Distributed Data Processing

with *Spark*

Apache *Spark*

- An engine for performing **large-scale data processing**
- Written in Scala
- APIs for Scala, Python, R and Java
- Large-scale means either or both
  - **Memory capacity** is not enough to hold data on a single machine
  - **Processing capacity** is not enough to process data on a single machine
  - Should process in parallel on multiples machines

Concurrency vs. Parallelism

- **Concurrency**
  - Multiples independent computations occurring simultaneously
- **Parallelism**
  - A single computation performed over multiple CPU, cores, machines...
  - **Split** the dataset into multiple **partitions** over multiples **nodes**
  - **Process** each partitions in parallel and produce a partial result for each
  - **Merge** all partial results and produce a final result
  - Split and merge should be extremely efficient not to destroy parallelism gains

Spark Core and RDDs

RDD

- **RDD** stands for Resilient Distributed Dataset
  - Somehow very large distributed collections
  - Allowing parallel processing
  - Retrying in case of processing failure

RDD Transformations and Actions

- **Transformations** allow to
  - Transform an RDD into another RDD (`map`, `flatMap`, `filter`, `sortBy`, `reduceByKey`...)
  - Combine RDDs to form an RDD (`join`, `leftOuterJoin`, `cogroup`...)
  - Does not perform any processing
  - Only describes a processing
- Only **Actions** trigger processing
  - Execute the **lineage** of RDDs every time
  - Unless explicit caching

Driver Program and RDDs

- **Driver program** is the main program
- Describes the computation (transformations)
- Triggers actual processing (actions)
  - Every action triggers a **job**
  - A job consists of multiples **stages** (that may execute concurrently)
  - Each stage is executed over multiples **tasks** in parallel each operating on a **partition** of the data
  - Before initial stages and between stages has to be be split into multiple partitions using a **partitioner**

RDD Demo

Spark SQL and Datasets

Datasets

- Distributed Datasets consisting of **rows**
- Every dataset has a **schema**
  - Names and types of columns
  - Can be hierarchical structure
- Similarly to **RDDs**
  - Datasets can be transformed and combined to form other datasets
  - Only actions trigger processing
- Looks very similar to **SQL** except that:
  - Queries are expressed with a DSL
  - Much more composable

Schema Allows Optimization

- *Spark* is aware of the structure of the data
- Can optimize **physical execution plan** of queries with the *Catalyst* optimizer
- Can optimize **in-memory storage of rows** with *Tungsten*
- Eventually everything runs over RDDs

Execution of Queries

- An Action on a **Dataset** will trigger a **query**
- Query is optimized by *Catalyst*
- Query is executed using RDDs optimized with *Tungsten*
- A query execution will result in 1 or more **jobs**

Dataset Demo

Further with *Spark*

Spark and Functional Programming

- *Spark* applications can be organized into pure functions
  - Taking RDDs or Datasets as input
  - Returning an RDD or Dataset as output
- These functions can be unit tested

Spark over a Cluster

- In production, *Spark* Jobs execute over a **cluster** of multiples machines
- Spark uses a **cluster manager** such as **YARN** to
  - Allocate memory and processing capacity for **executors** on machines in the cluster
  - Orchestrate execution of **tasks** on **executors**
  - Handle failures and retries