



Tutor

Fernando Amaral



O que Vamos Estudar!

1. Introdução

2. Datawarehouse, Datalake e Delta Lake

3. Databricks e Criação de Clusters

4. Introdução ao Spark

5. Data Frames, Delta e Delta Lake

6. Gráficos e Dashboards

7. Machine Learning com Databricks

8. Koalas

9. Construindo um Data Lakehouse

Databricks

- Plataforma de Análise de Dados
- Na nuvem
- Permite engenharia e ciência de dados



Baseado em Nuvem como Platform as a Service (PaaS)

- Não possui infraestrutura na Nuvem
- Você deve escolher qual infraestrutura usar
- Por exemplo, se usar AWS:
 - EC2
 - S3



Spark

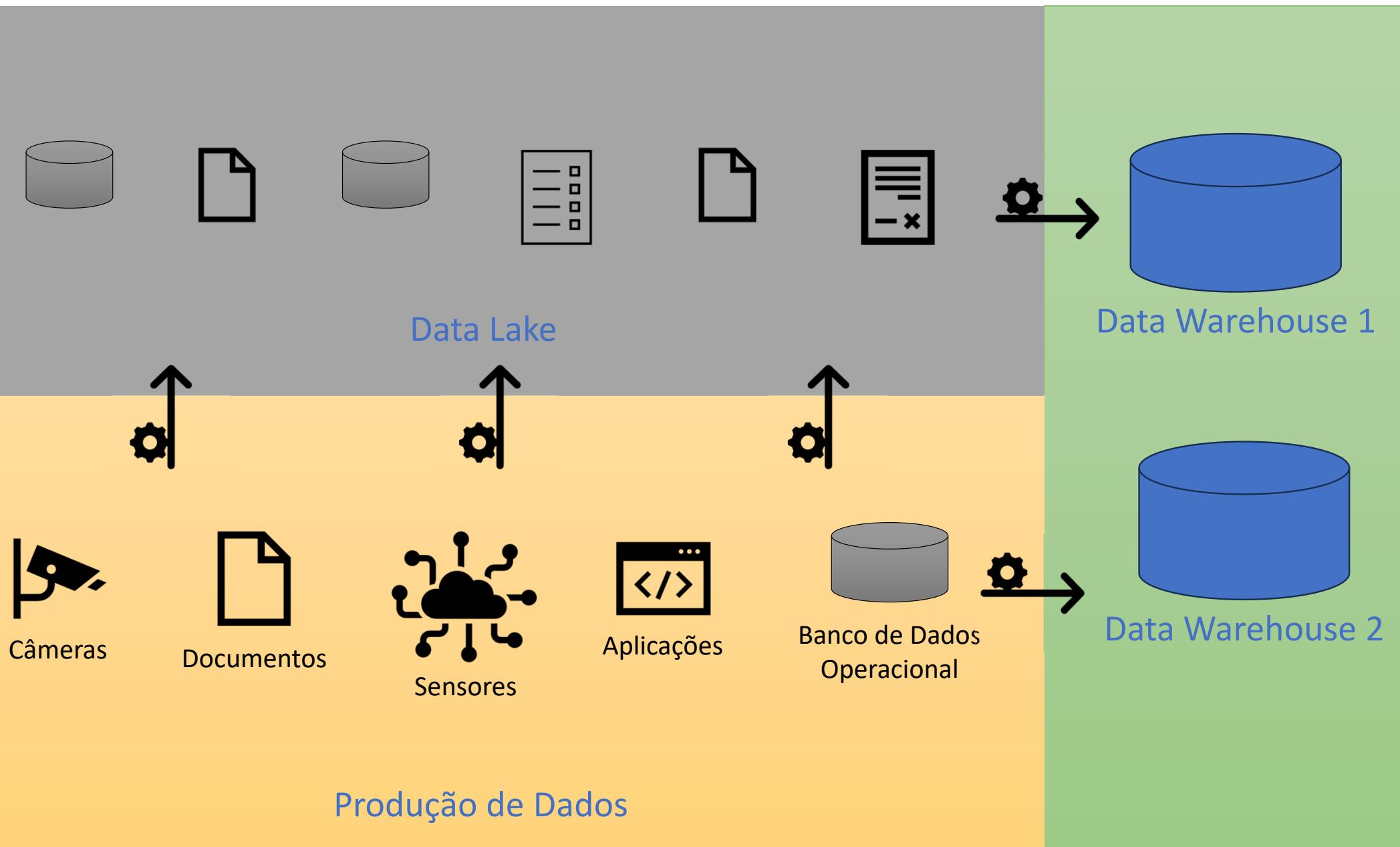
- Plataforma de computação em cluster e tolerante a falhas
- Open Source
- Em memória
- Possui componentes como SQL, Mllib e GraphX
- Suporta variedade de linguagens: Java, Scala, Python, R, SQL



Qual a relação?

- Databricks foi fundado pelos criadores originais do Apache Spark
- Objetivo é simplificar e potencializar o uso do Spark
- Databricks é uma plataforma comercial
- Spark é um produto open source
- O uso de Spark não é exclusivo do Databricks





Delta Lake

- Transações Acid
- Esquema de controle
- Histórico de versões
- Escalabilidade e desempenho
- Upserts e exclusões
- Compatibilidade



LakeHouse

- Combinação de elementos de Warehouse + Delta Lake
- Data Warehouse:
 - Desempenho
 - Qualidade de Dados
- Delta Lake:
 - Flexibilidade
 - ACID
 - Versionamento
 - Compatibilidade com Spark





Formatos para Big Data



Parquet



Apache
ORC™



Formatos para Big Data

- Data Lakes modernos tendem a armazenar dados em formatos “desacoplados” de ferramentas e abertos
- Formatos binários, compactados
- Suportam Schema
- Podem ser particionados entre discos:
 - Redundância
 - Paralelismo
- Intercambiáveis



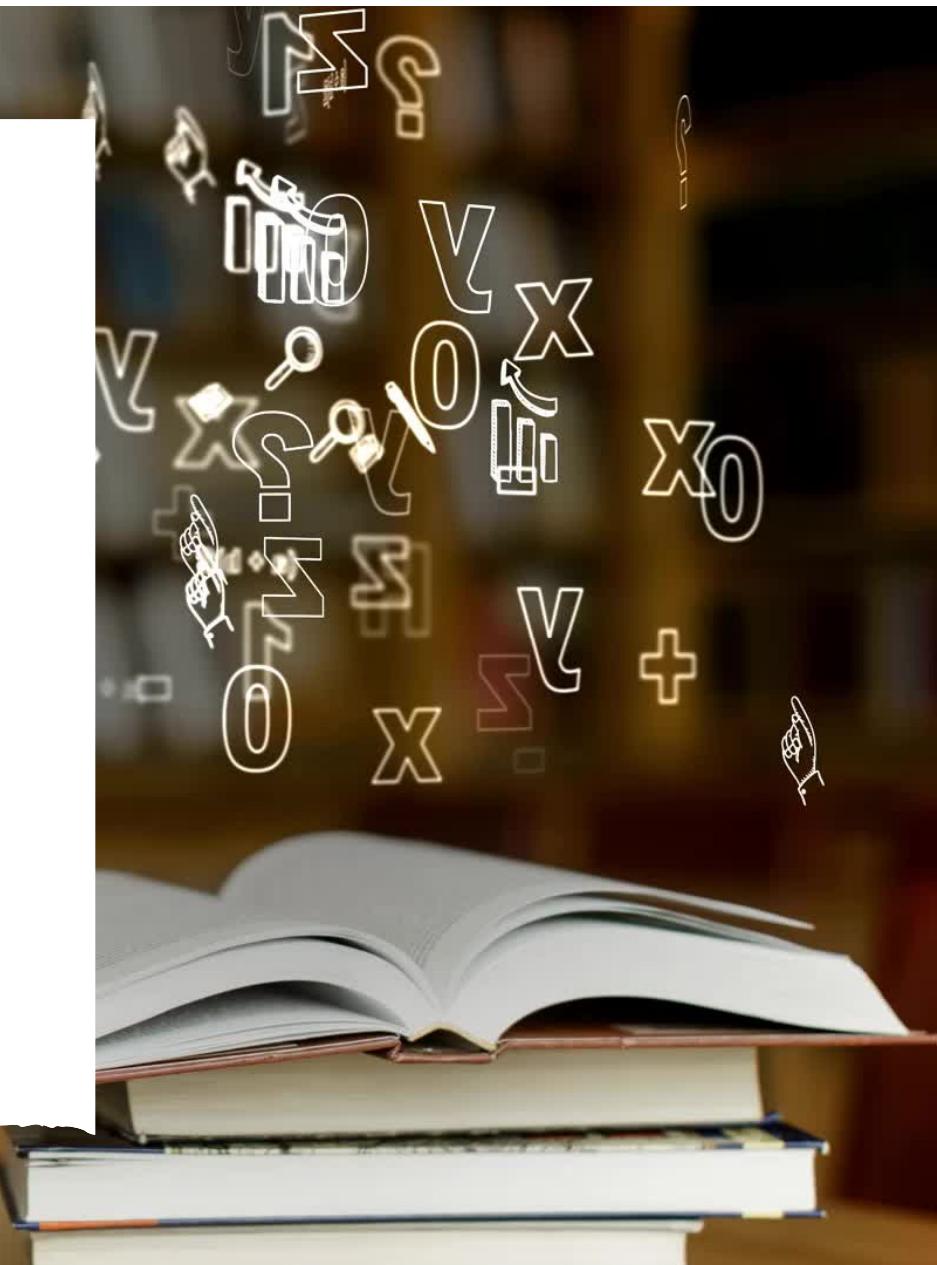
Formatos

- Parquet – Colunar, padrão do Spark
 - ORC – Colunar, padrão do Hive
 - Avro – Linha
-
- Muito atributos e mais escrita – linha
 - Menos atributos e mais leitura, coluna



Qual escolher?

- Em geral ORC é mais eficiente na criação (escrita) e na compressão
- Parquet tem melhor performance na consulta (leitura)
- O ideal é fazer um benchmark!



Bancos de dados de armazenamento por Linha e Coluna

Dados

OrderID	Product Name	Order Date	Price
1	Product A	2022-01-01	10.00
2	Product B	2022-01-02	25.00

Linha

OrderID	Product Name	Order Date	Price
1	Product A	2022-01-01	10.00

OrderID	Product Name	Order Date	Price
2	Product B	2022-01-02	25.00

Coluna

OrderID
1
2

Product Name
Product A
Product B

Order Date	Price
2022-01-01	10.00
2022-01-02	25.00

Armazenamento por Linha

- ✓ Cada linha representa uma tupla
- ✓ Armazenados juntos em disco
- ✓ Recupera-se a linha toda facilmente
- ✓ Ótimo para sistemas transacionais, onde ocorrem atualizações e inclusões de linhas
- ✓ Compressão: como uma linha pode ter diferentes tipos, é menos eficiente



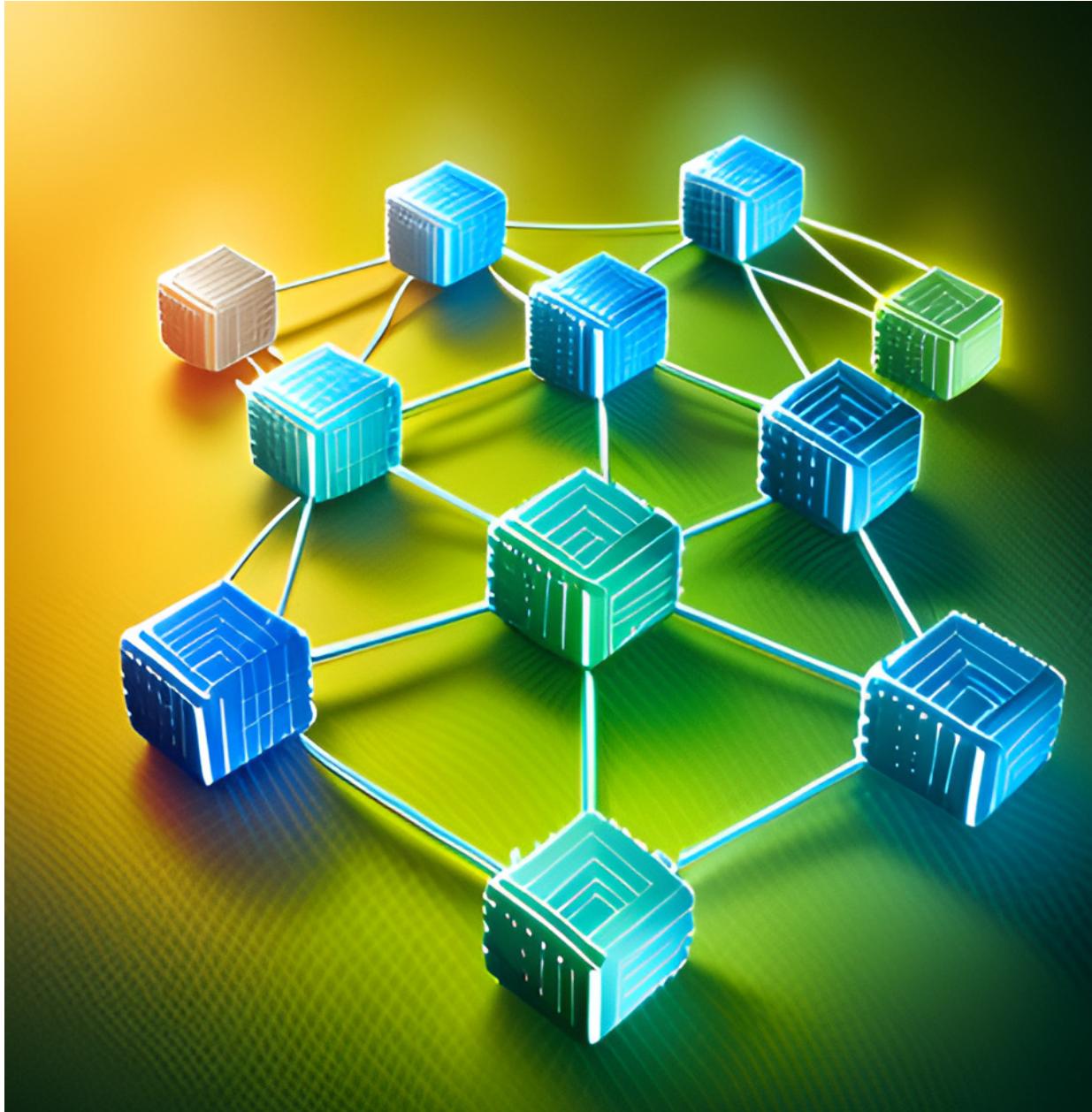
Armazenamento por Coluna

- ✓ Coluna é armazenada inteira em disco
- ✓ Em uma consulta, apenas as colunas necessárias são lidas
- ✓ Otimizado para consulta e análise, recuperando colunas inteiras com alta performance
- ✓ Compressão: como a coluna tem um mesmo tipo, é mais eficiente



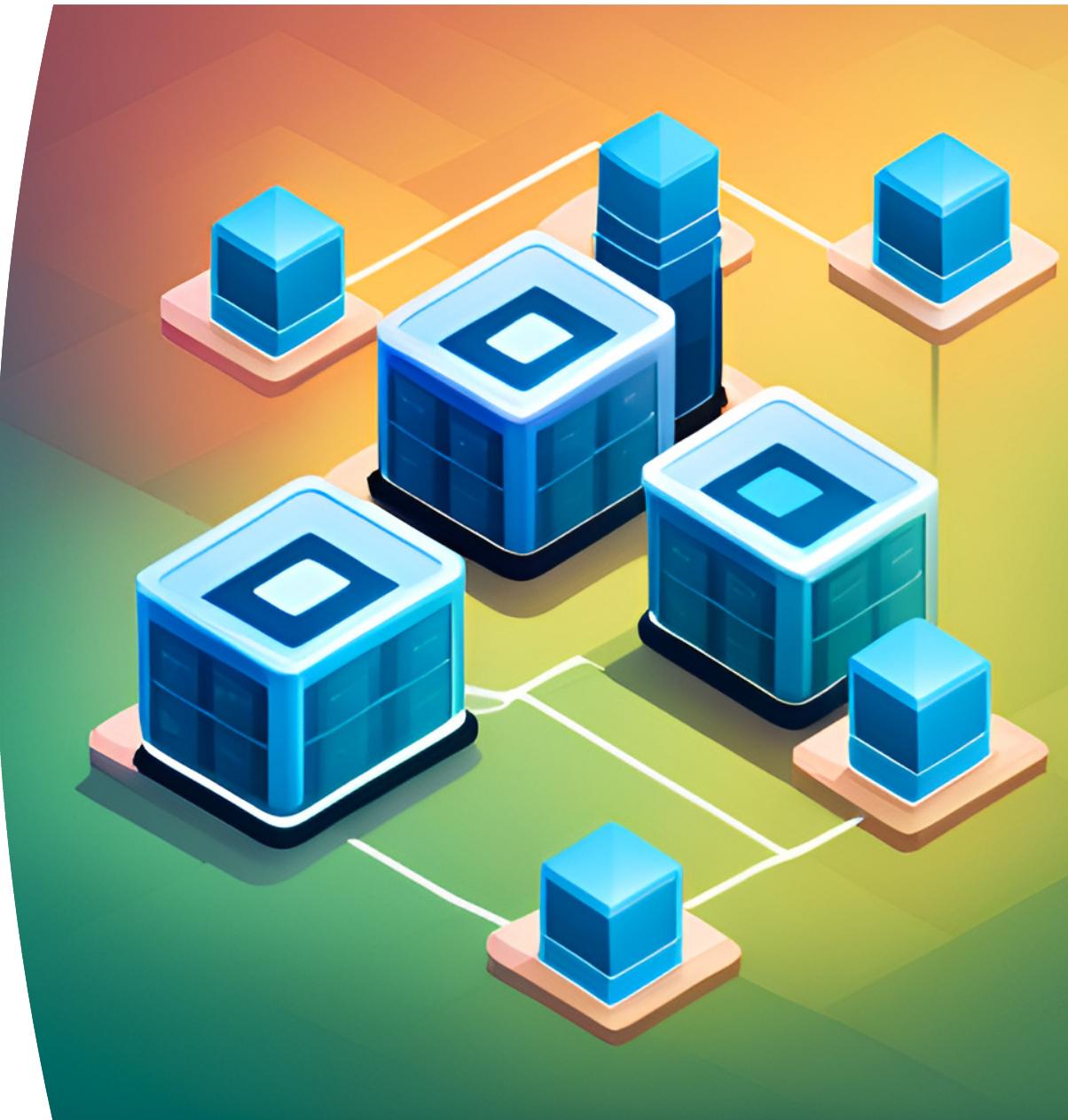
Sistema de Arquivos

- Um sistema de arquivos é uma maneira de organizar, armazenar e nomear dados em um dispositivo de armazenamento, como um disco rígido, SSD, cartão de memória ou pen drive. Ele controla como os arquivos são colocados no armazenamento e como são recuperados.



Sistemas de Arquivos Distribuidos

- Sistema de arquivos que permite que o acesso a arquivos e dados seja transparente, independentemente de onde os dados estejam fisicamente armazenados.
- Projetado para **armazenar e analisar grandes volumes de dados de maneira distribuída e paralela em um cluster de computadores**. Ele divide os arquivos em blocos, distribui-os em todo o cluster e mantém cópias redundantes para garantir a confiabilidade e a disponibilidade dos dados.
- Exemplos:
 - S3
 - Google File System
 - GlusterFS
 - HDFS



DBFS (Databricks File System)

- Abstração sobre o armazenamento de objetos na nuvem, como o Amazon S3 ou o Azure Blob Storage
- Permite que você armazene dados como se estivesse usando um sistema de arquivos distribuídos local, mas com a durabilidade e escalabilidade da infraestrutura de armazenamento em nuvem
- Para tarefas de processamento de dados, o Databricks utiliza o Apache Spark
- Também fornece uma camada de armazenamento transacional ACID para aprimorar a confiabilidade e o desempenho do sistema de arquivos (Delta Lake)



Databricks

- Community: **Gratuita**
- Edições **Comerciais**:
 - Implicam custo e conta em provedor Cloud
 - Existe ainda possibilidade de avaliação por 15 dias
 - Standard
 - Premium
 - Enterprise





Community

- Cluster único de 15GB de memória
- Totalmente gratuito
- Hospedada no AWS
 - Não precisa de conta no AWS
 - Não há custos
- Cluster é encerrado automaticamente entre 1 e 2 horas de inatividade

Workspace

- Notebooks
- Clusters
- Jobs
- Tabelas



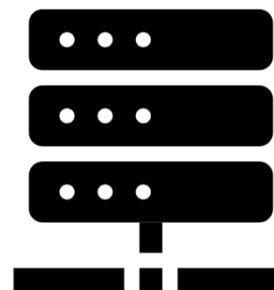
O Que é Spark

- Ferramenta de Processamento de Dados (Não é Data Storage)
- Distribuído em um Cluster
- Em memória
- Veloz
- Escalável
- Particionamento



Cluster

- Rede de Computadores



E-mail

Storage

ERP

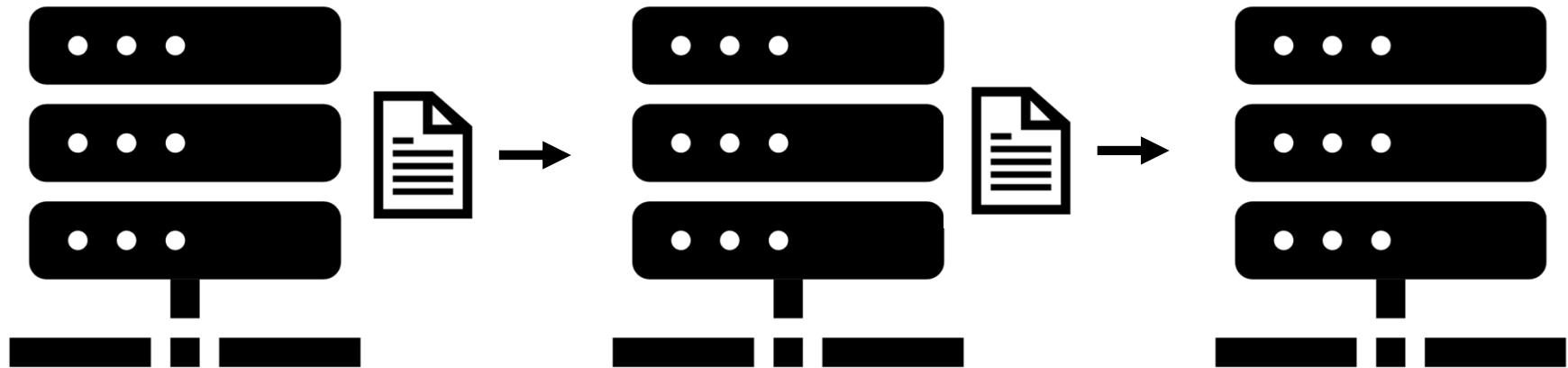
- Rede de Computadores - Cluster



Proc. Dados

Proc. Dados

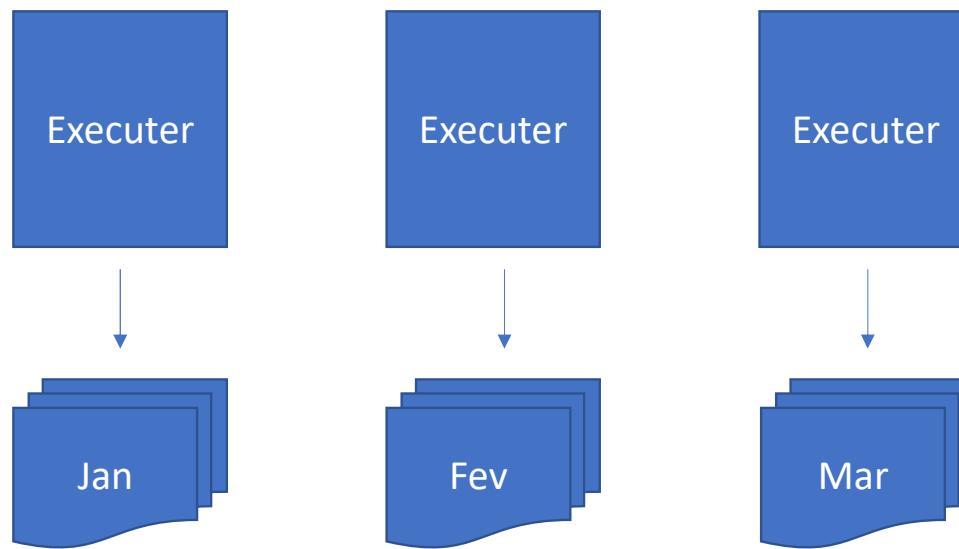
Proc. Dados



Replicação e Tolerância a Falha

- Dados são copiados entre os nós do cluster. Isso traz o benefício de, entre outras coisas, tolerância a falhas

Particionamento



Spark VS Python, R ou Banco de Dados

Você precisa Processar dados!

Custo computacional: CPU,
Memória, Rede etc.

Spark tem arquitetura voltada a
processar dados!

Melhor performance, porém:

Não substitui Python

Não substitui SQL ou um
SGBDR



Linguagens

Scala 

Python 

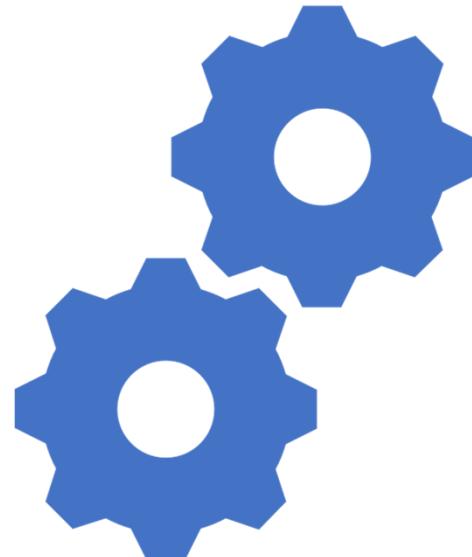
Java 

R 

SQL 

Projeto “Extremamente” Ativo

+ 1000 colaboradores ativos



Spark

Universidade da
Califórnia iniciou
projeto Spark em 2009

Versão 1.0 lançada em
Maio de 2014 pela
Fundação Apache



Dataframe

Tabelas com linhas e colunas

Imutáveis

Com schema conhecido

Linhagem preservada

Colunas podem ter tipos diferentes

Existem análises comuns: Agrupar, ordenar, filtrar

Spark pode otimizar estas análises através de planos de execução



Lazy Evaluation

O processamento de transformação de fato só ocorre quando há uma Ação: Lazy Evaluation

Tipos de Dados



Tipo
ByteType
ShortType
IntegerType
LongType
FloatType
DoubleType
DecimalType
StringType
BinaryType
BooleanType
TimestampType
DateType
ArrayType
MapType
StructType
StructField

Schema

- 
- Você pode deixar para o Spark inferir a partir de parte dos dados ou
 - Você pode definir o schema
 - Definir tem vantagens:
 - Tipo correto
 - Sem overhead



Spark

Memória

Operação em Cluster

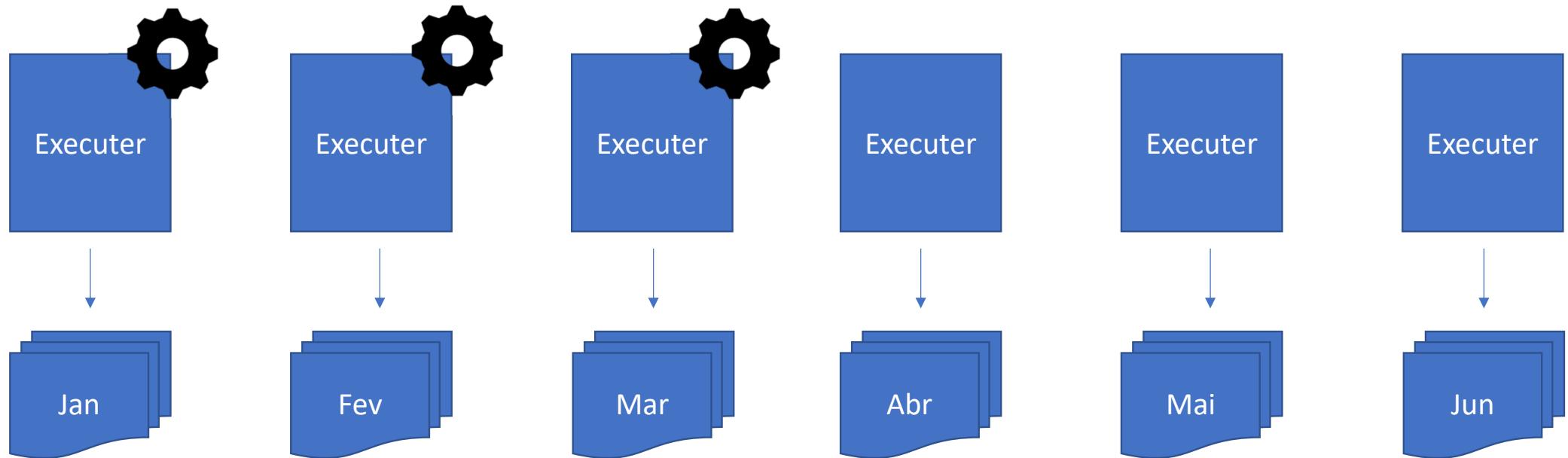
Particionamento (divisão de dados)

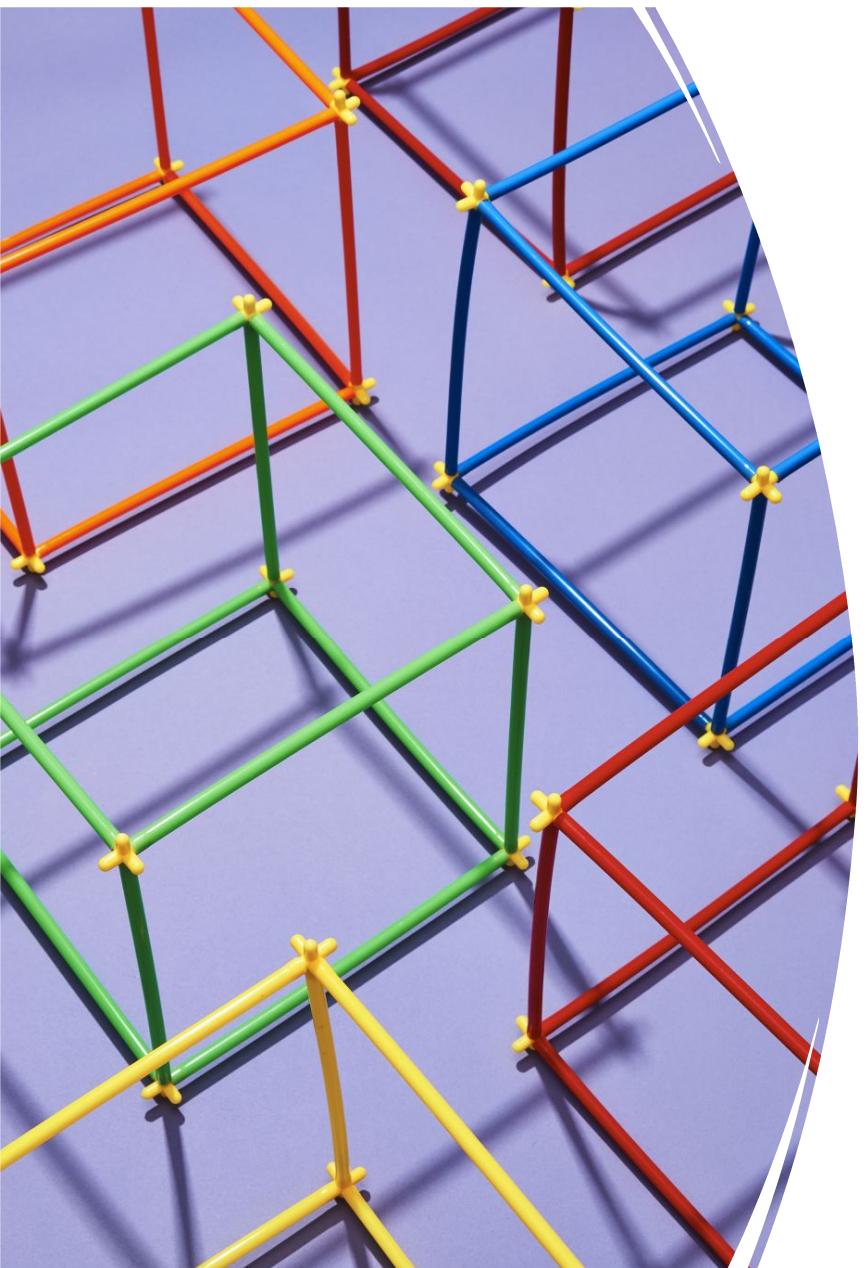
Paralelismo

Redundância

Particionamento

Total de vendas de Janeiro a Março





Particionamento

Por padrão dados são partionados de acordo com número de núcleos

Cada partição fica em um nó e tem uma task



Shuffle

Redistribuição de Dados entre partições

Particionamento

- Dados são particionados por padrão e dependem de vários fatores e configurações
- Podemos particionar explicitamente em disco (`partitionBy`)
- Ou em memória `repartition()` or `coalesce()`

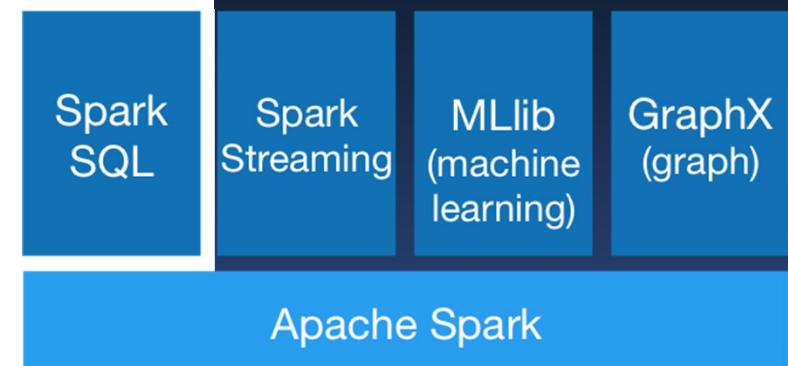
Bucketing



- Semelhante a particionamento, porém com número fixo de partições
- Ideal para coluna com alta cardinalidade
- Pode ser usado com conjunto com Particionamento

Componentes

- Machine Learning (Mlib)
- SQL (Spark SQL)
- Processamento em Streaming
- Processamento de Grafos (GraphX)



Spark SQL

Permite ler dados tabulares de várias fontes (CSV, Json, Parquet, ORC etc)

Pode usar sintaxe SQL



Streaming: Spark Structured Streaming

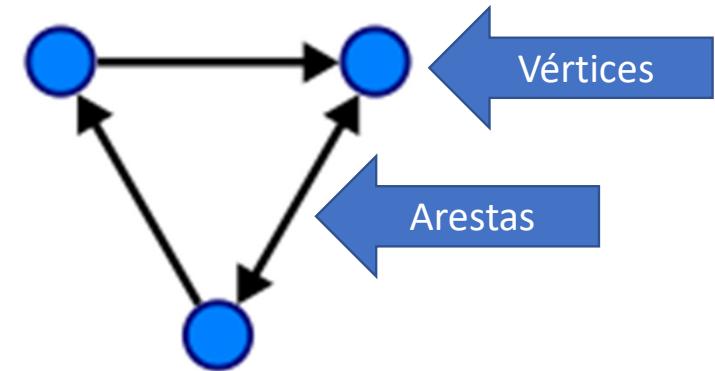
- Dados estruturados
- Novos registros adicionados ao final da tabela





Grafos acíclicos dirigidos

- Spark Constrói Gráficos Acíclicos Dirigidos



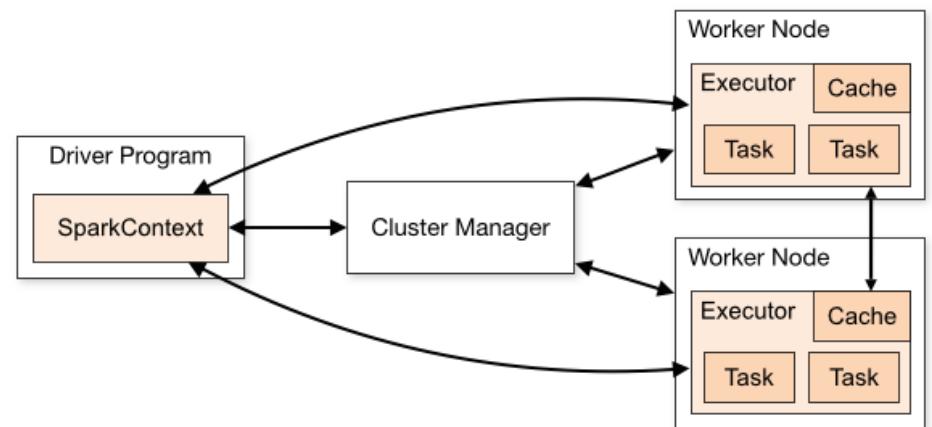
Tungsten

Motor de Execução



Estrutura

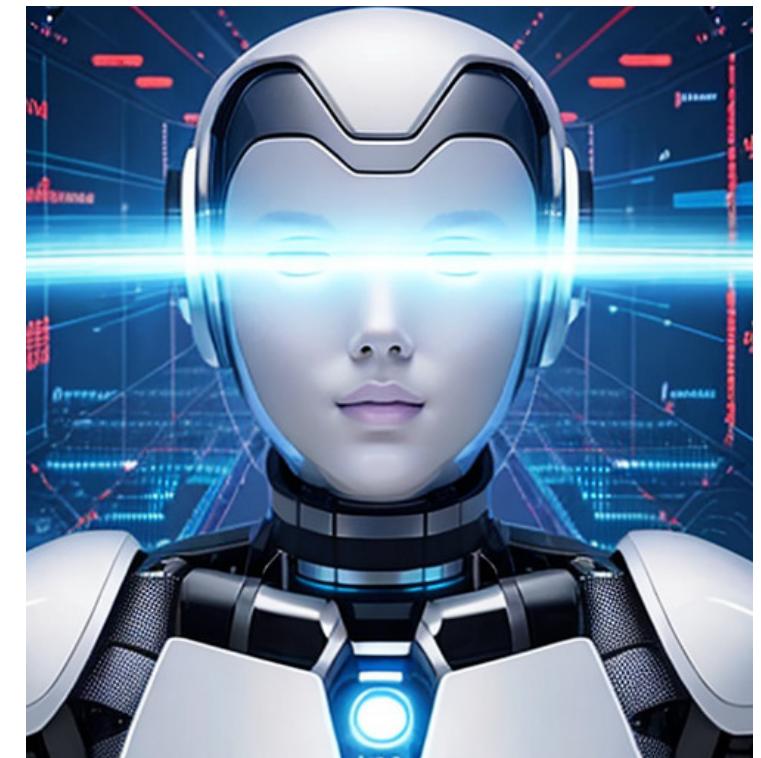
- **Driver:** Inicializa SparkSession, solicita recursos computacionais do Cluster Manager, transforma as operações em DAGs, distribui estas pelos executors
- **Manager:** Gerencia os recursos do cluster. Quatro possíveis: built-in standalone, YARN, Mesos e Kubernetes
- **Executer:** roda em cada nó do cluster executando as tarefas



Elementos

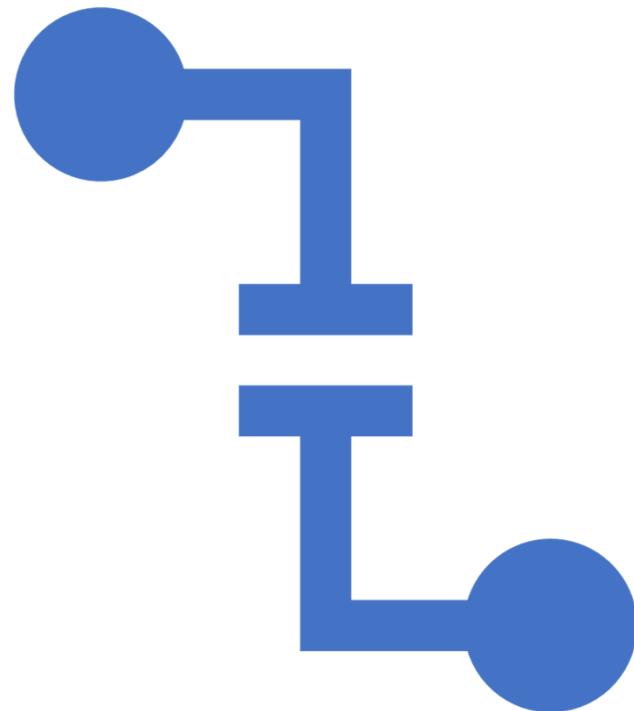
SparkSession: Seção

Aplication: Programa

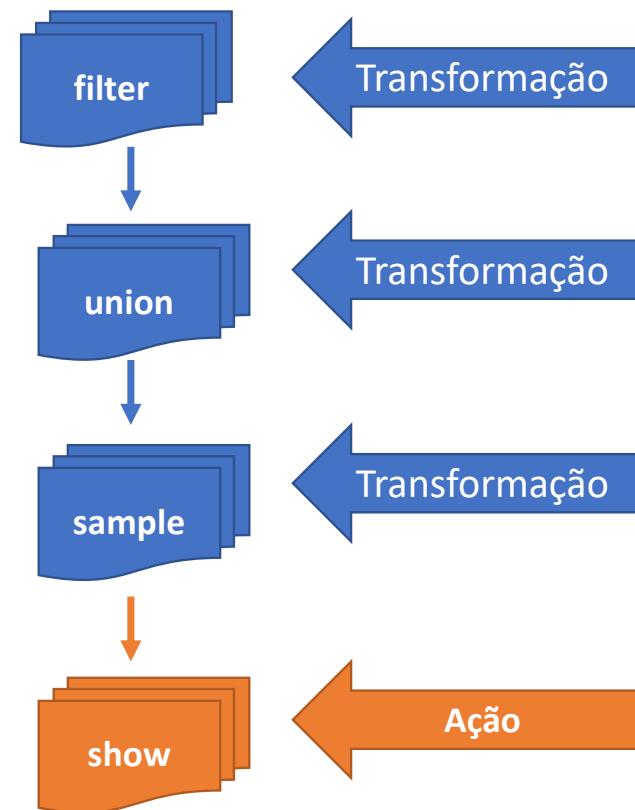
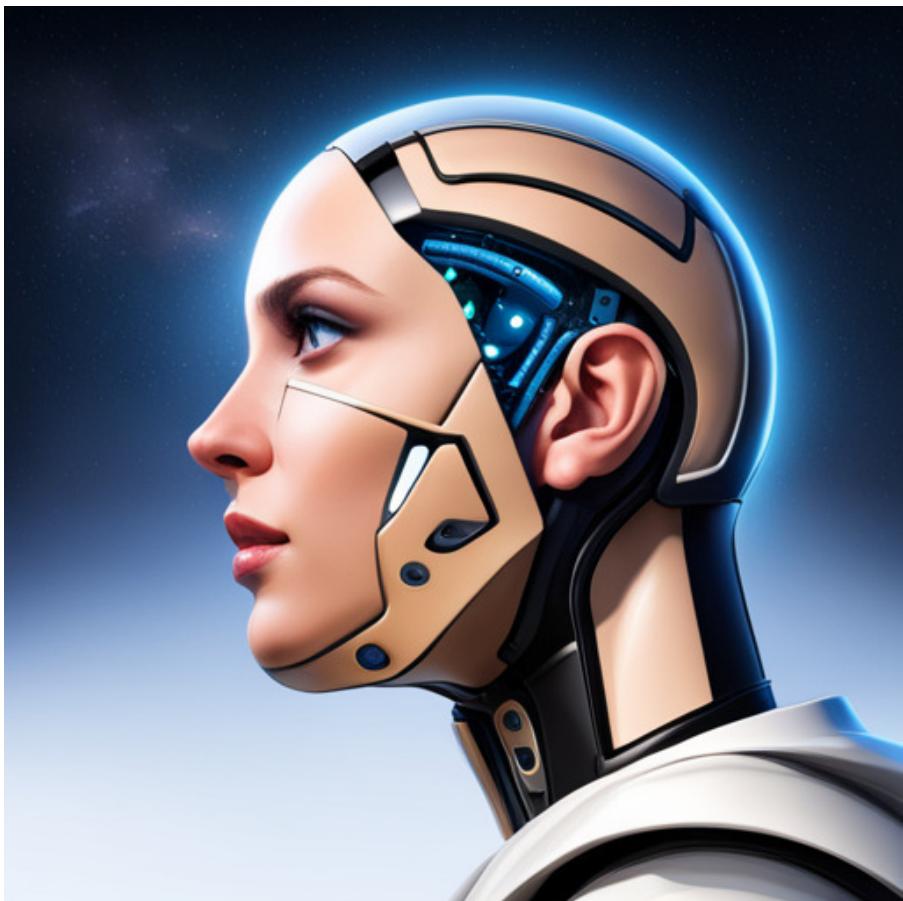


Transformações e Ações

- Um data frame é imutável: traz tolerância a falha
- Uma transformação gera um novo data frame.
- O processamento de transformação de fato só ocorre quando há uma Ação: Lazy Evaluation



Lazy Evaluation

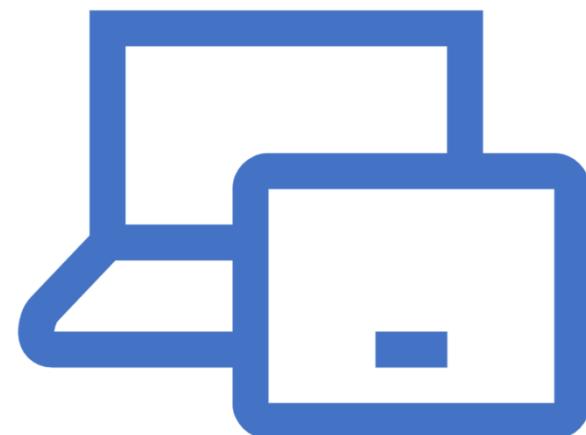


Transformações e Ações

Transformações	Ações
map	reduce
filter	collect
flatMap	count
mapPartitions	first
mapPartitionsWithIndex	take
sample	takeSample
union	takeOrdered
intersection	saveAsTextFile
distinct	saveAsSequenceFile
groupByKey	saveAsObjectFile
reduceByKey	countByKey
aggregateByKey	foreach
sortByKey	
join	
cogroup	
cartesian	
pipe	
coalesce	
repartition	
repartitionAndSortWithinPartitions	

Transformações: Narrow e Wide

- Os dados necessários estão em uma mesma partição
- Os dados necessários estão em mais de uma partição

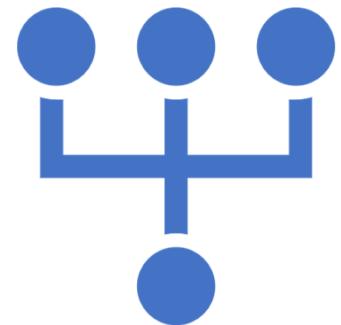


Componentes

Job: Tarefa

Stage: Divisão do Job

Task: Menor unidade de trabalho. Uma por núcleo e por partição



SparkContext

Conexão
transparente
com o Cluster

SparkSession:
acesso ao
SparkContext



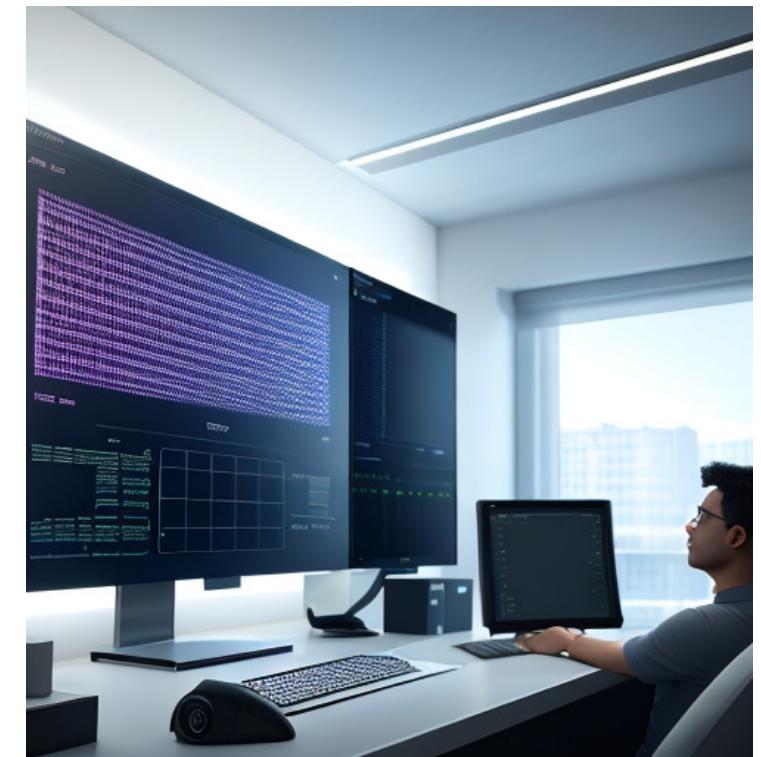
SparkContext / Session

Você pode rodar script Spark no shell (pyspark)

O Spark cria uma sessão automaticamente chamada **spark**

Se criar uma aplicação spark, você precisa criar, por exemplo:

```
spark = (SparkSession  
    .builder  
    .appName ("Meuapp")  
    .getOrCreate () )
```



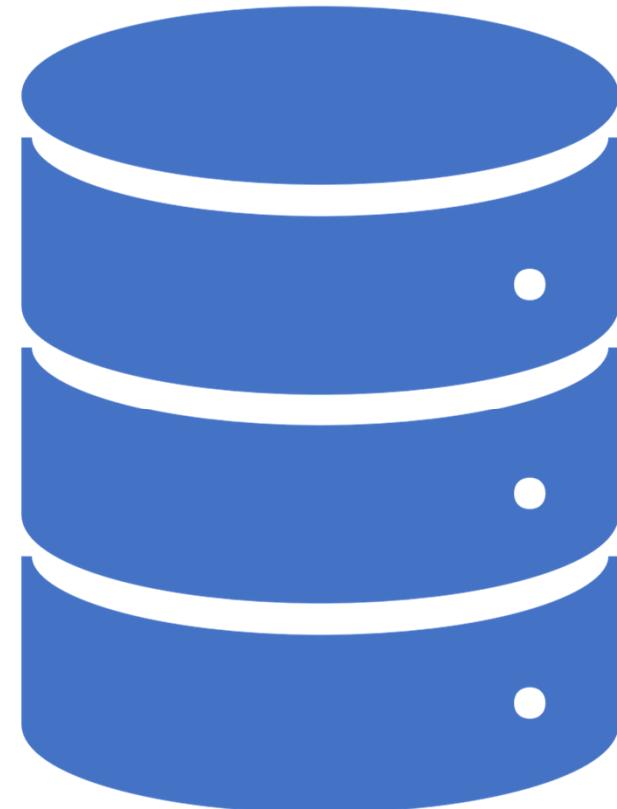
Delta Lake e Data Lakehouse

- Tecnologia de Armazenamento: Data Lake usando Delta Lake
- Transações ACID
- Mantém log de transações
- Gravamos e lemos arquivos no formato “delta”
 - Gravador de e para dataframes spark
 - Nos bastidores são arquivos parquet (não são manipulados diretamente como tal)



Arquivo

Objeto no DBFS



Tabelas

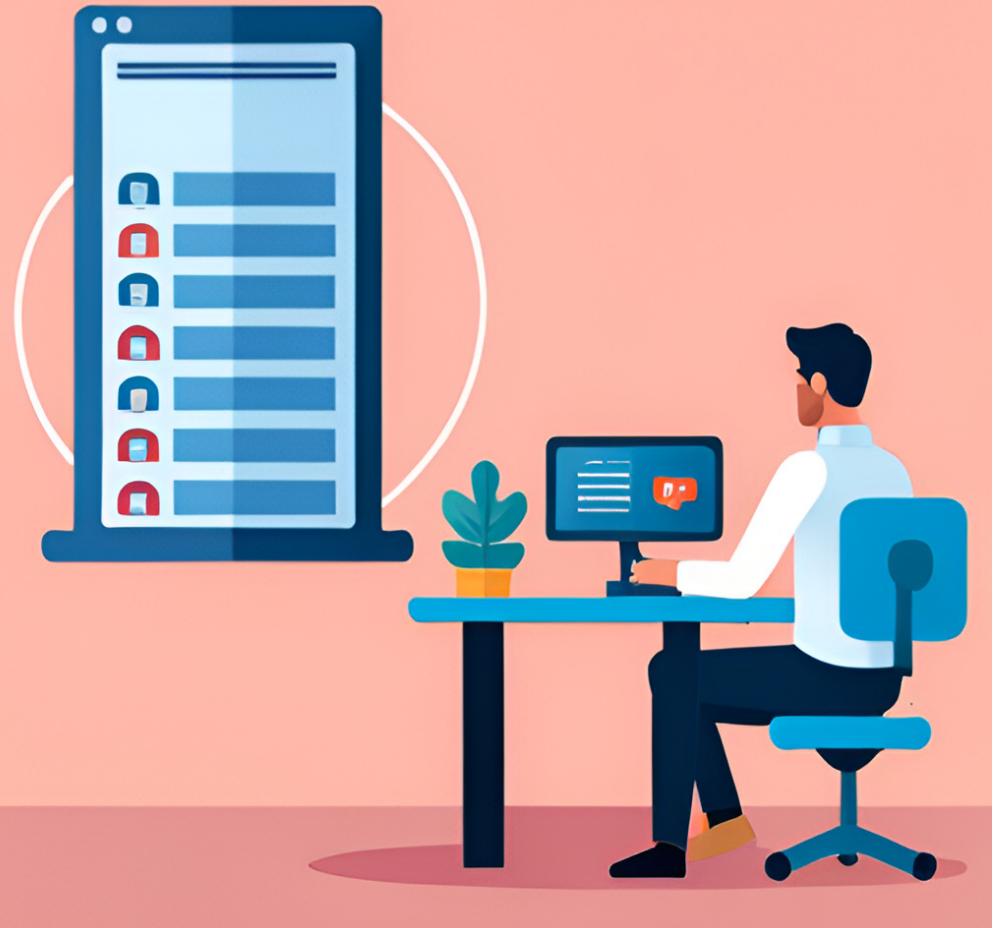
Tabela: dados em um formato específico com metadados no metastore.

Pode ser consultado usando SQL.

Criada a partir de um arquivo ou dataframe

Tabelas Externas: é criado um alias, sem cópia. Se o arquivo deixar de existir, a tabela não funciona mais.

Tabela Gerenciada: é criada uma cópia. Se você apagar o arquivo, a tabela continua existindo.



Data Frames

Formato ideal para processamento de dados

Dados efêmeros

São perdidos caso o cluster seja desligado

Para persistência devem ser salvos fisicamente (parquet)





Arquivo Delta (Delta Lake)

- Camada de armazenamento de código aberto que traz recursos ACID, controle de versão e controle de esquema
- Manipulados através de dataframes
- Podem ser consultados por SQL
- Não efemero

