Requirements

1. **Hardware Requirements:**
   - **Server:**
     CPU: Quad-core processor (Intel i5/i7 or AMD equivalent)
     RAM: Minimum 16 GB (32 GB recommended for larger datasets) Storage: SSD with at least 100 GB of free space
     Network: High-speed internet connection (1 Gbps recommended)
   - **Client:**
     CPU: Dual-core processor
     RAM: Minimum 4 GB Storage: At least 10 GB of free space
     Network: Reliable internet connection

2. **Software Requirements:**
   - **Operating System:**
     Server: Linux (Ubuntu 20.04 LTS or later, CentOS 7 or later)
     Client: Windows 10 or later, macOS, or Linux Programming
   - **Languages:**
     Python: Version 3.8 or later
   - **Frameworks and Libraries:**
     Flask: Version 2.0 or later
     Flask-CORS: Version 3.0 or later
     Apache Spark: Version 3.0 or later
     PySpark: Version 3.0 or later
     pandas: Version 1.0 or later
     cryptography: Version 3.0 or later
   - **Additional Tools:**
     Git: Version control system for managing code
     Docker: For containerization (optional, but recommended for deployment)
     Node.js and npm: For building and running the React frontend
     React: Version 17.0 or later

3. **Dependencies**
   - **Backend:**
     Flask: A lightweight WSGI web application framework for Python.
     Flask-CORS: A Flask extension for handling Cross-Origin Resource Sharing (CORS).

PySpark: The Python API for Spark, used for data processing and machine learning tasks.
cryptography: A library for encrypting and decrypting data to ensure security.

- **Frontend:**
  React: A JavaScript library for building user interfaces.
  axios: A promise-based HTTP client for making API requests.
  react-dropzone: A React component for handling file uploads via drag and drop.
  react-circular-progressbar: A React component for displaying circular progress bars.

Installation

1. **Install Prerequisites on All Nodes**

   First, update the package lists and install the required software on all nodes (both master and worker nodes).

   sudo apt update
   sudo apt install default-jdk scala git -y

   Verify the installation of Java, Scala, and Git:
   java -version
   javac -version
   scala -version
   git --version

2. **Download and Install Apache Spark**

   Download the Spark binaries and verify the integrity of the downloaded file:

   wget https://dlcdn.apache.org/spark/spark-3.5.3/spark-3.5.3-bin-hadoop3.tgz
   wget https://downloads.apache.org/spark/spark-3.5.3/spark-3.5.3-bin-hadoop3.tgz.sha512
   shasum -a 512 -c spark-3.5.3-bin-hadoop3.tgz.sha512

   Extract the downloaded Spark tarball:
   tar xvf spark-3.5.3-bin-hadoop3.tgz

   Move Spark to the `/opt` directory:
   sudo mv spark-3.5.3-bin-hadoop3 /opt/spark

   Verify Spark installation:
   /opt/spark/bin/spark-shell --version

3. **Configure Environment Variables**

   Add Spark environment variables to the `.bashrc` file:

   echo "export SPARK_HOME=/opt/spark" >> ~/.bashrc
   echo "export PATH=\$PATH:\$SPARK_HOME/bin:\$SPARK_HOME/sbin" >> ~/.bashrc
   source ~/.bashrc

4. **Configure Spark on the Master Node**

   On the master node, edit the `workers` file to list the worker nodes:
   nano $SPARK_HOME/conf/workers

   Add the following worker node entries:

   worker1
   worker2

   Copy the default Spark environment configuration file to the active configuration file:

   cp $SPARK_HOME/conf/spark-env.sh.template $SPARK_HOME/conf/spark-env.sh

   Disable the firewall on the master node:
   sudo ufw disable

   Edit the `spark-env.sh` configuration file:
   nano $SPARK_HOME/conf/spark-env.sh

   Add the following settings:

   export SPARK_MASTER_HOST='master'
   export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:/bin/java::")

   Distribute the updated configuration files to the worker nodes:

   scp -r $SPARK_HOME/conf worker1:$SPARK_HOME/
   scp -r $SPARK_HOME/conf worker2:$SPARK_HOME/

5. **Set Up SSH Key Authentication (Master Node)**

   Generate an SSH key on the master node (if not already done):
   ssh-keygen -t rsa -b 4096 -C "your_email@example.com"

   Copy the SSH public key to the worker nodes:

   ssh-copy-id rit-admin@172.1.44.71
   ssh-copy-id rit-admin@172.1.44.70

   Test the SSH connection to both worker nodes:

   ssh rit-admin@172.1.44.71
   ssh rit-admin@172.1.44.70

6. **Start the Spark Cluster**

   Stop all running Spark processes:
   stop-all.sh

   Start the Spark cluster:
   start-all.sh

   By following these steps, you will have successfully installed and configured Apache Spark on a cluster of nodes, ready to handle large-scale data processing tasks.

1. **Verify NFS (Network File System) Setup**

   Install NFS Server on the Master Node (Server)
   On the master node (the NFS server), update the package list and install the NFS server package:

   sudo apt update
   sudo apt install nfs-kernel-server -y

   Configure NFS Exports
   Edit the `/etc/exports` file to specify which directory to share and set permissions. We will share the `/shared/uploads` directory with all client nodes in the

-

`172.1.44.0/24` subnet:
sudo nano /etc/exports

Add the following line to the file:
/shared/uploads 172.1.44.0/24(rw,sync,no_subtree_check)

Export the Shared Directory
After modifying the `/etc/exports` file, apply the changes by exporting the shared directories:
sudo exportfs -a

Start the NFS Server
Restart the NFS server to apply the changes:
sudo systemctl restart nfs-kernel-server

2. **Install NFS Client on All Other Nodes (Worker Nodes)**

   On each worker node, install the NFS client to enable the node to mount shared directories:

   sudo apt update
   sudo apt install nfs-common -y

   Mount the Shared Directory on All Other Nodes
   On each worker node, create the mount point where the shared directory will be mounted, and mount the directory from the master node (replace `172.1.44.73` with the actual IP address of your master node):

   sudo mkdir -p /shared/uploads
   sudo mount 172.1.44.73:/shared/uploads /shared/uploads

3. **Verify the Mount**

   To verify that the shared directory has been correctly mounted on each worker node, use the following commands:

   df -h
   ls /shared/uploads

Both commands should show the shared `/shared/uploads` directory mounted from the master node.
Ensure Proper Permissions

To make sure the `/shared/uploads` directory is accessible to all users, set the correct permissions on the master node:
sudo chmod 777 /shared/uploads

Test File Accessibility
Create a Test File on the Master Node
On the master node, create a test file to confirm that file sharing works:
echo "test" > /shared/uploads/testfile.txt

Verify the File on All Other Nodes
On each worker node, check if the test file is accessible:
cat /shared/uploads/testfile.txt

You should see the output `test`, confirming that the shared directory is working and the file is accessible from all nodes.

By following these steps, you will have successfully set up NFS to share the `/shared/uploads` directory between the nodes in your Spark cluster. This ensures that all nodes can access the shared data directory, facilitating efficient data processing and management.

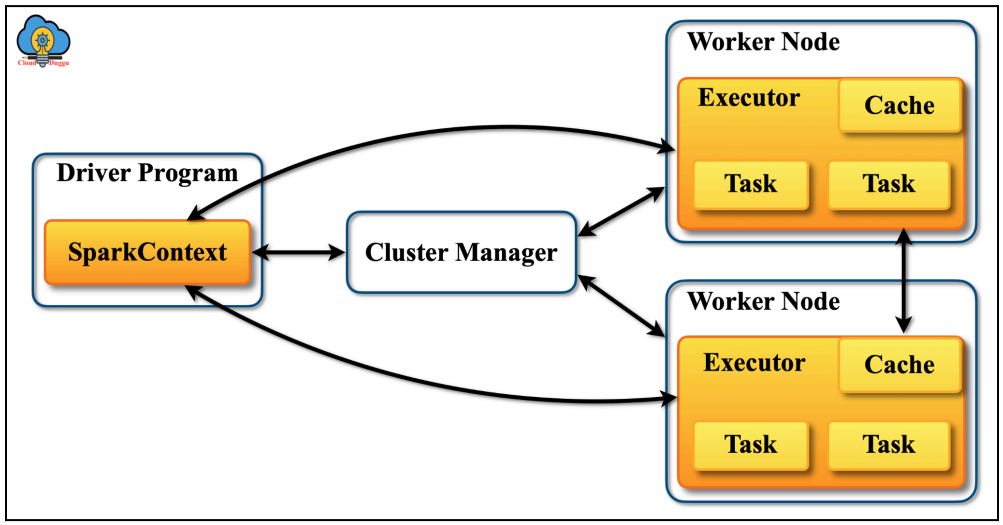

Fig. 1: Spark Cluster Manager

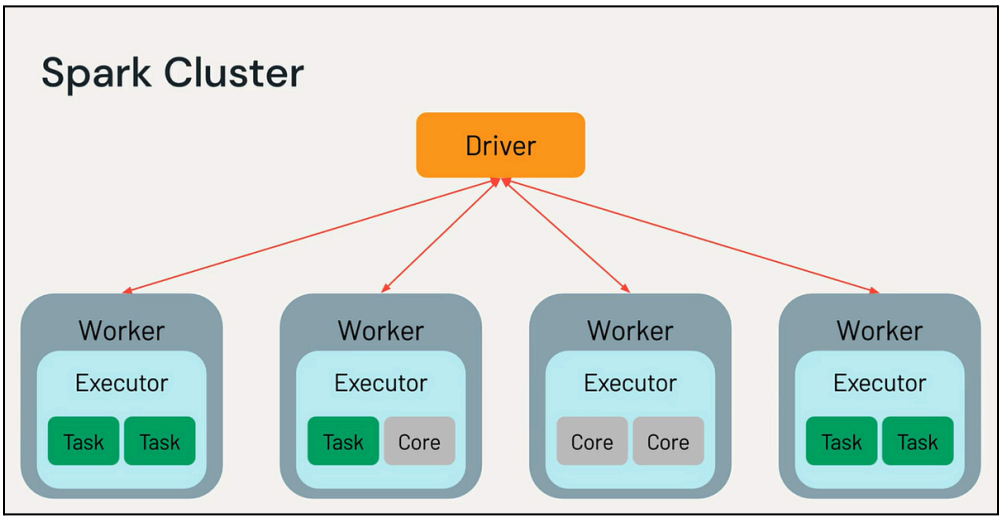

Fig. 2: Spark Cluster Worker And Executor

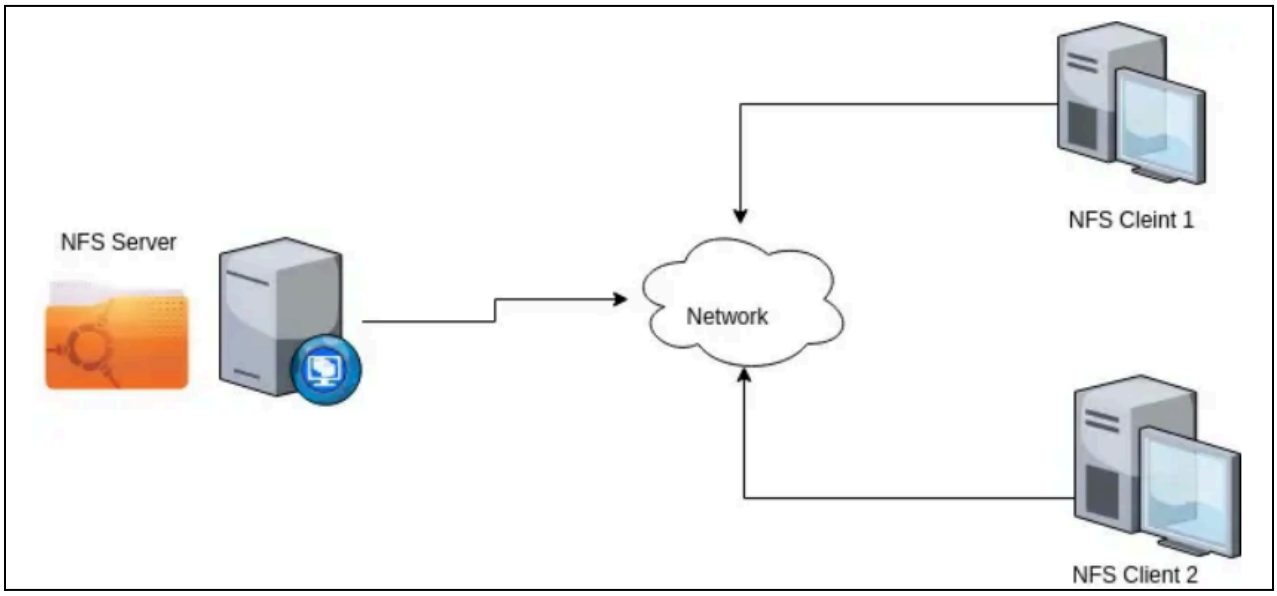

Fig. 3: Spark Cluster Worker And Executor