Dive into Apache Spark

TANG Gen

Outline

- Detailed description of RDD
    - Formalized conception of RDD
    - RDD graph
- Spark architecture
    - Spark runtime architecture
    - Memory design
- Tips for Spark coding
- Testing Spark

Detailed description of RDD

/What is RDD?

- Just for review:

```
val sc = new SparkContext(…)
val data = sc.textFile(“hdfs://”)
val data1 = rdd.filter(_.contains(“…”))
```

- We know that data and data1 are RDDs

- But, what is the formalized conception of RDD?

Detailed description of RDD

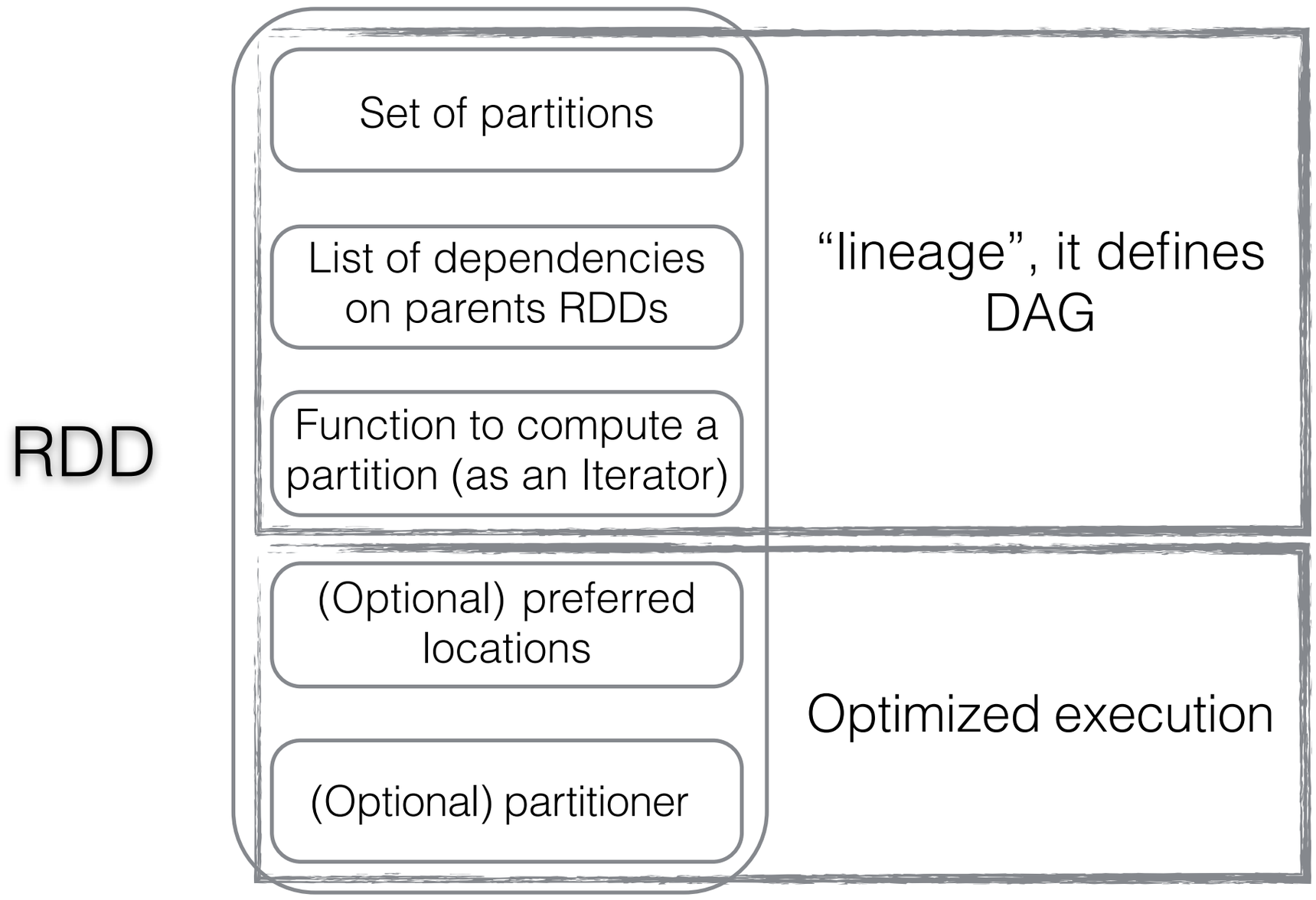
/Formalized conception of RDD

- RDD could be
  - A distributed collection of objects on disk
  - A distributed collection of objects on memory
  - A distributed collection of objects on HDFS
  - A distributed collection of objects on Cassandra
  - …

Detailed description of RDD

/Formalized conception of RDD

- Scientifically, RDD is an **interface** with five members



Detailed description of RDD

/Examples

- For the RDD “data”
  - partitions = one per HDFS block
  - dependencies = None
  - compute = read corresponding block
  - preferredLocations = HDFS block location
  - partitioner = None

Detailed description of RDD

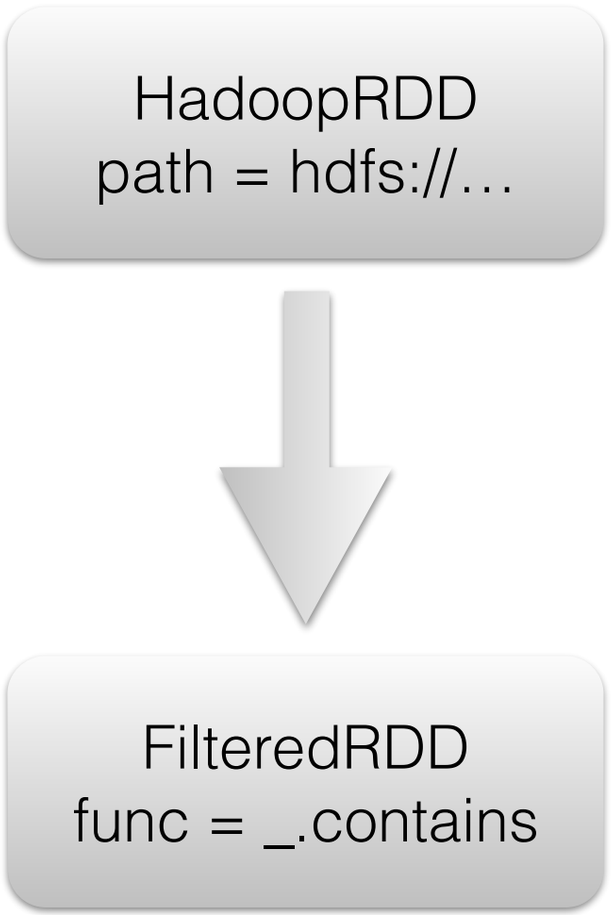
/Examples

- For the RDD “data1”
  - partitions = same as parent RDD
  - dependencies = “one-to-one” on parent
  - compute = compute parent and filter it
  - preferredLocations = None
  - partitioner = None

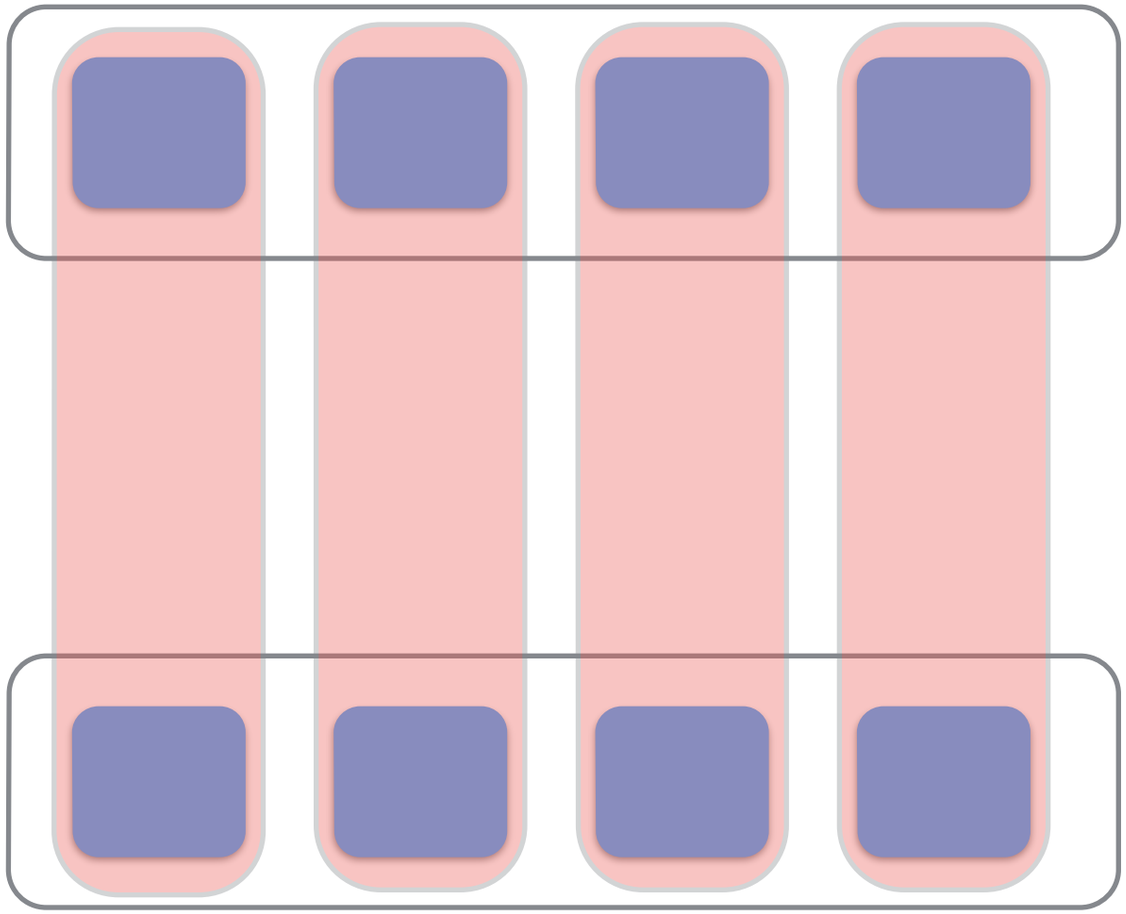
Detailed description of RDD

/RDD graph

Data-level view

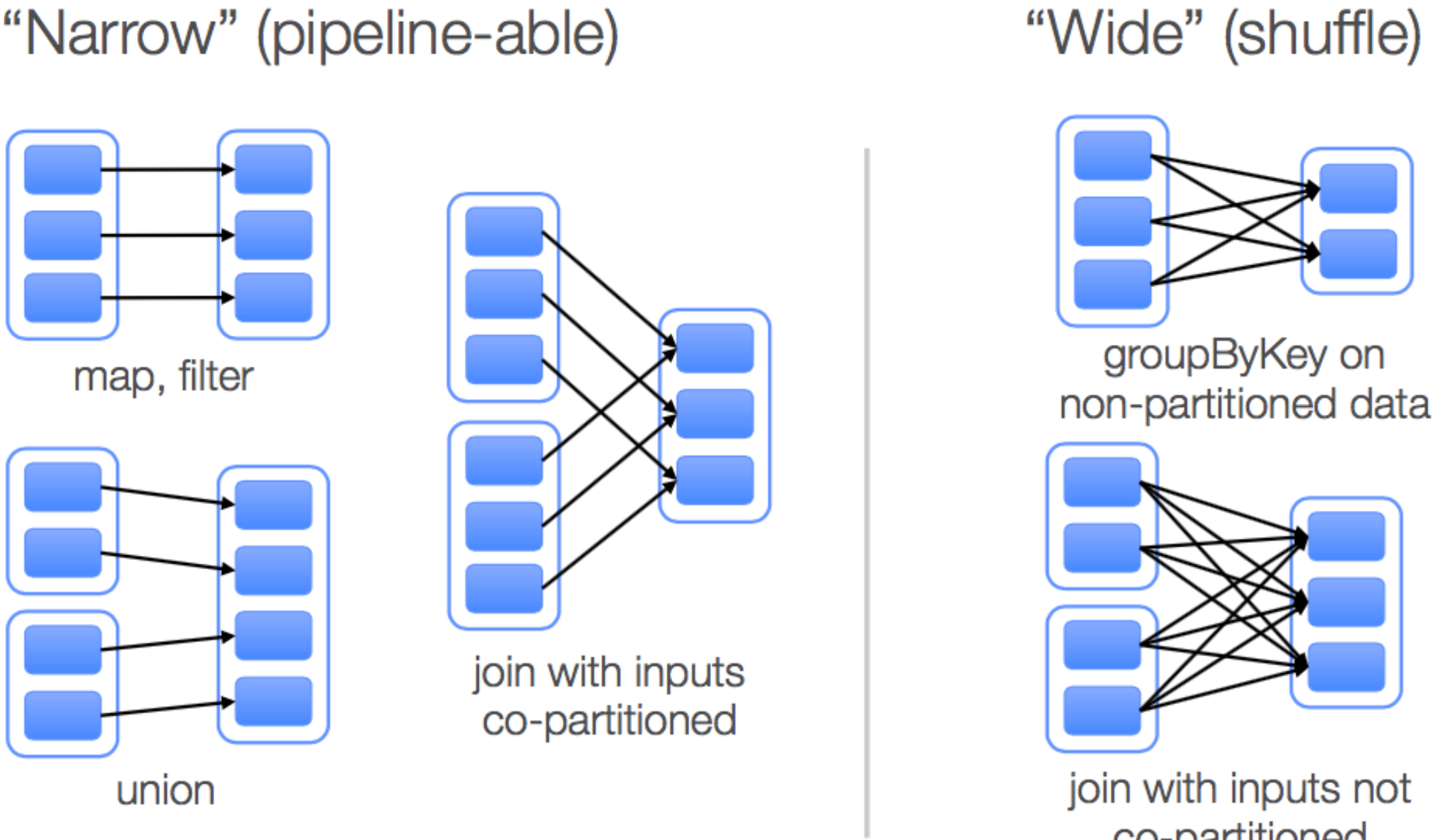


Task-level view



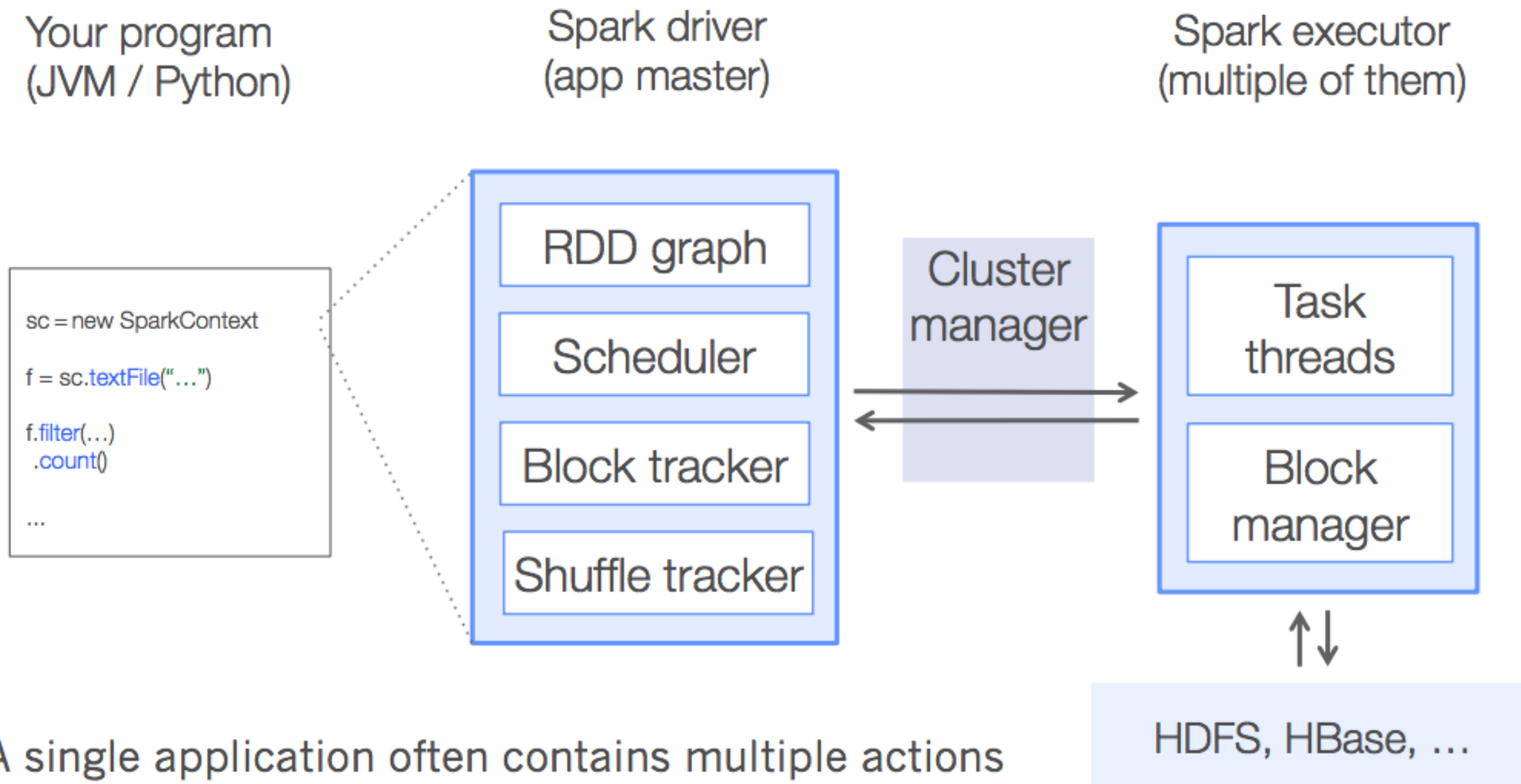
Detailed description of RDD

/RDD's dependencies and stage cutting



Spark architecture

/Runtime structure



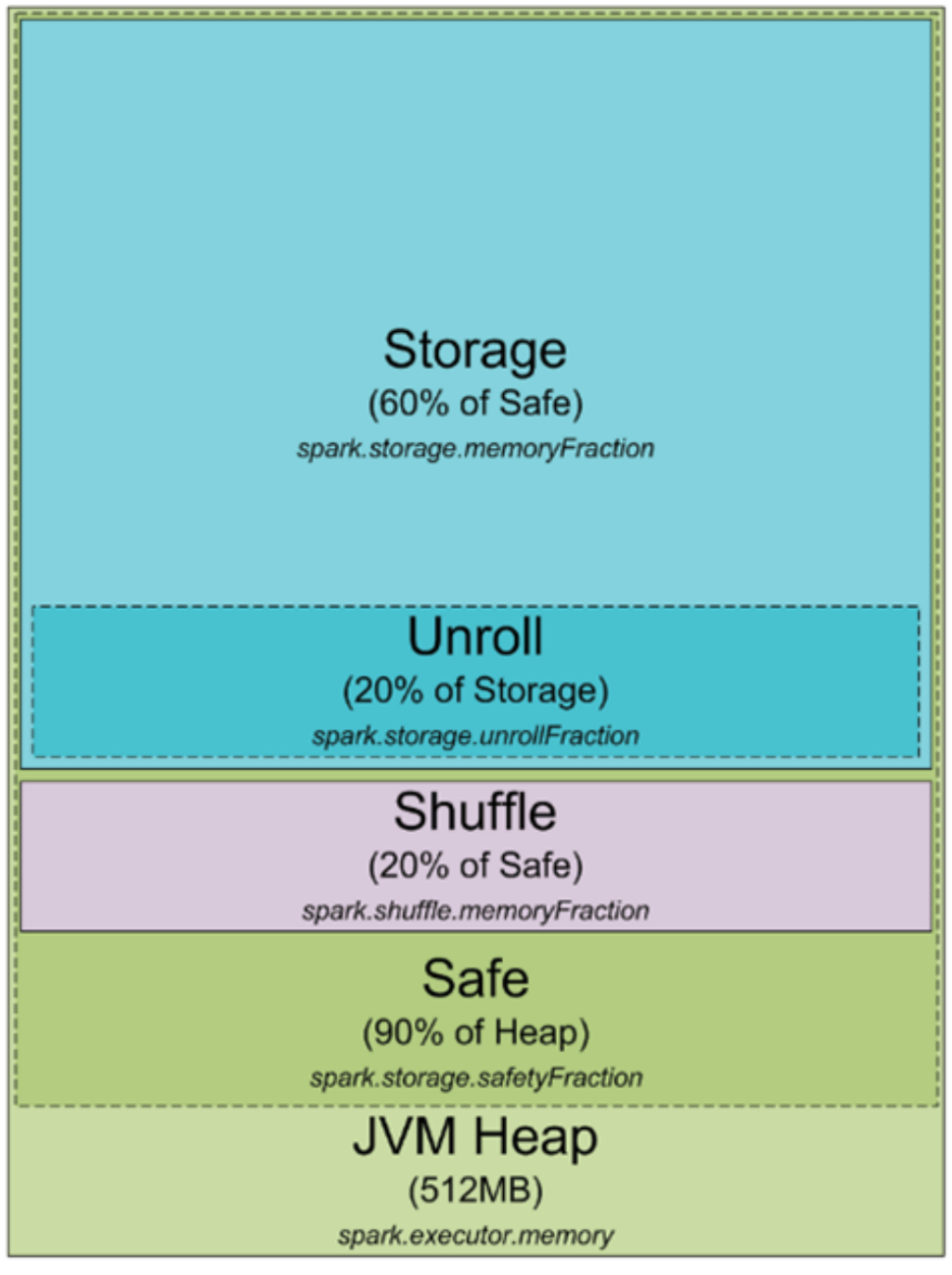
A single application often contains multiple actions

Spark architecture

/Memory design

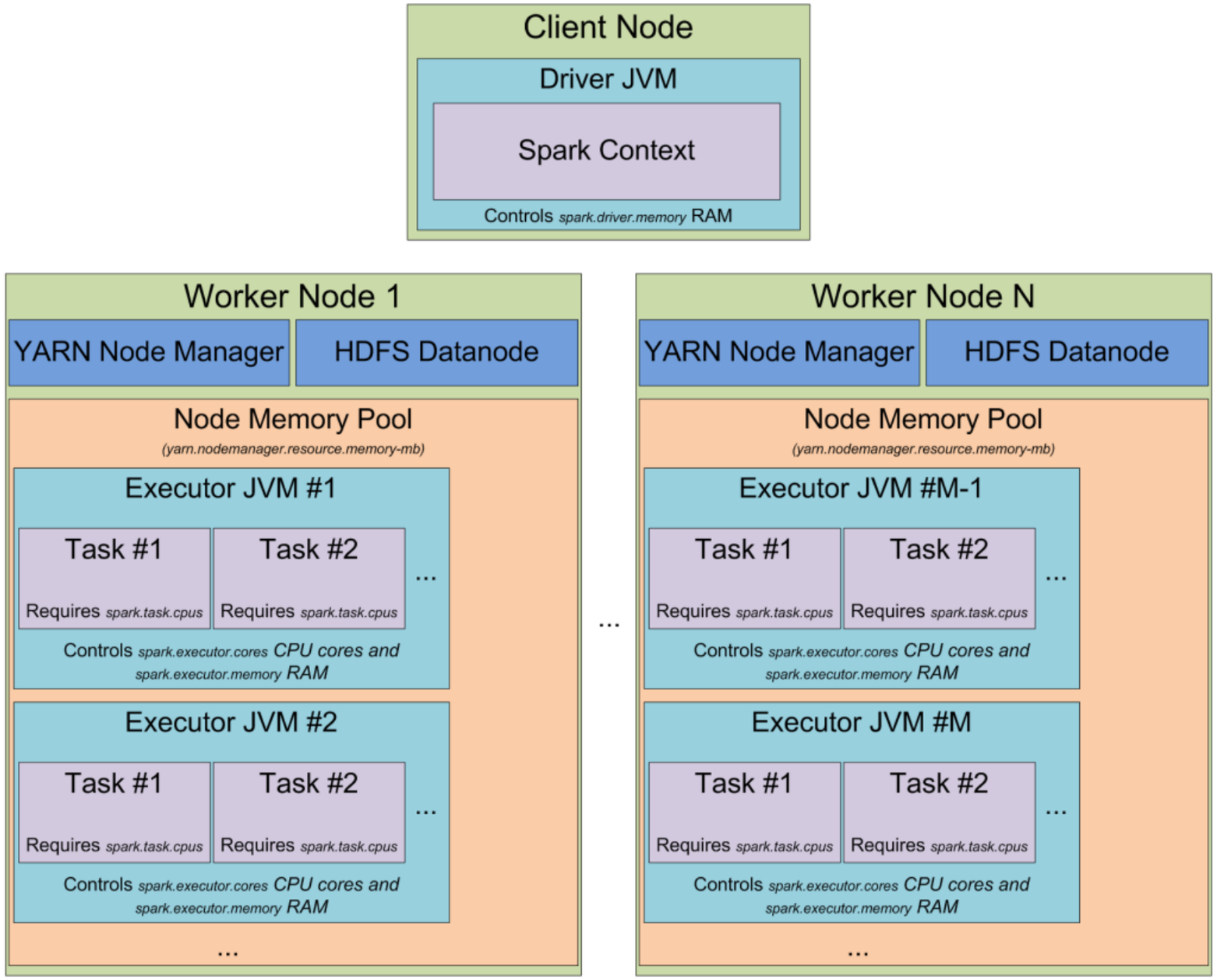
- Spark allows you to store the data both in serialized and deserialized form. The data in serialized form cannot be used directly, so you have to unroll it before using, so this is the RAM that is used for unrolling

- Spark is good in implementing the idea of efficient LRU cache. And Spark use shuffle part to store the sorted data during shuffling



Spark architecture

/Spark on YARN



Tips for Spark coding

/ReduceByKey vs. groupByKey

- Consider the wordcount example and data is RDD[word]
  - Use reduceByKey // [(a, 1), (a, 2)] => [(a, 3)]

```
data.map( x => (x, 1))
    .reduceByKey( {case(a, b) => a + b} )
```

  - Use groupByKey // [(a, 1), (a, 2)] => [(a, [1, 2])]

```
data.map( x => (x, 1))
    .groupByKey()
    .map( {case(a, b) => (a, b.reduce(_ + _))} )
```

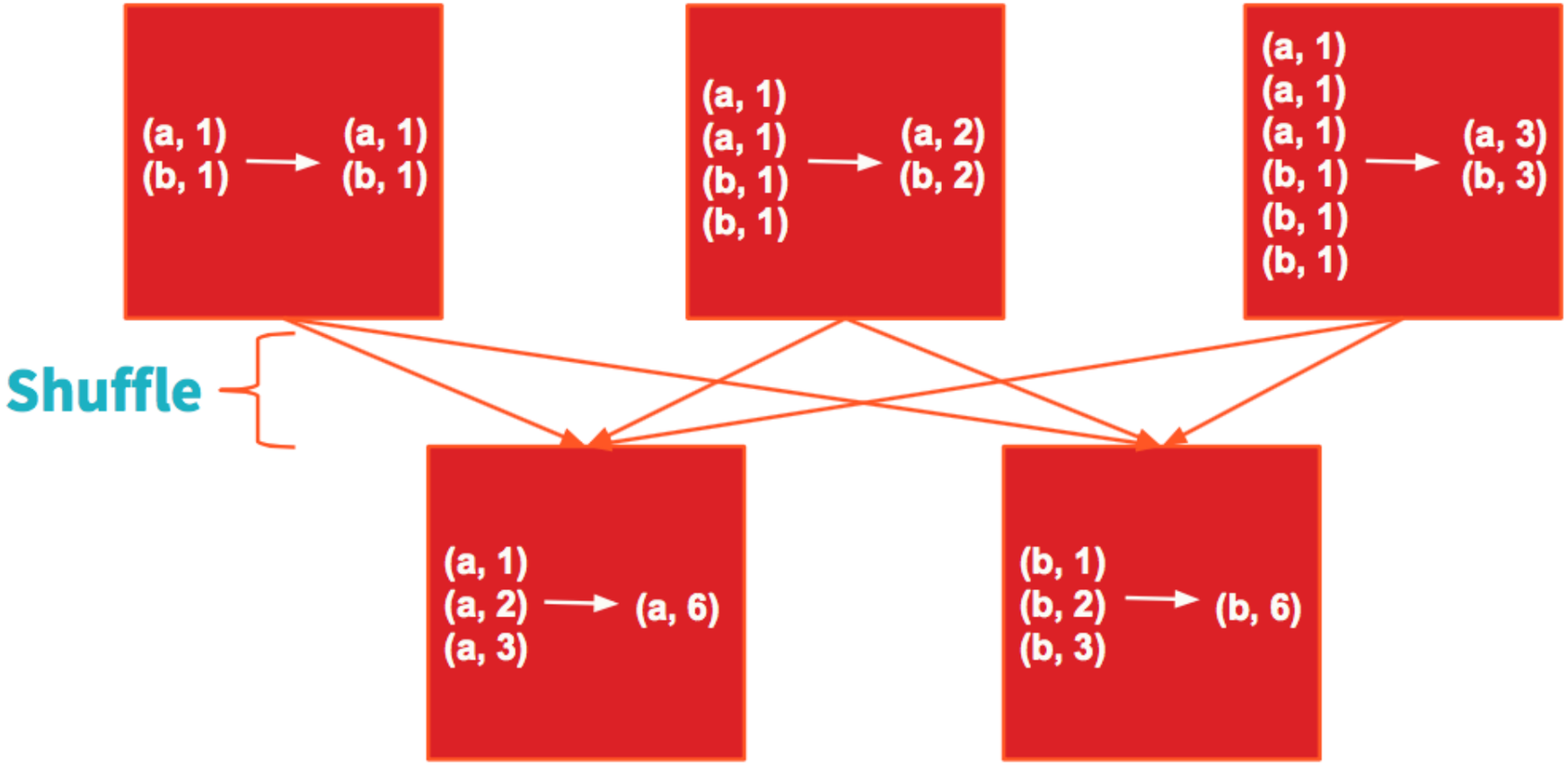
- Both will give you the same answer, but reduceByKey is more efficient

Tips for Spark coding

/ReduceByKey vs. groupByKey

- Both reduceByKey and groupByKey triggers shuffle of data.

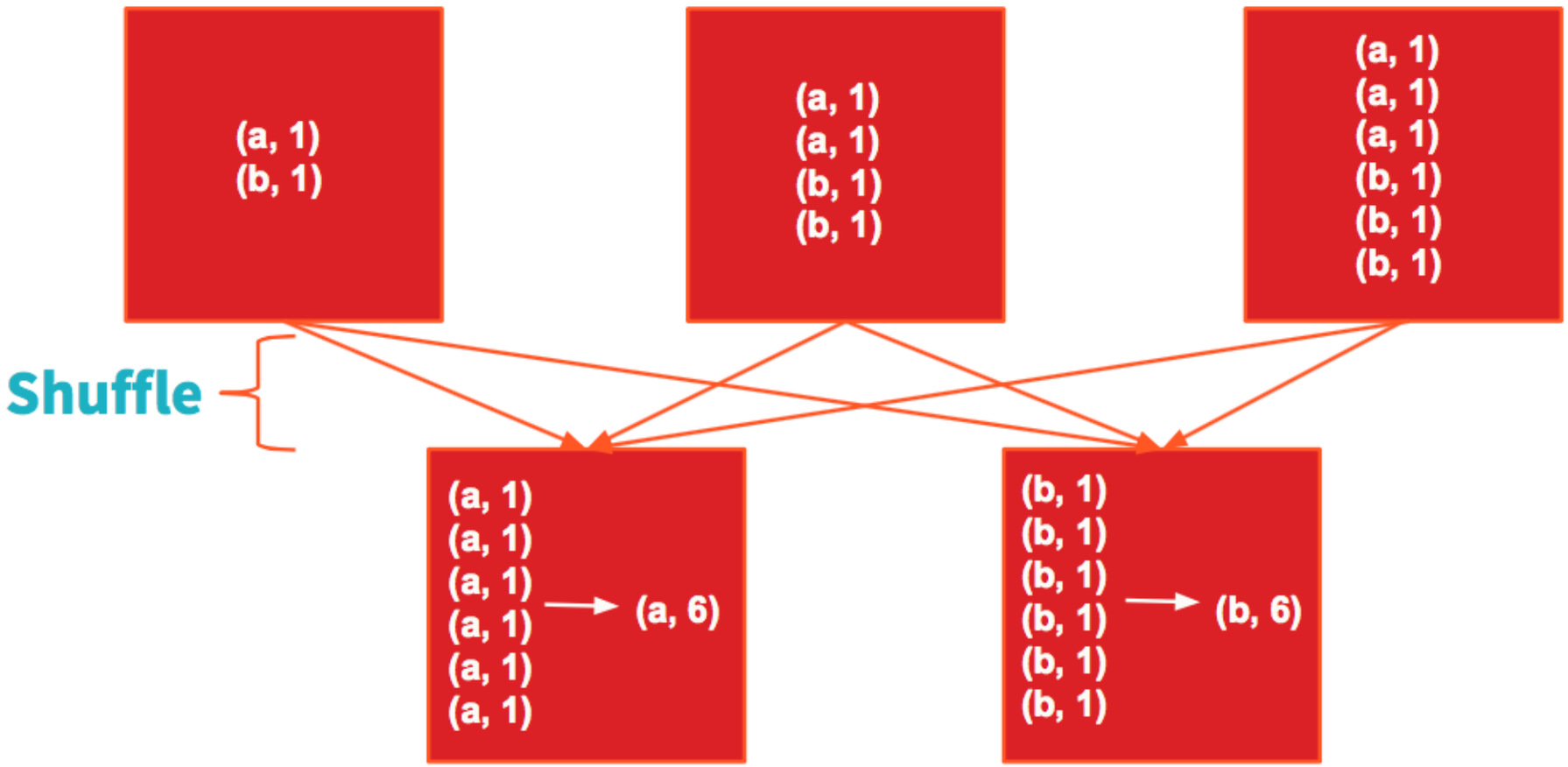
- ReduceByKey: shuffle step



Tips for Spark coding

/ReduceByKey vs. groupByKey

- GroupByKey: shuffle step



- Compared to reduceByKey, all the data is wastefully sent over the network and collected on the reduce workers

Tips for Spark coding

/Join

- Consider the following join a large table with a small table

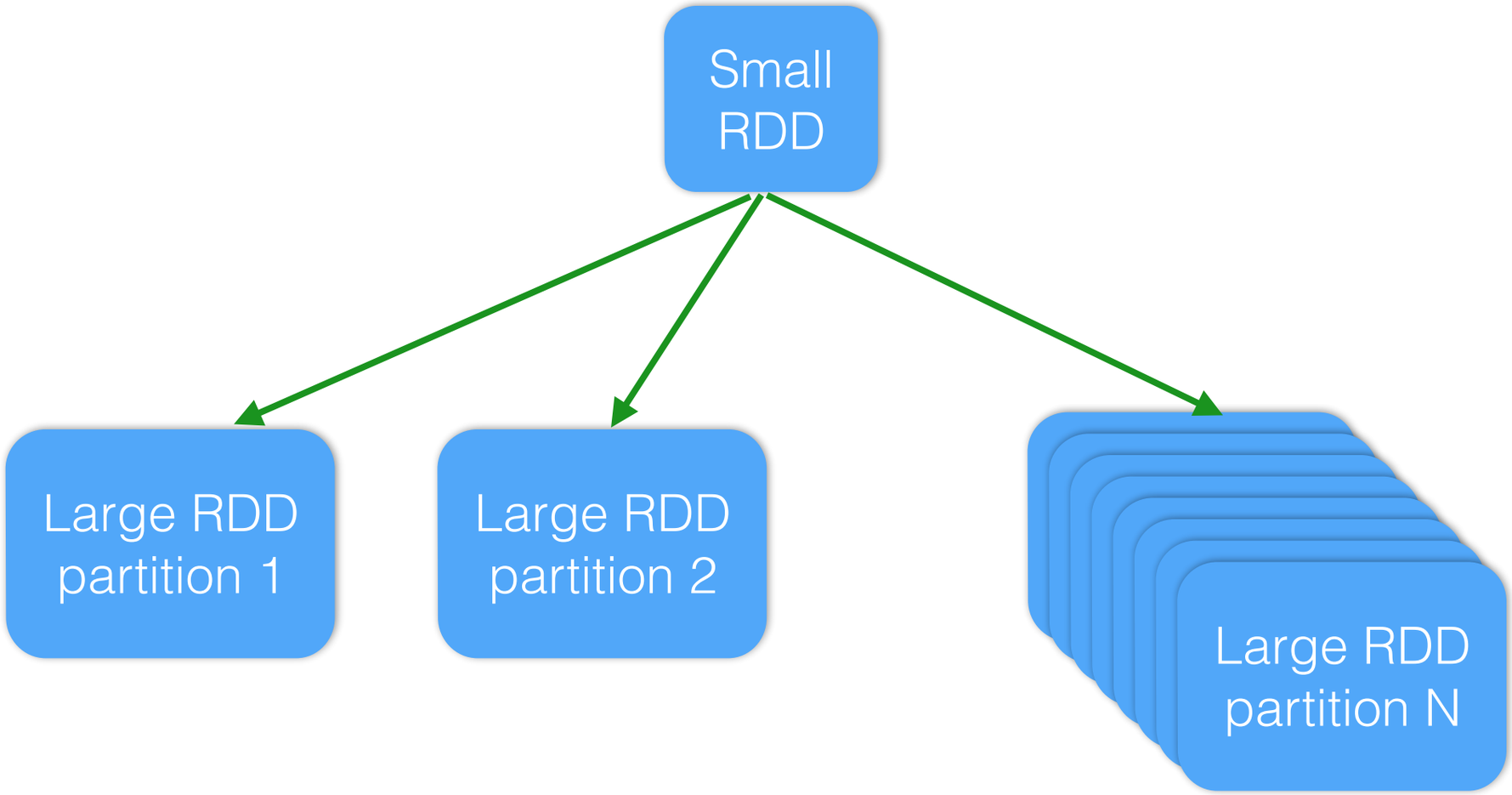
```
join_rdd = sqlContext.sql("select
    FROM people_in_China
    JOIN ShangHai
    ON people_in_China.province = ShangHai.name")
```

- If we use ShuffledHashJoin, all the data for China will be shuffled into only XX keys (one key for each province)
  - Uneven sharding
  - Limited parallelism with XX partitions

Tips for Spark coding

/Join

- Solution: BroadcastHashJoin; Broadcast the small RDD to all worker nodes



Tips for Spark coding

/Join

- How to configure BroadcastHashJoin
  - Set spark.sql.autoBroadcastJoinThreshold
  - sqlContext.sql("ANALYZE TABLE result_table COMPUTE STATISTICS noscan")
- Use RDD.toDebugString() or EXPLAIN to double check

Testing Spark

/Unit-tests of functions

instead of:

```
val splitLines = inFile.map(line => {
  val reader = new CSVReader(new StringReader(line))
  reader.readNext()
})
```

write:

```
def parseLine(line: String): Array[Double] = {
  val reader = new CSVReader(new StringReader(line))
  reader.readNext().map(_.toDouble)
}
```

then we can:

```
test("should parse a csv line with numbers") {
    MoreTestableLoadCsvExample.parseLine("1,2") should equal
(Array[Double](1.0, 2.0))
}
```

Testing Spark

/Testing with RDDs

- Include http://spark-packages.org/package/holdenk/spark-testing-base

spark-testing-base (homepage)

Base classes to use when writing tests with Spark

@holdenk / ⭐⭐⭐⭐⭐ (👤4)

The spark-testing-base package makes it easy to test your Spark code. Starting with v0.0.5, this package is cross-compiled against different Spark versions (1.1.1, 1.2.0, 1.3.0). The version number is [spark-version]_[spark-testing-version]. So, for example, to include this for Spark version 1.2.0 use a version number of 1.2.0_0.0.5.

Tags

3 testing | 1 streaming | 1 tools

How to [+]

Include this package in your Spark Applications using:

spark-shell, pyspark, or spark-submit

```
> $SPARK_HOME/bin/spark-shell --packages holdenk:spark-testing-base:1.3.0_0.0.5
```

Releases

Version: 1.3.0_0.0.5 (43fcb2 | zip | jar) / Date: 03/30/15 / License: Apache-2.0 / Scala version: 2.10

Spark Scala/Java API compatibility: 1.0.0 - 53% , 1.1.0 - 54% , 1.2.0 - 57% , 1.3.0 - 100%

Version: 1.2.0_0.0.5 (29c36c | zip | jar) / Date: 03/30/15 / License: Apache-2.0 / Scala version: 2.10

Spark Scala/Java API compatibility: 1.0.0 - 88% , 1.1.0 - 91% , 1.2.0 - 100% , 1.3.0 - 98%

Version: 1.1.1_0.0.5 (bec125 | zip | jar) / Date: 03/30/15 / License: Apache-2.0 / Scala version: 2.10

Spark Scala/Java API compatibility: 1.0.0 - 96% , 1.1.0 - 100% , 1.2.0 - 99% , 1.3.0 - 98%

Version: 0.0.5 (24cf50 | zip | jar) / Date: 03/30/15 / License: Apache-2.0 / Scala version: 2.10

Spark Scala/Java API compatibility: 1.0.0 - 53% , 1.1.0 - 54% , 1.2.0 - 57% , 1.3.0 - 100%

Version: 0.0.4 (b77e73 | zip | jar) / Date: 03/30/15 / License: Apache-2.0 / Scala version: 2.10

Spark Scala/Java API compatibility: 1.0.0 - 53% , 1.1.0 - 54% , 1.2.0 - 57% , 1.3.0 - 100%

Version: 0.0.3 (47bb73 | zip | jar) / Date: 03/30/15 / License: Apache-2.0 / Scala version: 2.10

Spark Scala/Java API compatibility: 1.0.0 - 96% , 1.1.0 - 100% , 1.2.0 - 99% , 1.3.0 - 98%

Version: 0.0.2 (47bb73 | zip | jar) / Date: 03/30/15 / License: Apache-2.0 / Scala version: 2.10

Spark Scala/Java API compatibility: 1.0.0 - 96% , 1.1.0 - 100% , 1.2.0 - 99% , 1.3.0 - 98%

Version: 0.0.1 (8dbdc5 | zip | jar) / Date: 02/07/15 / License: Apache-2.0

Case studies

/Spark at OOYALA

https://github.com/ooyala/spark-jobserver

- Company vision for Spark is as a multi-team big data service
- Shares Spark RDDs in one SparkContext among multiple jobs
- REST server for submitting, running, managing Spark jobs and contexts