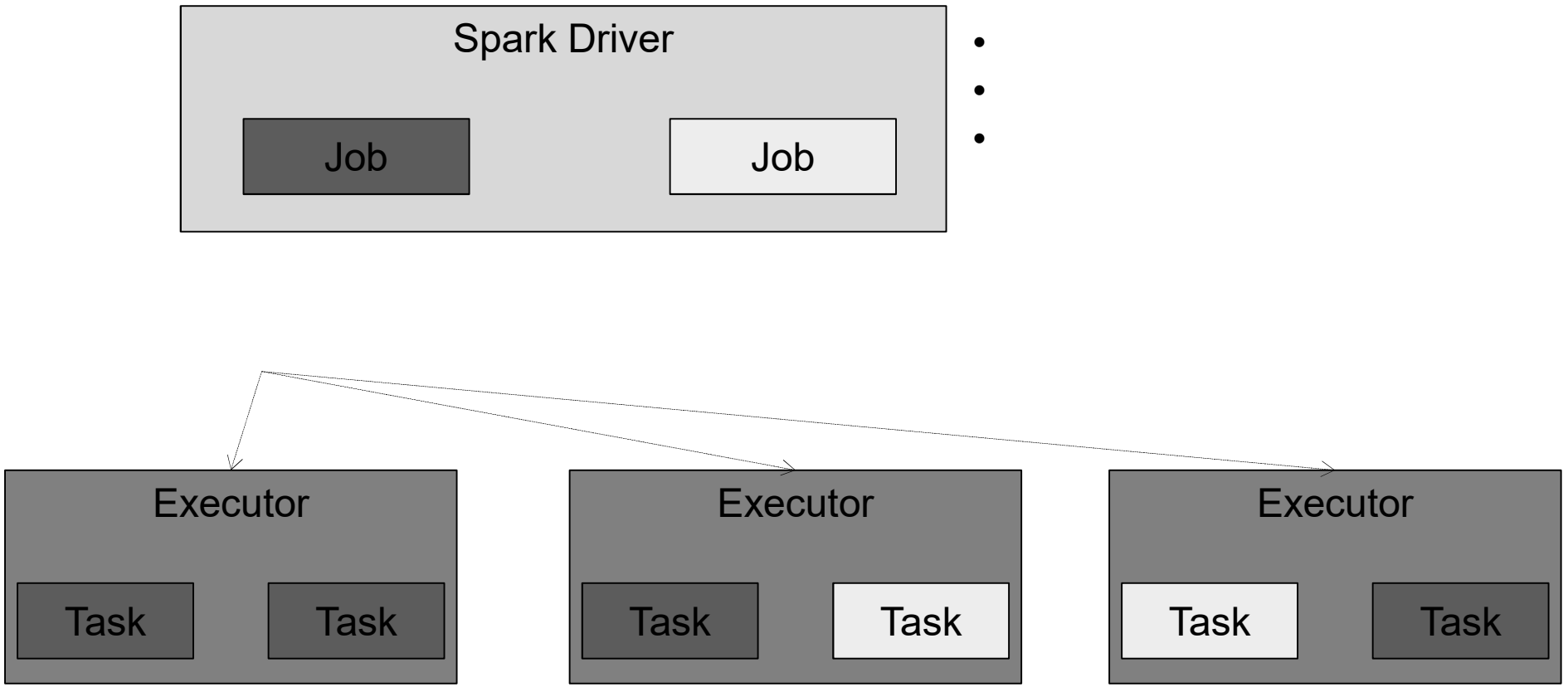
Apache Spark Deep Dive

By

http://researcher.watson.ibm.com/researcher/view.php?person=il-ROYL

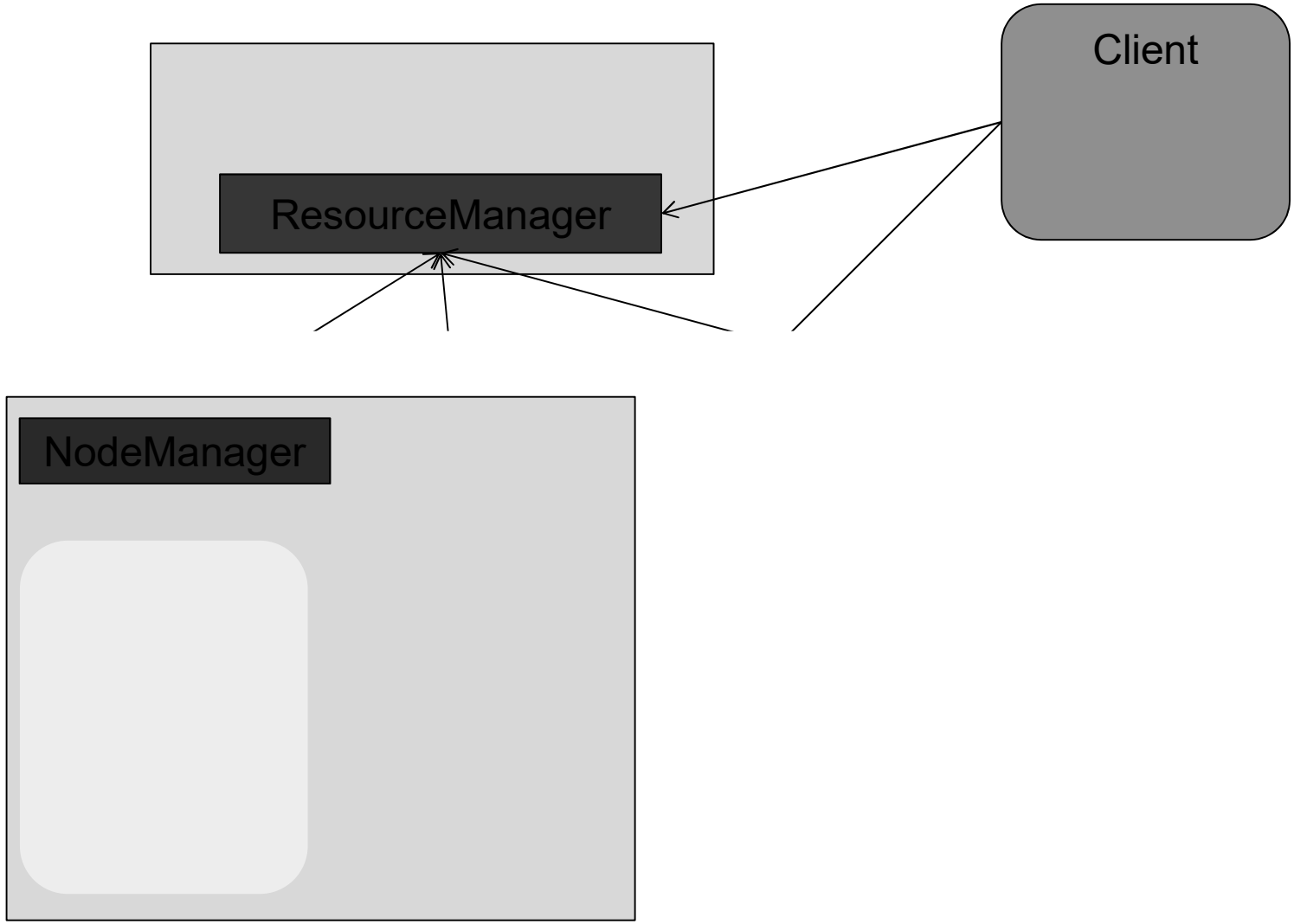
Spark Architecture

Spark Architecture

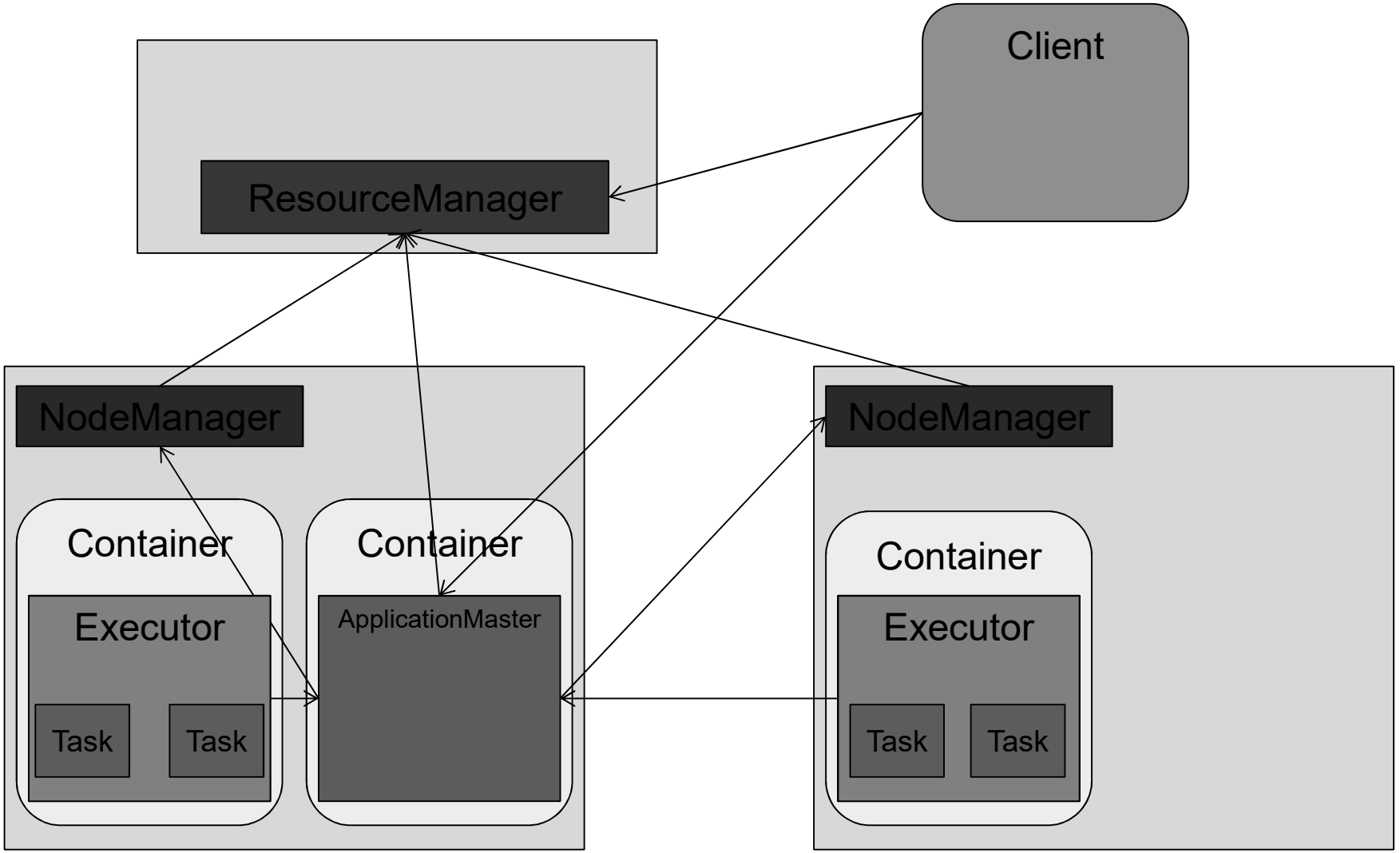


* Note: this is in contrast to MR where each task runs in its own container

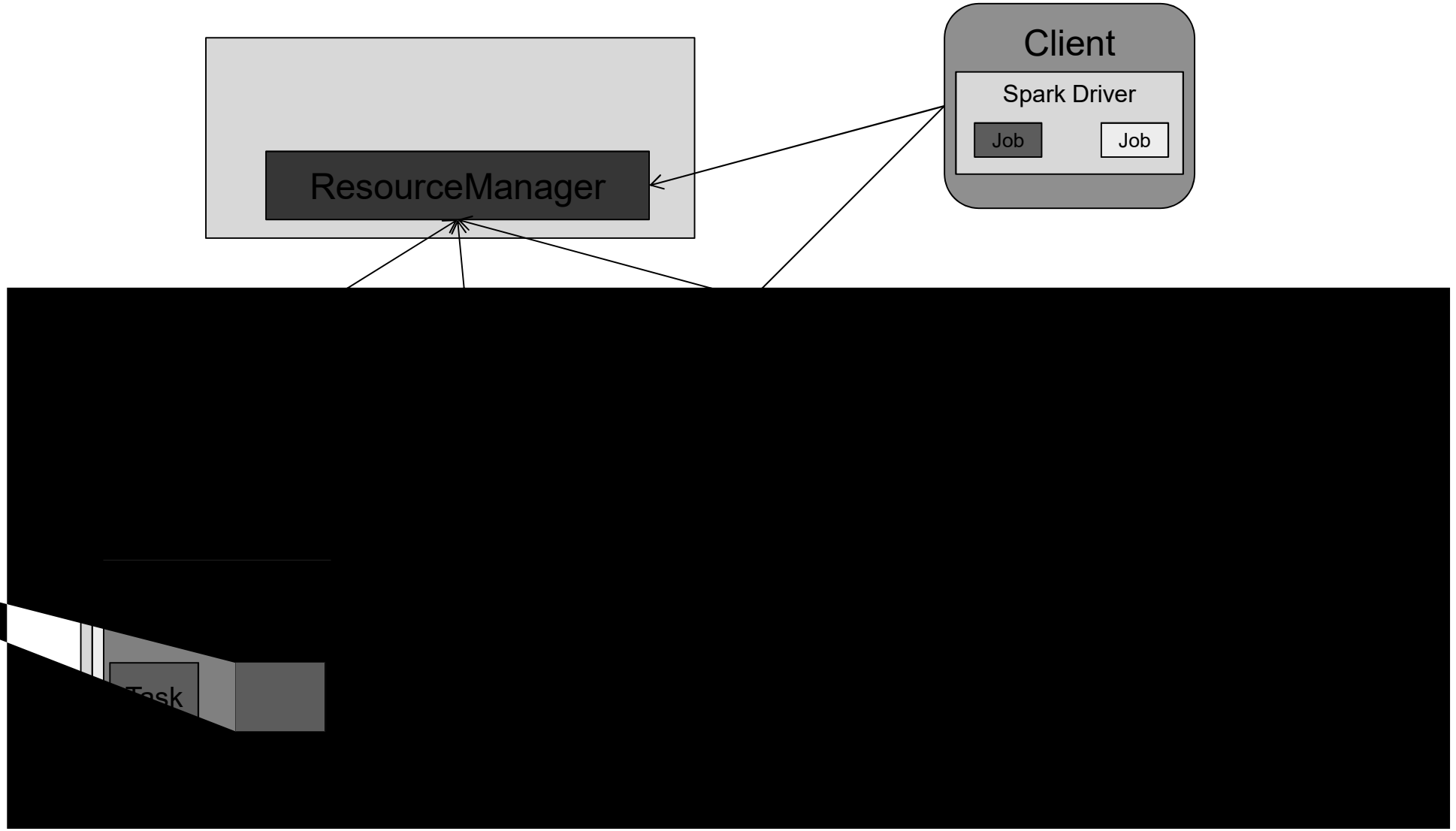
How Does Spark Run Over YARN?



How Does Spark Run Over YARN?



Where Does the Driver Go?



Controlling Spark on YARN

- Controlling number of executors/tasks:
  - (or) controls the number of executors that will run on the cluster
  - controls the number of tasks per executor

- For the exact technical details of how to run Spark on YARN see:
  - http://spark.apache.org/docs/latest/running-on-yarn.html
- For more general info also see:
  - http://badrit.com/blog/2015/2/29/running-spark-on-yarn#.VshHbZx96Uk
  - http://blog.cloudera.com/blog/2015/03/how-to-tune-your-apache-spark-jobs-part-2/

Translating Your Code Into Physical Execution

RDD API Example

```
// Read input file
val input = sc.textFile("input.txt")

val tokenized = input
  .map(line => line.split(" "))
  .filter(words => words.size > 0) // remove empty lines

val counts = tokenized    // frequency of log levels
  .map(words => (words(0), 1)).
  .reduceByKey{ (a, b) => a + b, 2 }
```

input.txt

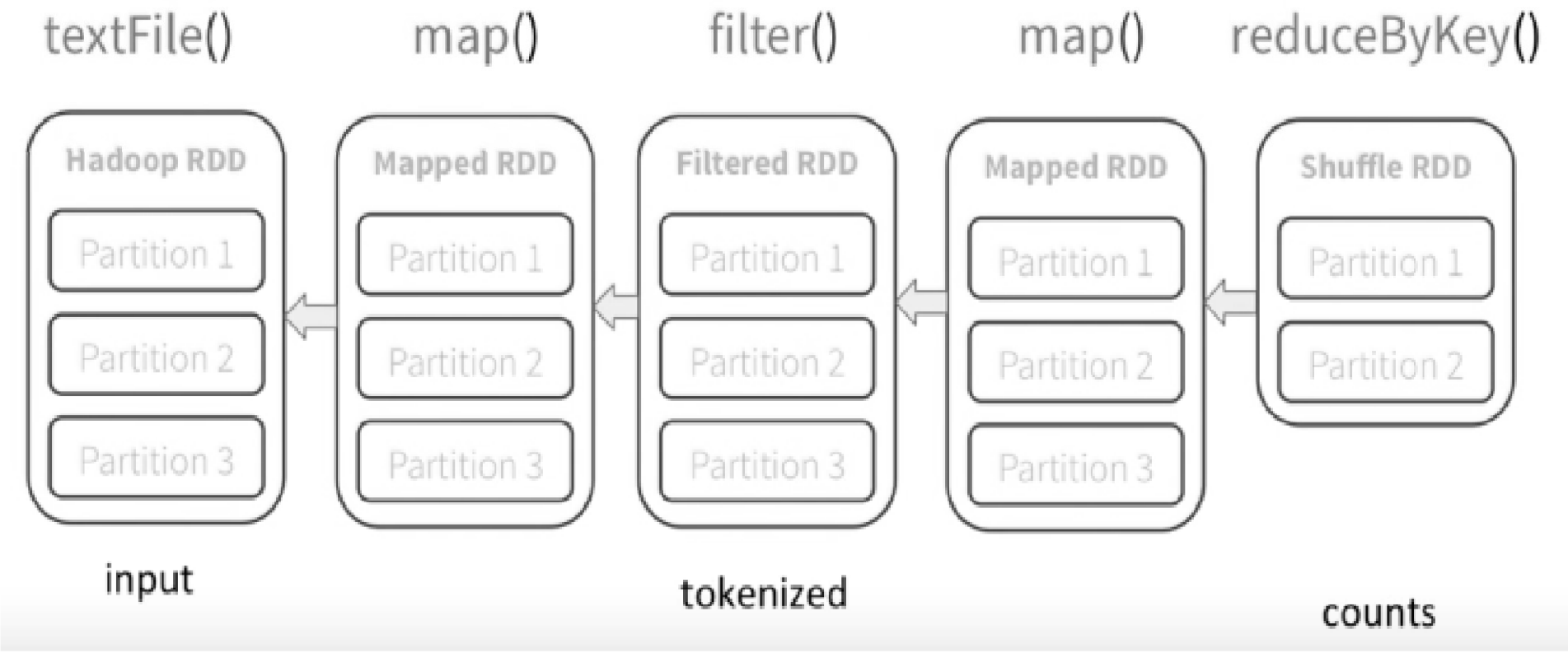
```
INFO Server started
INFO Bound to port 8080

WARN Cannot find srv.conf
```

RDD API Example

```
// Read input file
val input = sc.textFile(          )

val tokenized = input
  .map(                         )
  .filter(                        )

val counts = tokenized
  .map(                         ).
  .reduceByKey{                   }
```

View of the Transformations

.textFile().map().filter().map().reduceByKey()



Executing the

- When is the DAG actually evaluated?
- When we apply an action on an RDD

```
def runJob[T, U](
    rdd: RDD[T],                    // 1. RDD to compute
    partitions: Seq[Int],           // 2. Which partitions
    func: (Iterator[T]) => U))      // 3. Fn to produce results for partition
    : Array[U]                      // -> results for each partition
```

Example: Action

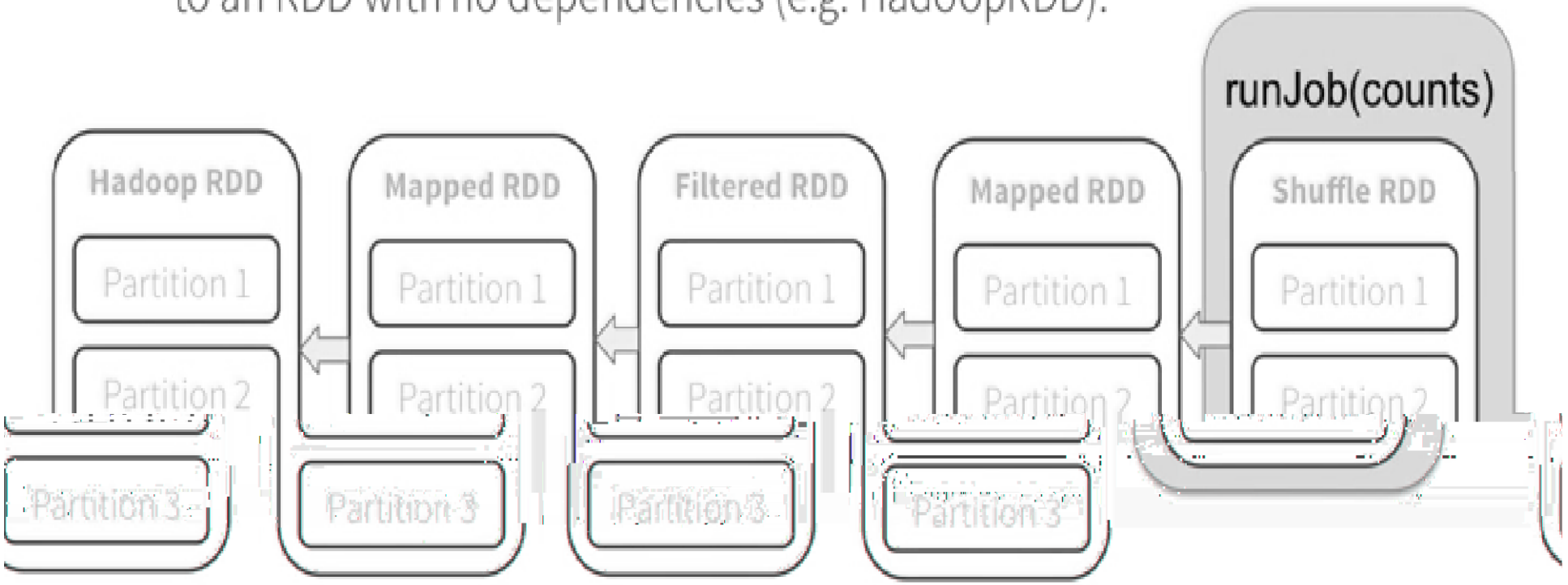
```
class RDD {
  def count(): Long = {
    results = sc.runJob(
      this,                          // RDD = self
      0 until partitions.size,       // Partitions = all
      it => it.size()                // Function = size of partition
  }

  return results.sum
}
```

Example: Action

```
class RDD[T] {
  def take(k: Int): Long = {
    val results = new ArrayBuffer[T]
    var partition = 0

    while (results.size < n) {
      results ++= sc.runJob(this, partition, it => it.toArray)
      partition += 1
    }

  return results.take(k)
}
```

Diving-in

Needs to compute my parents, parents, parents, etc all the way back to an RDD with no dependencies (e.g. HadoopRDD).



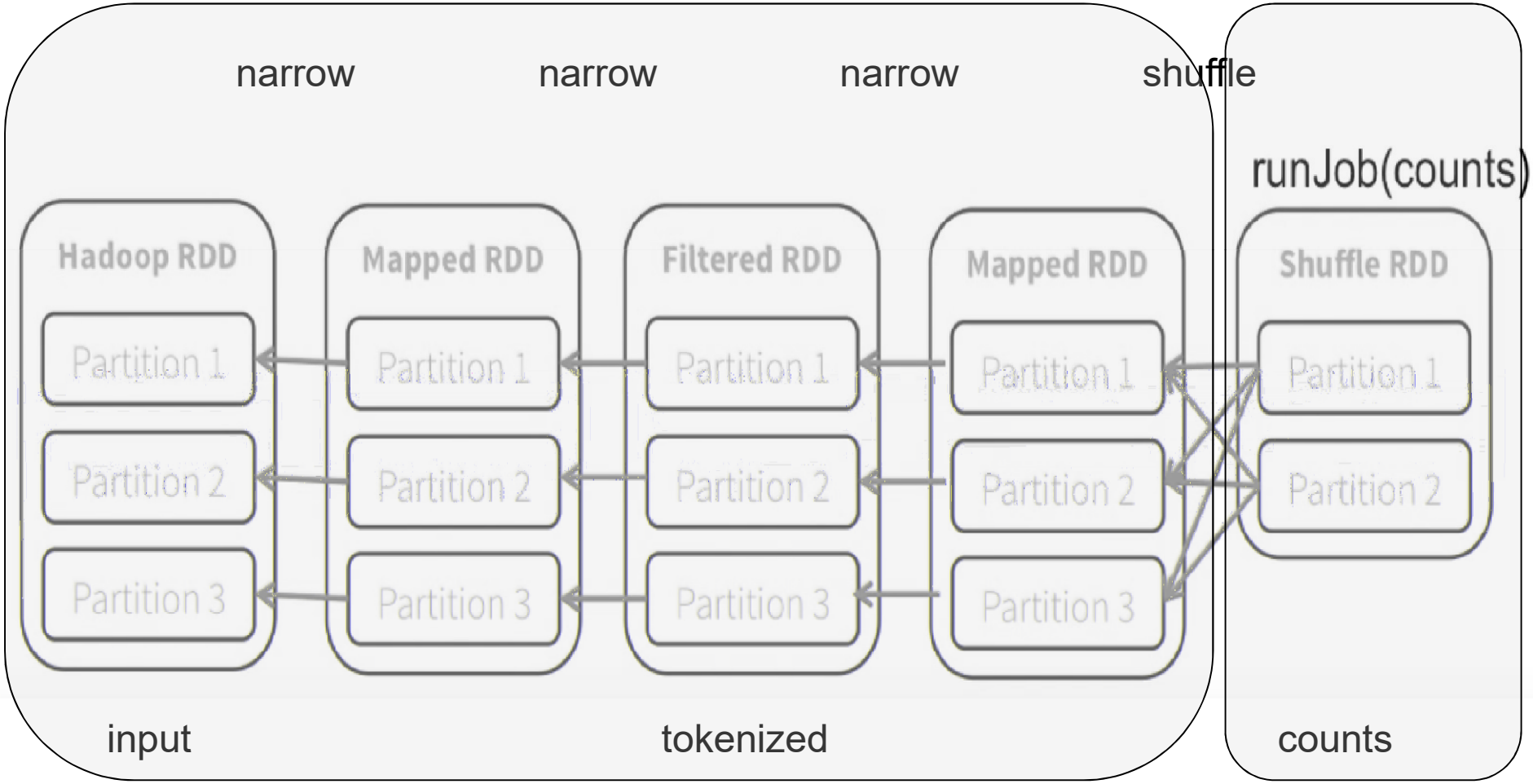
input tokenized counts

Laziness Allows Optimizations

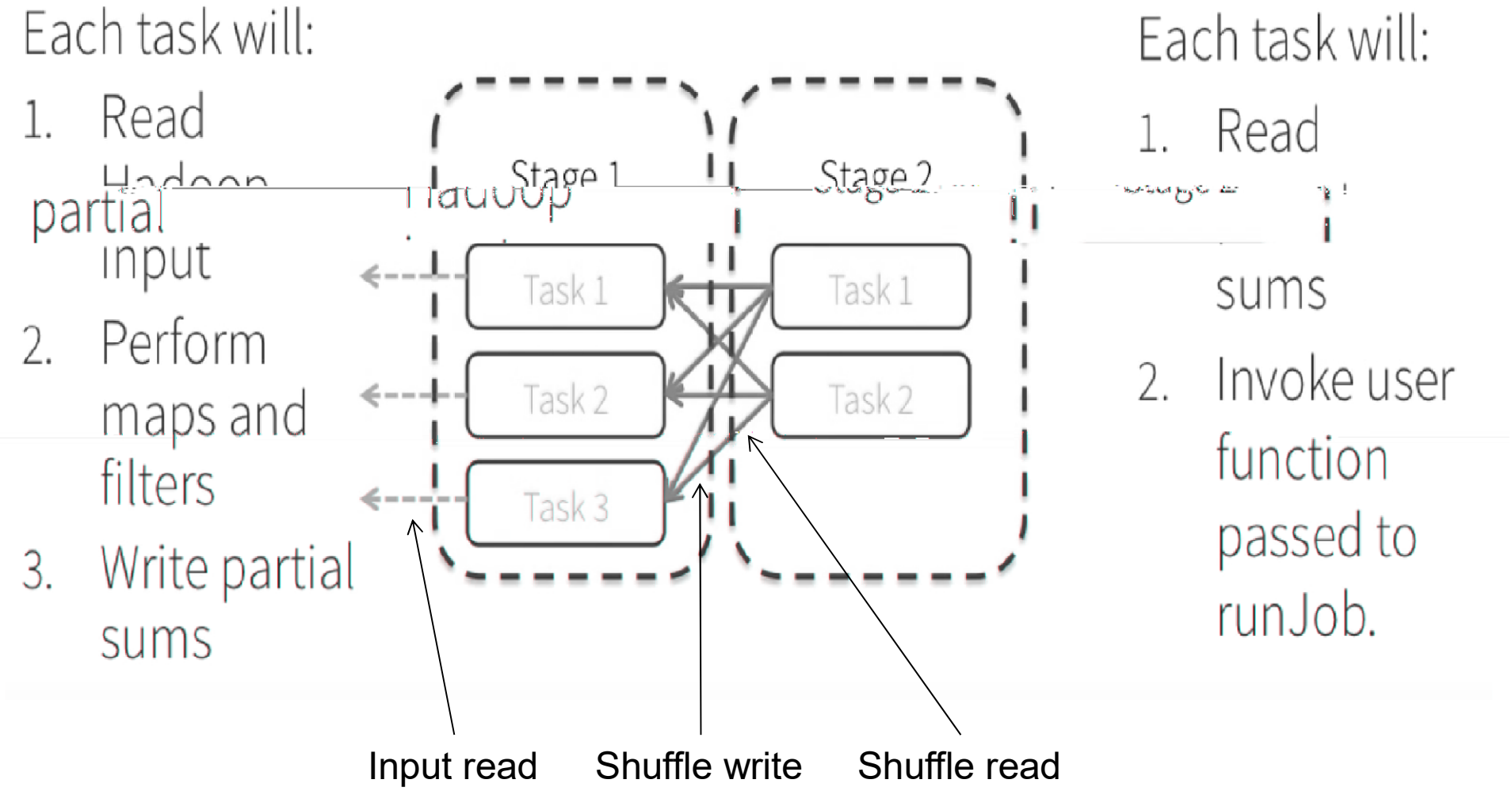
1. Certain types of transformations can be pipelined
2. If dependent RDDs have already been cached (or persisted in a shuffle) the graph can be truncated

Once (1) and (2) occur Spark produces
and each contains

Representing Dependencies



Physical Execution Plan



Units of Physical Execution

- work required to compute any runJob (usually a single action) on an RDD
- wave of work within job corresponding to pipelined RDDs
- unit of work within a stage corresponding to single partition in RDD
- transfer of data between stages

Writing Efficient Spark Code

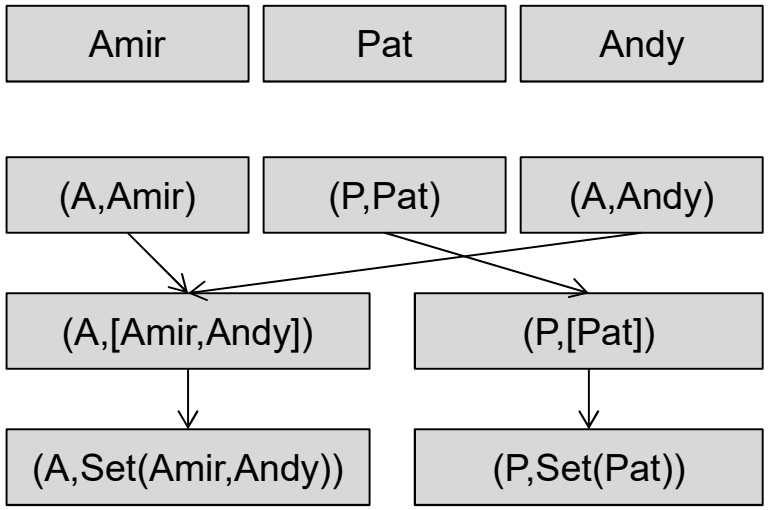
Our Task

- Find number of distinct words per first letter

```
 .textFile(“hdfs:///user/royl/names.txt”)
.map(name => (name.charAt(0), name))
.groupByKey
.mapValues(names => names.toSet.size)
.collect
```

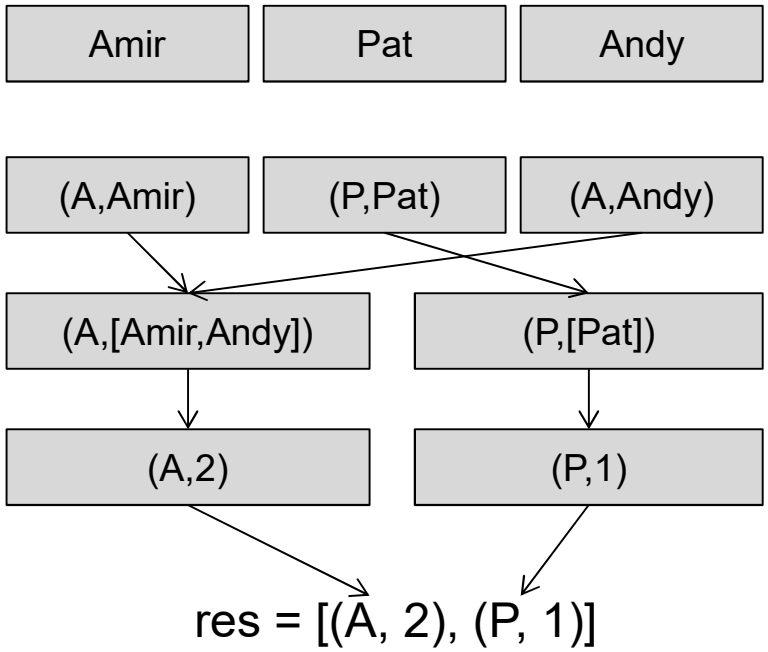
Our Task

```
.textFile(“hdfs:///user/royl/names.txt”)
 .map(name => (name.charAt(0), name))
 .groupByKey
 .mapValues(names => names.toSet.size)
 .collect
```



Our Task

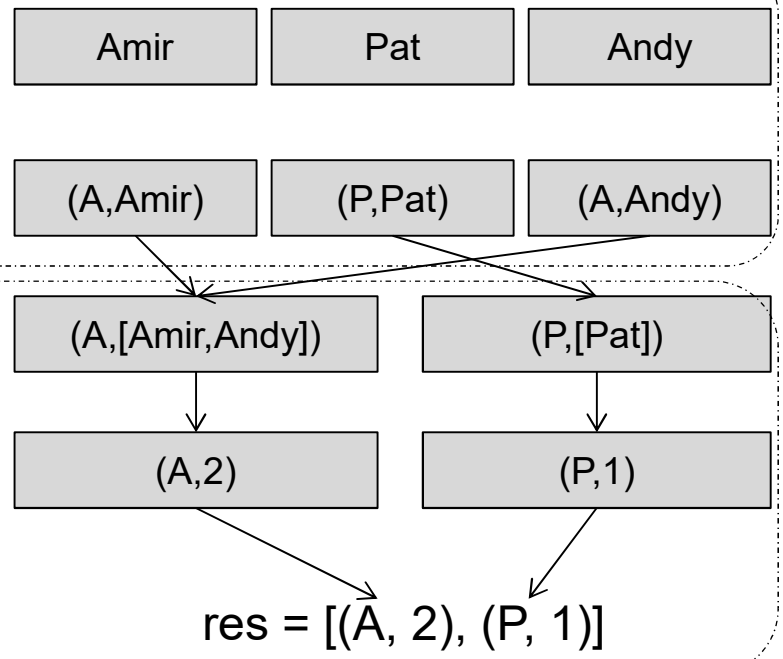
```
.textFile(“hdfs:///user/royl/names.txt”)
 .map(name => (name.charAt(0), name))
 .groupByKey
 .mapValues(names => names.toSet.size)
 .collect
```



Our Task

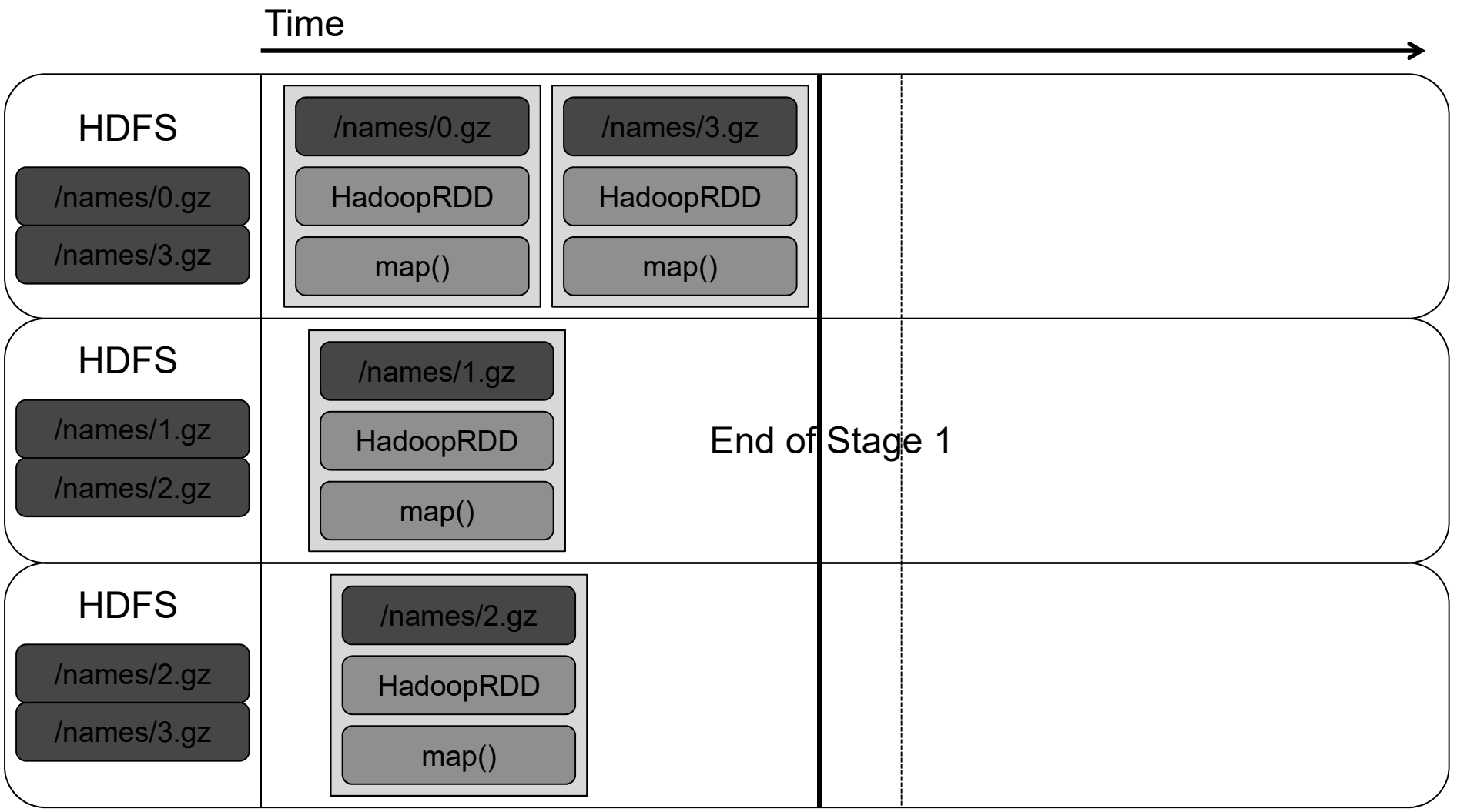
Stage 1

```
.textFile("hdfs:///user/royl/names.txt")

.map(name => (name.charAt(0), name))

.groupByKey

.mapValues(names => names.toSet.size)

.collect
```



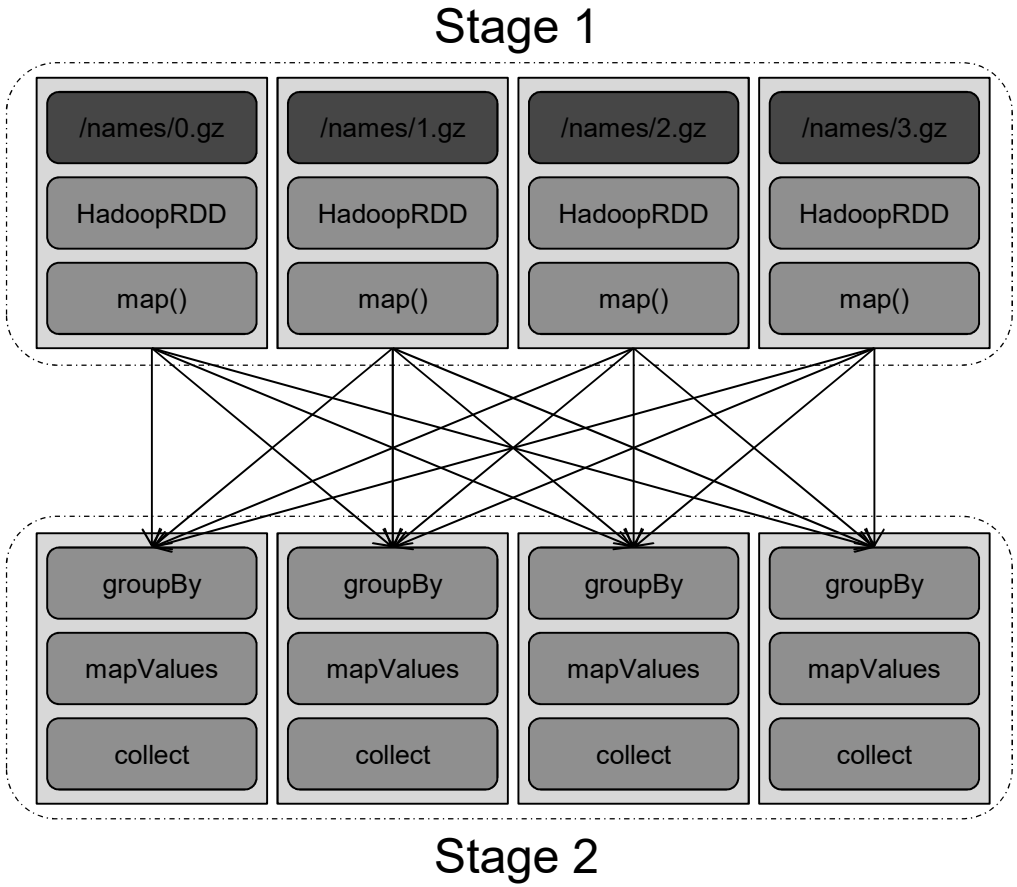
Stage 2

Scheduling the Tasks



The Shuffle

- Redistribute data among partitions
- Hash keys into buckets
- Optimizations:
  - Avoided when possible
  - Partial aggregation to reduce data movement



Executing the groupby

- Build hashmap with each partition
  - A=>[Arsalan,Aaron,Andrew,Andrew,Andrew,Ahir,Ali,…]
  - E=>[Erin,Eril,Ed,Erik,…]
- Note: can spill across keys but not a single key-value pair --- this must fit in memory!

So What's Wrong Here?

- Too few partitions
  - 3 machines and only 4 partitions
- Large per-key groupby
- Ship all data across the cluster

Fixing Our Example

```
 .textFile(“hdfs:///user/royl/names.txt”)
.map(name => (name.charAt(0), name))
.groupByKey
.mapValues(names => names.toSet.size)
.collect
```

Fixing Our Example

```
 .textFile(“hdfs:///user/royl/names.txt”)
.repartition(6)
.map(name => (name.charAt(0), name))
.groupByKey
.mapValues(names => names.toSet.size)
.collect
```

Fixing Our Example

```
 .textFile(“hdfs:///user/royl/names.txt”)
.repartition(6)
.distinct
.map(name => (name.charAt(0), name))
.groupByKey
.mapValues(names => names.size)
.collect
```

Fixing Our Example

```
 .textFile(“hdfs:///user/royl/names.txt”)
.distinct(6)
.map(name => (name.charAt(0), name))
.groupByKey
.mapValues(names => names.size)
.collect
```

Fixing Our Example

```
 .textFile(“hdfs:///user/royl/names.txt”)
.distinct(6)
.map(name => (name.charAt(0), 1))
.reduceByKey(_ + _)
.collect
```

References

- https://www.youtube.com/watch?v=N6pJhxCPe-Y
- https://www.youtube.com/watch?v=dmL0N3qfSc8
- http://ampcamp.berkeley.edu/amp-camp-one-berkeley-2012/
- https://www.youtube.com/watch?v=kkOG_aJ9KjQ