Programming with Frameworks

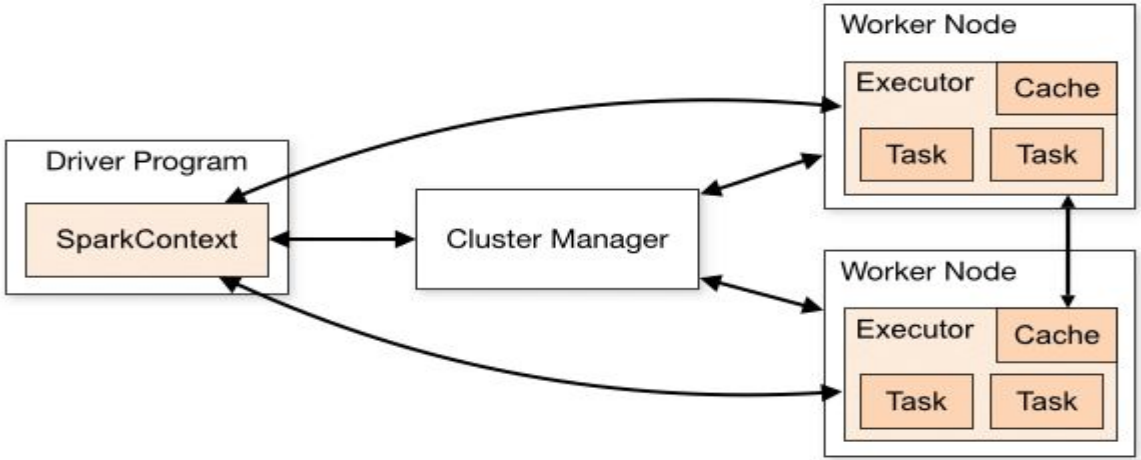
Not reinventing the wheel

Concentrate your effort on algorithms not on implementation

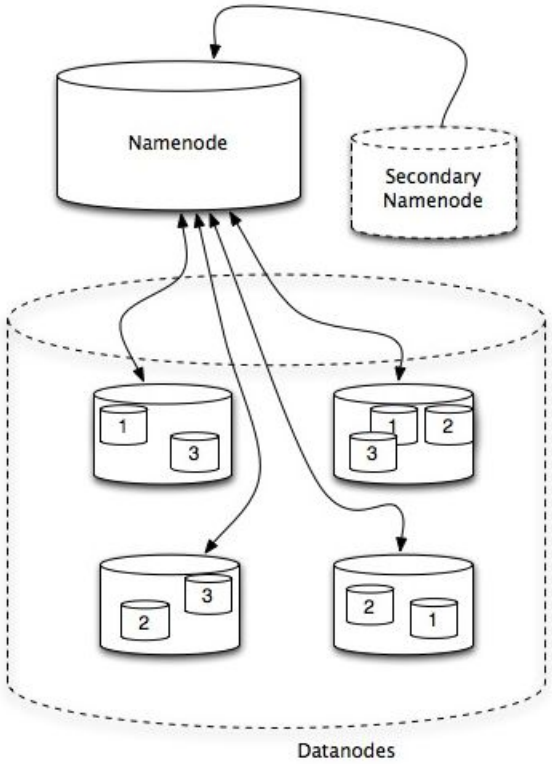
Spark is one of Hadoop's sub projects developed in 2009 in UC Berkeley's AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

An open-source cluster computing framework for real-time processing.

“ A computer cluster is a set of loosely or tightly connected computers that work together they can be viewed as a single system. “



Hadoop Distributed File System



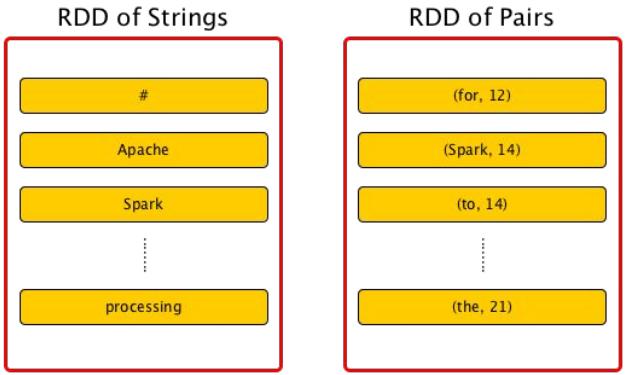
Abstraction of HDFS

RDD — Resilient Distributed Dataset

- Immutable
- In-Memory (for as long as possible)
- Lazy Evaluated
- Cacheable
- Type-safe

Various Types

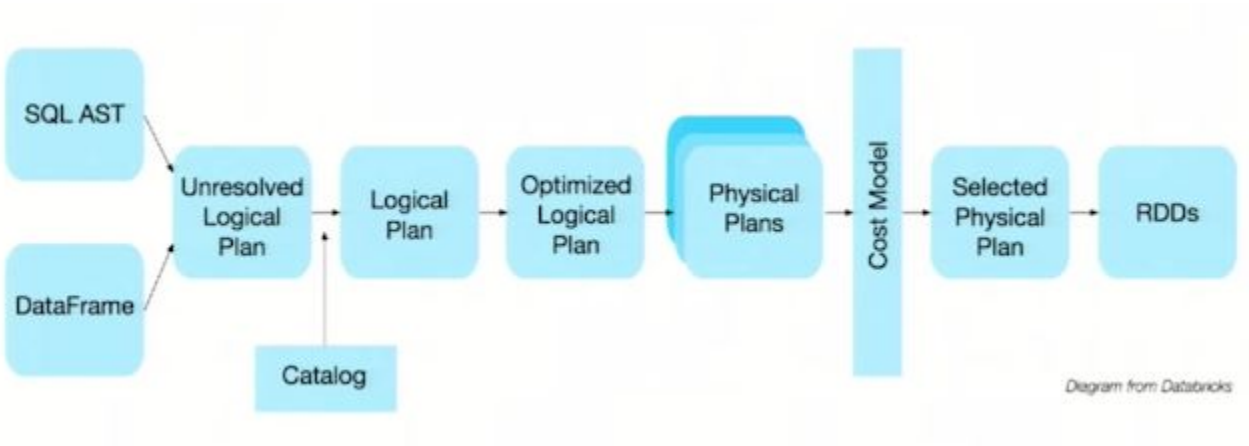
- Local vector
- Labeled point
- Local matrix
- Distributed matrix



Abstraction of HDFS

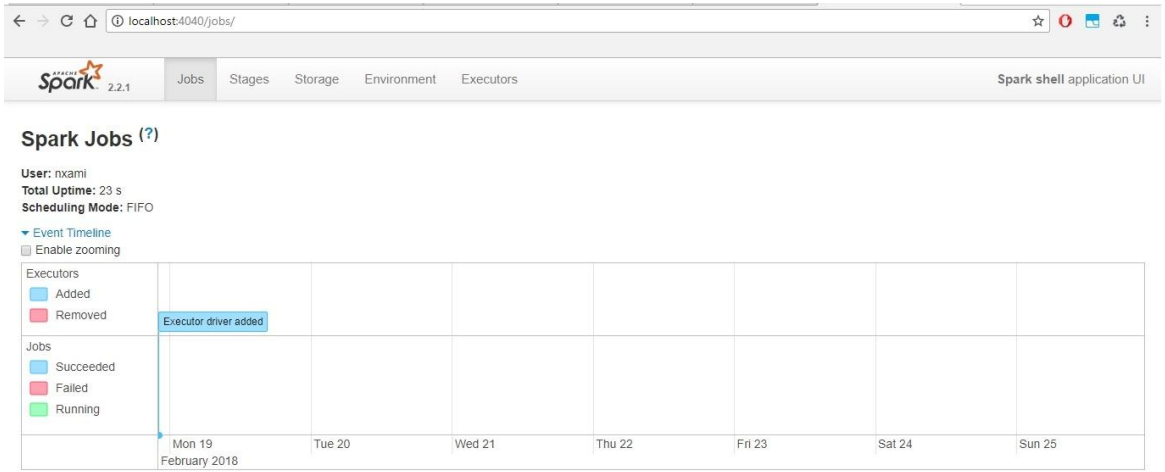
Dataframe

- Abstraction which gives a schema view of data
  - Custom Memory management
  - Optimized Execution Plans (***Catalyst Optimizer)***
  - Lazy Evaluation



Spark local (pseudo-cluster)

In this non-distributed single-JVM deployment mode, Spark spawns all the execution components - **driver**, **executor**, **LocalSchedulerBackend**, and **master** in the same single JVM.



Example CSV-SQL

```scala
object SparkSQL {
  def main(args: Array[String]): Unit = {
    val sc = new SparkContext("Csv loading example")
    val sqlContext = new org.apache.spark.sql.SQLContext(sc)
    val baby_names = sqlContext.read.format("com.databricks.spark.csv")
     .option("header", "true")
     .option("inferSchema", "true")
     .load(args(0))
     baby_names.printSchema
  }
}
```

Source CSV

```
Year,First Name,County,Sex,Count
2013,GAVIN,ST LAWRENCE,M,9
2013,LEVI,ST LAWRENCE,M,9
2013,LOGAN,NEW YORK,M,44
2013,HUDSON,NEW YORK,M,49
2013,GABRIEL,NEW YORK,M,50
2013,THEODORE,NEW YORK,M,51
2013,ELIZA,KINGS,F,16
2013,MADELEINE,KINGS,F,16
2013,ZARA,KINGS,F,16
2013,DAISY,KINGS,F,16
2013,JONATHAN,NEW YORK,M,51
2013,CHRISTOPHER,NEW YORK,M,52
```

Schema

```
root
|-- Year: integer (nullable = true)
|-- First Name: string (nullable = true)
|-- County: string (nullable = true)
|-- Sex: string (nullable = true)
|-- Count: integer (nullable = true)
```

Example CSV-SQL

Sort By popularity

```
def popularNames() : Unit  = {
    val popular_names = sqlContext.sql("""
      select distinct (`First Name`),count(County) as cnt
      from names
      group by `First Name`
      order by cnt"""
    )
   popular_names.collect.foreach(println)
}
```

```
[JACOB,284]
[EMMA,284]
[LOGAN,270]
[OLIVIA,268]
[ISABELLA,259]
[SOPHIA,249]
[NOAH,247]
[MASON,245]
[ETHAN,239]
[AVA,234]
```

Sorting by size

		Spark 100 TB *	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400	6592	6080
# Reducers	10,000	29,000	250,000
Rate	1.42 TB/min	4.27 TB/min	4.27 TB/min
Rate/node	0.67 GB/min	20.7 GB/min	22.5 GB/min
Sort Benchmark Daytona Rules	Yes	Yes	No
Environment	dedicated data center	EC2 (i2.8xlarge)	EC2 (i2.8xlarge)

Hadoop MapReduce 2009 for 1 PB of data 16 hours on 3800 machines