

# **Big data: architectures and data analytics**

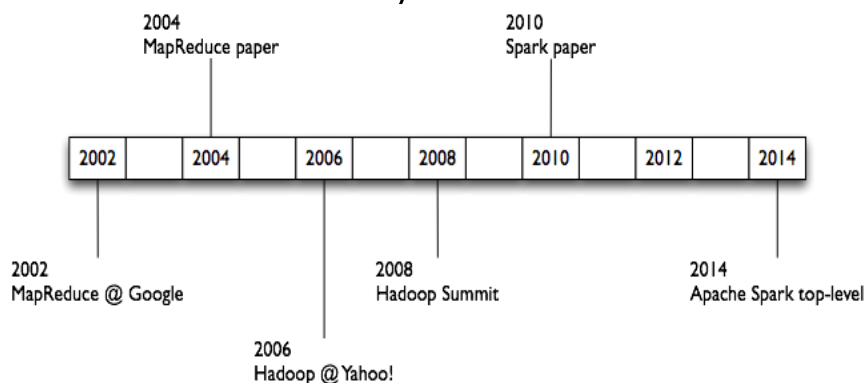
## **Introduction to Spark**

# Spark

- Apache Spark™ is a fast and general engine for large-scale data processing
- Spark aims at achieving the following goals in the Big data context
  - Generality: diverse workloads, operators, job sizes
  - Low latency: sub-second
  - Fault tolerance: faults are the norm, not the exception
  - Simplicity: often comes from generality

## Spark History

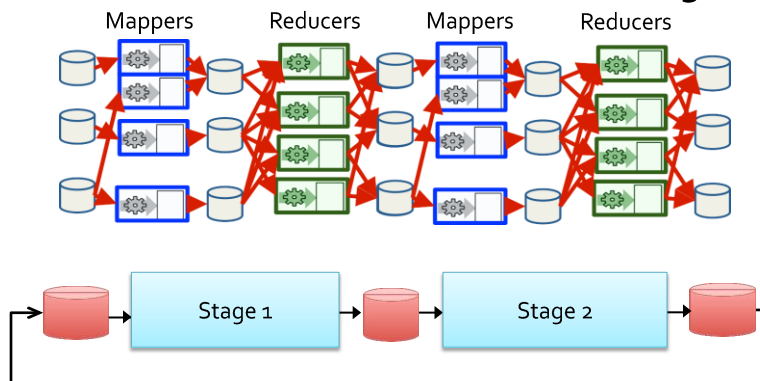
- Originally developed at the University of California - Berkeley's AMPLab



## Spark: Motivations

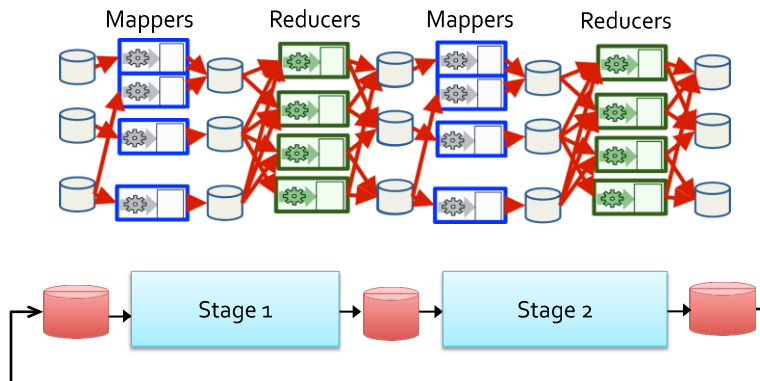
### MapReduce and Iterative Jobs

- Iterative jobs, with MapReduce, involve a lot of disk I/O for each iteration and stage



## MapReduce and Iterative Jobs

- Disk I/O is very slow (even if it is a local I/O)

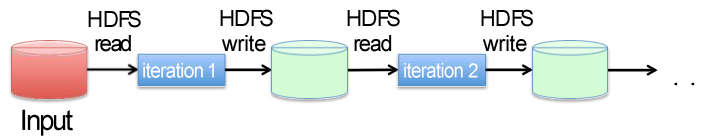


## Apache Spark: Motivation and Opportunity

- Motivation
  - Using MapReduce for complex **iterative jobs** or **multiple jobs on the same data** involves lots of disk I/O
- Opportunity
  - The **cost of main memory decreased**
    - Hence, large main memories are available in each server
- Solution
  - Keep **more data in main memory**
    - Basic idea of Spark

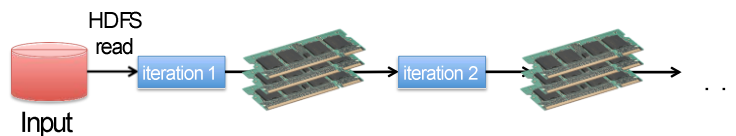
## From MapReduce to Spark

- MapReduce: Iterative job



## From MapReduce to Spark

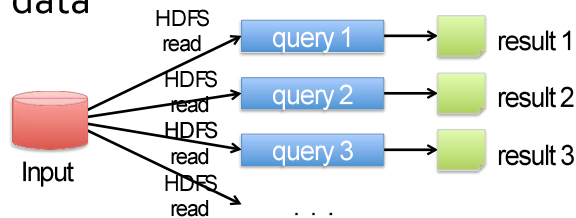
- Spark: Iterative job



- Data are shared between the iterations by using the main memory
  - Or at least part of them
- 10 to 100 times faster than disk

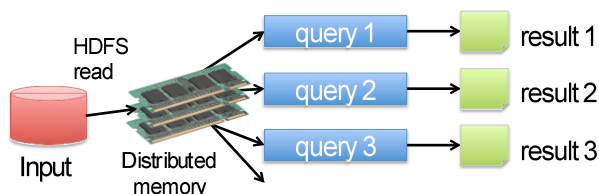
## From MapReduce to Spark

- MapReduce: Multiple analyses of the same data



## From MapReduce to Spark

- Spark: Multiple analyses of the same data



- Data are read only once from HDFS and stored in main memory
  - Split of the data across the main memory of each server

## Spark: Resilient Distributed Data sets (RDDs)

- Data are represented as Resilient Distributed Datasets (RDDs)
  - Partitioned/Distributed collections of objects spread across the nodes of a clusters
  - Stored in main memory (when it is possible) or on local disk
- Spark programs are written in terms of operations on resilient distributed data sets

## Spark: Resilient Distributed Data sets (RDDs)

- RDDs are built and manipulated through a set of parallel
  - Transformations
    - map, filter, join, ...
  - Actions
    - count, collect, save, ...
- RDDs are automatically rebuilt on machine failure

## Spark Computing Framework

- Provides a programming abstraction (based on RDDs) and transparent mechanisms to execute code in parallel on RDDs
  - Hides complexities of fault-tolerance and slow machines
  - Manages scheduling and synchronization of the jobs

## MapReduce vs Spark

	Hadoop Map Reduce	Spark
Storage	Disk only	In-memory or on disk
Operations	Map and Reduce	Map, Reduce, Join, Sample, etc...
Execution model	Batch	Batch, interactive, streaming
Programming environments	Java	Scala, Java, R, and Python



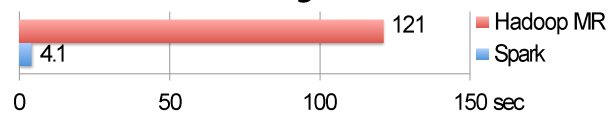
## MapReduce vs Spark

- Lower overhead for starting jobs
- Less expensive shuffles

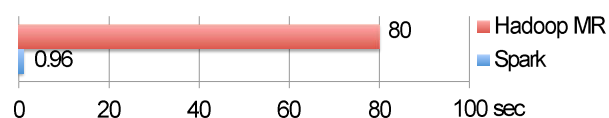
## In-Memory RDDs Can Make a Big Difference

- Two iterative Machine Learning algorithms:

- K-means Clustering



- Logistic Regression



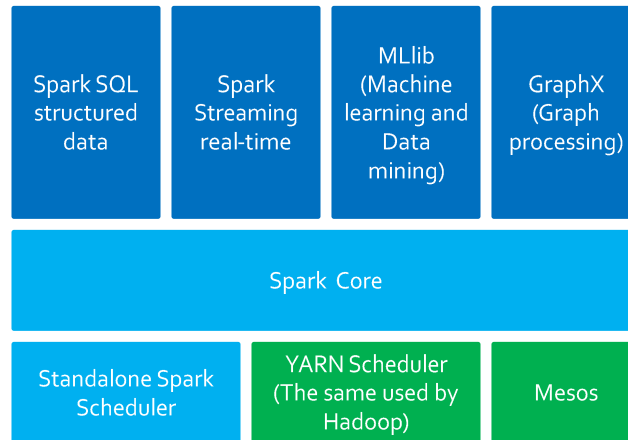
# Petabyte Sort Challenge

	<b>Hadoop MR Record</b>	<b>Spark Record</b>	<b>Spark 1 PB</b>
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
<b>Sort rate</b>	<b>1.42 TB/min</b>	<b>4.27 TB/min</b>	<b>4.27 TB/min</b>
<b>Sort rate/node</b>	<b>0.67 GB/min</b>	<b>20.7 GB/min</b>	<b>22.5 GB/min</b>

[Daytona Gray](#)  
100 TB sort  
benchmark  
record (tied  
for 1<sup>st</sup> place)

## Spark: Main components

## Spark Components



## Spark Components

- Spark is based on a basic component (the Spark Core component) that is exploited by all the high-level data analytics components
  - This solution provides a more uniform and efficient solution with respect to Hadoop where many non-integrated tools are available
- When the efficiency of the core component is increased also the efficiency of the other high-level components increases

## Spark Components

- Spark Core
  - Contains the basic functionalities of Spark exploited by all components
    - Task scheduling
    - Memory management
    - Fault recovery
    - ...
  - Provides the API that are used to create RDDs and apply transformations and actions on them

23

## Spark Components

- Spark SQL structured data
  - This component that is used to interact with structured datasets by means of SQL
  - It supports also
    - Hive Query Language (HQL)
  - It interacts with many data sources
    - Hive Tables
    - Parquet
    - JSON

24

## Spark Components

- Spark Streaming real-time
  - It is used to process live streams of data in real-time
  - The APIs of the Streaming real-time components operated on RDDs and are similar to the ones used to process standard RDDs associated with "static" data sources

25

## Spark Components

- MLlib
  - It is a machine learning/data mining library
  - It can be used to apply the parallel versions of many machine learning/data mining algorithms
    - Data preprocessing and dimensional reduction
    - Classification algorithms
    - Clustering algorithms
    - Itemset mining
    - ....

26

## Spark Components

- GraphX
  - A graph processing library
  - Provides many algorithms for manipulating graphs
    - Subgraph searching
    - PageRank
    - ....

27

## Spark Schedulers

- Spark can exploit many schedulers to execute its applications
  - HadoopYARN
    - Standard scheduler of Hadoop
  - Mesos cluster
    - Another popular scheduler
  - Standalone Spark Scheduler
    - A simple cluster scheduler included in Spark

28