Big Data

Christophe Troalen

July 2024

Disclaimer: These notes were originally created for personal reference during my studies at Imperial College London. As such, they may not be exhaustive or fully accurate, serving mainly as quick notes rather than comprehensive explanations.

1 Introduction to Big Data

Big Data refers to the vast volumes of data that are generated every day from various sources, including social media, sensors, transactions, and more. The challenge lies in processing, analyzing, and deriving meaningful insights from this data. Key technologies for managing Big Data include Hadoop and Spark, which facilitate distributed storage and processing.

2 Hadoop

2.1 Overview

Hadoop is an open-source framework that enables the distributed processing of large datasets across clusters of computers. It comprises two main components: the Hadoop Distributed File System (HDFS) for storage and MapReduce for processing.

2.2 Hadoop Distributed File System (HDFS)

HDFS is designed to store data across multiple machines, providing high fault tolerance and easy scalability. It divides large data blocks into smaller blocks distributed across multiple nodes in a cluster, allowing for parallel processing. This parallelization enables exceeding the storage capacity of any individual machine and allows for failures of machines without compromising the system.

2.3 MapReduce

MapReduce is Hadoop's processing model, which handles data processing by dividing the task into smaller parts. The processing occurs in two phases:

- **Map Phase**: The input data is divided into smaller sub-tasks, processed in parallel across different nodes.
- **Reduce Phase**: The results from the Map phase are combined to produce the final output.

A set of machines running HDFS and MapReduce is called a *Hadoop cluster*, and the individual machines are known as *nodes* or, more precisely, DataNodes. A group of nodes (usually between 30 and 50) connected to the same network switch is also known as a *rack*, and a large Hadoop cluster consists of multiple racks.

The syntax in the HDFS shell is similar to standard Unix commands:

```
hadoop fs -ls # List all files in the user's home directory on HDFS
hadoop fs -mkdir data # Create a directory named 'data'
hadoop fs -cp data/testdata.txt testdata2.txt
```

3 MapReduce

The MapReduce paradigm is inspired by the map and reduce functions in Lisp. The map function takes a <key, value> pair as input and can return one or more <key, value> pairs as output. The shuffle and sort phase is executed across all the <key, value> pairs submitted, allowing keys from across the cluster to be sent to the same reducer. Shuffling refers to transferring <key, value> pairs from the mappers to the reducers, where these pairs are sorted by key.

The shuffle and sort phase is executed automatically in Hadoop MapReduce, making it ideal for sorting large datasets (where the mapper outputs <line, _> and the reducer is the identity function). MapReduce is also useful for searching; the mapper finds matched patterns and outputs <line, _> if matches occur, while the reducer remains the identity function.

When possible, it is recommended to use a combiner, which acts as a mini-reducer applied locally on the output of the map function, reducing the amount of data passed to the reducer.

3.1 Challenges of MapReduce

- **Disk-based Storage:** Hadoop stores data on disks, which can be slower than memory storage, affecting the speed of data processing (high I/O).
- **Batch Processing:** Hadoop processes data in batches, meaning it must complete one batch before starting another. This sequential dependency can lead to delays, as processes need to wait for the completion of preceding tasks.

4 Apache Spark

4.1 Overview

Apache Spark is an open-source distributed computing system that provides an alternative to Hadoop's MapReduce model. It is designed for speed and ease of use, enabling in-memory processing of large datasets.

4.2 Resilient Distributed Datasets (RDDs)

Spark introduces the concept of Resilient Distributed Datasets (RDDs), an abstraction that allows programmers to perform in-memory computations on large clusters in a fault-tolerant manner. RDDs can be cached across computing nodes in a cluster, significantly improving the speed of iterative algorithms and interactive data mining tasks. Spark can be up to 100 times faster than Hadoop when data is stored in memory and up to 10 times faster when accessing disk.

4.3 Key Components of Spark

- **Driver Process:** The driver process runs the `main()` function of the application and coordinates tasks by distributing them to executors.
- **Executors:** Executors are processes that run computations and store data for the application on worker nodes. Each executor runs multiple tasks in multiple threads.
- **Spark Session:** The entry point for programming Spark applications. It is created by the Spark driver and allows the user to interact programmatically with the Spark cluster. Through a Spark session, DataFrames can be manipulated, which are distributed collections of data organized into named columns and partitioned across the cluster.

4.4 DataFrame Partitions

DataFrames in Spark are distributed across different machines in a cluster. Each machine handles a slice or "partition" of the DataFrame, allowing for distributed data processing and fault tolerance.

Spark can be launched with the `pyspark` command, which loads a SparkContext `sc` and SparkSession `spark` within the environment. SparkSession objects represent the entry point for programming with DataFrames.

```
import pyspark
sc = pyspark.SparkContext(appName='appName')
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('appName').getOrCreate()
```

Actions return values to the driver after running a computation or export data to storage. The command `take(n)` is an action that returns the n top rows from a single data partition. The command `reduce(func)` reduces the elements of an RDD using a function `func` with two arguments. The command `count()` returns the number of elements in the RDD.

```
numbers.reduce(lambda x,y: x+y)
poem.count()
set.foreach(lambda x: print(x))
```

Typically, DataFrames in PySpark are created using a SparkSession in two ways:

- From an existing RDD, using `spark.createDataFrame`, where `spark` is the default SparkSession in PySpark.
- Importing data from data sources in persistent memory.

4.5 Machine Learning with Spark

ML is Spark's machine learning library, which operates exclusively on DataFrames. Its predecessor, MLlib, only operates on RDDs.

ML has three main abstract classes:

- **Transformer:** Used to preprocess data.
- **Estimator:** Represents statistical models fitted on observed data.
- **Pipeline:** Provides an end-to-end combination of transformer and estimator for a complete framework of data analysis.