This document outlines the setup process for a multi-node Hadoop and Spark cluster to support the SBOA-OGRU big data classification workflow, The cluster is configured to process large datasets (e.g., Epsilon, ECBDL14-ROS) efficiently, with Kafka for real-time streaming.

**Technologies Used**

* **Hadoop 3.3.6**: Provides HDFS for distributed storage and YARN for resource management.
* **Apache Spark 3.4.1**: Enables distributed data processing for feature selection (SBOA) and classification (OGRU).
* **Kafka 2.x**: Facilitates real-time data streaming for producer-consumer architecture.
* **Java 8 (OpenJDK)**: Runtime environment for Hadoop and Spark.
* **Python 3.8+**: Used for scripting SBOA, OGRU, and Kafka integration.
* **Jupyter Notebook**: For interactive development and testing of PySpark workflows.
* **Libraries**:
  + pyspark, tensorflow, pyarrow, kafka-python, numpy, scikit-learn
  + Installed via: pip install tensorflow==2.10.0 pyspark==3.3.0 pyarrow==8.0.0 kafka-python==2.0.2 numpy==1.22.0 scikit-learn==1.0.2

**Configuration Details**

The cluster consists of one master node and two slave nodes, configured as follows:

* **Nodes**:
  + Master: Hostname master, username ubuntu
  + Slaves: Hostnames slave1, slave2, username ubuntu
* **HDFS**: Configured at hdfs://master:9000, with data directories at /usr/local/hadoop/data/namenode (master) and /usr/local/hadoop/data/datanode (slaves).
* **YARN**: ResourceManager runs on master:8088.
* **Spark**: Master node runs Spark Master, with workers on slave1 and slave2.
* **Kafka**: Brokers at localhost:9092, topic y\_topic.
* **Environment Variables** (in ~/.bashrc):
* export HADOOP\_HOME=/usr/local/hadoop
* export PATH=$PATH:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin
* export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64
* export PATH=$JAVA\_HOME/bin:$PATH
* export SPARK\_HOME=/usr/local/spark
* export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin
* export PYSPARK\_DRIVER\_PYTHON=jupyter
* export PYSPARK\_DRIVER\_PYTHON\_OPTS="notebook --ip=0.0.0.0 --port=8888 --no-browser"

export PYSPARK\_PYTHON=$(which python3)

**Hadoop Configuration Files**

* **core-site.xml** (/usr/local/hadoop/etc/hadoop/core-site.xml):
* <configuration>
* <property>
* <name>fs.defaultFS</name>
* <value>hdfs://master:9000</value>
* </property>
* <property>
* <name>hadoop.tmp.dir</name>
* <value>/usr/local/hadoop/tmp</value>
* </property>
* <property>
* <name>hadoop.proxyuser.ubuntu.hosts</name>
* <value>\*</value>
* </property>
* <property>
* <name>hadoop.proxyuser.ubuntu.groups</name>
* <value>\*</value>
* </property>

</configuration>

* **hdfs-site.xml** (/usr/local/hadoop/etc/hadoop/hdfs-site.xml):
* <configuration>
* <property>
* <name>dfs.namenode.name.dir</name>
* <value>file:///usr/local/hadoop/data/namenode</value>
* </property>
* <property>
* <name>dfs.datanode.data.dir</name>
* <value>file:///usr/local/hadoop/data/datanode</value>
* </property>
* <property>
* <name>dfs.replication</name>
* <value>2</value>
* </property>
* <property>
* <name>dfs.permissions</name>
* <value>false</value>
* </property>

</configuration>

* **yarn-site.xml** (/usr/local/hadoop/etc/hadoop/yarn-site.xml):
* <configuration>
* <property>
* <name>yarn.resourcemanager.hostname</name>
* <value>master</value>
* </property>
* <property>
* <name>yarn.resourcemanager.webapp.address</name>
* <value>master:8088</value>
* </property>

</configuration>

* **mapred-site.xml** (/usr/local/hadoop/etc/hadoop/mapred-site.xml):
* <configuration>
* <property>
* <name>mapreduce.framework.name</name>
* <value>yarn</value>
* </property>

</configuration>

* **masters** (/usr/local/hadoop/etc/hadoop/masters):

master

* **slaves** (/usr/local/hadoop/etc/hadoop/slaves):
* slave1

slave2

* **hadoop-env.sh** (/usr/local/hadoop/etc/hadoop/hadoop-env.sh):

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

**Spark Configuration Files**

* **spark-env.sh** (/usr/local/spark/conf/spark-env.sh):
* export SPARK\_MASTER\_HOST=master
* export SPARK\_HOME=/usr/local/spark
* export PATH=$SPARK\_HOME/bin:$PATH
* export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64
* export HADOOP\_CONF\_DIR=$SPARK\_HOME/conf

export PYSPARK\_PYTHON=python3

* **workers** (/usr/local/spark/conf/workers):
* slave1

slave2

**Steps to Set Up the Cluster**

The setup is based on a fresh Ubuntu environment (VirtualBox VMs), as described in your March 27, 2025 conversation, with three nodes (1 master, 2 slaves).

**1. Prepare the Environment**

1. **Install Ubuntu**: Install Ubuntu 20.04+ on three VMs (master, slave1, slave2).
2. **Set Hostnames**:
   * On master:

sudo hostnamectl set-hostname master

* + On slave1:

sudo hostnamectl set-hostname slave1

* + On slave2:

sudo hostnamectl set-hostname slave2

1. **Update Hosts File** (/etc/hosts on all nodes):

sudo nano /etc/hosts

Add:

<master-ip> master

<slave1-ip> slave1

<slave2-ip> slave2

1. **Install Dependencies**:

sudo apt update && sudo apt install -y openjdk-8-jdk openssh-server python3-pip

**2. Configure SSH**

1. **Install SSH** (all nodes):

sudo apt install openssh-server -y

1. **Generate SSH Keys** (on master):

ssh-keygen -t rsa -P "" -f ~/.ssh/id\_rsa

1. **Copy Public Key** (from master to all nodes):

cat ~/.ssh/id\_rsa.pub

On each node (master, slave1, slave2), append the public key to ~/.ssh/authorized\_keys:

nano ~/.ssh/authorized\_keys

1. **Test Passwordless SSH**:
2. ssh ubuntu@master
3. ssh ubuntu@slave1

ssh ubuntu@slave2

**3. Install and Configure Hadoop**

1. **Download Hadoop** (all nodes):
2. wget https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
3. tar -xvzf hadoop-3.3.6.tar.gz

sudo mv hadoop-3.3.6 /usr/local/hadoop

1. **Set Environment Variables** (all nodes, in ~/.bashrc):

nano ~/.bashrc

Add:

export HADOOP\_HOME=/usr/local/hadoop

export PATH=$PATH:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

export PATH=$JAVA\_HOME/bin:$PATH

Apply:

source ~/.bashrc

1. **Configure Hadoop Files** (master node):
   * Copy the provided core-site.xml, hdfs-site.xml, yarn-site.xml, mapred-site.xml, masters, and slaves (see Configuration Details).
   * Update hadoop-env.sh:

nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh

Add:

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

1. **Create Data Directories** (all nodes):
2. sudo mkdir -p /usr/local/hadoop/data/namenode
3. sudo mkdir -p /usr/local/hadoop/data/datanode

sudo chown -R ubuntu:ubuntu /usr/local/hadoop

1. **Format Namenode** (master):

hdfs namenode -format

1. **Start Hadoop Services** (master):
2. start-dfs.sh

start-yarn.sh

1. **Verify Services** (master):

jps

Expected output (~6 processes):

* + NameNode, SecondaryNameNode, ResourceManager (master)
  + DataNode, NodeManager (slaves)

**4. Install and Configure Spark**

1. **Download Spark** (all nodes):
2. wget https://downloads.apache.org/spark/spark-3.4.1/spark-3.4.1-bin-hadoop3.tgz
3. tar -xvzf spark-3.4.1-bin-hadoop3.tgz

sudo mv spark-3.4.1-bin-hadoop3 /usr/local/spark

1. **Set Environment Variables** (all nodes, in ~/.bashrc):

nano ~/.bashrc

Add:

export SPARK\_HOME=/usr/local/spark

export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin

export PYSPARK\_DRIVER\_PYTHON=jupyter

export PYSPARK\_DRIVER\_PYTHON\_OPTS="notebook --ip=0.0.0.0 --port=8888 --no-browser"

export PYSPARK\_PYTHON=$(which python3)

Apply:

source ~/.bashrc

1. **Configure Spark Files** (master):
   * Copy spark-env.sh and workers (see Configuration Details).
2. **Start Spark Services** (master):
3. start-master.sh

start-slaves.sh

1. **Verify Services** (master):

jps

Expected output (~8 processes total):

* + Master (master)
  + Worker (slaves)

**5. Install Kafka**

1. **Download Kafka** (master):
2. wget https://downloads.apache.org/kafka/3.6.0/kafka\_2.13-3.6.0.tgz
3. tar -xvzf kafka\_2.13-3.6.0.tgz

sudo mv kafka\_2.13-3.6.0 /usr/local/kafka

1. **Start Zookeeper and Kafka** (master):
2. /usr/local/kafka/bin/zookeeper-server-start.sh /usr/local/kafka/config/zookeeper.properties &

/usr/local/kafka/bin/kafka-server-start.sh /usr/local/kafka/config/server.properties &

1. **Create Topic**:

/usr/local/kafka/bin/kafka-topics.sh --create --topic y\_topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

**6. Install Python Dependencies**

1. **Install Libraries** (master):

pip3 install tensorflow==2.10.0 pyspark==3.3.0 pyarrow==8.0.0 kafka-python==2.0.2 numpy==1.22.0 scikit-learn==1.0.2 jupyter

1. **Configure Jupyter** (master):

pyspark --master yarn

Access Jupyter at http://<master-ip>:8888.

**7. Upload Dataset**

1. **Upload to HDFS** (master):
2. hdfs dfs -mkdir /user/ubuntu/mydir

hdfs dfs -put epsilon\_normalized.t /user/ubuntu/mydir/

**Issues Encountered and Resolutions**

1. **SSH Connection Timed Out** (March 27, 2025):
   * **Issue**: Running start-all.sh resulted in a "connection timed out" error when connecting to the master via SSH.
   * **Resolution**:
     + Verified SSH service: sudo systemctl status ssh
     + Checked firewall: sudo ufw allow 22
     + Ensured correct IP mappings in /etc/hosts.
     + Regenerated SSH keys and copied to authorized\_keys on all nodes.
     + Tested SSH: ssh ubuntu@master, ssh ubuntu@slave1, ssh ubuntu@slave2.
2. **Incomplete HDFS URI** (March 30, 2025):
   * **Issue**: SBOA script failed due to missing host in HDFS path.
   * **Resolution**:
     + Updated dataset path to hdfs://master:9000/user/ubuntu/mydir/epsilon\_normalized.t in featur\_selection2.py.
     + Verified HDFS file existence: hdfs dfs -ls /user/ubuntu/mydir/.
3. **High SBOA Runtime (4-5 hours)** (March 29, 2025):
   * **Issue**: Feature selection took too long, impacting workflow efficiency.
   * **Resolution**:
     + Reduced sample size to 0.5% in featur\_selection2.py.
     + Adjusted SBOA parameters: num\_iterations=10, num\_butterflies=3.
     + Increased Spark parallelism: spark.default.parallelism=100, spark.sql.shuffle.partitions=100.
4. **Kafka Producer Crash** (April 1, 2025):
   * **Issue**: High RAM usage caused s3\_producer.py to crash when loading large Parquet files.
   * **Resolution**:
     + Implemented batch processing with BATCH\_SIZE=1000.
     + Adjusted producer settings: buffer\_memory=33554432, batch\_size=16384, linger\_ms=5.
5. **TensorFlow Model Loading Error** (April 1, 2025):
   * **Issue**: Consumer script (s3\_consumer.py) failed due to a batch\_shape keyword error.
   * **Resolution**:
     + Updated TensorFlow to 2.10.0.
     + Inspected model with h5py to verify structure.
     + Used custom object loading: model = load\_model(model\_path, custom\_objects={'GRU': tf.keras.layers.GRU}).

**Verification**

* **Hadoop**: Access NameNode UI at http://master:9870 and YARN at http://master:8088.
* **Spark**: Access Spark Master UI at http://master:8080.
* **Kafka**: Verify topic: /usr/local/kafka/bin/kafka-topics.sh --list --bootstrap-server localhost:9092.
* **Jupyter**: Run pyspark --master yarn and test SBOA-OGRU workflow.