CORRELAID

GOOD CAUSES. BETTER EFFECTS.

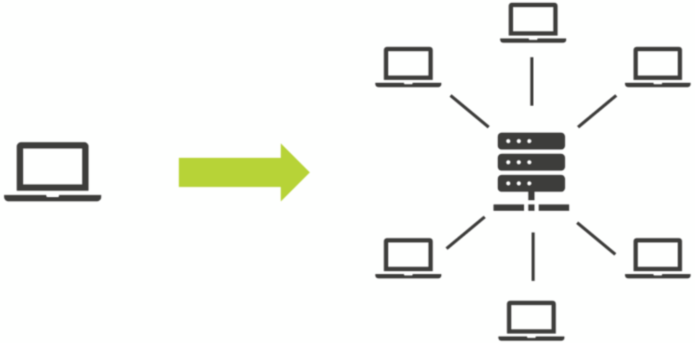
Spark

Outline

- Theoretical aspects
- Working with Spark
- Exercises

Big data

What do you do when the amount of data you have is too much for your single machine?



Distributed Computing

Runtime system

What do you do when the amount of data you have is too much for your single machine?

What needs to be handled?

- Parallelization/synchronisation
- Distribution of computation
- Distribution of data
- Communication between nodes
- Node failures

Difficult to implement everything yourself!

Apache Spark

An open-source unified analytics engine for large-scale data processing

What makes it special?

- Builds upon Hadoop MapReduce (another runtime system for distributed data processing) and extends it to allow for more types of computations
- Fast recovery mechanisms in case of node failure
- Allows to express more complex data pipelines
- Much faster than Hadoop

Apache Spark

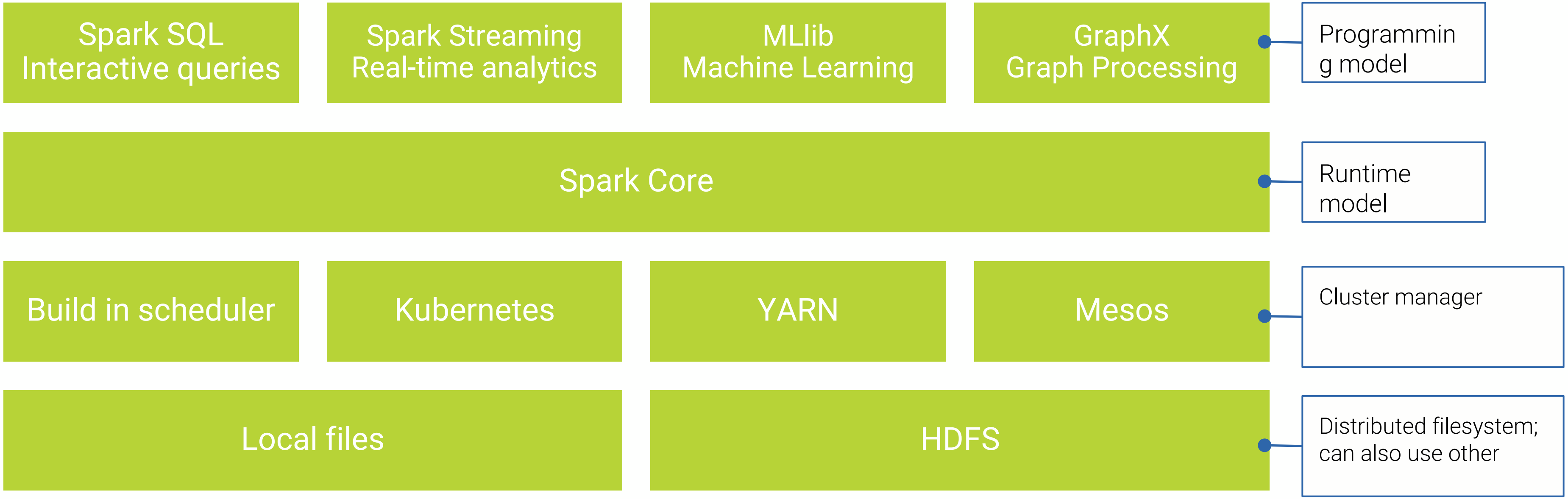
Spark Components

What do you need for Spark to work?

1. Programming model
2. Runtime model
3. Cluster manager
4. Distributed filesystem

Apache Spark

Spark Components



Spark SQL
Interactive queries

Spark Streaming
Real-time analytics

MLlib
Machine Learning

GraphX
Graph Processing

Programmin
g model

Spark Core

Runtime
model

Build in scheduler

Kubernetes

YARN

Mesos

Cluster manager

Local files

HDFS

Distributed filesystem;
can also use other

Apache Spark

Spark Libraries and Programming models

Spark SQL
Interactive queries

Spark Streaming
Real-time analytics

MLlib
Machine Learning

GraphX
Graph Processing

Programming models

- Provide an interface to data processing with Spark as processing engine underneath
- Spark provides several libraries with different functionality and different types of data processing in mind
- Spark Libraries and APIs are available in several programming languages like Scala, Java, Python and R
- In Python there is also a really nice Pandas API which allows to directly use Pandas on Spark

Core concept: RDDs

RDDs - Resilient Distributed Datasets

RDDs

- Fundamental data structure upon every other data structure like DataFrames and Datasets are build upon
- immutable: can not be changed
- tracks lineage information for data recovery

Allows to perform two types of functions: transformations and actions

- Transformations are operations applied on the input data (examples: map(), filter(), sortBy())
- Actions are processes which trigger the creation of new RDDs
- Spark uses lazy evaluation, meaning: when applying transformations to data a new RDDs is only created once you use an action

Apache Spark

Spark Core

A few Remarks

- Algorithms and data structures in Spark are able to exploit memory hierarchy -> can exploit faster access to cached datasets
- Spark utilizes column-oriented storage which allows for faster computation
- Data is split up into partitions: can be adapted manually; should be kept in mind when calling certain functions (shuffle is expensive)

Outline

- Theoretical aspects
- Working with Spark
- Exercises

Working with Spark

Writing your Spark app

Basic principle

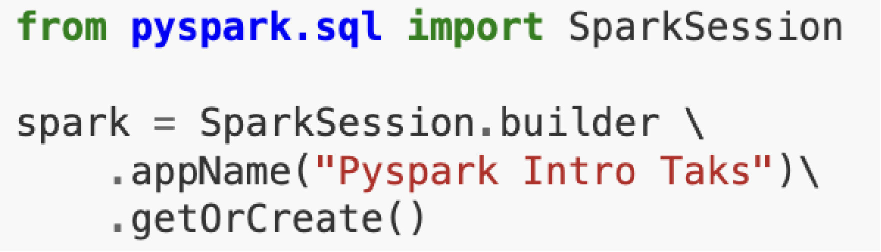
1. Create initial dataset.
2. Analyze it by calling the corresponding methods, e.g. limit(...), filter(...). Each method again returns a Dataset object.
3. Evaluation only starts once the result is required, e.g. by using count(), show() or collect()

Working with Spark

Writing your Spark app

Spark Session

- Instantiates Spark + SQL context.
- From a SparkSession, one can access all contexts and configurations.



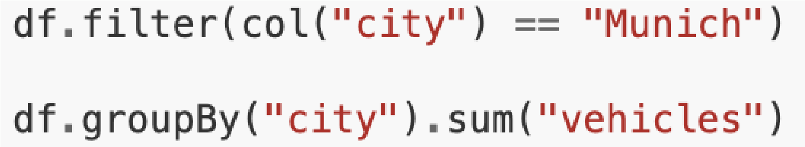
```python
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Pyspark Intro Taks")\
    .getOrCreate()
```

Working with Spark

Writing your Spark app

Spark Transformations

- Instructions on how to modify a data structure
- Lazy evaluation
- Input partitions mapped to output partitions:
  - 1:1 -> **narrow**
  - 1:n -> **wide** (shuffle required)



```
df.filter(col("city") == "Munich")

df.groupBy("city").sum("vehicles")
```

Working with Spark

Writing your Spark app

Spark Transformations

- agg

- except
- flatMap
- intersect
- limit
- orderBy
- select
- union
- distinct
- filter
- groupBy
- joinWith
- map
- sample
- sort

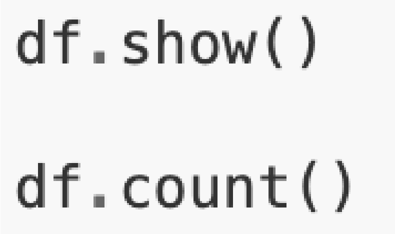
See docs for more information:
https://spark.apache.org/docs/latest/sql-ref.html

Working with Spark

Writing your Spark app

Spark Actions

- Triggers the computation immediately
- Different purposes
  - View data
  - Collect data to objects
  - Write to output



```
df.show()

df.count()
```

Working with Spark

Writing your Spark app

Managing Datasets Collecting
results

- persist
- unpersist
- explain
- printSchema

- describe
- first
- count
- show
- collect
- foreach

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

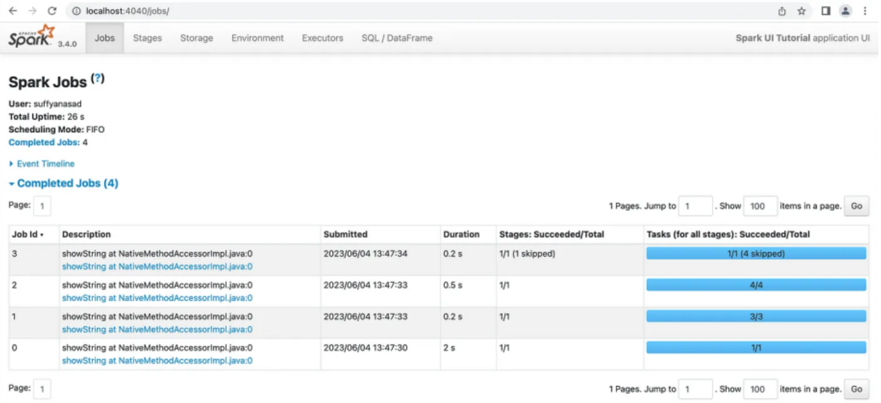
See docs for more information:
https://spark.apache.org/docs/latest/sql-ref.html

Working with Spark

Writing your Spark app

Web UI for development and monitoring

- Monitor the job progress
- Available at http://localhost:4040
- For tuning and debugging



Spark Jobs (?)

User: suffyanasad
Total Uptime: 26 s
Scheduling Mode: FIFO
Completed Jobs: 4

▸ Event Timeline

▾ **Completed Jobs (4)**

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
3	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2023/06/04 13:47:34	0.2 s	1/1 (1 skipped)	1/1 (4 skipped)
2	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2023/06/04 13:47:33	0.5 s	1/1	4/4
1	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2023/06/04 13:47:33	0.2 s	1/1	3/3
0	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2023/06/04 13:47:30	2 s	1/1	1/1

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Outline

- Theoretical aspects
- Working with Spark
- **Exercises**