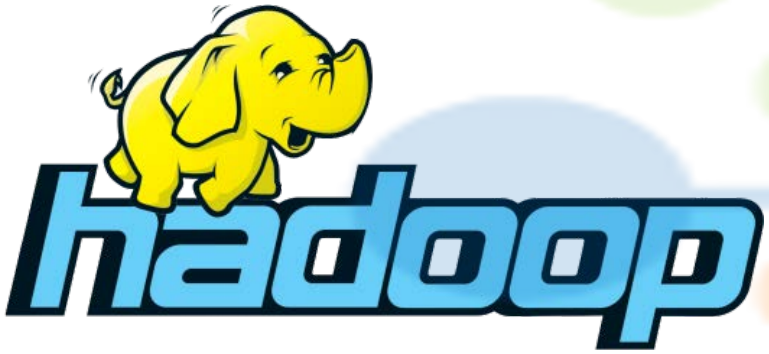
Engenharia de Dados com Hadoop e Spark

Data Science Academy rodrigo.c.abreu@hotmail.com 5e207d48e32fc335fa60447d

Bem-vindo(a)

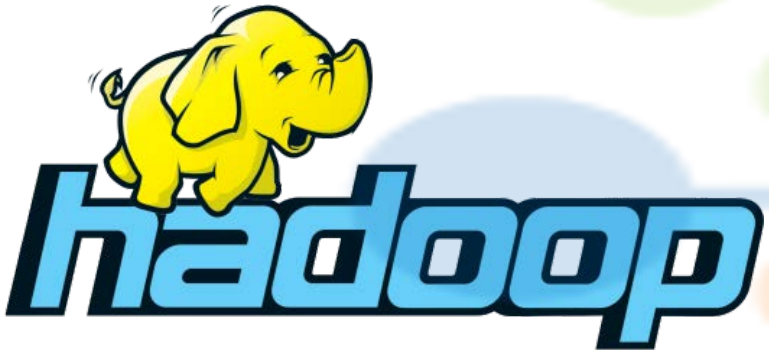
Hadoop x Spark

Hadoop x Spark



Framework para desenvolvimento de aplicações distribuídas

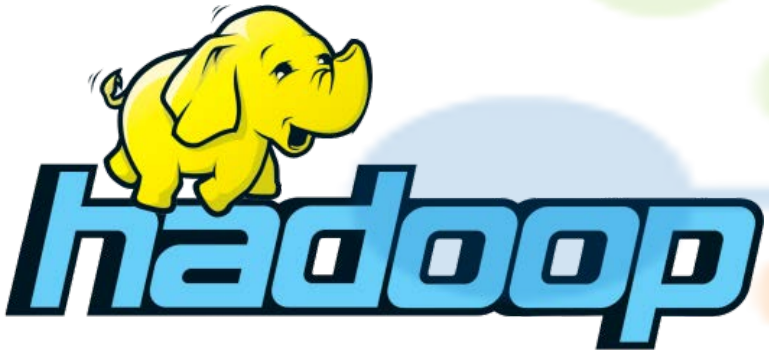
Hadoop x Spark



HDFS
Sistema de Arquivos Distribuído

Data Science Academy rodrigo.c.abreu@hotmail.com 5e207d48e32fc335fa60447d

Hadoop x Spark



MapReduce
Processamento Distribuído

Data Science Academy rodrigo.c.abreu@hotmail.com 5e207d48e32fc335fa60447d

Hadoop x Spark

Framework para processamento de Big Data

Hadoop x Spark

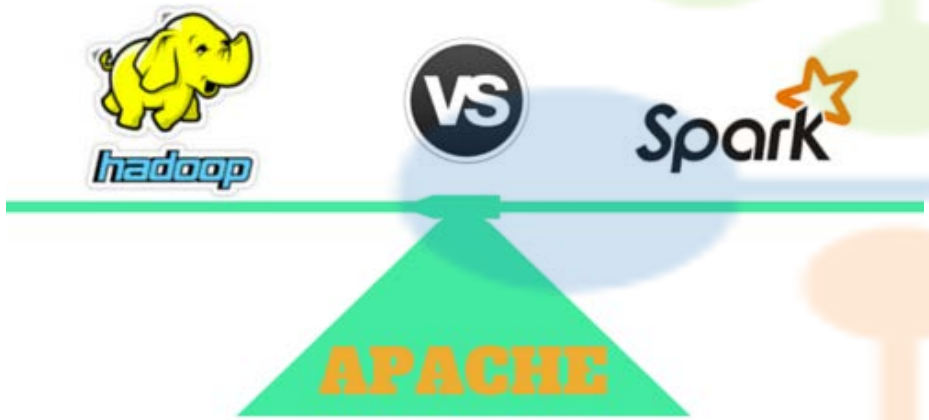
Se Hadoop e Spark são produtos diferentes, com propósitos diferentes e operam de formas diferentes, por que são frequentemente comparados?

Hadoop x Spark

Comparamos o Apache Spark com o Apache MapReduce.

Mas o Apache Spark não possui um sistema de armazenamento, podendo usar, por exemplo, o HDFS.

Hadoop x Spark



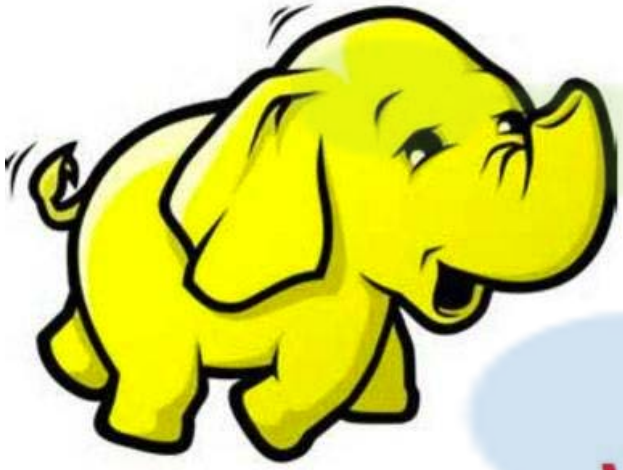
- Custo
- Velocidade
- Versatilidade
- Competências
- Fornecedores

Hadoop x Spark

Embora alguns departamentos de TI podem se sentir compelidos a escolher entre Hadoop e Spark, o fato é que provavelmente, muitas empresas usarão os dois, por serem tecnologias complementares.

Data Science Academy rodrigo.c.abreu@hotmail.com 5e207d48e32fc335fa60447d

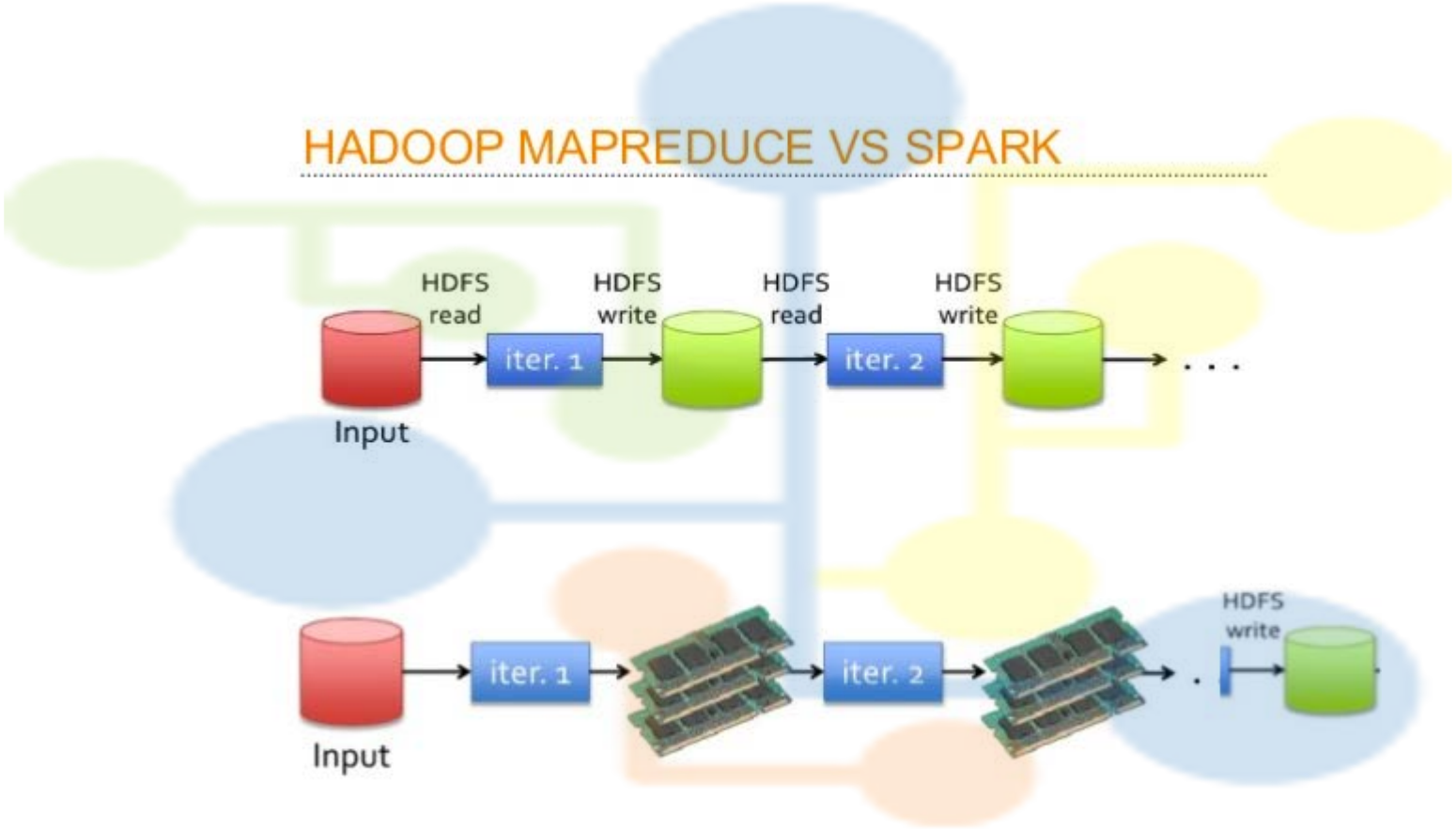
Hadoop x Spark



vs.

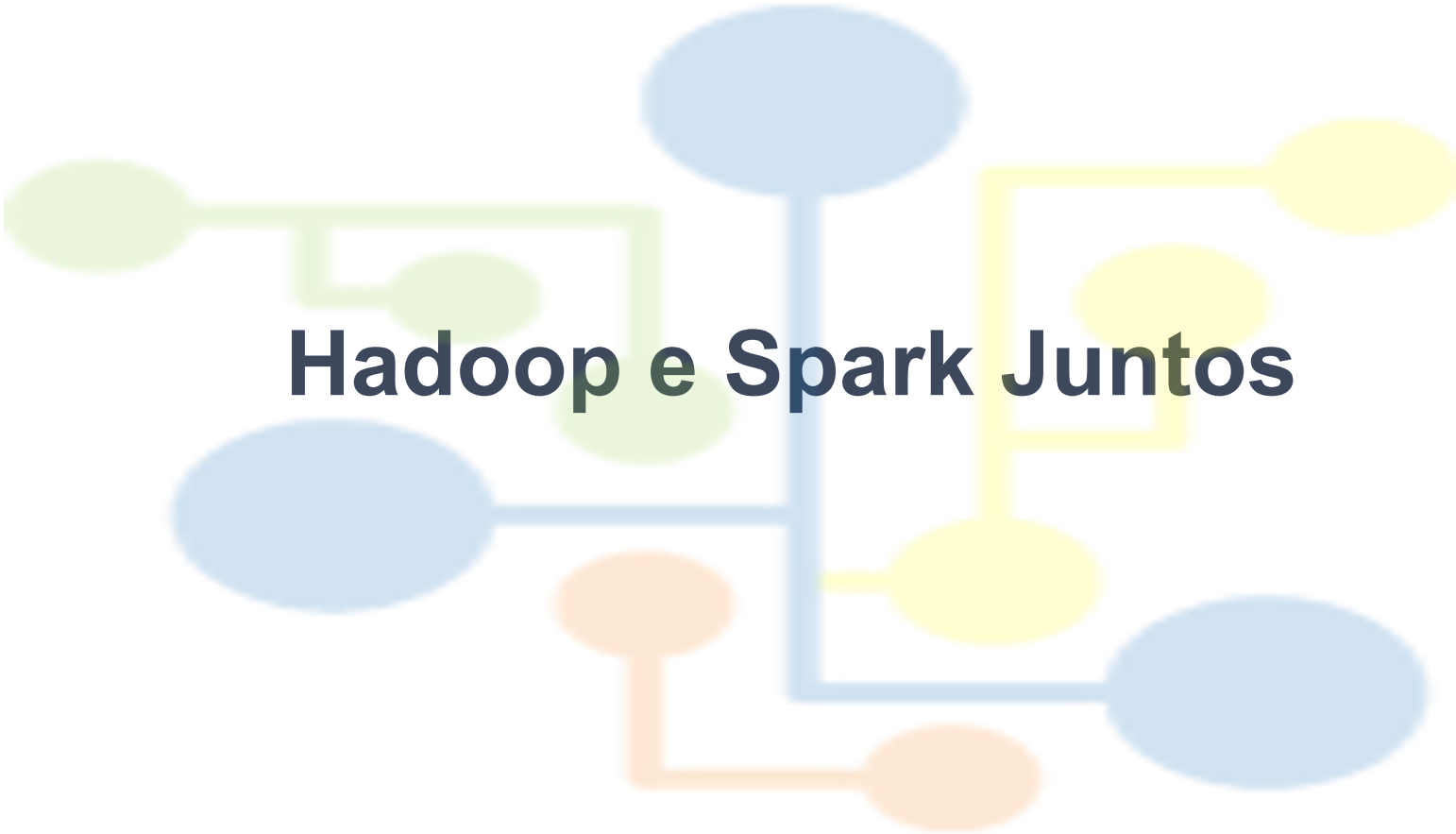
É possível usar um sem o outro!

Hadoop x Spark



Hadoop x Spark

Portanto, não existe escolha entre Hadoop e Spark e sim o objetivo do seu projeto.
Os 2 frameworks são complementares e podem ser usados em conjunto.



Hadoop e Spark Juntos

Hadoop e Spark

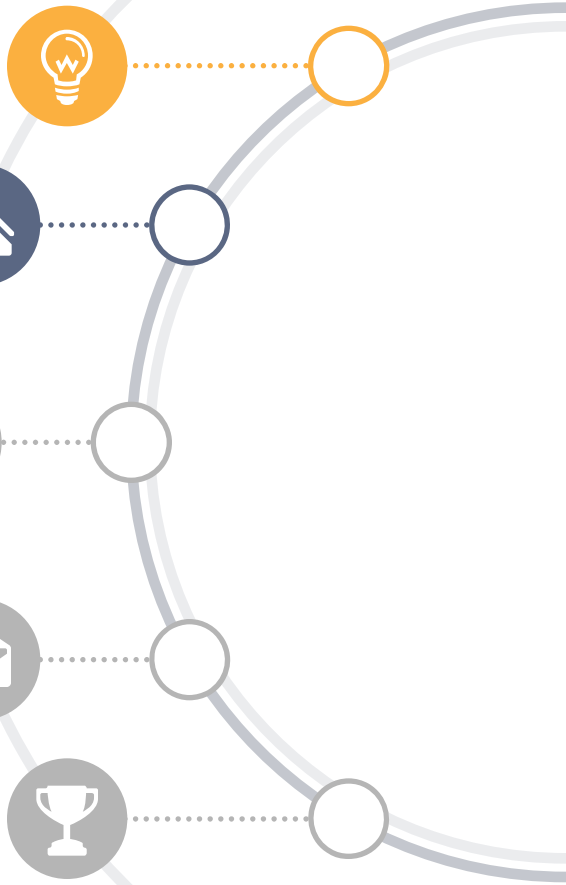
Hadoop e Spark fazem coisas diferentes

Você pode usar um sem o outro

O Spark é mais rápido

Mas você pode não precisar da velocidade do Spark

Mecanismos diferentes de recuperação a falhas



Hadoop e Spark

Data Science Academy rodrigo.c.abreu@hotmail.com 5e207d48e32fc335fa60447d

Spark é considerado o futuro do processamento distribuído no ecossistema Hadoop!

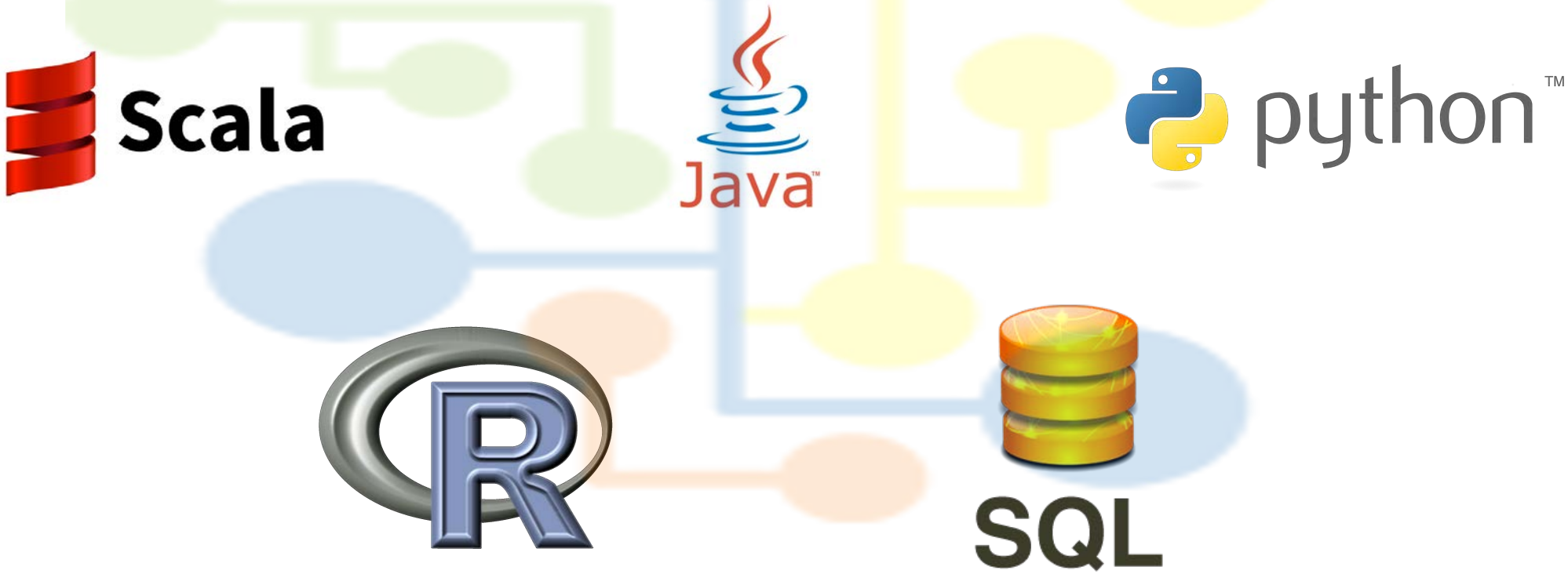
Hadoop e Spark

Com o Spark podemos realizar:

- Operações de ETL
- Análise Preditiva e Machine Learning
- Operações de Acesso a Dados com SQL
- Text Mining
- Processamento de Eventos em Tempo Real
- Aplicações Gráficas
- Reconhecimento de Padrões
- Sistemas de Recomendação

Hadoop e Spark

Embora seja escrito em Scala, o Spark suporta:



Hadoop e Spark

O Spark é normalmente utilizado com o HDFS, mas outros sistemas de arquivos ou sistemas de armazenamento podem ser usados, tais como:

- Sistema de arquivos local ou de rede (NFS)
- Amazon S3
- RDBMS
- NoSQL (Apache Cassandra, Hbase)
- Sistemas de Mensagens (Kafka)

Como o Spark Funciona Sobre o HDFS?

Data Science Academy rodrigo.c.abreu@hotmail.com 5e207d48e32fc335fa60447d

Hadoop e Spark

Para ***ler*** arquivos do HDFS com Spark usamos:

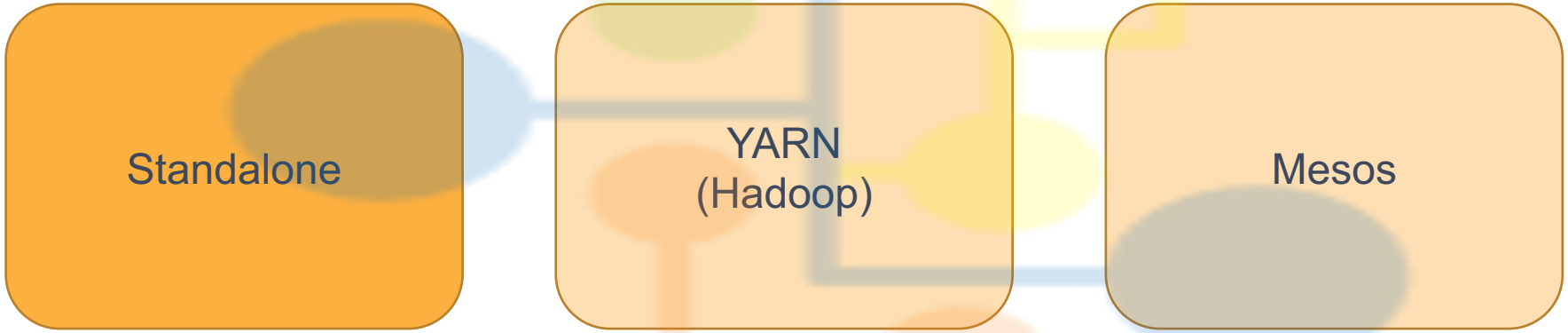
```
textfile = sc.textFile("hdfs://mycluster/data/file.txt")
```

Para ***gravar*** arquivos no HDFS com Spark usamos:

```
myRDD.saveAsTextFile("hdfs://mycluster/data/output.txt")
```

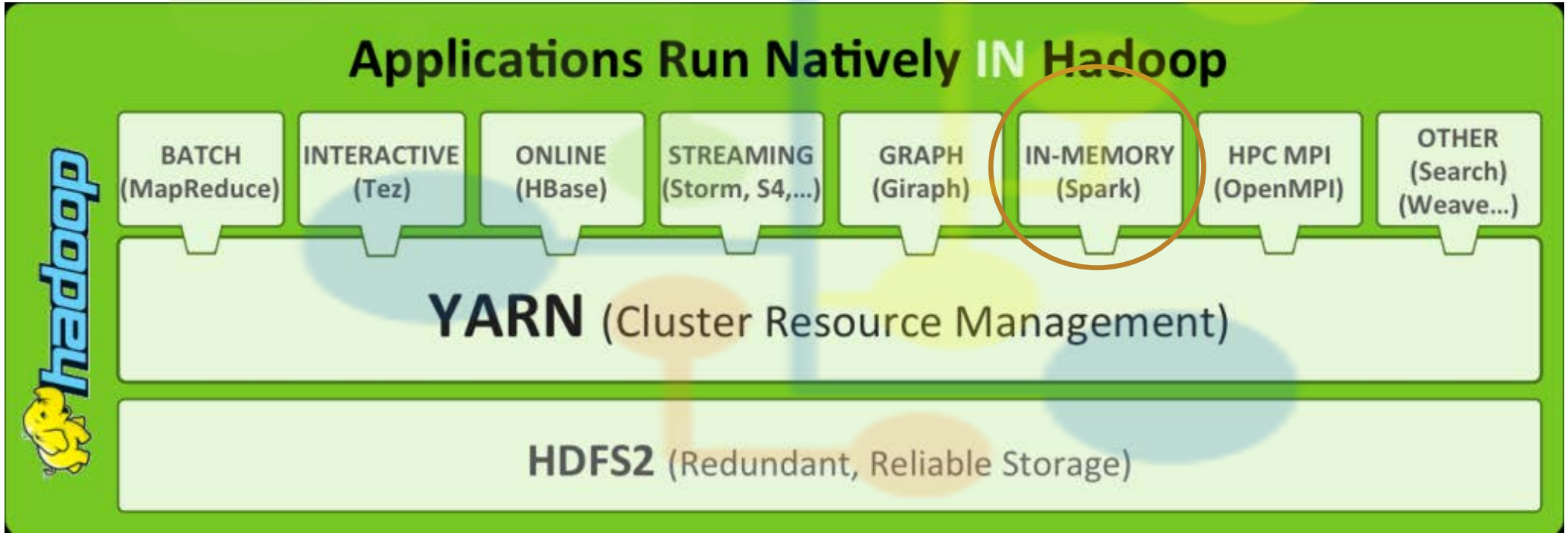
Instalação do Spark

Instalação do Spark



Hadoop e Spark

YARN e Spark



Anatomia de Uma Aplicação Spark

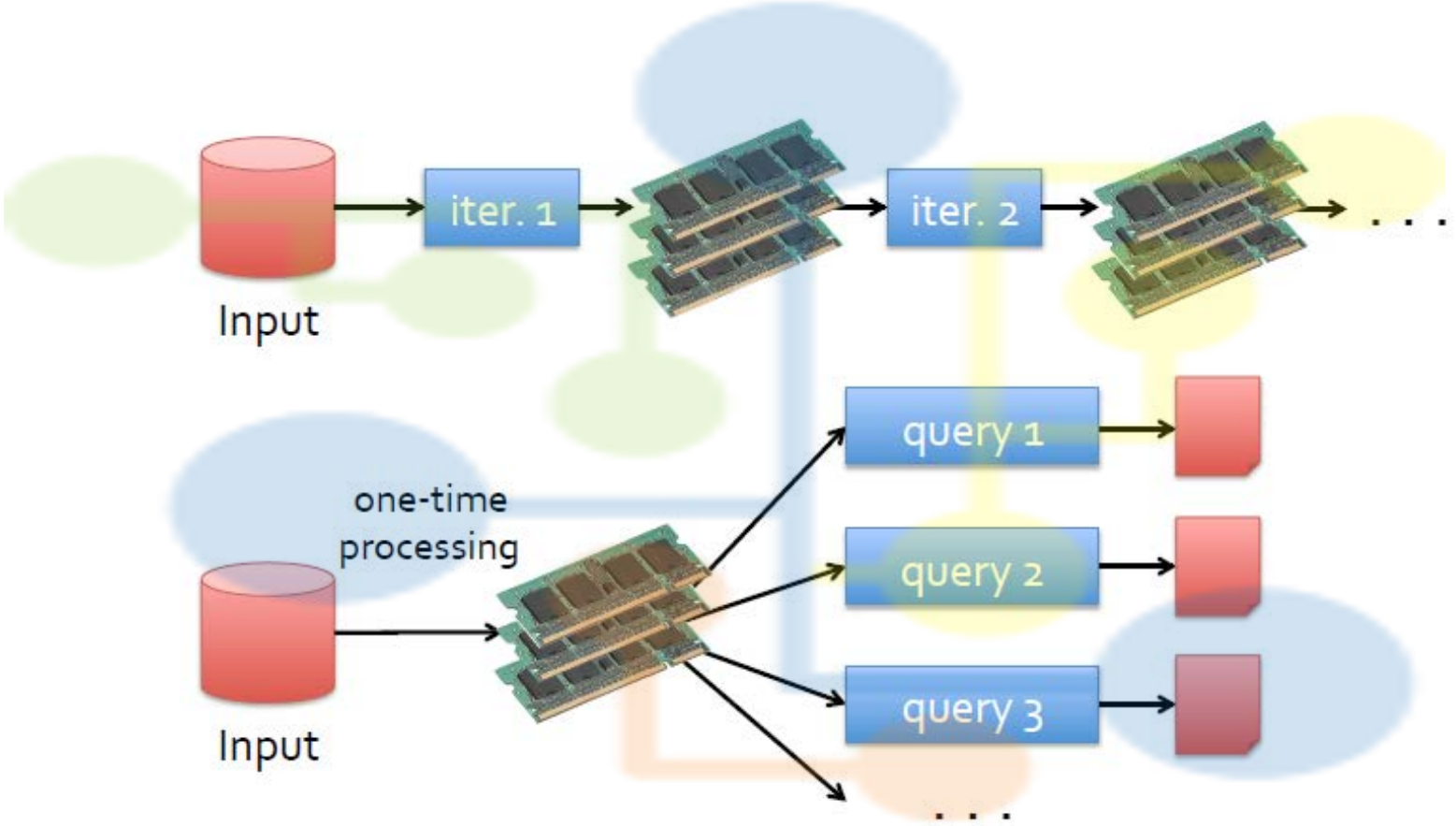
Anatomia de Uma Aplicação Spark

Se eu posso construir meu processo de análise com Python ou R, por exemplo, por que usaria o Spark?

1- Porque você precisa processar um grande volume de dados.

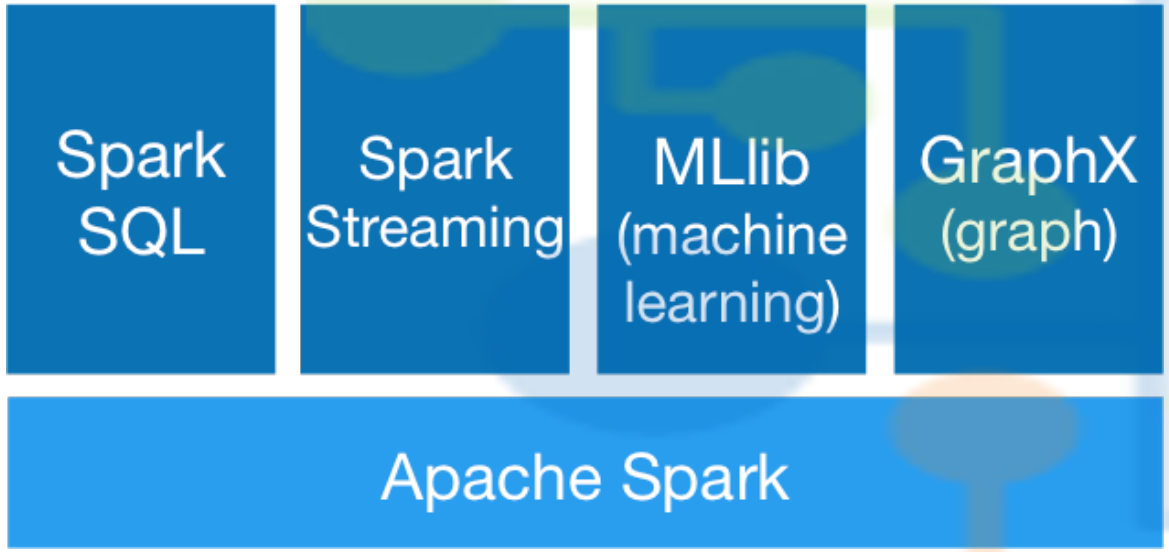
2- Porque você quer usar uma das APIs prontas do Spark, como SQL ou Streaming, por exemplo!

Anatomia de Uma Aplicação Spark



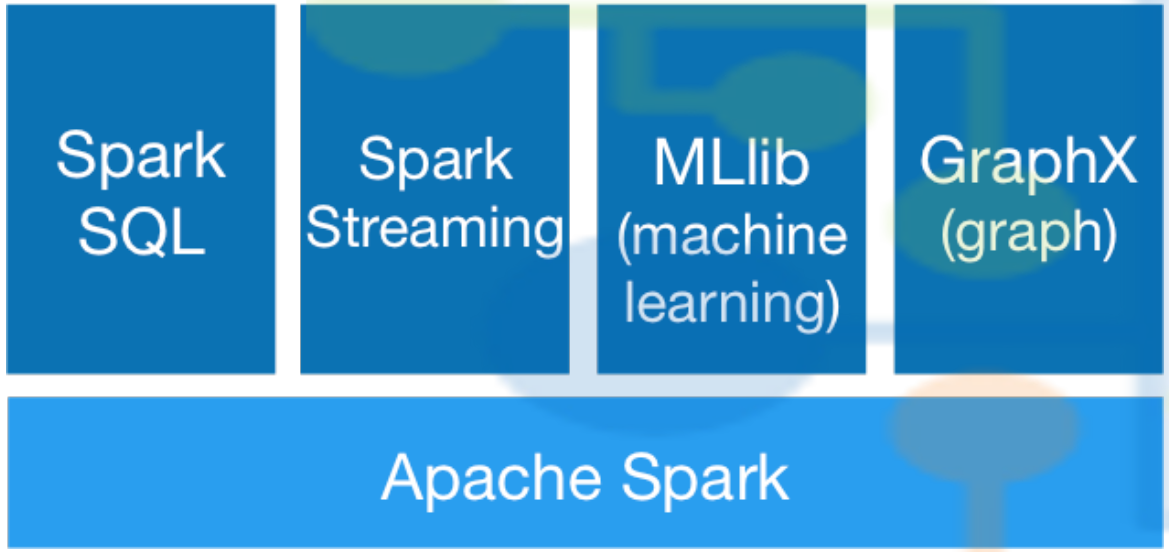
Anatomia de Uma Aplicação Spark

Anatomia de Uma Aplicação Spark



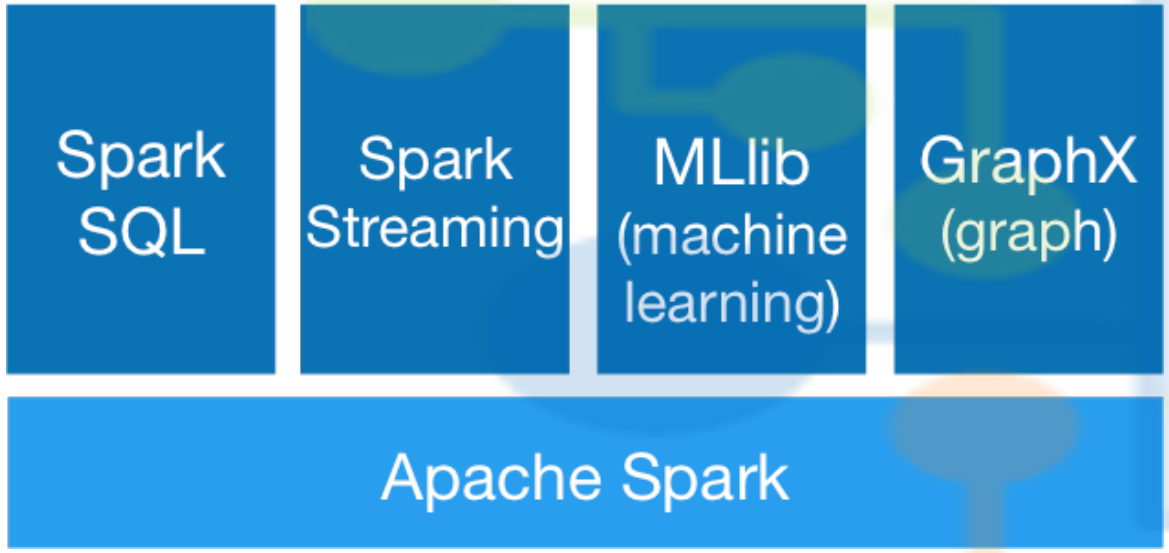
Spark SQL

Anatomia de Uma Aplicação Spark



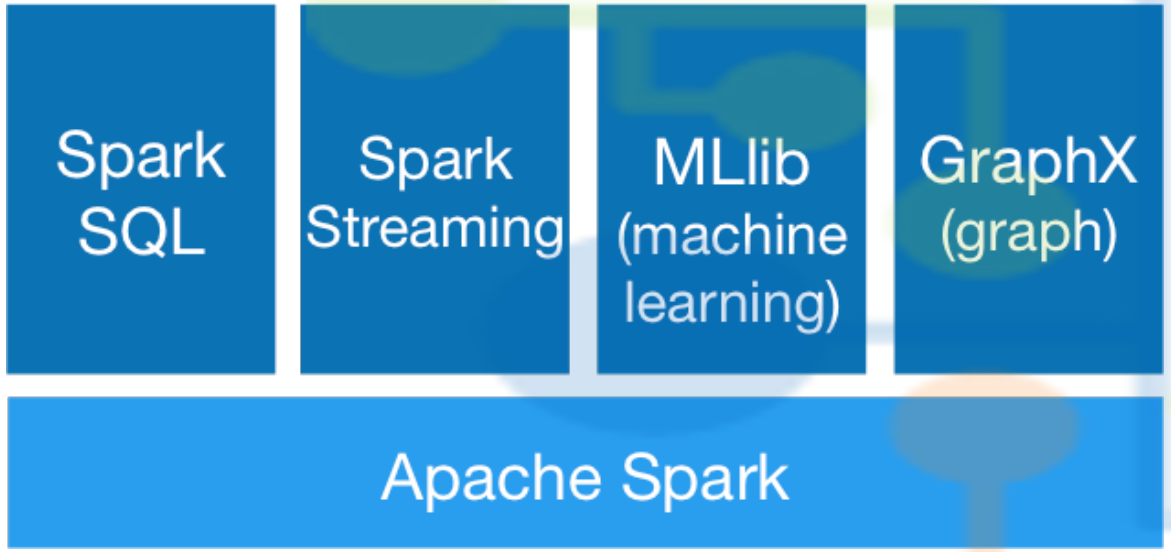
Spark Streaming

Anatomia de Uma Aplicação Spark



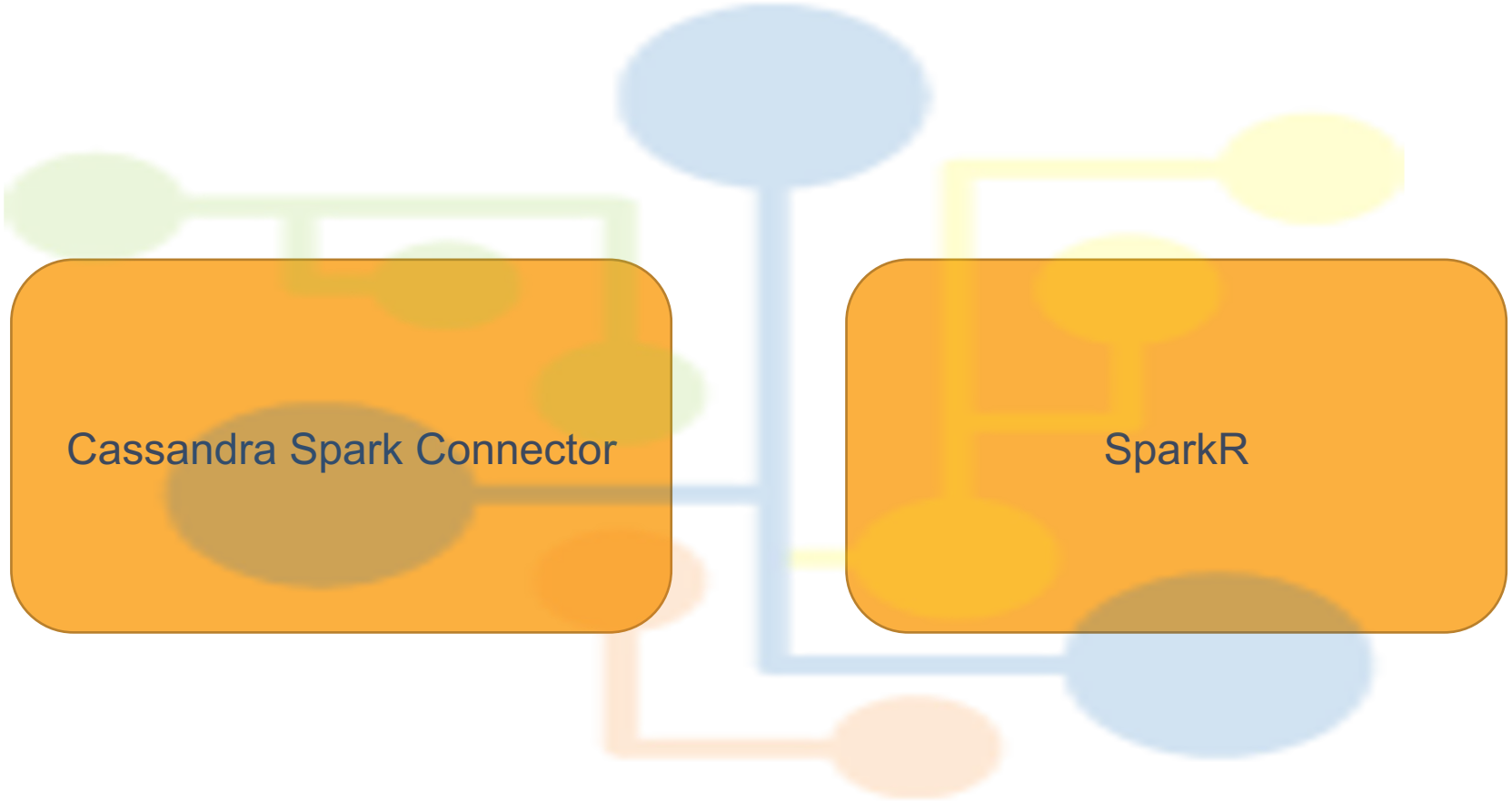
Spark MLLib

Anatomia de Uma Aplicação Spark



Spark GraphX

Anatomia de Uma Aplicação Spark

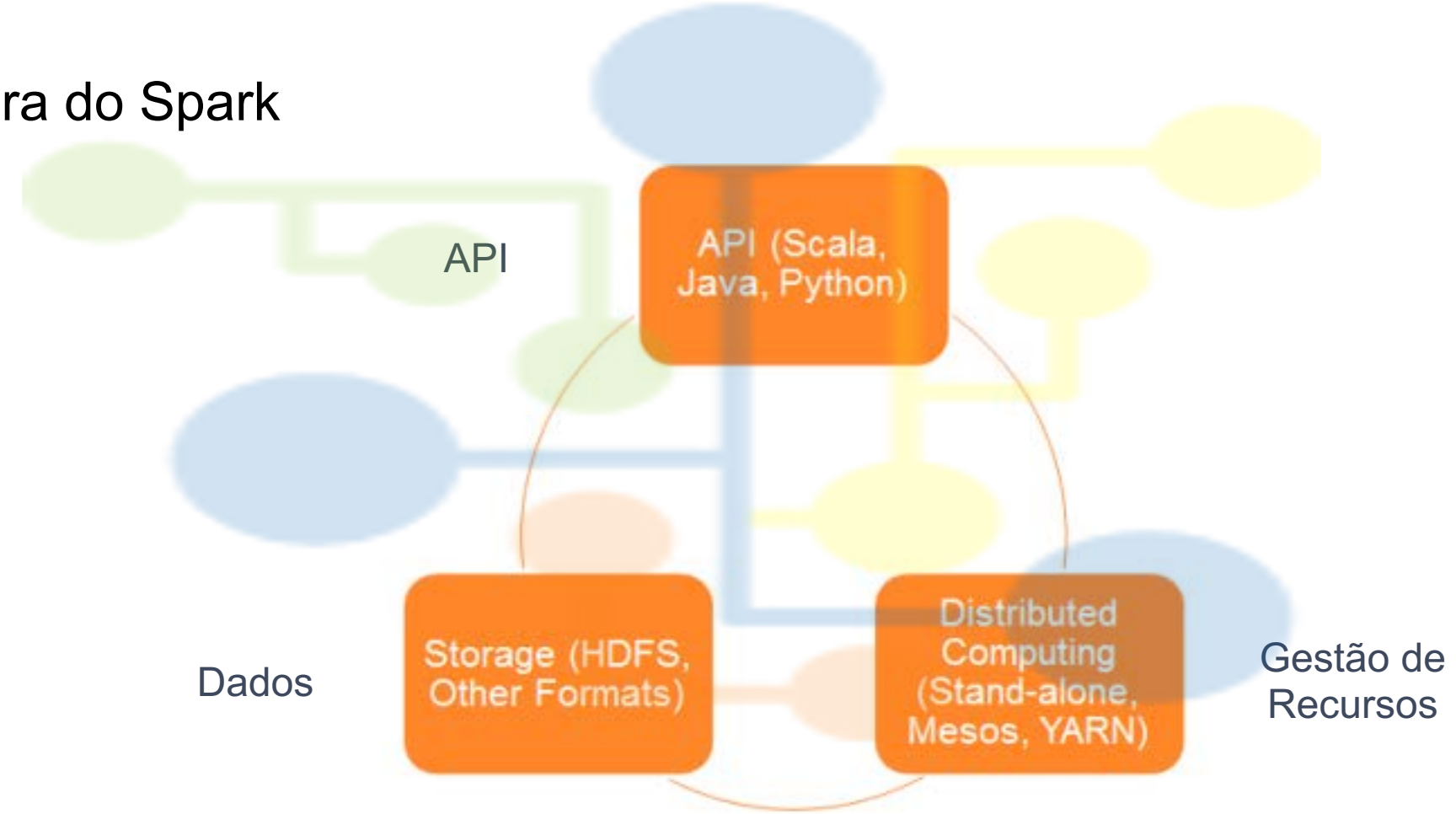


Anatomia de Uma Aplicação Spark

Arquitetura do Spark

Anatomia de Uma Aplicação Spark

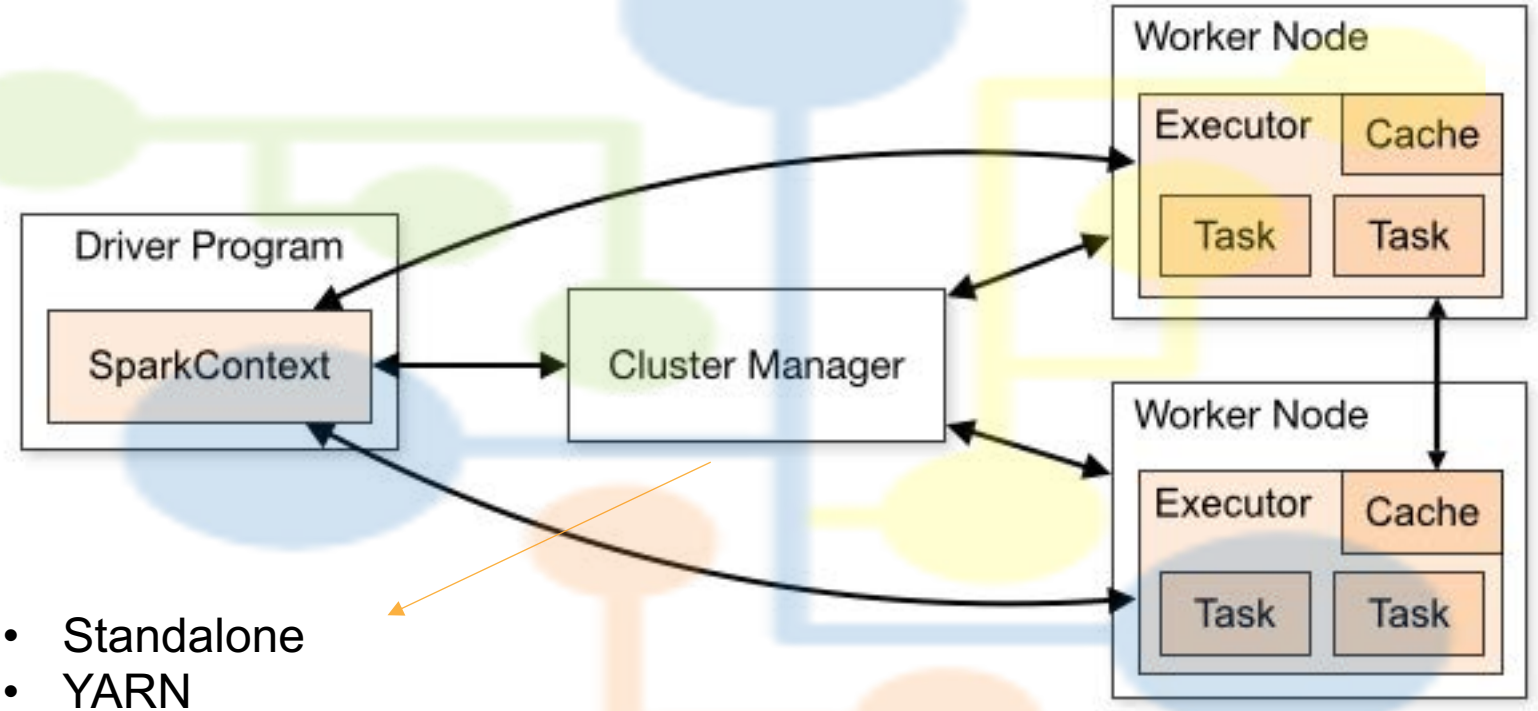
Arquitetura do Spark



Anatomia de Uma Aplicação Spark

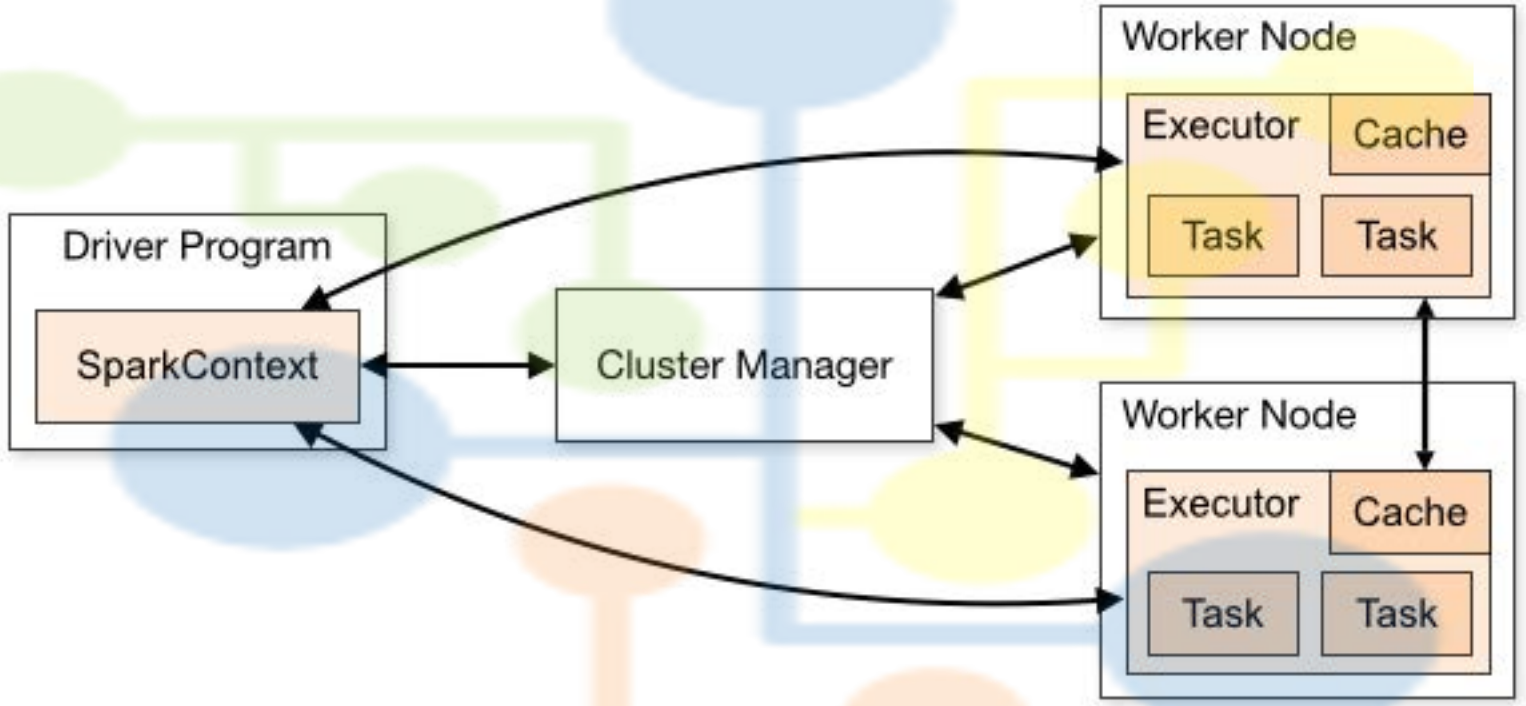
Processamento de Uma Aplicação Spark

Anatomia de Uma Aplicação Spark



- Standalone
- YARN
- Mesos

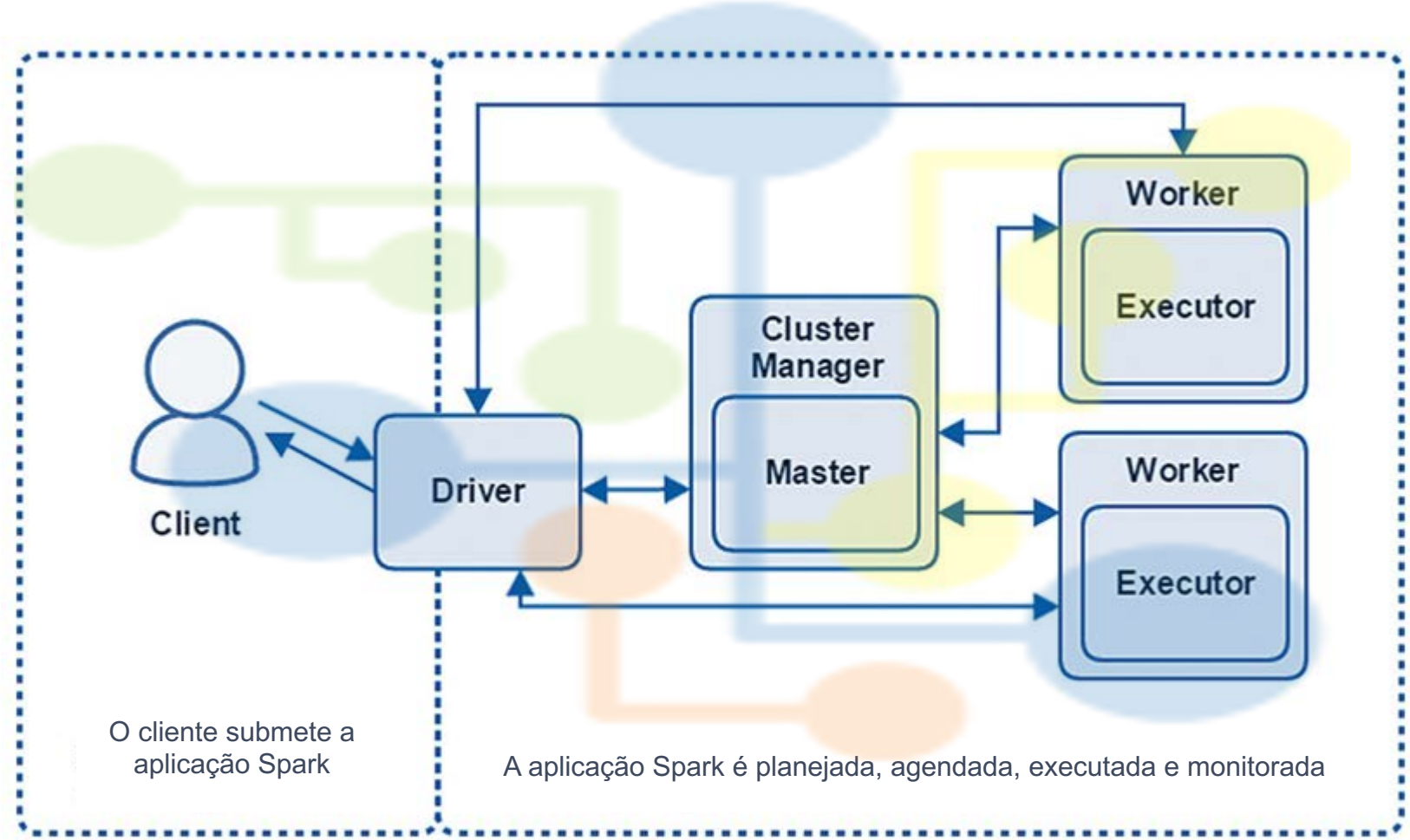
Anatomia de Uma Aplicação Spark



O cliente submete a aplicação Spark

A aplicação Spark é planejada, agendada, executada e monitorada

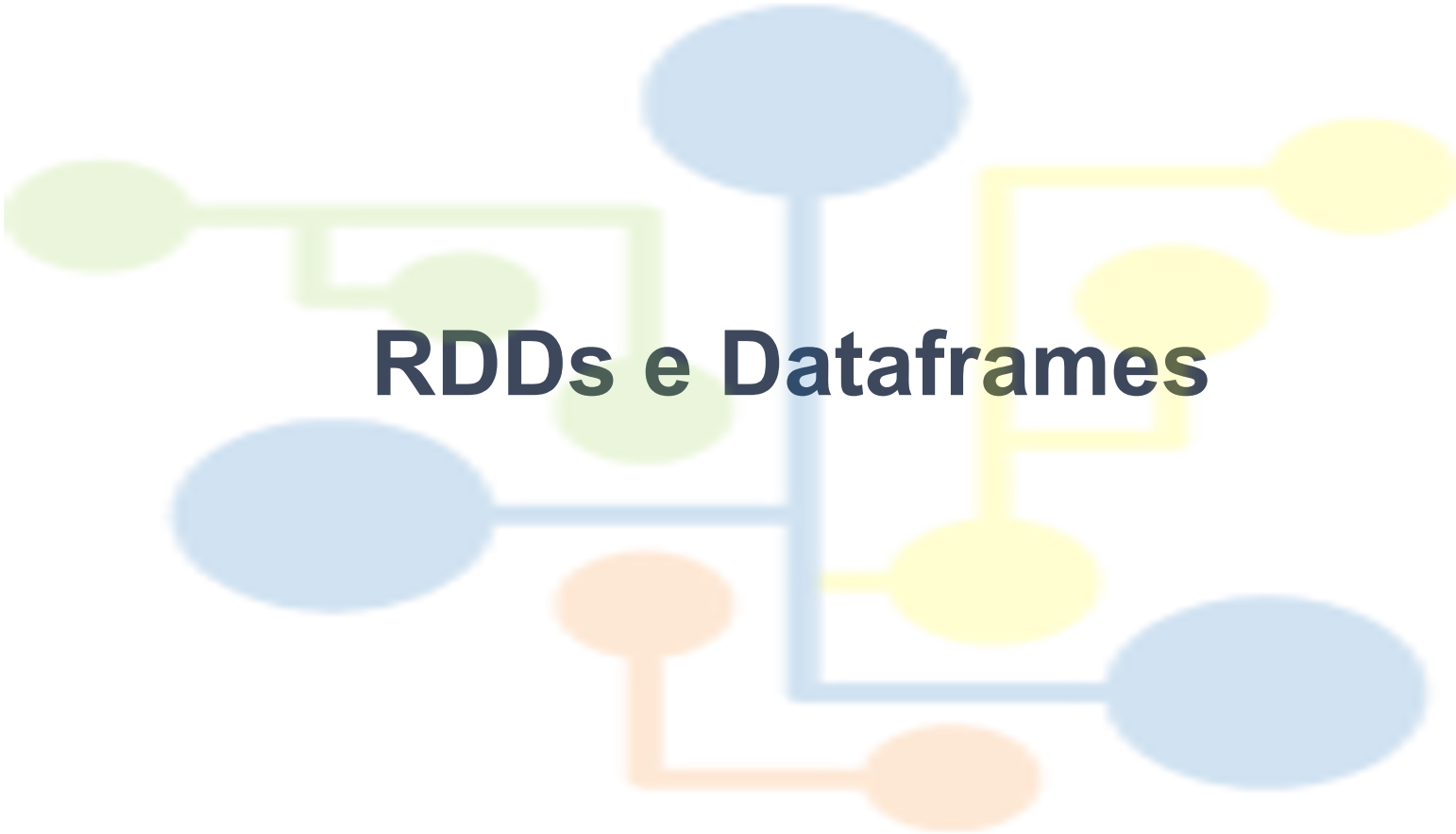
Anatomia de Uma Aplicação Spark



Anatomia de Uma Aplicação Spark

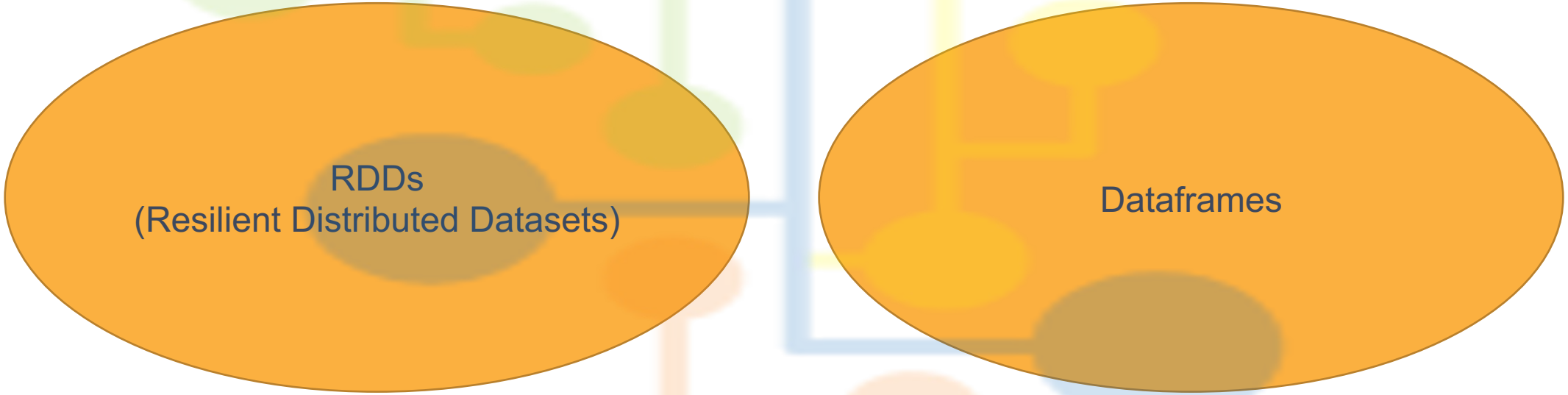
Outras Características do Spark:

- Suporta mais do que apenas as funções de Map e Reduce
- Otimiza o uso de operadores de grafos arbitrários
- Avaliação sob demanda de consultas de Big Data contribui com otimização do fluxo global do processamento de dados
- Fornece APIs concisas e consistentes em Scala, Java e Python
- Oferece shell interativo para Scala, Python e R. O shell ainda não está disponível em Java

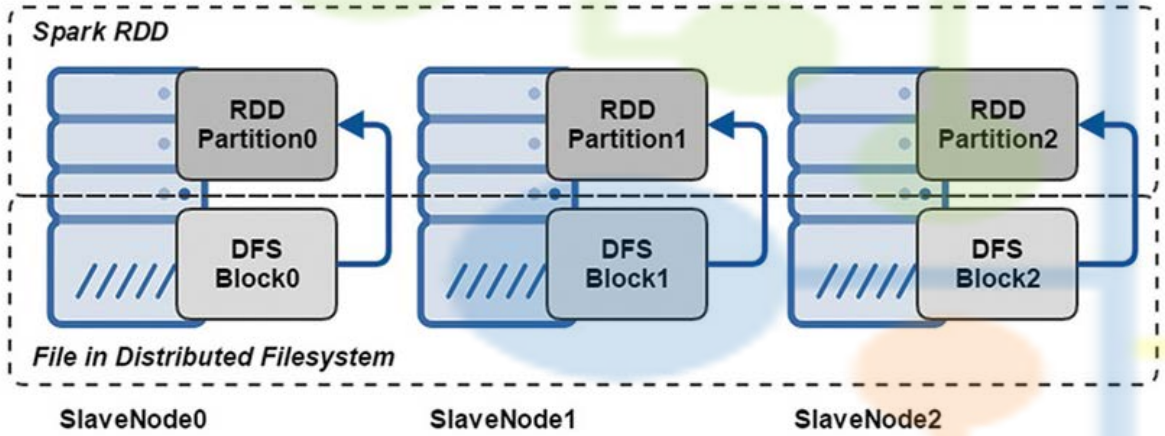


RDDs e Dataframes

RDD's e Dataframes

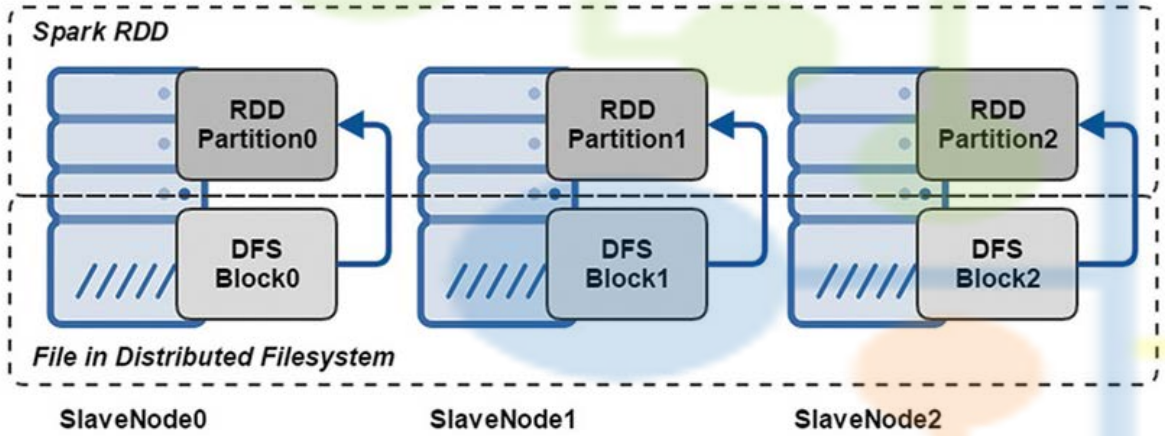


RDD's e Dataframes



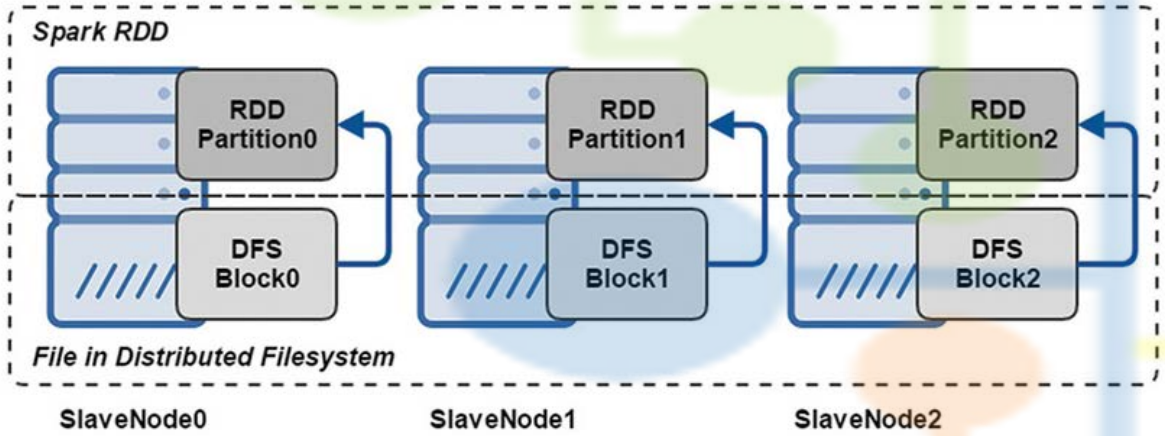
RDD é uma coleção de objetos distribuída e imutável. Cada conjunto de dados no RDD é dividido em partições lógicas, que podem ser computadas em diferentes nodes do cluster.

RDD's e Dataframes



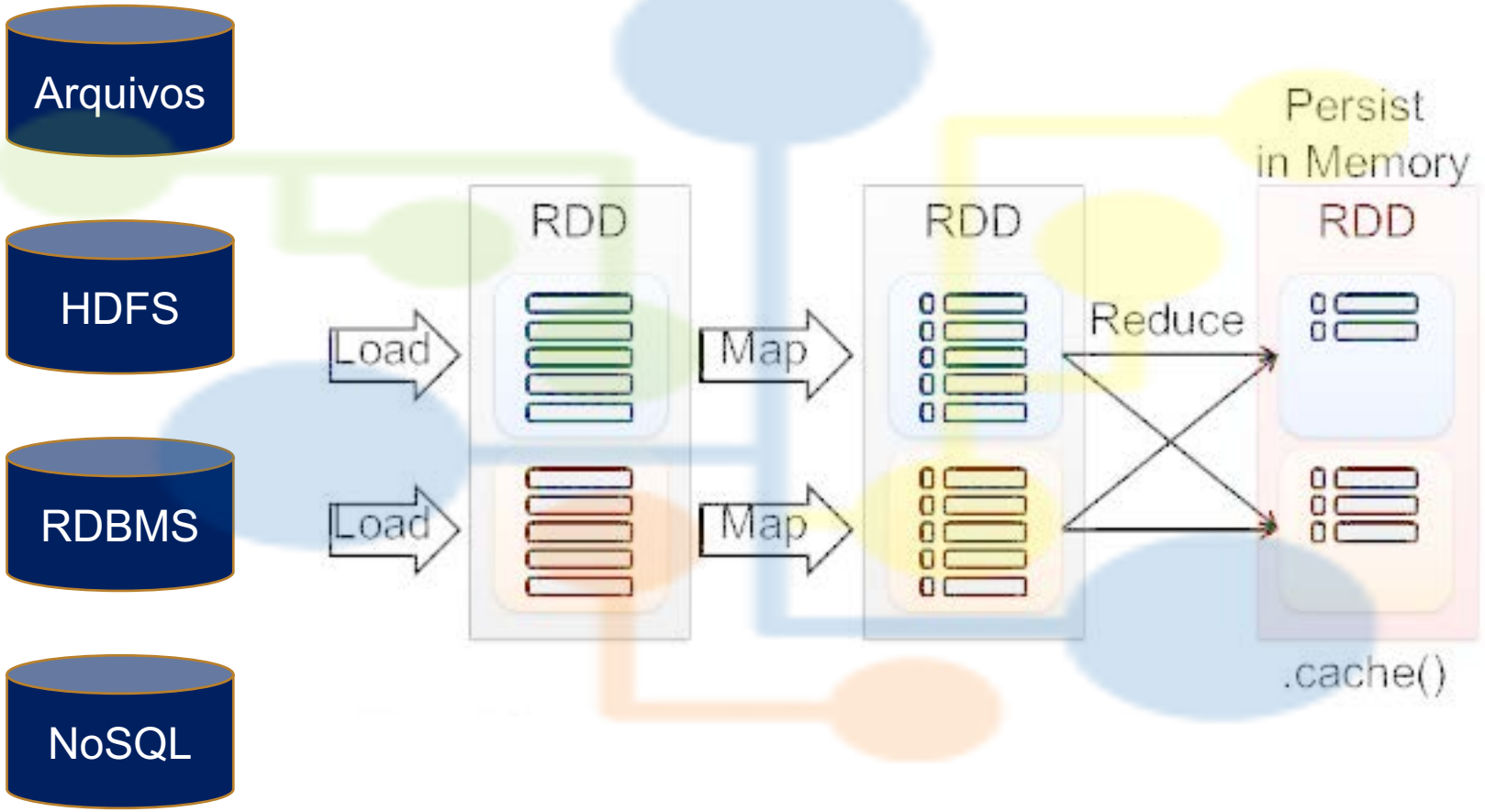
RDD é Conceito Central do Framework Spark!

RDD's e Dataframes



RDD's são imutáveis!

RDD's e Dataframes



RDD's e Dataframes

Existem 2 formas de criar um RDD:

Paralelizando uma coleção existente
(função sc.parallelize)

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

RDD's e Dataframes

Hadoop MapReduce

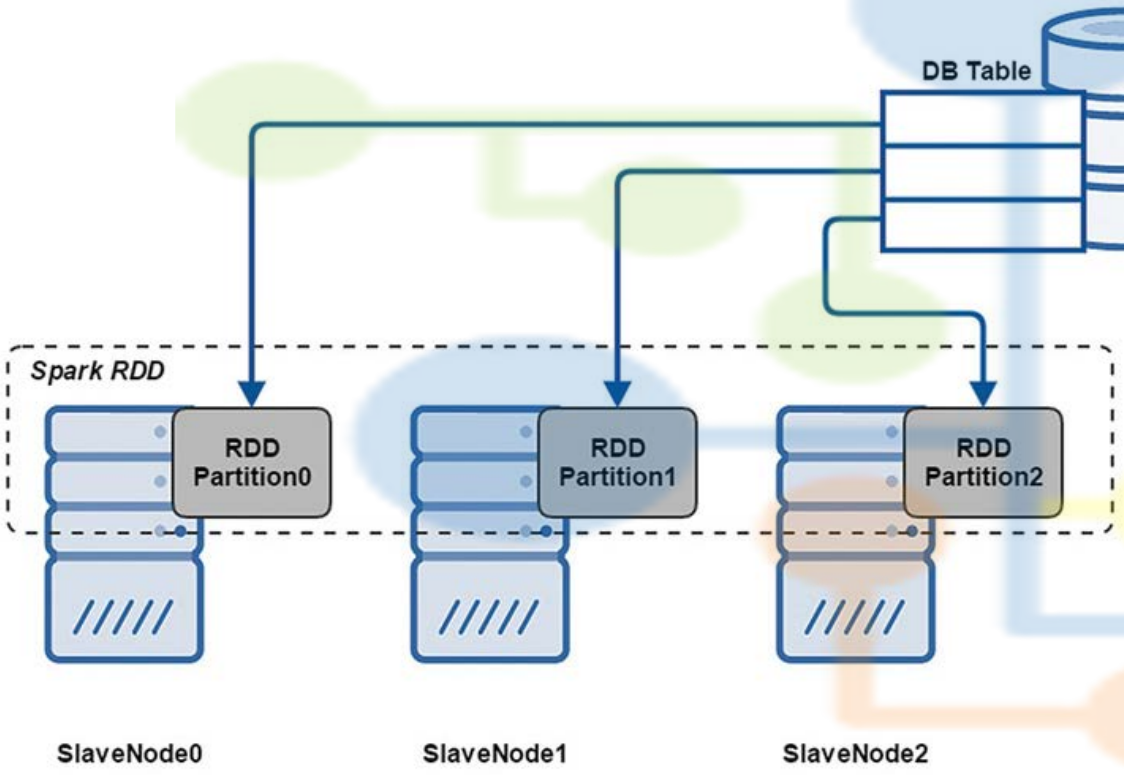
No Hadoop MapReduce (que não possui o conceito de RDD), cada resultado intermediário é gravado em disco. Ou seja, imagine um algoritmo de ML que precisa realizar diversas iterações nos dados. O disco será usado com muita frequência, deixando o processo mais lento.

RDD's e Dataframes

Spark

Com o conceito de RDD, o Spark armazena os resultados intermediários em memória, permitindo que operações iterativas que precisam acessar os dados diversas vezes, possam recorrer a memória do computador e não ao disco. Os dados serão gravados em disco apenas ao fim do processo ou se durante o processo, não houver memória disponível. Lembre que estamos falando aqui de cluster de computadores, com Terabytes de memória RAM quando se combina a memória de cada node do cluster.

RDD's e Dataframes



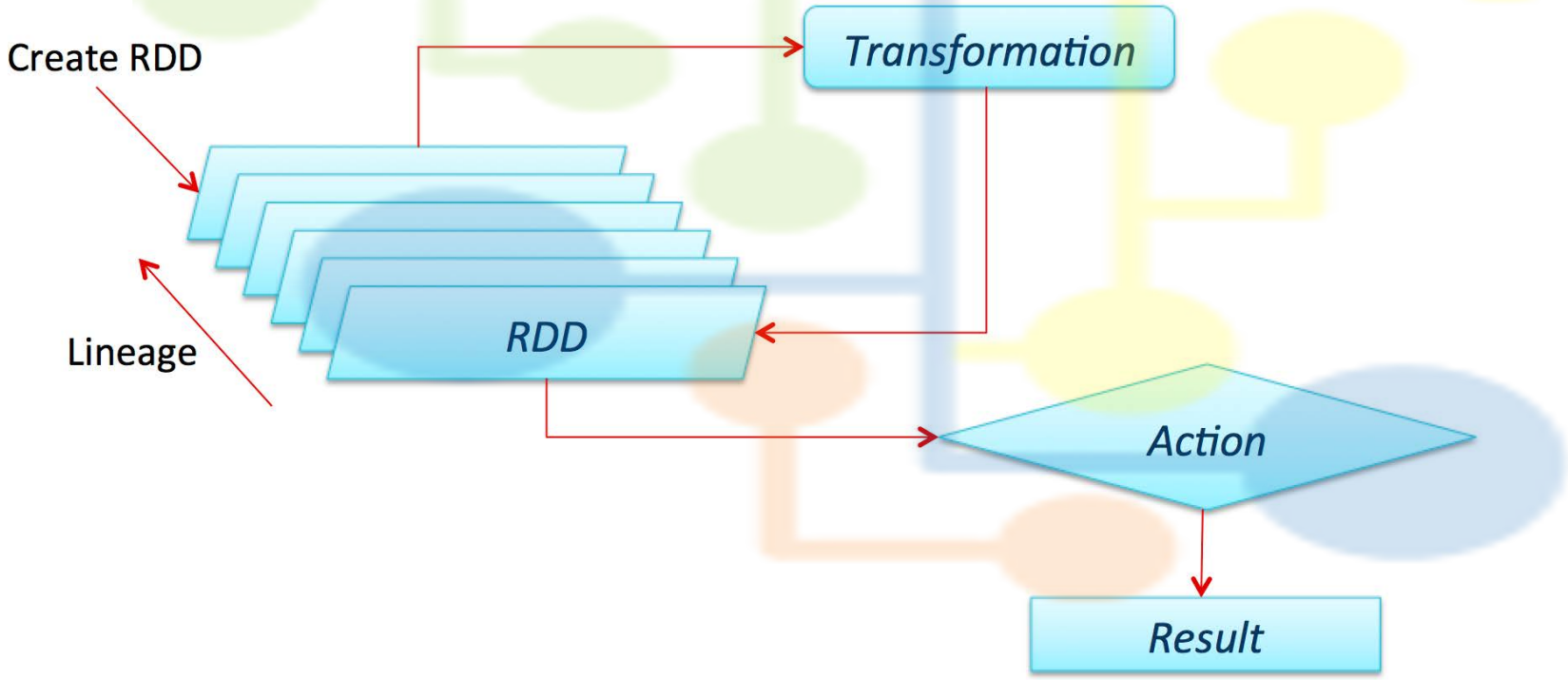
Os RDD's podem ser particionados e persistidos em memória ou disco!

RDD's e Dataframes

O RDD suporta dois tipos de operações:

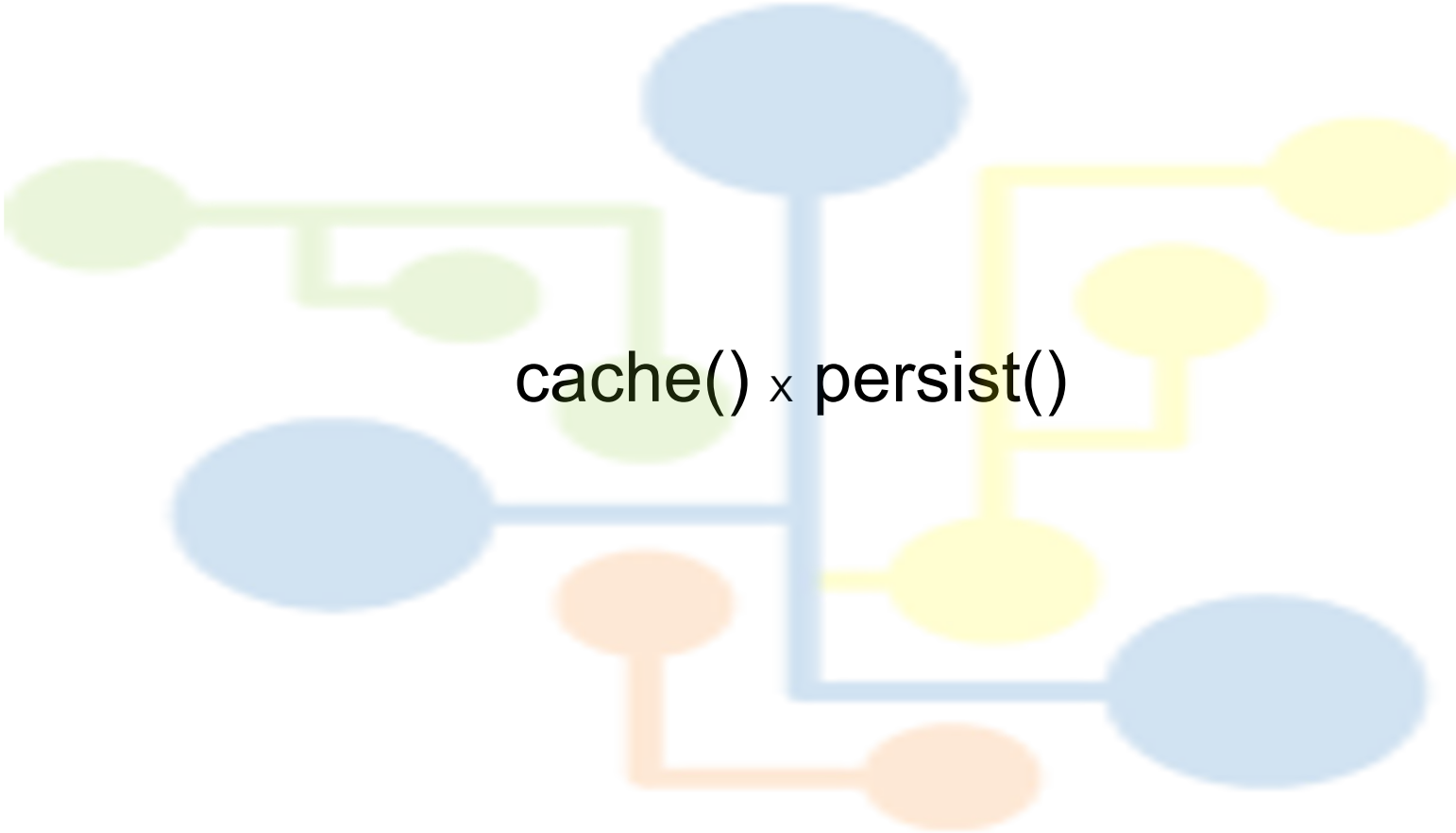
RDD's e Dataframes

Transformações e Ações



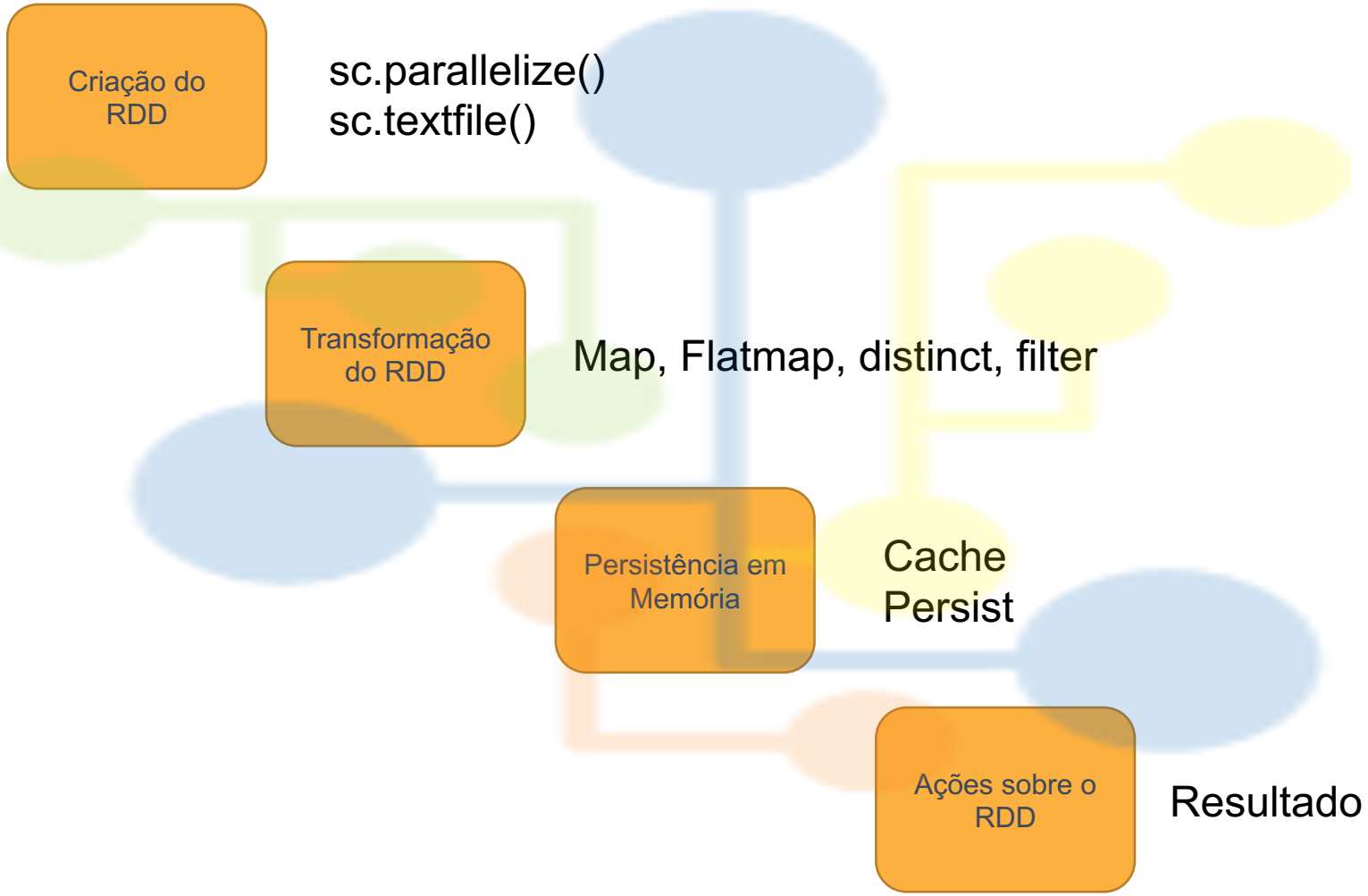
Lazy Evaluation

RDD's e Dataframes



cache() x persist()

RDD's e Dataframes

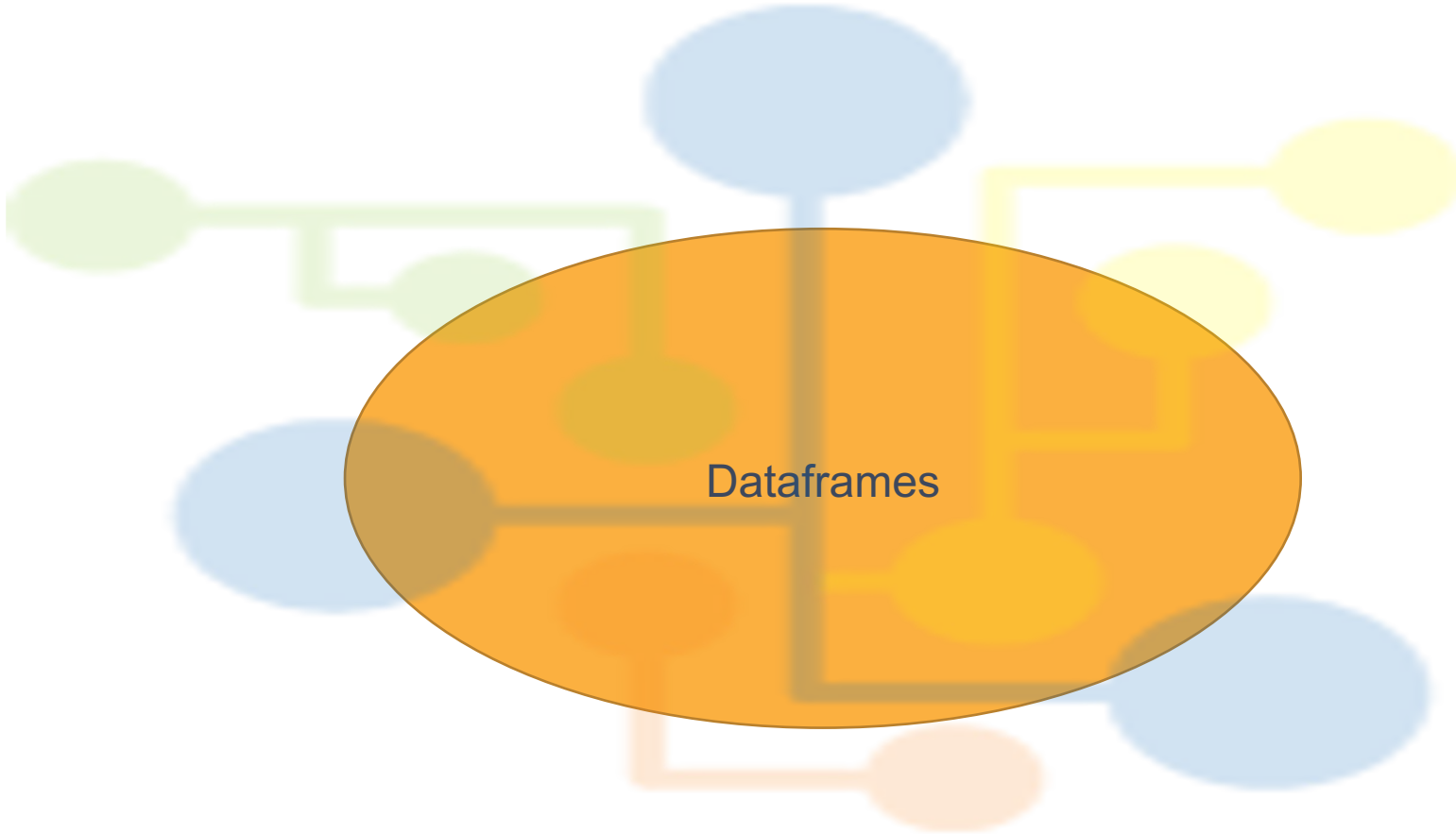


RDD's e Dataframes

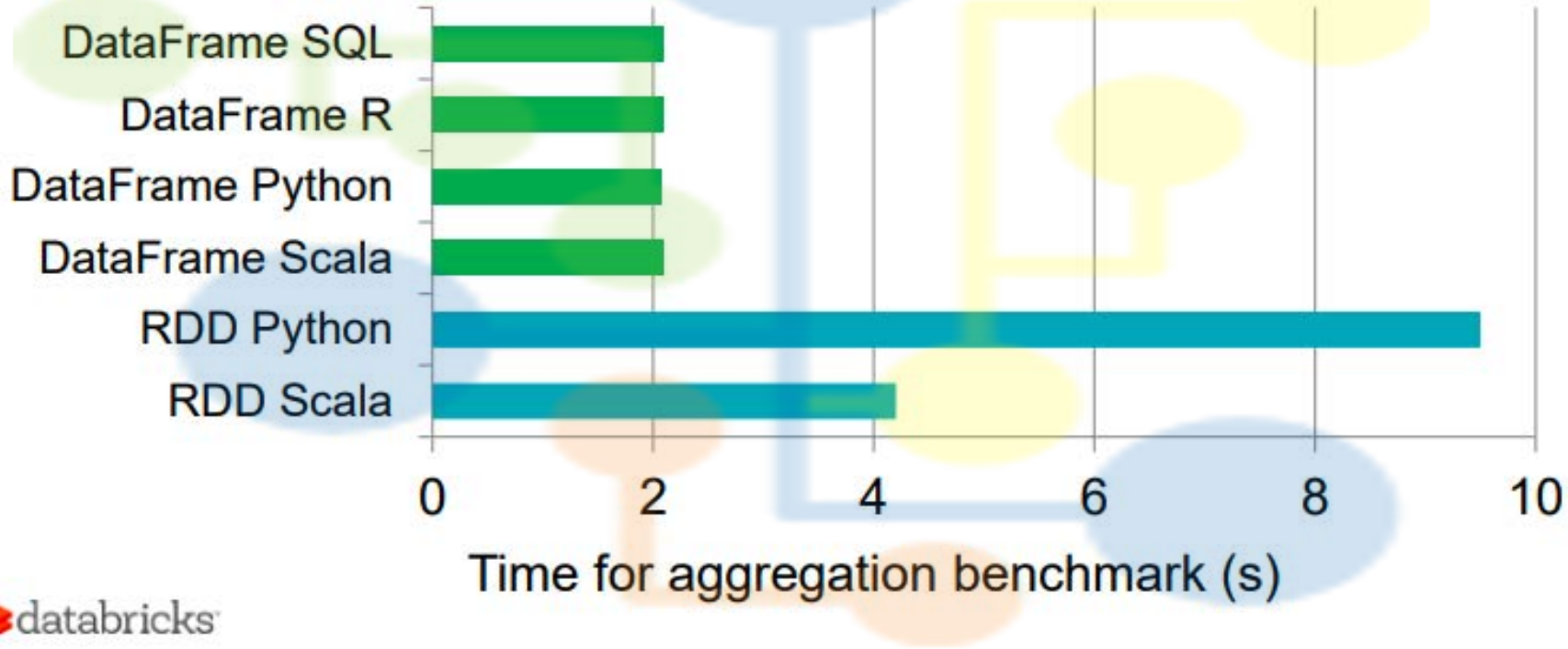
Quando usamos RDDs?

- Você deseja transformações e ações de baixo nível e controle no seu conjunto de dados. Seus dados não são estruturados, como fluxos de mídia ou de texto;
- Você deseja manipular seus dados com construções de programação funcional;
- Você não se preocupa em impor um esquema, como formato colunar, ao processar ou acessar atributos de dados por nome ou coluna;
- Você pode renunciar a alguns benefícios de otimização e desempenho disponíveis com Dataframes e Datasets para dados estruturados e semiestruturados.

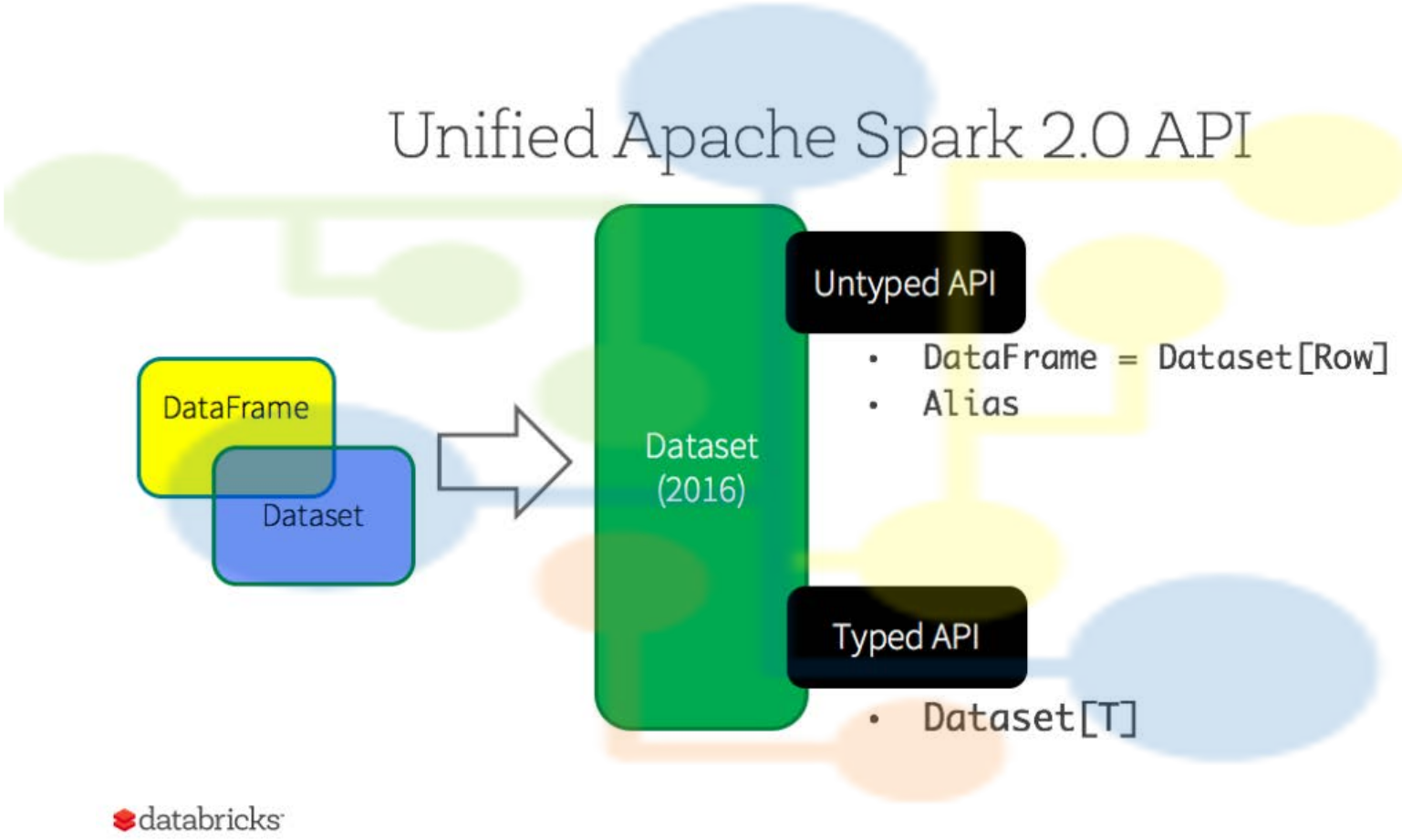
RDD's e Dataframes



RDD's e Dataframes



RDD's e Dataframes



RDD's e Dataframes

Language	Main Abstraction
Scala	Dataset[T] & DataFrame (alias for Dataset[Row])
Java	Dataset[T]
Python*	DataFrame
R*	DataFrame

RDD's e Dataframes

- Como um RDD, um Dataframe é uma coleção distribuída imutável de dados.
- Ao contrário de um RDD, os dados são organizados em colunas nomeadas, como uma tabela em um banco de dados relacional.
- Projetado para facilitar ainda mais o processamento de grandes conjuntos de dados, o Dataframe permite que os desenvolvedores imponham uma estrutura em uma coleção distribuída de dados, permitindo abstração de nível superior.

RDD's e Dataframes

Quando usamos Dataframes?

- Se você deseja semântica rica, abstrações de alto nível e APIs específicas do domínio, use Dataframe ou Dataset.
- Se o seu processamento exigir expressões de alto nível, filtros, mapas, agregação, médias, soma, consultas SQL, acesso colunar e uso de funções lambda em dados semiestruturados, use Dataframe ou Dataset.
- Se você deseja unificação e simplificação de APIs nas bibliotecas Spark, use Dataframe ou Dataset.
- Se você é um usuário R, use Dataframes.
- Se você é um usuário Python, use Dataframes e recorra aos RDDs se precisar de mais controle.

RDD's e Dataframes

Em resumo, a escolha de quando usar RDD ou Dataframe e/ou Dataset parece óbvia. Enquanto o primeiro oferece funcionalidade e controle de baixo nível, o último permite visualização e estrutura personalizadas, oferece operações específicas de alto nível e domínio, economiza espaço e é executado em velocidades superiores.

Obrigado