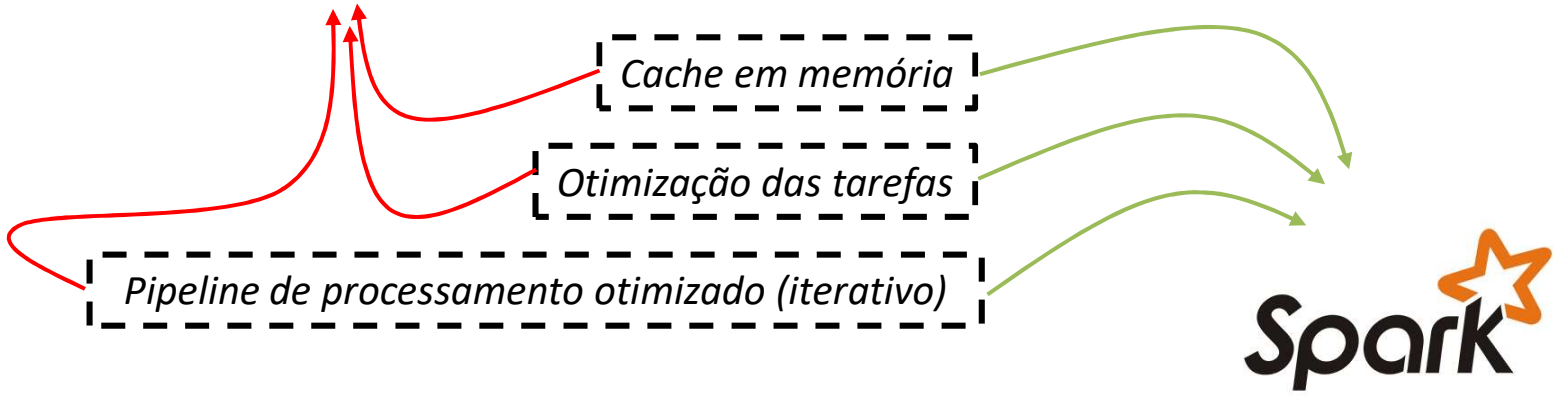
Apache Spark

- Maioria dos algoritmos executam processos iterativos
  - A cada rodada os resultados são melhorados
  - E.g. *Machine Learning*
- ***Qual o problema da abordagem baseada em MapReduce para análise iterativa dos dados?***

- ***Escrita/Leitura em disco é lenta!***

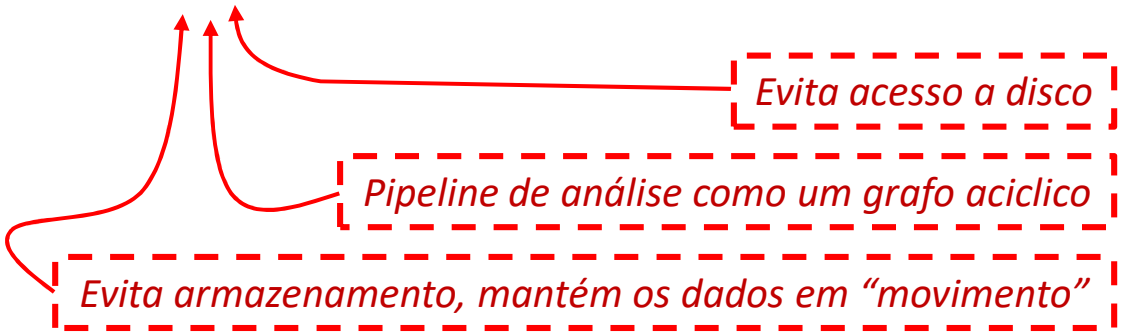
Apache Spark

- Dificuldades no MapReduce
  - Programação
  - Gargalos de performance, especialmente relacionados a disco
    - Em média cada job escreve em disco 3 vezes
  - Modelo orientado a lote
  - Dificuldade para execução de processos iterativos



Apache Spark

- Permite processamento rápido em cluster computacional de propósito geral



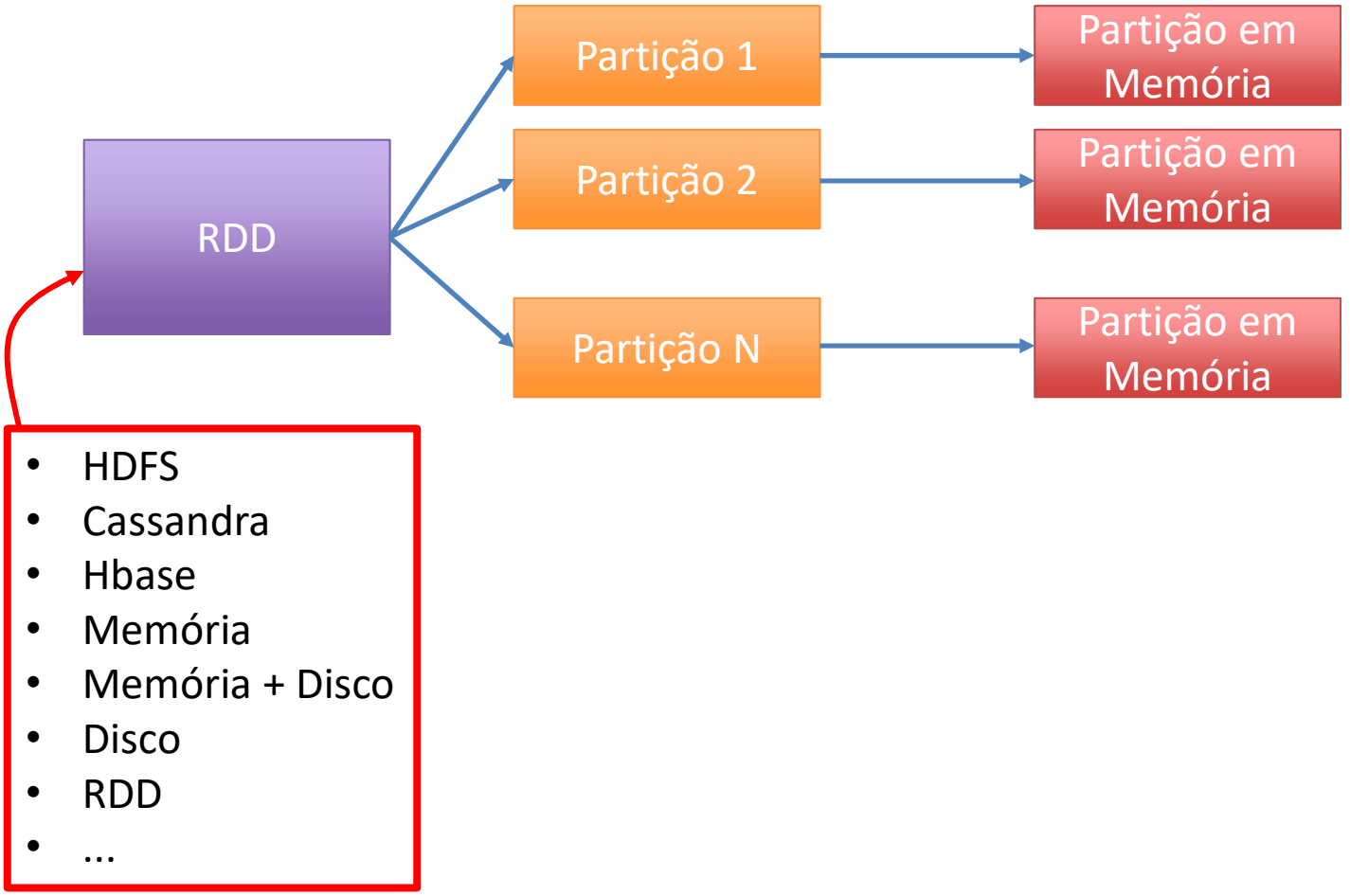
- Apache Spark
  - Integrado com ambiente do Hadoop
  - Permite leitura dos dados existentes no HDFS
  - APIs em Java, Scala, ***Python***

Apache Spark

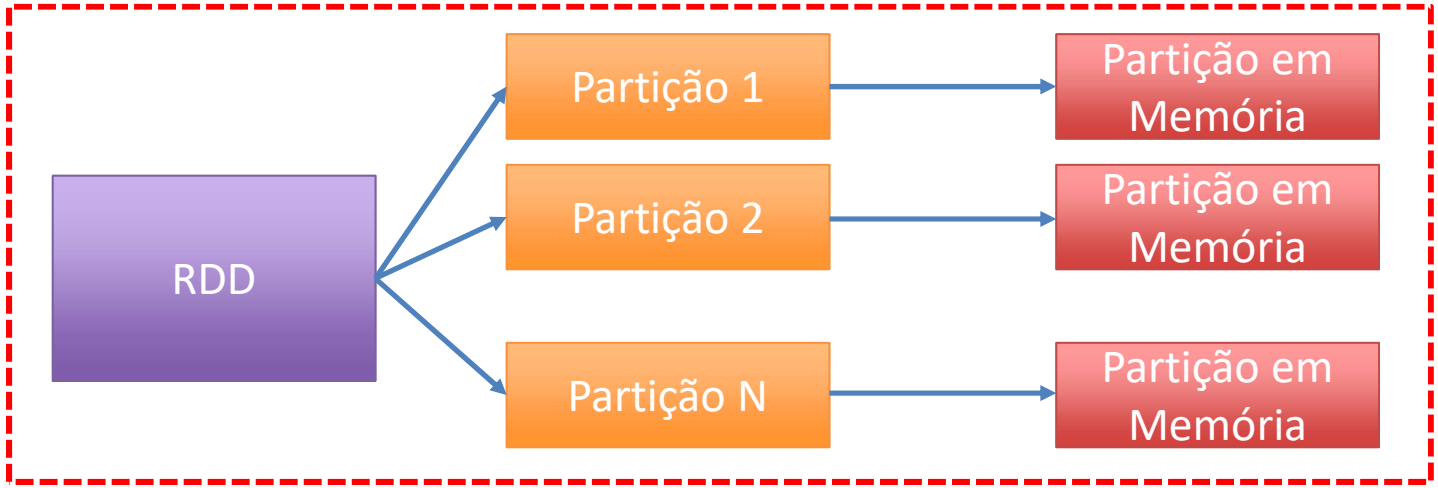
- Apache Spark opera sobre RDDs

  - Coleção de dados, divididos em partições, e armazenados nos *worker nodes* de um cluster
    - Em caso de falha, a partição é recalculada
  - Interface para **transformar** os dados
  - Abstração dos dados a serem analisados no Spark
    - HDFS
    - Cassandra
    - Hbase
    - Memória
    - Memória + Disco
    - Disco
    - RDD
    - ...

Apache Spark



Apache Spark



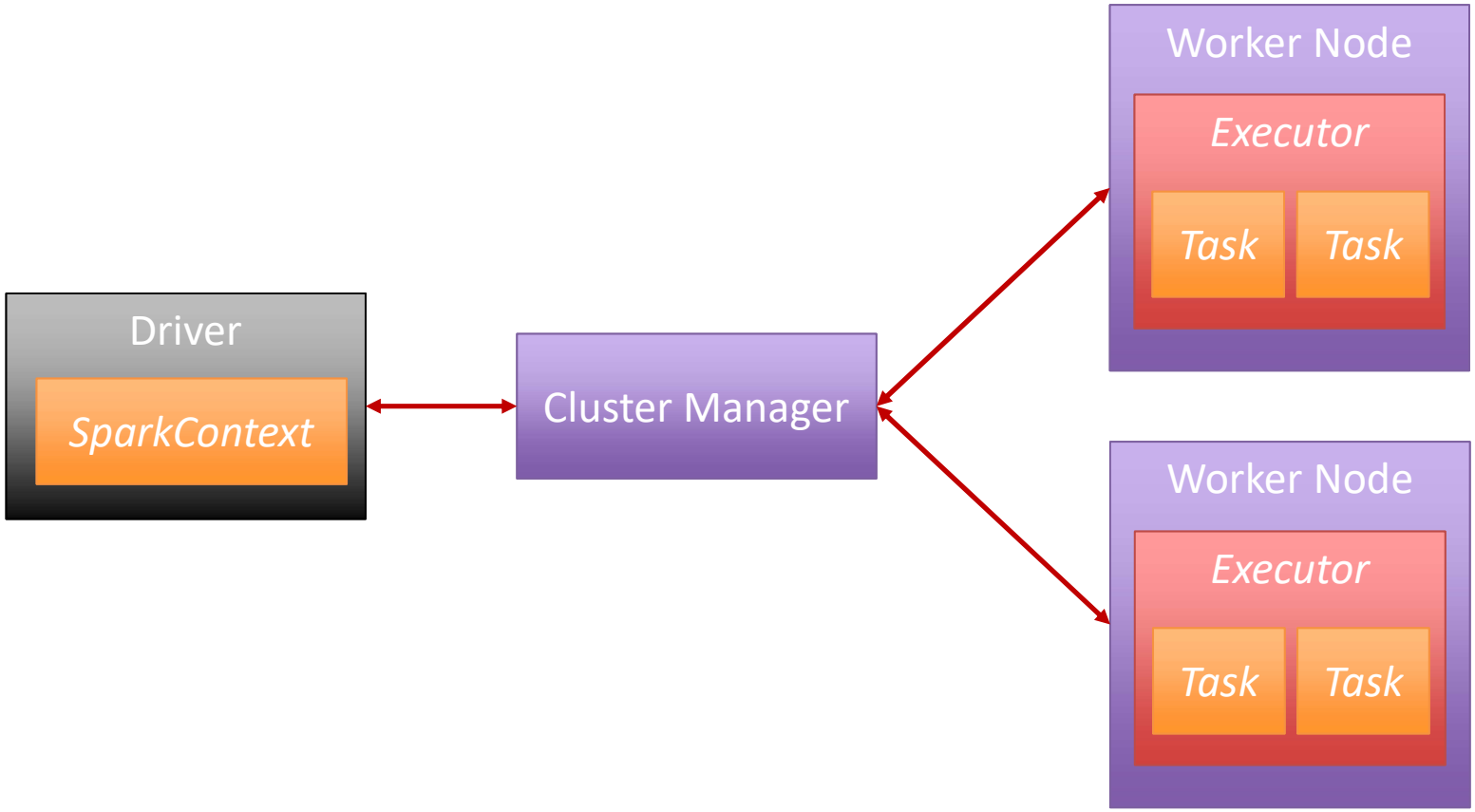
- Mantém referencia aos dados das partições
- Cada partição possui um subconjunto dos dados
- Partições são atribuídas aos nós do cluster
- Framework armazena a partição em “memória”

Apache Spark

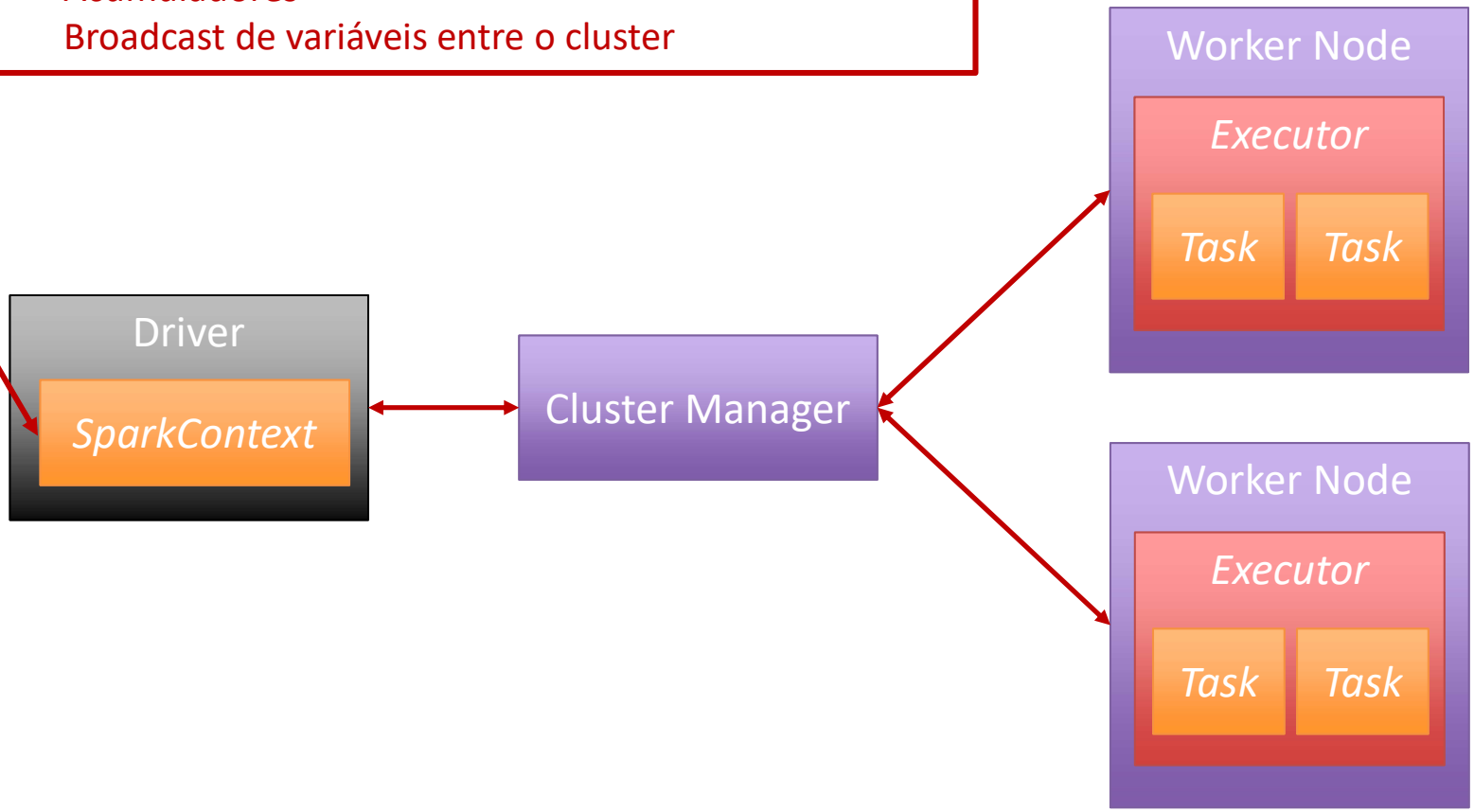
- Um RDD contém
  - Conjunto de dependências de RDDs “pais”
    - Linhagem do RDD (obtido através do DAG)
  - Conjunto de partições
    - Dados imutáveis
  - Função para calculo do RDD, baseado em seu RDD “pai”
  - Metadados sobre seu particionamento e localização dos dados

Modelo de processamento do Apache Spark, veremos isso em breve =)

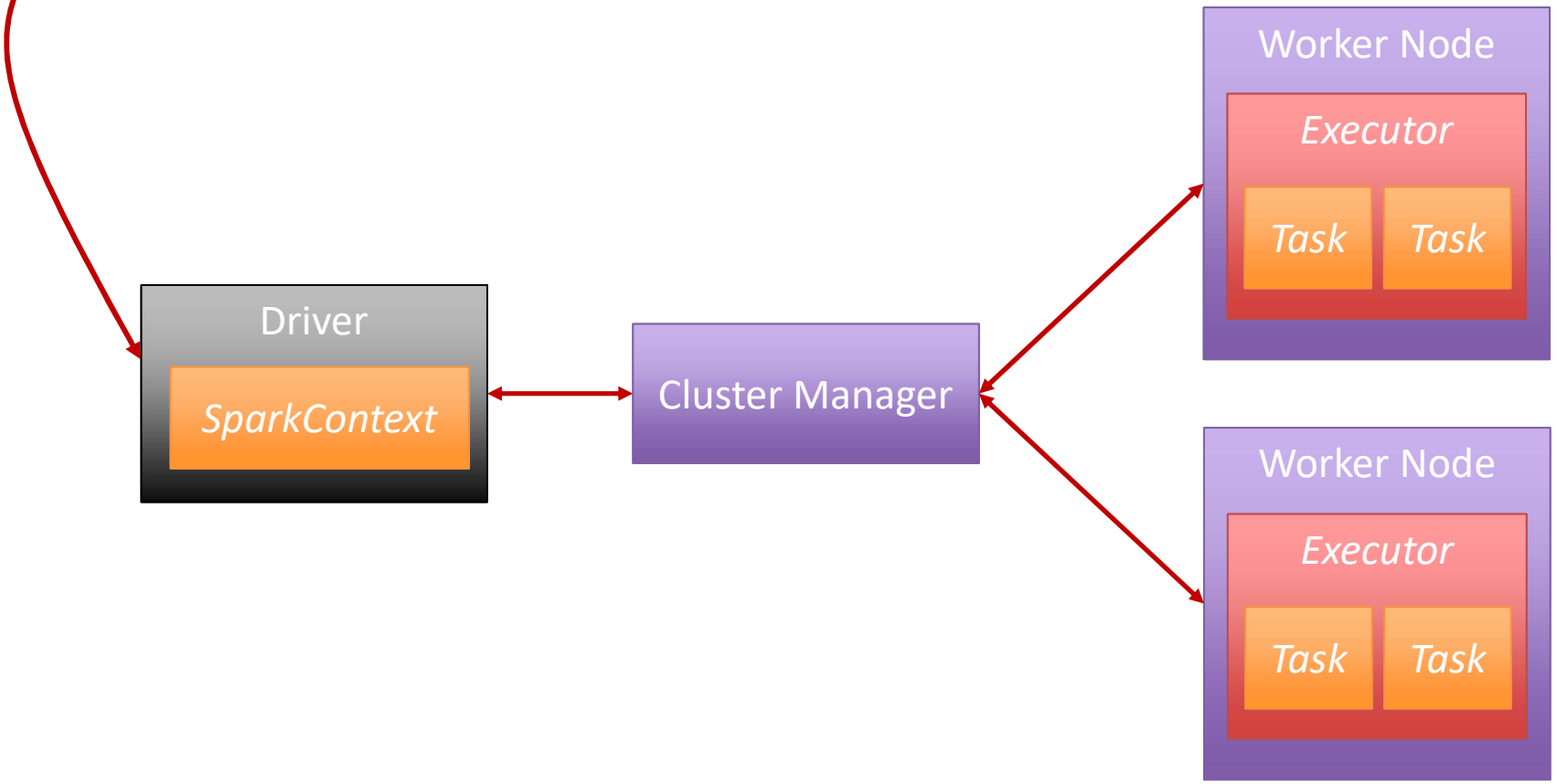
- RDD visa tratar os “problemas” de manipulação de dados
- Para tratar o “problema” da análise iterativa dos dados o Apache Spark utiliza o processamento em DAG
  - *Direct Acyclic Graph*


Worker Node
Executor
Task
Task
Driver
SparkContext
Cluster Manager
Worker Node
Executor
Task
Task

Ponto de entrada para o cluster Spark
Representa as conexões com o cluster
Determina como e quando acessar o cluster
Utilizado para
- Criar RDDs
- Acumuladores
- Broadcast de variáveis entre o cluster



Processo principal coordenado pelo objeto do SparkContext
Permite configurar um processo spark
Ações do Spark são executadas no driver
Aplicação -> Driver Program + Executores

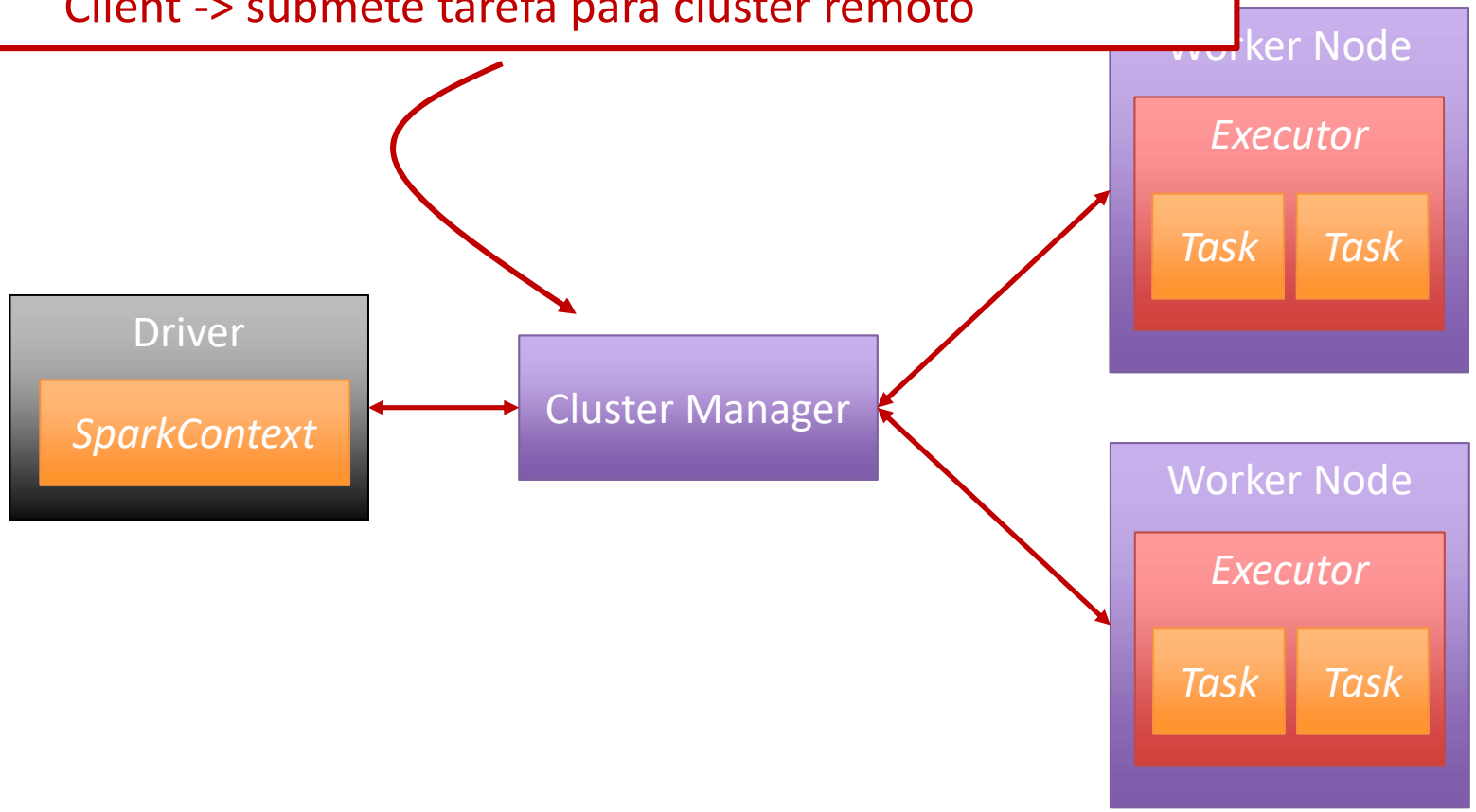


Serviço externo ao Apache Spark para receber recursos do cluster

- Local, StandAlone, YARN, Mesos, ...

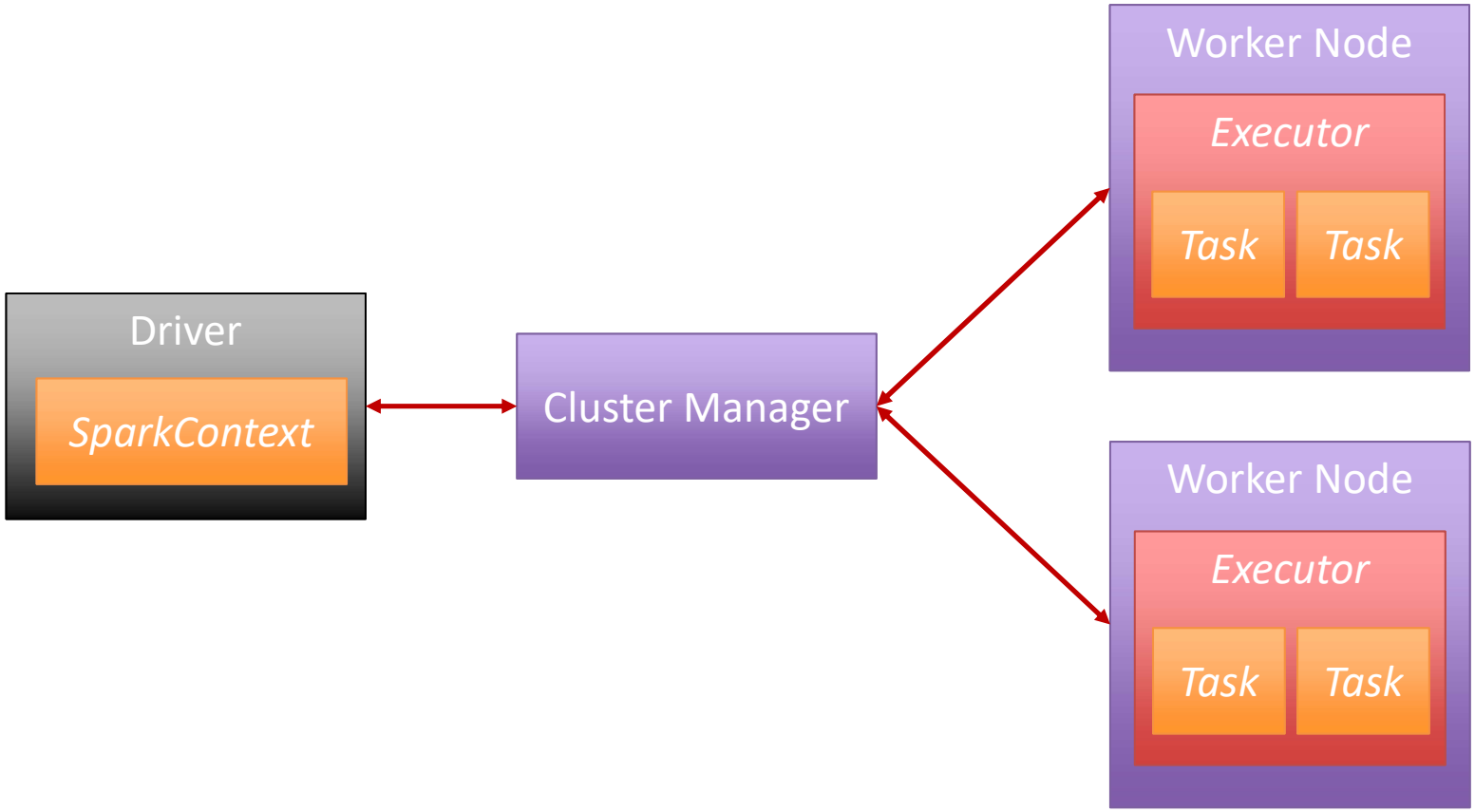
Modos de uso

- Cluster -> utiliza o cluster para envio da tarefa
- Client -> submete tarefa para cluster remoto

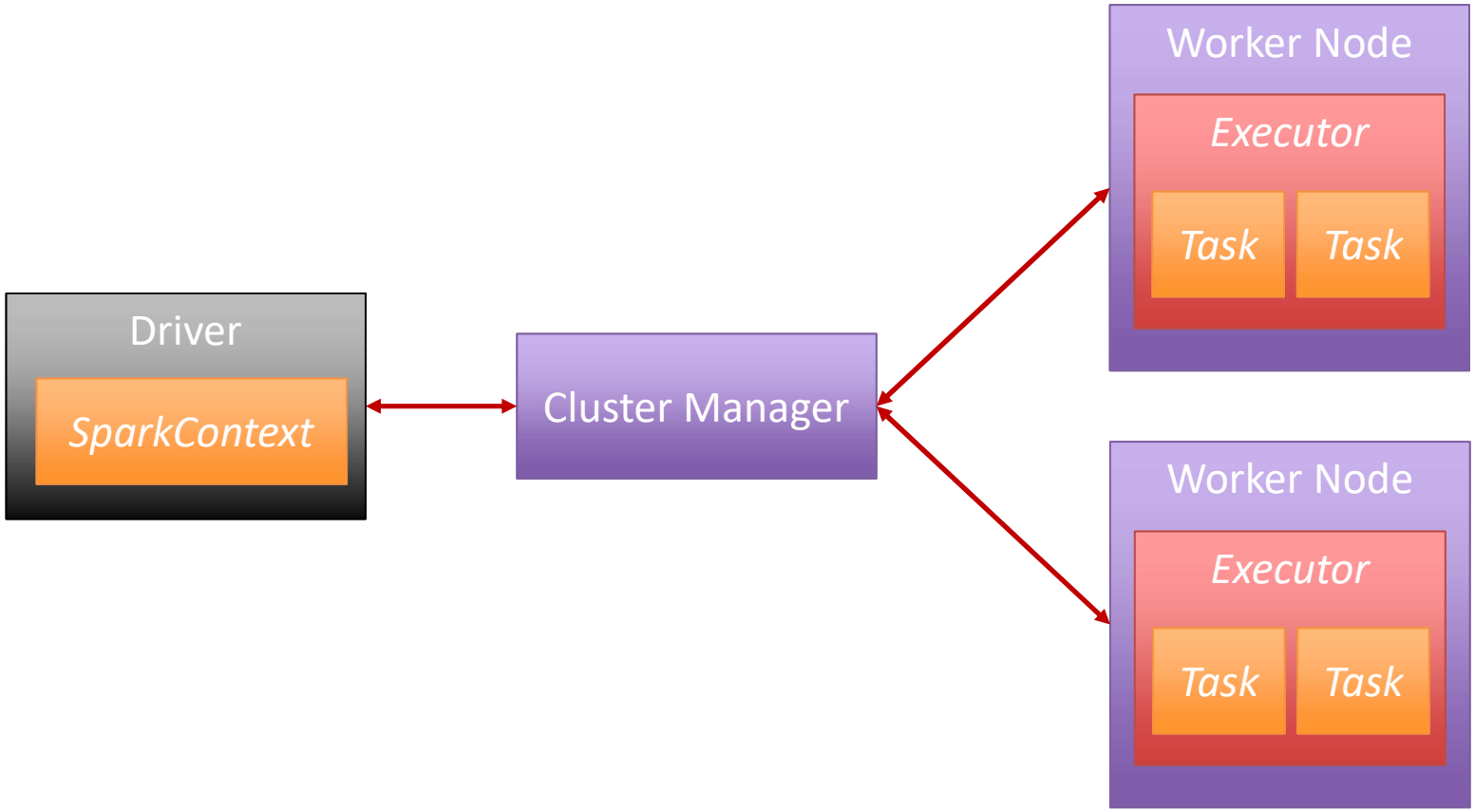


Executor, processo para execução da aplicação
Task, unidade de trabalho que será enviada ao executor
Job, tarefa executada no cluster

Nó responsável pela execução das tarefas

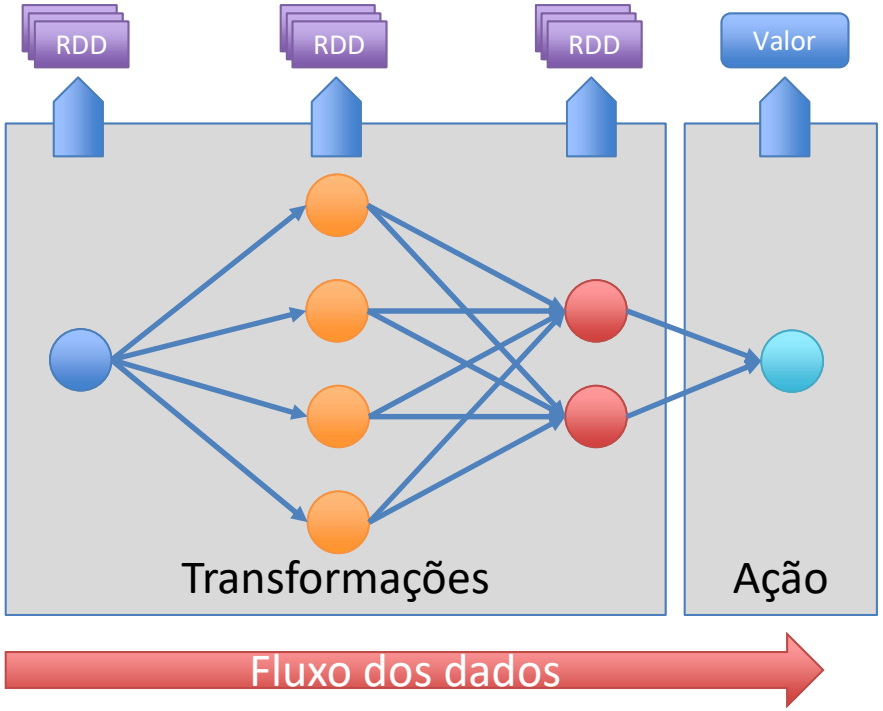


Apache Spark



Modelo de processamento

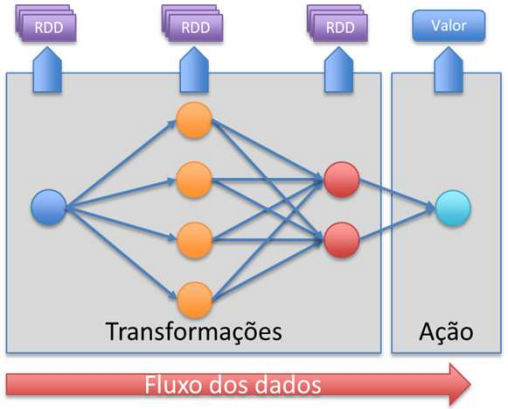
- Para tratar o "problema" da análise iterativa dos dados o Apache Spark utiliza o processamento em DAG
  - *Direct Acyclic Graph*



Modelo de processamento

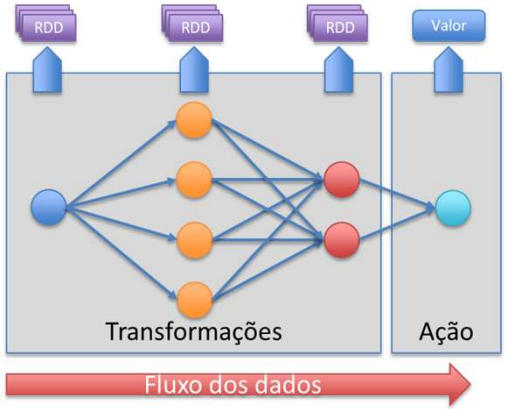
- Transformações
  - Cria um novo RDD a partir de um RDD existente
  - *Lazy evaluated*, a execução só ocorre quando efetuamos uma ação
  - *Exemplos:*
    - *MAP*
    - *FLATMAP*
    - *REDUCEBYKEY*
    - *FILTER*
    - *SAMPLE*
    - *DISTINCT*
    - *GROUPBYKEY*
    - *…*

Lista completa: https://spark.apache.org/docs/latest/rdd-programming-guide.html



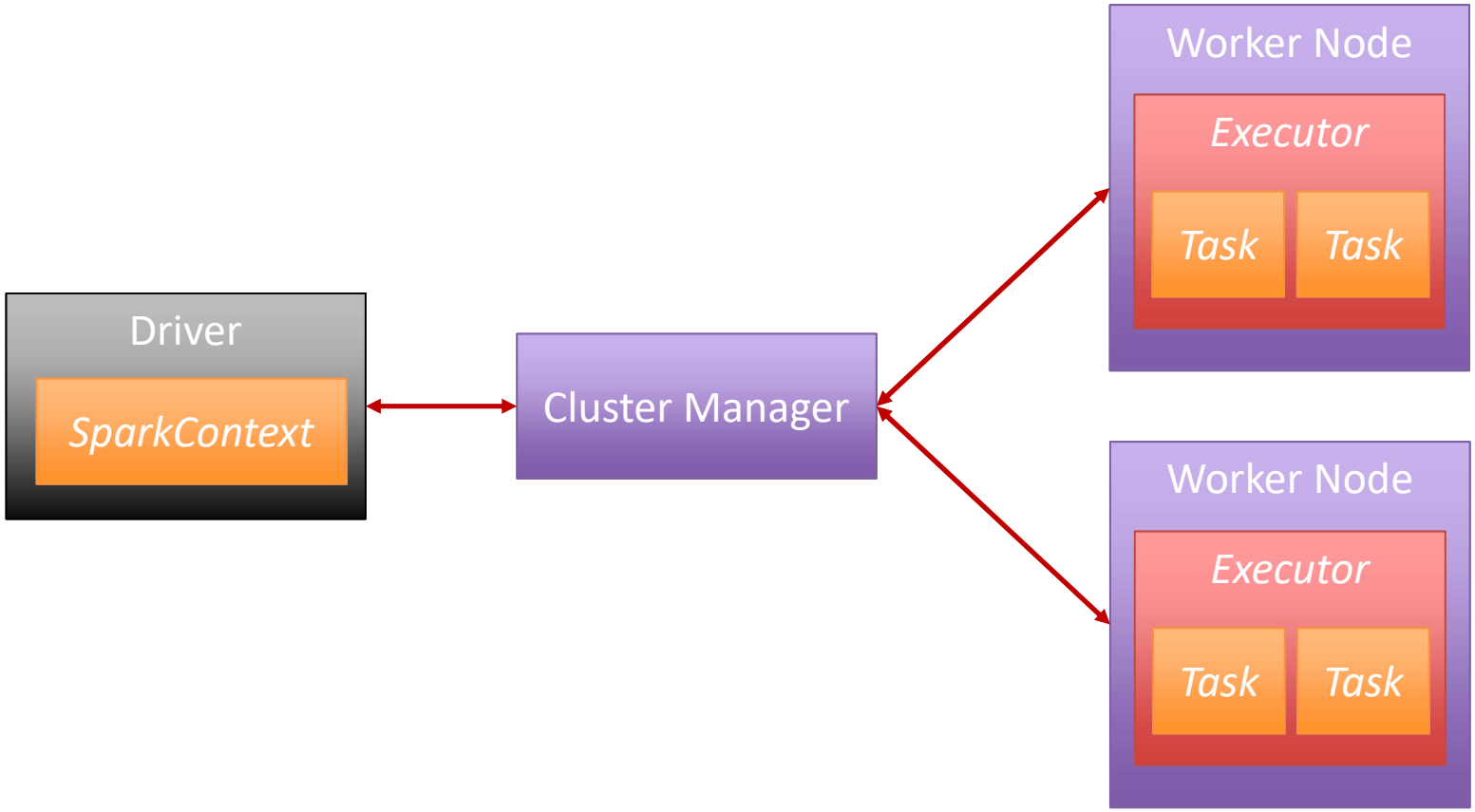
Modelo de processamento

- Ação
  - Retorna um valor para o **driver** depois do calculo sobre o RDD
  - *Exemplos:*
    - *COUNT*
    - *REDUCE*
    - *TAKE*
    - *COLLECT*
    - *FIRST*
    - *...*



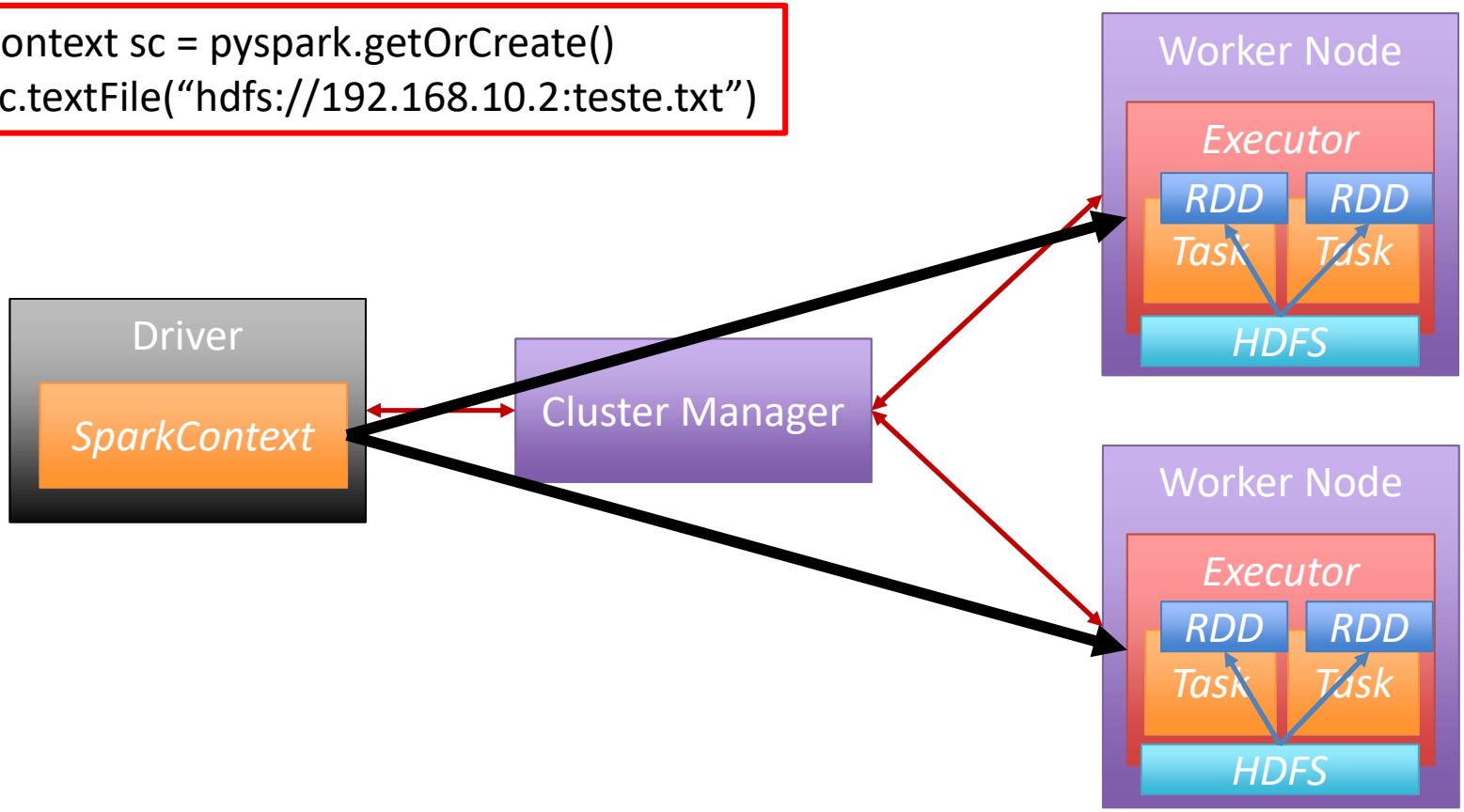
Lista completa: https://spark.apache.org/docs/latest/rdd-programming-guide.html

Exemplo: filtrando um dataset para contar entradas com um valor específico



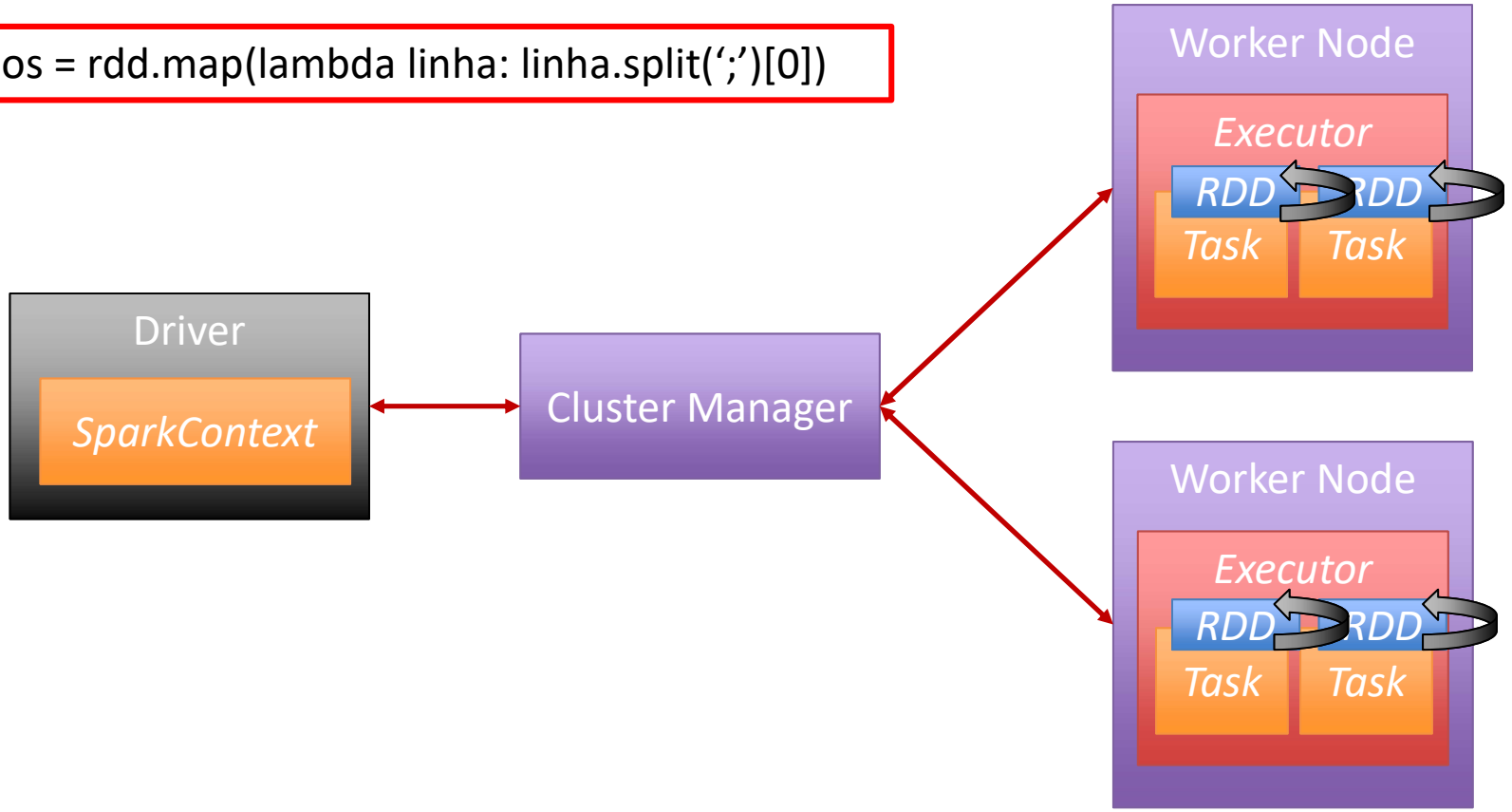
Exemplo: filtrando um dataset para contar entradas com um valor específico

```
SparkContext sc = pyspark.getOrCreate()
rdd = sc.textFile("hdfs://192.168.10.2:teste.txt")
```

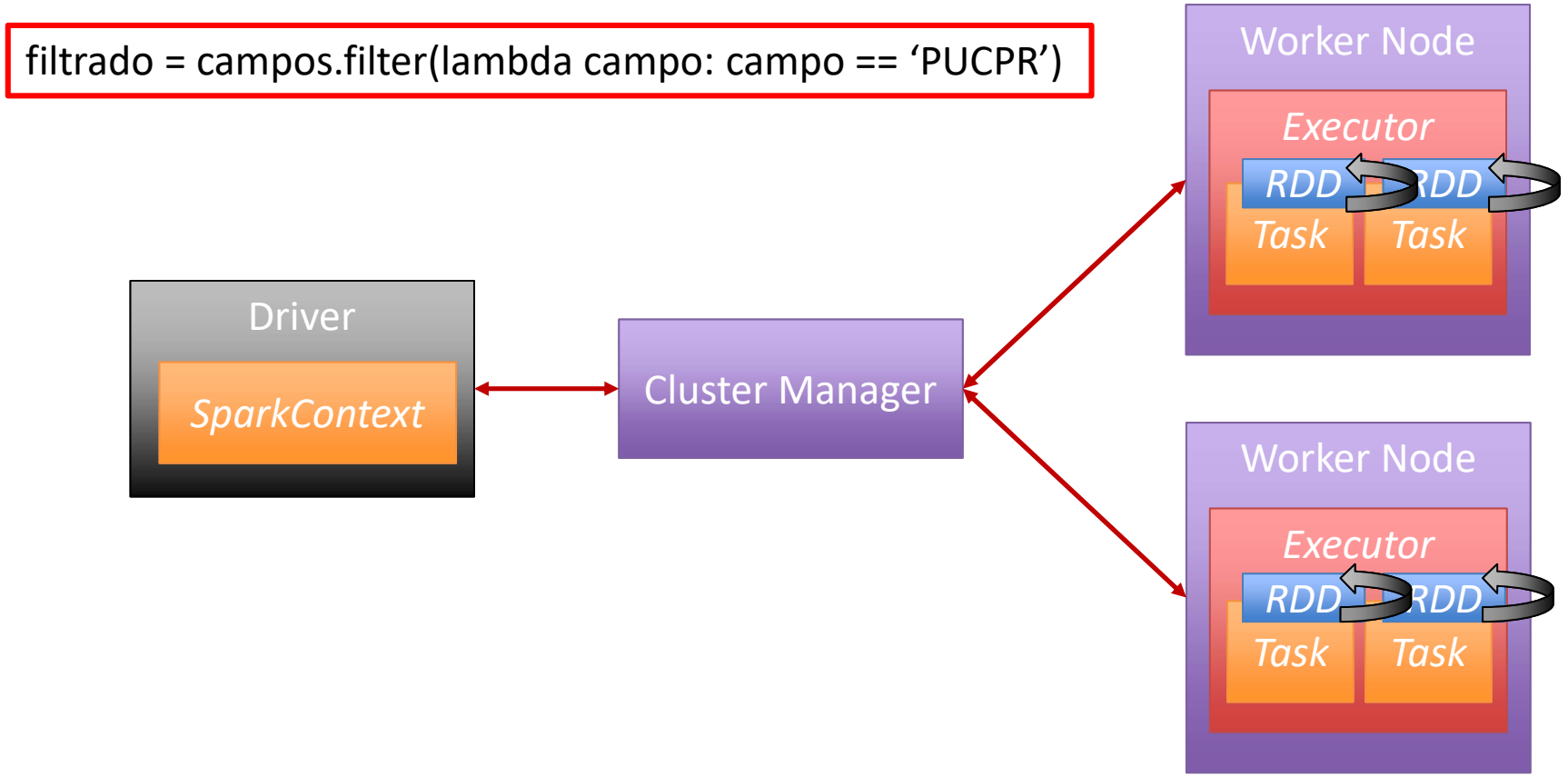


Exemplo: filtrando um dataset para contar entradas com um valor específico

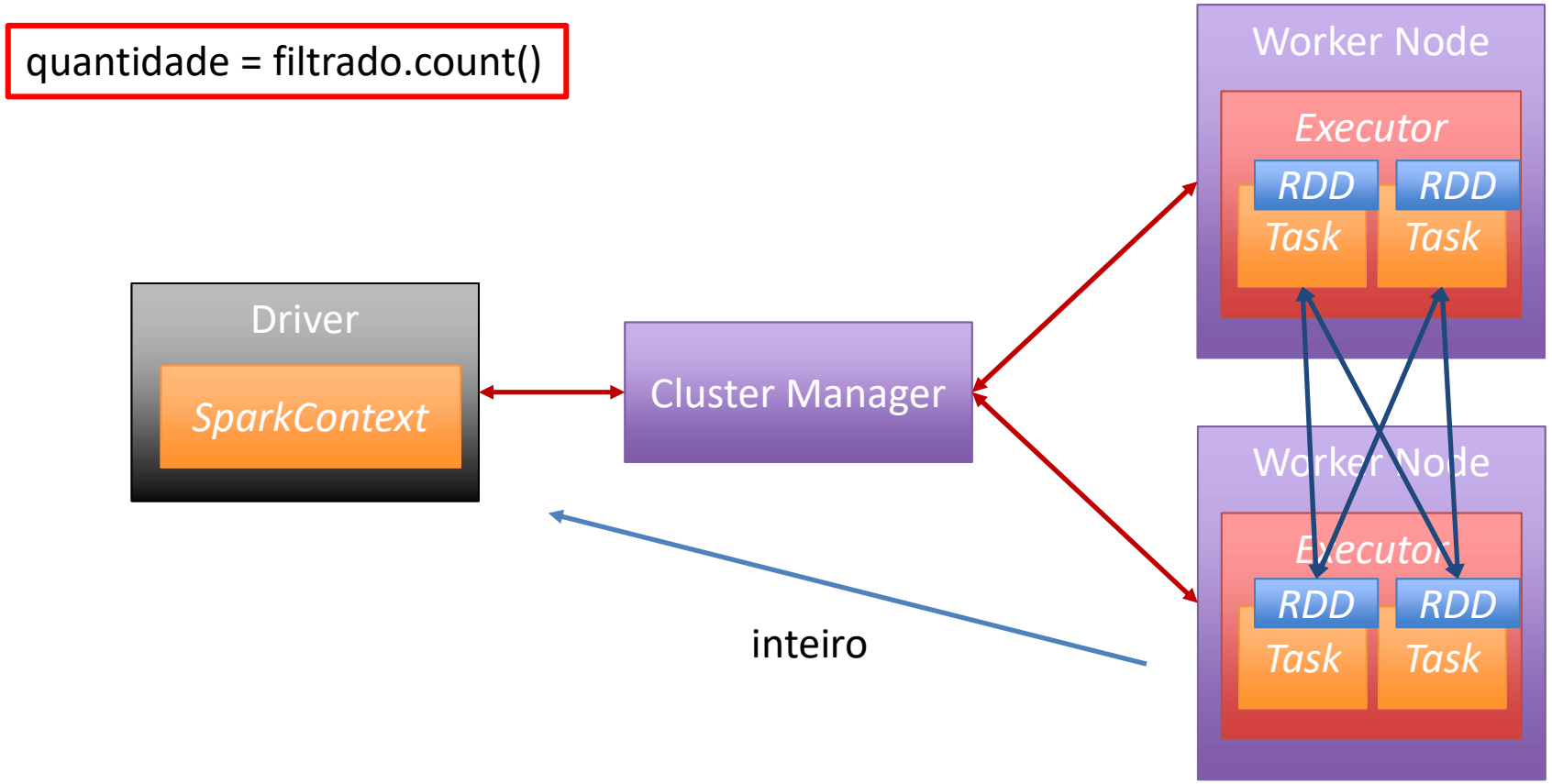
```
campos = rdd.map(lambda linha: linha.split(';')[0])
```



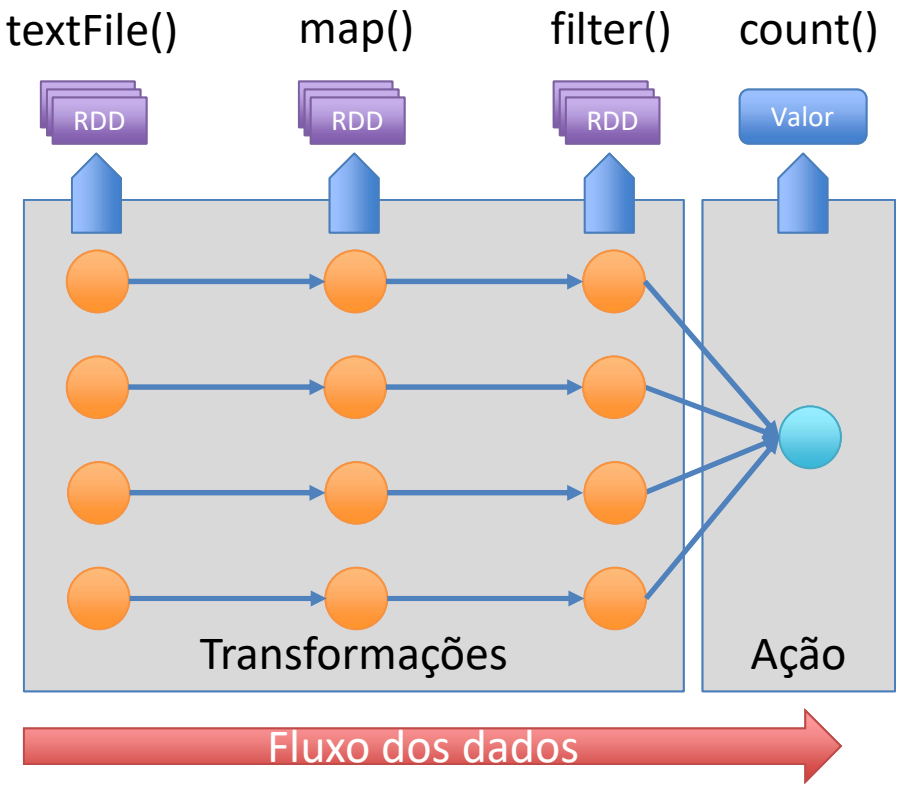
Exemplo: filtrando um dataset para contar entradas com um valor específico



Exemplo: filtrando um dataset para contar entradas com um valor específico



Exemplo: filtrando um dataset para contar entradas com um valor específico



Principais transformações

Transformação	Descrição	Código
map	Gera RDD, executando a FUNC sobre todos elementos do RDD	campos = arquivo.map(lambda x: x.split(";")[0])
filter	Gera RDD com apenas os elementos que retornaram True através da FUNC	campos.filter(lambda x: x == "Brazil")

flatmap	Gera RDD, executando a FUNC sobre todos elementos do RDD, pode retornar 0 ou mais elementos	def mapFunc(x):
		  y = x.split(";")
		  return y
		arquivo.map(mapFunc)

sample	Gera um RDD com uma fração do RDD de entrada	arquivo.sample(withReplacement = True, fraction = 0.1, seed = 1)
distinct	Gera um RDD com elementos distintos	arquivo.distinct()
groupBy	Gera um RDD com elementos agrupados pela FUNC, no formato (chave, (valores))	arquivo.groupBy(lambda x: x.split(";")[0])
reduceByKey	Gera um RDD com o formato (chave, valor), quando chamado sobre um RDD no formato (chave, (valores))	arquivo.map(lambda x: [x.split(";")[0], 1]).reduceByKey(lambda x,y: x+y).collect()
...		

Lista completa: https://spark.apache.org/docs/latest/rdd-programming-guide.html
https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

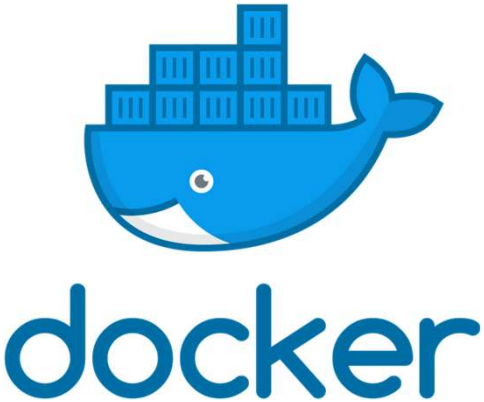
Principais ações

Ação	Descrição	Código
reduce	Agrega todos elementos do RDD através da FUNC, gerando um único valor	arquivo.map(lambda x: 1).reduce(lambda x,y: x+y)
collect	Retorna todos elementos para o driver	arquivo.collect()
count	Retorna um inteiro com a quantidade de elementos no RDD	arquivo.count()
first	Retorna o primeiro elemento do RDD	arquivo.first()
take	Retorna N elementos do RDD	arquivo.take(10)
saveAsText	Salva os elementos do RDD no arquivo especificado	arquivo.saveAsTextFile("hdfs://172.18.0.11:9000/base2")
...		

Lista completa: https://spark.apache.org/docs/latest/rdd-programming-guide.html
https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

Prática

- Liguem a máquina virtual
  - Usuário: docker
  - Senha: docker



- Vamos ligar o nosso “cluster” hadoop agora

Apache Spark - Implementação

- *Vamos acessar o jupyter com o pyspark para implementação das soluções*

Atualize o IP para o spark-master =)

SparkContext, “Driver”

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Atualize o IP para o namenode =)

```
arquivo = sc.textFile(“hdfs://172.24.0.12:9000/base.csv”)
```

Lista completa: https://spark.apache.org/docs/latest/rdd-programming-guide.html
https://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD