ORACLE Certified Professional

Oracle Certified Professional Java Programmer

Microsoft Certified Professional

sohailimran@yahoo.com

Sohail IMRAN سهيل عمران

intro

Ref: Web

intro

Spark is an Apache project advertised as “**lightning fast cluster computing**”. It has a thriving open-source community and is the most active Apache project at the moment.

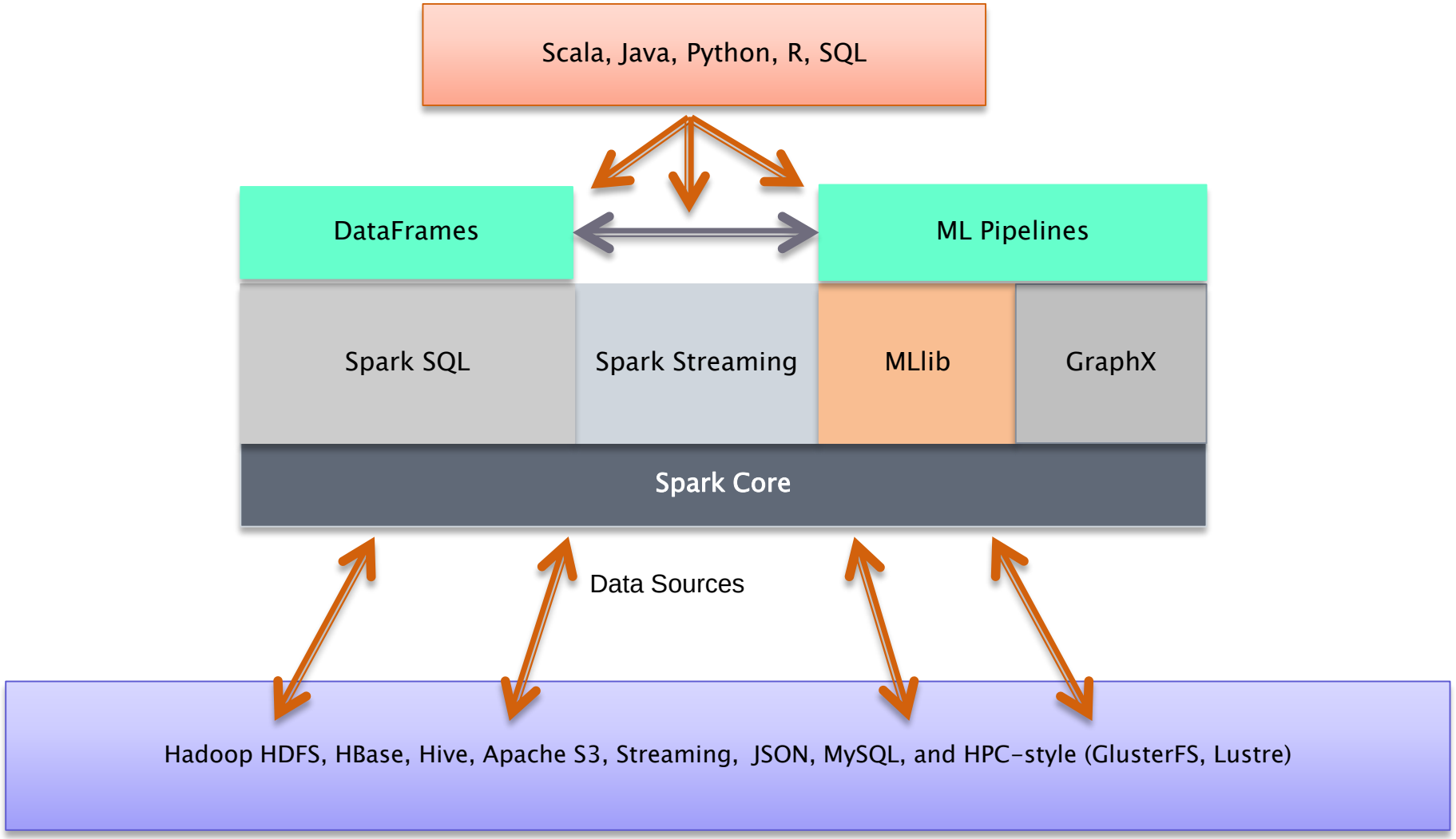
Spark provides a faster and more general data processing platform. Spark lets you run programs up to **100x** faster in memory, or **10x** faster on disk, **than Hadoop**. Last year, Spark took over Hadoop by completing the 100 TB Daytona GraySort contest 3x faster on one tenth the number of machines and it also became the fastest open source engine for sorting a petabyte.

Additional key features of Spark include:

Currently provides APIs in **Scala**, **Java**, and **Python**, with support for other languages (such as **R**) on the way Integrates well with the Hadoop ecosystem and data sources (**HDFS**, **Amazon S3**, **Hive**, **HBase**, etc.)

Can run on clusters managed by Hadoop **YARN** or Apache **Mesos**, and can also run standalone.

spark ecosystem framework



features


Fault Tolerance
Dynamic in Nature
Lazy Evaluation
Real Time
Stream Processing
Spark
Reusability
Speed
Advanced Analytics
In Memory Computing

map-reduce vs spark

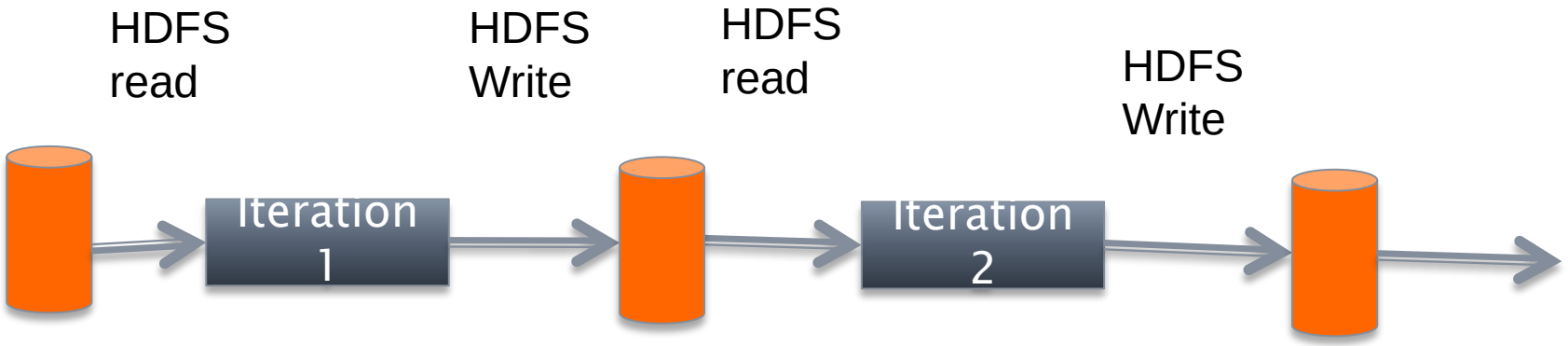
MapReduce	Spark
It is a programming model within the Hadoop framework for distributed computing.	It is a Hadoop enhancement to MapReduce used for big data workloads.
It operates in sequential steps by reading data from the cluster and performing its operations on the data.	It is a general purpose distributed data processing engine that processes data in parallel across a cluster.
It is relatively slower as it performs operations on the disk.	It is 100 times faster in-memory and 10 times faster on disk.
It cannot deliver near real-time analytics from the data.	It can deal with real-time processing.
It needs to handle low level APIs to process the data, which requires lots of coding.	Its general programming model makes it relatively easy to use.

MapReduce - limitations

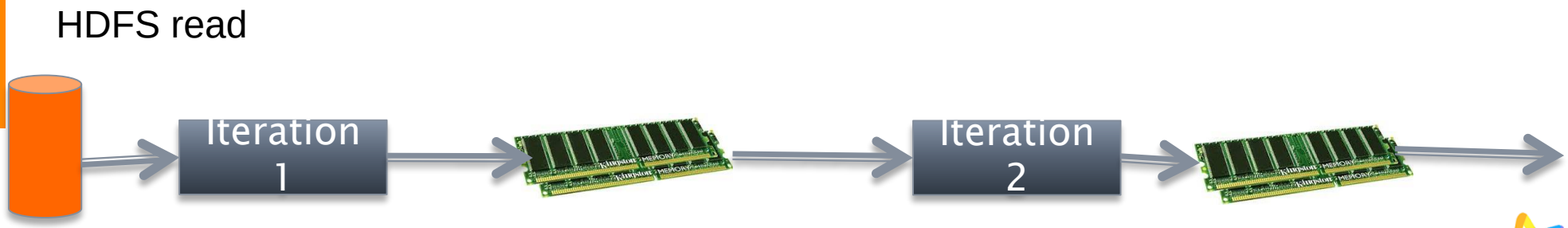
- Scalability
  - Maximum Cluster Size – 4000 Nodes
  - Maximum Concurrent Tasks – 40000
  - Coarse synchronization in Job Tracker
- Single point of failure
  - Failure kills all queued and running jobs
  - Jobs need to be resubmitted by users
- Restart is very tricky due to complex state
- MapReduce problems:
  - Many problems aren't easily described as map-reduce
  - Persistence to disk typically slower than in-memory work

Spark uses Memory instead of Disk

Hadoop: Use Disk for Data Sharing



Spark: In-Memory Data Sharing



Core

Spark Core is the base engine for large-scale parallel and distributed data processing. It is responsible for:

- memory management and fault recovery
- scheduling, distributing and monitoring jobs on a cluster
- interacting with storage systems

Spark introduces the concept of an RDD (**Resilient Distributed Dataset**), an immutable fault-tolerant, distributed collection of objects that can be operated on in parallel. An RDD can contain any type of object and is created by loading an external dataset or distributing a collection from the driver program.

RDDs support two types of operations:

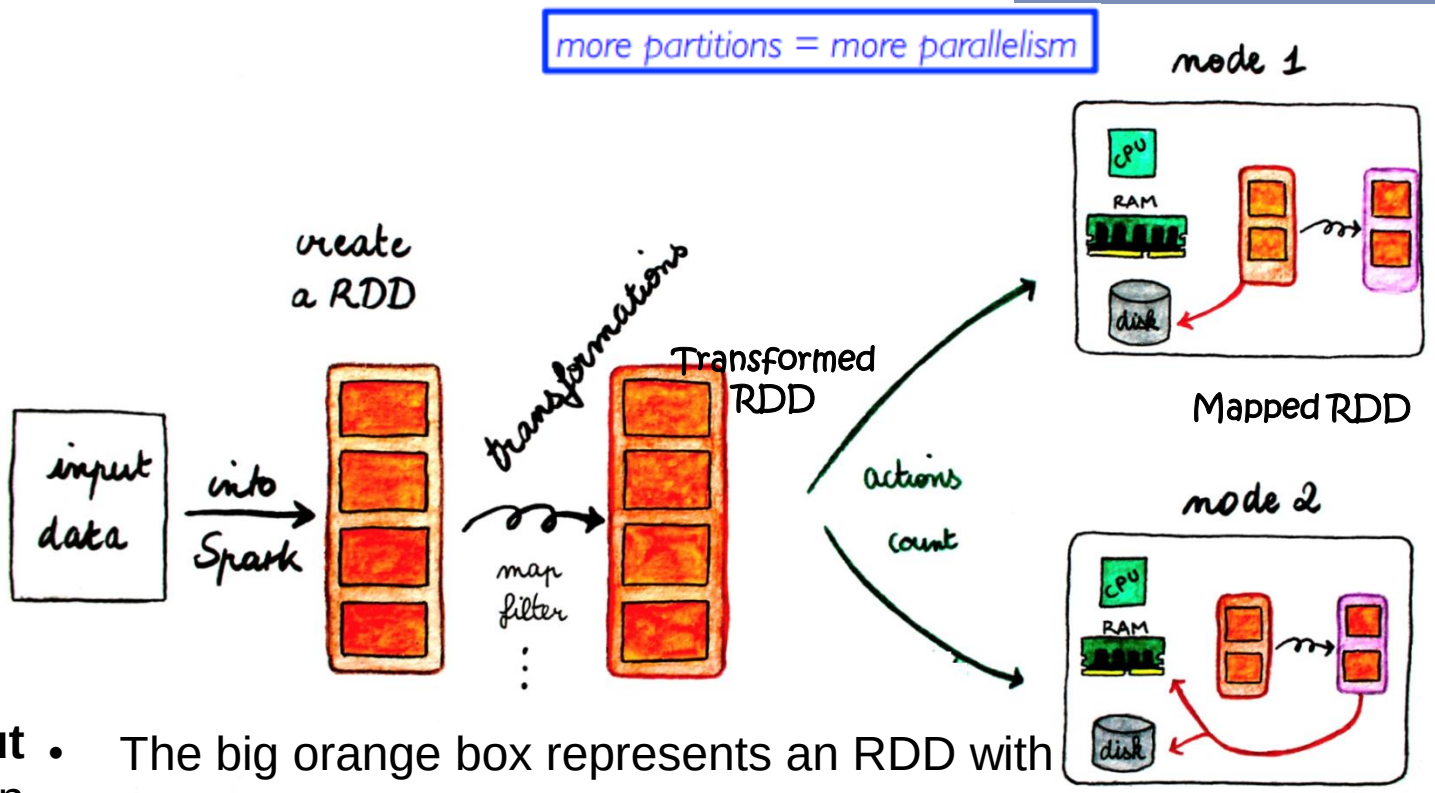
Transformations are operations (such as **map**, **filter**, **join**, **union**, and so on) that are performed on an RDD and which yield a new RDD containing the result.

Actions are operations (such as **reduce**, **count**, **first**, and so on) that return a value after running a computation on an RDD.

transformation

Transformations in Spark are "**lazy**", meaning that they **do not compute** their results right away. Instead, they just "remember" the **operation to be performed** and the dataset (e.g., **file**) to which the operation is to be performed. The transformations are only actually **computed when an action is called** and the result is returned to the driver program. This design enables Spark to run more efficiently. For example, if a big file was transformed in various ways and passed to first action, Spark would **only process and return the result** for the first line, rather than do the work for the entire file. By default, each transformed RDD may be **recomputed each time** you run an action on it. However, you may also persist an RDD in memory using the persist or cache method, in which case Spark will keep the elements around on the cluster for much faster access the next time you query it.

Operation on Resilient Distributed Datasets



To the left, the **input data** comes from an external storage. The data are loaded into Spark and an Resilient Distributed Dataset (RDD) is created.

- The big orange box represents an RDD with its partitions (small orange boxes). You can chain **transformations** on RDDs. As the transformations are lazy, the partitions will be sent across the nodes of the cluster when you will call an **action** on the RDD.
- Once a partition is located on a node, you can continue to operate on it.

methods

Some Transformations

Transformation	Description
map(func)	return a new distributed dataset formed by passing each element of the source through a function func
filter(func)	return a new dataset formed by selecting those elements of the source on which func returns true
distinct ([numTasks]))	return a new dataset that contains the distinct elements of the source dataset
flatMap(func)	similar to map, but each input item can be mapped to 0 or more output items (so func should return a Seq rather than a single item)

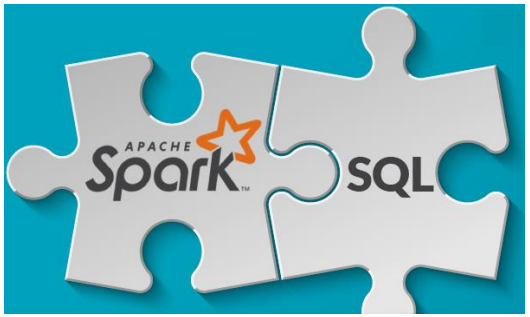
Some Key-Value Transformations

Key-Value Transformation	Description
reduceByKey(func)	return a new distributed dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function func, which must be of type (V,V) -> V
sortByKey()	return a new dataset (K, V) pairs sorted by keys in ascending order
groupByKey()	return a new dataset of (K, Iterable) pairs

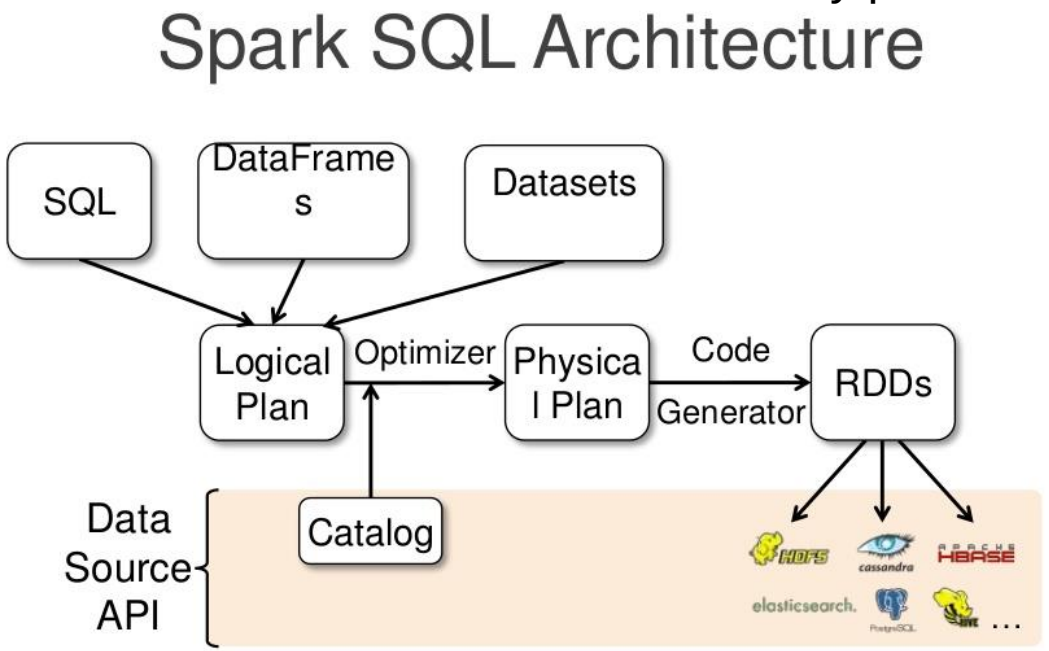
Some Actions

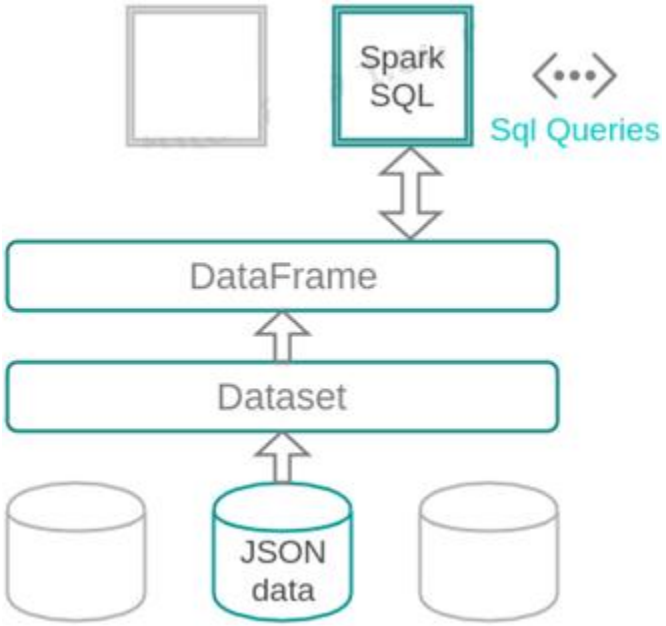
Action	Description
reduce(func)	aggregate dataset's elements using function func. func takes two arguments and returns one, and is commutative and associative so that it can be computed correctly in parallel
take(n)	return an array with the first n elements
collect()	return all the elements as an array WARNING: make sure will fit in driver program
takeOrdered (n, key=func)	return n elements ordered in ascending order or as specified by the optional key function

Spark SQL



SparkSQL is a Spark component that supports querying data either via **SQL** or via the **Hive Query Language**. It originated as the Apache Hive port to run on top of Spark (in place of MapReduce) and is now integrated with the Spark stack. In addition to providing support for various data sources, it makes it possible to weave SQL queries with code transformations which results in a very powerful tool.





Streaming analytics with Spark Streaming

The Spark Streaming API closely matches that of the Spark Core, making it easy for programmers to work in the worlds of both batch and streaming data.
Spark Streaming supports real time processing of streaming data, such as production web server log files (e.g. Apache Flume and HDFS/S3), social media like Twitter, and various messaging queues like Kafka. Under the hood, Spark Streaming receives the input data streams and divides the data into batches.

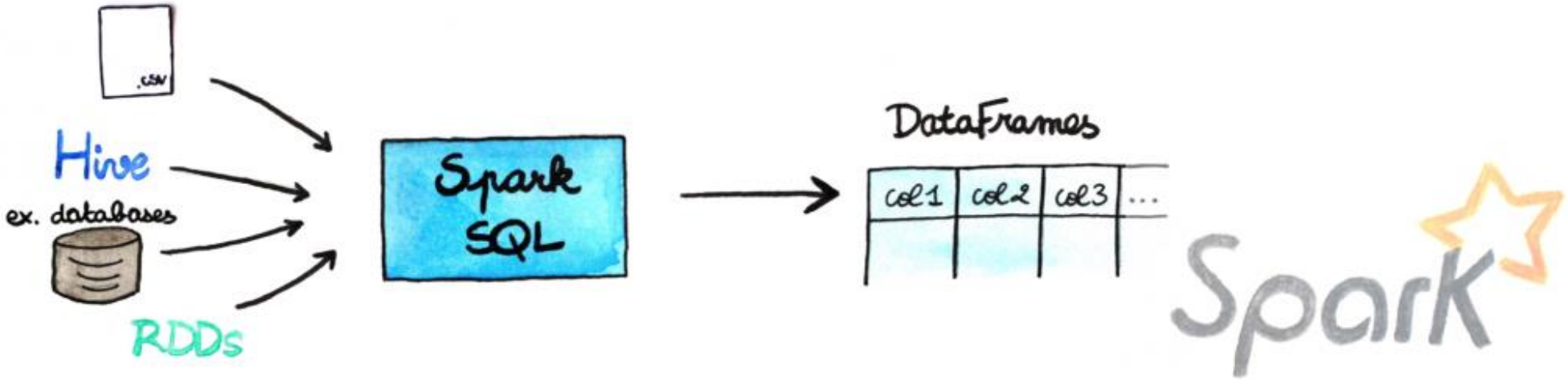
Spark Streaming receives live input data streams and divides the data into batches, which are then processed by the Spark engine to generate the final stream of results in batches.

The abstraction, which represents a continuous stream of data is the DStream (discretized stream).

Structured data with the DataFrame

A DataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood.
In fact, the Spark SQL/dataframe component provides a SQL like interface. So you can apply SQL like queries directly on the RDDs.

DataFrames can be constructed from different sources such as: structured data files, tables in Hive, external databases, or existing RDDs.



Spark MLlib

Spark MLLib is one of the elements of the Apache Spark framework and it utilizes all of its advantages. It allows to apply machine learning to large data sets with no scalability concerns. The system has dozens of built-in machine learning algorithms which may be applied depending on a particular business case. These include:

- **Classification**: logistic regression, naive Bayes
- **Regression**: generalized linear regression, isotonic regression
- **Decision trees**: random forests, and gradient-boosted trees
- **Recommendation**: alternating least squares (ALS)
- **Clustering**: K-means, Gaussian mixtures (GMMs)
- **Topic modeling**: latent Dirichlet allocation (LDA)
- **Feature transformations**: standardization, normalization, hashing
- Model **evaluation** and hyper-parameter **tuning**
- **ML Pipeline** construction
- **ML persistence**: saving and loading models and pipelines
- **Survival analysis**: accelerated failure time model
- Frequent itemset and sequential **pattern mining**: FP-growth, association rules, PrefixSpan
- **Distributed linear algebra**: singular value decomposition (SVD), principal component analysis (PCA)
- **Statistics**: summary statistics, hypothesis testing

Spark GraphX

GraphX is a library for manipulating graphs and performing graph-parallel operations.
It provides a uniform tool for ETL, exploratory analysis and iterative graph computations.
At a high-level, GraphX extends the Spark RDD abstraction by introducing the Resilient Distributed Property Graph: a directed multigraph with properties attached to each vertex and edge. To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and mapReduceTriplets) as well as an optimized variant of the Pregel API. In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks.
Apart from built-in operations for graph manipulation, it provides a library of common graph algorithms such as PageRank.

GraphX library provides graph operators like **subgraph**, **joinVertices**, and **aggregateMessages** to transform the graph data. It provides several ways of building a graph from a collection of **vertices** and **edges** in an **RDD** or on **disk**. GraphX also includes a number of graph **algorithms** and **builders** to perform graph analytics tasks.