|  |  |  |  |
| --- | --- | --- | --- |
| **Course** | MSc in Data Science | **Lecturer** | Michael Gleeson |
| **Module** | Infrastructure for Big Data | **Student** | Shon Tom (C00313480) |
| **Assignment** | CA 3 | | |

**1. Problem Statement**

It was difficult to move big data from Snowflake – a cloud-based database– to Hadoop’s HDFS for storage then processing. We wanted to design a system that would enable effective, efficient and massive transfer, storage and processing of data.

**Challenges:**

* **Performance**: Handling high-volume data efficiently.
* **Integration**: Ensuring compatibility between Snowflake, Hadoop, and Spark.
* **Scalability**: Maintaining performance as data sizes increase.

**2. Solution Proposal**

To address these challenges, we proposed a big data solution leveraging the Hadoop ecosystem and Spark for data processing. The architecture includes:

* **Snowflake** as the data source for cloud-based data warehousing.
* **HDFS (Hadoop Distributed File System)** for distributed data storage.
* **Apache Spark** for efficient data processing and analysis.

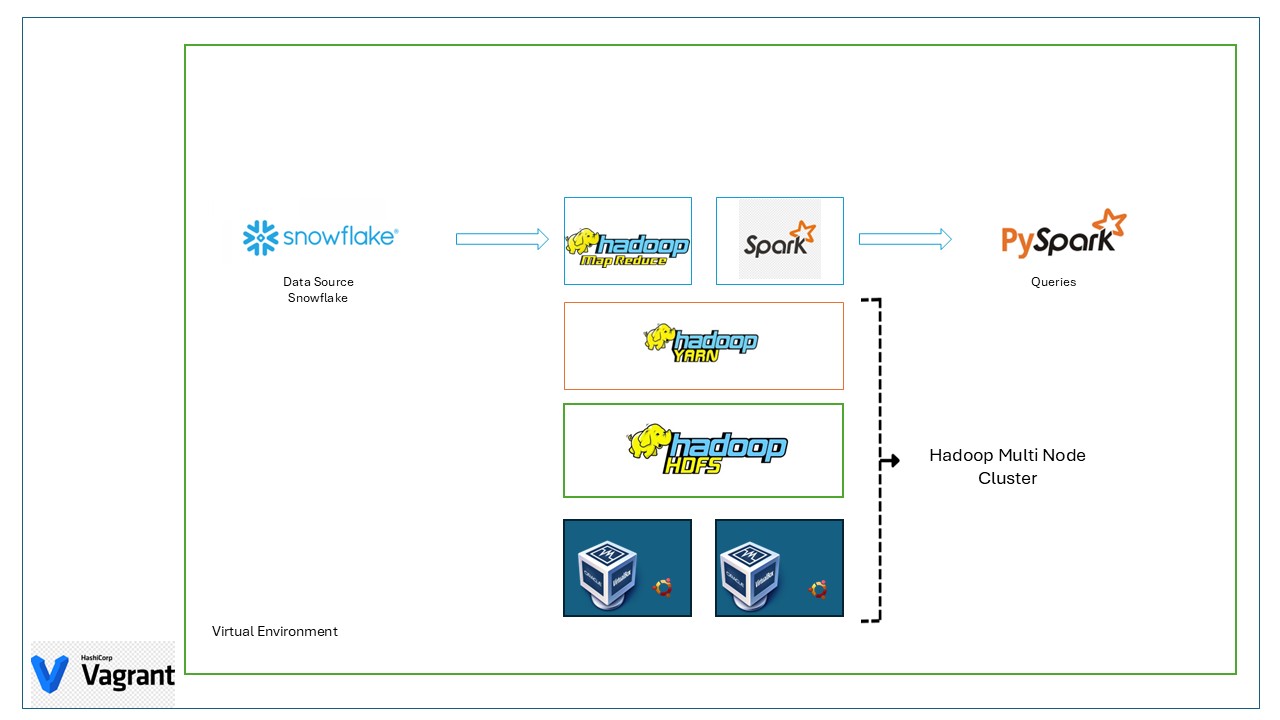
**Workflow Overview:**

A diagram of a computer program

Description automatically generated

1. Extract data from Snowflake and load it into a Spark DataFrame.
2. Write the data to HDFS in a format like CSV or Parquet for storage.
3. Read the data from HDFS using Spark and display or process it for further analysis.

**Proposal Diagram**

****

**3. Prerequisite**

2 Ubuntu Server VMs (for cluster resources)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hostname** | **OS** | **IP address** | **Function** | **Hardware** |
| Node01 | Ubuntu 18.04 | 192.168.50.4 | Master/Name Node | 62GB/ 4096 MB/4 Core |
| Node02 | Ubuntu 18.04 | 192.168.50.5 | Data Node/Slave | 62GB/ 4096 MB/2 Core |

**Pre-downloaded package**

1. Oracle Java openjdk-11

sudo apt install openjdk-11-jdk -y

1. Hadoop 3.4.1

wget <https://downloads.apache.org/hadoop/common/hadoop-3.4.1/hadoop-3.4.1.tar.gz>

1. Spark 3.5.3

wget https://downloads.apache.org/spark/spark-3.5.3/spark-3.5.3-bin-hadoop3.tgz

1. snowflake-jdbc-3.17.0.jar

wget <https://repo1.maven.org/maven2/net/snowflake/snowflake-jdbc/3.17.0/snowflake-jdbc-3.17.0.jar>

1. spark-snowflake\_2.12.jar

wget <https://repo1.maven.org/maven2/net/snowflake/spark-snowflake_2.12/3.0.0/spark-snowflake_2.12-3.0.0.jar>

1. spark-sql\_2.12-3.5.3.jar

wget [https://search.maven.org/remotecontent?filepath=org/apache/spark/spark-sql\_2.12/3.5.3/spark-sql\_2.12-3.5.3.jar -O spark-sql\_2.12-3.5.3.jar](https://search.maven.org/remotecontent?filepath=org/apache/spark/spark-sql_2.12/3.5.3/spark-sql_2.12-3.5.3.jar%20-O%20spark-sql_2.12-3.5.3.jar)

**4. Source Code Tree**

|  |  |
| --- | --- |
| **README.md** | The markdown. Please read this first |
| **Big Data Report C00313480.docx** | File. Report of the BigData CA3 |
| **Onedrive link** | File. The link to the OneDrive |
| **Vagrantfile** | File. To build up virtual machines with configuration |
| **commands used.docx** | File. Contain all scripts to be run in Vagrantfile |

**5. Step-by-Step Implementation**

**Step 1**

* setup vagrant file for creation of 2 VMs
* vagrant up

A screenshot of a computer program

Description automatically generated

**Step 2**

* ssh to VM from vagrant terminal
* check IP, ping nodes, check hostname and hosts files
* Install Java and export PAT
* Install Hadoop, export PATH, configure files, and grant permission to Vagrant user
* Install Spark, export PATH, configure files, and grant permission to Vagrant user
* Hadoop Configuration
* Spark configuration

**Step 3**

* SSH keygen in both nodes
* Format HDFS Name Node before starting HDFS
* Start HDFS
* Start Spark
* Testing the configuration by calculating Pi value

**Step 4**

* To integrate Snowflake, download snowflake jdbc, spark snowflake, spark SQL
* Place these jar files in opt/Spark/jars folder
* Create a .py file with DB details to create the connection

**Step 5**

* To check the working of cluster, add the python code in .py to read the data from snowflake and write the data to HDFS.
* Once its written using python code read the data in spark.

**6. Issues Faced**

* JAR files for connection of spark with snowflake
* Unable to connect to 8088 port due to configuration mistake
* Password less SSH to another node was unable due to not generating key
* Unable to connect to DB because of the user role
* Permission issues for the folders and files

**7. Results**

1. Running processes and their Process IDs

Node01

**A screenshot of a computer

Description automatically generated**

Node02

**A screenshot of a computer

Description automatically generated**

1. Data Nodes

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. Data in Snowflake

**A screenshot of a computer

Description automatically generated**

1. .py file for reading data from Snowflake and saving in csv

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

1. Reading the saved CSV using Spark

A screenshot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated

**8. Conclusion**

The best proof of the proposed solution’s feasibility is presented by the execution of the task of copying large datasets from the Snowflake cloud warehouse to the Hadoop HDFS cluster and further processing with Apache Spark. Thus, we implemented a multi-step process of configuring and integrating different technologies to construct a stable and high-performing data pipeline. It involved installation of Hadoop cluster, configuring Spark and incorporation of Snowflake using right JDBC and Spark libraries.

The existence of the implemented system was confirmed by successfully extracting data from Snowflake, loading it to HDFS, reading and processing it using Spark. In addition to serving as a practical solution for managing large data sets, this workflow established a foundation for additional improvements of data management, synchronization of data transfers and transformations, and the integration with data visualization tools.

**9. Reference**

<https://medium.com/@yogeshdarji/installing-multinode-hadoop-cluster-in-yarn-mode-86741c3df45b>

<https://towardsdatascience.com/assembling-a-personal-data-science-big-data-laboratory-in-a-raspberry-pi-4-or-vms-cluster-e4c5a0473025>

<https://medium.com/@sonia_samipillai/step-by-step-tutorial-to-setup-a-multi-node-cluster-in-hadoop-on-ubuntu-linux-b4b6d376b163>

<https://medium.com/@yogeshdarji/installing-multinode-hadoop-cluster-in-yarn-mode-86741c3df45b>

<https://medium.com/solving-the-human-problem/installing-spark-on-ubuntu-20-on-digital-ocean-in-2020-a7e4b5b65ffb>

<https://medium.com/@kuldeepkumawat195/how-to-install-spark-on-ubuntu-3e2c99778932>

<https://docs.snowflake.com/en/user-guide/spark-connector>

<https://stackoverflow.com/questions/50264423/connection-from-spark-to-snowflake>