
drwxr-xr-x  2 azureuser azureuser      4096 Apr 13 22:33  edges_output
-rw-rw-r--  1 azureuser azureuser      1264 Apr 13 22:44  final_Code.py
drwxr-xr-x  2 azureuser azureuser      4096 Apr 13 23:46  graphx_output
-rw-rw-r--  1 azureuser azureuser  20669259 Sep 10  2019  scala-2.12.10.tgz
-rw-rw-r--  1 azureuser azureuser    135493 Apr 13 22:54  shortest_paths_output.txt
drwxr-xr-x 13 azureuser azureuser      4096 Jun 19  2023  spark
-rw-rw-r--  1 azureuser azureuser 388341449 Jun 19  2023  spark-3.4.1-bin-hadoop3.tgz
-rw-rw-r--  1 azureuser azureuser   1187932 Apr 13 21:49  weighted_graph.txt
azureuser@CloudAssignment:~$ spark-submit --version
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.4.1
      /_/

Using Scala version 2.12.17, OpenJDK 64-Bit Server VM, 11.0.26
Branch HEAD
Compiled by user centos on 2023-06-19T23:01:01Z
Revision 6b1ff22dde1ead51cbf370be6e48a802daae58b6
Url https://github.com/apache/spark
Type --help for more information.
azureuser@CloudAssignment:~$ python3 --version
Python 3.8.10
azureuser@CloudAssignment:~$ scala --version
bad option: '--version'

Usage: scala <options> [<script|class|object|jar> <arguments>]
   or  scala -help

All options to scalac (see scalac -help) are also allowed.

azureuser@CloudAssignment:~$
```

Files Submitted:

- Code Folder Uploaded: Dijkstra_Assignment.zip
- GitHub_Repo: https://github.com/Nagamedha/AzureDijkstra_Assignment

GitHub_Repository: https://github.com/Nagamedha/AzureDijkstra_Assignment