FIRST STEPS ON APACHE SPARK

BIG DATA II

Jesús Maillo (jesusmh@decsai.ugr.es)

08/03/2018

Outline

□ **Apache Hadoop VS Apache Spark**

□ Weaknesses of Hadoop

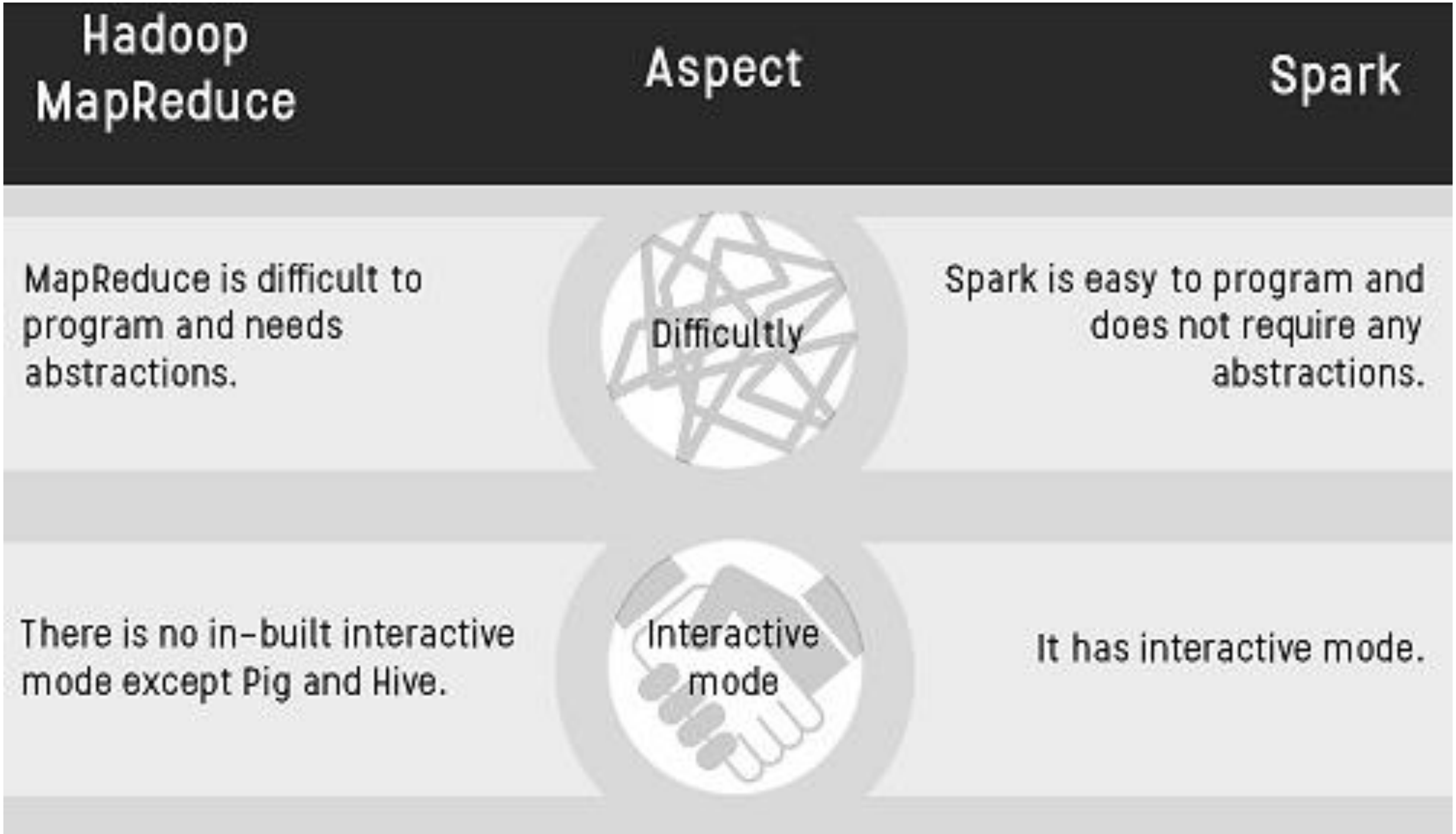
□ Apache Spark

□ MLlib: Machine Learning on Spark

□ ML: API on top of DataFrame

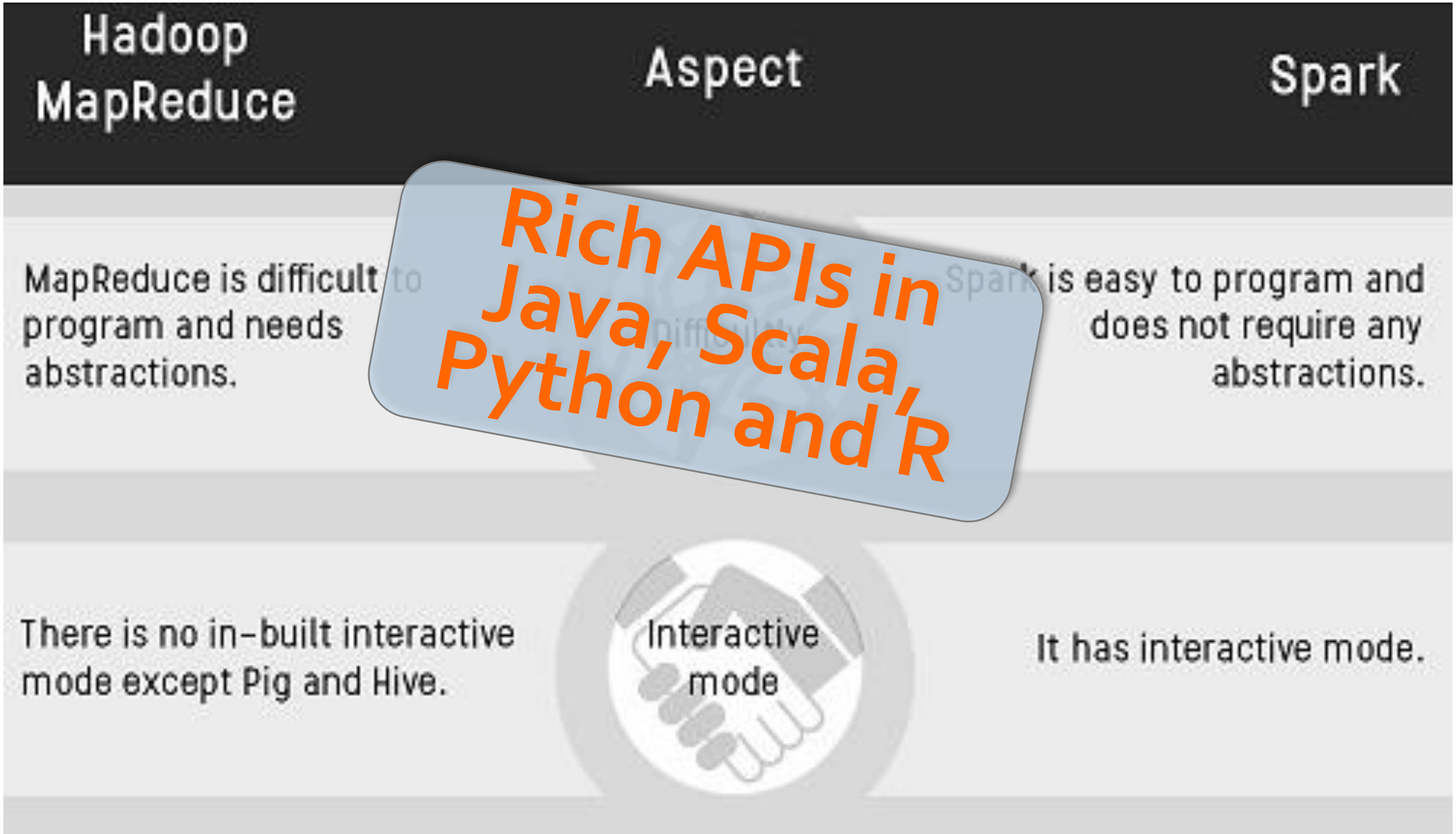
□ SparkR & PySpark

Hadoop VS Spark



Hadoop MapReduce	Aspect	Spark
MapReduce is difficult to program and needs abstractions.	Difficultly	Spark is easy to program and does not require any abstractions.
There is no in-built interactive mode except Pig and Hive.	Interactive mode	It has interactive mode.

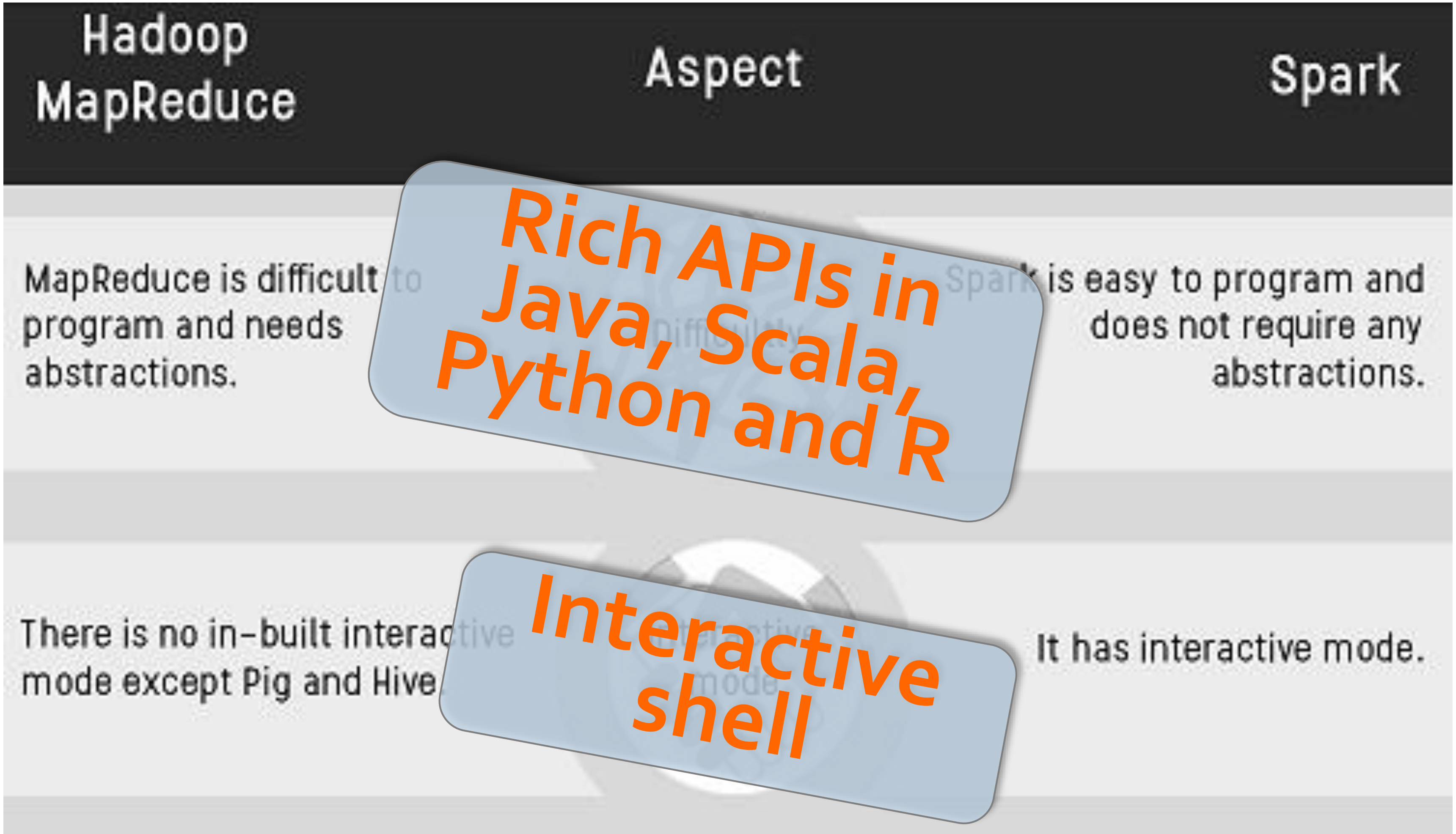
Hadoop VS Spark



Hadoop MapReduce	Aspect	Spark
MapReduce is difficult to program and needs abstractions.	Difficulty	Spark is easy to program and does not require any abstractions.
There is no in-built interactive mode except Pig and Hive.	Interactive mode	It has interactive mode.

Rich APIs in Java, Scala, Python and R

Hadoop VS Spark

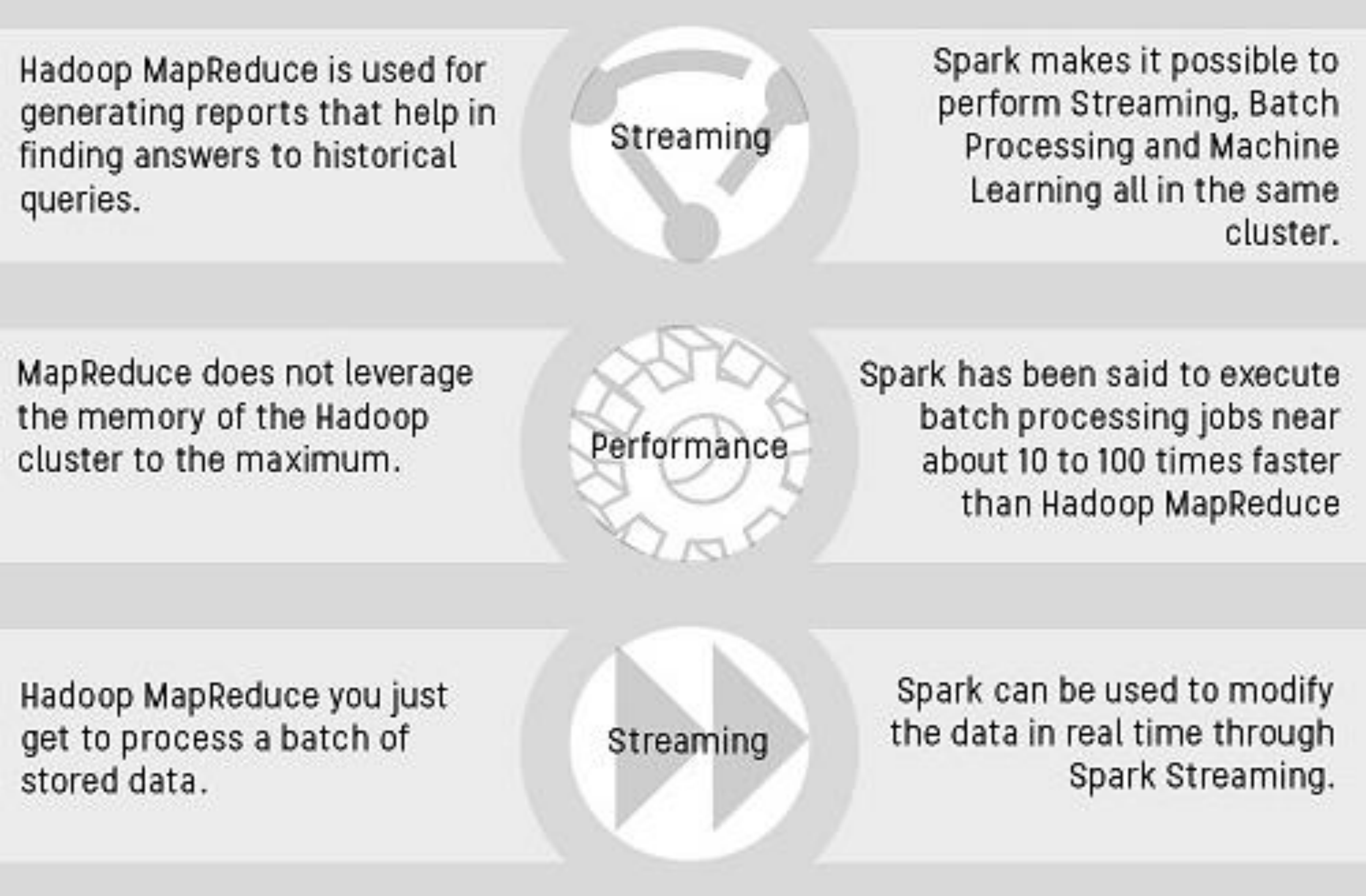


Hadoop MapReduce	Aspect	Spark
MapReduce is difficult to program and needs abstractions.	Difficulty	Spark is easy to program and does not require any abstractions.
There is no in-built interactive mode except Pig and Hive.	Interactive mode	It has interactive mode.

Rich APIs in Java, Scala, Python and R

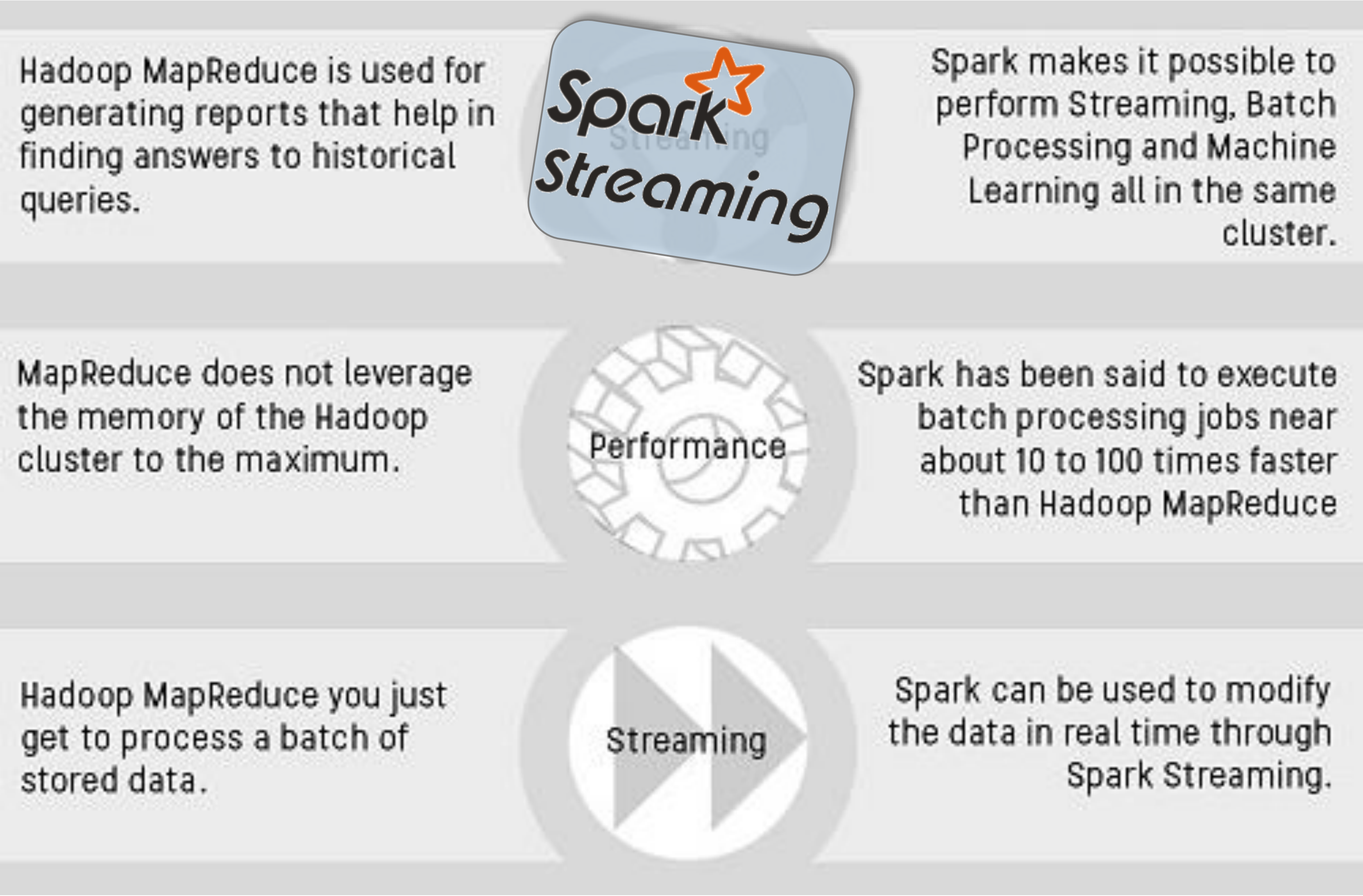
Interactive shell

Hadoop VS Spark



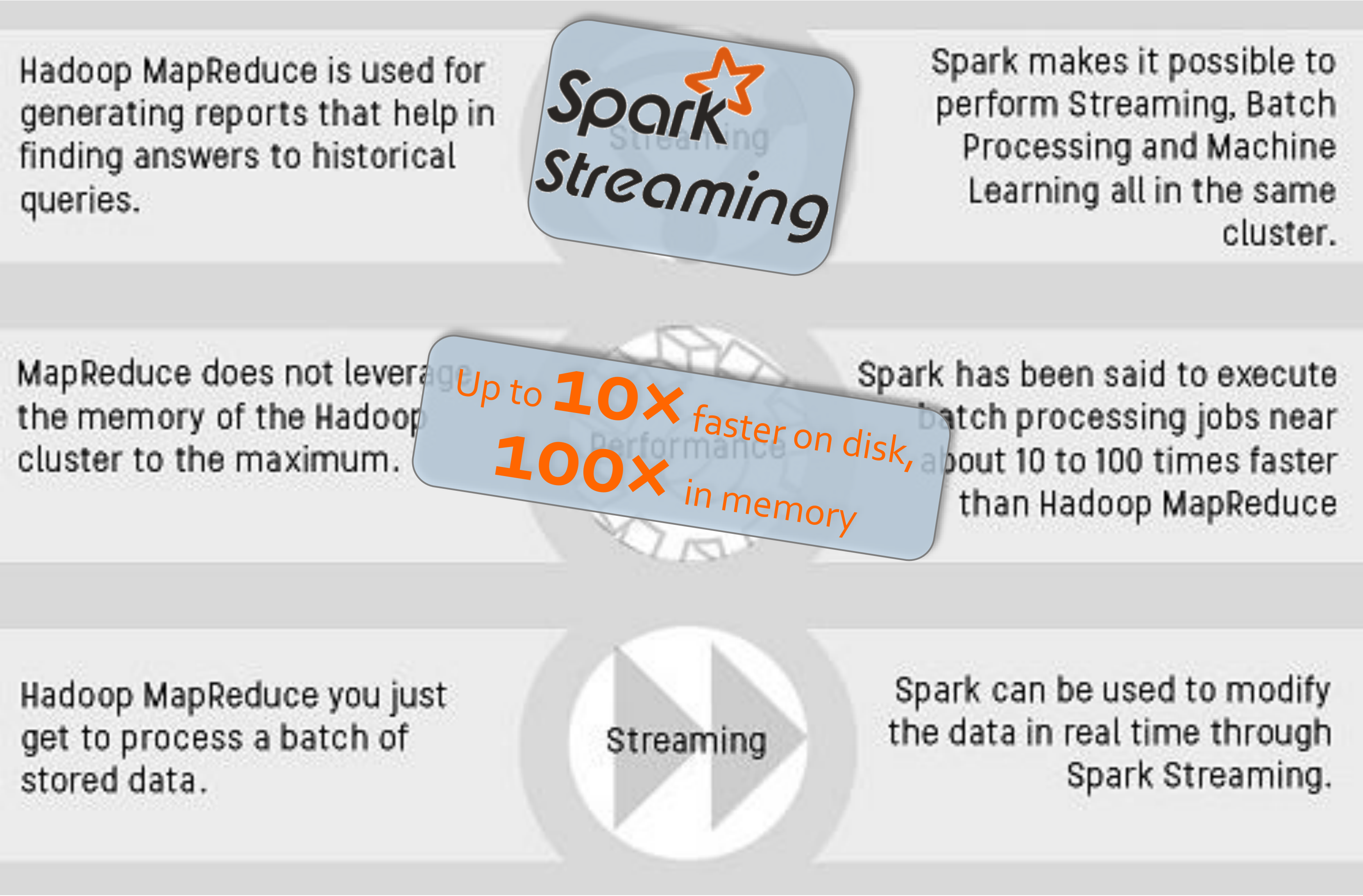
Hadoop MapReduce is used for generating reports that help in finding answers to historical queries.	Streaming	Spark makes it possible to perform Streaming, Batch Processing and Machine Learning all in the same cluster.
MapReduce does not leverage the memory of the Hadoop cluster to the maximum.	Performance	Spark has been said to execute batch processing jobs near about 10 to 100 times faster than Hadoop MapReduce
Hadoop MapReduce you just get to process a batch of stored data.	Streaming	Spark can be used to modify the data in real time through Spark Streaming.

Hadoop VS Spark



Hadoop MapReduce is used for generating reports that help in finding answers to historical queries.	Spark Streaming Streaming	Spark makes it possible to perform Streaming, Batch Processing and Machine Learning all in the same cluster.
MapReduce does not leverage the memory of the Hadoop cluster to the maximum.	Performance	Spark has been said to execute batch processing jobs near about 10 to 100 times faster than Hadoop MapReduce
Hadoop MapReduce you just get to process a batch of stored data.	Streaming	Spark can be used to modify the data in real time through Spark Streaming.

Hadoop VS Spark



Hadoop MapReduce is used for generating reports that help in finding answers to historical queries.

Spark makes it possible to perform Streaming, Batch Processing and Machine Learning all in the same cluster.

MapReduce does not leverage the memory of the Hadoop cluster to the maximum.

Spark has been said to execute batch processing jobs near about 10 to 100 times faster than Hadoop MapReduce

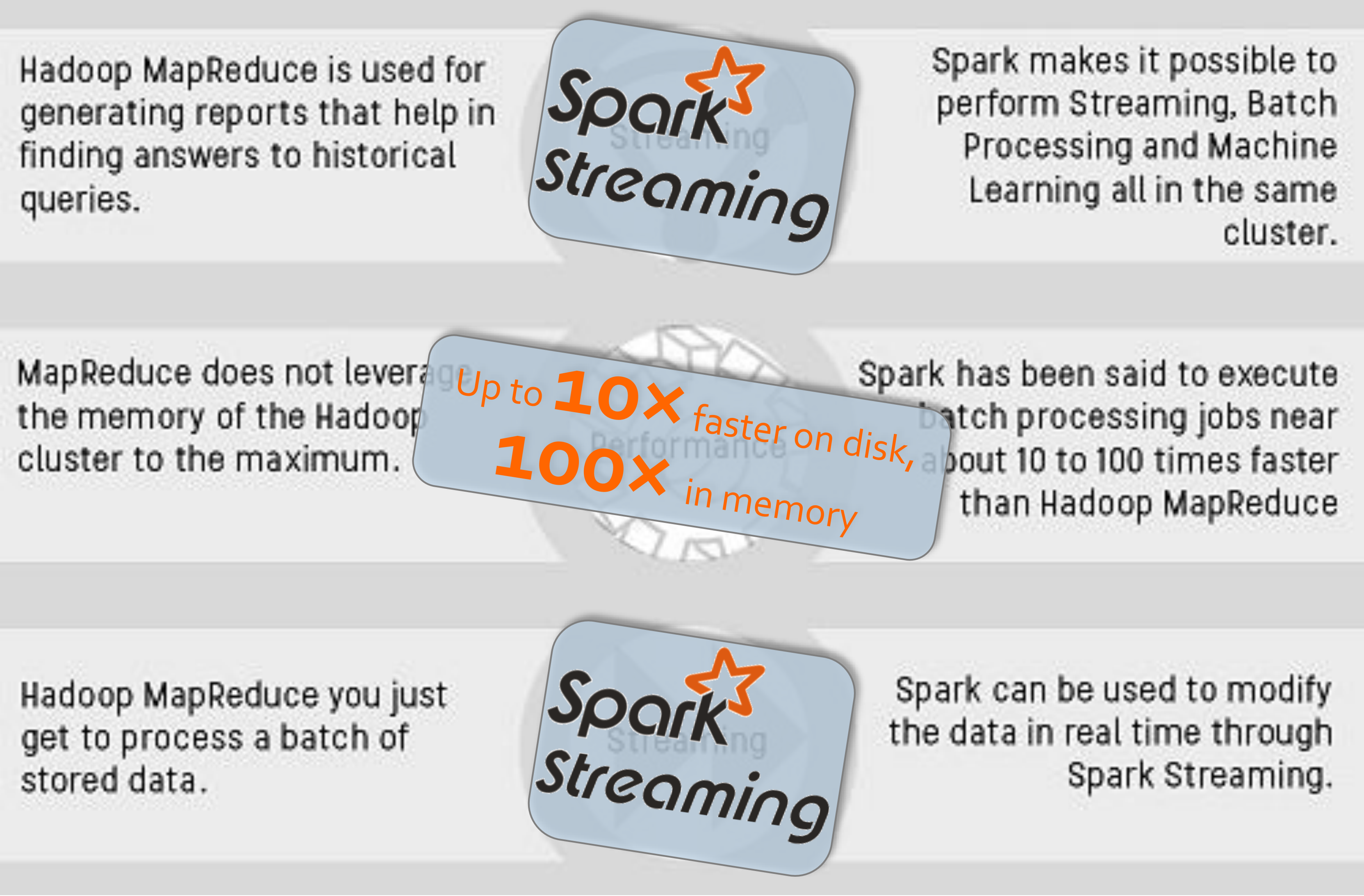
Hadoop MapReduce you just get to process a batch of stored data.

Spark can be used to modify the data in real time through Spark Streaming.

Spark Streaming

Up to 10x faster on disk, 100x in memory

Hadoop VS Spark



Hadoop MapReduce is used for generating reports that help in finding answers to historical queries.

Spark makes it possible to perform Streaming, Batch Processing and Machine Learning all in the same cluster.

MapReduce does not leverage the memory of the Hadoop cluster to the maximum.

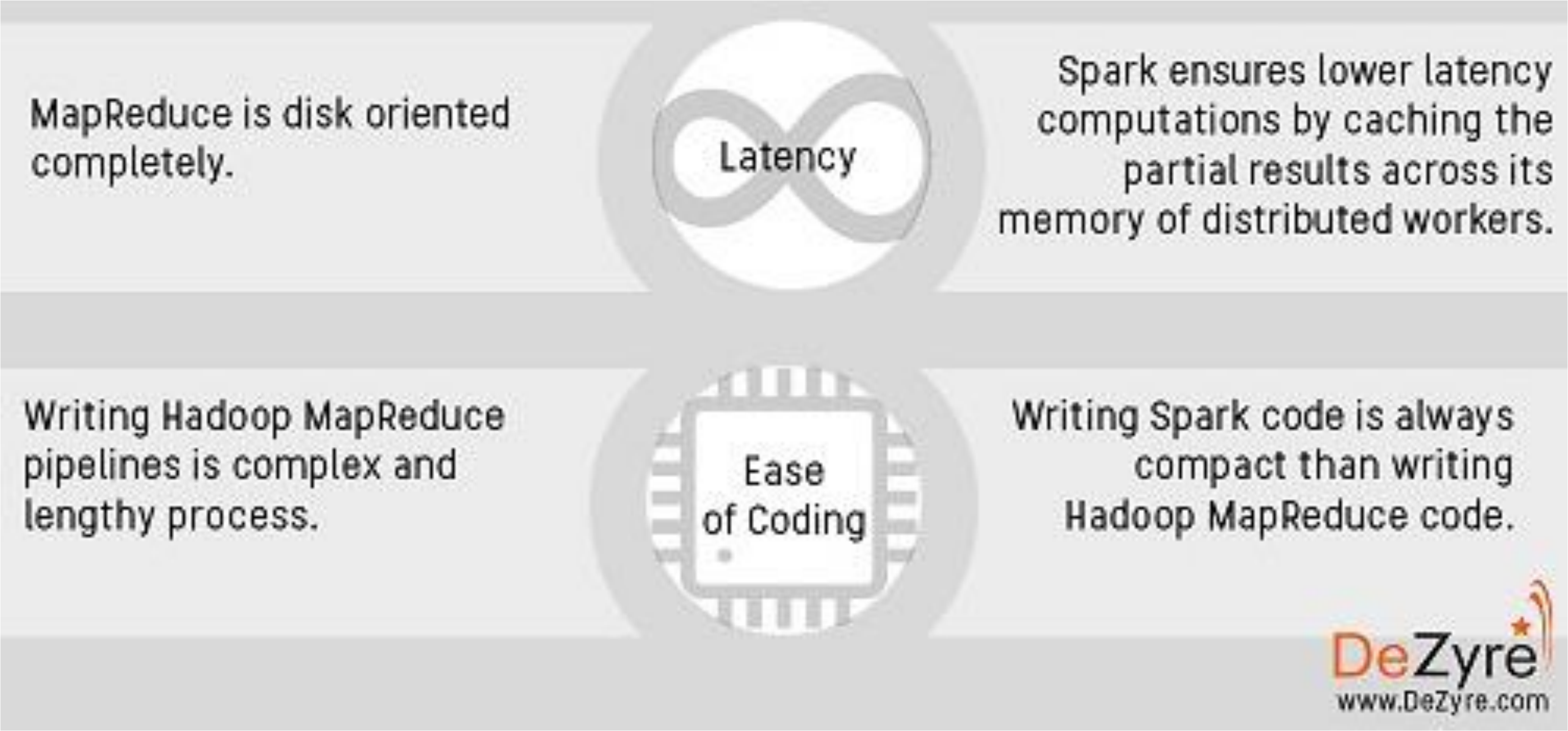
Up to **10x** faster on disk, **100x** in memory

Spark has been said to execute batch processing jobs near about 10 to 100 times faster than Hadoop MapReduce

Hadoop MapReduce you just get to process a batch of stored data.

Spark can be used to modify the data in real time through Spark Streaming.

Hadoop VS Spark



MapReduce is disk oriented completely.

Latency

Spark ensures lower latency computations by caching the partial results across its memory of distributed workers.

Writing Hadoop MapReduce pipelines is complex and lengthy process.

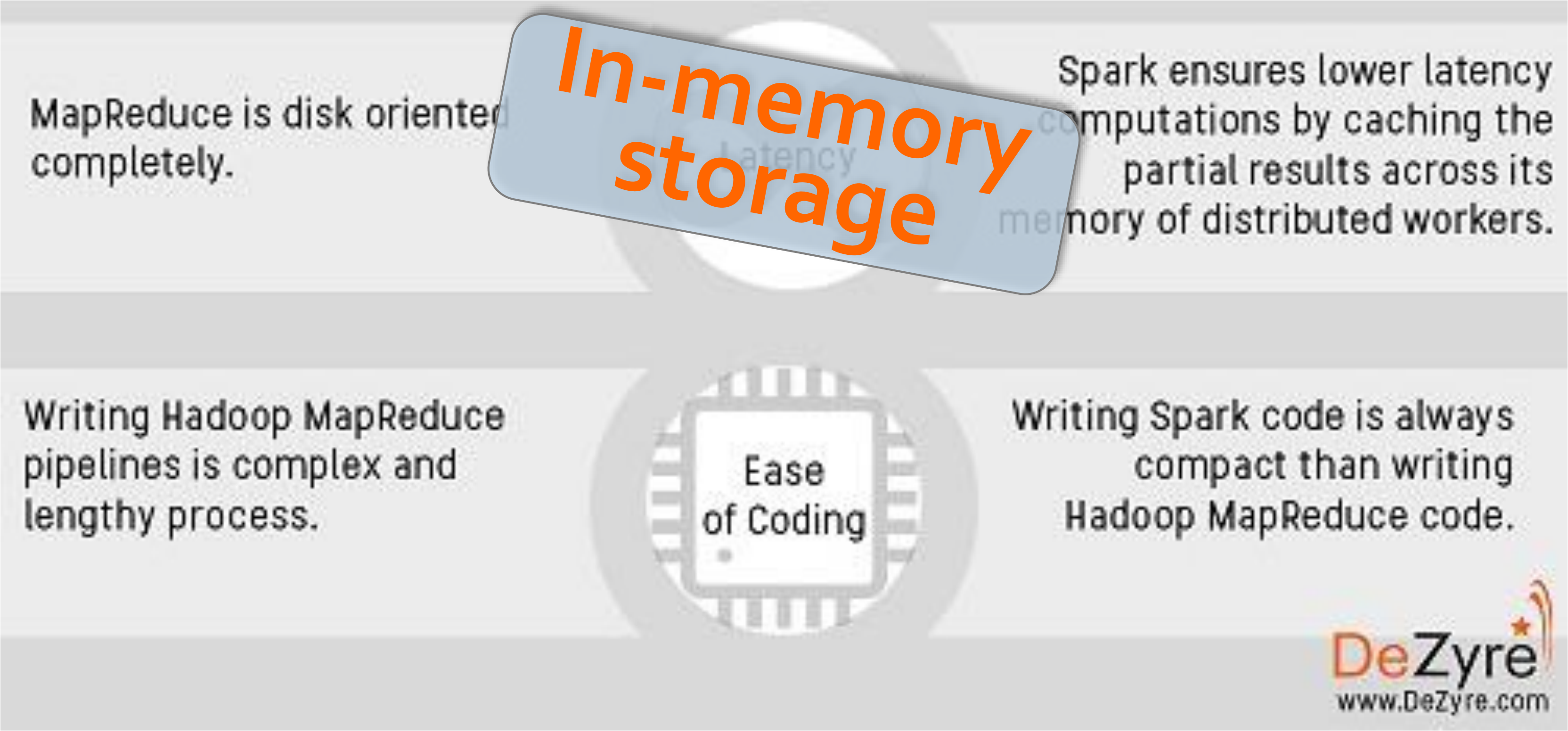
Ease of Coding

Writing Spark code is always compact than writing Hadoop MapReduce code.

DeZyre
www.DeZyre.com

https://www.dezyre.com/article/hadoop-mapreduce-vs-apache-spark-who-wins-the-battle/83

Hadoop VS Spark



In-memory storage

MapReduce is disk oriented completely.

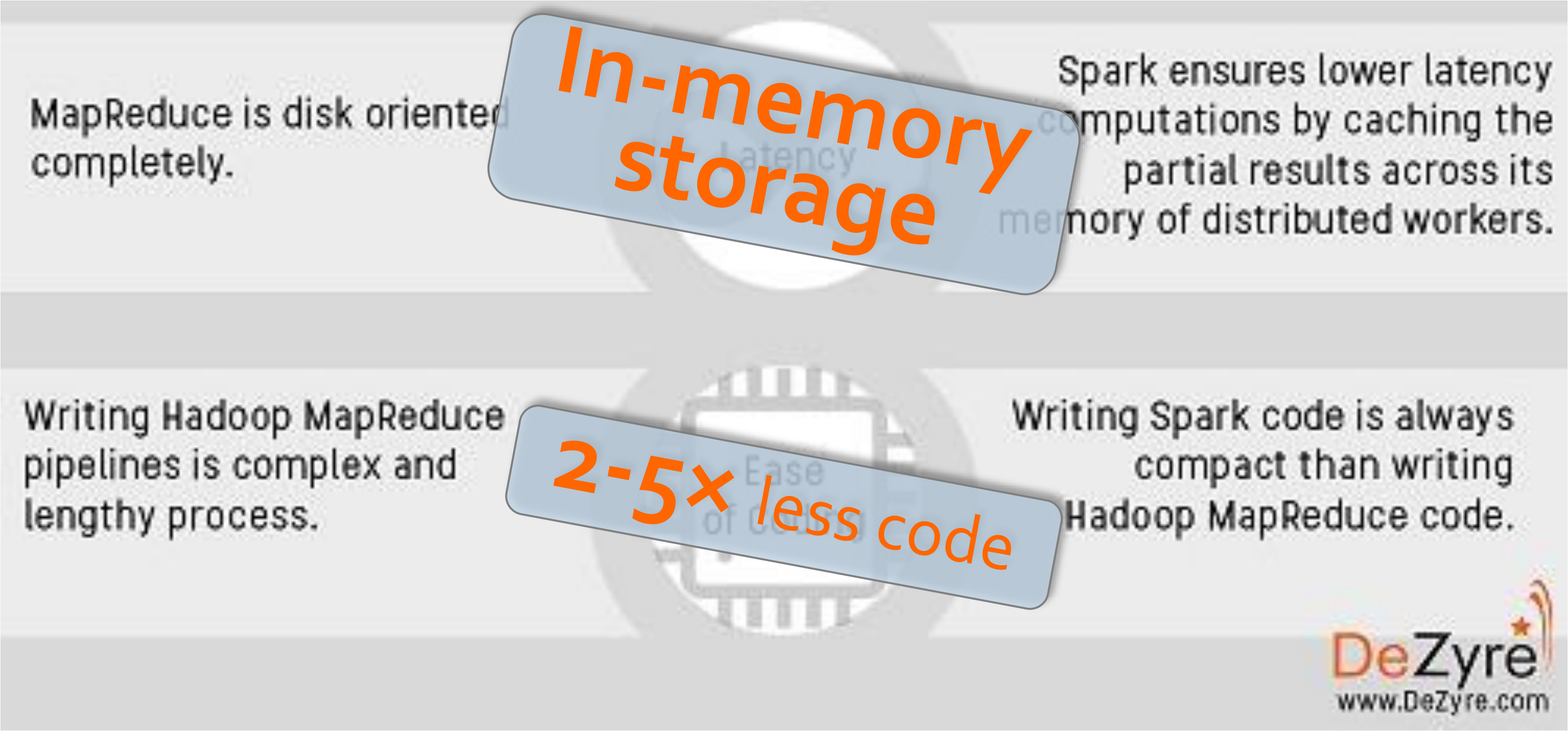
Spark ensures lower latency computations by caching the partial results across its memory of distributed workers.

Writing Hadoop MapReduce pipelines is complex and lengthy process.

Writing Spark code is always compact than writing Hadoop MapReduce code.

https://www.dezyre.com/article/hadoop-mapreduce-vs-apache-spark-who-wins-the-battle/83

Hadoop VS Spark



MapReduce is disk oriented completely.

Spark ensures lower latency computations by caching the partial results across its memory of distributed workers.

Writing Hadoop MapReduce pipelines is complex and lengthy process.

Writing Spark code is always compact than writing Hadoop MapReduce code.

Latency
Ease of Coding
DeZyre
www.DeZyre.com

In-memory storage

2-5× less code

https://www.dezyre.com/article/hadoop-mapreduce-vs-apache-spark-who-wins-the-battle/83

Outline

- Apache Hadoop VS Apache Spark
- **Weaknesses of Hadoop**
- Apache Spark
- MLlib: Machine Learning on Spark
- ML: API on top of DataFrame
- SparkR & PySpark

Weaknesses of Hadoop

- Use of HDD disc
- Java programming
  - There is no interactive shell
- You can not iterate over the data
- However, it is widely used for its great advantages

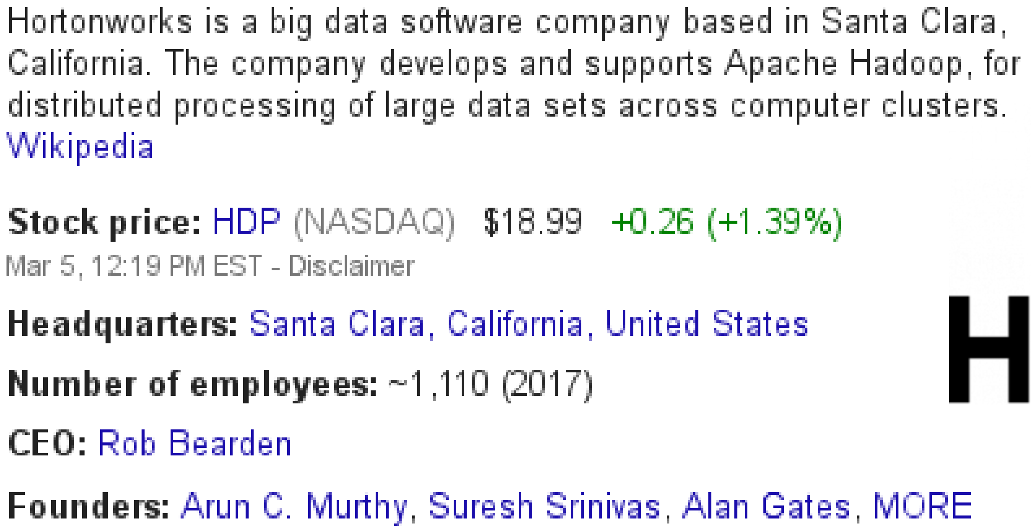
Weaknesses of Hadoop

Open Source Community

□ 1000+ meetup members

□ 70+ contributors from 20 companies

□ In use at Intel, Yahoo!, Adobe, etc.



Hortonworks is a big data software company based in Santa Clara, California. The company develops and supports Apache Hadoop, for the distributed processing of large data sets across computer clusters. Wikipedia

Stock price: HDP (NASDAQ) $18.99 +0.26 (+1.39%)
Mar 5, 12:19 PM EST - Disclaimer

Headquarters: Santa Clara, California, United States

Number of employees: ~1,110 (2017)

CEO: Rob Bearden

Founders: Arun C. Murthy, Suresh Srinivas, Alan Gates, MORE

Weaknesses of Hadoop

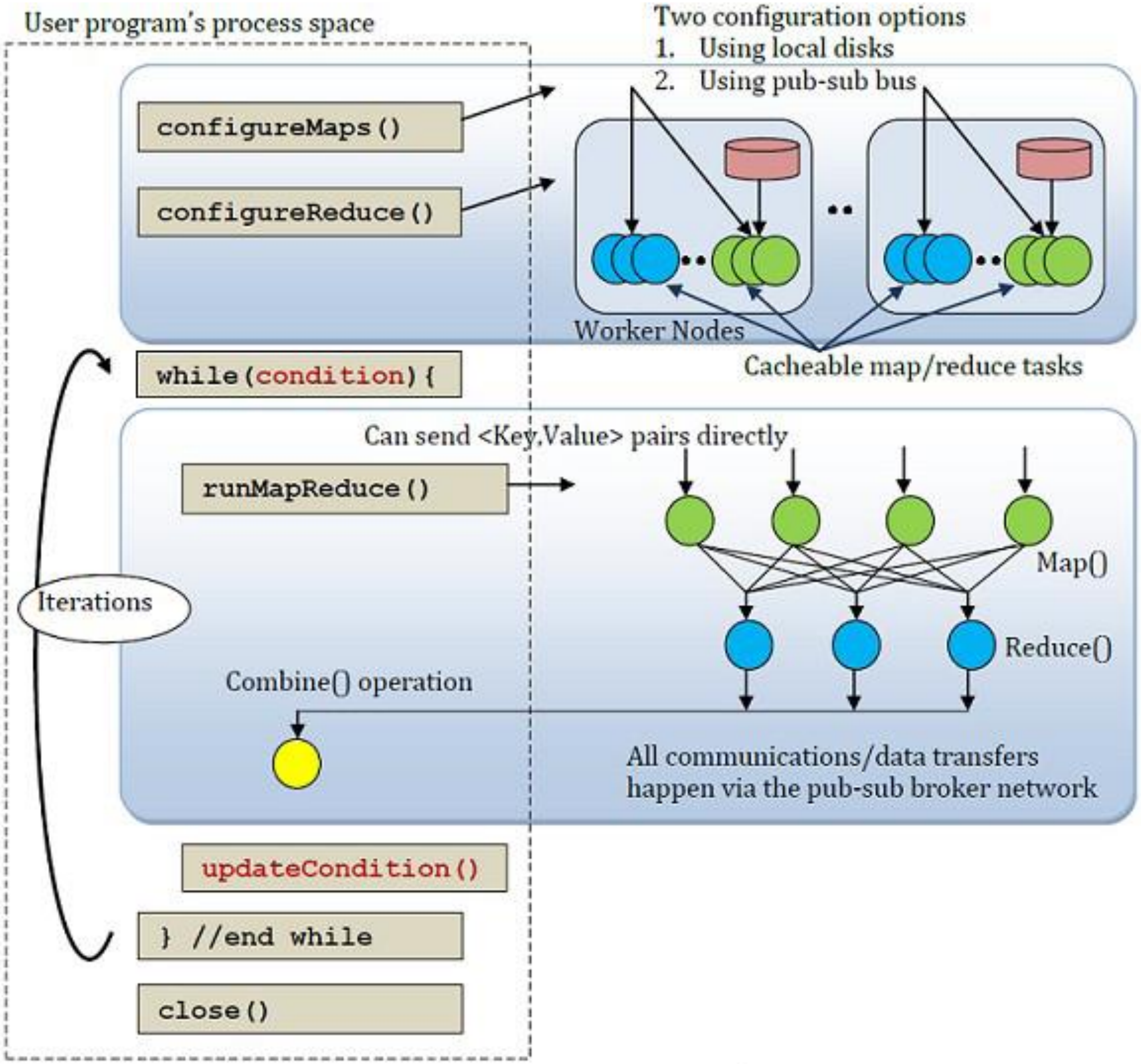
- A wide variety of Solutions:

Twister

Iterative MapReduce

HaLoop

Spark



```
configureMaps()
configureReduce()
while(condition){
runMapReduce()
updateCondition()
} //end while
close()
```

User program's process space
Two configuration options
1. Using local disks
2. Using pub-sub bus
Worker Nodes
Cacheable map/reduce tasks
Iterations
Can send <Key,Value> pairs directly
Map()
Reduce()
Combine() operation
All communications/data transfers happen via the pub-sub broker network

Outline

□ Apache Hadoop VS Apache Spark

□ Weaknesses of Hadoop

□ **Apache Spark**

□ MLlib: Machine Learning on Spark

□ ML: API on top of DataFrame

□ SparkR & PySpark

Apache Spark

Retain the attractive properties of MapReduce

□ Fault tolerance

□ Data locality

□ Scalability

Solution: augment data flow model with "resilient distributed datasets" (RDDs)

Apache Spark

What is a RDD?

- □ A RDD is an immutable, partitioned, logical collection of records
- □ Built using transformations over another RDDs
- □ Can be cached for future reuse
- □ Partitioning can be based on a key in each record

Apache Spark

Transformations (define a new RDD)

map flatMap mapPartition

filter repartition sample

union intersection distinct

aggregateByKey reduceByKey

http://spark.apache.org/docs/latest/programming-guide.html#transformations

Actions (return a result to driver program)

count first take collect

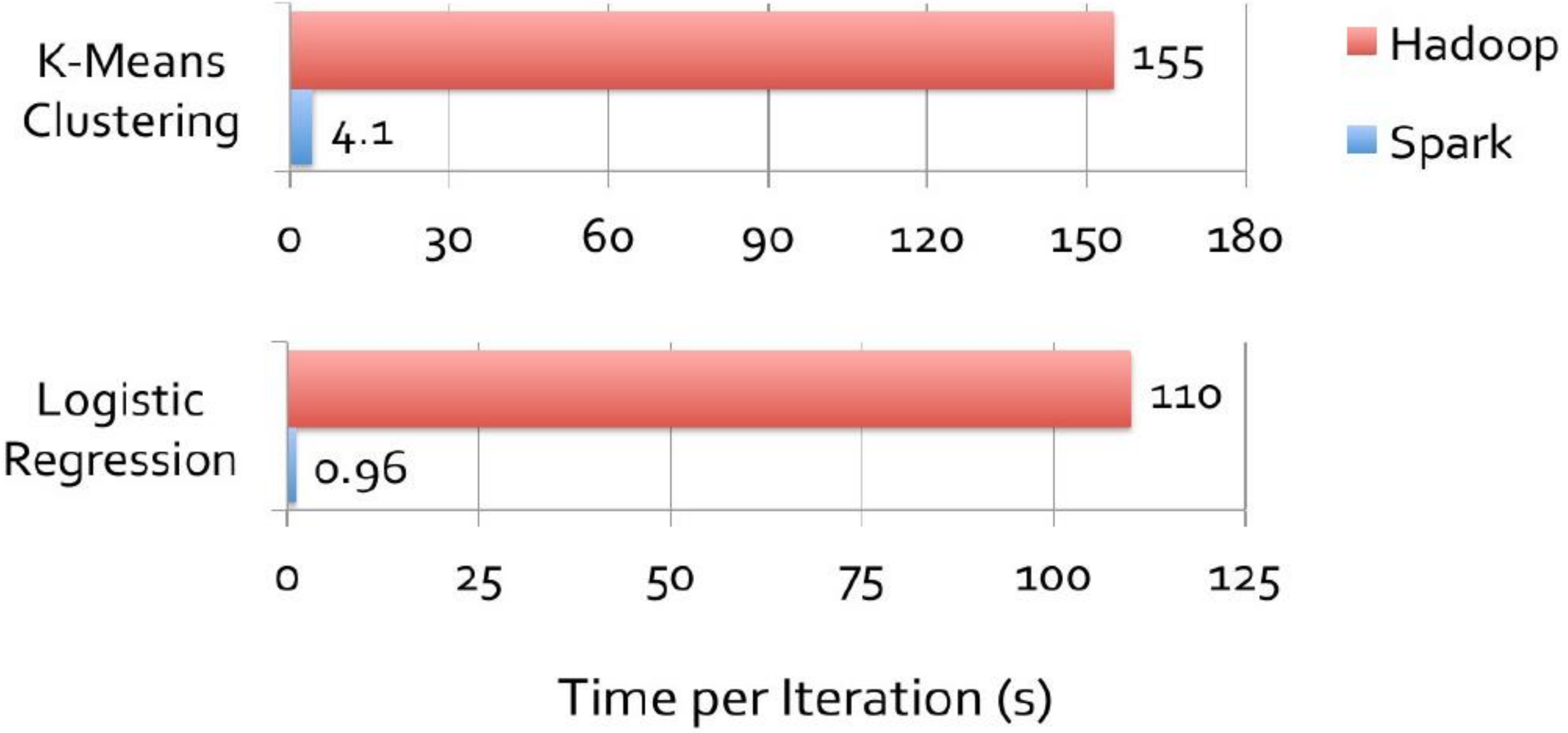
reduce takeSample takeOrdered

saveAsTextFile saveAsSequenceFile

http://spark.apache.org/docs/latest/programming-guide.html#actions

Apache Spark

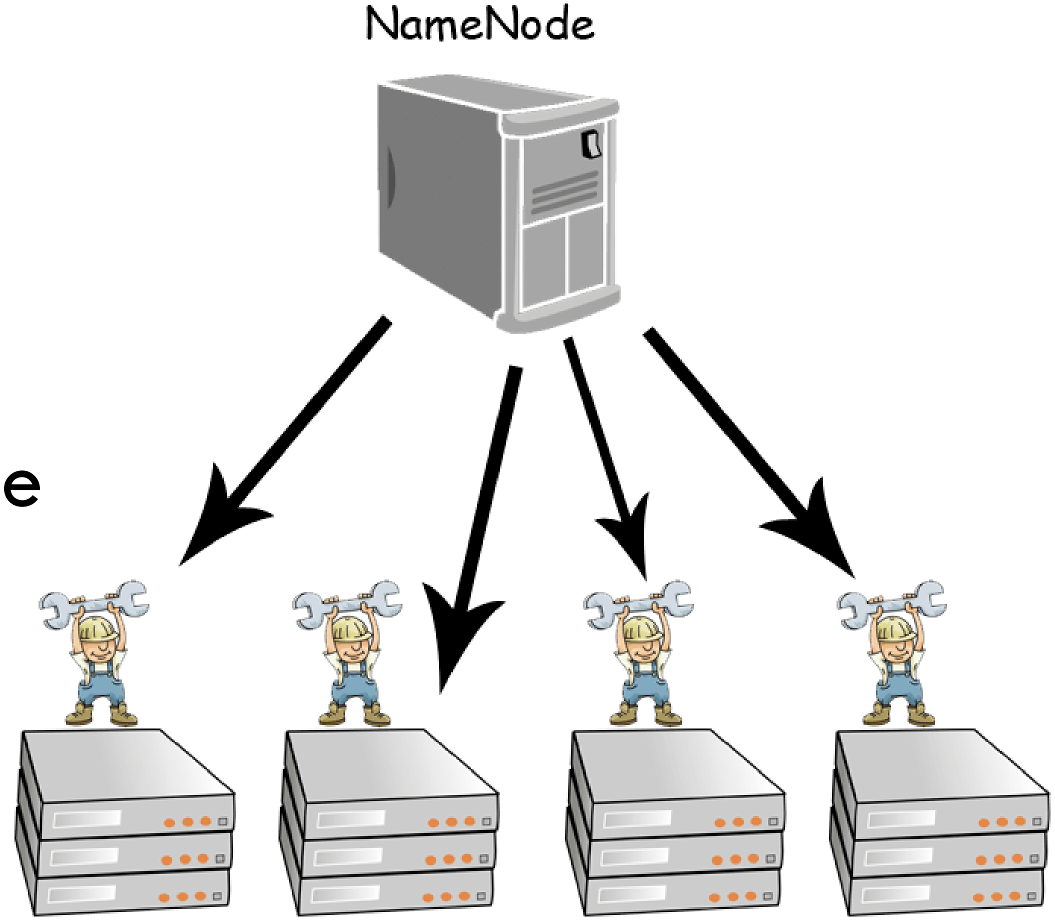
Iterative Algorithms



Apache Spark

HDFS

- □ Hadoop Distributed File System
- □ Commodity Hardware
- □ HDD disk



HDFS

/home/antoniolopez/

/user/antoniolopez/

HDFS storage is different from user's local storage

Outline

- □Apache Hadoop VS Apache Spark
- □Weaknesses of Hadoop
- □Apache Spark
- □**MLlib: Machine Learning on Spark**
- □ML: API on top of DataFrame
- □SparkR & PySpark

Mllib: Machine Learning on

Mllib is a scalable machine learning library

- □ Easy to Deploy

- □ Take advantage of hadoop environment
- □ Contains many algorithms and utilities.

https://spark.apache.org/docs/latest/mllib-guide.html

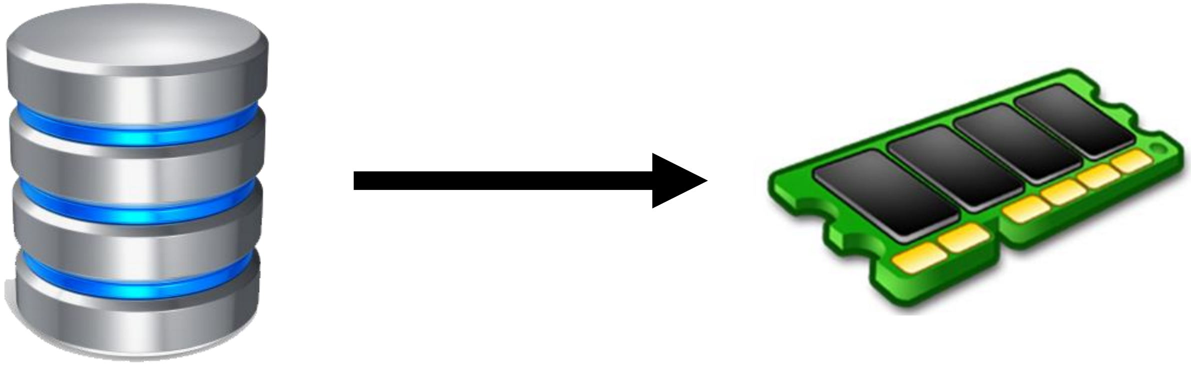
Mllib: Machine Learning on

Algorithms and utilities

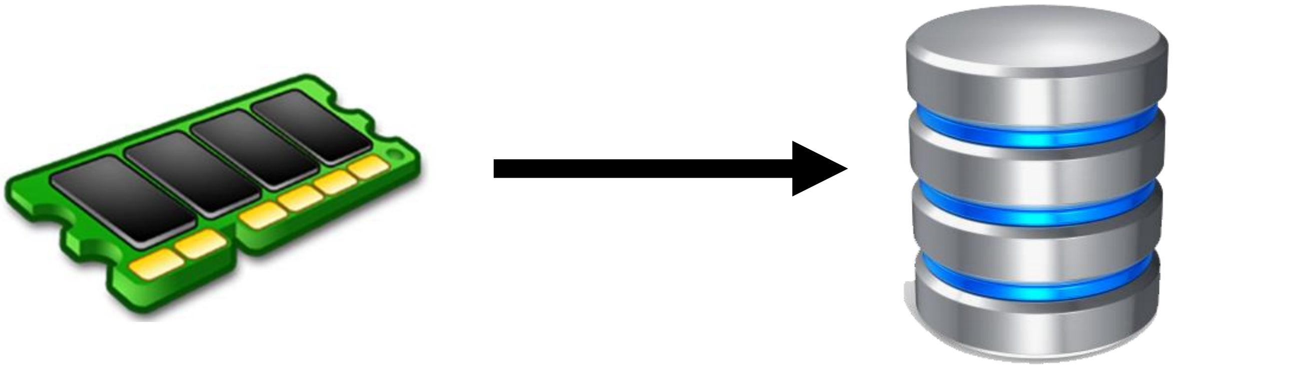
- Classification
  - Naive Bayes, Decision Tree classifier, Random Forest
- Regression
  - Linear regression, Decision Tree regression
- Clustering
  - K-means
- Statistics
  - Summary Statistics, Pearson’s test for independence

Mllib: Read & Write

□ Reading from HDFS to main memory



□ Writing intermediate or final results to HDFS



Mllib: Cache

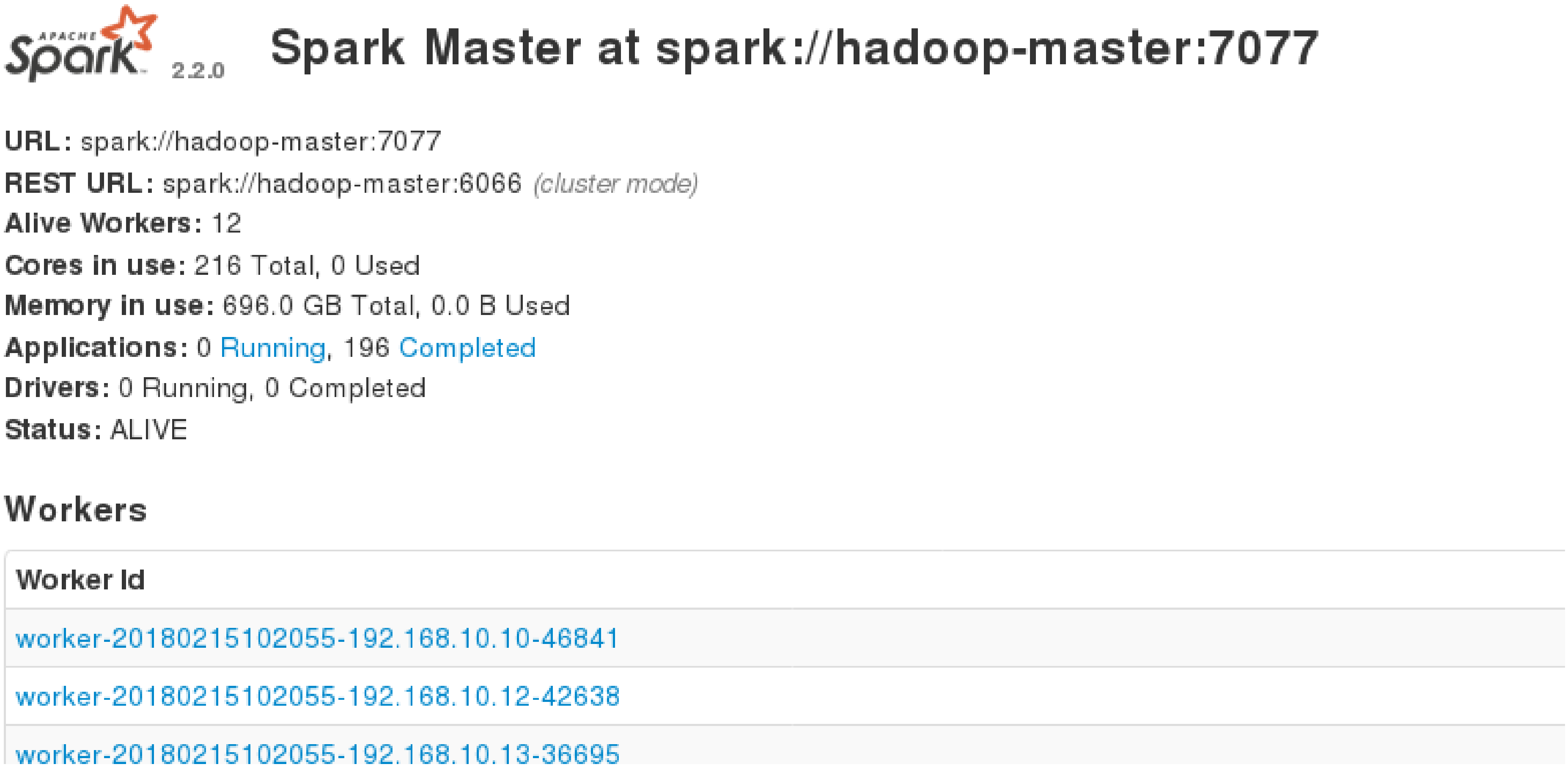
Algorithms and utilities

- □ "Cache" operation forces Spark to distribute the data
- □ Allocate the data in main memory
- □ Import for reuse of data
- □ Iterative algorithm

} Efficiently

Mllib: Web UI

Web UI (http://hadoop.ugr.es:8079/cluster)



Spark Master at spark://hadoop-master:7077

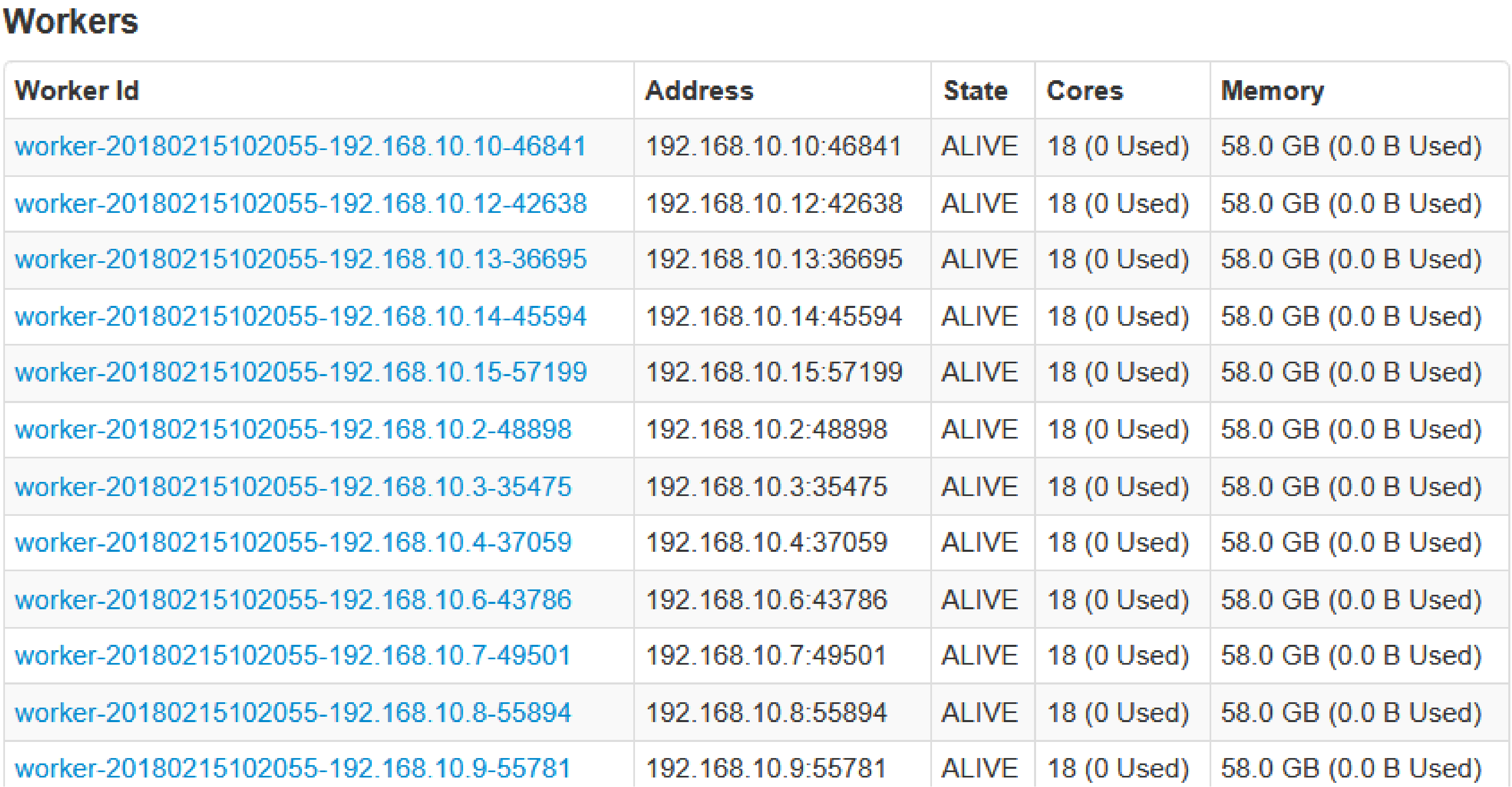
URL: spark://hadoop-master:7077
REST URL: spark://hadoop-master:6066 *(cluster mode)*
Alive Workers: 12
Cores in use: 216 Total, 0 Used
Memory in use: 696.0 GB Total, 0.0 B Used
Applications: 0 Running, 196 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id
worker-20180215102055-192.168.10.10-46841
worker-20180215102055-192.168.10.12-42638
worker-20180215102055-192.168.10.13-36695

Mllib: Web UI

12 Workers

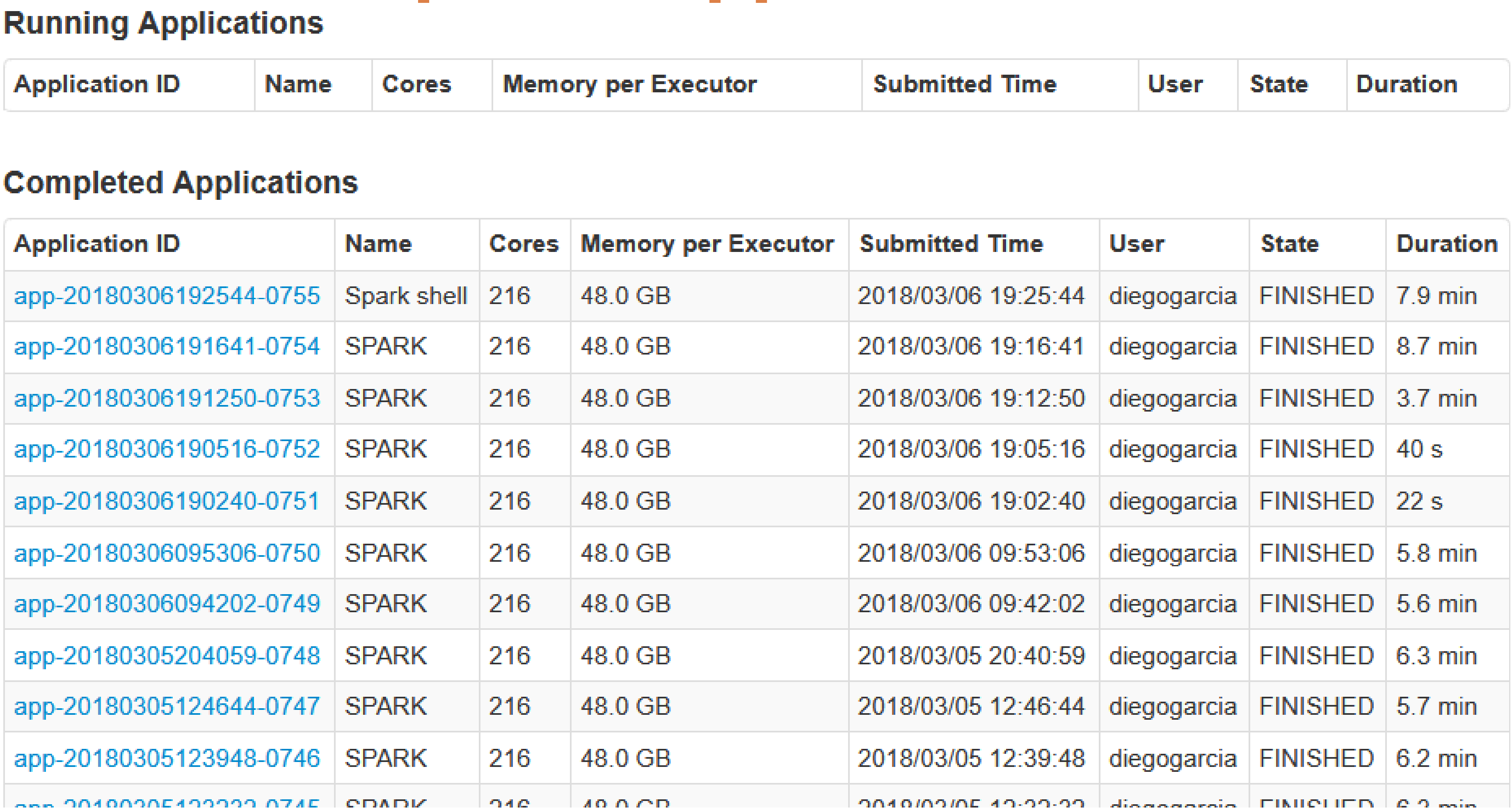


Workers

Worker Id	Address	State	Cores	Memory
worker-20180215102055-192.168.10.10-46841	192.168.10.10:46841	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.12-42638	192.168.10.12:42638	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.13-36695	192.168.10.13:36695	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.14-45594	192.168.10.14:45594	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.15-57199	192.168.10.15:57199	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.2-48898	192.168.10.2:48898	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.3-35475	192.168.10.3:35475	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.4-37059	192.168.10.4:37059	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.6-43786	192.168.10.6:43786	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.7-49501	192.168.10.7:49501	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.8-55894	192.168.10.8:55894	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)
worker-20180215102055-192.168.10.9-55781	192.168.10.9:55781	ALIVE	18 (0 Used)	58.0 GB (0.0 B Used)

Mllib: Web UI

Run & completed applications



Running Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

Completed Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20180306192544-0755	Spark shell	216	48.0 GB	2018/03/06 19:25:44	diegogarcia	FINISHED	7.9 min
app-20180306191641-0754	SPARK	216	48.0 GB	2018/03/06 19:16:41	diegogarcia	FINISHED	8.7 min
app-20180306191250-0753	SPARK	216	48.0 GB	2018/03/06 19:12:50	diegogarcia	FINISHED	3.7 min
app-20180306190516-0752	SPARK	216	48.0 GB	2018/03/06 19:05:16	diegogarcia	FINISHED	40 s
app-20180306190240-0751	SPARK	216	48.0 GB	2018/03/06 19:02:40	diegogarcia	FINISHED	22 s
app-20180306095306-0750	SPARK	216	48.0 GB	2018/03/06 09:53:06	diegogarcia	FINISHED	5.8 min
app-20180306094202-0749	SPARK	216	48.0 GB	2018/03/06 09:42:02	diegogarcia	FINISHED	5.6 min
app-20180305204059-0748	SPARK	216	48.0 GB	2018/03/05 20:40:59	diegogarcia	FINISHED	6.3 min
app-20180305124644-0747	SPARK	216	48.0 GB	2018/03/05 12:46:44	diegogarcia	FINISHED	5.7 min
app-20180305123948-0746	SPARK	216	48.0 GB	2018/03/05 12:39:48	diegogarcia	FINISHED	6.2 min
app-2018030512[illegible]-0745	SPARK	216	48.0 GB	2018/03/05 12:[illegible]	diegogarcia	FINISHED	[illegible] min

Outline

- Apache Hadoop VS Apache Spark
- Weaknesses of Hadoop
- Apache Spark
- MLlib: Machine Learning on Spark
- **ML: API on top of DataFrame**
- SparkR & PySpark

ML: API on top of DataFrame

“Spark **ML** standardizes APIs for machine learning algorithms to make it easier to combine multiple algorithms into a single pipeline, or workflow.”

scikits learn
machine learning in Python

https://spark.apache.org/docs/latest/ml-guide.html

ML: API on top of DataFrame

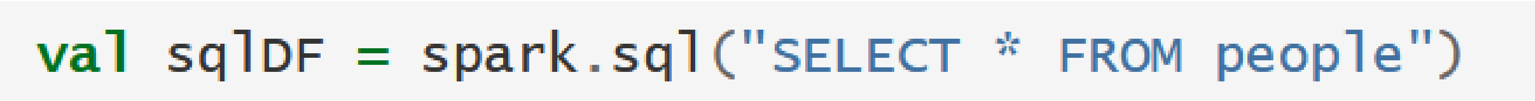
"A **DataFrame** is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood"

http://spark.apache.org/docs/latest/sql-programming-guide.html#dataframes

ML: API on top of DataFrame

SQL Queries

- SQL queries using string
- Return the result as a new DataFrame



```
val sqlDF = spark.sql("SELECT * FROM people")
```

http://spark.apache.org/docs/latest/sql-programming-guide.html#running-sql-queries-programmatically

ML: API on top of DataFrame

Interoperating with RDDs

- Transforming a RDD into a DataFrame
- Build like a RDD with column names

http://spark.apache.org/docs/latest/sql-programming-guide.html#inferring-the-schema-using-reflection

Outline

- □Apache Hadoop VS Apache Spark
- □Weaknesses of Hadoop
- □Apache Spark
- □MLlib: Machine Learning on Spark
- □ML: API on top of DataFrame
- □**SparkR & PySpark**

SparkR & PySpark

https://spark.apache.org/docs/latest/api/python/index.html

https://spark.apache.org/docs/latest/sparkr.html

FIRST STEPS ON APACHE SPARK

BIG DATA II

Jesús Maillo (jesusmh@decsai.ugr.es)

08/03/2018