Introduction to Spark

Agenda

- What is Spark?
- What is Hadoop MapReduce?
- The Spark Ecosystem.
- Resilient Distributed Datasets (RDDs).
- Useful Features of Spark.
- Differences between Spark and Hadoop MapReduce.
- Word Count Demo.

What is Spark?

- It's a distributed cluster-computing software framework.
- It uses the Hadoop MapReduce distributed computing framework as its foundation.
- Spark was intended to improve on several aspects of the MapReduce project, such as performance and ease of use while preserving many of MapReduce's benefits.
- Includes a core data processing engine, as well as libraries for SQL, machine learning, and stream processing.

What is Hadoop MapReduce?

- MapReduce is a Java-based, distributed execution framework within the Apache Hadoop Ecosystem.
- It takes away the complexity of distributed programming by exposing two processing steps that developers implement: Map and Reduce
- Splits big data processing tasks into smaller ones, distributes the small tasks across different nodes, then runs each task.

The Spark Ecosystem

The Spark ecosystem consists of five primary modules:

1. **Spark Core:** Underlying execution engine that schedules and dispatches tasks and coordinates input and output (I/O) operations.

2. **Spark SQL:** Gathers information about structured data to enable users to optimize structured data processing.

   sql operations on distributed db

3. **Spark Streaming and Structured Streaming:** Both add stream processing capabilities. Spark Streaming takes data from different streaming sources and divides it into micro-batches for a continuous stream. Structured Streaming, built on Spark SQL, reduces latency and simplifies programming.

   lma ya5od batches so8yra hat7s eno hoa 484al m3 stream

4. **Machine Learning Library (MLlib):** A set of machine learning algorithms for scalability plus tools for feature selection and building ML pipelines. The primary API for MLlib is DataFrames, which provides uniformity across different programming languages like Java, Scala and Python.

5. **GraphX:** User-friendly computation engine that enables interactive building, modification and analysis of scalable, graph-structured data.

Resilient Distributed Datasets (RDD)

- The fundamental data structure of Apache Spark.

- Immutable collection of objects which computes on the different node of the cluster.
  ya3ni read only

- Each and every dataset in Spark RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster.

Resilient Distributed Datasets (RDD)

- Decomposing the name RDD:
  - **Resilient**, i.e. fault-tolerant with the help of RDD lineage graph(DAG) and so able to recompute missing or damaged partitions due to node failures. al DAG da feh al operation ali at3mlt 3la al RDD 34an ywsl le al 4kl ali nta 3aizo fa hybwa by-keep track of those operations 34an lo 7sl failure a3rf agib al RDD de tani by appluing those operations
  - **Distributed**, since Data resides on multiple nodes.
  - **Dataset** represents records of the data you work with. The user can load the data set externally which can be either JSON file, CSV file, text file or database via JDBC with no specific data structure.

Features of RDDs

1. In-memory Computation
Spark RDDs have a provision of in-memory computation. It stores intermediate results in distributed memory(RAM) instead of stable storage(disk).

2. Fault Tolerance
Spark RDDs are fault tolerant as they track data lineage information to rebuild lost data automatically on failure. They rebuild lost data on failure using lineage, each RDD remembers how it was created from other datasets (by transformations like a map, join or groupBy) to recreate itself. Follow this guide for the deep study of RDD Fault Tolerance.

3. Partitioning
Partitioning is the fundamental unit of parallelism in Spark RDD. Each partition is one logical division of data which is mutable. One can create a partition through some transformations on existing partitions.

4. Persistence
Users can state which RDDs they will reuse and choose a storage strategy for them (e.g., in-memory storage or on Disk).

RDDs Operations

- RDDs support two types of operations:
  - Transformations are operations (such as map, filter, join, union, and so on) that are performed on an RDD and which yield a new RDD containing the result.

    in: RDD out: RDD

  - Actions are operations (such as reduce, count, first, and so on) that return a value after running a computation on an RDD.

    in: RDD out: val

Useful Features of Spark

- Provides APIs in Scala, Java, and Python, with support for other languages (such as R) on the way
- Integrates well with the Hadoop ecosystem and data sources (HDFS, Amazon S3, Hive, HBase, Cassandra, etc.)
- Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
- Advanced Analytics – Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

Hadoop MapReduce Vs Spark

1. **Performance:** Spark is faster because it uses random access memory (RAM) instead of reading and writing intermediate data to disks. Hadoop stores data on multiple sources and processes it in batches via MapReduce.

2. **Cost:** Hadoop runs at a lower cost since it relies on any disk storage type for data processing. Spark runs at a higher cost because it relies on in-memory computations for real-time data processing, which requires it to use high quantities of RAM to spin up nodes.

3. **Processing:** Though both platforms process data in a distributed environment, Hadoop is ideal for batch processing and linear data processing. Spark is ideal for real-time processing and processing live unstructured data streams.

Hadoop MapReduce Vs Spark

1. **Scalability**: When data volume rapidly grows, Hadoop quickly scales to accommodate the demand via Hadoop Distributed File System (HDFS). In turn, Spark relies on the fault tolerant HDFS for large volumes of data.

2. **Security**: Spark enhances security with authentication via shared secret or event logging, whereas Hadoop uses multiple authentication and access control methods. Though, overall, Hadoop is more secure, Spark can integrate with Hadoop to reach a higher security level.

3. **Machine learning (ML)**: Spark is the superior platform in this category because it includes MLlib, which performs iterative in-memory ML computations. It also includes tools that perform regression, classification, persistence, pipeline construction, evaluation, etc.

References

- https://www.databricks.com/glossary/mapreduce
- https://www.ibm.com/cloud/blog/hadoop-vs-spark
- https://www.toptal.com/spark/introduction-to-apache-spark
- https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm
- https://towardsdatascience.com/1-introduction-to-apache-spark-299db7a4b68d
- https://data-flair.training/blogs/spark-rdd-tutorial/