Spark Overview

Apache Spark: Short overview

- Apache Spark is a tool for processing large amounts of data.
- It does this efficiently across a large number of machines, or nodes.
- It is mostly used for analytics.

Apache Spark: Short overview (cont.)

- Main advantage: **speed**.
- It is quite fast with large datasets.
    - Large = larger than what a single-server system can work with.
- Up to 100x faster than Hadoop/MapReduce.

Apache Spark: Short overview (cont.)

- Easily scalable: if you need more power, just add more nodes.
- Easy to use, with APIs to R and Python.
- SQL and DataFrame libraries that are useful for basic exploratory analysis and ETL tasks.

Apache Spark: Short overview (cont.)

- Support for multiple file types: Apache Parquet, ORC, JSON.
- Open Source: maintained since 2013 by the Apache Software Foundation.

Databricks: Managed Spark

- Cloud service build by the people who created Spark in 2009.
- Removes all friction to help you get started quickly.
- Drawback: only available in the cloud (Amazon, Microsoft and Databricks).

Spark architecture

Clustering

- A **cluster** is a set of machines, commonly referred as **nodes**, working together.
- Large servers with hundreds of cores are usually more expensive than smaller machines.
- Clusters are more resilient: nodes are easily replaceable.
- Spark can run on a single node, but it would not bring any significant benefit.
- Massively parallel processing: comercial solutions like Teradata have a tight coupling between hardware and software. Spark is loosely coupled.

Processing

- Master/slave architecture.
- The **driver** node is the controller and the head.
  - It keeps the status and state of everything in the cluster.
  - It also sends tasks for execution.

Processing (cont.)

- When the job is being sent to the driver, a `SparkContext` is initiated.
- A `SparkContext` is the way your code communicates with Spark.
- The software in the driver node creates an execution plan (called Directed Acyclic Graph, DAG) and figures what resources it needs.
- Tasks are sent from the driver to the executors.
- The executors communicate back to the driver once they are done. If successful, they are freed by the driver.

RDD

- The core data structure is called **Resilient Distributed Dataset** or RDD.
- RDDs are stored by record (without a schema).
- Typically not the way to work with data in Spark: one uses datasets and DataFrames.

DataFrame

- Similar to tables in a database.
- We can define a schema and make sure that the data will respect it.
- They are similar to R or Pandas dataframes, but not the same.
- Spark uses the same code for RDDs than for DataFrames internally.

Data Processing

- RDDs support two things: **transformations** and **actions**.
- Spark does lazy evaluation: builds a map of everything that will be required, but does not execute it.

Example

```
# Not executed
df = spark.sql('select * from sales')
df.select('country', 'sales')
.filter(df['region'] == 'EU')
.groupBy('country')
```

```
# Executed
display(df)
df.count()
df.write.parquet('/tmp/p.parquet')
```

Actions

- Actions return non-RDD results, like data to the driver or to storage.
- Transformations map RDDs to RDDs.
- Lazy evaluation makes optimization possible!

Storage

- DBFS (Databricks File System)
- Azure Blob Storage

Components on top of Spark Core

- Spark MLlib
  - Machine learning library optimized for Spark.
- Spark GraphX
  - Graph data (social networks, logistics).
- Spark SQL
  - Run SQL queries in Spark in a similar way as in an SQL database.

Hands-on Time

- Lab 1 - Getting Started with Spark in the `labs/` folder.