```
db.collection_name.deleteOne({...});
```

These MongoDB notes provide foundational knowledge for freshers, helping them prepare for typical interview questions and scenarios.

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

Apache Spark is a popular distributed data processing framework used for big data applications. Here's a comprehensive Spark guide for freshers, covering important concepts, terminology, and example code in PySpark (Python API for Spark), which is commonly used in interviews.

---

1. Introduction to Apache Spark

- **Apache Spark** is an open-source, distributed computing system that processes large datasets across multiple nodes.
- It supports various programming languages, such as Python (PySpark), Scala, Java, and R.
- **Components** of Spark:
  - **Spark Core**: The foundational engine for processing large datasets.
  - **Spark SQL**: Module for structured data processing using SQL queries.
  - **Spark Streaming**: Real-time data stream processing.
  - **MLlib**: Machine learning library for scalable ML algorithms.
  - **GraphX**: Library for graph and graph-parallel computations.

---

2. Spark Architecture

Spark's architecture consists of the following main components:

- **Driver**: The main program that creates the SparkContext and SparkSession.
- **SparkContext**: An entry point that connects with the cluster manager and allocates resources.
- **Cluster Manager**: Manages the resources for Spark (can be standalone, YARN, or Mesos).
- **Executor**: Processes and executes tasks assigned by the driver.
- **Tasks**: Units of work performed on data partitions by executors.

---

3. Core Spark Concepts

a. RDD (Resilient Distributed Dataset)

- **RDD** is the fundamental data structure in Spark, representing a distributed collection of objects.
- **Features of RDD**:
  - Immutable and distributed.
  - Can be cached in memory.
  - Supports transformations and actions.

b. DataFrames

- **DataFrame** is a distributed collection of data organized into named columns (similar to a table in SQL).
- Built on top of RDDs, DataFrames are optimized and provide SQL-like operations.

c. SparkSession

- **SparkSession** is the entry point for all Spark functionalities.
- **Example**:

```python
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder \
    .appName("MyApp") \
    .getOrCreate()
```

---

4. Spark RDD Operations

RDDs support two types of operations: **transformations** and **actions**.

a. Transformations

Transformations are operations that create a new RDD from an existing one and are lazily evaluated.

- **Example Transformations**:
  - `map()`: Applies a function to each element.

    python

    ```
    rdd = spark.sparkContext.parallelize([1, 2, 3, 4])
    rdd2 = rdd.map(lambda x: x * 2)
    rdd2.collect()  # Output: [2, 4, 6, 8]
    ```

  - `filter()`: Filters elements based on a condition.

    python

    ```
    rdd2 = rdd.filter(lambda x: x % 2 == 0)
    rdd2.collect()  # Output: [2, 4]
    ```

  - `flatMap()`: Similar to `map`, but returns multiple items for each element.

    python

    ```
    rdd = spark.sparkContext.parallelize(["hello world", "hi"])
    rdd2 = rdd.flatMap(lambda x: x.split(" "))
    rdd2.collect()  # Output: ["hello", "world", "hi"]
    ```

b. Actions

Actions trigger the execution of transformations and return a result to the driver.

- **Example Actions:**
  - `collect()`: Returns all elements as a list.

    ```python
    rdd.collect()  # Output: [1, 2, 3, 4]
    ```

  - `count()`: Returns the number of elements.

    ```python
    rdd.count()  # Output: 4
    ```

  - `reduce()`: Aggregates elements using a binary function.

    ```python
    sum = rdd.reduce(lambda x, y: x + y)  # Output: 10
    ```

---

5. DataFrames in Spark

DataFrames are an optimized and easy-to-use API for structured data processing.

a. Creating DataFrames

- **From RDDs:**

  ```python
  rdd = spark.sparkContext.parallelize([(1, "Alice"), (2, "Bob")])
  df = spark.createDataFrame(rdd, schema=["id", "name"])
  df.show()
  ```

- **From Files:**

  ```python
  df = spark.read.csv("data.csv", header=True, inferSchema=True)
  df.show()
  ```

b. Basic DataFrame Operations

- **Show Data:**

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

- **Select Columns:**

```python
df.select("name", "age").show()
```

- **Filter Rows:**

```python
df.filter(df.age > 25).show()
```

- **Group By and Aggregate:**

```python
df.groupBy("department").count().show()
```

---

6. Spark SQL

Spark SQL allows querying of DataFrames using SQL syntax. It's often faster for data manipulation tasks.

a. Using SQL with Spark DataFrames

- **Registering DataFrame as a Temporary Table:**

```python
df.createOrReplaceTempView("people")
```

- **Executing SQL Queries:**

```python
```

```
spark.sql("SELECT name, age FROM people WHERE age > 25").show()
```

b. DataFrame API vs Spark SQL

- DataFrame API and Spark SQL are interchangeable. Many operations done with DataFrame API can be done with SQL syntax and vice-versa.

---

7. Spark Streaming

Spark Streaming processes real-time data streams, such as log data or live sensor data, and performs transformations on the streaming data.

a. Spark Streaming with Socket Text Stream

python

```python
from pyspark.streaming import StreamingContext

# Create StreamingContext
ssc = StreamingContext(spark.sparkContext, 1)

# Connect to a socket stream
lines = ssc.socketTextStream("localhost", 9999)

# Count words in each line
words = lines.flatMap(lambda line: line.split(" "))
pairs = words.map(lambda word: (word, 1))
wordCounts = pairs.reduceByKey(lambda x, y: x + y)

# Print word counts
wordCounts.pprint()

# Start streaming
ssc.start()
ssc.awaitTermination()
```

---

8. Spark MLlib

MLlib is Spark's machine learning library, providing scalable algorithms and tools.

a. Basic Example: Linear Regression

python

```python
from pyspark.ml.regression import LinearRegression
from pyspark.ml.linalg import Vectors
from pyspark.sql import SparkSession

# Prepare Data
data = spark.createDataFrame([
    (1, Vectors.dense([0.0]), 1.0),
    (2, Vectors.dense([1.0]), 3.0),
    (3, Vectors.dense([2.0]), 5.0)
], ["id", "features", "label"])

# Initialize and Fit Model
lr = LinearRegression(featuresCol="features", labelCol="label")
model = lr.fit(data)

# Print coefficients and intercept
print("Coefficients: ", model.coefficients)
print("Intercept: ", model.intercept)
```

---

9. Optimizations in Spark

- **Partitioning**: Controls the distribution of data across executors.

  python

  ```python
  rdd = rdd.repartition(10)  # Increase partitions to 10
  ```

- **Caching**: Caches data in memory to improve performance in iterative algorithms.

  python

  ```python
  rdd.cache()
  ```

- **Broadcast Variables**: Used for sharing a read-only variable across executors.

```python
broadcastVar = spark.sparkContext.broadcast([1, 2, 3])
```

---

10. Common Interview Questions for Spark Freshers

1. **Explain Spark architecture and its components.**
2. **What is an RDD, and how is it different from a DataFrame?**
3. **What are transformations and actions in Spark? Give examples.**
4. **What is lazy evaluation in Spark?**
5. **Explain the concept of partitions and how they improve performance.**
6. **How is Spark SQL different from Hive?**
7. **How does Spark handle memory management?**
8. **What are the use cases of Spark Streaming?**
9. **Explain caching and persistence in Spark.**
10. **Describe the purpose of Spark MLlib and give an example of its use.**

---

11. Spark Commands Cheat Sheet

- **Initialize Spark Session**:

```python
spark = SparkSession.builder.appName("MyApp").getOrCreate()
```

- **Load DataFrame**:

```python
df = spark.read.csv("data.csv", header=True, inferSchema=True)
```

- **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.
