Big Data Technologies

Agenda

- Apache Spark

Apache Spark

- Spark is Distributed computing framework, that can process huge amount of data.
- Spark can be used as eco-system of Hadoop or can be used as independent distributed computing framework.
- Developed by UCB AMPlabs division in 2009.
    - UCB = University of California Berkley
    - AMP = Algorithms, Machines and People
- Further developed/maintained by DataBricks.
- Open sourced under Apache License in 2010.
- Popular Spark vendors
    - DataBricks
    - AWS EMR
    - Cloudera
    - MapR
- Spark Toolkit
    - Low-level APIs = RDD & DAG
    - High-level APIs = Dataframes
    - Spark SQL + Spark Streaming + Spark ML + Spark GraphX
    - Programming Languages: Scala, Python, R, Java
- Spark Philosophy
    - Unified
        - Same APIs in all available programming Languages
        - Similar performance in any Language
    - Compute Engine

    - Only distributed computing
    - Works with any storage e.g. HDFS, S3, Azure FS, Local FS, etc.
  - Libraries
    - Community given third-party packages
    - https://spark-packages.org/

Hadoop vs Spark

- Hadoop
  - Distributed framework = Distributed storage + Distributed computing
  - Hadoop is developed in Java (JVM based).
  - Designed for commodity hardware.
  - Data is processed in RAM and spills on disk.
  - In MapReduce job, mappers & reducers are executed as independent JVM processes.
- Spark
  - Distributed framework = Distributed computing
  - Not tied up with particular storage.
  - Spark is developed in Scala (JVM based).
  - Needs better hardware config.
  - Data is processed fully in RAM to achieve faster execution.
  - In Spark job, tasks are executed as threads in Executor process.

Spark - Structured/High Level API

- Spark Structured API is abstraction on Lower level concepts (RDD & DAG).
  - Dataframes = Abstraction on RDD - Like in-memory tables (row-col)/pandas dataframes
  - Dataset = Collection of Immutable objects - Only in Scala/Java.
- Dataframe Example

```
df = spark.read.csv('/path/to/csv')
df.printSchema()
df.show()
```

SparkSession

- Wrapper on SparkContext. It can encapsulate additional contexts as needed e.g. SQLContext, HiveContext, StreamingContext, ...
- Spark 2.4 deprecated SparkContext.
- SparkSession is singleton i.e. one application will have single SparkSession.
- It is created using builder design pattern.

```
spark = SparkSession.builder\
            .appName('app-name')\
            .master('local[*]')\
            .config('key', 'value')\
            .getOrCreate()
```

- SparkSession (and SparkContext) is pre-created in Spark shell.

Data Frames

- Created using SparkSession. Typically using Spark "DataframeReader" i.e. "spark.read".
- Abstraction/wrapper on RDD.
- Similar to Pandas dataframes or R dataframes or RDBMS table - in memory.
- Dataframe have structure (metadata) and rows & columns (data).
  - df.printSchema() -- print structure
  - df.show() -- displays rows
- The operations on dataframes is similar to SQL operations e.g. select(), groupBy(), orderBy(), limit(), where(), join(), ...

Spark Data Types

- Similar to Hive types
- https://spark.apache.org/docs/latest/sql-ref-datatypes.html
- `from pyspark.sql.types import *`

- Numeric types
    - ByteType
    - ShortType
    - IntegerType
    - LongType
    - FloatType
    - DoubleType
    - DecimalType
- String types
    - StringType
    - VarcharType
    - CharType
- Binary type
    - BinaryType
- Boolean type
    - BooleanType
- Datetime types
    - DateType
    - TimestampType
- Complex types
    - ArrayType
    - MapType
    - StructType