Lecture Notes

Apache Spark

In this module, you grasped the concepts of another big data analytics framework called Apache Spark. It is an open source cluster-computing framework. It is known mainly for its high-speed cluster computing capabilities and support for programming languages such as Python, R, Java, SQL, Scala etc.

Fundamentals of Spark

In this session, you first saw an **overview** of Spark. Then you compared it with MapReduce. Subsequently, you studied RDDs, In-memory processing, transformation & actions.

Some features of Spark

- Outperforms MapReduce by a large margin, especially when the scale of data reaches more than a few hundreds of gigabytes.
- An open-source project, and thus has active contributions from all around the world. You can contribute to the Spark project too.
- Can be accessed using any of the standard languages in a data scientist's toolkit (R, SQL, Python)
- Capable of connecting to a variety of data sources like S3, Cassandra, HDFS, HBase, Avro, Parquet, etc.

Libraries that are bundled with Spark:

- SparkSQL
- ML-Lib
- Spark Streaming (out of scope)
- GraphX (out of scope)

Spark is built to optimise for:

- Speed
- Ease of use
- Sophisticated Analyses

Next, we studied **Resilient Distributed Dataset**. An RDD is the primary structure on which all Spark operations are performed at the level of the computation engine.

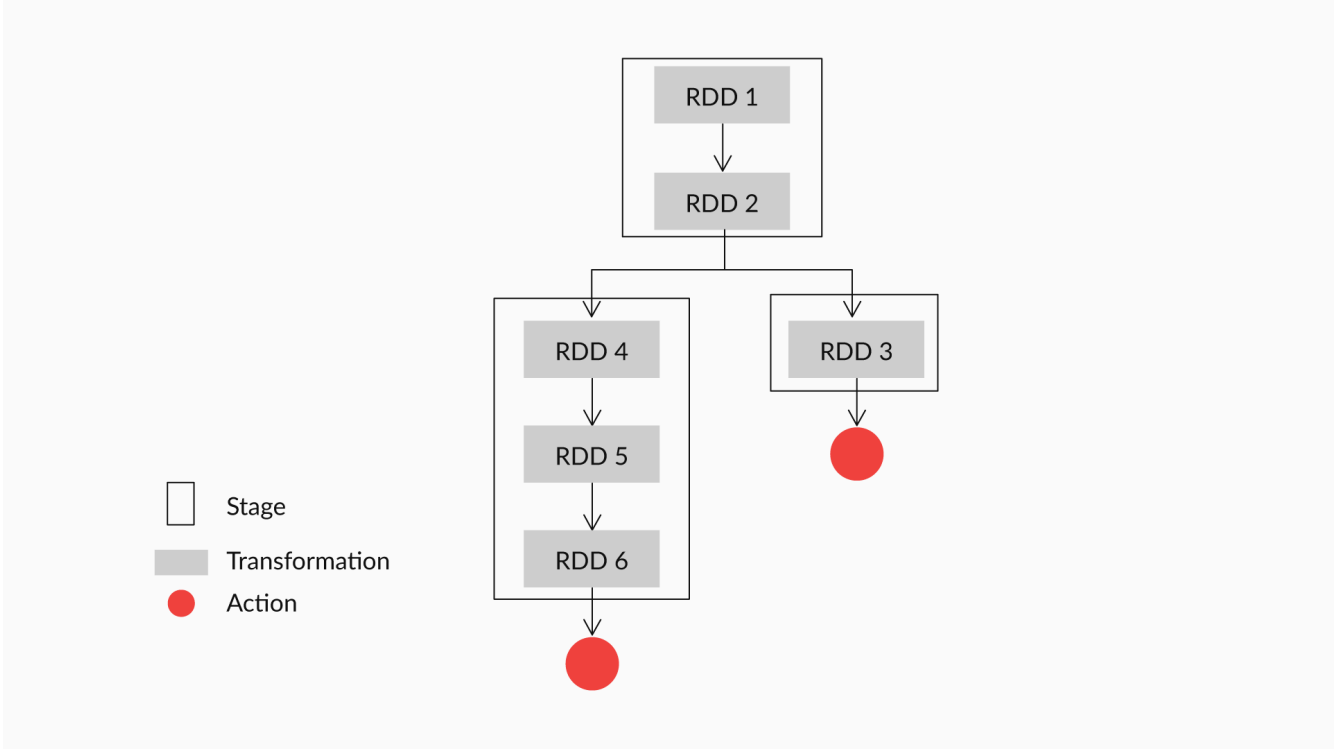
Since Spark 2.0, RDDs are no longer used directly with SparkR. The main data object we work with is a Spark DataFrame. RDDs only work behind the scenes.

Properties of RDDs

- These are the primary data structure of Spark
- They are fault-tolerant, and they do not change the data in the original source
- Their operations are divided into "transformations" and "actions". Transformations get lined up in a graph, and only when an action is called, is a transformation triggered.

RDDs perform **in-memory computing** – all the storage and transaction happens on the main memory. To deal with the volatility of the main memory, RDDs come with the support of Directed Acyclic Graphs (DAGs) to maintain the lineage of data. Hence, data lost due to power outage or memory failure can be recovered.

Following is an example graph of how the **lineage of an RDD** is maintained using a **Directed Acyclic Graph.**



RDD Operations are of two types:

- **Transformations**: They return another RDD
- **Actions**: They either return a numerical value, or save the data to disk

Spark possesses an important property called "**lazy evaluation**". This essentially means that until an action is called, no transformation is evaluated. In the diagram given above, RDD2 and RDD3 won't be computed unless the action on the right is called.

Some transformations:

- map()
- flatMap()
- groupBy()
- filter()

Some actions:

- reduce()
- count()
- reduceByKey()
- countByValue()
- collect()

Working with Spark

In this session, we first compared Spark with Hive. A query that took close to 17 seconds on Hive, ran within a couple of seconds on Spark. This difference becomes more and more significant as the data size grows.

Subsequently, we looked at some analysis on Spark. Specifically, we dealt with the following types of analysis:

- Reading and examining data
- Plotting data
- Grouping and aggregating data
- Model-building

These steps typically form the basis of any data analysis. In this session, we saw how to perform these analyses for data sets whose size cannot be handled by our local R system.

The SparkR library has some useful functions. Some important functions used in the hands-on analysis:

- read.df() : Reads data into a Spark DataFrame
- select() : Selects specific columns
- filter() : Filters with a certain condition
- groupBy() : Performs aggregation over certain categorical variables

Building machine learning models in SparkR is not very different from doing the same in R - you use **training data** in the form of a SparkR DataFrame to **build a model**, and then you **test your model** on a testing or validation dataset. Under the hood, however, SparkR uses the **MLLib** library to train & evaluate models. Also, the **syntax** of SparkR is slightly different.

We also learnt some broad principles in this session about the behaviour of Spark's APIs

- The basic activities of any analysis - reading, examining, plotting, grouping, filtering etc. can be performed on a huge data set in Spark, **without the need of taking subsets**.
- SparkR comes with a very useful **SQL functionality**. As a data scientist, this is an invaluable addition to your toolkit. Now, you can run R and SQL from the same environment, on an immensely powerful computation engine.
- Spark DataFrames work on the **same basic principles as an R data.frame**, but the **nuances of the syntax are different**.

----------THE END----------