Lab Sheet: Distributed Hadoop Application

Objective

In this lab, you will set up a distributed Hadoop environment, configure a multi-node cluster using Docker, and deploy a MapReduce job to process sample data. By the end of the lab, you should understand how to run distributed applications on Hadoop.

Prerequisites

1. Basic knowledge of Linux command line and Hadoop.
2. Docker and Docker Compose installed on each machine (or preconfigured in the lab environment).
3. Access to the internet to pull Hadoop images from Docker Hub.

Lab Setup

Cluster Configuration

1. **Master Node**: Manages resources and coordinates tasks.
2. **Worker Nodes**: Execute MapReduce tasks on data blocks.

Required Docker Images

- **Hadoop Base Image**: `bde2020/hadoop-namenode`
- **Hadoop DataNode Image**: `bde2020/hadoop-datanode`
- **Hadoop HistoryServer Image** (for tracking job history): `bde2020/hadoop-historyserver`

Tasks

Task 1: Setting Up the Distributed Hadoop Cluster

1. **Step 1: Prepare the Docker Compose File**
    - Create a `docker-compose.yml` file to define the multi-node Hadoop cluster.

yaml

```yaml
version: '3'

services:

  namenode:

    image: bde2020/hadoop-namenode:latest

    container_name: namenode

    environment:

      - CLUSTER_NAME=testhadoop

      - CORE_CONF_fs_defaultFS=hdfs://namenode:8020

    ports:

      - "9870:9870" # Web UI

      - "9000:9000" # NameNode port

    volumes:

      - namenode-data:/hadoop/dfs/name


  datanode:

    image: bde2020/hadoop-datanode:latest
```

```
    container_name: datanode
    environment:
      - CORE_CONF_fs_defaultFS=hdfs://namenode:8020
    volumes:
      - datanode-data:/hadoop/dfs/data
    depends_on:
      - namenode

  historyserver:
    image: bde2020/hadoop-historyserver:latest
    container_name: historyserver
    depends_on:
      - namenode
      - datanode
    ports:
      - "8188:8188" # Job History server UI
    environment:
      - CORE_CONF_fs_defaultFS=hdfs://namenode:8020

volumes:
  namenode-data:
  datanode-data:
```

2.

3. **Step 2: Deploy the Cluster**

Run the following command to start the Hadoop cluster with Docker Compose:
bash

```
docker-compose up -d
```

   -
4. **Step 3: Verify Cluster Status**
   - Access the NameNode web UI by opening http://localhost:9870.
   - Check that both the NameNode and DataNode are active.

Task 2: Uploading Data to HDFS

1. **Download Sample Data**
   - Use a dataset for testing (e.g., word count dataset).
   - Download or create a simple text file named `sample.txt`.
2. **Upload Data to HDFS**

Use the following command to upload the file to HDFS:
bash

```
docker exec -it namenode hdfs dfs -mkdir -p /input

docker exec -it namenode hdfs dfs -put /path/to/sample.txt /input
```

   -

Verify the upload:
bash

```
docker exec -it namenode hdfs dfs -ls /input
```

   -

Task 3: Running a MapReduce Job

1. **Run the WordCount MapReduce Job**

Hadoop includes a WordCount example. Run it with the following command:
bash

```
docker exec -it namenode hadoop jar
/opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
wordcount /input /output
```

    -

2. **Monitor the Job**
    - You can view job details in the ResourceManager web UI. Visit http://localhost:8088 to check job progress.
3. **View Job Output**

After the job completes, view the output files in HDFS:
bash

```
docker exec -it namenode hdfs dfs -ls /output
```

    -

Download the results:
bash

```
docker exec -it namenode hdfs dfs -cat /output/part-r-00000
```

    -

Task 4: Analyze and Clean Up

1. **Interpret Output Results**
    - Examine the word count results and make a note of any observations.
2. **Clean Up the Cluster**

To stop and remove the containers, use:
bash

```
docker-compose down
```

    -

Task 5: Custom MapReduce Program

1. **Objective**: Develop a custom MapReduce program to analyze a dataset of your choice.
    - **Dataset**: Choose a public dataset (e.g., from Kaggle or UCI Machine Learning Repository).
    - **Program**: Write a MapReduce program to perform analysis on the dataset. For example:
        - Calculate average values (e.g., average temperature per month in a weather dataset).
        - Find the most frequent item (e.g., top product in a retail sales dataset).
    - **Output**: Your output should be clear and informative, and it should be stored in an HDFS directory named `/custom_output`.
2. **Submission**:
    - Push your code to a GitHub repository.
    - Include a README file describing your dataset, analysis goals, and results.
    - Share the link to the GitHub repository.

Task 6: Optimize Hadoop Configuration

1. **Objective**: Experiment with Hadoop configurations to optimize job performance.
    - **Configuration Changes**:
        - Increase the number of mapper and reducer tasks.
        - Adjust memory allocation for tasks.
        - Experiment with block sizes in HDFS.
    - **Testing**: Run the same MapReduce job with different configurations, tracking changes in performance metrics like job completion time.
2. **Documentation**:
    - Record your configuration changes and their impact on performance.
    - Upload the documentation to a GitHub repository with a file named `Optimization_Report.md`.

Questions for Reflection

1. How does Hadoop distribute tasks across the nodes in the cluster?
2. What are the advantages of using a distributed file system (HDFS) for big data applications?
3. How would you configure the cluster to improve performance with a larger dataset?

Additional Resources

- Hadoop Documentation
- MapReduce Programming Guide