Introduction to Apache Spark

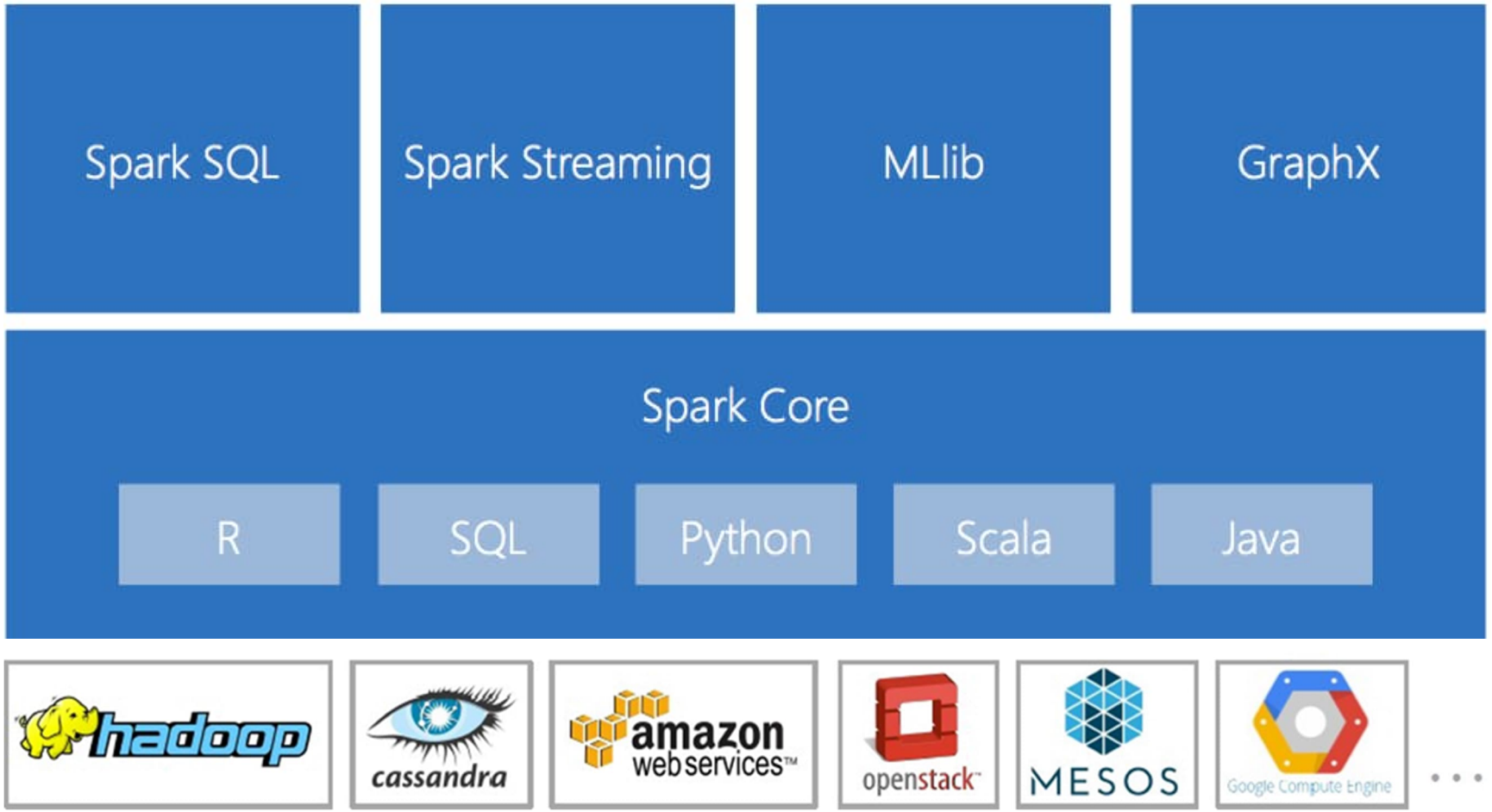
1TD169 - Data Engineering I

Usama Zafar

2025-02-10

UPPSALA
UNIVERSITET

Recap: Lecture 1

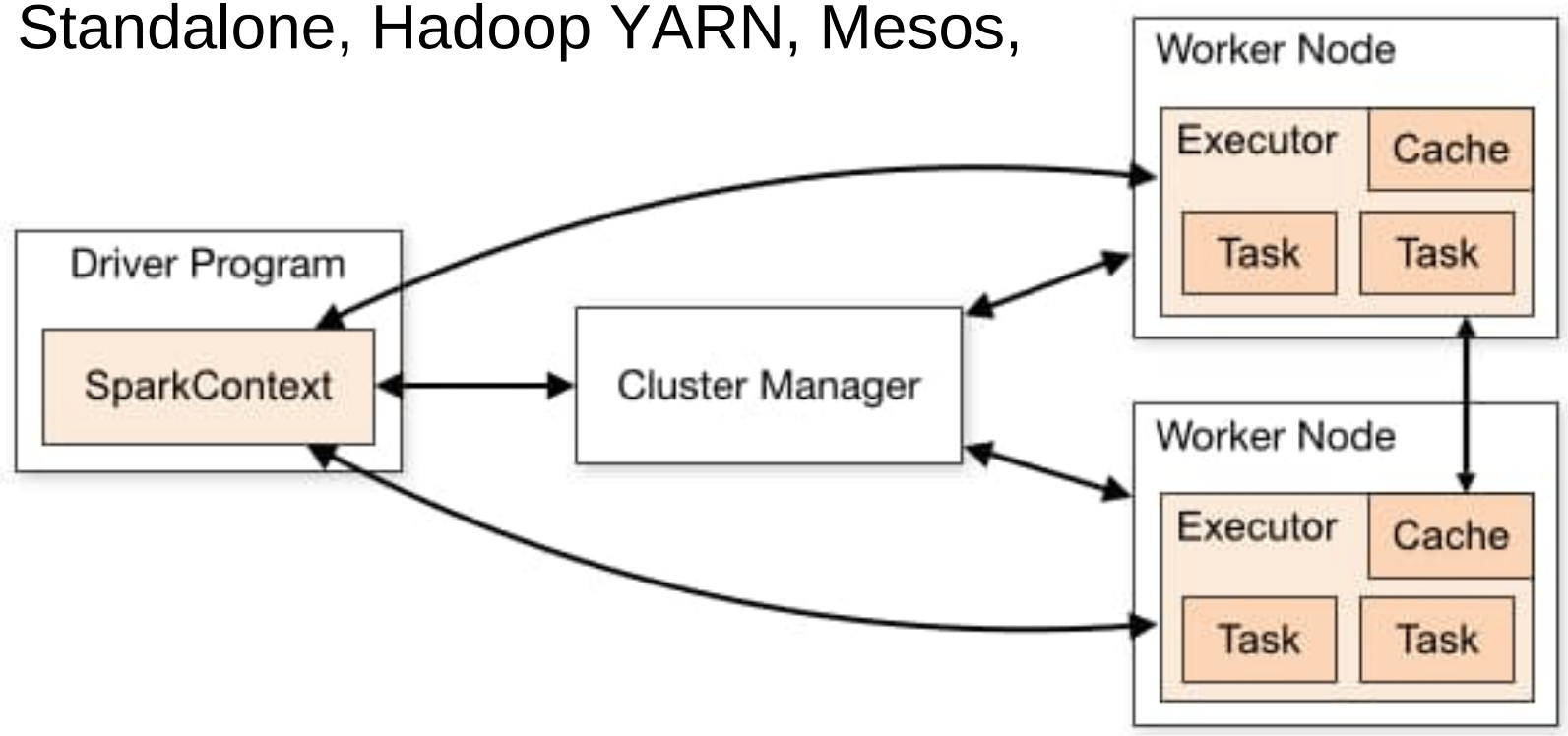


Apache Spark: A distributed computation framework

Recap: Lecture 1

Spark requires a **cluster manager** and a **distributed storage system**.

Cluster manager: Standalone, Hadoop YARN, Mesos, Kubernetes, etc.



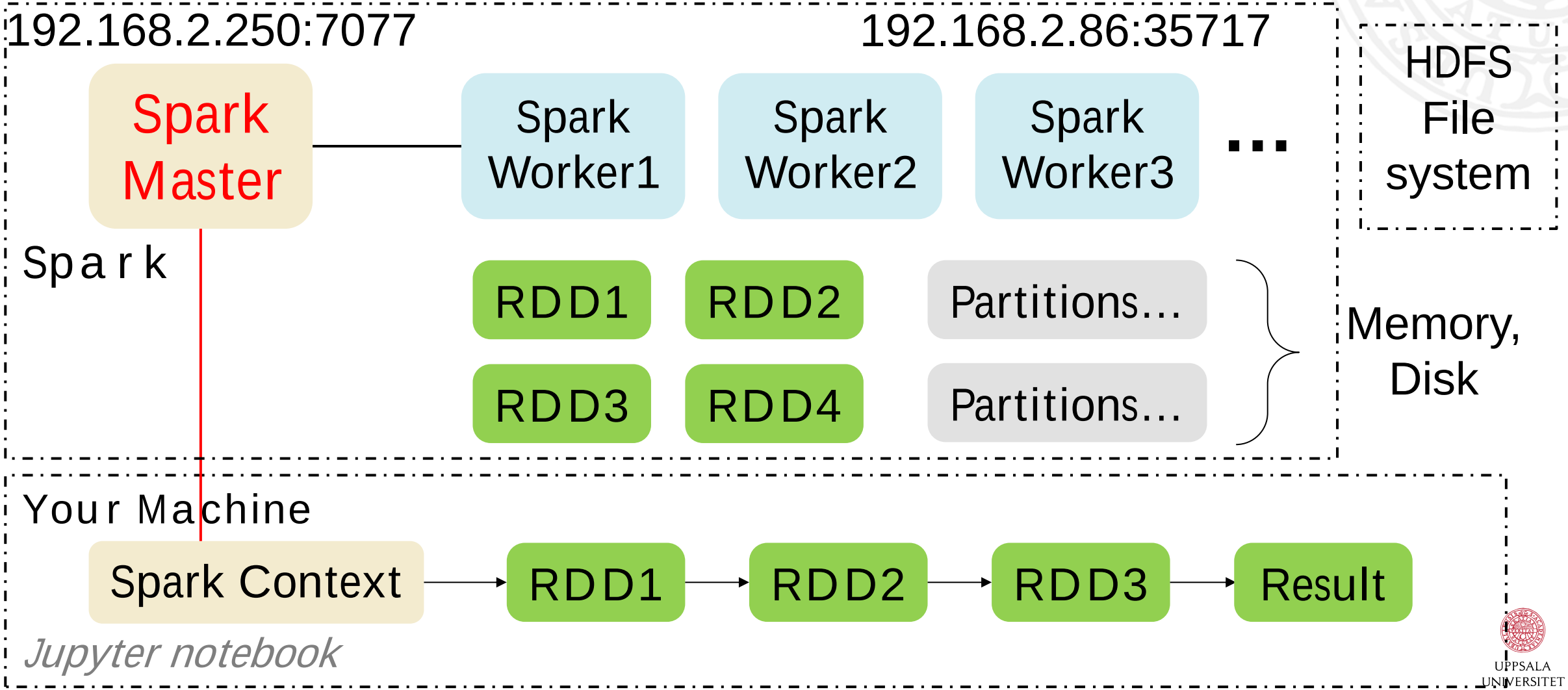
Recap: Lecture 1

- RDD: Resilient Distributed Datasets
- DAG: Directed Acyclic Graphs
- Spark APIs, PySpark
- Spark Web-UI, HDFS Web-UI
- Demo codes

Recap: RDD Operations

Action	**Transformation**
Can have side effects (like saving to disk)	No side effects
Imperative Evaluation (triggers jobs)	Lazily Evaluated. Can be re-evaluated.
Invocation to request output, or to ‘do’ something	Describes how to transform from one step to the next.
Returns or outputs data	Returns another RDD
reduce(..), save(...), collect(..), count(..),	map(...), sort(..), filter(..), reduceByKey(..)
https://spark.apache.org/docs/latest/rdd-programming-guide.html#actions	https://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations

Recap: Basic Spark Deployment



Spark Lecture 2

Spark Lecture 2

- RDD Operations: transformations vs actions – *demos*
- Spark application: Key Concepts
- Shared Variables: Broadcast variables, accumulators
- SparkSQL: Datasets / DataFrames
- Other APIs: MLlib, GraphX, Streaming
- Newer frameworks

Examples 3–5: Methods on RDD

Notebooks: Examples

- Lecture1_Example3_RDD_methods.ipynb
- Lecture1_Example4_RDD_Gutenborg.ipynb
- Lecture1_Example5_RDD_CommonCrawl_Demo.ipynb

Key Application Concepts

Application:

A user program built on Spark using its APIs. It consists of a driver program and executors on the cluster.

SparkSession:

An object that provides a point of entry to interact with underlying Spark functionality and allows programming Spark with its APIs.

UPPSALA
UNIVERSITET

Key Application Concepts

Job:

A parallel computation consisting of multiple tasks that gets spawned in response to a Spark action (e.g., save(), collect()).

Stage:

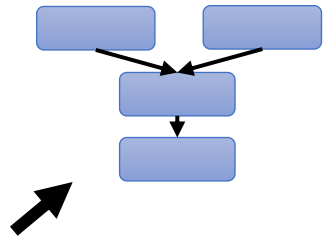
Each job gets divided into smaller sets of tasks called stages that depend on each other.

Task:

A single unit of work or execution that will be sent to a Spark executor.

Job Scheduling

RDD Objects



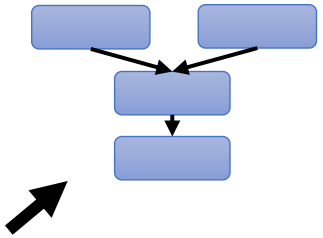
```
rdd1.join(rdd2)
    .groupBy(…)
    .filter(…)
```

build operator DAG

source: https://cwiki.apache.org/confluence/display/SPARK/Spark+Internals

Job Scheduling

RDD Objects

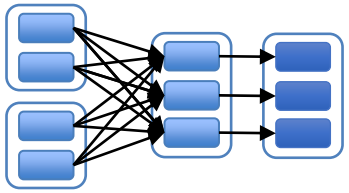


```
rdd1.join(rdd2)
    .groupBy(…)
    .filter(…)
```

build operator DAG

DAG

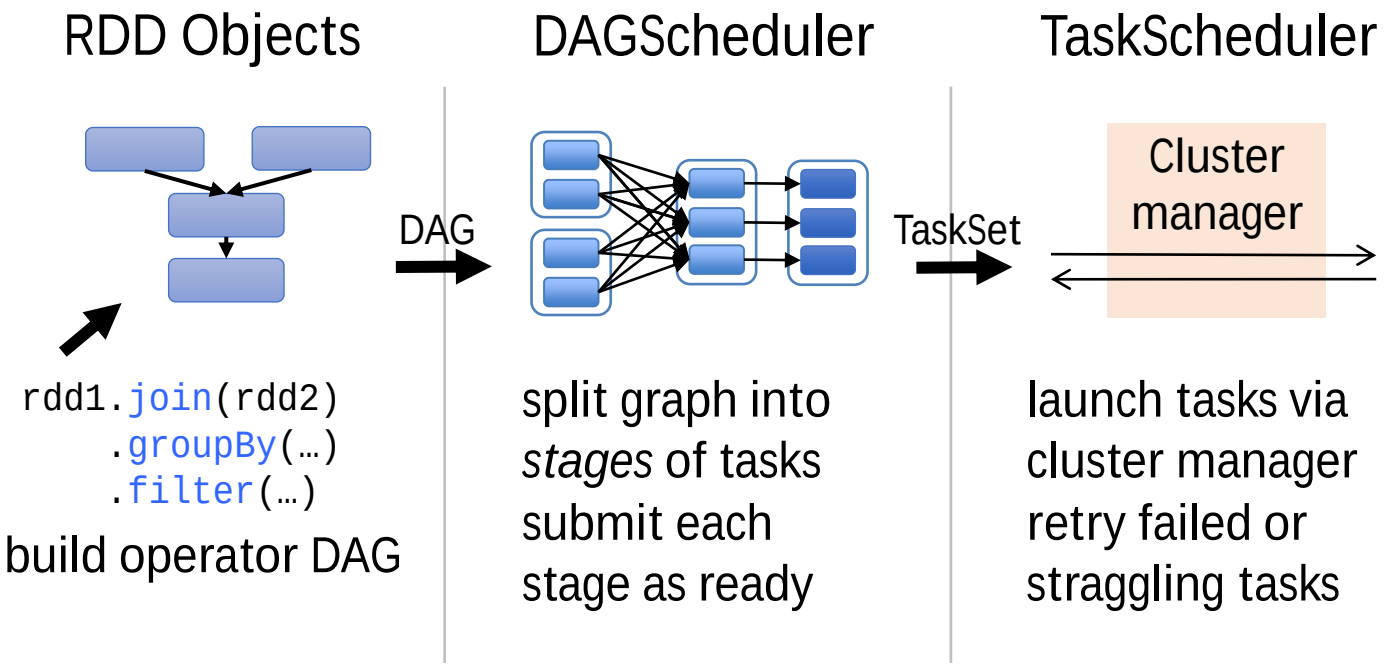
DAGScheduler



split graph into *stages* of tasks
submit each stage as ready

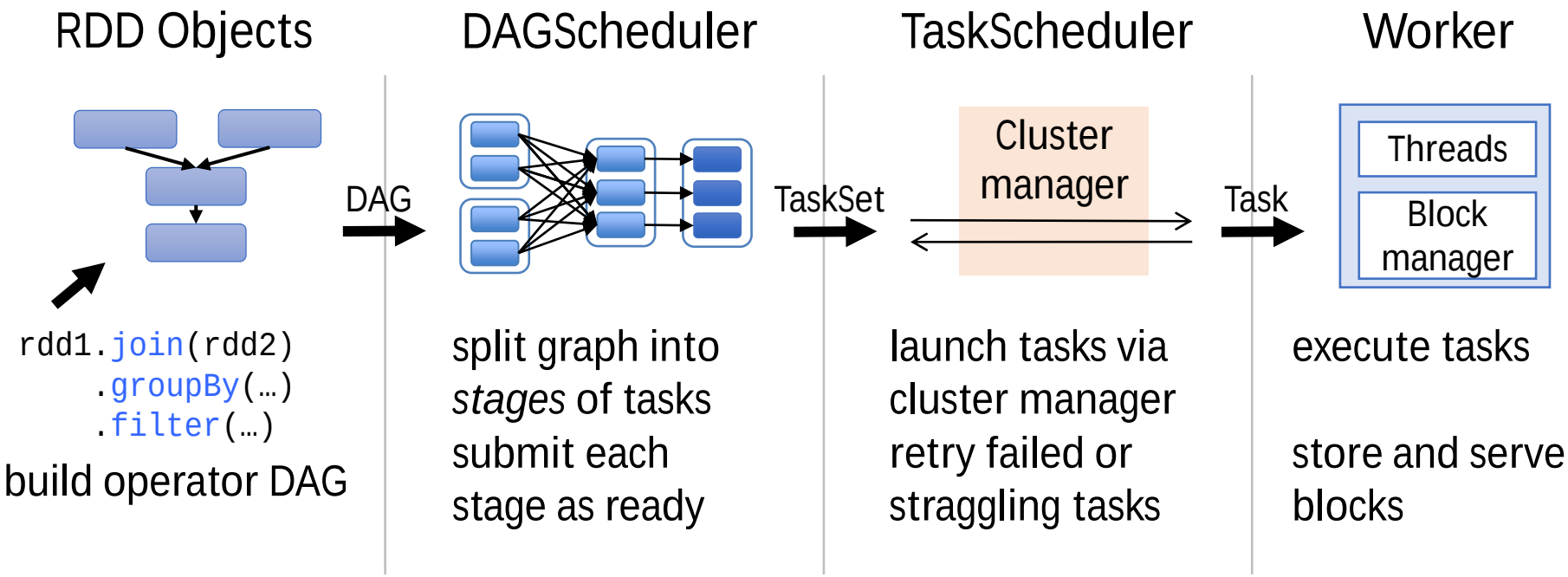
source: https://cwiki.apache.org/confluence/display/SPARK/Spark+Internals

Job Scheduling



source: https://cwiki.apache.org/confluence/display/SPARK/Spark+Internals

Job Scheduling



source: https://cwiki.apache.org/confluence/display/SPARK/Spark+Internals

Shared Variables

Broadcast variables:

Immutable broadcast variables shared between executors. Large read-only data (like models, static datasets).

Accumulators:

A shared variable that can be accumulated / synchronized between executors.

- Lecture2_Example1_Shared_State.ipynb

UPPSALA UNIVERSITET

RDD vs DataFrame

RDD: Collection of elements, Serialization (JAVA).

aggregations and grouping operations?

Structured data? e.g. relational database, data frame…

SparkSQL – Datasets / DataFrameAPI

Spark SQL and DataFrames (apache.org)

RDD vs DataFrame

Is your data…?	Interface	Examples	Model	API	Paradigm	Python Package
Unstructured	RDD	Text, HTML, XML, JSON	Set of values, or Set of key/value pairs	`SparkContext`	Low Level, Fine control of MapReduce functions.	`pyspark.*`
Structured	DataFrames, SQL, Datasets	CSV, any record-based file format.	Tables (Rows and Columns)	`SparkSession`	High Level, more like SQL (MapReduce abstracted away) (Can actually use ‘database SQL)	`pyspark.sql.*`

DataFrame APIs

`SparkSession/SQLcontext:` the entry point to the Dataset and DataFrame API.

`SparkSession.builder:` to create a Spark session.

`SparkSession.sparkContext:` returns the underlying SparkContext.

`SparkSession.read:` read data in as a DataFrame.

`SparkSession.sql(sqlQuery):` returns a DataFrame as the result of given query.

`SparkSession.udf:` User Defined Function

UPPSALA
UNIVERSITET

Demo

Example 2:

Spark SQL with Bitcoin Historical Data: btcusd_1-min_data.csv

Lecture2_Example2_bitcoin_trend_analysis.ipynb

MLlib

Apache Spark's scalable machine learning library.

MLlib | Apache Spark

MLlib: Main Guide (apache.org)

RDD-based / DataFrame-based APIs

Demo

Example 3:

Spark MLlib with Bitcoin Historical Data:

Feature Engineering + ElasticNet Linear Regression

Lecture2_Example3_bitcoin_trend_analysis_mllib.ipynb

GraphX

Apache Spark's API for graphs and graph-parallel computation.

GraphX | Apache Spark GraphX - Spark Documentation

Only in Scala! For PySpark, consider package GraphFrame

Overview - GraphFrames Documentation

Aims to provide both the functionality of GraphX and extended functionality taking advantage of Spark DataFrames.

Demo

Example 4:

Graph Processing with pyspark using package GraphFrame

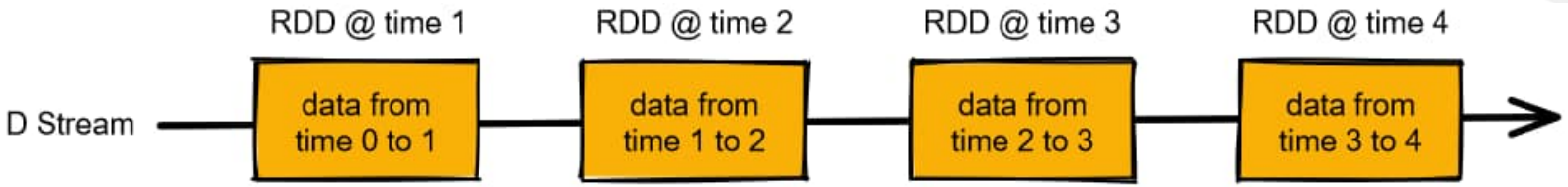
Lecture2_Example4_GraphFrames.ipynb

Streaming

Spark Streaming - Spark Documentation

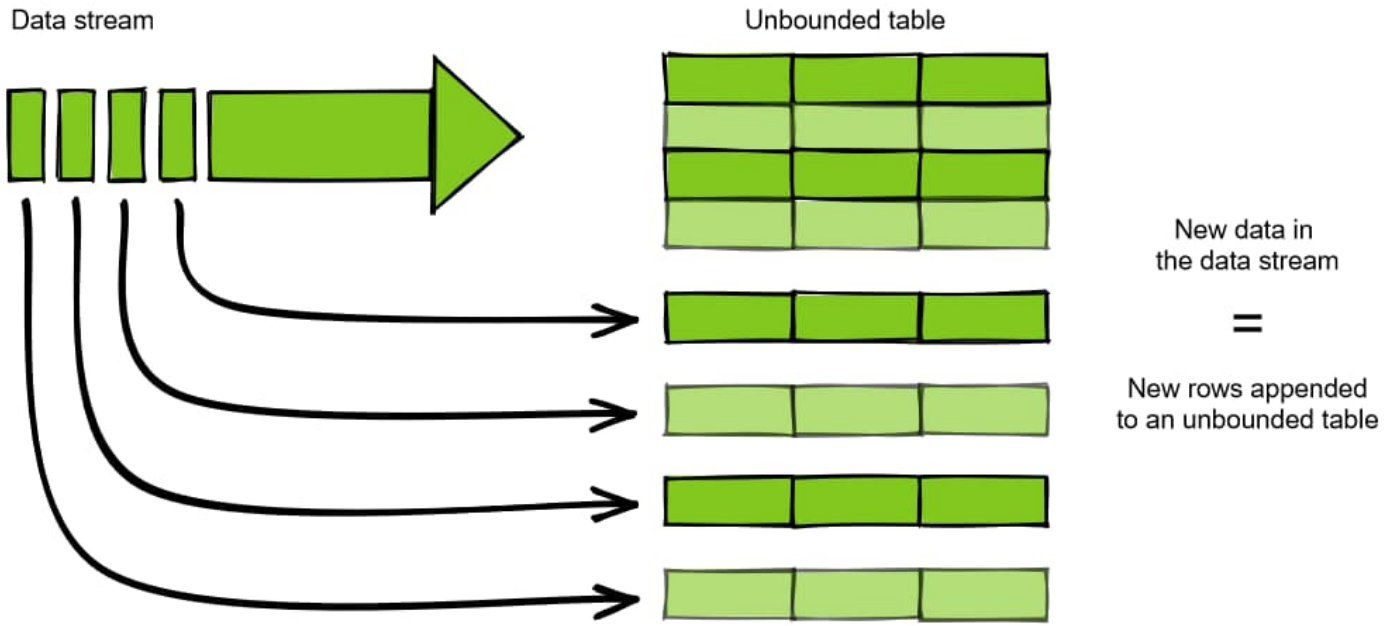
Batch_execution:

(SparkRDD)



Spark Structured Streaming

(SparkSQL)



Kafka Streams vs. Spark Streaming (redpanda.com)

UPPSALA UNIVERSITET

Streaming

Pros: Strong compatibility:

high-throughput, fault-tolerant

SQL, APIs in multi languages

Cons: tuning; stateless; back pressure; batch processing → latency? adaptability?

Event_based: Apache **Flink**; Lightweight: **Kafka**; Google Cloud **Dataflow…**

More examples in DE-II

Wrap up

Spark vs Hadoop: memory vs disk, High speed SSD?

In AI / ML, GPU accelerating? huge-scale simulation?

Distributed / Federated learning? Ray

Streaming: Storm, Kafka, Flink, Pulsar, etc.