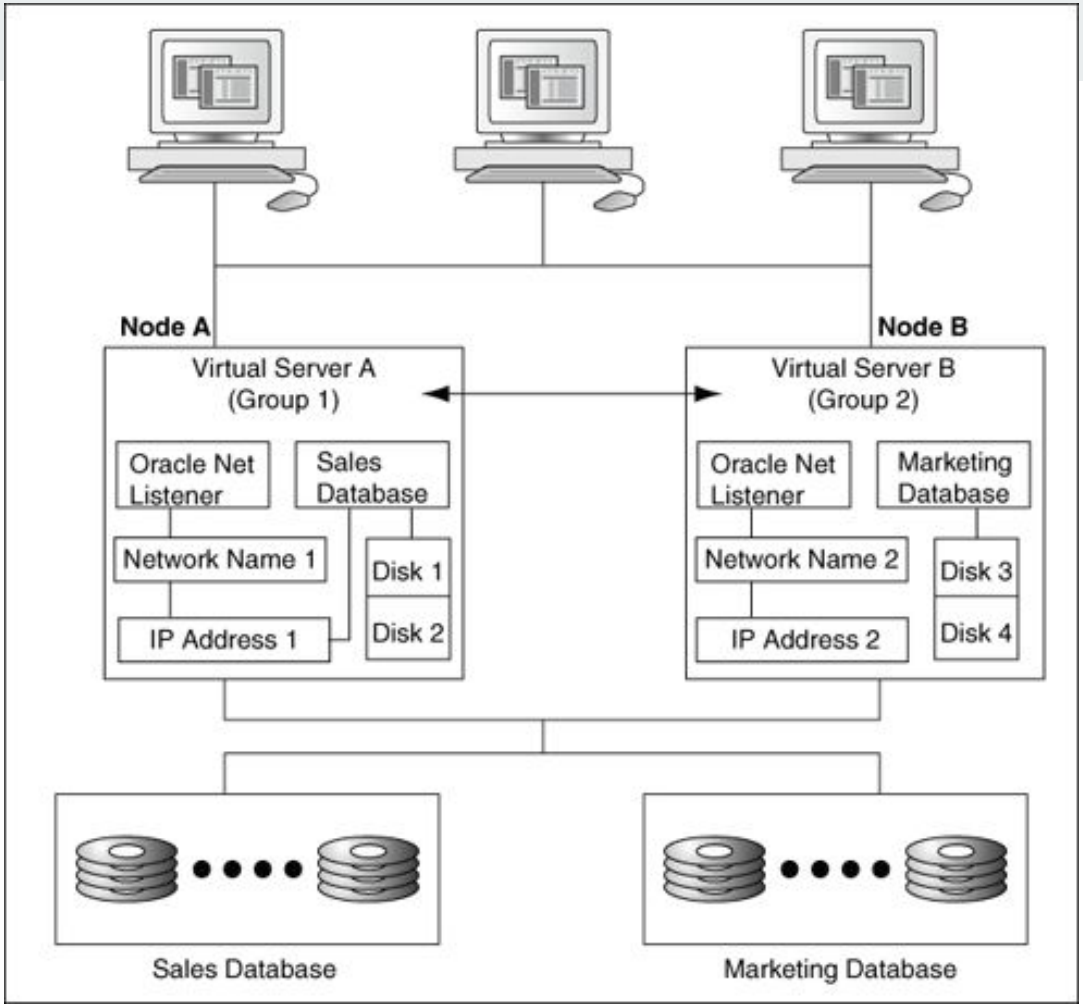
APACHE Spark™

Strongest ETL tool on the market

CONTENTS

Custer and Node

Nodes represent physical or virtual machines that provide CPU and RAM resources for container-based applications. Nodes are grouped together into clusters.



Map Reduce and why Spark was created

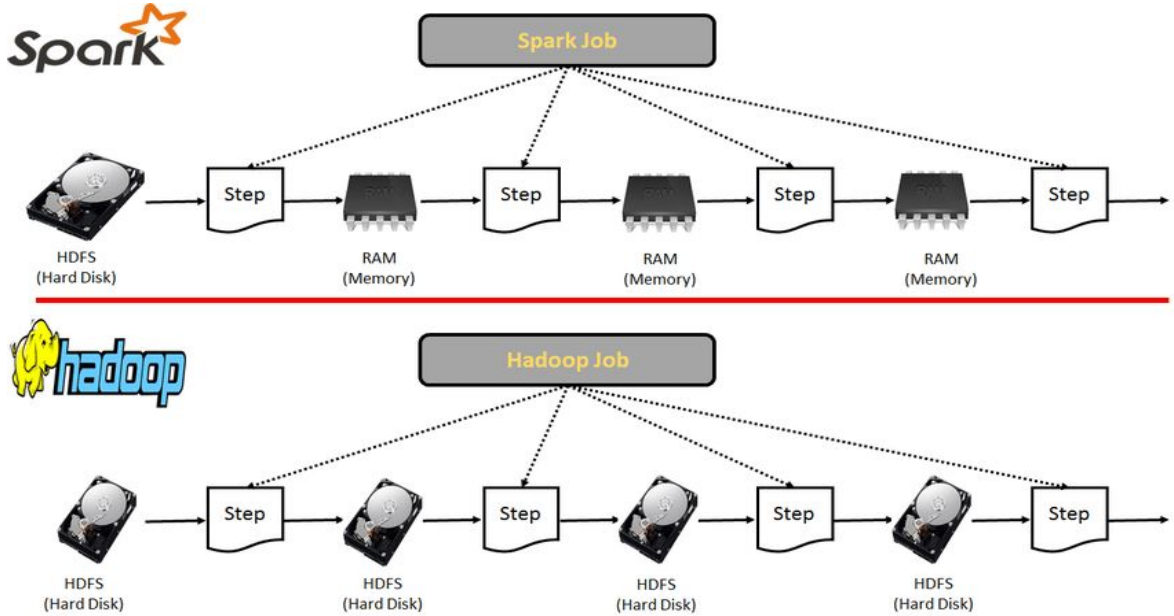
- MapReduce is a Hadoop framework used for writing applications that can process vast amounts of data on large clusters.
- In 2005 builded a distributed computing infrastructure for a Java-based free software search engine. Its basis was a publication of Google employees.
- Major companies in the market have worked with him. In February 2008, Yahoo launched a clustered search engine based on 10,000 processor cores operated by Hadoop

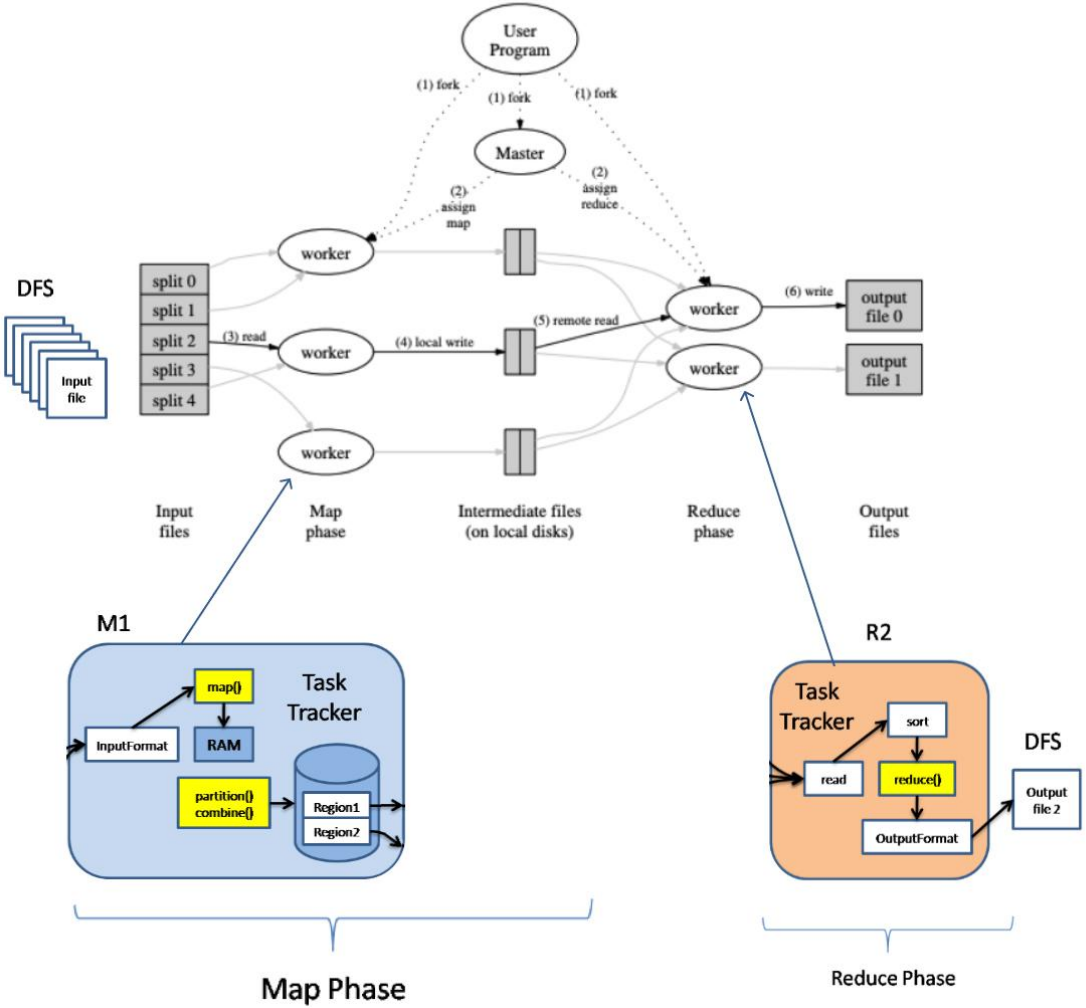
Problems, The Big Problems...

1. Each calculation is saved to the hard disk. This delays the execution of the program
2. You can only write code in Java
3. Difficult code architecture. At the end of it comes out a very long code
4. Your face when you wrote the code for Map Reduce:

Each calculation is saved to the hard Disk

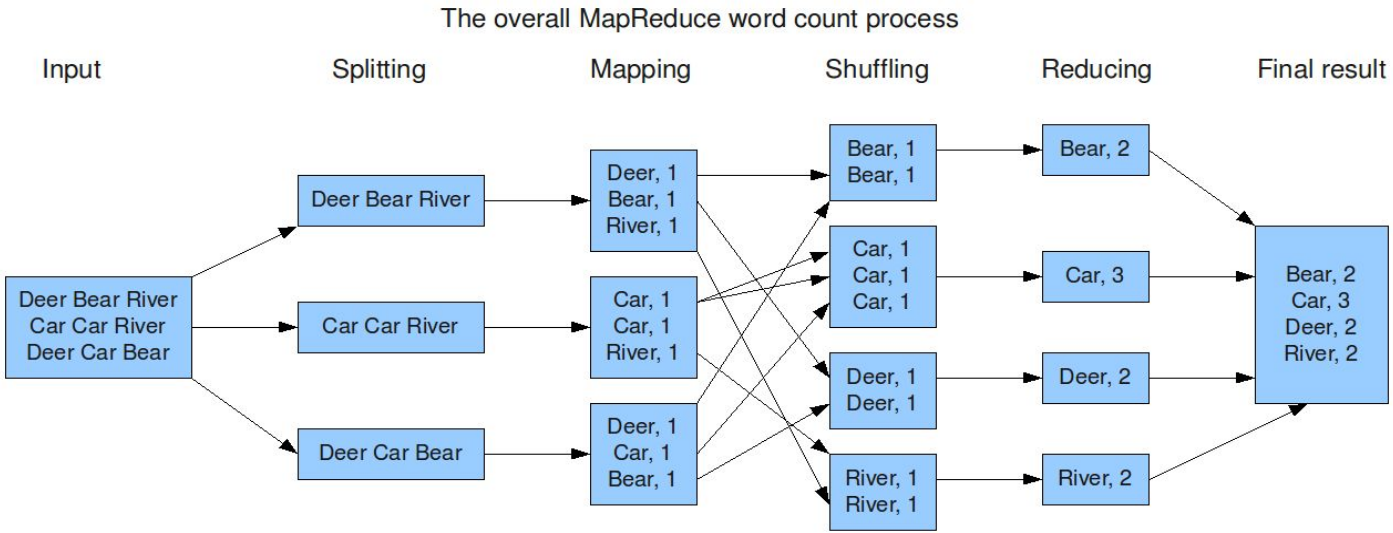
- Every operation it saves to HDFS. In HDFS, files are stored in folders as in a file system, so in order to save the information in HDFS it needs to convert it into the correct format




User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
DFS
Input file
split 0
split 1
split 2
split 3
split 4
(3) read
worker
worker
worker
(4) local write
(5) remote read
worker
worker
(6) write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files
M1
map()
Task Tracker
InputFormat
RAM
partition()
combine()
Region1
Region2
R2
Task Tracker
sort
read
reduce()
OutputFormat
DFS
Output file 2
Map Phase
Reduce Phase

Map Reduce example, word count

- In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair



You need a lot of time for programming

```java
// Importing libraries
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,
                                                Text, Text, IntWritable> {

    // Map function
    public void map(LongWritable key, Text value, OutputCollector<Text,
                IntWritable> output, Reporter rep) throws IOException
    {

        String line = value.toString();

        // Splitting the line on spaces
        for (String word : line.split(" "))
        {
            if (word.length() > 0)
            {
                output.collect(new Text(word), new IntWritable(1));
            }
        }
    }
}
```

```java
// Importing libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements Reducer<Text,
                                        IntWritable, Text, IntWritable> {

    // Reduce function
    public void reduce(Text key, Iterator<IntWritable> value,
                OutputCollector<Text, IntWritable> output,
                        Reporter rep) throws IOException
    {

        int count = 0;

        // Counting the frequency of each words
        while (value.hasNext())
        {
            IntWritable i = value.next();
            count += i.get();
        }

        output.collect(key, new IntWritable(count));
    }
}
```

```java
// Importing libraries
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {

    public int run(String args[]) throws IOException
    {
        if (args.length < 2)
        {
            System.out.println("Please give valid inputs");
            return -1;
        }

        JobConf conf = new JobConf(WCDriver.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        conf.setMapperClass(WCMapper.class);
        conf.setReducerClass(WCReducer.class);
        conf.setMapOutputKeyClass(Text.class);
        conf.setMapOutputValueClass(IntWritable.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        JobClient.runJob(conf);
        return 0;
    }

    // Main Method
    public static void main(String args[]) throws Exception
    {
        int exitCode = ToolRunner.run(new WCDriver(), args);
        System.out.println(exitCode);
    }
```

How I would like to see the code

get File.txt

read.File.txt

split.File

map(splited.File)

suffle(maped.File))

reduce(suffled.File)

give.output

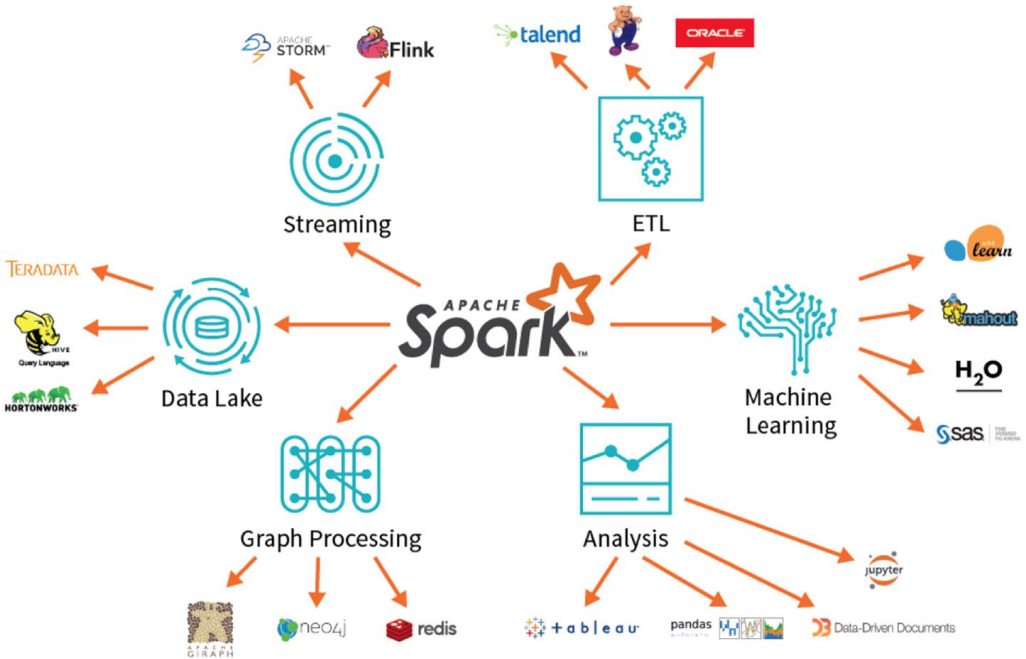
What is Spark and what is it used for

Smart problem solving

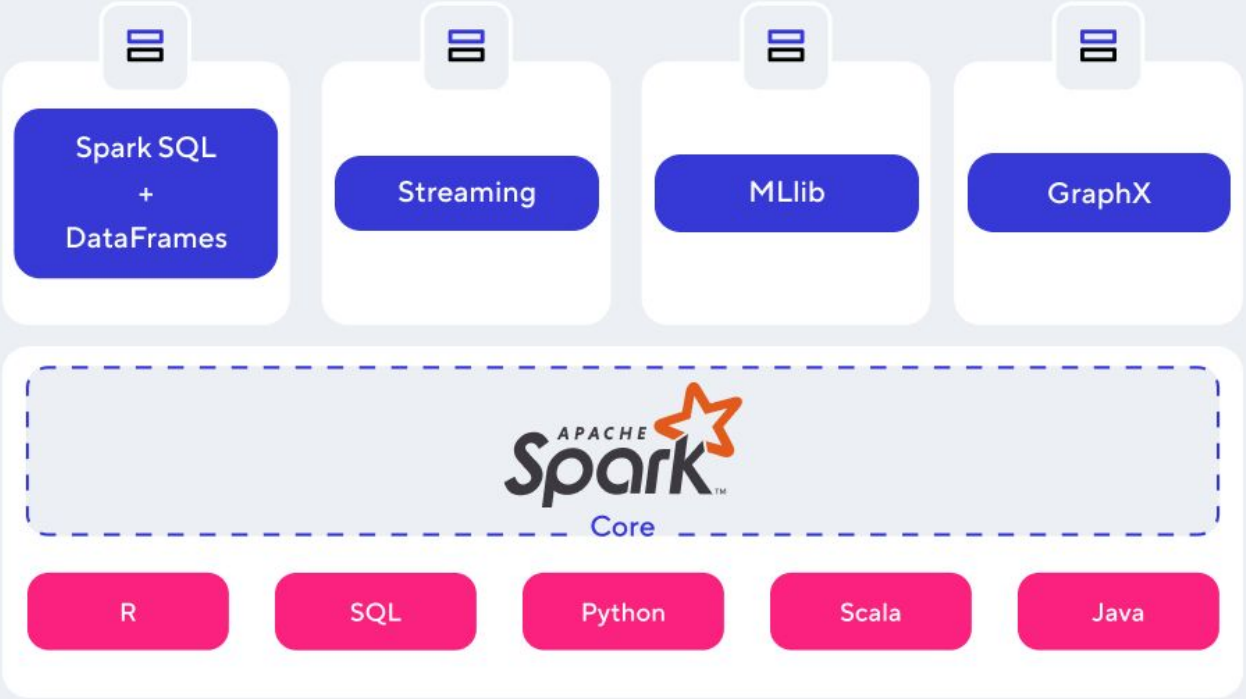
The Spark project was launched by Matei Zaharia AMPLab of Berkeley University in 2009, and its code became open in 2010 under the BSD license.

Apache Spark was created, smart problem solving

- Allows you to write code in different languages (Python)
- Very short code
- Good optimization (100 times faster than MapReduce), many additional libraries



Spark Ecosystem, what libraries Spark has

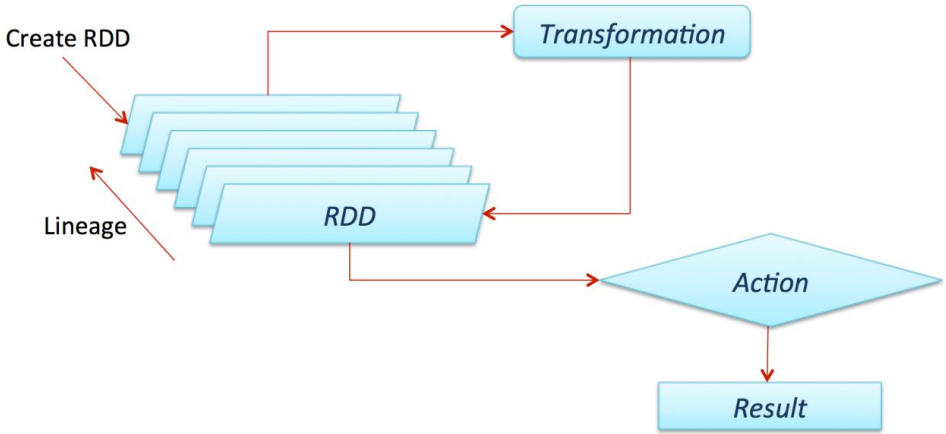


Spark RDD

resilient distributed dataset

is a simple, immutable, distributed collection of objects in the Apache Spark framework. RDD is a distributed dataset that is divided into many parts that are processed by different nodes in the cluster.

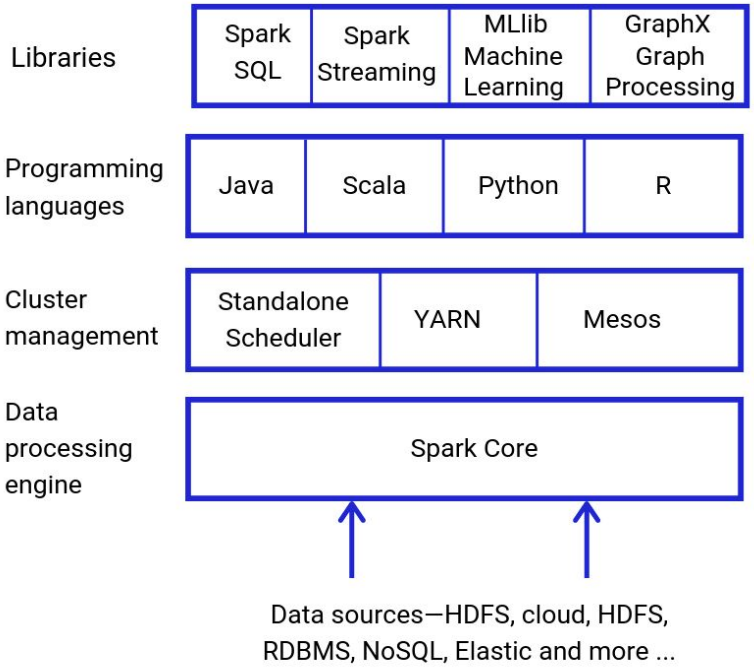
1. RDDs are fault-tolerant because they monitor data flow to automatically recover lost data in the event of failure
2. Distribute collection of JVM model
3. Not modifiable
4. Available in different programming languages (Python)



Spark Core

1. Connected to Data sources
2. Does all processing with RDD
3. Do cluster management
4. Connecting to spark UI
5. Support API of programming languages

Spark core:
It is the underlying processing engine that underpins the entire platform. The kernel interacts with storage systems, manages memory, schedules and distributes the load in the cluster. It is also responsible for API support of programming languages.

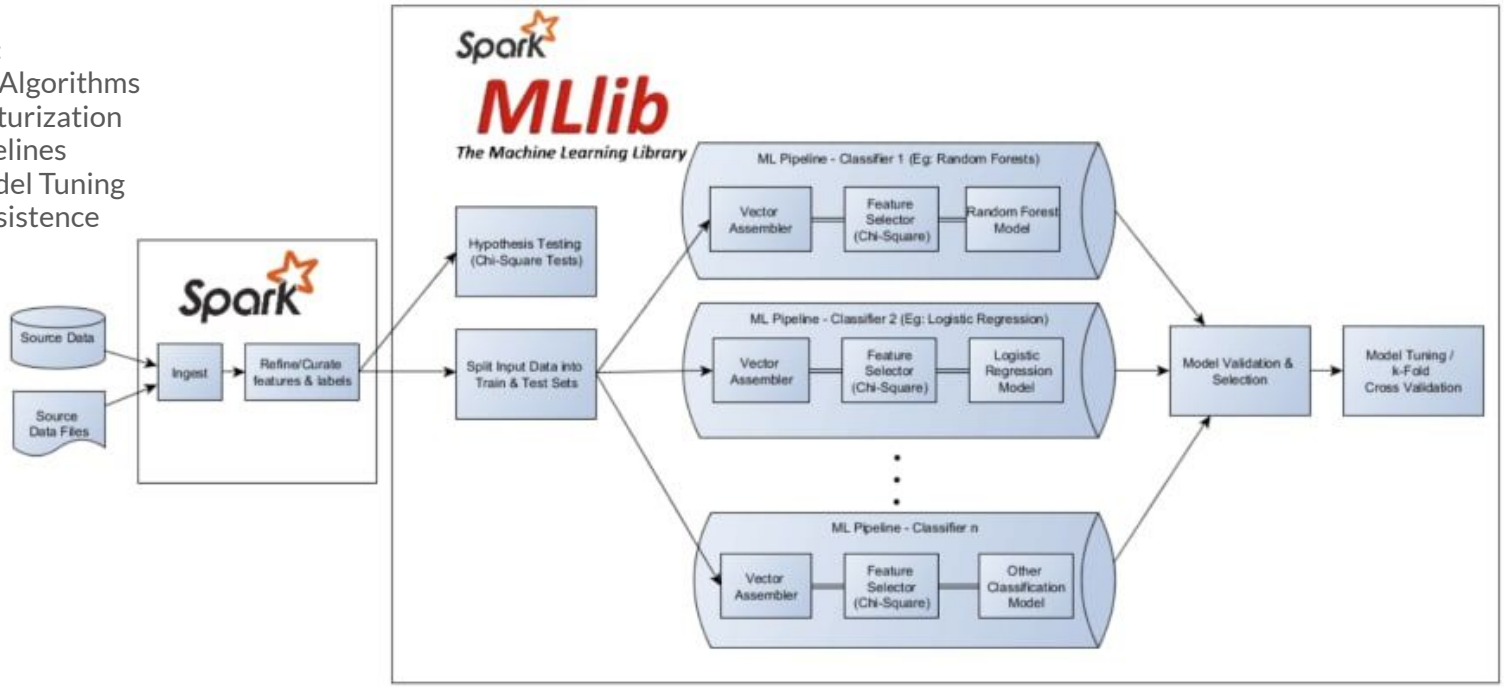


Spark ML Lib

Library that helps in managing and simplifying many of the machine learning models for building tasks, such as featurization, pipeline for constructing, evaluating and tuning of the model.

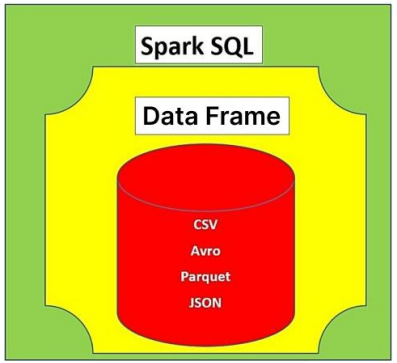
Spark core:

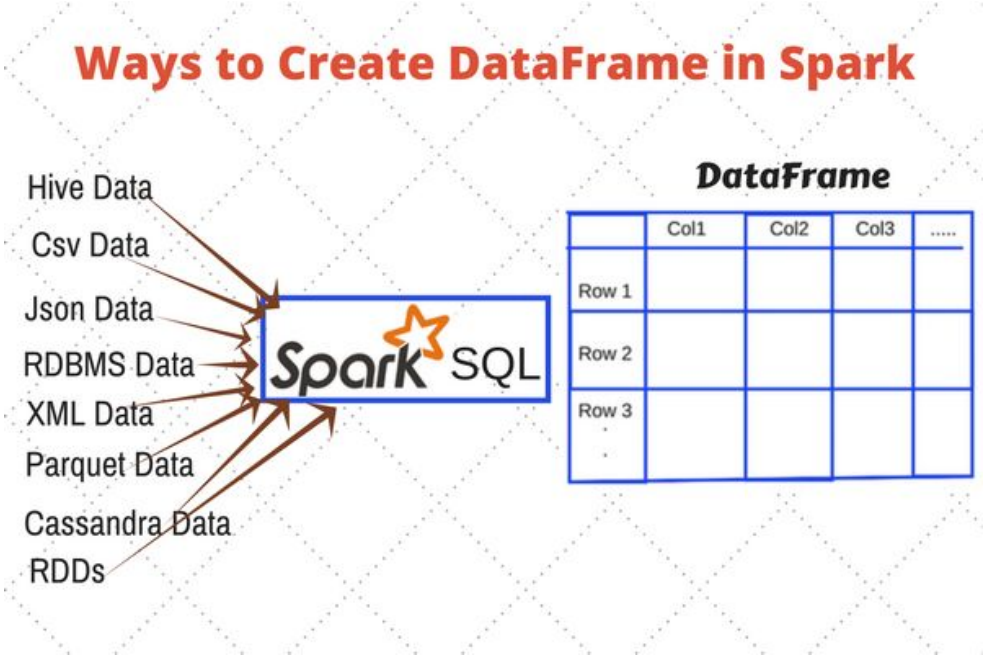
1. ML Algorithms
2. Featurization
3. Pipelines
4. Model Tuning
5. Persistence



Spark Data Frame

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

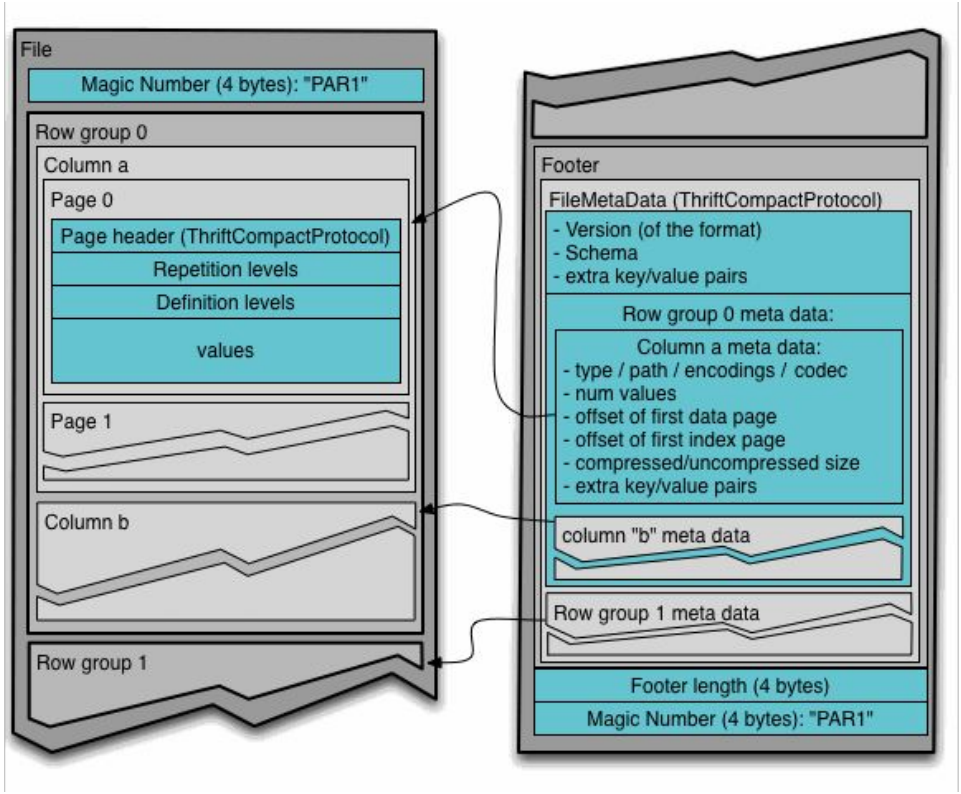




Parquet File Format

A binary, column-oriented big data storage format originally created for the Hadoop ecosystem, allowing you to take advantage of the compressed and efficient column-oriented presentation of information

Parquet

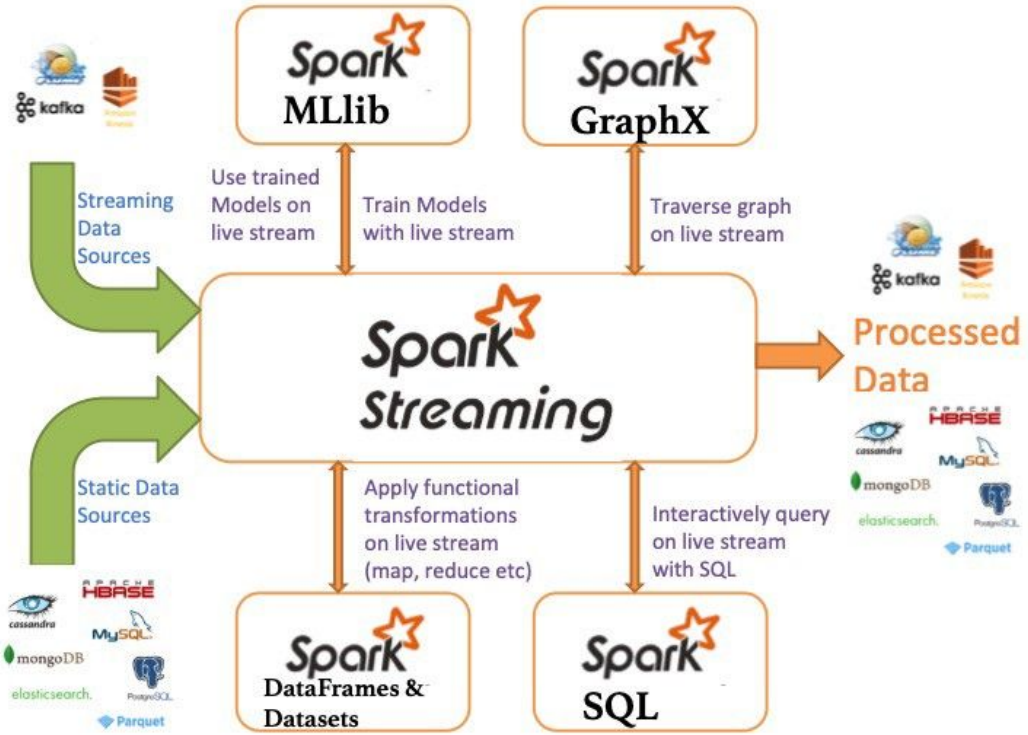


Spark Data Structures

	RDD	Data Frame	Data Set
Immutable	Yes	Yes	Yes
Schema	No	Yes	Yes
Optimizer engine	No	Catalyst	Catalyst
Language Support	Python, Java, Scala, R	Python, Java, Scala, R	Java, Scala
Level	Low	High (build on RDD)	High (DataFrame extension)
Date	2011 (Spark 1.0)	2013 (Spark 1.3)	2015 (Spark 1.6)
Memory Use	High	Medium	Low (Tungsten Optim.)
SQL query	No	Yes	Yes

Spark Streaming

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams



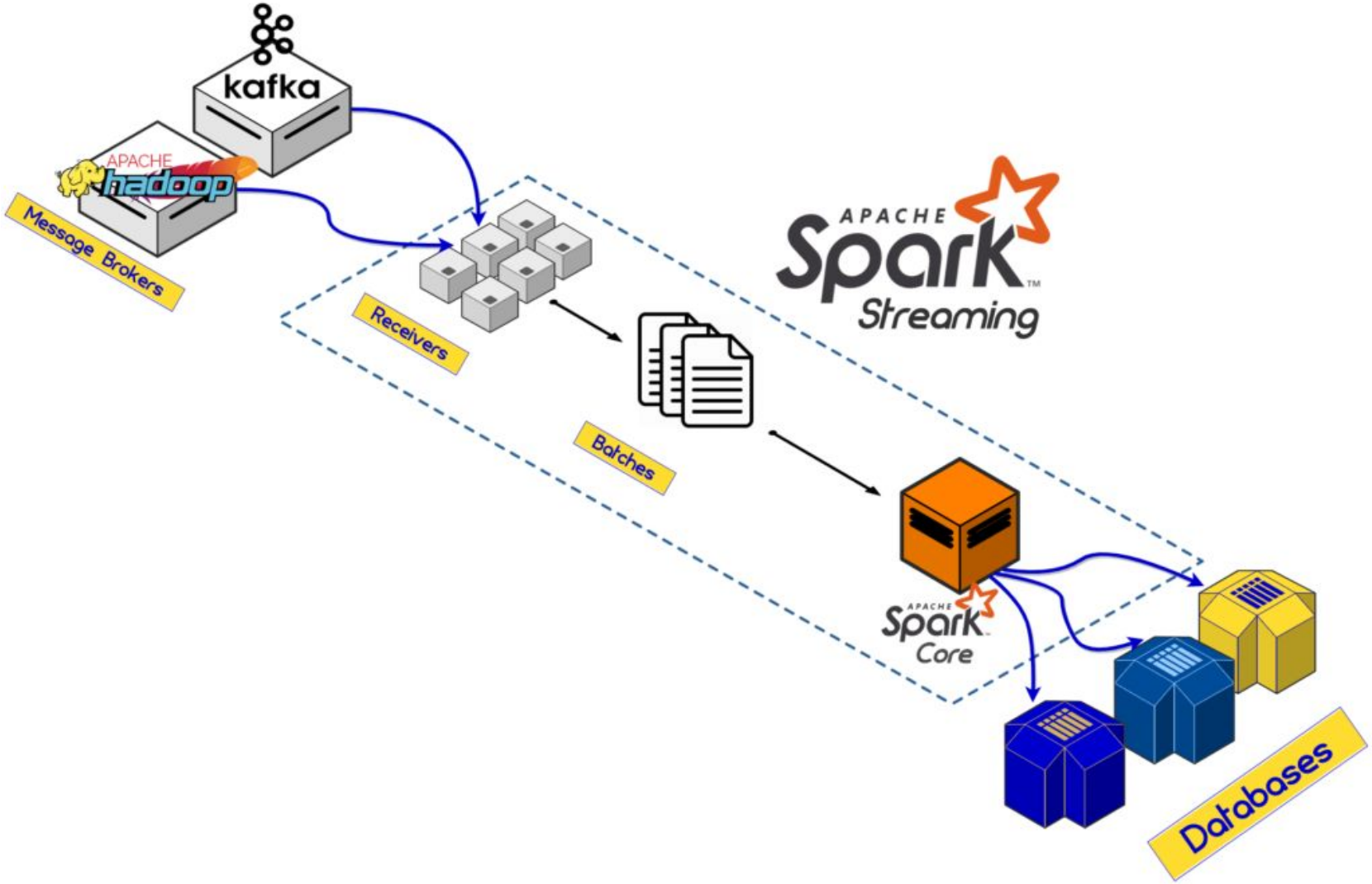
Spark Structure Streaming

Spark Structure Streaming

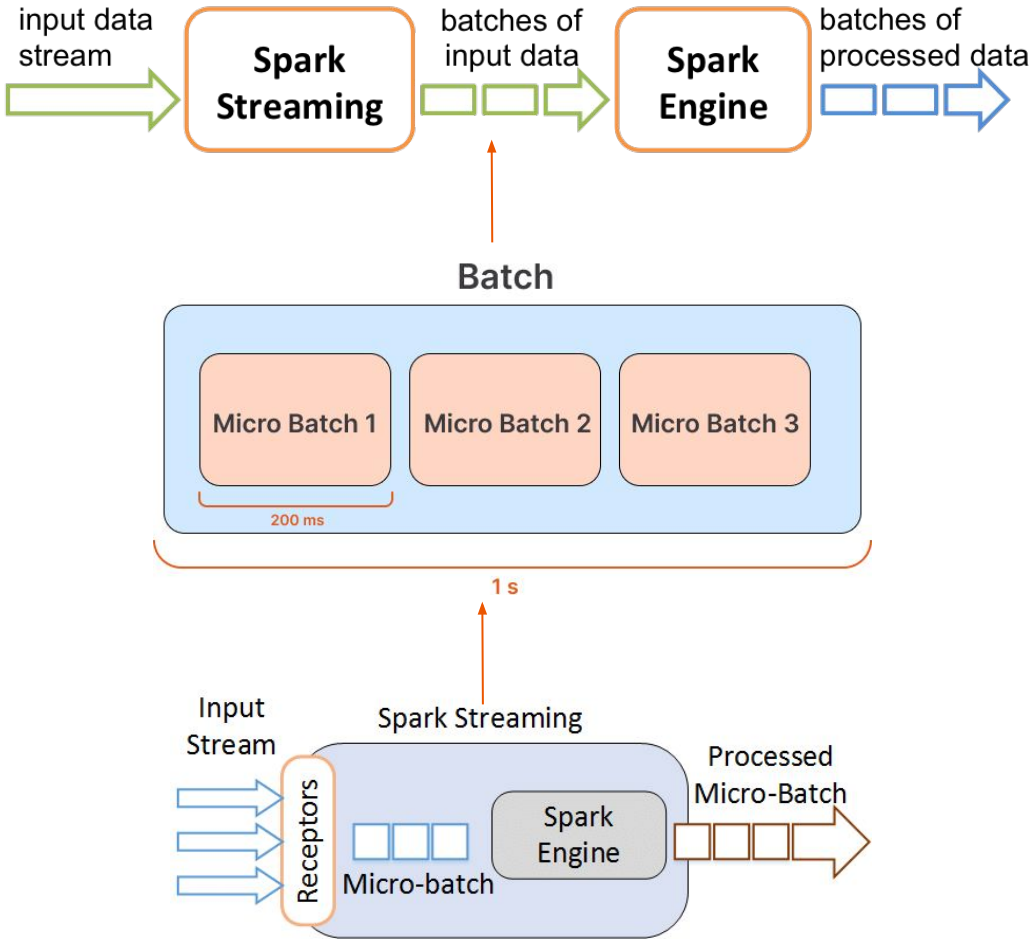
Structured Streaming is a scalable and fault-tolerant stream processing engine built on the Spark SQL engine. You can use the Dataset/DataFrame API in Scala, Java, Python or R to express streaming aggregations, event-time windows, stream-to-batch joins

Spark DStream

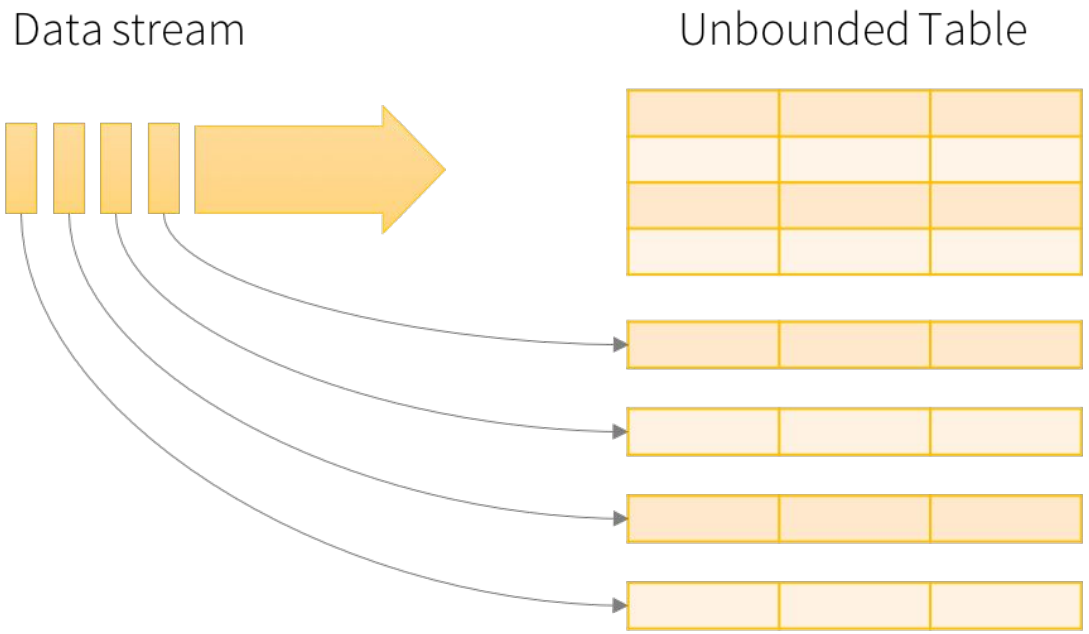
Spark Streaming is the previous generation of Spark's streaming engine. There are no longer updates to Spark Streaming and it's a legacy project. (Worked with RDD)


kafka
APACHE
hadoop
Message Brokers
Receivers
Batches
APACHE
Spark
Streaming
APACHE
Spark
Core
Databases

Spark Streaming Batch



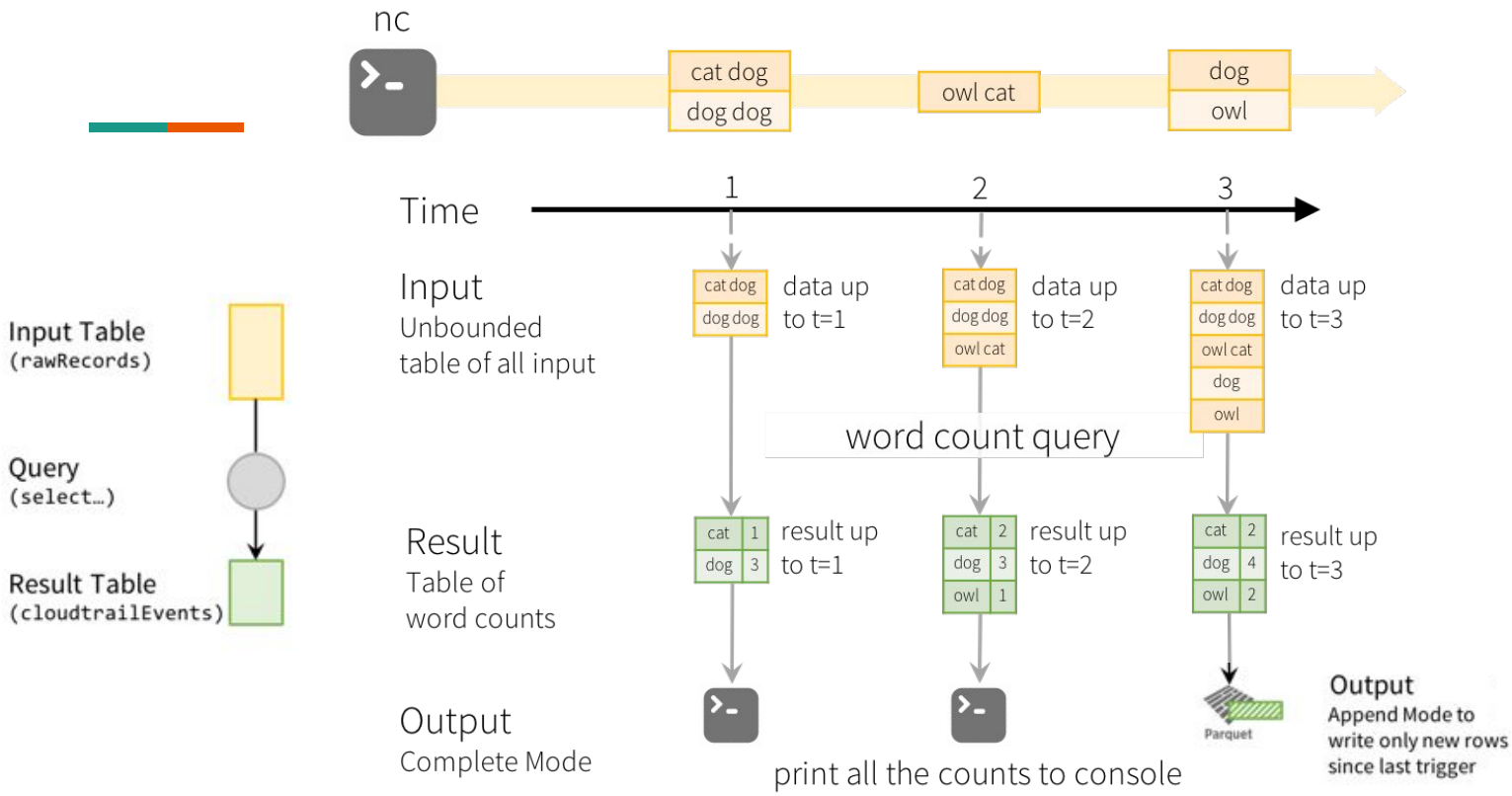
How stream works?



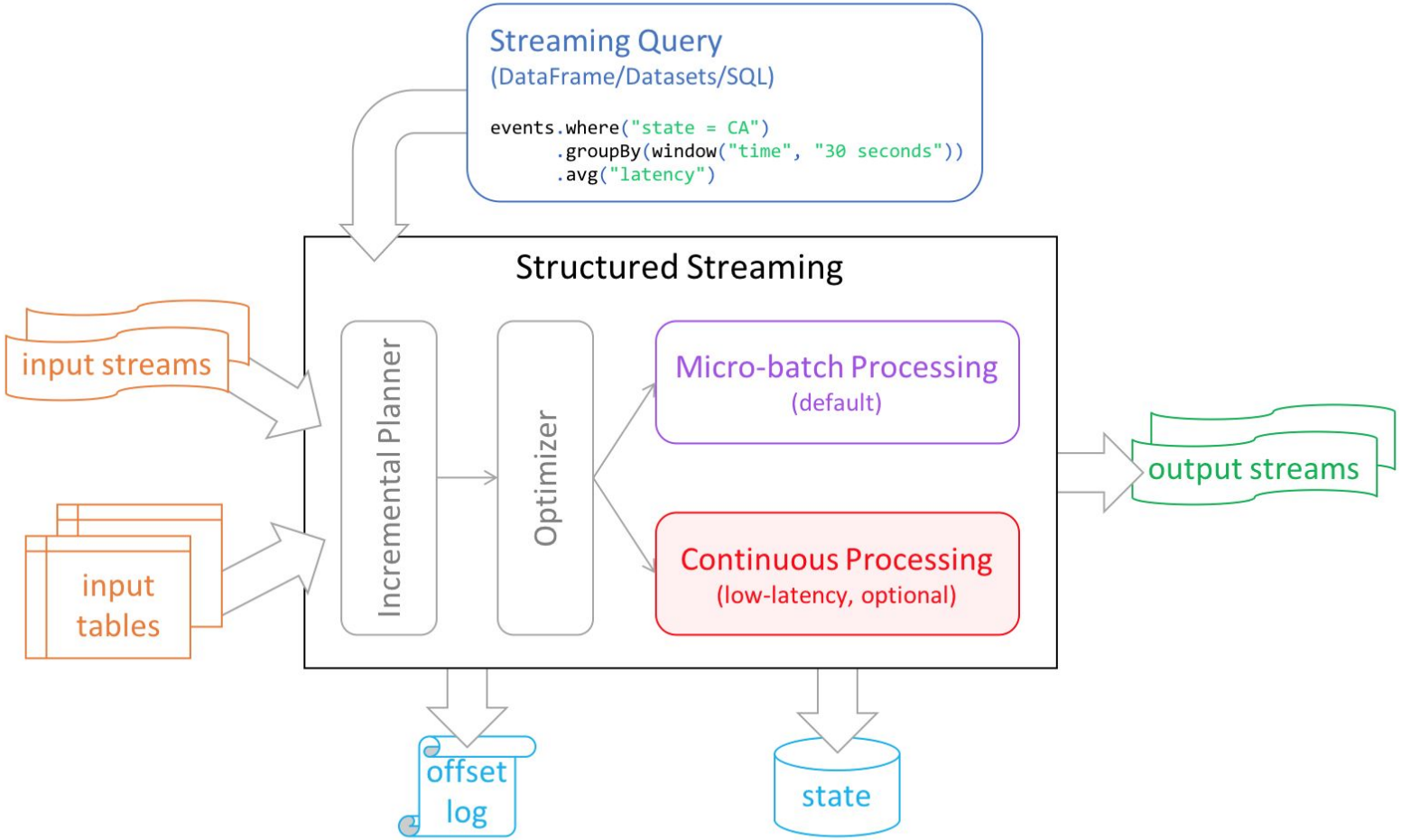
new data in the
data stream
=
new rows appended
to a unbounded table

Data stream as an unbounded table

Example of work



Model of the Quick Example



Structured Streaming processing modes: Micro-batch and Continuous Processing

SPARK ($FLR)

ECOSYSTEM

@stedas

 # OF ACCOUNTS

110,259

SET UP THE CLAIM

XRP BALANCE

23.5B

SINCE DEC 09, 2020

XRP HOLDINGS

694M

INCREASED

NEW XRP ACCOUNTS

51K

CREATED

Flare Time Series Oracle (FTSO) is the decentralised on-chain mechanism providing time series estimates to the network

FTSO ORACLE

0100 1011 0100 1011

DATA PROVIDER

DATA PROVIDER

Different participants provide data to the FTSO oracles

FUSION

TOKEN HOLDERS

AGENT

F-ASSET AGENTS

AGENT

Stake Spark (FLR)
Earns underlying Asset
of F-Asset like FXRP, FUSD, FLTC

F-Asset is a 1:1 representation on Flare of any token from another chain. The F-Asset protocol is designed to allow any asset from any chain to be used with full Ethereum style smart contracts on Flare without relying on a centralized third party.

GOVERNANCE
VOTES AND PROPOSALS

FLARE FOUNDATION

Delegated votes from FLR owners only

FTSO VOTE

Detachable from Spark (FLR) Token
(Earn more Spark - FLR)

SPARK ECOSYSTEM

GOVERNANCE VOTE

Detachable from Spark (FLR) Token

FLARE FINANCE
DeFi Dapps

YIELD FLARE

TOKEN

YieldFlare Token (YFLR) is the ecosystem's main platform token. It will be utilized across all products and has a main focus on utilization within FlareX and FlareFarm.

TOTAL SUPPLY
110,000,000 YFLR

YIELD FIN

TOKEN

YieldFin Token (YFIN) is the ecosystem's main governance token. It will be utilized across all 6 products within the ecosystem with a primary focus on governance and reward structure.

TOTAL SUPPLY
11,000 YFIN

YieldFlare (YFLR)
YieldFin (YFIN)
YieldMine (YMIN)

3 TOKENS

YieldMine Token (YMIN) is the ecosystem's main rewards token. It will be utilized primarily on the FlareMine and FlareX platforms

6 DEFI PRODUCTS

FlareX
FlareFarm
FlareUSD
FlareLoans
FlareMutual
FlareMine

OTHER APPS Built on Flare*

? ? ?

* Other dapps to be announced

CLAIM RATIO

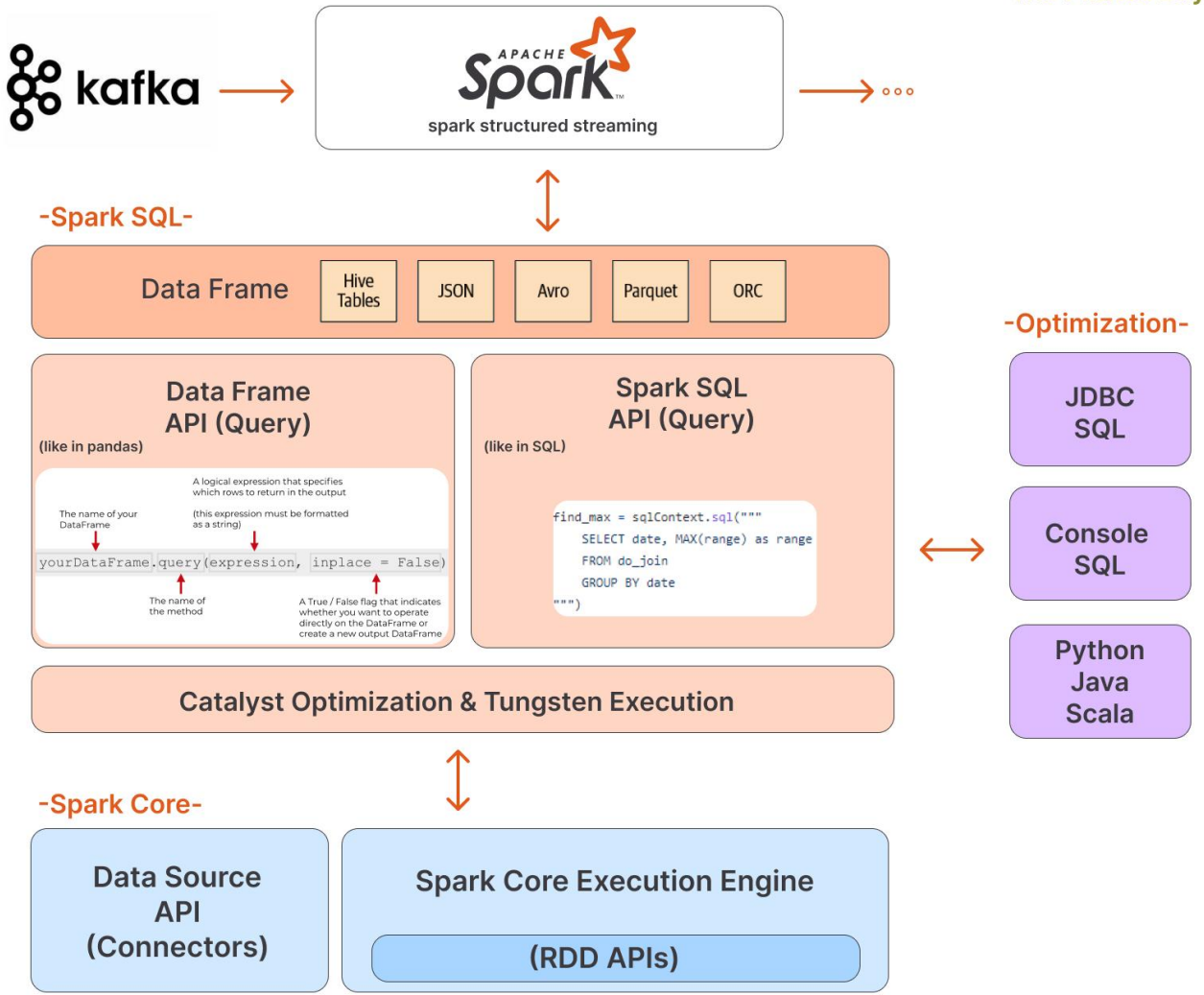
XRP : FLR

1=1.0073

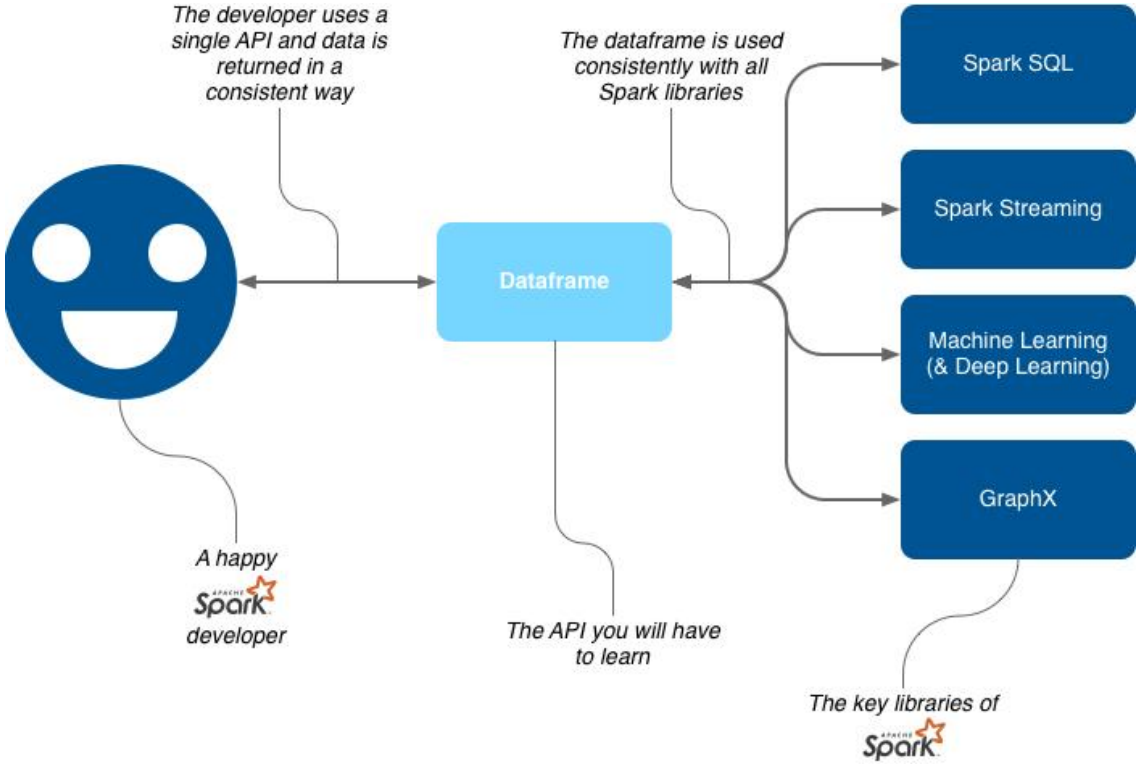
Spark Distribution
At network launch, each account will receive 15% of the total Spark (Q1/Q2 2021). The remaining Spark will be distributed over the next 2-3 years.

Spark SQL

Spark SQL is a Spark module for structured data processing. It provides a programming abstraction called DataFrames and can also act as a distributed SQL query engine. It enables unmodified Hadoop Hive queries to run up to 100x faster on existing deployments and data.



What do I need to learn from this?



Sample PySpark code example, word count

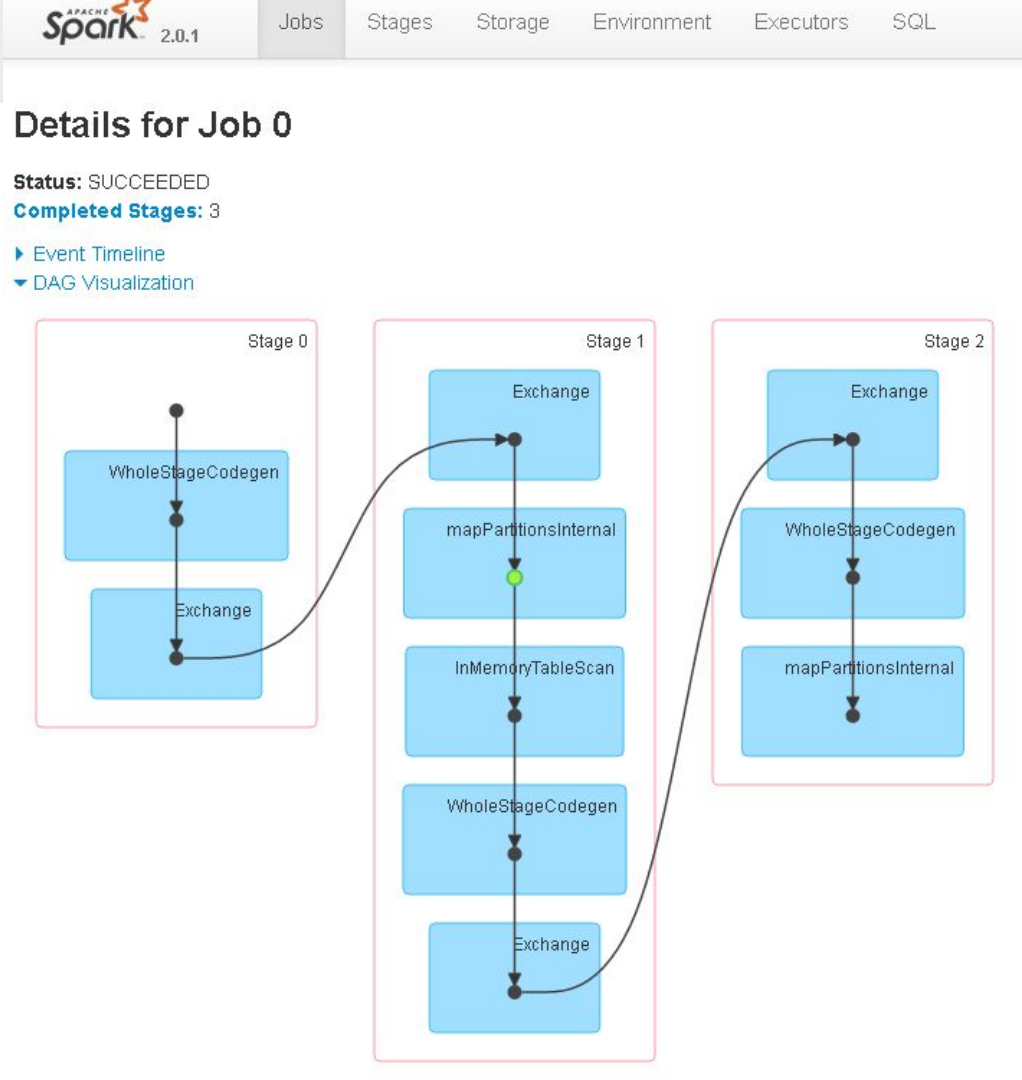
```
import sys
from pyspark import SparkContext, SparkConf

if __name__ == "__main__":

        # create Spark context with necessary configuration
        sc = SparkContext("local","PySpark Word Count Exmaple")

        # read data from text file and split each line into words
        words = sc.textFile("D:/workspace/spark/input.txt").flatMap(lambda line: line.split(" "))

        # count the occurrence of each word
        wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda a,b:a +b)

        # save the counts to output

        wordCounts.saveAsTextFile("D:/workspace/spark/output/")
```

Sample SparkSQL code example, word count

```
# create a silly test dataframe from Python collections (lists)
wordsDF = sqlContext.createDataFrame([('look',), ('spark',), ('tutorial',) , ('spark',), ('look', ), ('python', )], ['word'])

wordsLengthsDF = wordsDF.select(length('word').alias('lengths')) # transformation

wordCountsDF = (words DF.groupBy('word').count())

wordCountsDF.show()
```

Spark UI

The web interface of a running Spark application to monitor and inspect Spark job executions in a web browser



Show Additional Metrics

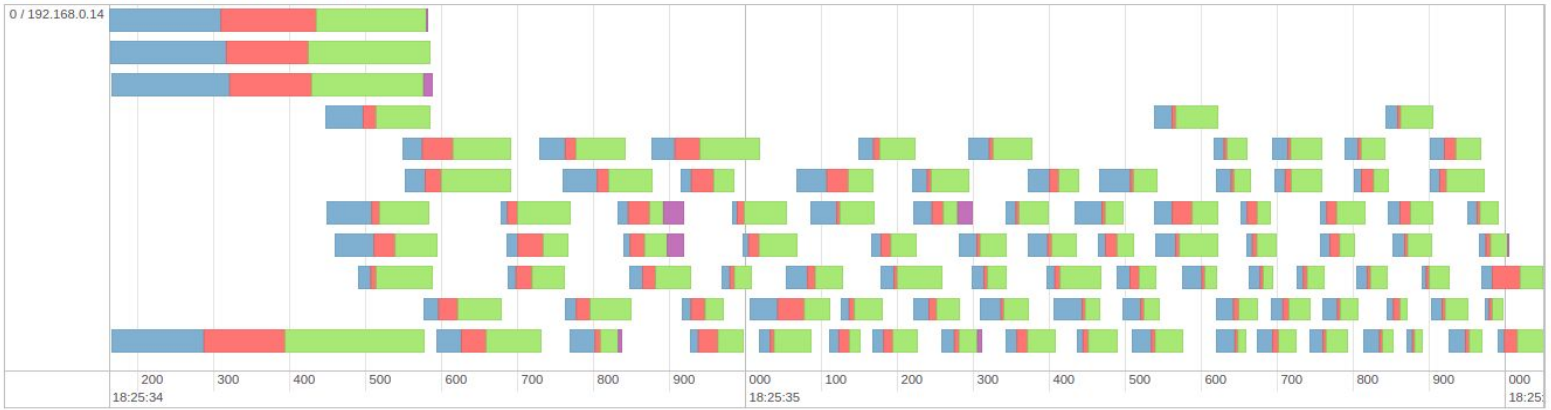
- Select All
- Scheduler Delay
- Task Deserialization Time
- Shuffle Read Blocked Time
- Shuffle Remote Reads
- Result Serialization Time
- Getting Result Time
- Peak Execution Memory

Event Timeline

- Enable zooming

Scheduler Delay
Task Deserialization Time
Shuffle Read Time
Executor Computing Time
Shuffle Write Time
Result Serialization Time
Getting Result Time

Tasks: 200. 2 Pages. Jump to 1 . Show 100 items in a page. Go



Summary Metrics for 200 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Task Deserialization Time	2.0 ms	4.0 ms	5.0 ms	13.0 ms	0.1 s
Duration	6.0 ms	16.0 ms	24.0 ms	36.0 ms	0.2 s
GC Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	11.0 ms
Result Serialization Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	27.0 ms
Getting Result Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	0.0 ms
Scheduler Delay	5.0 ms	11.0 ms	16.0 ms	23.0 ms	0.2 s
Peak Execution Memory	8.1 MiB	8.1 MiB	8.1 MiB	8.1 MiB	8.1 MiB
Shuffle Read Size / Records	10.6 KiB / 872	11.7 KiB / 968	12.1 KiB / 1002	12.4 KiB / 1034	13.3 KiB / 1118
Shuffle Read Blocked Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	14.0 ms
Shuffle Remote Reads	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B

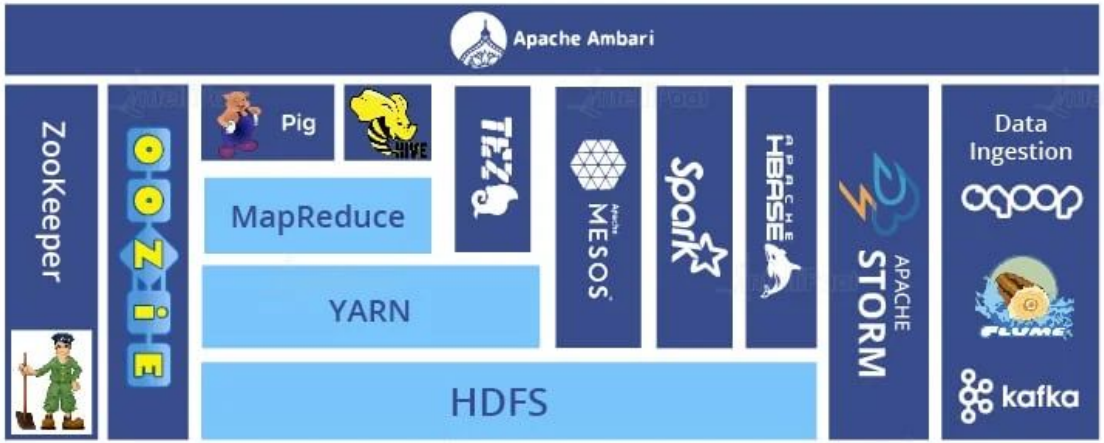
Showing 1 to 10 of 10 entries

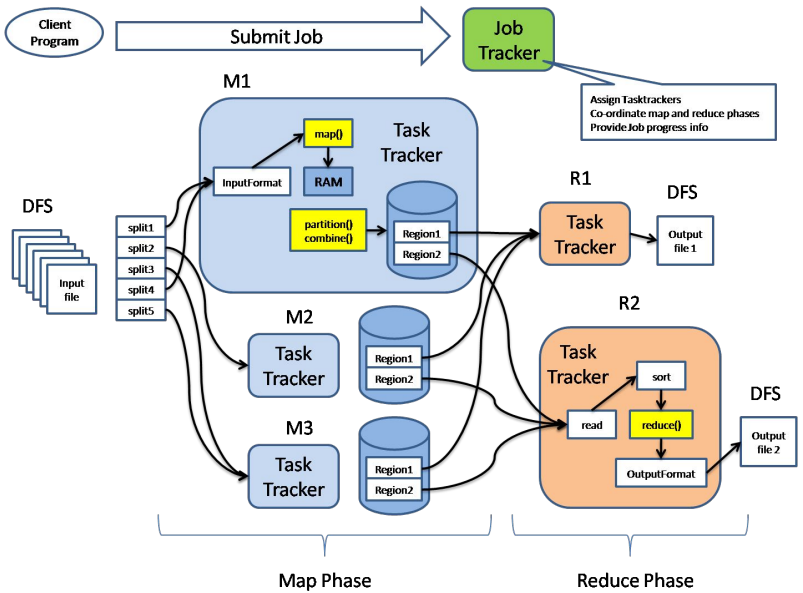
What we haven't talked about

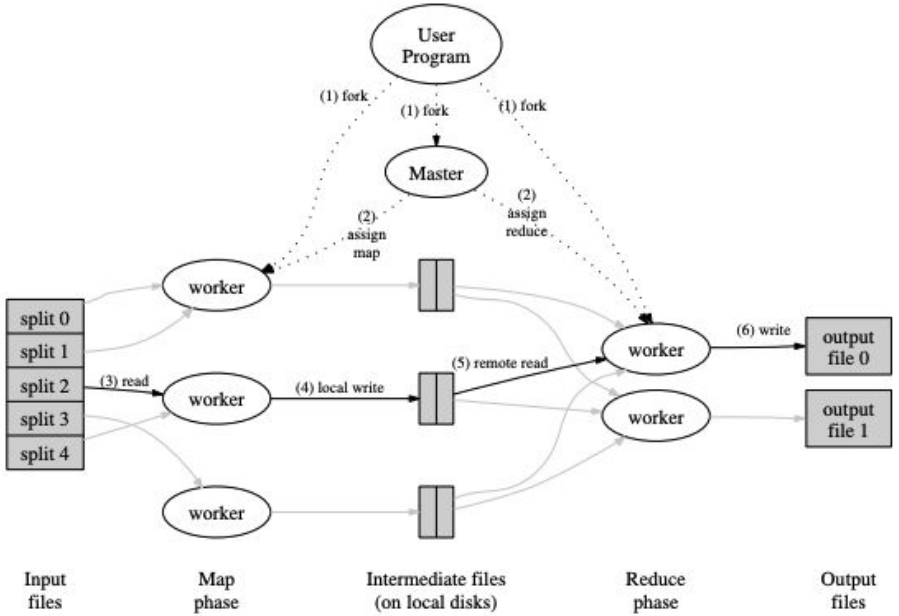
1. Lazy Evolution
2. Efficiency/Memory use
3. Executor/Worker Models
4. Catalyst Optimizer
5. Tungsten
6. Driver
7. ClusterManager
8. Fault Tolerance
9. Dependency Types
10. Stages and Tasks

Thanks!

Core Hadoop Ecosystem




Client Program
Submit Job
Job Tracker
Assign Tasktrackers
Co-ordinate map and reduce phases
Provide Job progress info
M1
map()
Task Tracker
InputFormat
RAM
partition()
combine()
Region1
Region2
DFS
Input file
split1
split2
split3
split4
split5
R1
Task Tracker
DFS
Output file 1
M2
Task Tracker
Region1
Region2
R2
Task Tracker
sort
read
reduce()
OutputFormat
DFS
Output file 2
M3
Task Tracker
Region1
Region2
Map Phase
Reduce Phase


User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
worker
worker
worker
worker
split 0
split 1
split 2
split 3
split 4
(3) read
(4) local write
(5) remote read
(6) write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files