
Data Science Academy
Data Science Academy marxv49@gmail.com 5e6362be32fc3445a0e403b
Big Data Real-Time Analytics com Python e Spark
28%
36%
40%
35%
38%
36%
35%
32%
28%
31%
30%
March
April
May
June
July August
September
October
November
December
January
February
March
April
May
November
December
October
September
July August

Big Data Real-Time Analytics com Python e Spark

Seja muito bem-vindo(a)!

Processando Big Data com Apache Spark

Big Data Real-Time Analytics com Python e Spark

Por que Aprender Apache Spark?

Como Vamos Estudar o Spark?

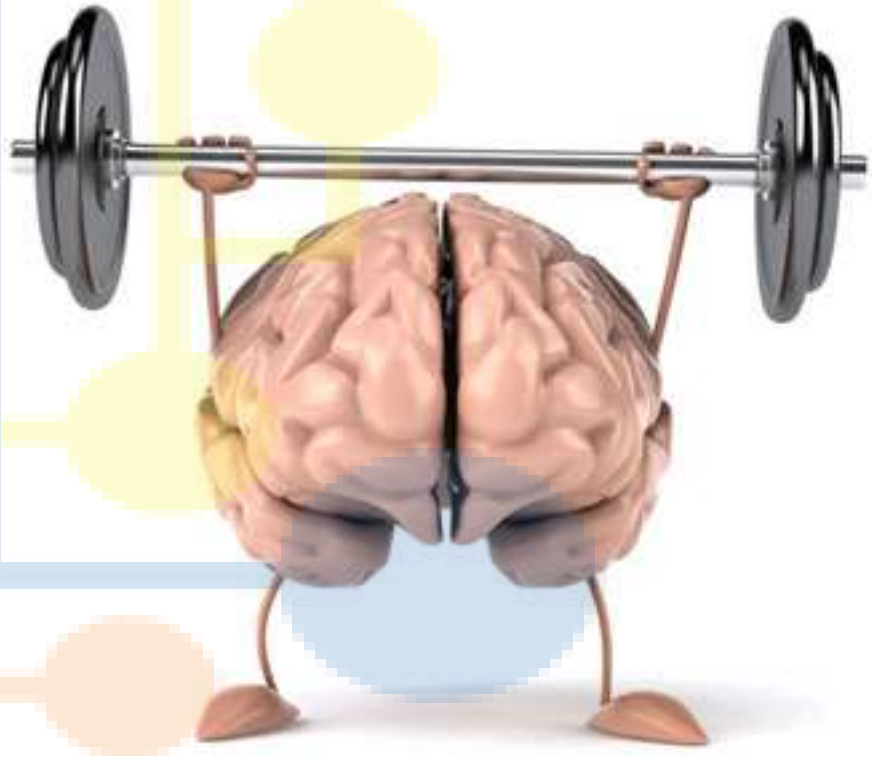
Big Data Real-Time Analytics com Python e Spark

Como Vamos Estudar o Spark?

- Capítulo 7 - Arquitetura do Spark, Transformações, Ações, PySpark
- Capítulo 8 - Spark SQL
- Capítulo 9 - Spark Streaming e Análise de Dados em Tempo Real
- Capítulo 10 - Machine Learning em Streaming de Dados com Spark MLlib

Big Data Real-Time Analytics com Python e Spark

O Spark é um tema avançado!

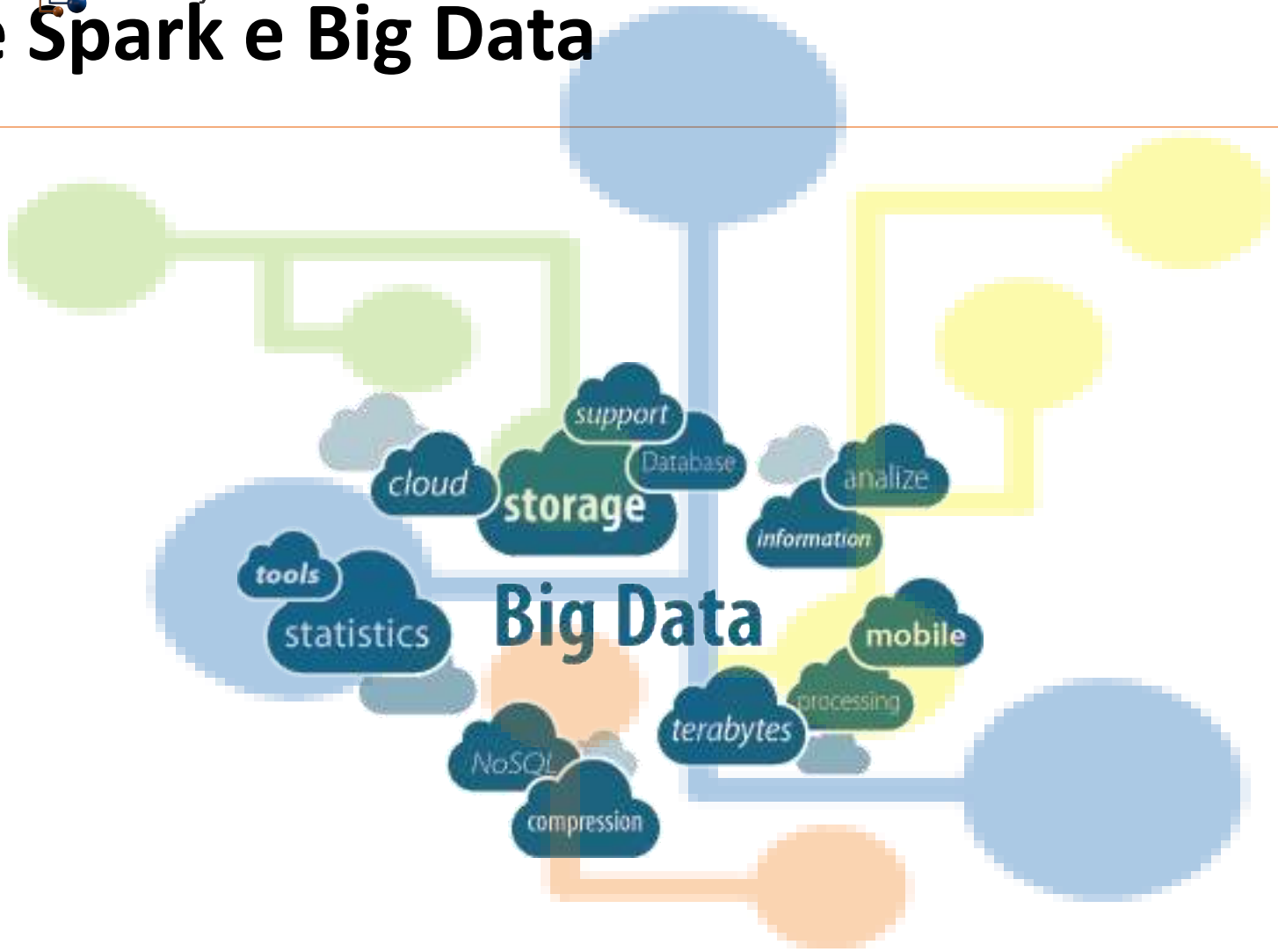


Big Data Real-Time Analytics com Python e Spark

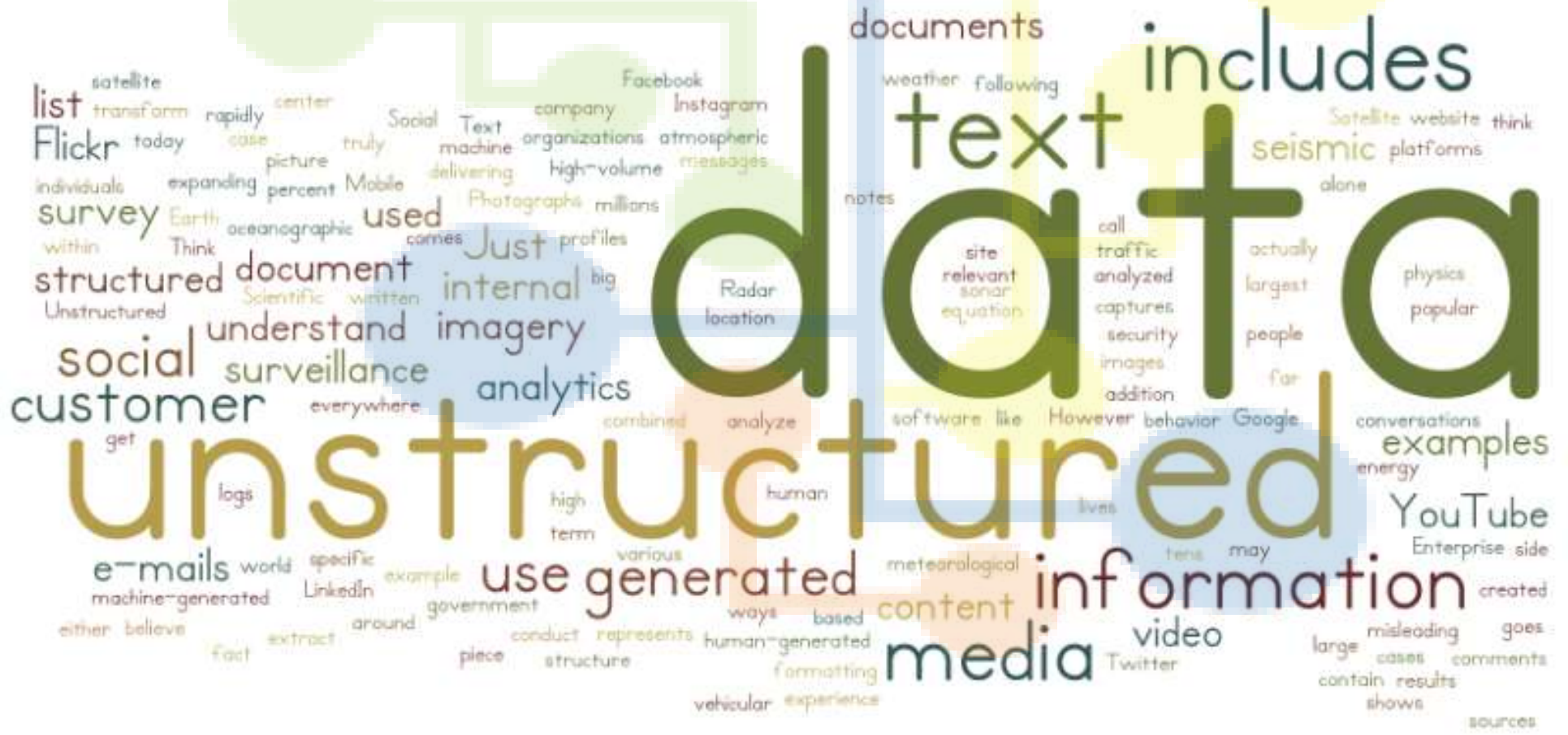
A dedicação é o combustível que nos move.
Ela é a responsável pelo nosso sucesso.

Apache Spark e Big Data

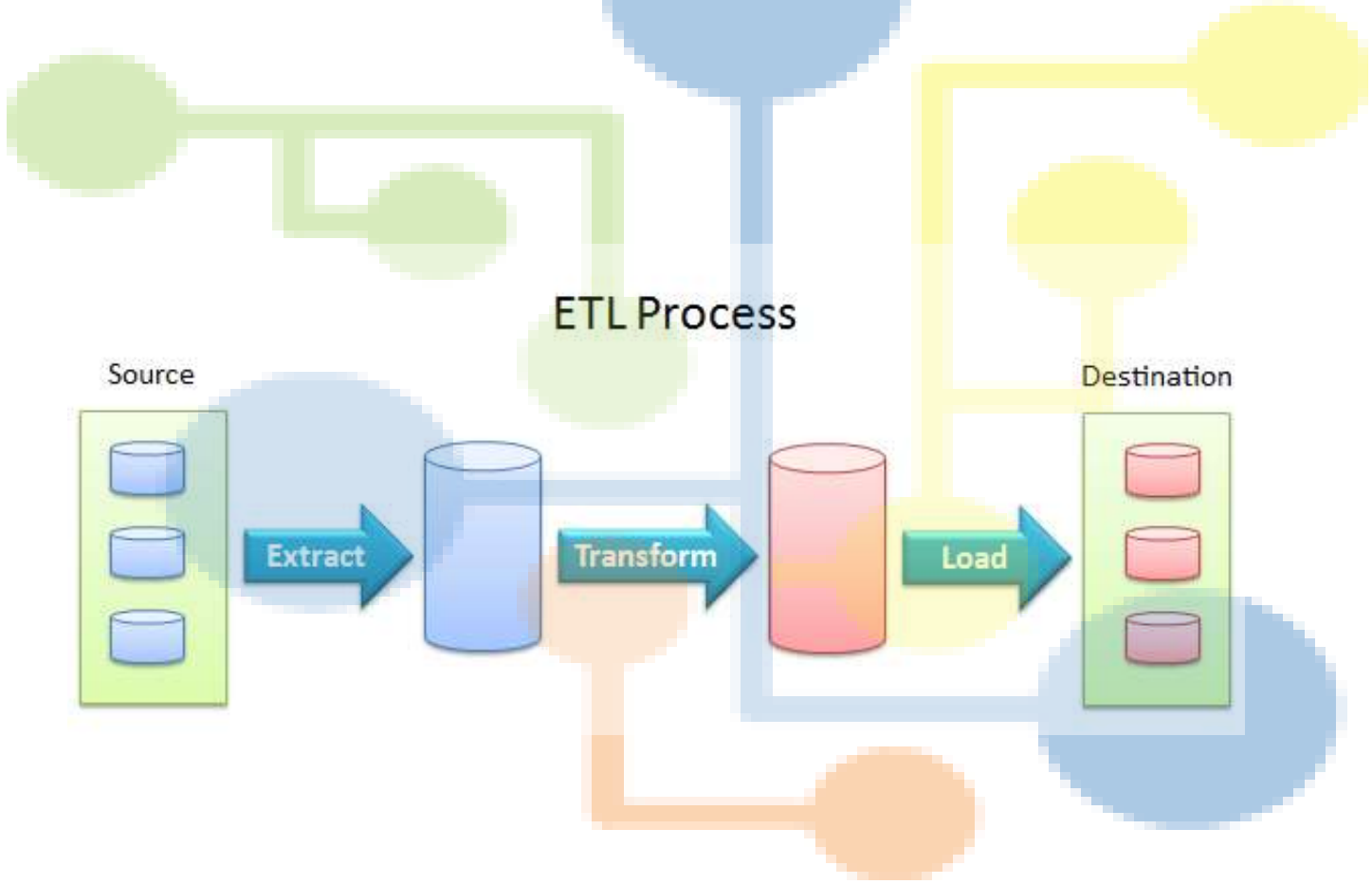
Apache Spark e Big Data



Apache Spark e Big Data



Apache Spark e Big Data

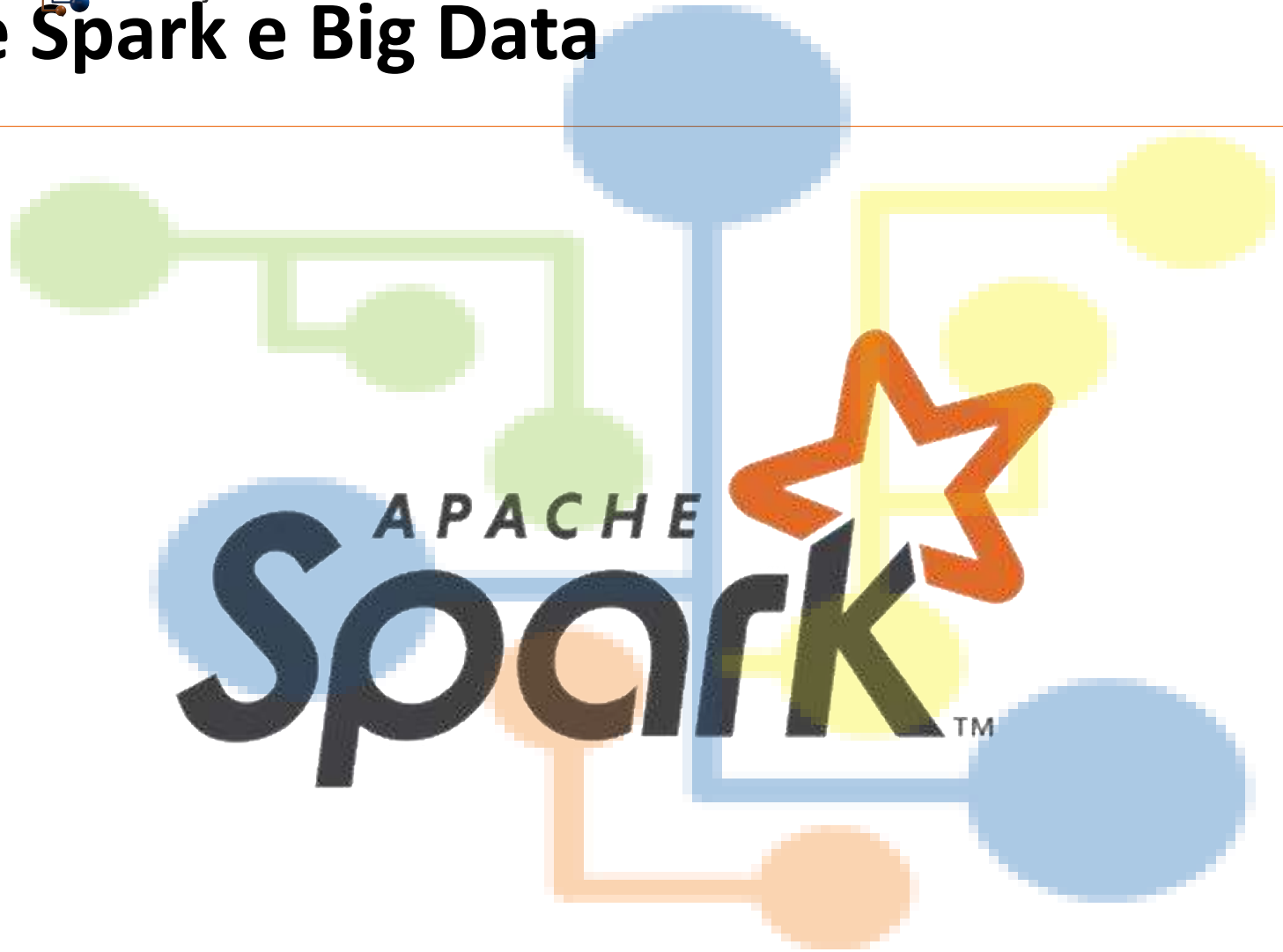


Como armazenar e processar todos esses dados, se o volume aumenta de forma exponencial?

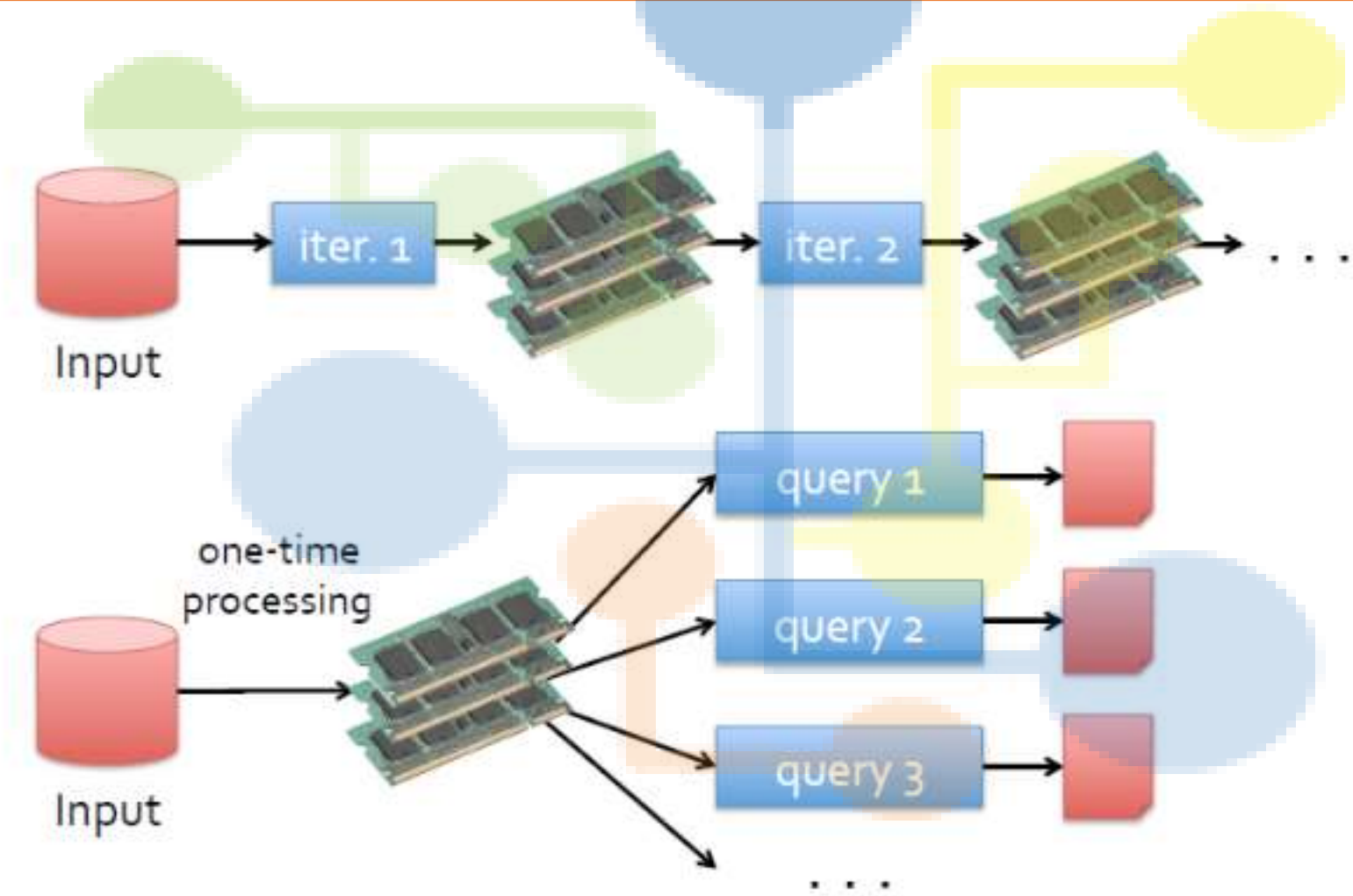
Apache Spark e Big Data

Clusters são conjuntos de computadores (servidores) conectados, que executam como se fossem um único sistema.

Apache Spark e Big Data



Apache Spark e Big Data



Apache Spark e Big Data

Apache Spark é um framework open-source para processamento de Big Data construído para ser veloz, fácil de usar e para análises sofisticadas.

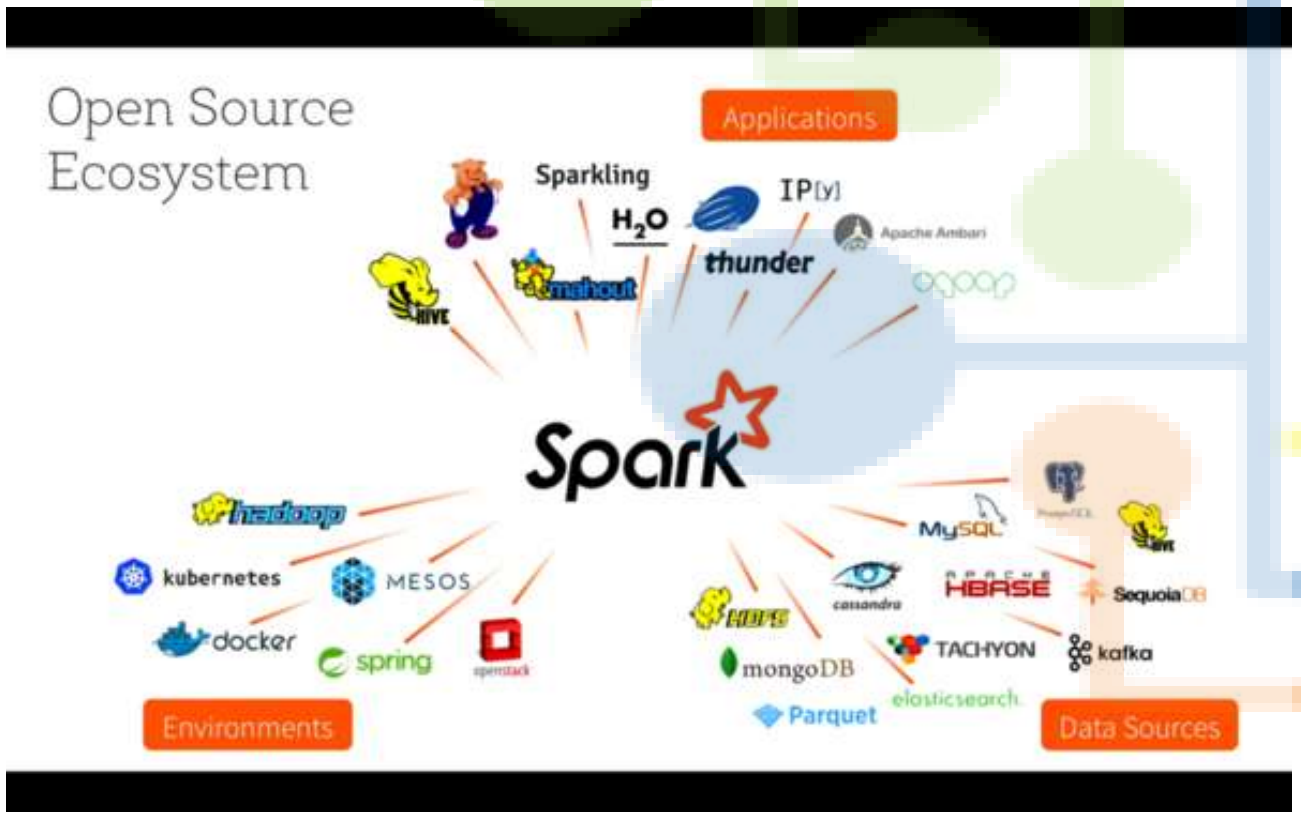
Apache Spark e Big Data

Apache Spark é uma ferramenta de análise de Big Data, escalável e eficiente.

Apache Spark e Big Data

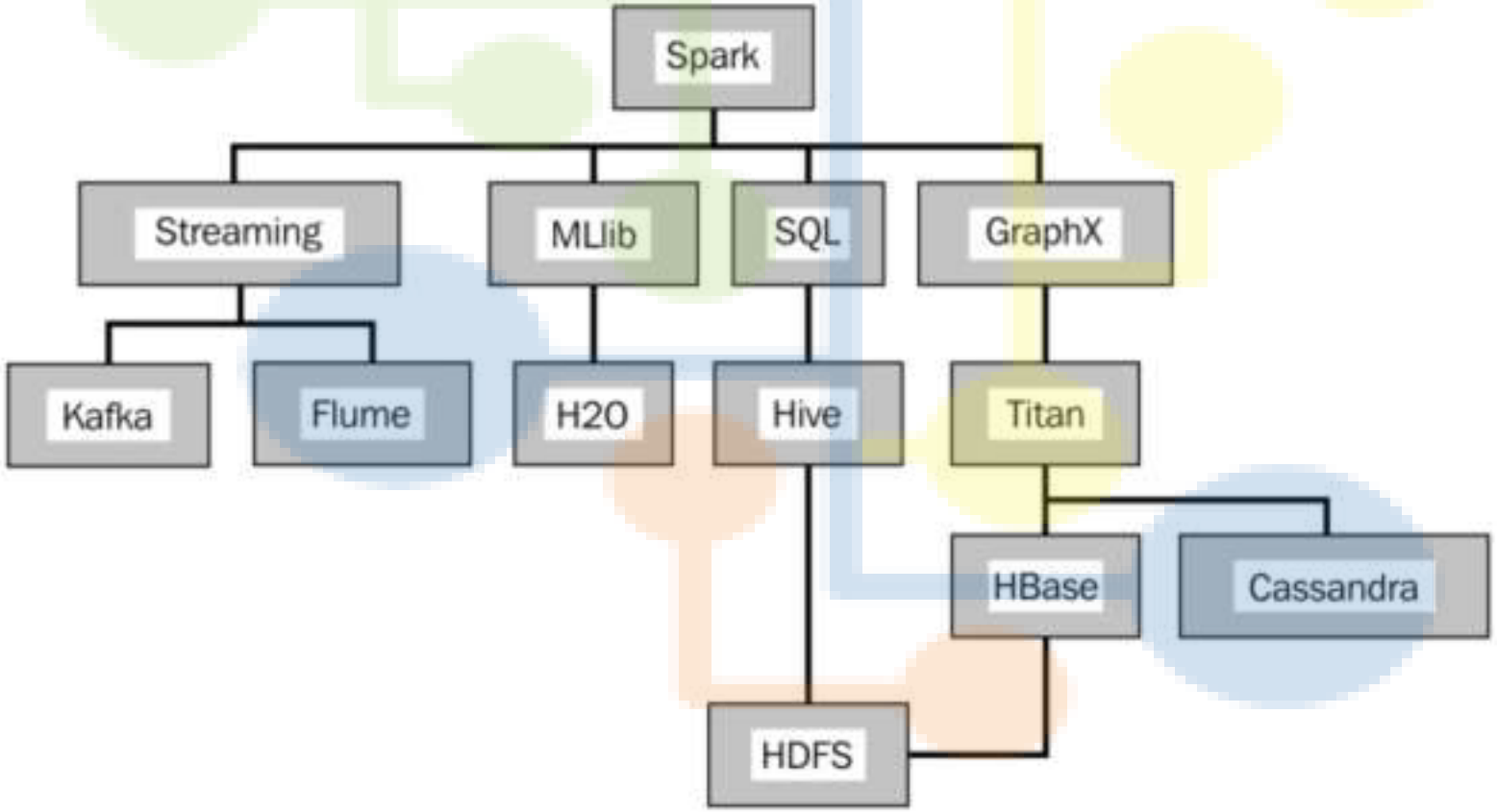
Ecossistema e Componentes do Apache Spark

Ecossistema e Componentes do Apache Spark

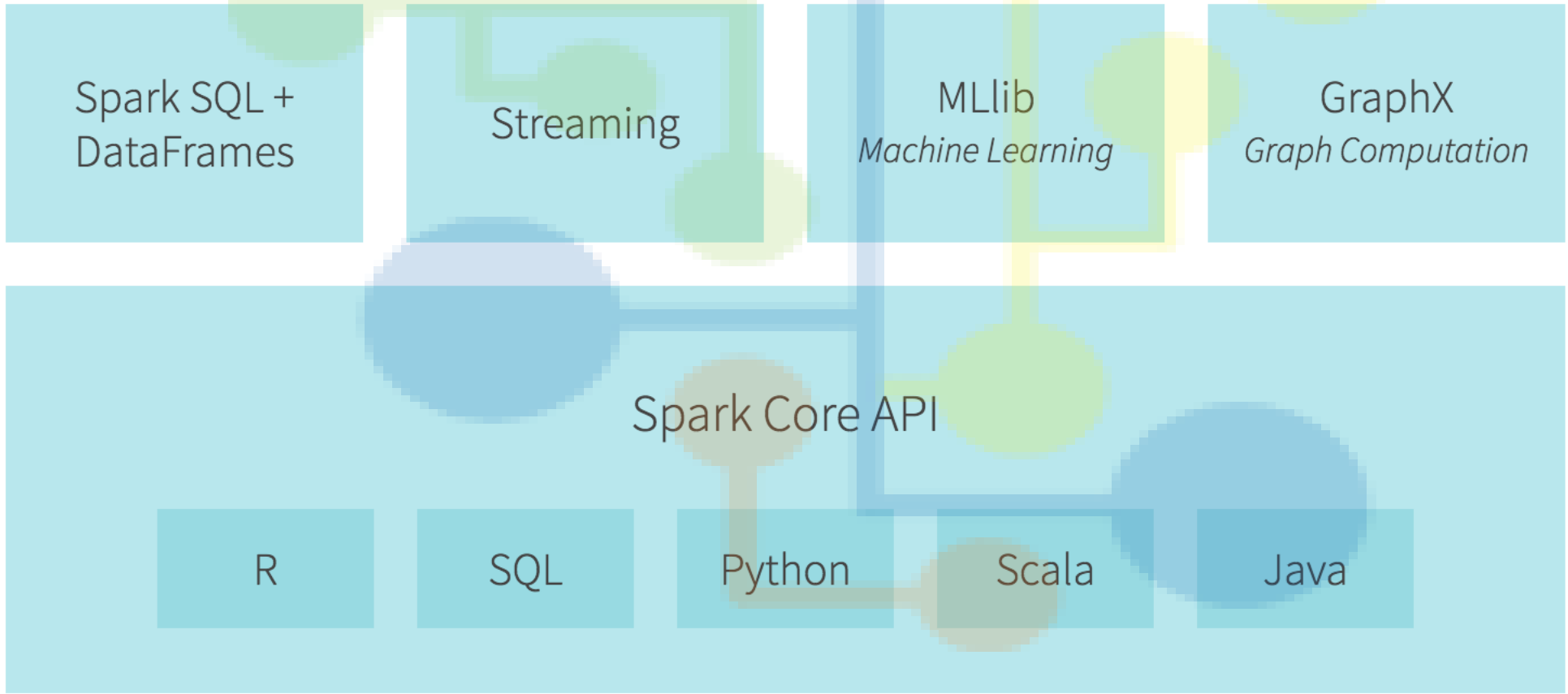


Ecossistema Spark

Ecossistema e Componentes do Apache Spark

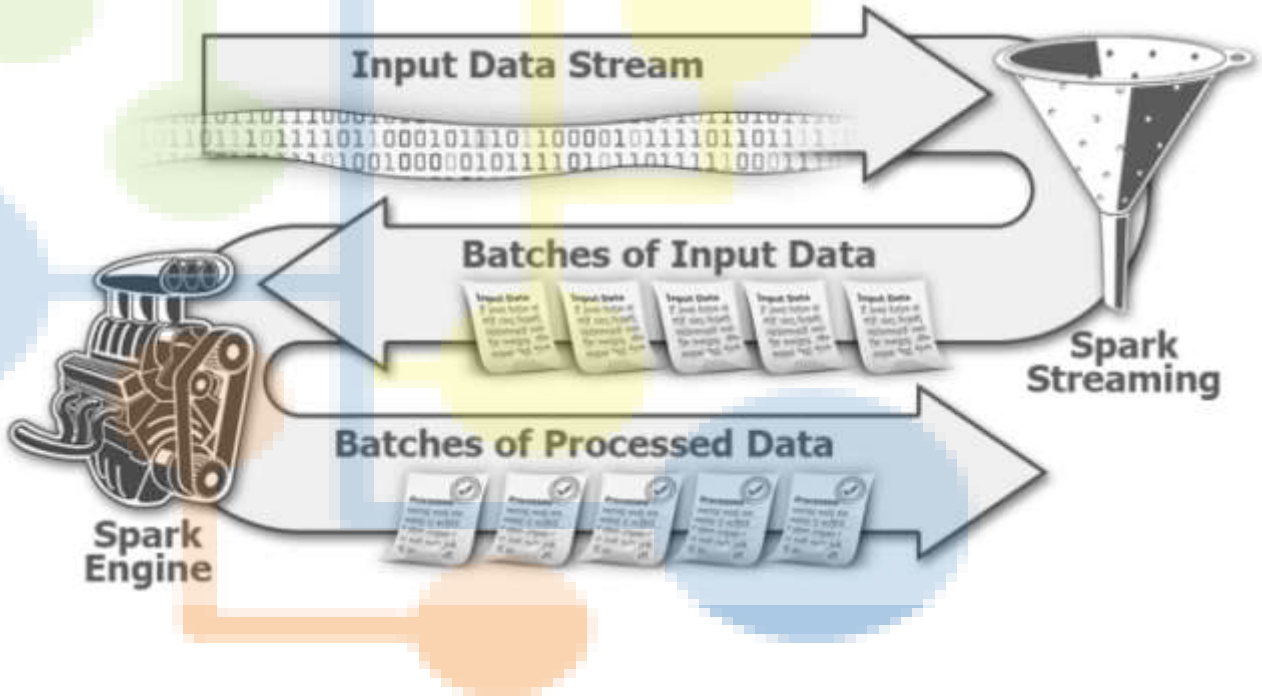


Ecossistema e Componentes do Apache Spark

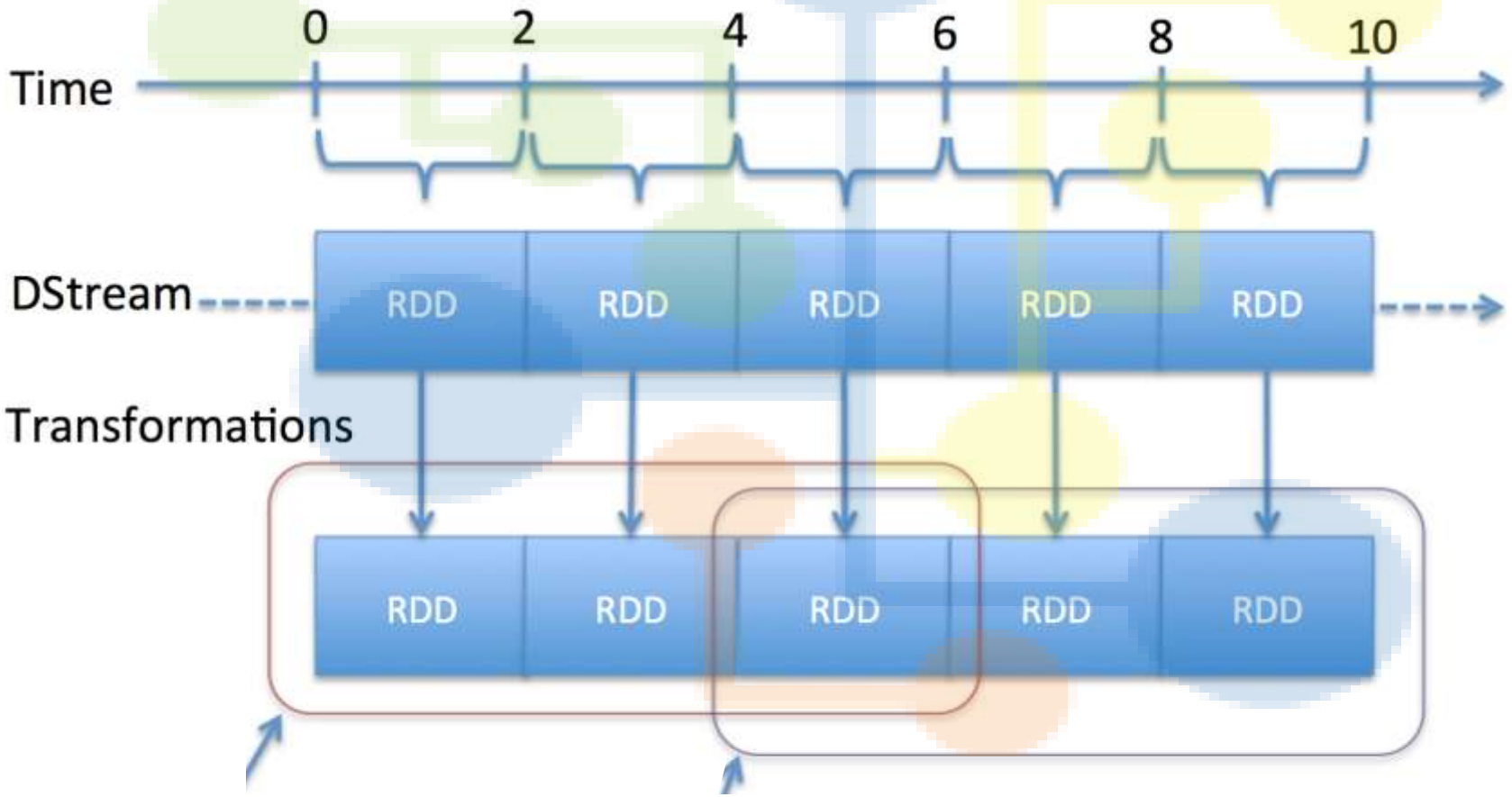


Ecossistema e Componentes do Apache Spark

Spark Streaming



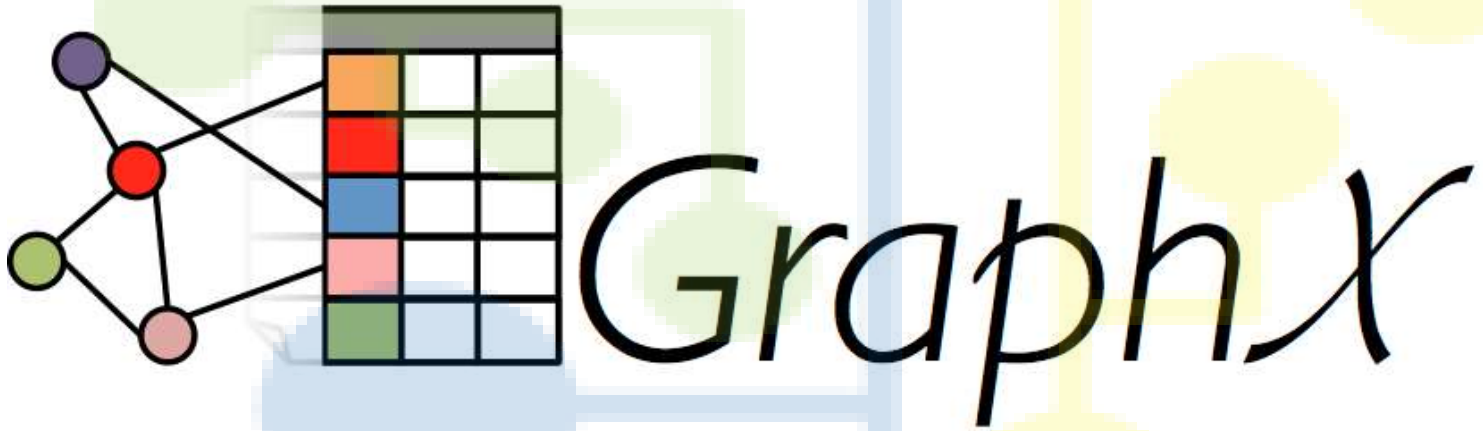
Ecossistema e Componentes do Apache Spark

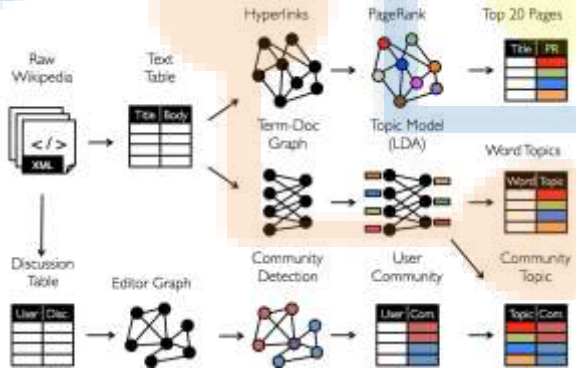


Ecossistema e Componentes do Apache Spark

Ecossistema e Componentes do Apache Spark

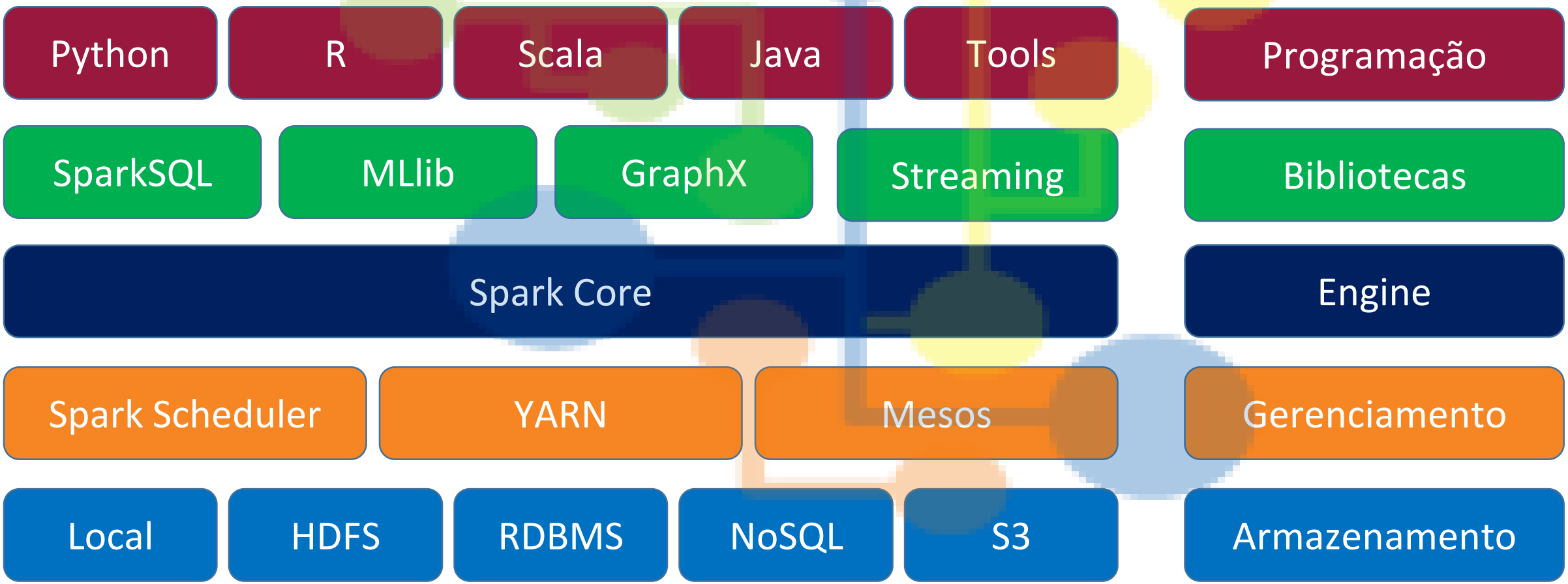
Ecossistema e Componentes do Apache Spark





Ecossistema e Componentes do Apache Spark

Ecossistema e Componentes do Apache Spark



Ecossistema e Componentes do Apache Spark

Quando Devemos Usar o Spark?

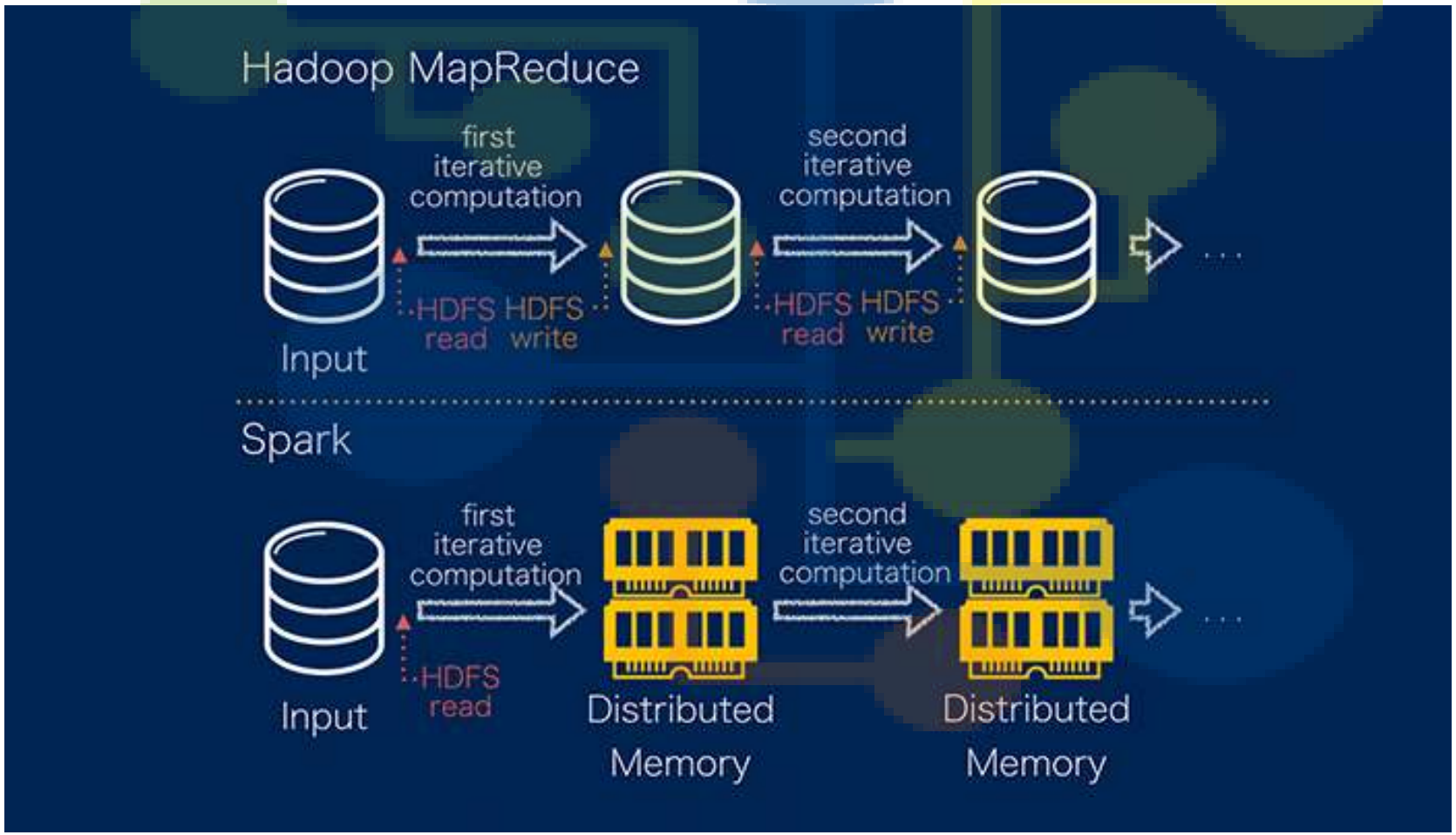
Ecossistema e Componentes do Apache Spark

Quando Devemos Usar o Spark?

- Integração de Dados e ETL
- Análises Interativas
- Computação em Batch de Alta Performance
- Análises Avançadas de Machine Learning
- Processamento de Dados em Tempo Real

Principais Características do Apache Spark

Principais Características do Apache Spark



Principais Características do Apache Spark

- Spark realiza operações de MapReduce
- Spark pode utilizar o HDFS
- Spark permite construir um workflow de Analytics
- Spark utiliza a memória do computador de forma diferente e eficiente
- Spark é veloz
- Spark é flexível
- Spark é gratuito

Big Data Real-Time Analytics com Python e Spark

Hadoop MapReduce
x
Apache Spark

Hadoop MapReduce x Apache Spark

Hadoop MapReduce
x
Apache Spark

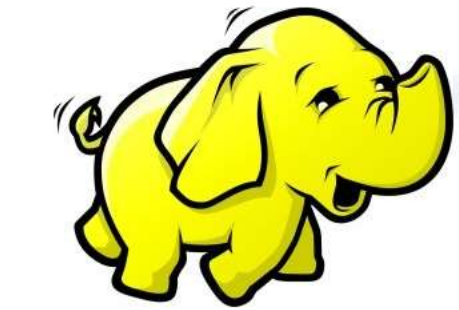
Hadoop MapReduce e Apache Spark são os dois frameworks mais populares para computação em cluster e análise de dados de larga escala (Big Data).

Hadoop MapReduce x Apache Spark

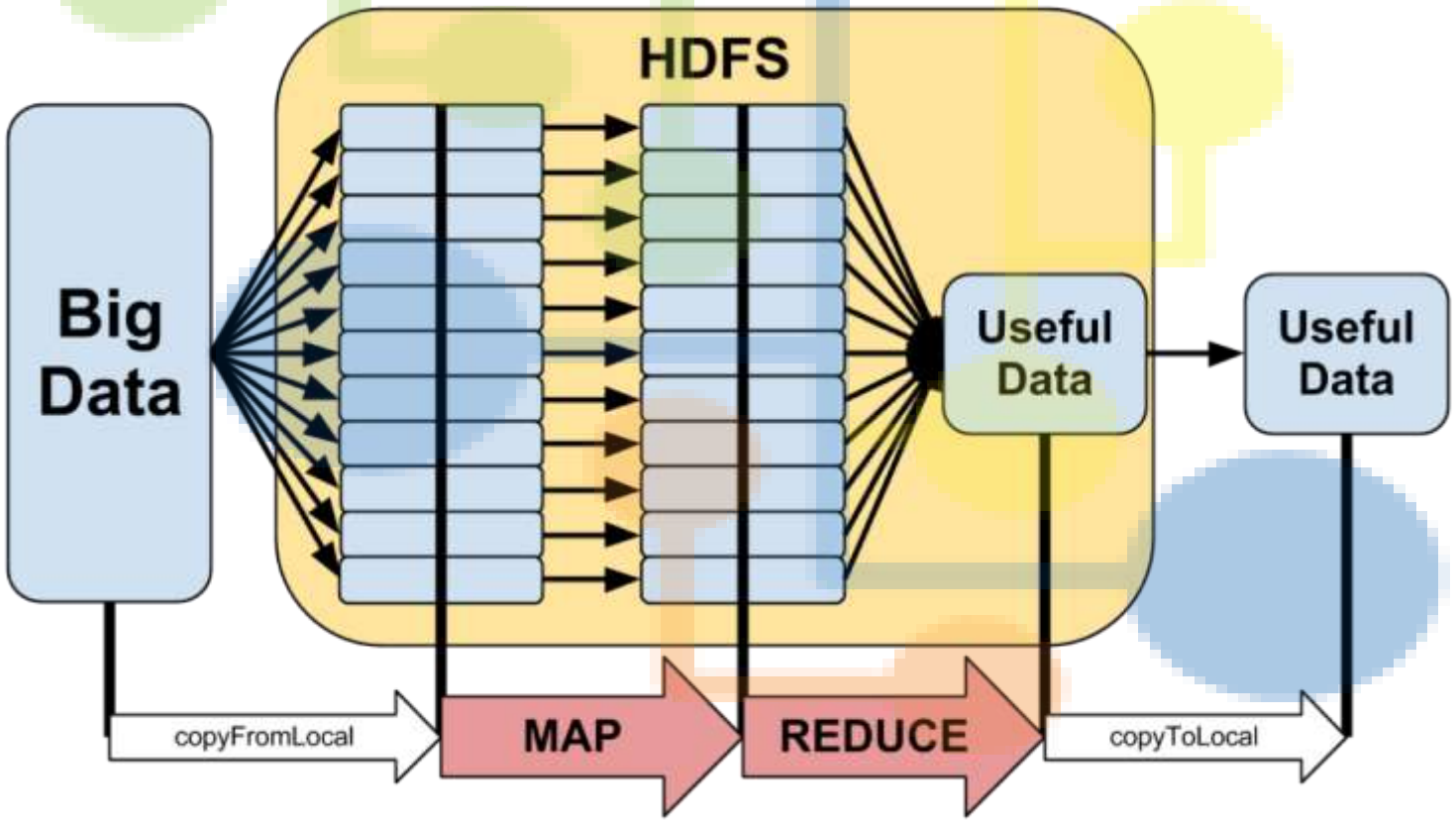
Hadoop MapReduce
x
Apache Spark

Estes dois frameworks escondem a complexidade existente no tratamento de dados com relação a paralelismo entre tarefas e tolerância a falha por meio da exposição de uma simples API com informações para os usuários.

Hadoop MapReduce x Apache Spark



Hadoop MapReduce x Apache Spark



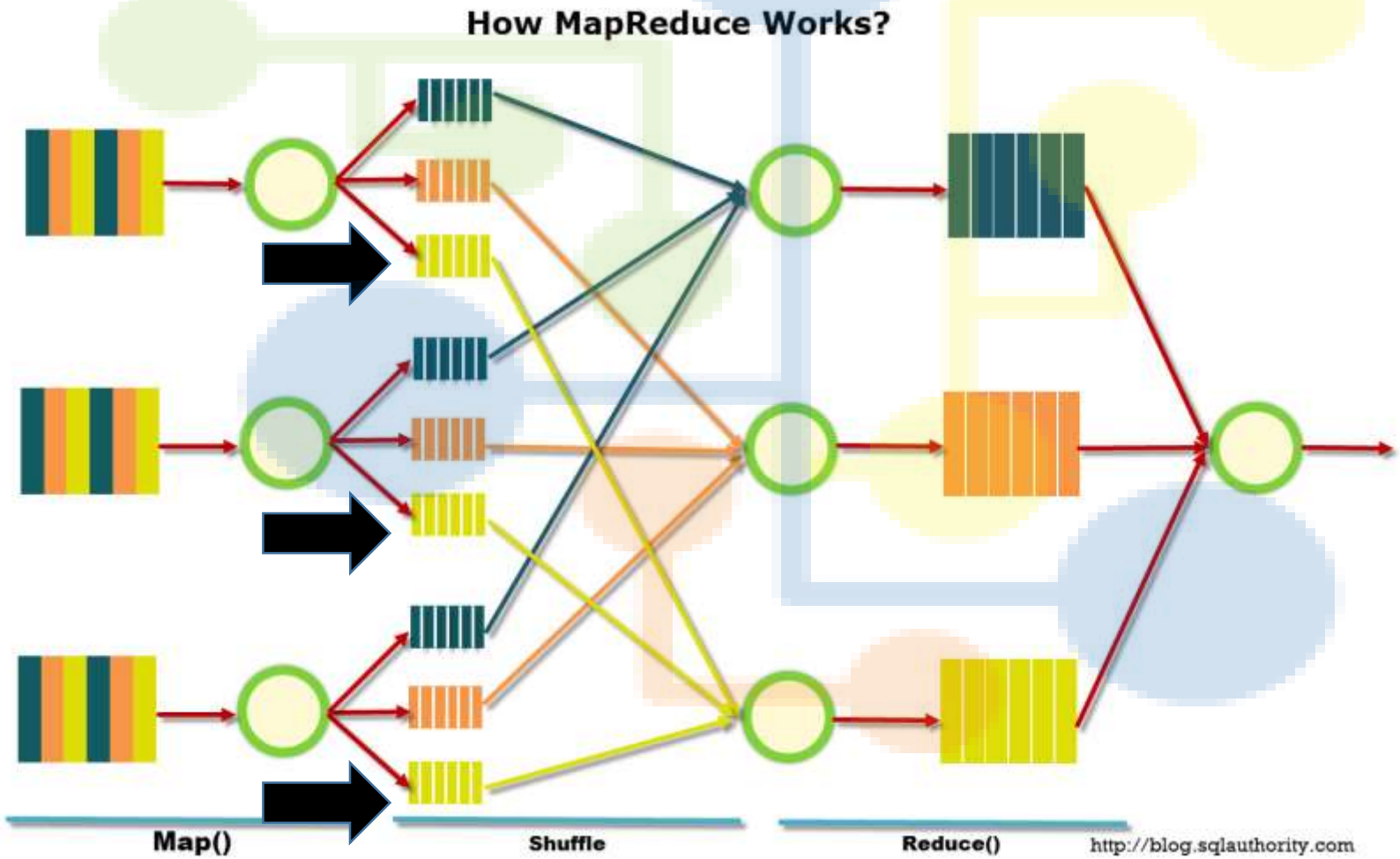
Hadoop MapReduce x Apache Spark

O Spark realiza o processamento distribuído, de forma similar ao Hadoop MapReduce, porém com muito mais velocidade.

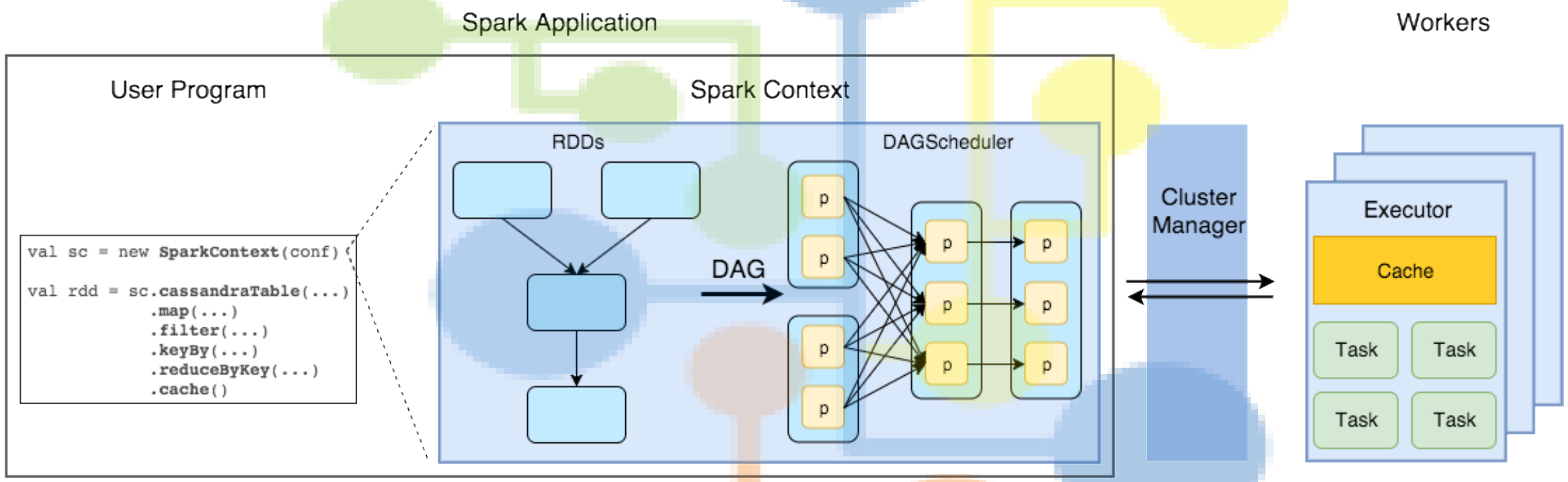
Hadoop MapReduce x Apache Spark

O Spark não possui sistema de armazenamento, podendo usar o HDFS como fonte/destino de dados.

O Processo de MapReduce no Apache Spark

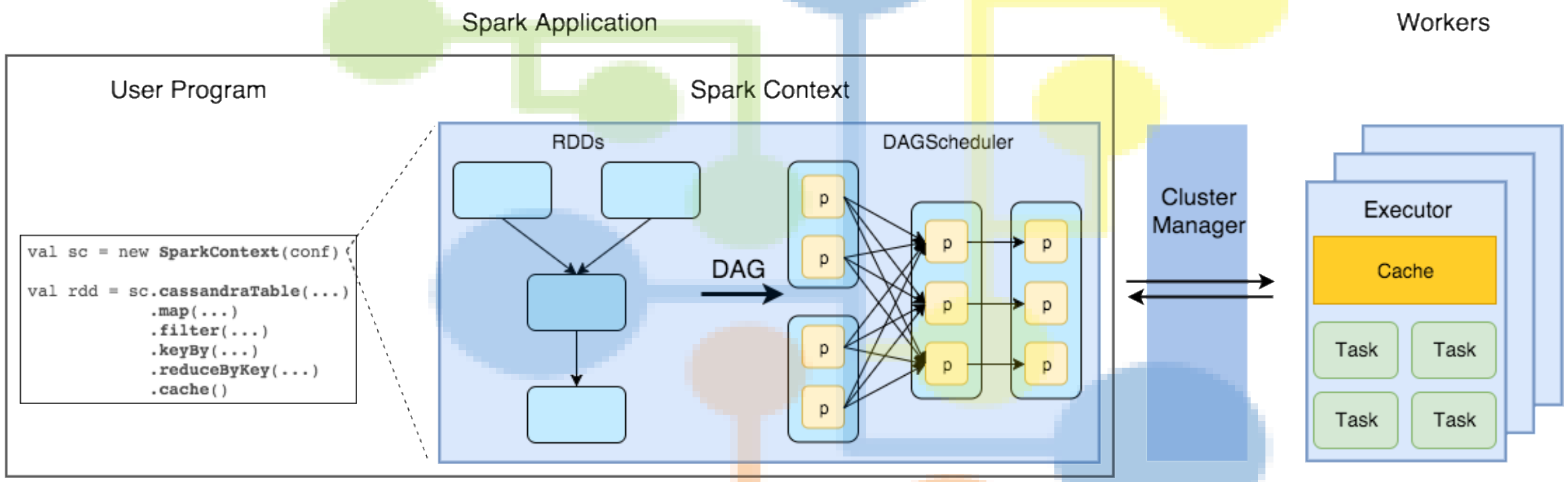


O Processo de MapReduce no Apache Spark



RDD's = Resilient Distributed Datasets

O Processo de MapReduce no Apache Spark



RDD's = Resilient Distributed Datasets

O Processo de MapReduce no Apache Spark

- O Spark suporta mais do que apenas as funções de Map e Reduce.
- Hadoop MapReduce grava os resultados intermediários em disco, enquanto o Spark grava os resultados intermediários em memória, o que é muito mais rápido.
- O Spark fornece APIs concisas e consistentes em Scala, Java e Python (e mais recentemente em R).
- O Spark oferece shell interativo para Scala, Python e R.
- O Spark pode utilizar o HDFS como uma de suas fontes/destinos de dados.

O Processo de MapReduce no Apache Spark

O Cientista de Dados é responsável por definir as regras de manipulação e análise de dados.

O Engenheiro de Dados é responsável por garantir o processamento distribuído, cuidando do pipeline, fontes de dados e destino, bem como pela segurança dos dados.

O Processo de MapReduce no Apache Spark

O Spark e o Hadoop MapReduce tem uma característica principal em comum: são responsáveis por gerenciar o processamento distribuído.

Porém o Spark faz isso de forma muito mais rápida e eficiente que o Hadoop MapReduce.

Profissionais Que Trabalham com Apache Spark

Profissionais Que Trabalham com Apache Spark

Exstem basicamente 3 perfis de profissionais que vão trabalhar com Spark:

- Cientistas de Dados
- Engenheiros de Dados
- Administradores

Anatomia de Uma Aplicação Spark

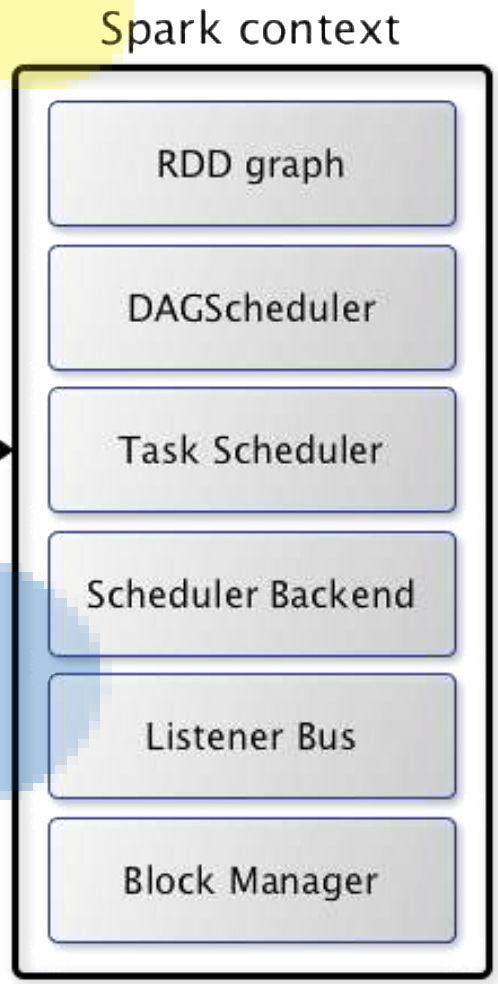
Anatomia de Uma Aplicação Spark

Cada aplicação Spark inicia uma instância de um Spark Context.

```
val sc = new SparkContext(master="local[*]",
          appName="SparkMe App", new SparkConf)

val lines = sc.textFile(...).cache()

val c = lines.count()
println(s"There are $c lines in $fileName")
```



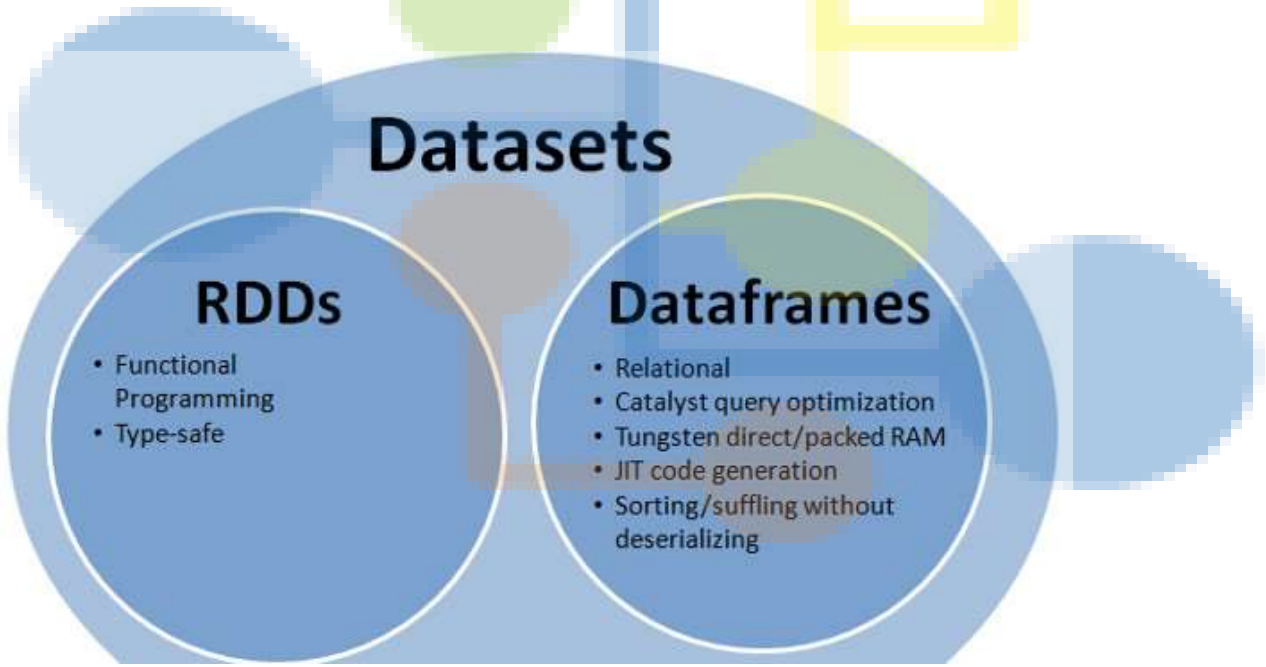
Sem um Spark Context, nada pode ser feito no Spark. Cada aplicação Spark é uma instância de um Spark Context.

Anatomia de Uma Aplicação Spark

O Spark Context é basicamente uma espécie de cliente que estabelece a conexão com o ambiente de execução do Spark e age como o processo principal da sua aplicação.

Anatomia de Uma Aplicação Spark

Com o Spark Context criado, podemos então definir os nossos RDD's e Dataframes, que são os objetos que vão armazenar os dados para o processamento pelo Spark.



Anatomia de Uma Aplicação Spark

E como criamos um Spark Context?

1. **Shell** – área de trabalho via linha de comando
2. **Spark Context** – conexão ao ambiente Spark criado quando iniciamos o pyspark

Ou seja, uma vez que iniciamos o PySpark, é criado um Spark Context, que nos permite criar RDD's e Dataframes e realizar transformações e ações. Cada operação Spark gera um job que então é executado ou agendado para ser executado ao longo do cluster de computadores ou localmente em nossa máquina.

Anatomia de Uma Aplicação Spark

Quando utilizamos o PySpark e o Jupyter Notebook não precisamos nos preocupar com a criação do Spark Context.

Anatomia de Uma Aplicação Spark

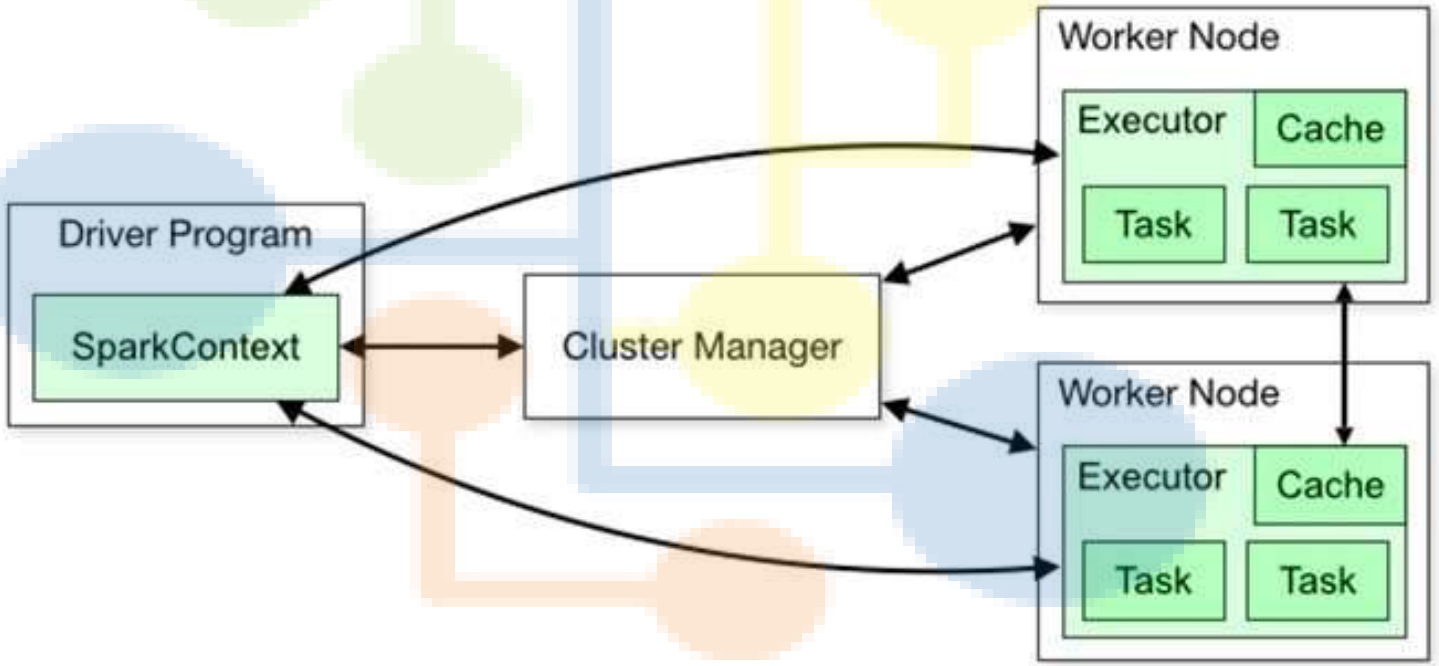
Quando utilizamos o PySpark e o Jupyter Notebook não precisamos nos preocupar com a criação do Spark Context.

Mas quando criamos uma aplicação de análise de dados (um arquivo .py por exemplo) e utilizamos o spark-submit para iniciar nossa aplicação, precisamos explicitamente criar um Spark Context.

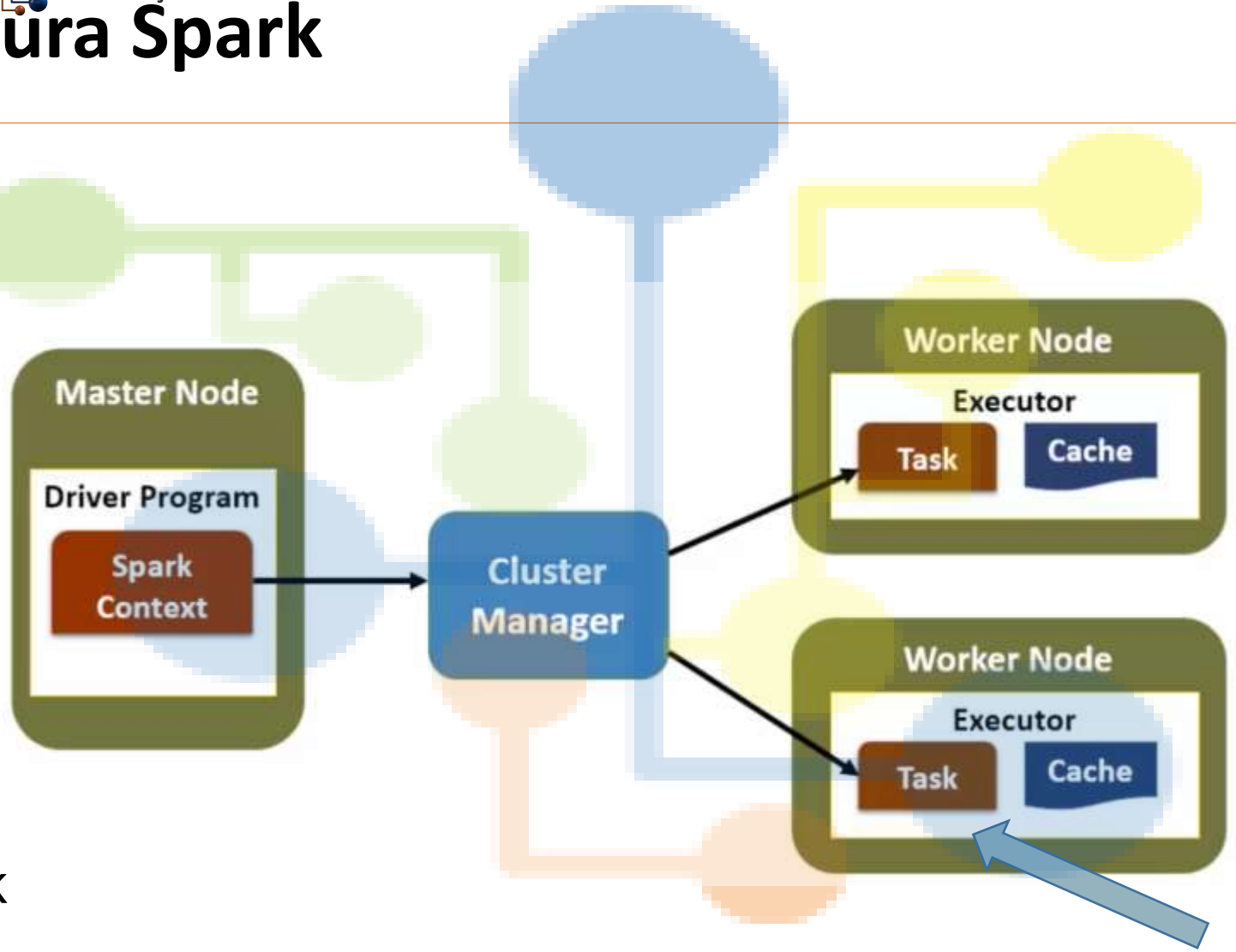
Arquitetura Spark

Arquitetura Spark

Arquitetura Spark
Master/Worker



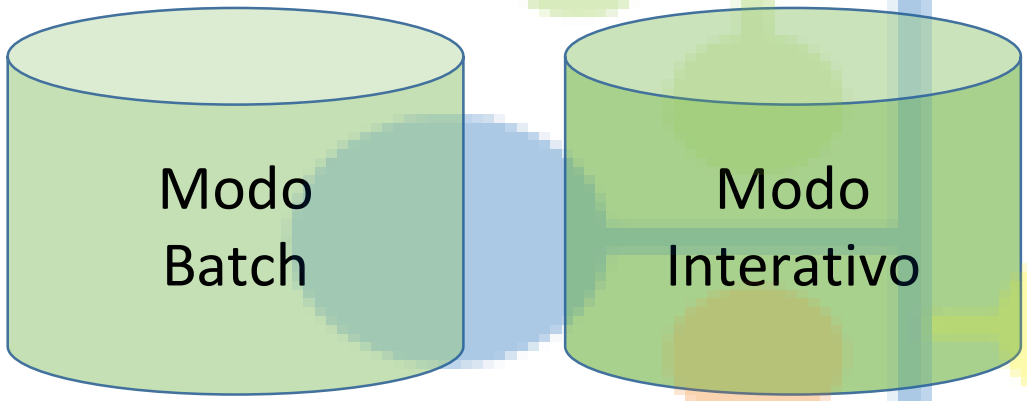
Arquitetura Spark



Arquitetura Spark
Task

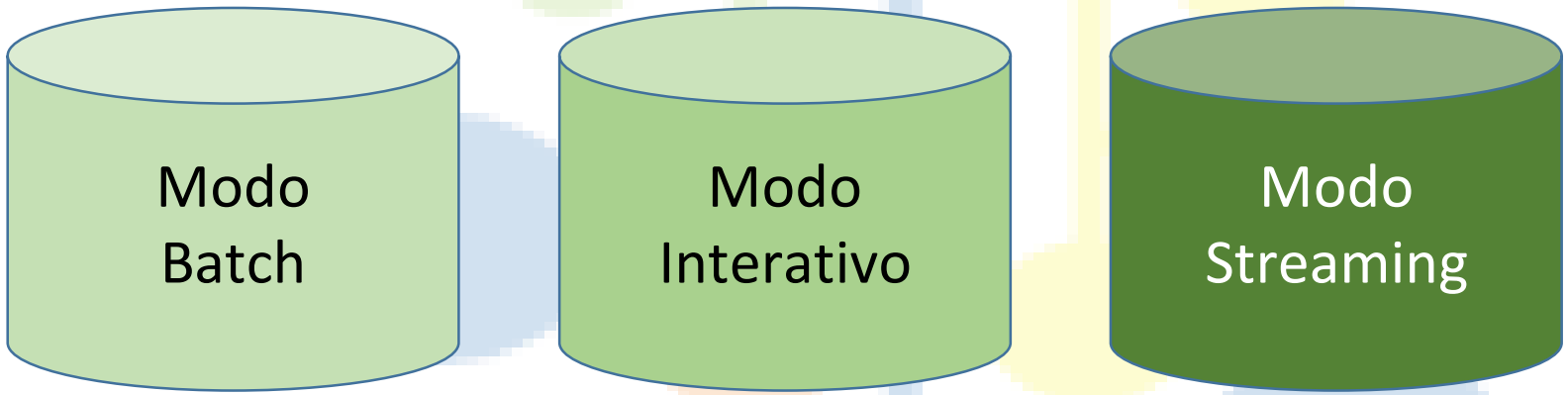
Spark Modes

Spark Modes



Utiliza o shell para executar comandos no cluster. O shell age como um driver program e provê um SparkContext.

Spark Modes



Um programa que executa continuamente para processar os dados à medida que eles chegam, em tempo real.

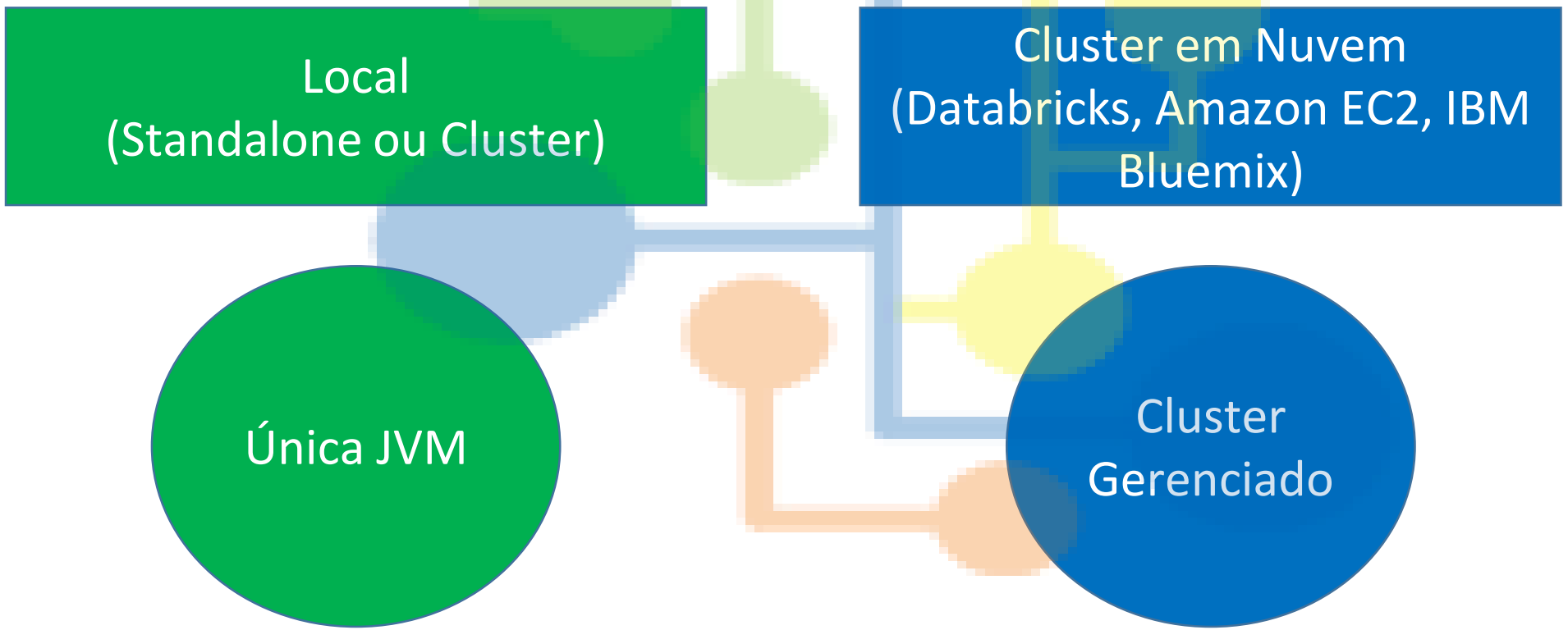
Deploy Mode e Fontes de Dados

Deploy Mode

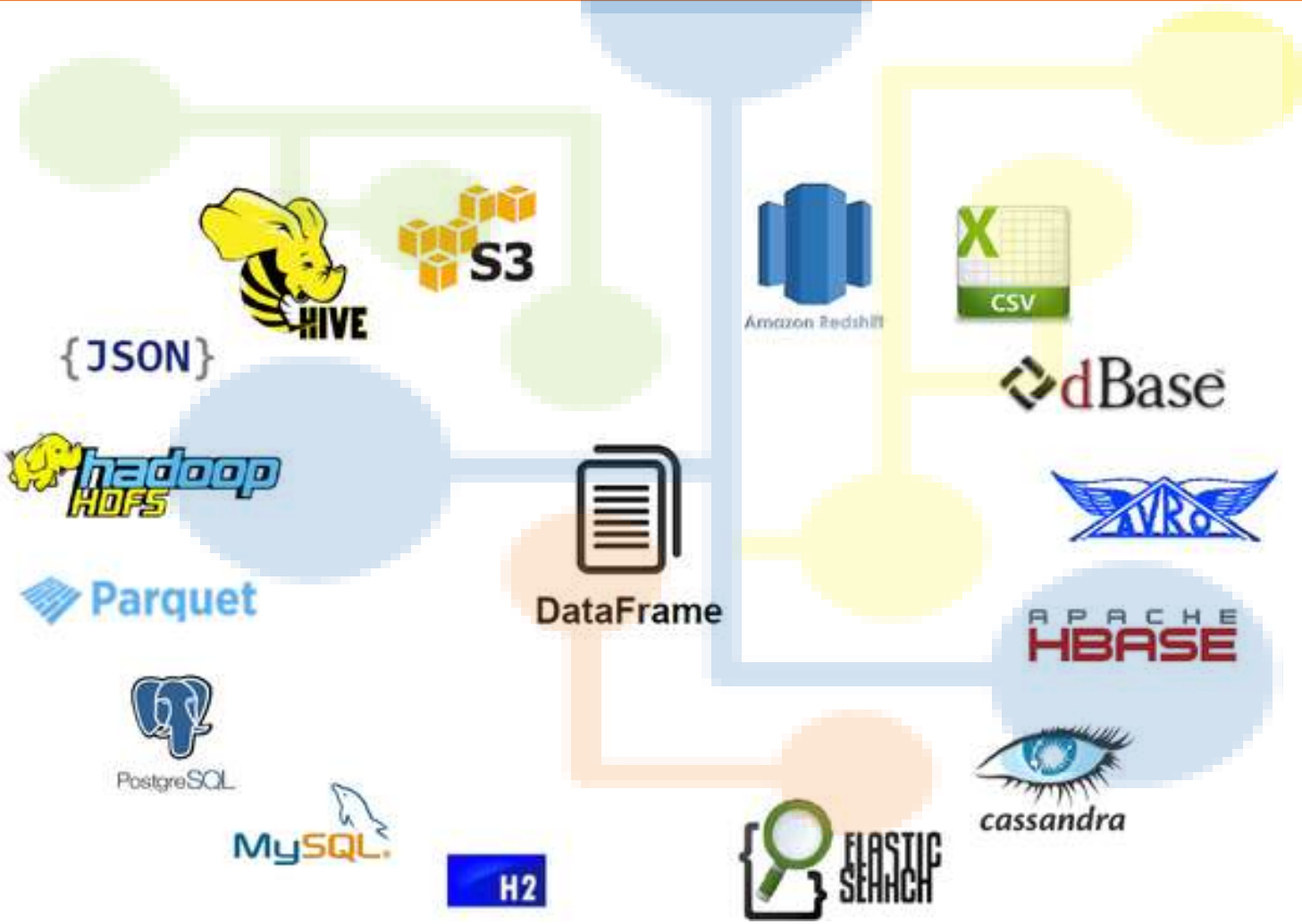
Spark Mode → Modo de processamento de dados no Spark (Batch, Interativo ou Streaming).

Deploy Mode → Modo de execução do Spark. Em cada Deploy Mode podemos usar um ou mais Spark Modes.

Deploy Mode



Fontes de Dados



RDD's
Resilient Distributed Datasets

RDD's - Resilient Distributed Datasets

RDD é uma coleção de objetos, distribuída e imutável. Cada conjunto de dados no RDD é dividido em partições lógicas, que podem ser computados em diferentes nodes do cluster.

RDD's - Resilient Distributed Datasets

RDD’s são imutáveis!

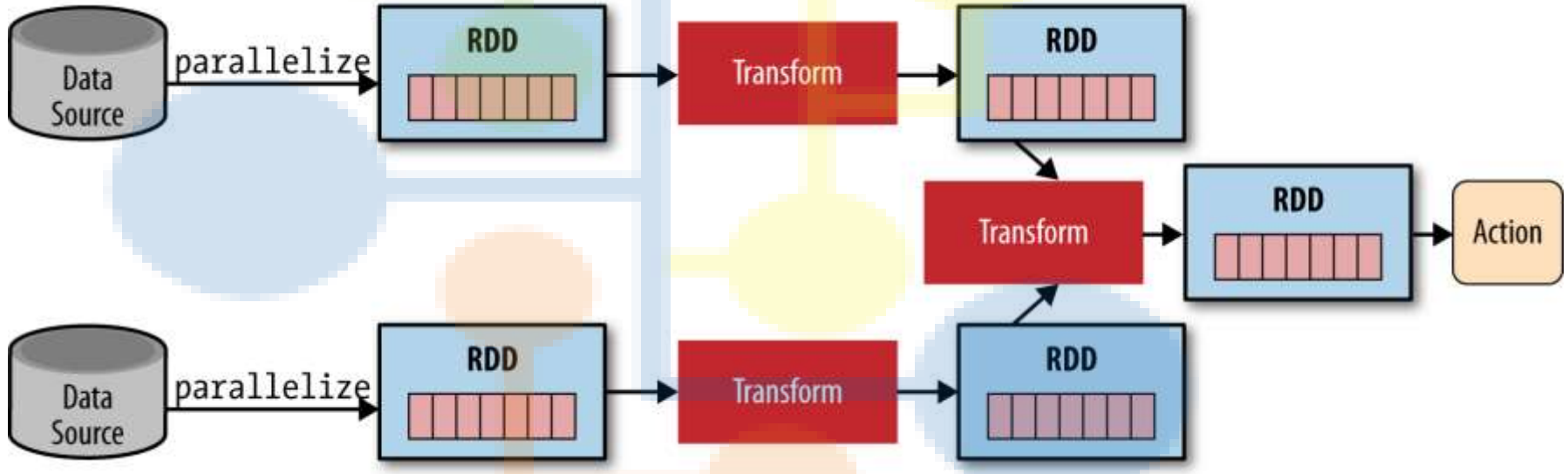
RDD's - Resilient Distributed Datasets

Arquivos

HDFS

RDBMS

NoSQL



RDD's - Resilient Distributed Datasets

Existem 2 formas de criar o RDD

Paralelizando uma coleção existente
(função sc.parallelize)

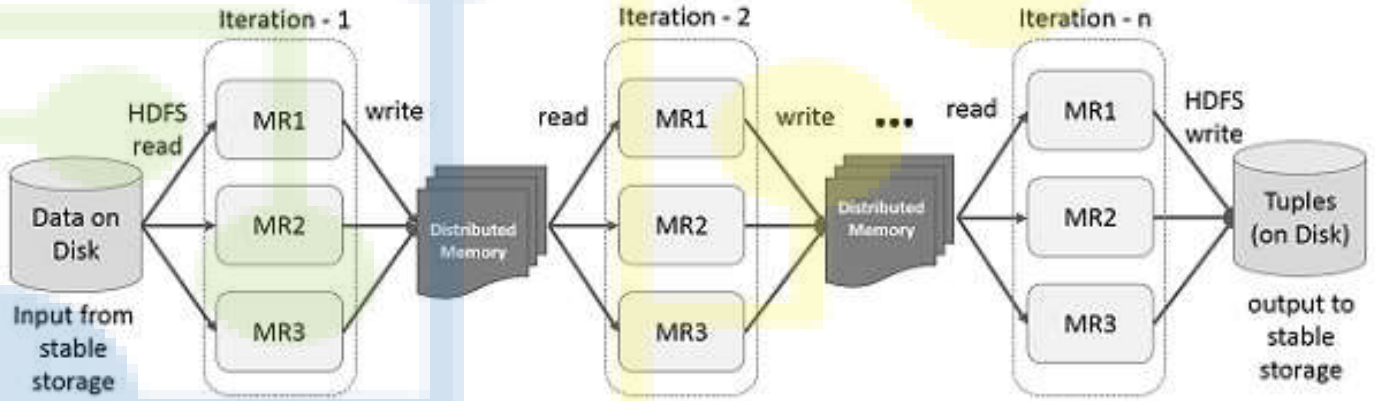
Referenciando um dataset externo
(HDFS, RDBMS, NoSQL, S3)

RDD's - Resilient Distributed Datasets

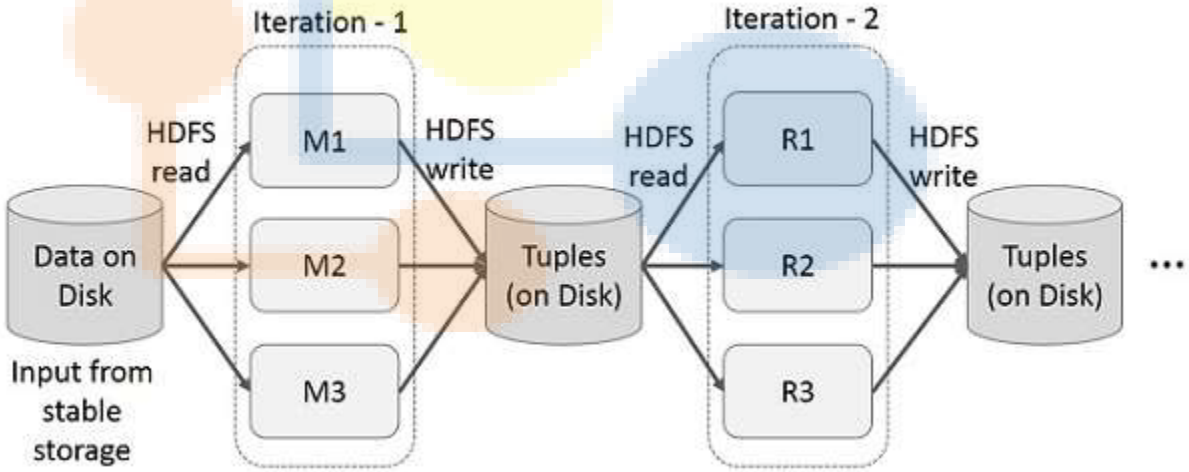
As RDD’s são a essência do funcionamento do Spark

RDD's - Resilient Distributed Datasets

Apache Spark



Hadoop MapReduce

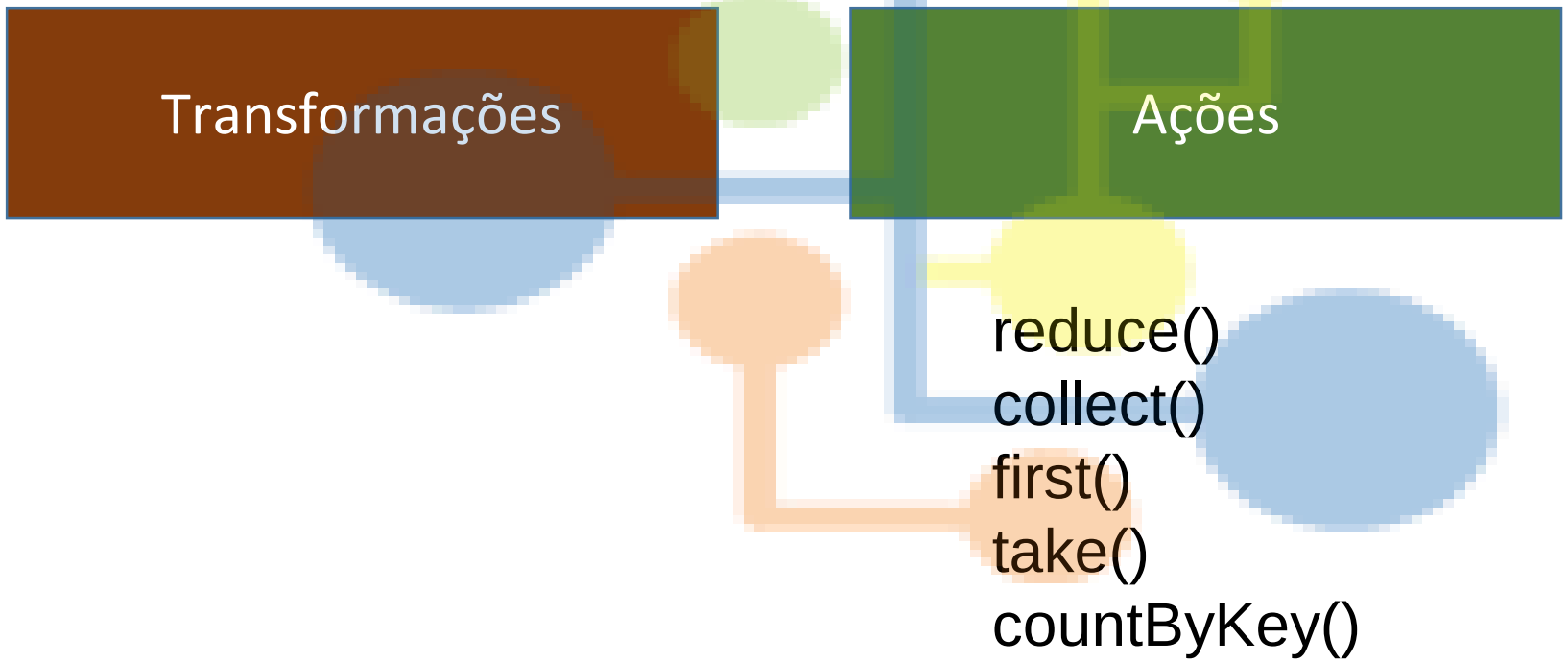


RDD's - Resilient Distributed Datasets

Por padrão, os RDD’s são computados cada vez que executamos uma Ação. Entretanto, podemos ”persistir” o RDD na memória (ou mesmo no disco) de modo que os dados estejam disponíveis ao longo do cluster e possam ser processados de forma muito mais rápida pelas operações de análise de dados criadas por você, Cientista de Dados.

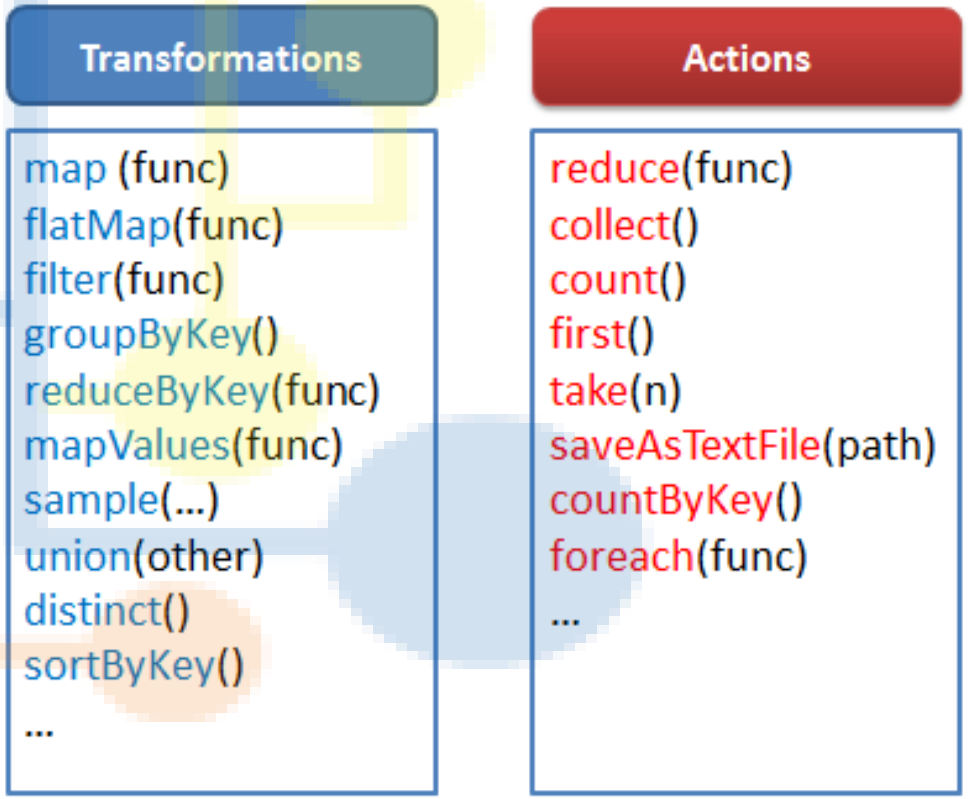
RDD's - Resilient Distributed Datasets

O RDD suporta dois tipos de operações:

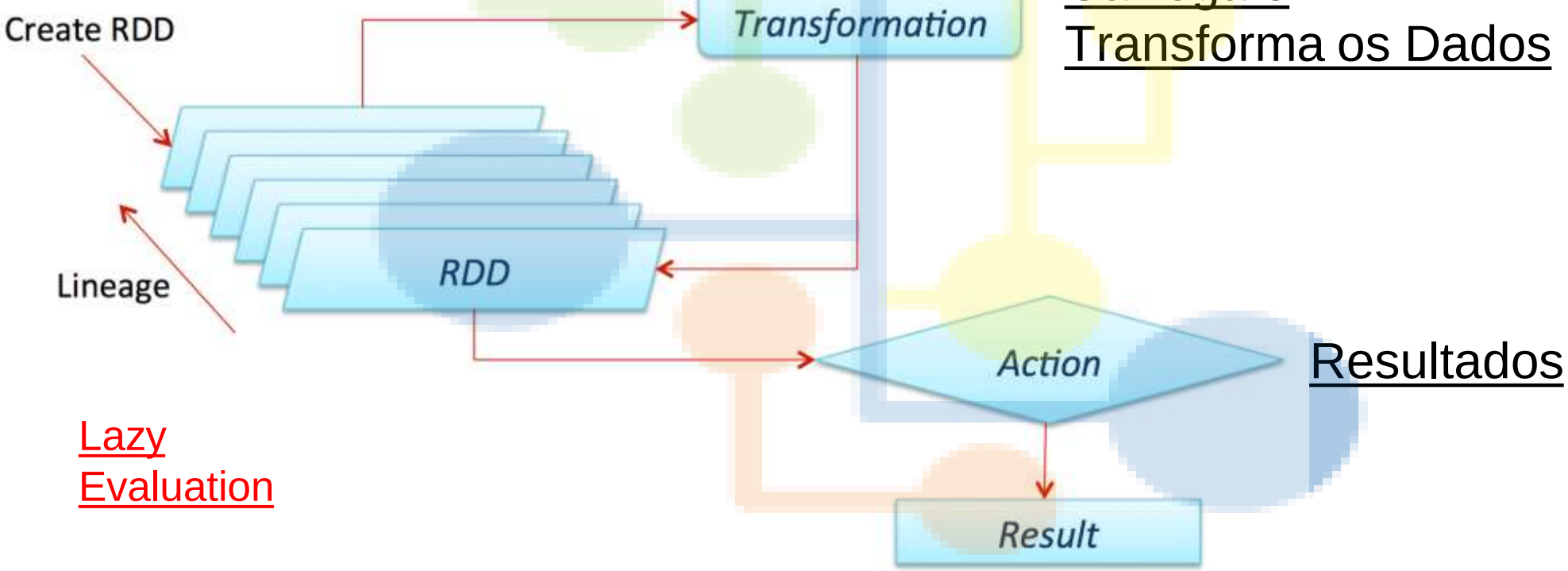


RDD's - Resilient Distributed Datasets

Transformações e Ações



RDD's - Resilient Distributed Datasets



Carrega e Transforma os Dados

Resultados

Lazy Evaluation

RDD's - Resilient Distributed Datasets

As ”Ações” aplicam as ”Transformações” nos RDD’s e retornam resultado.

RDD's - Resilient Distributed Datasets

- Spark é baseado em RDD’s. Criamos, transformamos e armazenamos RDD’s com Spark.

RDD's - Resilient Distributed Datasets

- Spark é baseado em RDD’s. Criamos, transformamos e armazenamos RDD’s com Spark.
- RDD representa uma coleção de elementos de dados particionados que podem ser operados em paralelo.
- RDD’s são objetos imutáveis. Eles não podem ser alterados uma vez criados.
- RDD’s podem ser colocados em cache e permitem persistência (mesmo objeto usado entre sessões diferentes).
- Ao aplicarmos Transformações em RDD’s criamos novos RDD’s.

RDD's - Resilient Distributed Datasets

Portanto, as 3 características principais dos RDD's são:

Imutablidade

Importante quando se realiza processamento paralelo.

Particionado e Distribuído

Permite processar arquivos através de diversos computadores.

Armazenamento em Memória

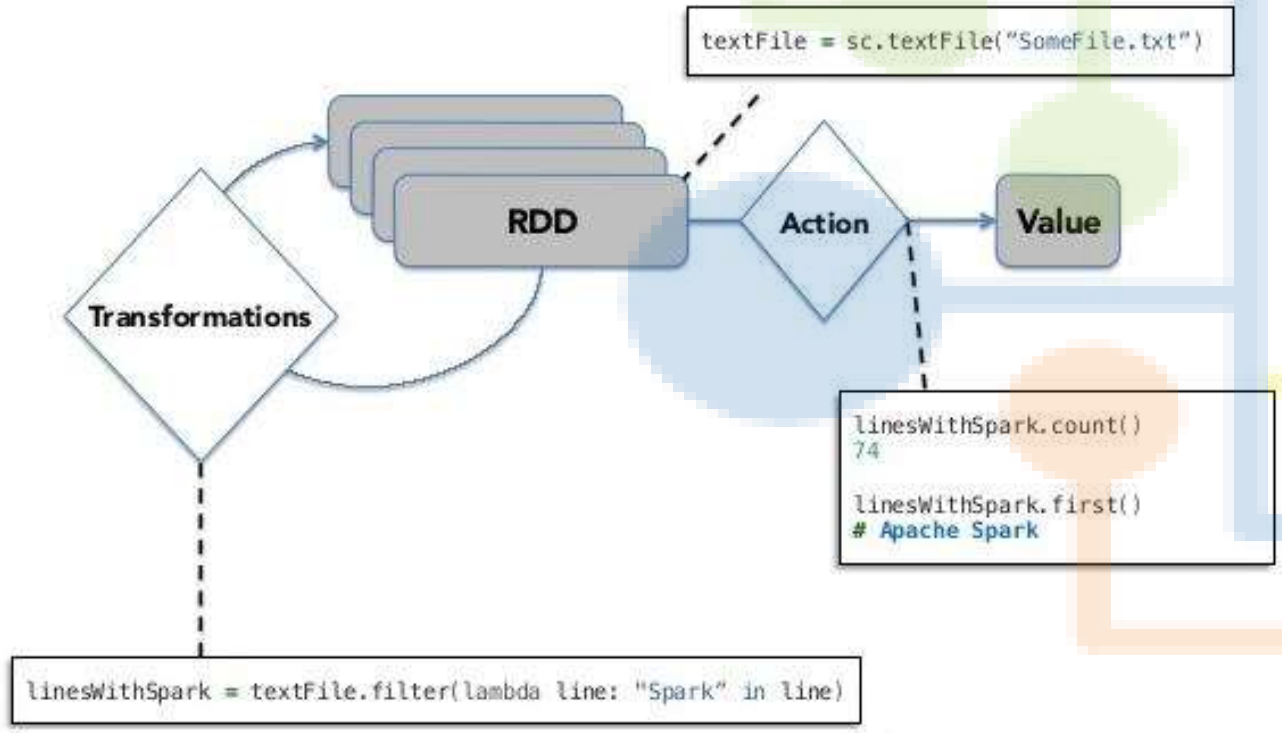
Processamento muito mais veloz, permitindo armazenar os resultados intermediários em memória.

O Que São Transformações?

O Que São Transformações?

Transformações são “operações preguiçosas” (lazy operations) executadas sobre os RDD’s e que criam um ou mais RDD’s.

O Que São Transformações?

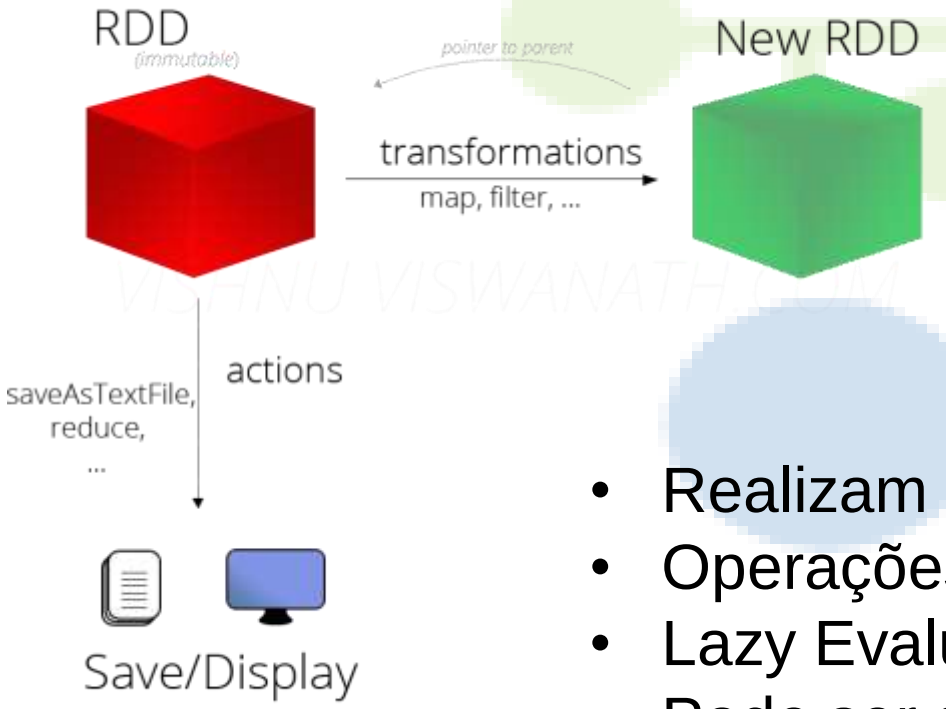


Dizemos que as transformações são operações lazy, porque elas não são executadas imediatamente, mas sim no momento em que as operações de ação são executadas.

O Que São Transformações?

Após executar operações de Transformação no RDD, o RDD resultante será diferente do RDD original e poderá ser menor (se usadas as funções filter, count, distinct, sample) ou maior (se usadas as funções flatMap, union, cartesian).

O Que São Transformações?

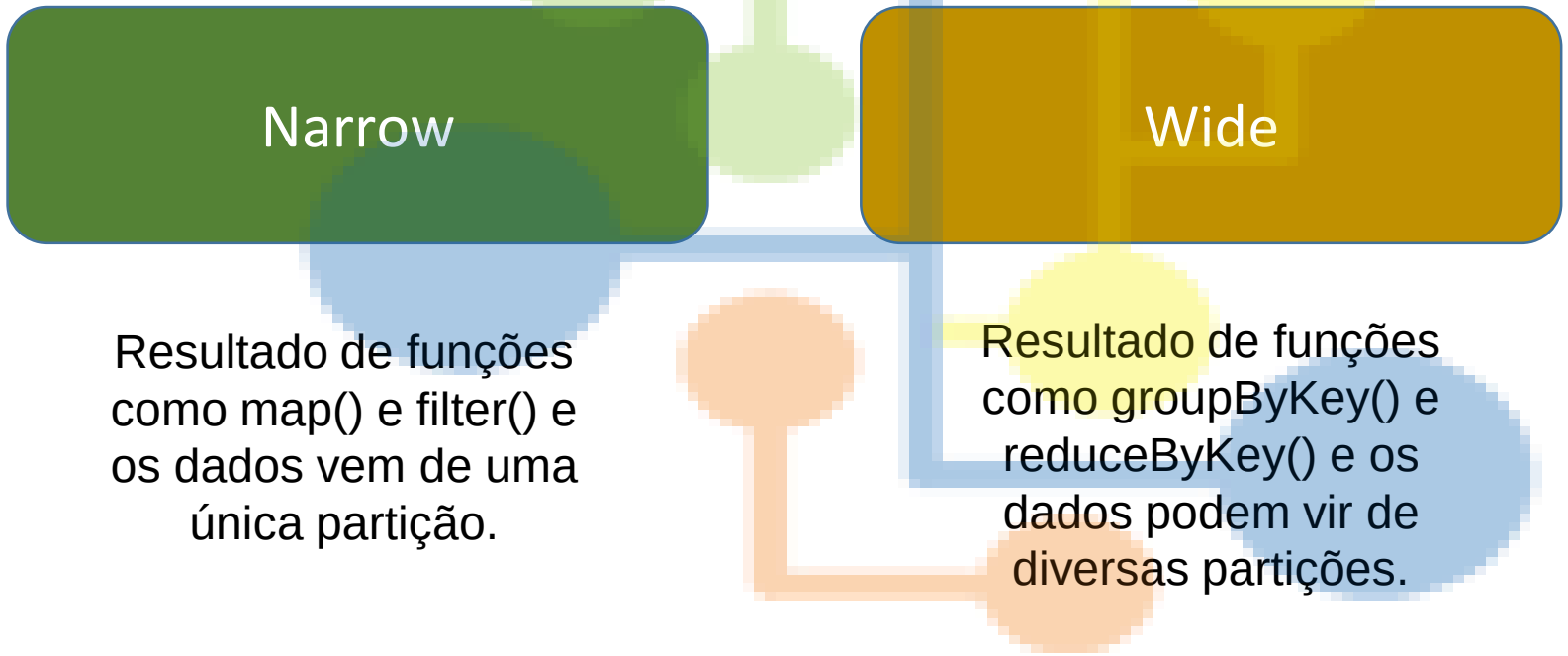


- Realizam as operações em um RDD e criam um novo RDD.
- Operações são feitas em um elemento por vez.
- Lazy Evaluation.
- Pode ser distribuída através de múltiplos nodes.

O Que São Transformações?

Algumas operações de Tranformação podem ser colocadas no que o Spark chama de Pipeline, que é um encadeamento de transformações visando aumentar a performance.

O Que São Transformações?



Principais Operações de Transformação

Principais Operações de Transformação

Map

- Conceito de MapReduce
- Age sobre cada elemento e realiza a mesma operação

Principais Operações de Transformação

flatMap

- Funciona como a função Map, mas retorna mais elementos

Principais Operações de Transformação

Filter

- Filtra um RDD para retornar elementos

Principais Operações de Transformação

Set

- São realizadas em duas RDD's, com operações de união e interseção

Principais Operações de Transformação

mapPartitions

- Quando utilizamos como fontes de dados bancos de dados como Hbase ou Cassandra, temos dados armazenados com pares chave-valor. A transformação mapPartitions garante a atomicidade dos dados, evitando problemas de overhead na manipulação de dados e garantindo performance.

Principais Operações de Transformação

Lista com Todas as Operações de Transformação:

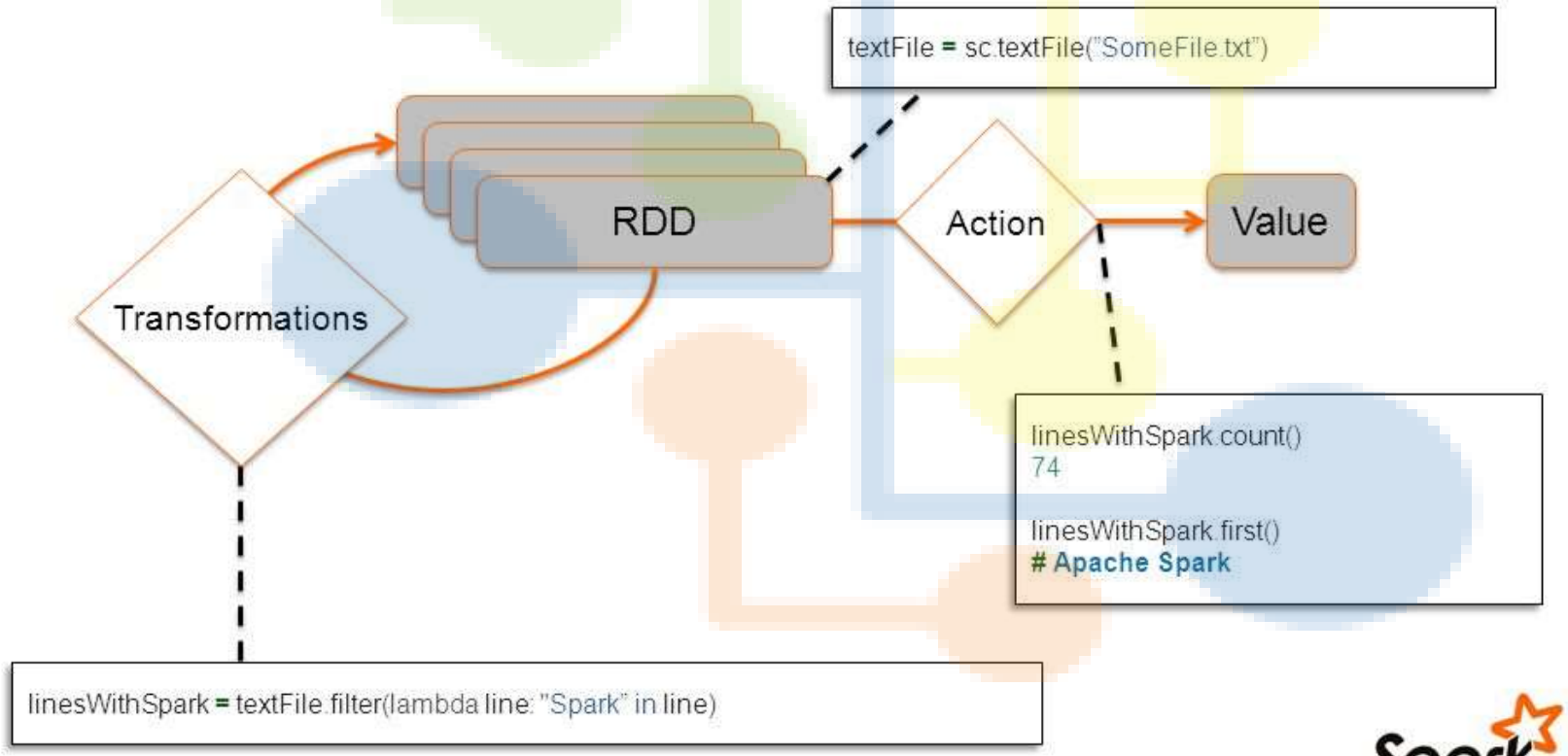
http://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations

O Que São Ações?

O Que São Ações?

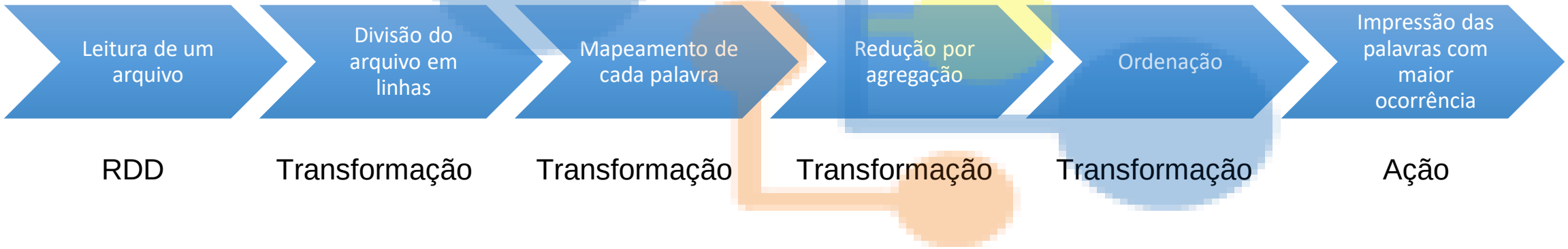
São operações executadas sobre os RDD's que geram um resultado.

O Que São Ações?



O Que São Ações?

Ações

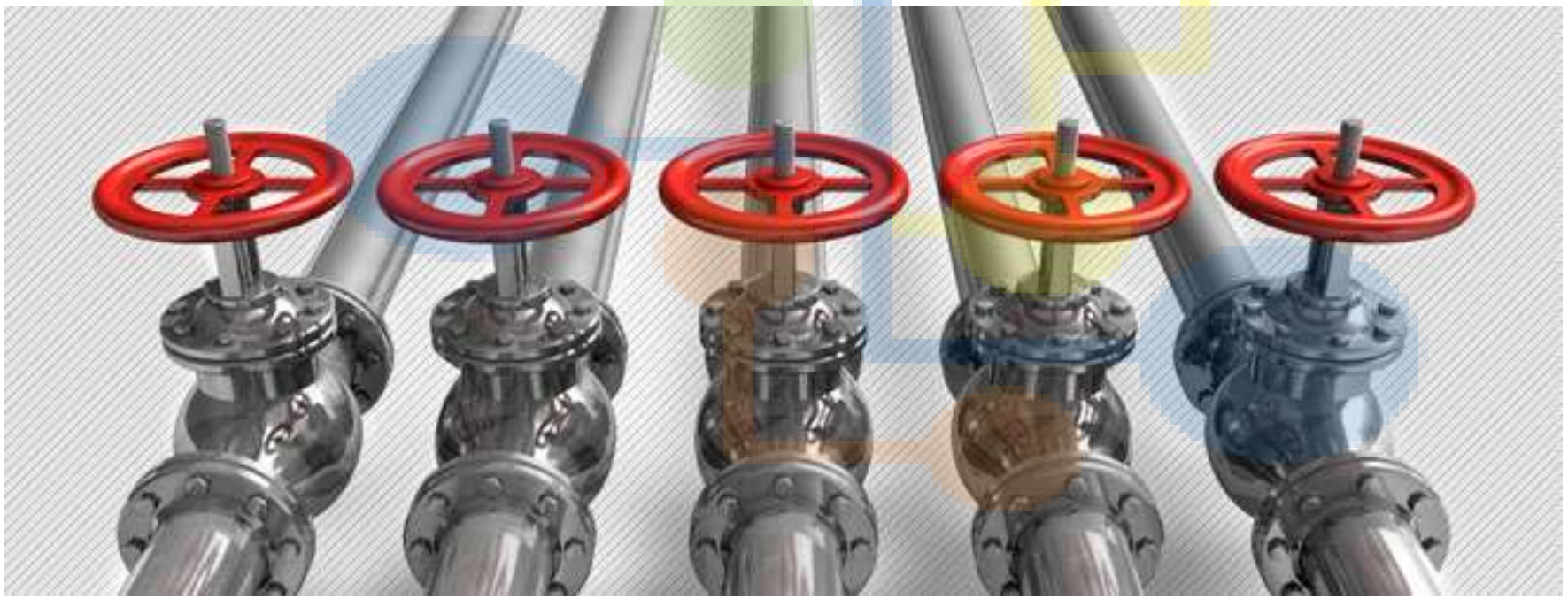


O Que São Ações?

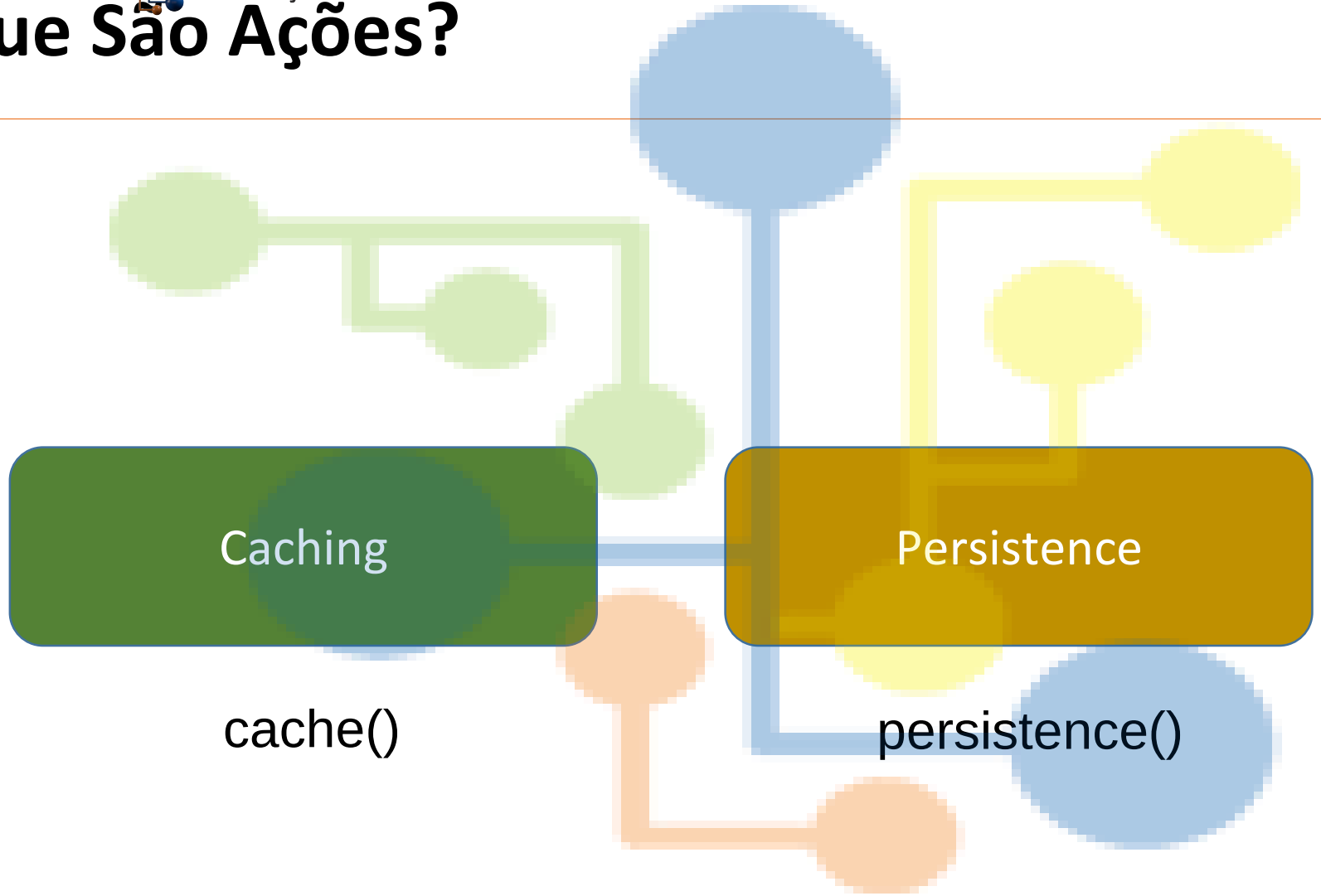
Ações são operações síncronas mas podemos usar a função **AsyncRDDActions()** para tornar as operações assíncronas.

O Que São Ações?

Podemos pensar nas ações como válvulas. Os dados estão prontos para serem processados e operações de transformação já foram definidas, mas somente quando abrirmos as válvulas, ou seja, executarmos as ações, o processamento será realmente iniciado.

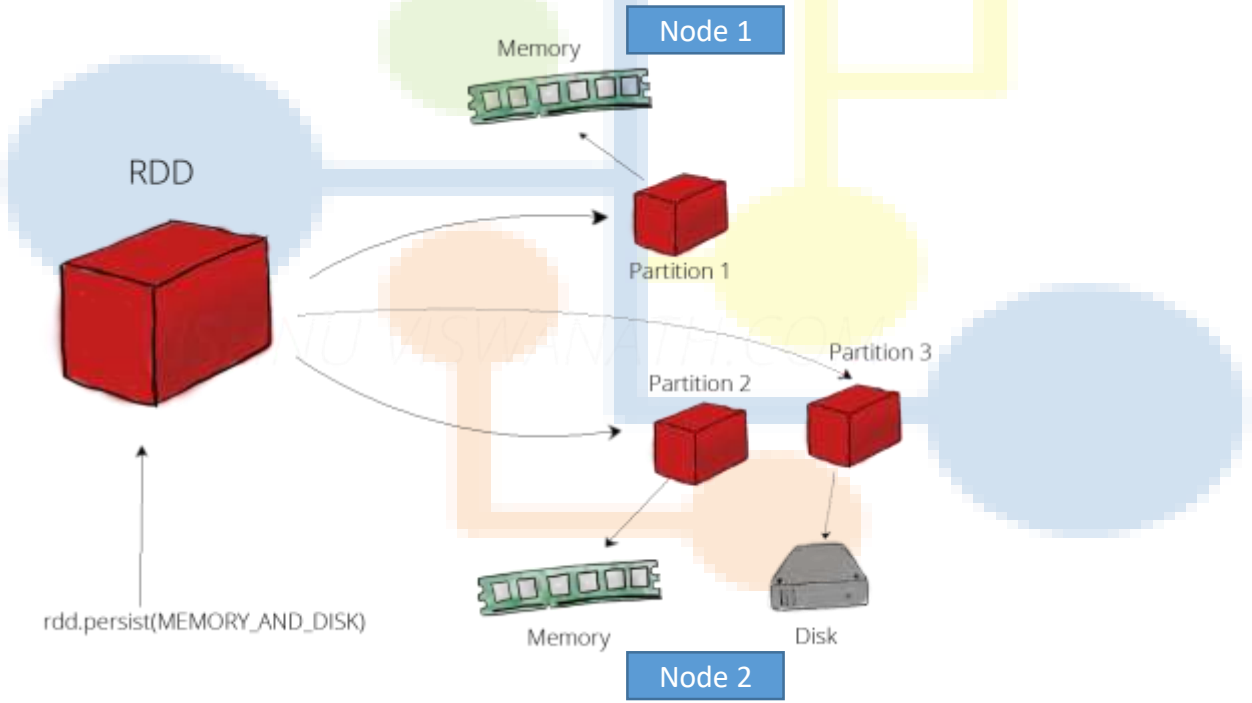


O Que São Ações?



O Que São Ações?

Nós devemos colocar os RDD's em cache, sempre que for necessário executar duas ou mais Ações no conjunto de dados. Isso melhora a performance.



O Que São Ações?

Lista com Todas as Operações de Ação:

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

Tenha uma Excelente Jornada de Aprendizagem.

Muito Obrigado por Participar!

Equipe Data Science Academy