- **Show DataFrame**:

```python
df.show()
```

- **SQL Queries**:

```python
df.createOrReplaceTempView("table")
spark.sql("SELECT * FROM table WHERE age > 25").show()
```

- **Create RDD**:

```python
rdd = spark.sparkContext.parallelize([1, 2, 3, 4, 5])
```

Give hadoop notes for freshers interview in detail in more deatils with code

Here's a comprehensive guide for Apache Hadoop, focusing on the key concepts and practical examples that freshers should know for an interview.

---

1. Introduction to Apache Hadoop

- **Apache Hadoop** is an open-source framework that allows for the distributed processing of large datasets across clusters of computers.
- Key **Features** of Hadoop:
  - **Distributed Processing**: Processes data over multiple nodes.
  - **Scalability**: Can easily scale from single servers to thousands of machines.
  - **Fault Tolerance**: Automatically handles failures by replicating data across multiple nodes.
  - **Cost Efficiency**: Uses inexpensive commodity hardware.

---

2. Hadoop Ecosystem Components

- **HDFS (Hadoop Distributed File System)**: A distributed file system that stores data across multiple nodes and handles fault tolerance.
- **MapReduce**: A programming model for processing large datasets with a parallel, distributed algorithm on a cluster.
- **YARN (Yet Another Resource Negotiator)**: A resource management layer that allocates system resources to various applications.
- **Hadoop Common**: Core libraries and utilities required by other Hadoop modules.

Other components of the Hadoop ecosystem include **Hive** (data warehouse for querying), **Pig** (data flow language), **HBase** (NoSQL database), **Oozie** (workflow scheduler), **Zookeeper** (coordination service), **Flume** (data ingestion), and **Sqoop** (data transfer between Hadoop and relational databases).

---

3. Hadoop Architecture

- **Master-Slave Architecture**:
  - **Master Node**: Contains the NameNode and ResourceManager.
  - **Slave Nodes**: Each node has a DataNode and NodeManager to handle data storage and processing tasks.
- **Data Replication**:
  - HDFS replicates data blocks to ensure fault tolerance.
  - By default, each data block has 3 replicas.

---

4. HDFS (Hadoop Distributed File System)

- HDFS is the primary storage system for Hadoop, optimized for storing large files across distributed nodes.

a. HDFS Architecture

- **NameNode**: Manages the metadata, including directory structure, file names, and locations of data blocks.
- **DataNode**: Stores the actual data blocks.

b. Basic HDFS Commands

- **Start HDFS**:

```bash
start-dfs.sh
```

- **List files in an HDFS directory**:

```bash
hdfs dfs -ls /directory_name
```

- **Copy a file from the local filesystem to HDFS**:

```bash
hdfs dfs -put /local_path/file.txt /hdfs_path/
```

- **Copy a file from HDFS to the local filesystem**:

```bash
hdfs dfs -get /hdfs_path/file.txt /local_path/
```

- **View the contents of a file in HDFS**:

```bash
hdfs dfs -cat /hdfs_path/file.txt
```

- **Delete a file in HDFS**:

```bash
hdfs dfs -rm /hdfs_path/file.txt
```

---

5. MapReduce

MapReduce is a programming model used in Hadoop for processing large datasets in parallel by splitting them into independent chunks.

a. MapReduce Process

1. **Map Phase**: Each mapper processes input data and produces intermediate key-value pairs.
2. **Shuffle and Sort Phase**: Sorts and distributes the mapper outputs based on keys.
3. **Reduce Phase**: Each reducer processes key-value pairs from the mappers and produces the final output.

b. MapReduce Example

Let's create a simple WordCount program using MapReduce in Java.

1. **Mapper Class**:

```java
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<Object, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
        String[] words = value.toString().split("\\s+");
        for (String str : words) {
            word.set(str);
            context.write(word, one);
        }
    }
}
```

2. **Reducer Class**:

```java
```

```java
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

3. **Driver Class:**

java

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCountDriver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCountDriver.class);
        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

```
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

4. **Running the MapReduce Job**:

```bash
hadoop jar WordCount.jar WordCountDriver /input_directory /output_directory
```

---

6. YARN (Yet Another Resource Negotiator)

YARN is the resource management layer in Hadoop, allowing multiple applications to share cluster resources effectively.

- **Components of YARN**:
  - **ResourceManager**: Allocates cluster resources.
  - **NodeManager**: Monitors resources on each node and reports to ResourceManager.
  - **ApplicationMaster**: Manages each application and coordinates tasks.

YARN enables Hadoop to run a variety of distributed applications, not just MapReduce.

---

7. Hive

Hive is a data warehousing tool that allows querying and managing large datasets in HDFS using SQL-like syntax, called HiveQL.

a. Hive Key Concepts

- **Tables**: Represent datasets stored in HDFS.
- **Partitions**: Allow faster queries by dividing tables based on column values.
- **Buckets**: Distribute data within each partition for efficient processing.

b. Basic Hive Commands

- **Create a Table:**

```sql
CREATE TABLE employee (
    id INT,
    name STRING,
    salary FLOAT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

- **Load Data into a Table:**

```sql
LOAD DATA INPATH '/path/to/data.csv' INTO TABLE employee;
```

- **Query Data:**

```sql
SELECT name, salary FROM employee WHERE salary > 50000;
```

- **Drop a Table:**

```sql
DROP TABLE employee;
```

---

8. Common Hadoop Interview Questions

1. **Explain Hadoop's master-slave architecture.**
2. **Describe the HDFS and its primary components.**
3. **How does MapReduce work, and what are its phases?**
4. **What are YARN's primary functions?**
5. **Explain the differences between HDFS and a traditional file system.**

6. **What is data replication in HDFS? Why is it important?**
7. **Describe the purpose of Hive and how it differs from SQL databases.**
8. **What is a partition in Hive, and why is it useful?**
9. **Explain the role of ResourceManager in YARN.**
10. **How do you optimize a MapReduce job?**

---

9. Hadoop Command Cheat Sheet

- **Start Hadoop Cluster:**

```bash
start-all.sh
```

- **Stop Hadoop Cluster:**

```bash
stop-all.sh
```

- **List HDFS Directory:**

```bash
hdfs dfs -ls /path
```

- **Copy Local to HDFS:**

```bash
hdfs dfs -put local_file /hdfs_path
```

- **Run MapReduce Job:**

```bash
hadoop jar job.jar MainClass /input /output
```

These Hadoop notes should help you solidify your understanding of the framework and prepare you for a variety of interview questions.

Give NLP notes for freshers interview in detail in more deatils with code

Here's a comprehensive guide to NLP (Natural Language Processing) for freshers, covering essential concepts and practical examples.

---

1. Introduction to NLP

- **Natural Language Processing (NLP)** is a branch of AI that focuses on the interaction between computers and human language.
- NLP combines **linguistics** and **computer science** to enable machines to understand, interpret, and generate human language.
- **Applications**: NLP is used in **chatbots**, **sentiment analysis**, **machine translation**, **speech recognition**, and **text summarization**.

---

2. Key Concepts in NLP

1. **Tokenization**:
   - Splitting text into smaller parts, usually words or sentences.
   - **Word Tokenization**: Splits text into individual words.
   - **Sentence Tokenization**: Splits text into sentences.

```python
from nltk.tokenize import word_tokenize, sent_tokenize

text = "Natural Language Processing is fun! Let's learn more about it."
```