UNIVERSITAT DE BARCELONA

Master in Fundamental Principles of Data Science

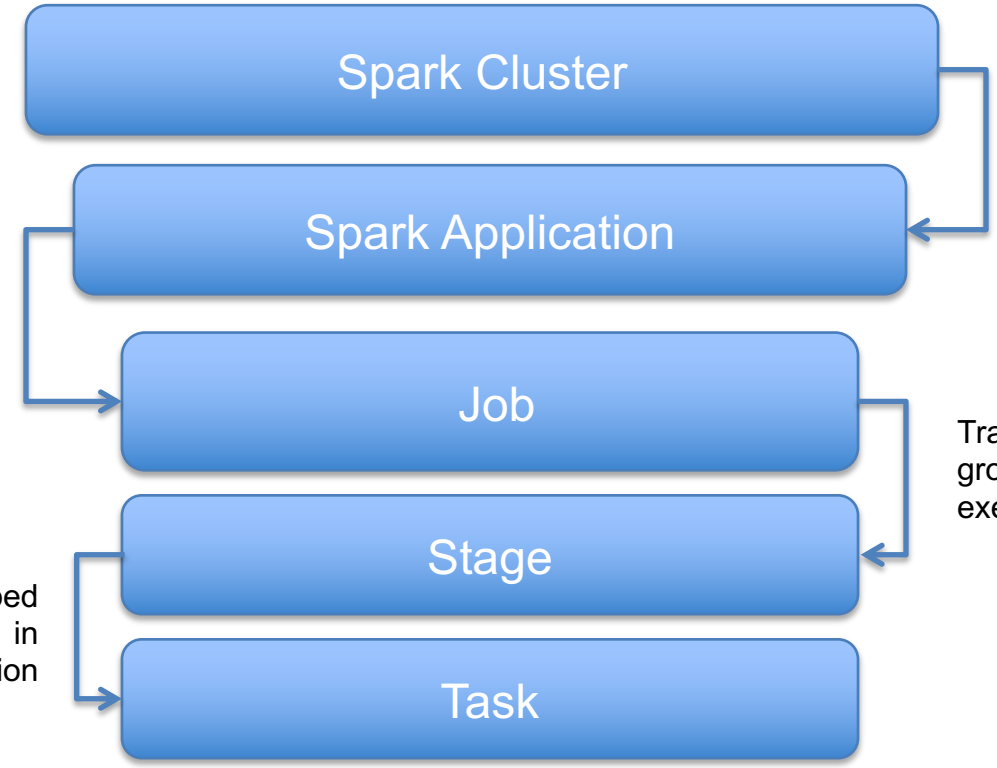
Dr Rohit Kumar

UNIVERSITAT DE BARCELONA

Spark Advanced

UNIVERSITAT DE BARCELONA

Spark Execution Life cycle



Each Spark submit creates an application Spark shell or a notebook is also an application.

Every Action creates a Job.

Transformations are grouped in stages for execution.

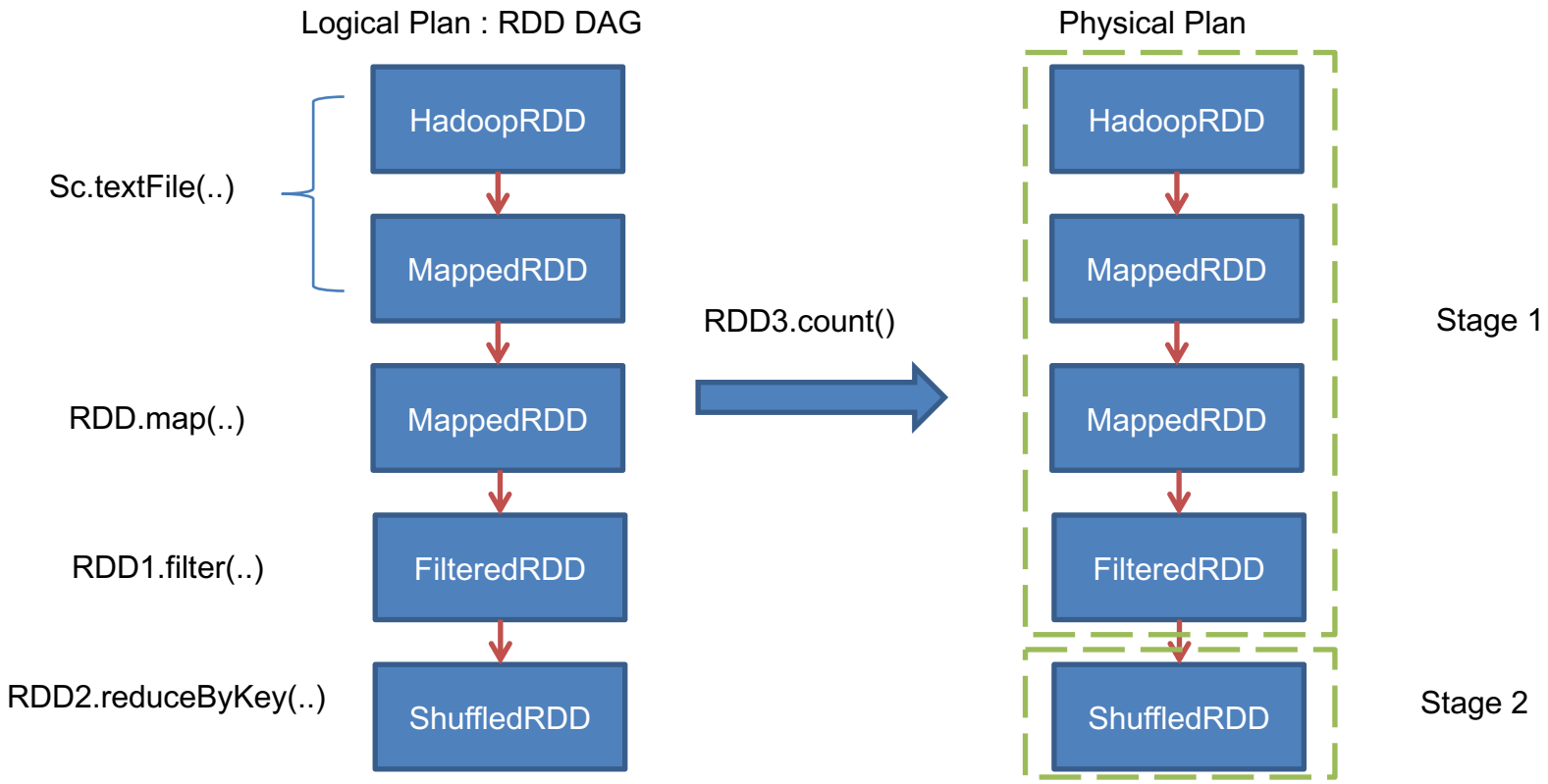
The executions grouped for a stage runs in parallel for every partition called as task.

Monitoring and Debugging Spark

Spark transformations creates a DAG which is a “logical plan” when Spark action is called the DAG is used to create a “physical plan” which consist of grouping transformations together in stages.

UNIVERSITAT DE BARCELONA

Logical Plan to Physical plan



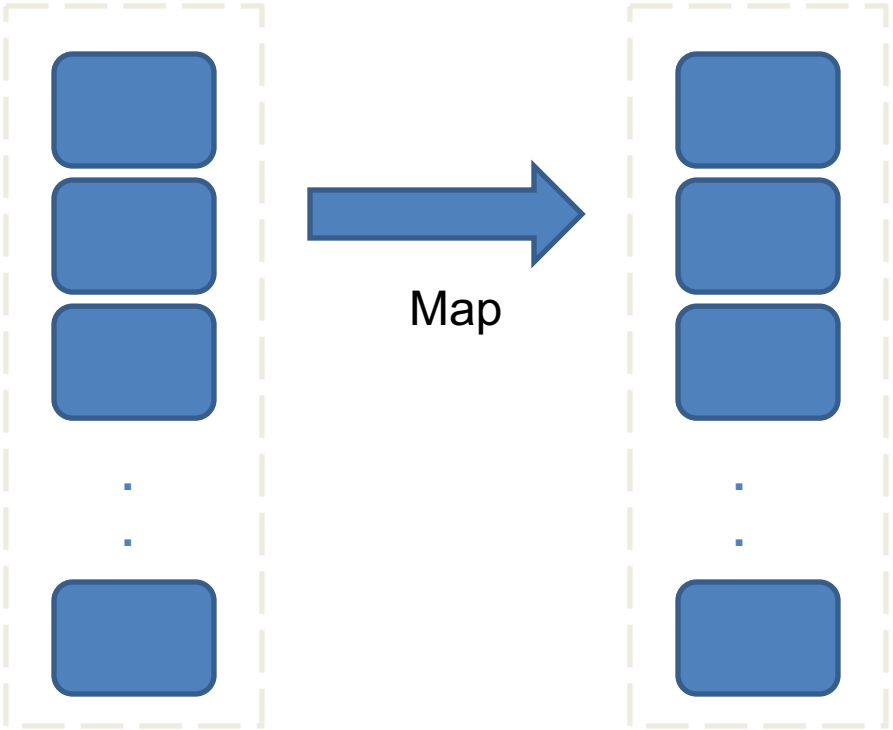
UNIVERSITAT DE BARCELONA

Pipelining

- Spark tries to run as many transformations in one stage as possible when pipelining is possible.
- New stage is created when data needs to move across different stages like shuffle.

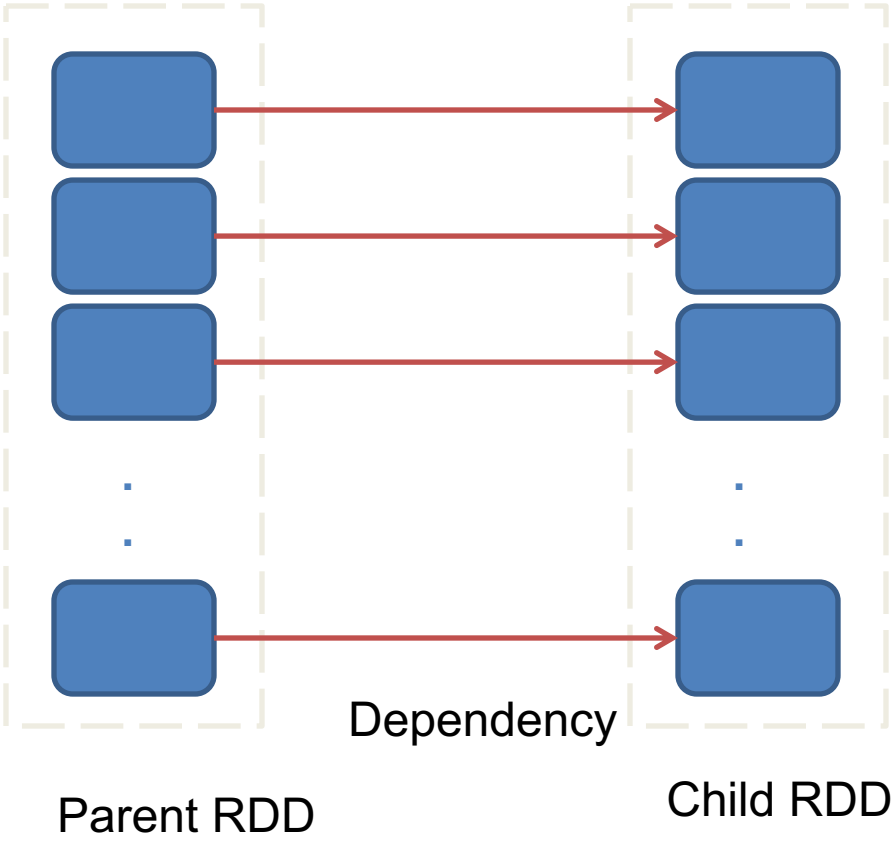
UNIVERSITAT DE BARCELONA

RDD dependency



UNIVERSITAT DE BARCELONA

RDD dependency



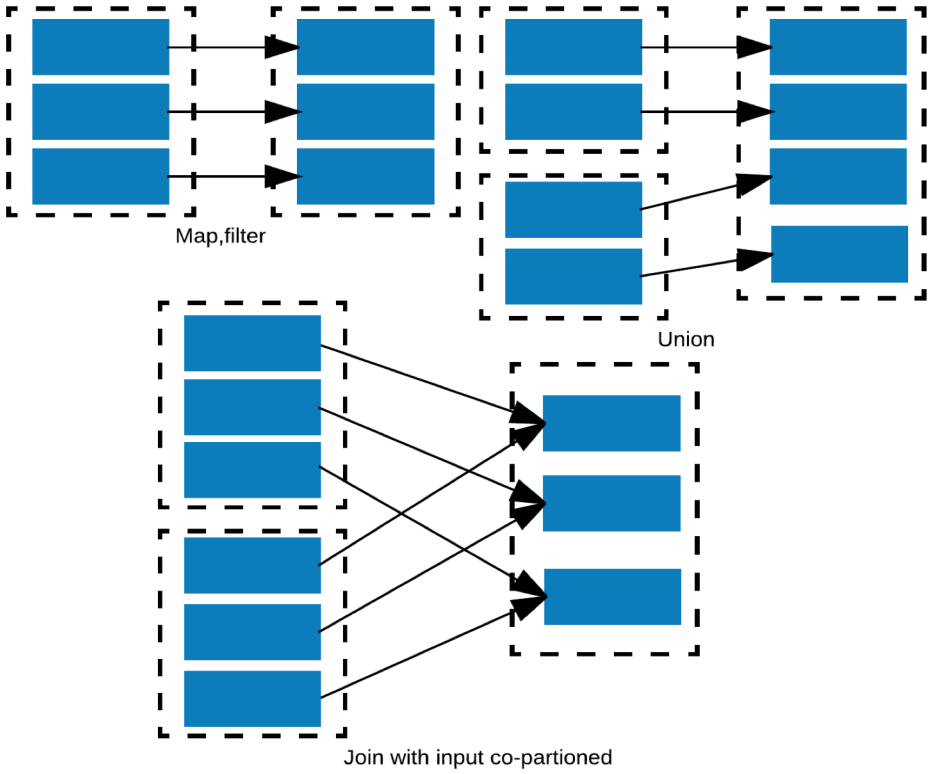
UNIVERSITAT DE BARCELONA

RDD dependency

- Narrow dependency: Each partition of the parent RDD is used by at most one partition of the child RDD. Fast!!
- Wide dependency: Each partition of the parent RDD may be used by multiple child partitions. Slow!!

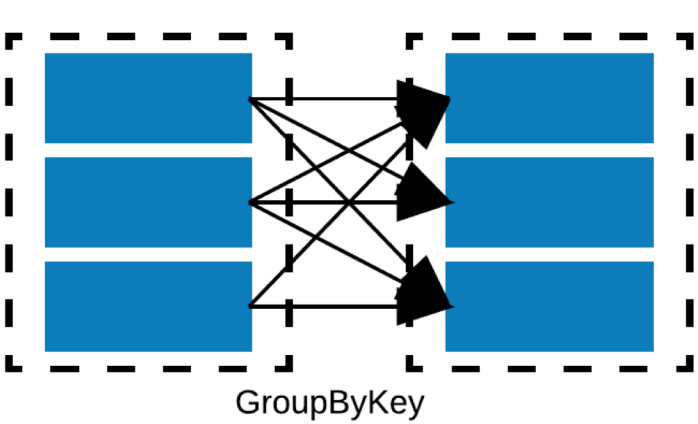
UNIVERSITAT DE BARCELONA

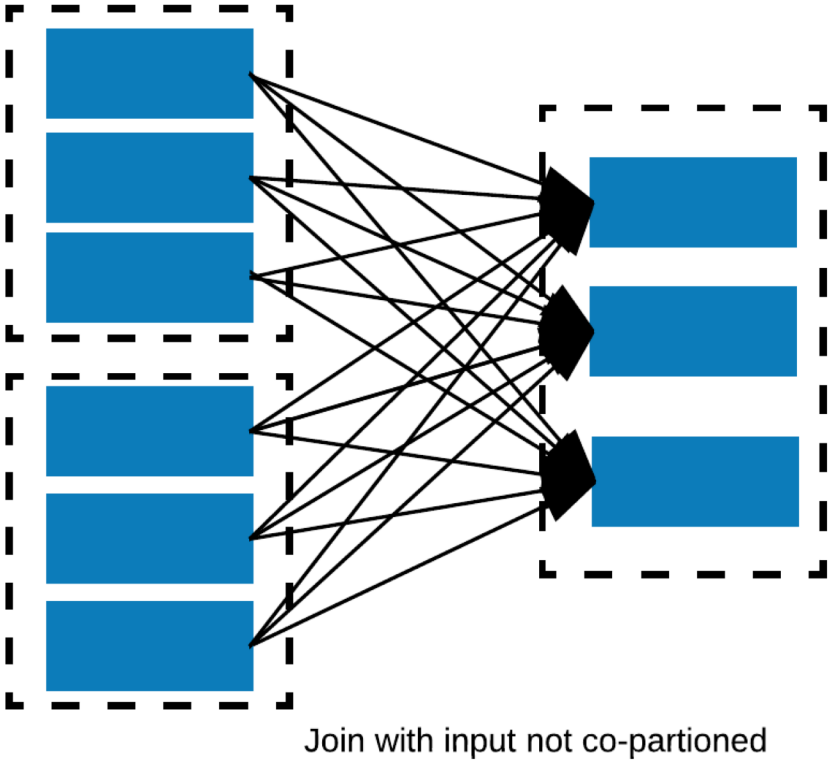
Narrow Dependency



UNIVERSITAT DE BARCELONA

Wide Dependency





UNIVERSITAT DE BARCELONA

Debugging Spark

- Spark web UI : Real time monitoring
- Driver and Executors logs : After the fact.

UNIVERSITAT DE BARCELONA

Recap

The important components of execution:

- **Task**: a unit of execution that runs on a single machine
- **Stage**: a group of tasks, based on partitions of the input data, which will perform the same computation in parallel
- **Job**: has one or more stages
- **Pipelining**: collapsing of RDDs into a single stage, when RDD transformations can be computed without data movement
- **DAG**: Logical graph of RDD operations
- **RDD**: Parallel dataset with partitions

UNIVERSITAT DE BARCELONA

Spark web UI

Every SparkContext (spark application) launches a web UI, by default on port 4040 of the driver node, that displays useful information about the application. This includes:

- A list of scheduler stages and tasks
- A summary of RDD sizes and memory usage
- Environmental information.
- Information about the running executors

If multiple host is running in same host they will get assigned to 4041,4042..

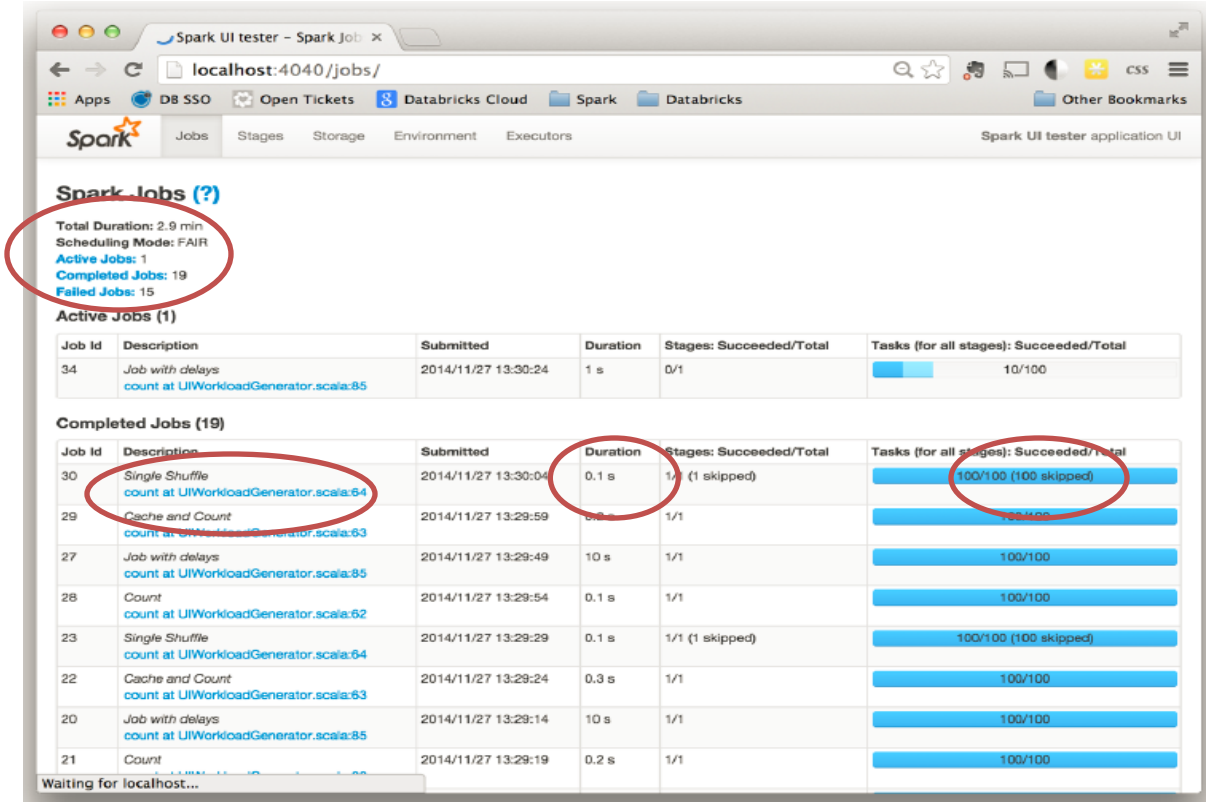
UNIVERSITAT DE BARCELONA

Spark web UI

- Jobs tab: Contains information about the job details of all the jobs.
- Stage tab: Contains information about all the stages details.
- Storage tab: Contains information about cached RDDs.
- Environment tab: Contains information about the spark cluster configuration details.
- Executors tab: Contains information about all the live and dead executors.

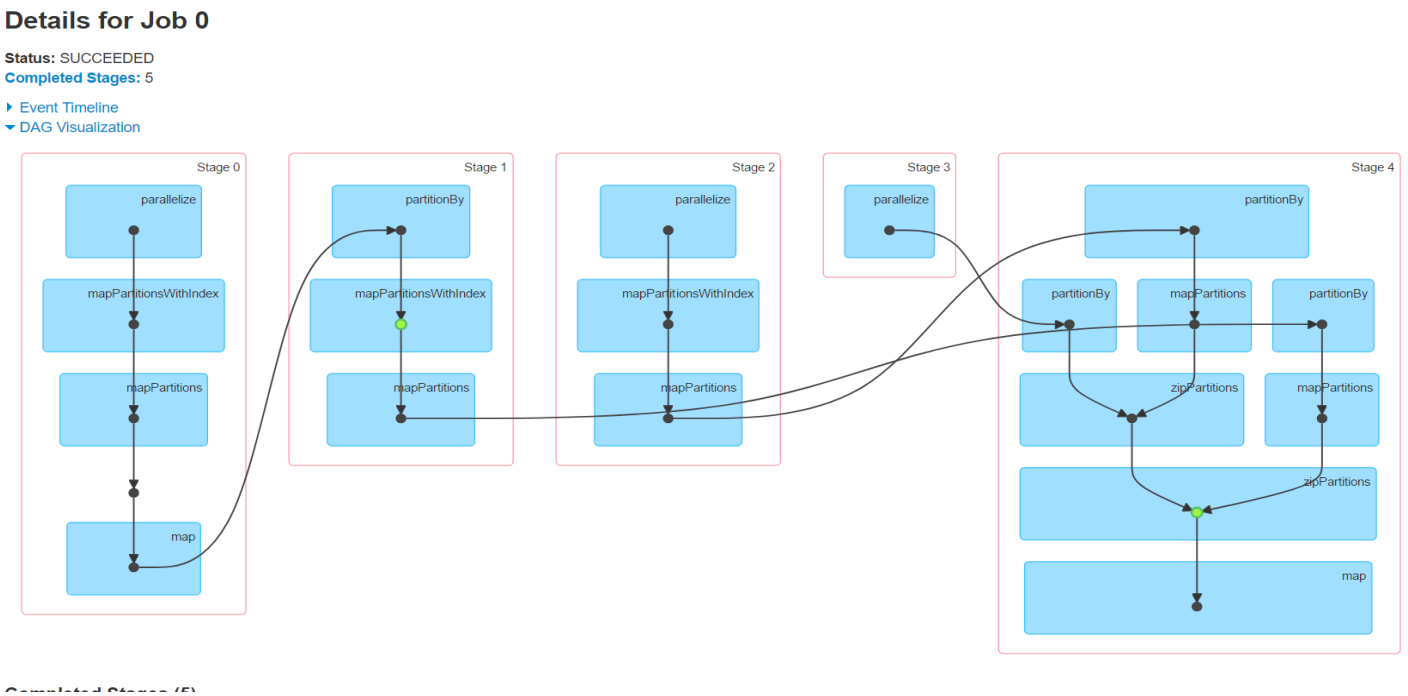
UNIVERSITAT DE BARCELONA

Spark web UI: Jobs



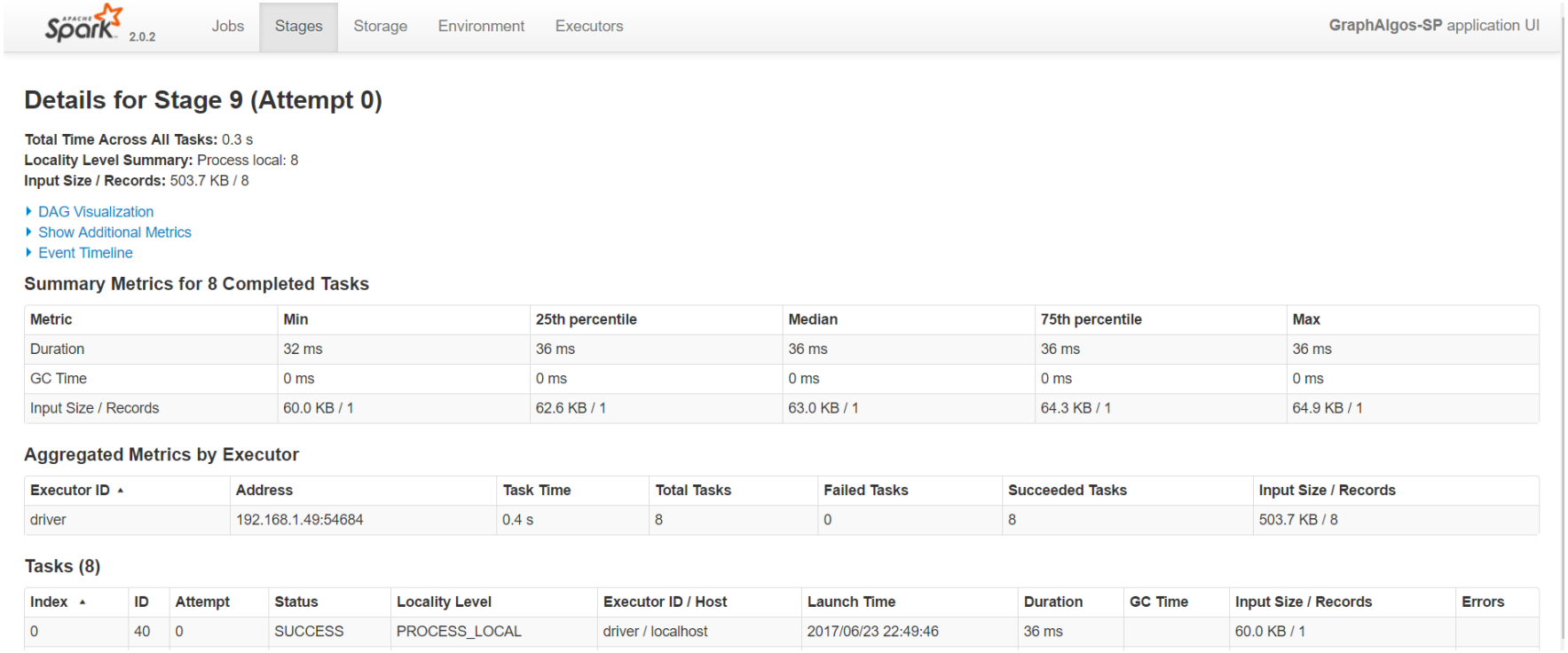
UNIVERSITAT DE BARCELONA

Spark web UI: Jobs



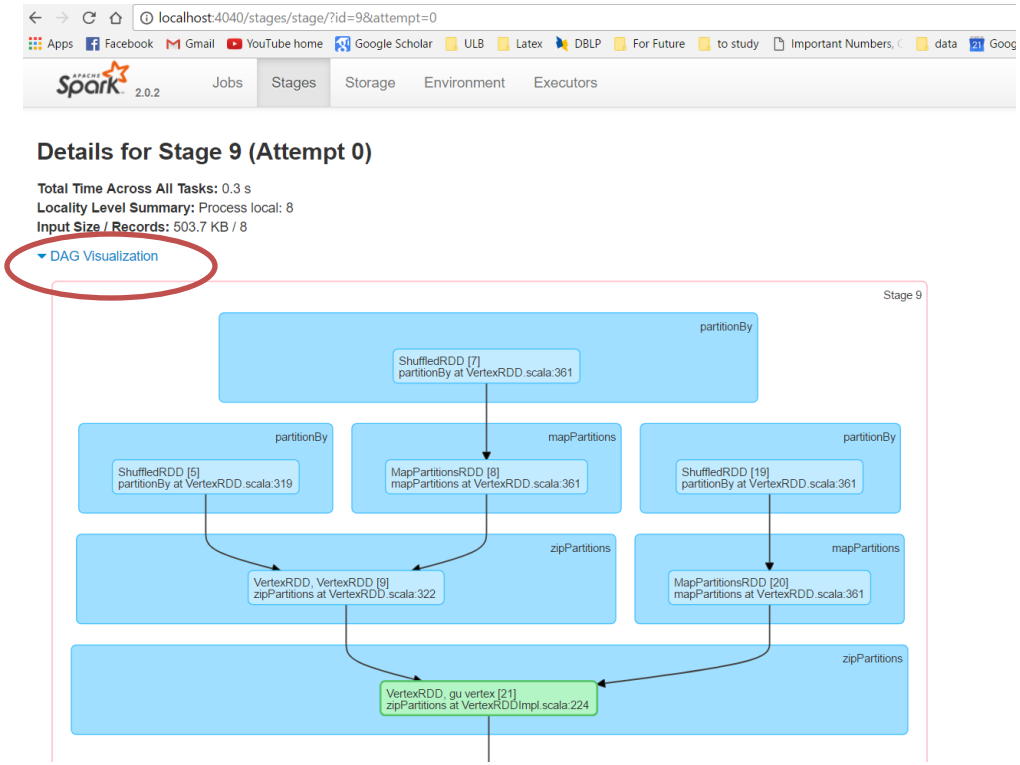
UNIVERSITAT DE BARCELONA

Spark web UI: Stages



UNIVERSITAT DE BARCELONA

Spark web UI: Stages



UNIVERSITAT DE BARCELONA

Spark web UI: Storage

Storage

RDDs

RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size on Disk
EdgeRDD, gu edges	Memory Deserialized 1x Replicated	8	100%	2.9 MB	0.0 B
EdgeRDD	Memory Deserialized 1x Replicated	8	100%	2.9 MB	0.0 B
VertexRDD	Memory Deserialized 1x Replicated	8	100%	331.4 KB	0.0 B
VertexRDD, gu vertex	Memory Deserialized 1x Replicated	8	100%	503.7 KB	0.0 B
EdgeRDD	Memory Deserialized 1x Replicated	8	100%	2.9 MB	0.0 B

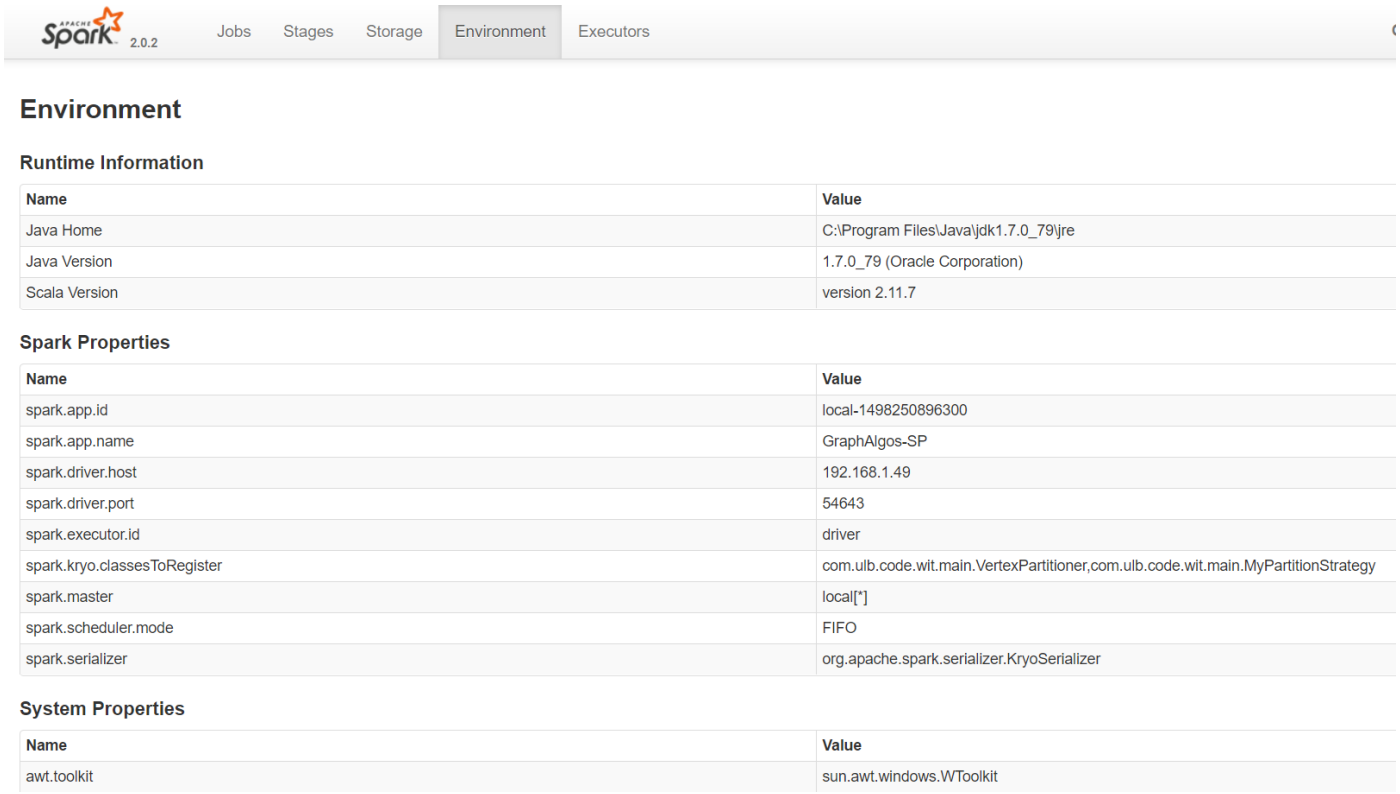
The Fraction cached can show more than 100% in case of more than one replica or when an RDD is
Recovered from failure.

Size on Disk will be shown when persist uses disk mode.

You can assign name to your RDDs and it will appear here.

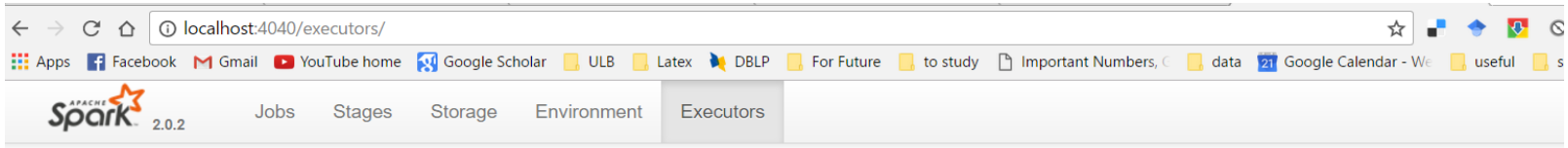
UNIVERSITAT DE BARCELONA

Spark web UI: Environment



UNIVERSITAT DE BARCELONA

Spark web UI: Executor



Executors

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write
Active(1)	0	0.0 B / 3.3 GB	0.0 B	8	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B
Total(1)	0	0.0 B / 3.3 GB	0.0 B	8	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B

Executors

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump
driver	192.168.1.49:54684	Active	0	0.0 B / 3.3 GB	0.0 B	8	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	Thread Dump

UNIVERSITAT DE BARCELONA

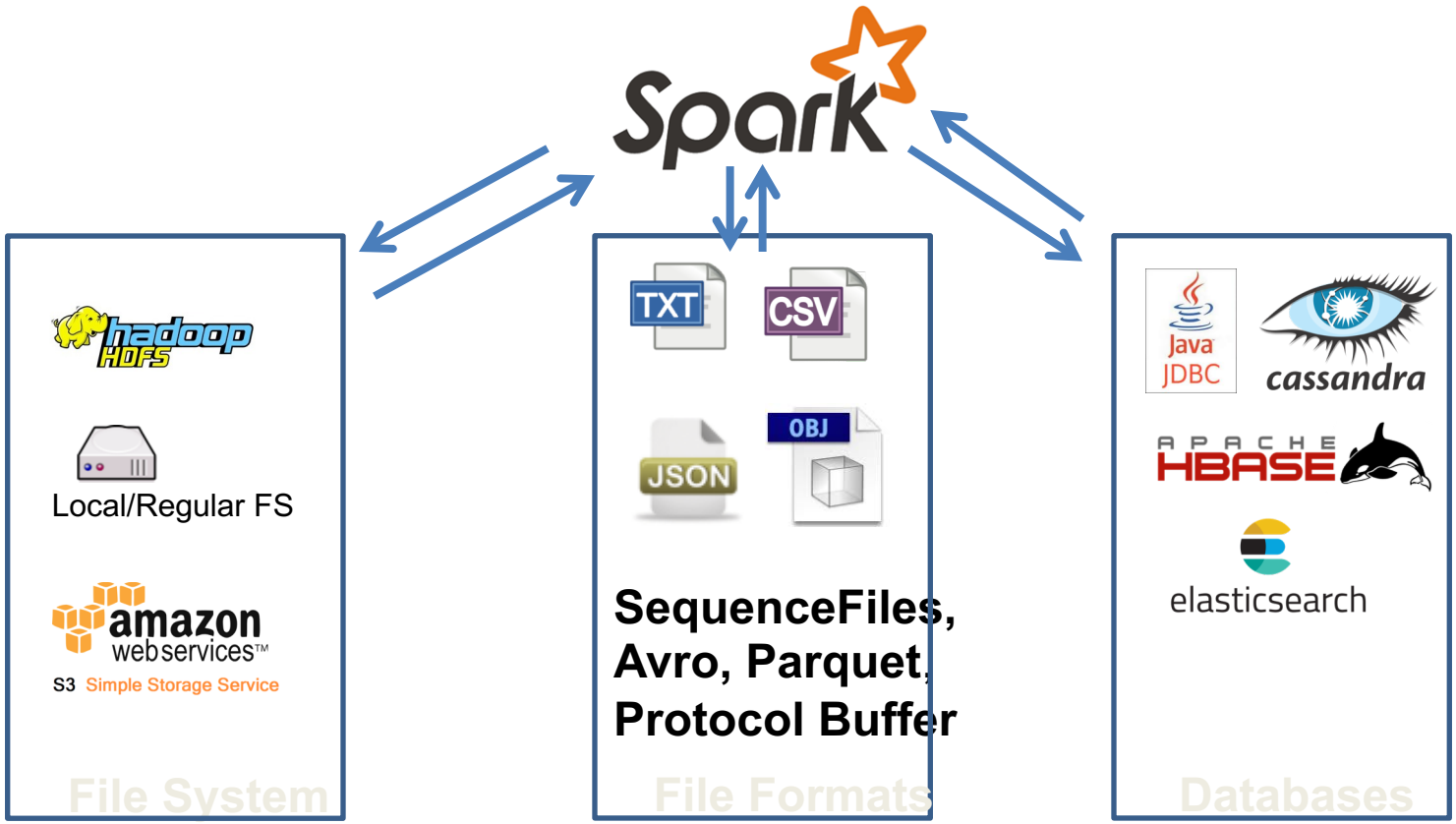
Driver and Executor Log

The exact location of Spark's logfiles depends on the deployment mode:

- **In Spark's Standalone mode**: application logs are directly displayed in the standalone master's web UI. They are stored by default in the *work/* directory of the Spark distribution on each worker.
- **In Mesos**: logs are stored in the *work/* directory of a Mesos slave, and accessible from the Mesos master UI.
- **In YARN mode**: the easiest way to collect logs is to use YARN's log collection tool (running yarn logs -applicationId <*app ID*>) to produce a report containing logs from your application.

UNIVERSITAT DE BARCELONA

Loading and Saving Data in Spark



UNIVERSITAT DE BARCELONA

File Formats

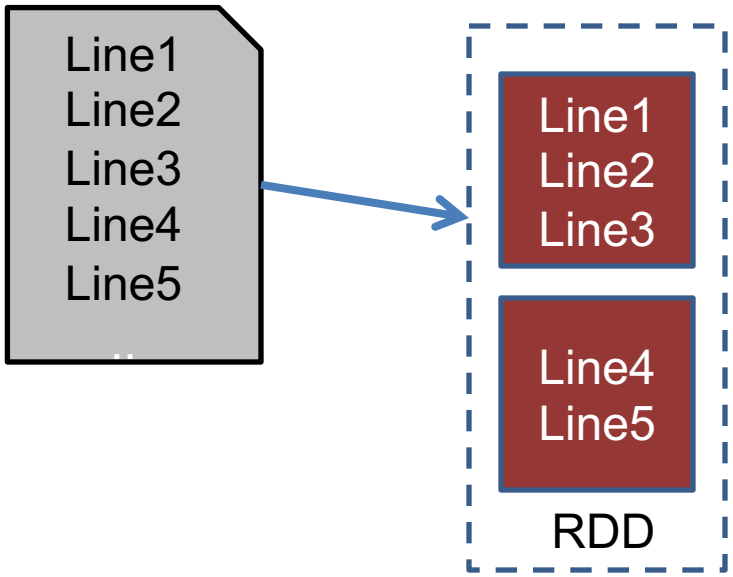
Format	Structured	Comments
Text	No	Plain old text files. Records are assumed to be one per line.
JSON	Semi	Common text-based format, semistructured; most libraries require one record per line.
CSV	Yes	Very common text-based format, often used with spreadsheet applications.
SequenceFiles	Yes	A common Hadoop file format used for key/value data.
Protocol Buffer	Yes	A fast, space-efficient multilanguage format.
Object	Yes	Useful for saving data from a Spark job to be consumed by shared code. Breaks if you change your classes, as it relies on Java Serialization.

UNIVERSITAT DE BARCELONA

Loading Text Files

Loading a single text file

input = sc.textFile("file://fullpath of the file")

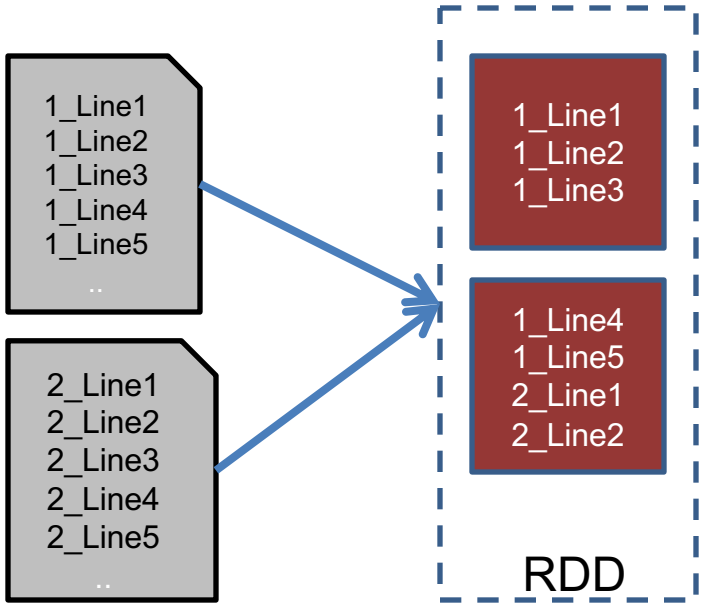


UNIVERSITAT DE BARCELONA

Loading Text Files

Loading a multiple text file

input = sc.textFile("file://fullpath of the folder")

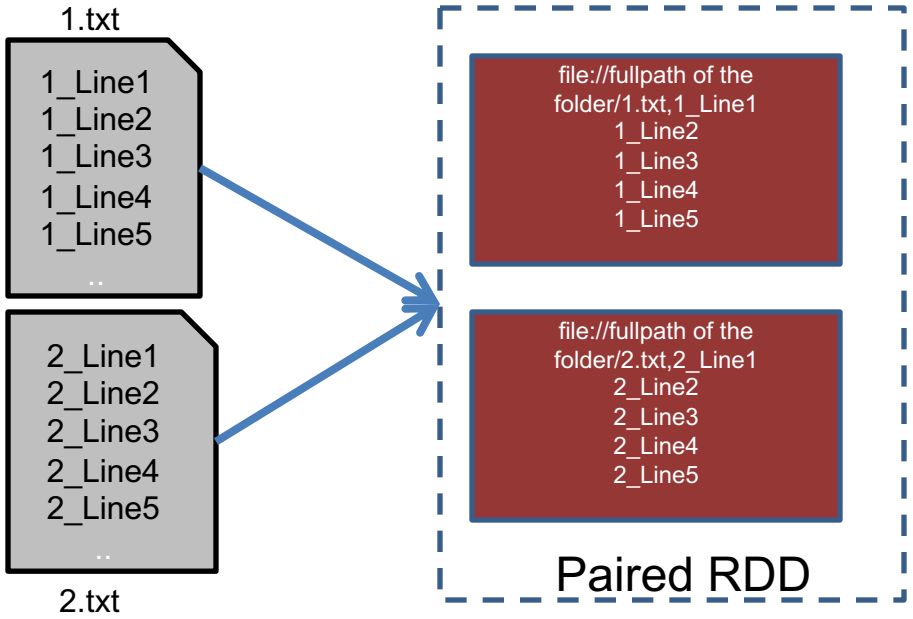


UNIVERSITAT DE BARCELONA

Loading Text Files

Loading a multiple text file

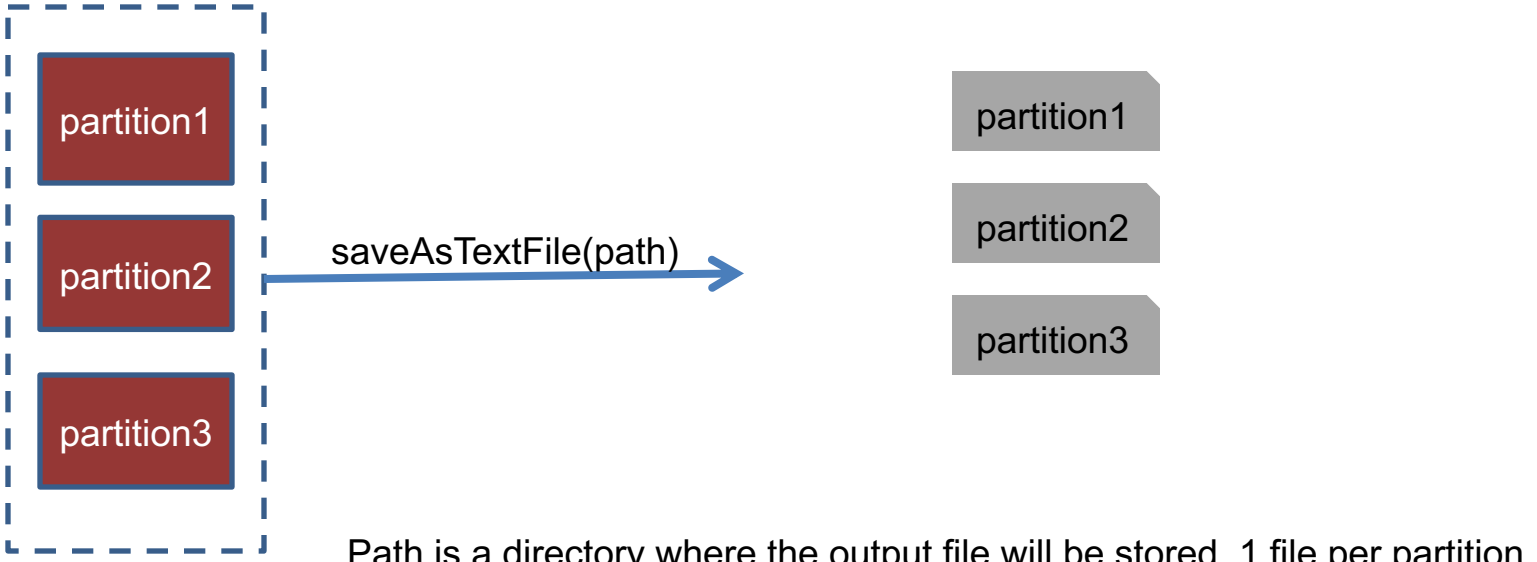
input = sc.wholeTextFiles("file://fullpath of the folder")



Spark supports reading all the files in a given directory and doing wildcard expansion on the input (e.g., part-*.txt).

UNIVERSITAT DE BARCELONA

Saving Text File



Path is a directory where the output file will be stored. 1 file per partition.

UNIVERSITAT DE BARCELONA

Loading JSON files

line delimited JSON

```
{"string":"string1","int":1,"array":[1,2,3],"dict": {"key": "value1"}}
{"string":"string2","int":2,"array":[2,4,6],"dict": {"key": "value2"}}
{"string":"string3","int":3,"array":[3,6,9],"dict": {"key": "value3", "extra_key": "extra_value3"}}
```

import json

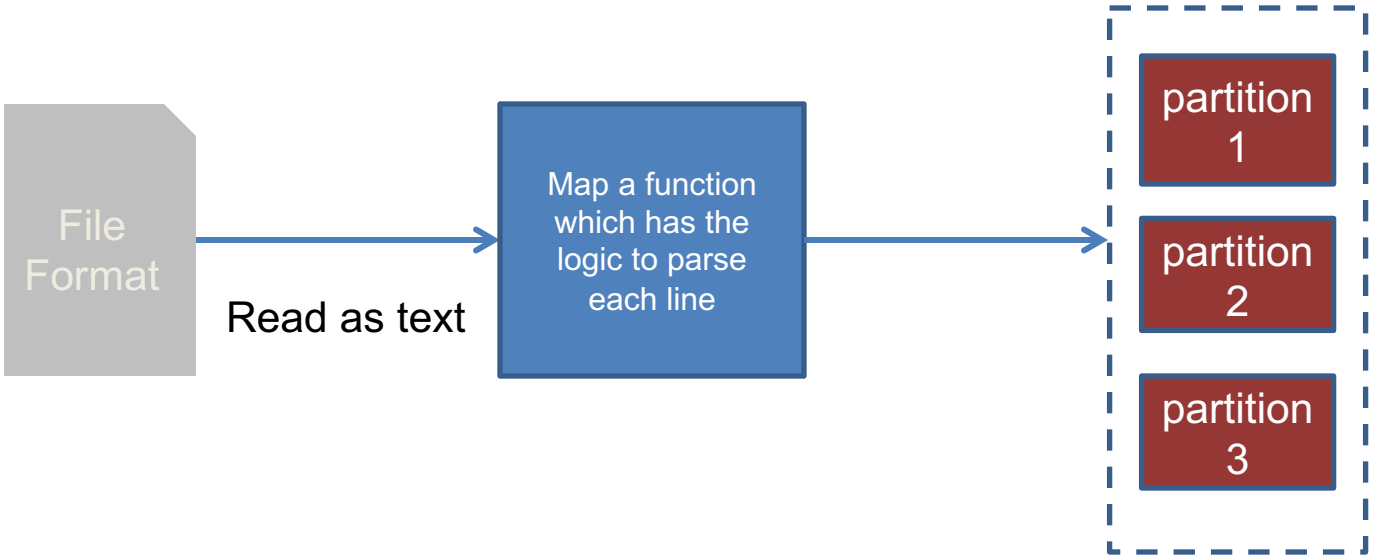
input = sc.textFile("file://fullpath of the file")

data = input.map(lambda x: json.loads(x))

Easy way is to use Spark SQL

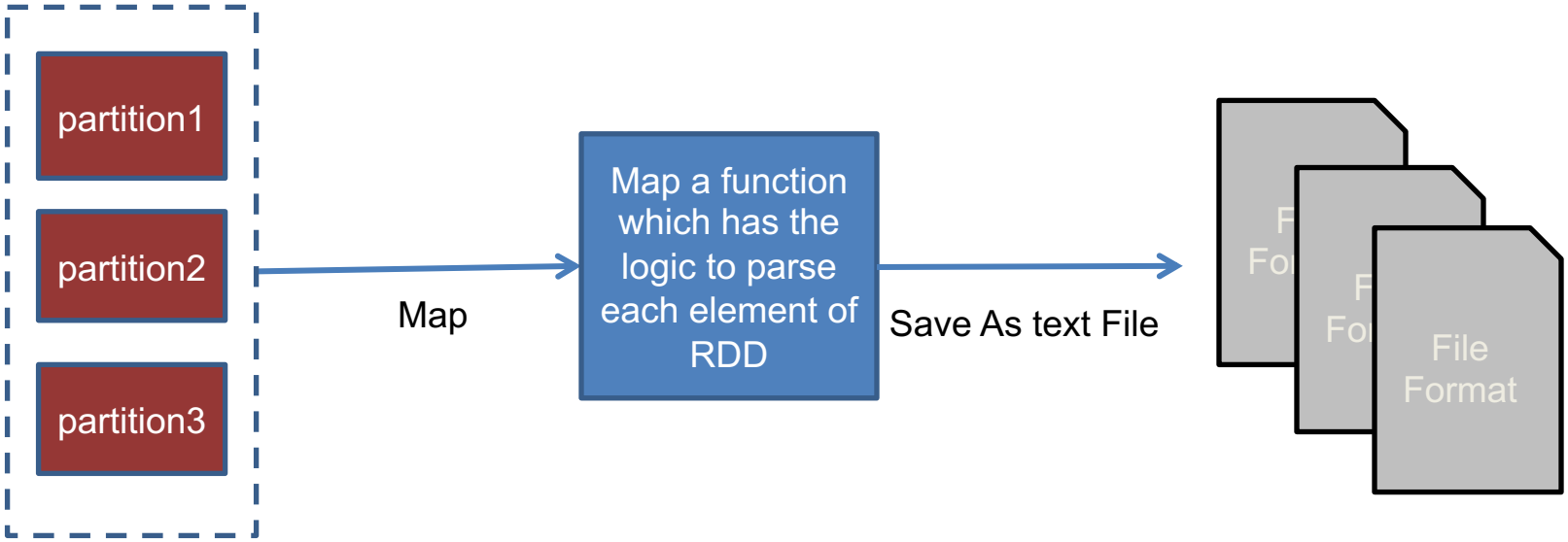
UNIVERSITAT DE BARCELONA

General Principle Reading



UNIVERSITAT DE BARCELONA

General Principle Writing



UNIVERSITAT DE BARCELONA

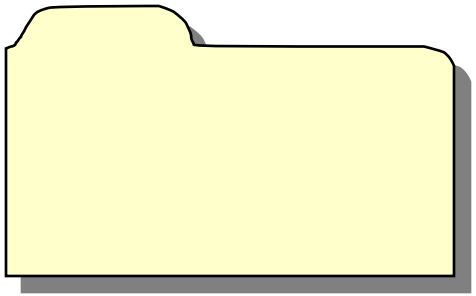
Writable classes

Scala	Java	Hadoop Writable
Int	Integer	IntWritable
Long	Long	LongWritable
Float	Float	FloatWritable
Double	Double	DoubleWritable
Boolean	Boolean	BooleanWritable
Array[Byte]	byte[]	BytesWritable
String	String	Text
Array[T]	T[]	ArrayWritable<TW>3
List[T]	List<T>	ArrayWritable<TW>3
Map[A,B]	Map<A,B>	MapWritable<AW,BW>

The Python Spark API knows only how to convert the basic Writables available in Hadoop to Python, and makes a best effort for other classes based on their available getter methods.

UNIVERSITAT DE BARCELONA

File Compression



Spark can read from compressed data source but not all type could be distributed.

UNIVERSITAT DE BARCELONA

Compression Options

Format	Splittable	Average compression speed	Effectiveness on text	Hadoop compression codec	Comments
gzip	N	Fast	High	org.apache.hadoop.io.compress.GzipCodec	
lzo	Y	Very Fast	Medium	com.hadoop.compression.lzo.LzoCodec	LZO requires installation on every worker node
bzip2	Y	Slow	Very High	org.apache.hadoop.io.compress.BZip2Codec	Uses pure Java for splittable version
zlib	N	Slow	Medium	org.apache.hadoop.io.compress.DefaultCodec	Default compression codec for Hadoop.

UNIVERSITAT DE BARCELONA

File System

- Spark supports many file systems will go through some of them.

Local/Regular FS

UNIVERSITAT DE BARCELONA

Local/Regular FS

rdd **=** sc.textFile("file://path to the file")

The **Path** could be a shared NFS system or local system.

UNIVERSITAT DE BARCELONA

HDFS

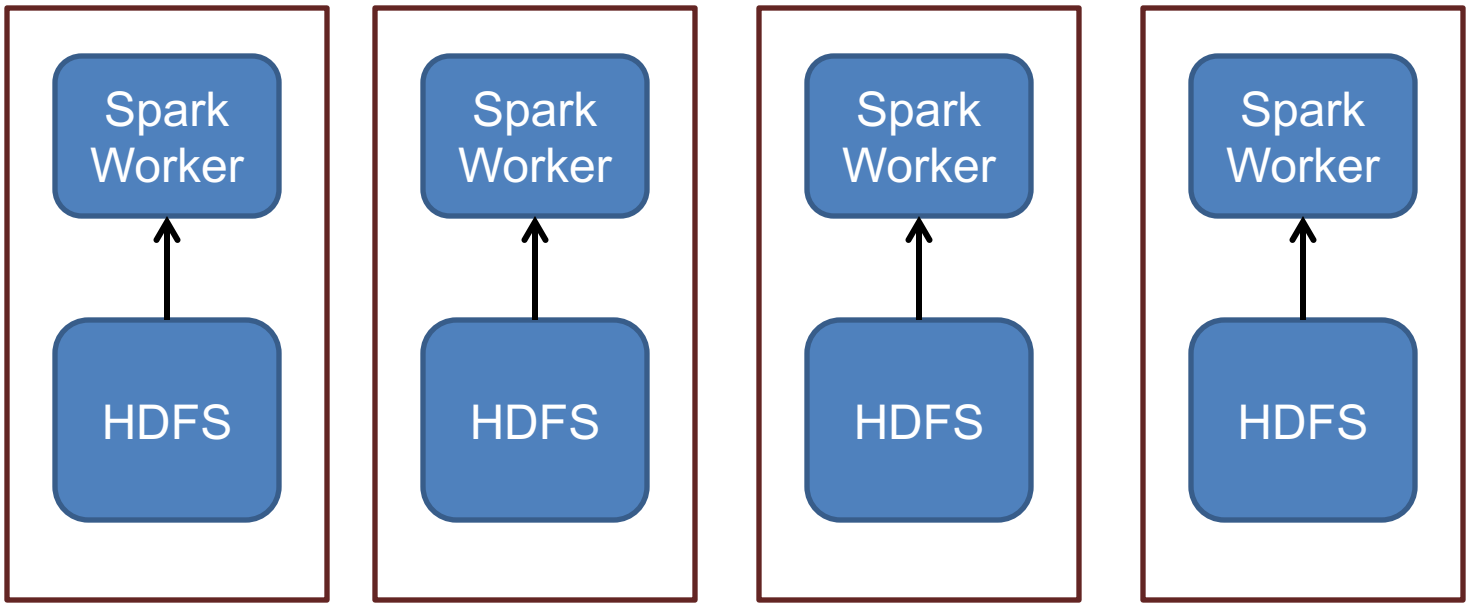
sc.textFile("hdfs://master:port/path")

- Spark and HDFS can be collocated on the same machines, and Spark can take advantage of this data locality to avoid network overhead

The HDFS protocol changes across Hadoop versions, so if you run a version of Spark that is compiled for a different version it will fail. If you build from source, you can specify SPARK_HADOOP_VERSION= as a environment variable to build against a different version; or you can download a different precompiled version of Spark.

UNIVERSITAT DE BARCELONA

Data locality



Every worker will load data from local hdfs partition in parrallel.

UNIVERSITAT DE BARCELONA

Databases

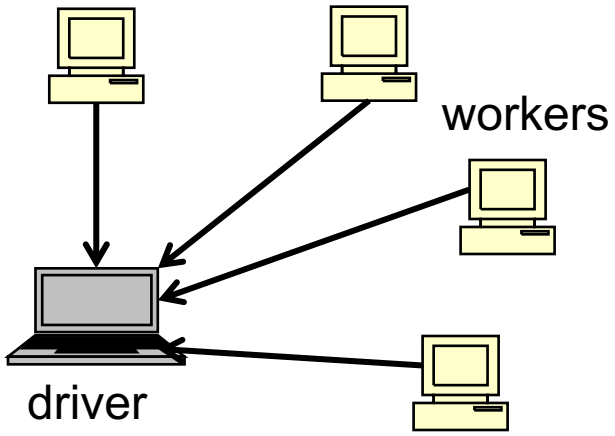
- Spark can access several popular databases using either Hadoop connectors or custom Spark Connectors.

You can write your own connector for a database!

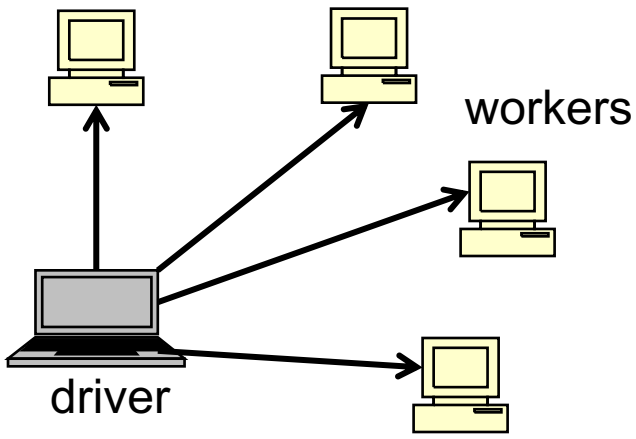
UNIVERSITAT DE BARCELONA

Spark Programing Advanced

Accumulators



Broadcast Variables



UNIVERSITAT DE BARCELONA

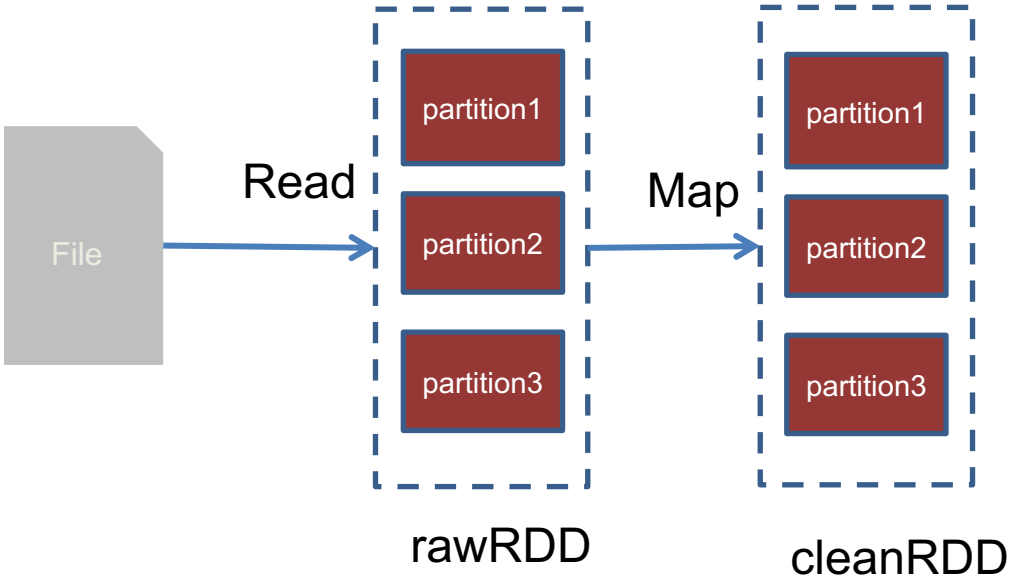
Accumulators

Provides a simple syntax for aggregating values from worker nodes back to the driver program

Example: Need to keep count of badly formatted lines.

UNIVERSITAT DE BARCELONA

Accumulator Example

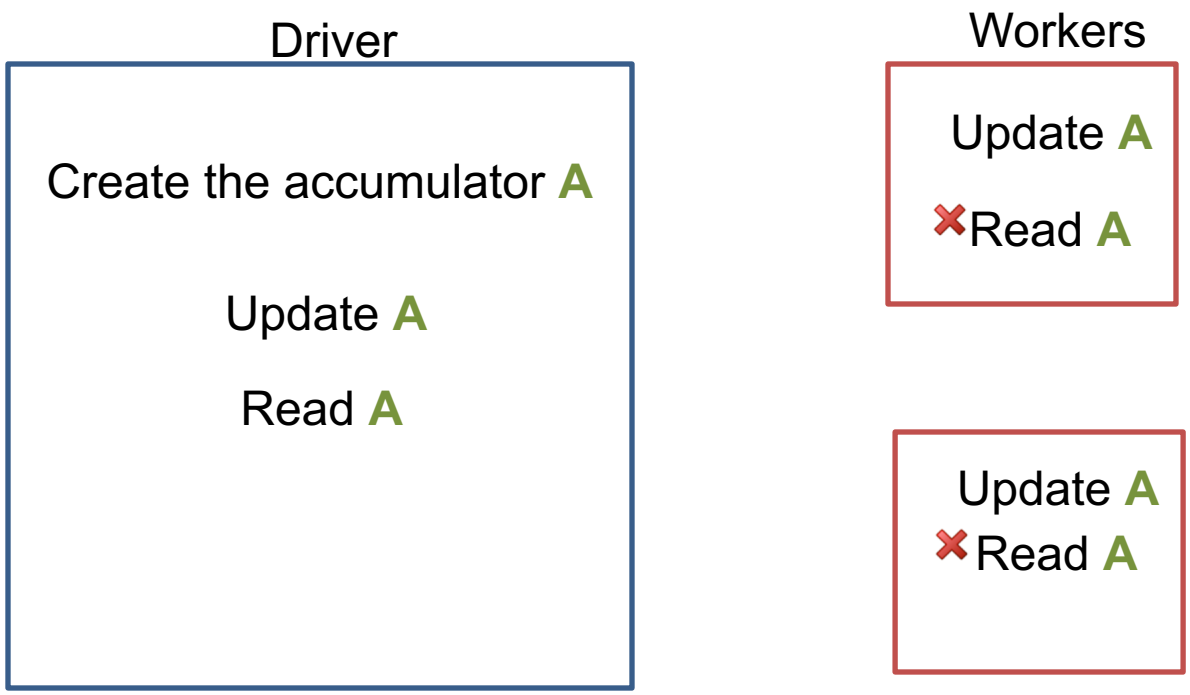


Code:

```
rawRDD= sc.textFile(inputFile)
# Create Accumulator[Int] initialized to 0
corruptLines = sc.accumulator(0)
def extractCallSigns(line):
        # Make the global variable accessible
        global corruptLines
        if (isFormatedWell(line)):
            corruptLines += 1
        return line.split(" ")
cleanRDD = rawRDD.flatMap(extractCallSigns)
cleanRDD.saveAsTextFile(outputDir + "/Data")
print "Corrupt lines: %d" % corruptLines.value
```

UNIVERSITAT DE BARCELONA

Accumulators access



UNIVERSITAT DE BARCELONA

Accumulators and Fault Tolerance

Action	Transformation
Accumulator Updates only Once	Accumulator may do multiple run on re execution.

UNIVERSITAT DE BARCELONA

Custom Accumulators

- Spark supports Int, Double, Long and Float accumulators.
- Extend AccumulatorParam to define your custom accumulator.
- Register your custom accumulator with Spark context.
- You can use any operation inside accumulator, provided that operation is ***commutative*** and ***associative***.

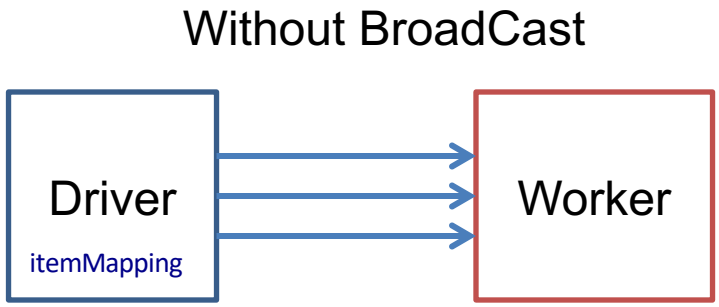
UNIVERSITAT DE BARCELONA

Broadcast Variable

```
itemMapping= loadItemTable()
def processPurchase(sale):
        country = lookupCountry(sale.StateCode, itemMapping)
        count = sale.Count
        return (country, count)
countrySaleCounts = (saleRDD
                .map(processSignCount)
                .reduceByKey((lambda x, y: x+ y)))
```

UNIVERSITAT DE BARCELONA

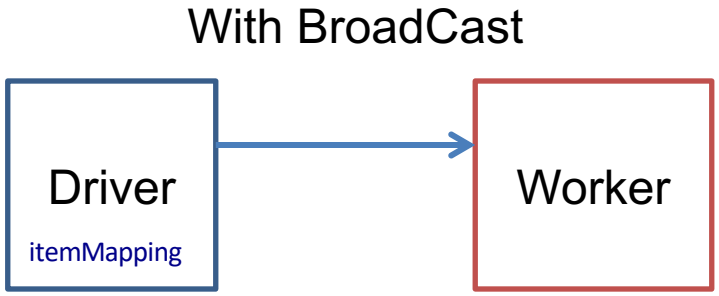
Broadcast Variable



saleRDD.map(processSignCount)

saleRDD.map(processSignCount) -> with new data

returnRDD.map(processSignCount) -> with other data



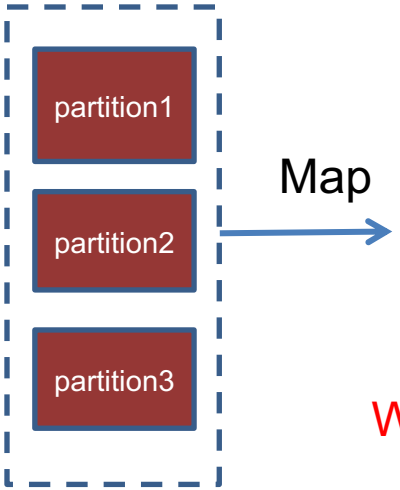
UNIVERSITAT DE BARCELONA

Broadcast Variable

```
itemMapping= sc.broadcast(loadItemTable())
def processPurchase(sale):
        country = lookupCountry(sale.StateCode, itemMapping.value)
        count = sale.Count
        return (country, count)
countrySaleCounts = (saleRDD
                .map(processSignCount)
                .reduceByKey((lambda x, y: x+ y)))
```

UNIVERSITAT DE BARCELONA

Working on a Per-Partition Basis



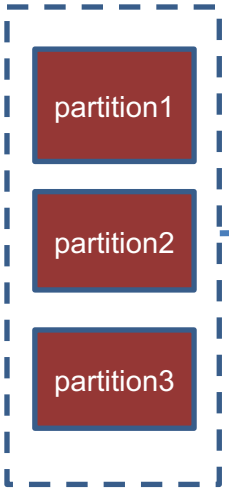
Map

Executes the code for each item in every partition.

What if you want to save the data in a database?

UNIVERSITAT DE BARCELONA

Working on a Per-Partition Basis



mapPartitions

Executes the code for each partitions only once.

Use it for setup work like opening database connection

UNIVERSITAT DE BARCELONA

Working on a Per-Partition Basis

Function name	Called with	Signature
mapPartitions()	Iterator of the elements in that partition	f: (Iterator[T]) → Iterator[U]
mapPartitionsWithIndex()	Integer of partition number, and Iterator of the elements in that partition	f: (Int, Iterator[T]) → Iterator[U]
foreachPartition()	Iterator of the elements	f: (Iterator[T]) → Unit