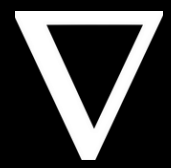




DATA

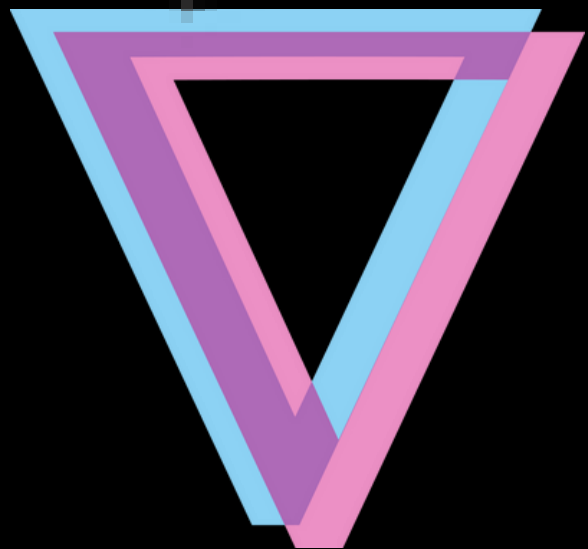
ETL COM SPARK E PYSPARK

Luiz Felipe
e
Victor Zaneti

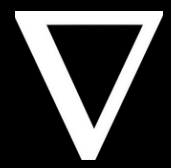


Índice

- O que é o Sparky
- Processamento Distribuído de Dados
- Cluster, Lazy Evaluation e Outros Componentes
- Introdução ao ETL com PySpark
- Prática com PySparky



O que é o **Sparky**



O que é o Spark?



Spark é um **framework open-source** para **processamento distribuído de dados**. Sua arquitetura baseia-se em um modelo de computação **inmemory**, o que o torna muito mais rápido que o Hadoop tradicional.

1

Escalabilidade

2

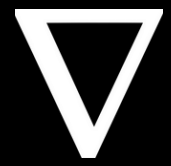
Diversidade

3

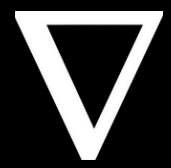
Interação

4

Popularidade

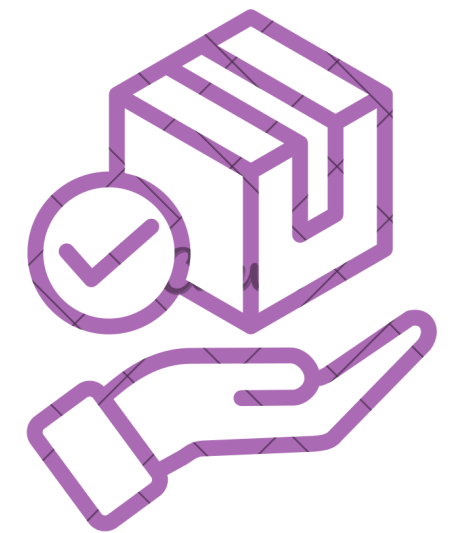


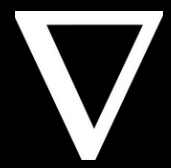
Distribuição de Dados



Processamento Distribuído

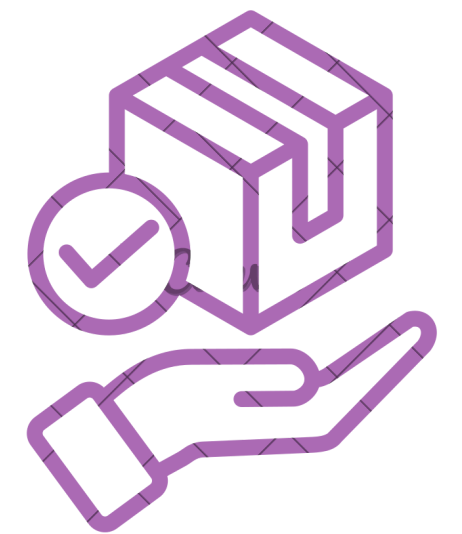
- Técnica que **divide** o trabalho de processamento de dados entre várias máquinas.
- Cada máquina executa uma parte da tarefa, **acelerando a conclusão** do trabalho.
- Capacidade de lidar com **grandes volumes de dados** de forma eficiente.

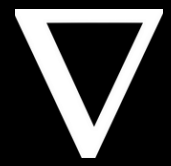




Vantagens

- Distribuição de tarefas em um **cluster** de máquinas interconectadas.
- Aproveitamento de recursos combinados para maior **velocidade** de processamento.
- Ideal para grandes conjuntos de dados que exigem maior poder computacional.



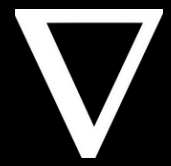


Computação Inmemory

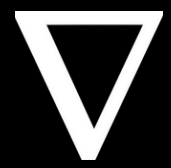
Computação Inmemory

Técnica de processamento de dados diretamente na **memória RAM**, em vez de depender de discos rígidos ou armazenamento externo.



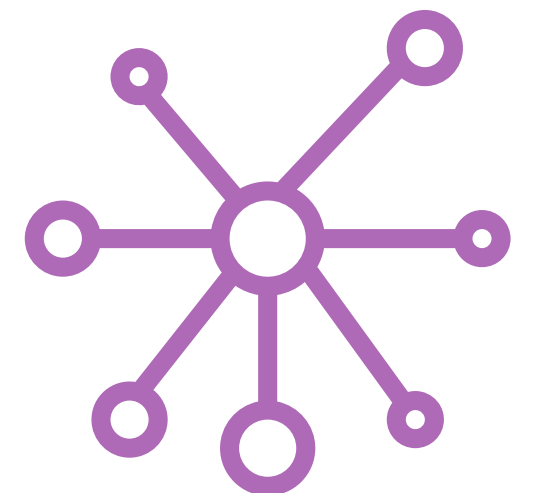


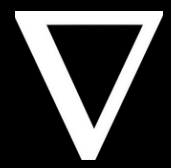
Cluster e Lazy Evaluation



Cluster

Um cluster é um conjunto de máquinas (nós) interconectadas que trabalham juntas para processar dados. Cada nó no cluster pode ser uma máquina física ou virtual e tem sua própria memória, CPU e armazenamento.





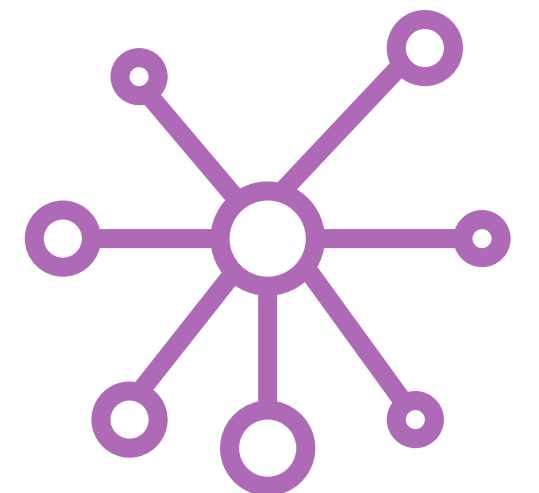
Como Funciona?

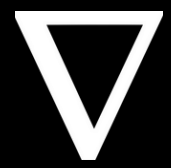
O Spark distribui as tarefas entre esses nós, garantindo que cada parte do trabalho seja realizada simultaneamente, o que resulta em um desempenho significativamente mais rápido em comparação com abordagens de processamento sequencial.

1
Driver

2
Executores

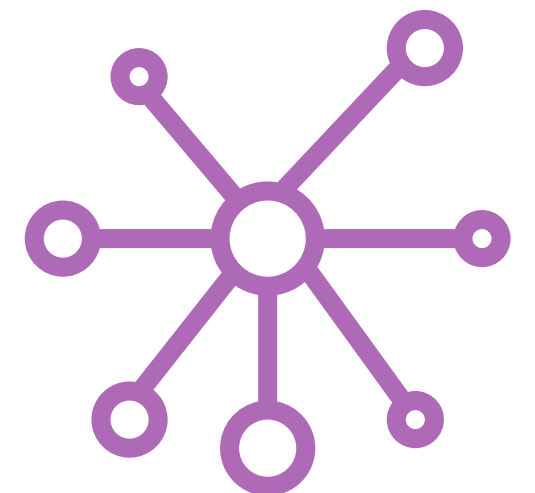
3
**Gerenciador
de cluster**

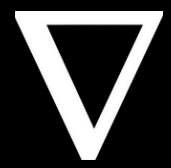




Vantagens

- Paralelismo : A principal vantagem de um cluster é a capacidade de processar dados em paralelo, o que reduz drasticamente o tempo de execução de tarefas.
- Escalabilidade: É possível adicionar mais nós ao cluster para aumentar a capacidade de processamento, permitindo que o Spark escale horizontalmente conforme necessário.
- Resiliência: O Spark é projetado para ser resiliente, lidando automaticamente com falhas de nós e redistribuindo tarefas conforme necessário

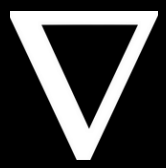




Lazy Evaluation

Lazy Evaluation é um conceito de programação onde a avaliação de expressões é adiada até que o resultado seja realmente necessário. Em vez de calcular os resultados de operações imediatamente, o Spark constrói um plano de execução que será executado apenas quando os dados forem realmente solicitados.

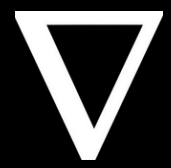




Como Funciona?

- Criação de RDDs: Quando um usuário cria um RDD (Resilient Distributed Dataset) e aplica operações a ele (como map, filter ou flatMap), o Spark não executa essas operações imediatamente. Em vez disso, ele registra essas operações em um plano lógico.
- Construção do DAG: À medida que mais operações são aplicadas, o Spark constrói um grafo acíclico dirigido (DAG) que representa todas as transformações a serem realizadas.
- Execução Sob Demanda: A execução real do plano ocorre somente quando uma ação é chamada, como collect() ou count(). Nesse ponto, o Spark avalia o DAG, otimizando as operações e executando-as em paralelo nos nós do cluster.

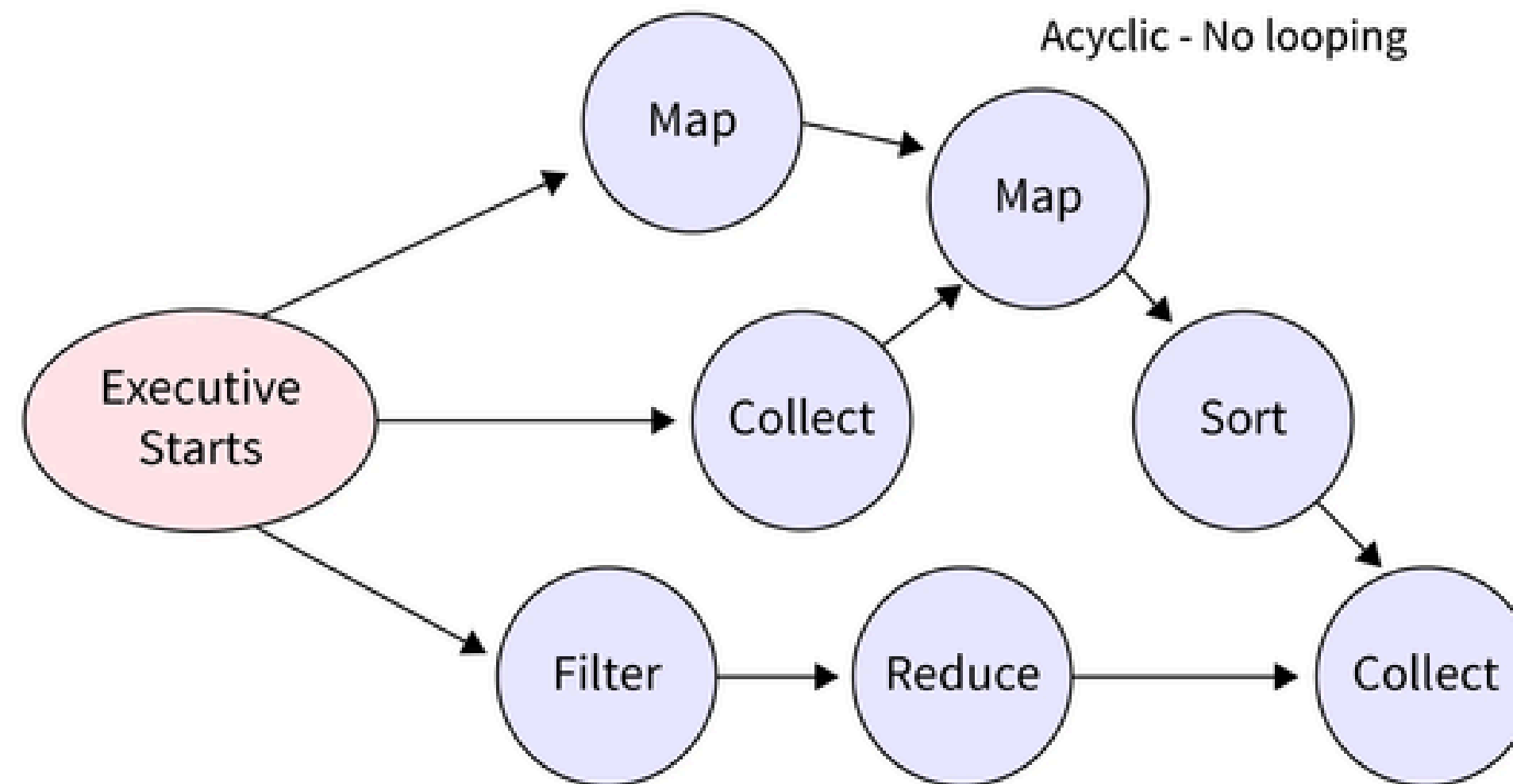


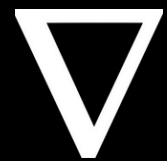


DAG (Directed Acyclic Graph)

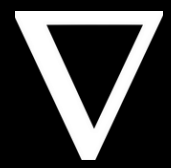
Directed - Only in a single direction

Acyclic - No looping



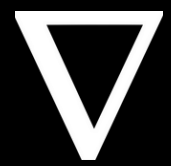


PySpark



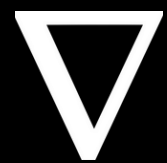
PySpark

O PySpark é a interface do Apache Spark para a linguagem Python, oferecendo a capacidade de processar grandes volumes de dados em paralelo e distribuído sobre clusters de máquinas. Ele permite que desenvolvedores e cientistas de dados aproveitem a simplicidade do Python junto com a escalabilidade e o desempenho do Spark, facilitando o processamento e a análise de dados em larga escala



Principais Componentes





Principais Componentes

SparkSQL

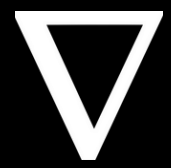
O PySpark oferece suporte para consultas SQL, através do módulo Spark SQL. Você pode escrever consultas SQL diretamente em DataFrames, permitindo uma combinação de consultas SQL com a programação tradicional do Python

PySpark Streaming

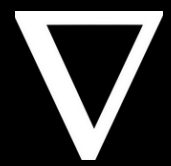
Facilita o processamento em tempo real de fluxos de dados contínuos. Com PySpark Streaming, você pode processar dados que chegam continuamente de fontes como Kafka, sockets TCP, ou sistemas de arquivo distribuído, aplicando transformações em tempo real

MLlib

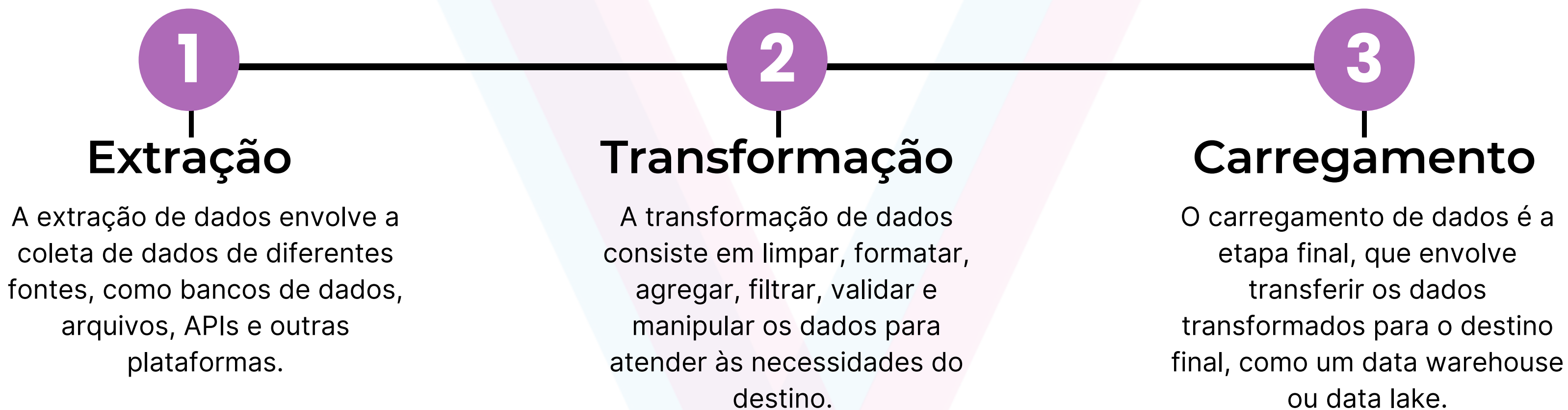
O PySpark inclui o MLlib, uma biblioteca escalável de aprendizado de máquina que facilita a construção e a execução de pipelines de aprendizado de máquina em grandes volumes de dados.

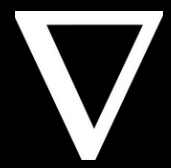


Introdução ao ETL com PySpark



ETAPAS DO ETL





ETL de dados com Spark

O Spark fornece ferramentas e APIs para ler e transformar dados de diferentes formatos e fontes. Além de conseguir operar em Streaming

Leitura

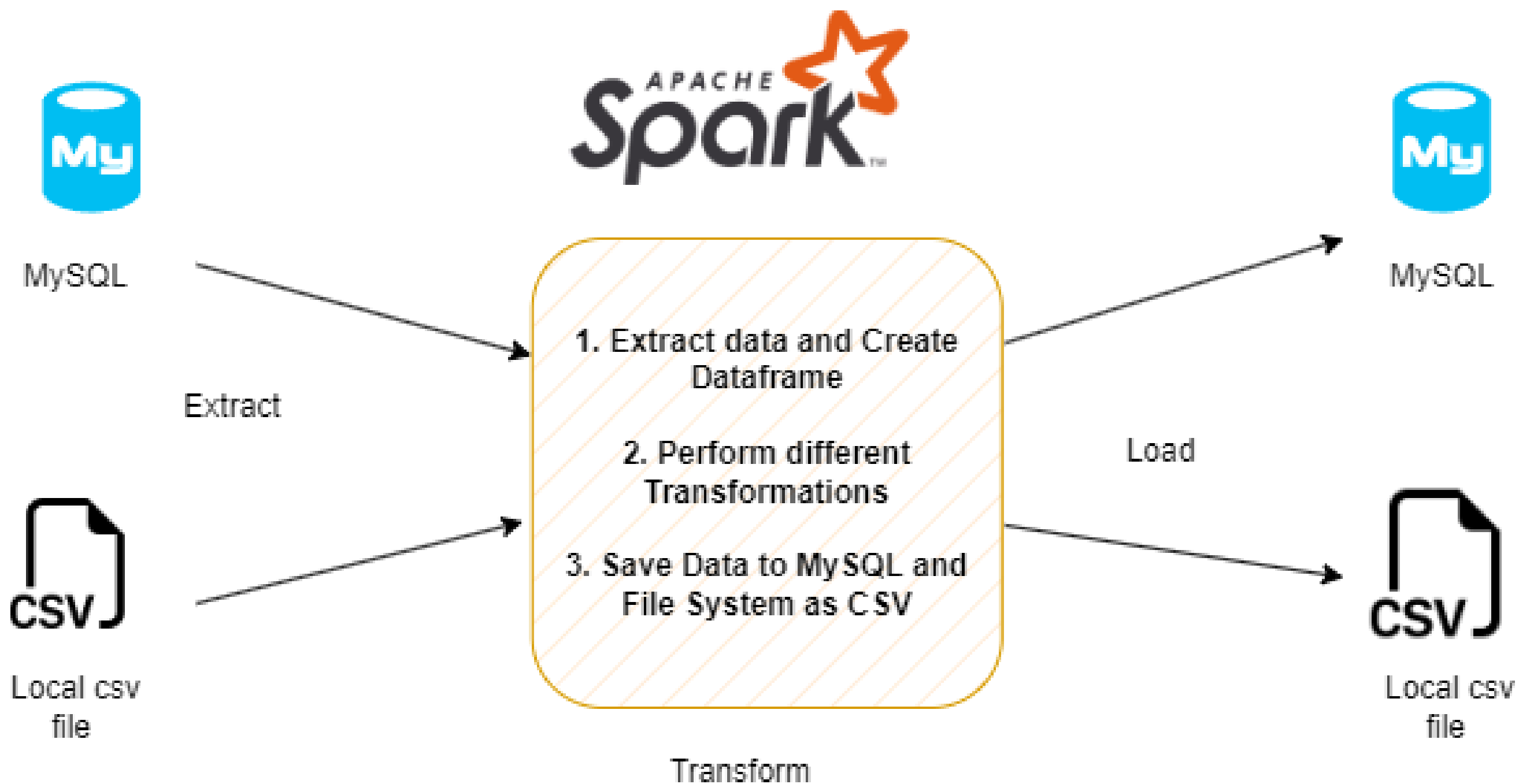
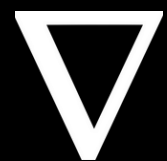
O Spark pode ler dados de diversos formatos, como CSV, JSON, Parquet e Avro. Além de conseguir fazer leituras de dados em bancos de dados

Transformação

O Spark oferece uma ampla gama de transformações para manipular dados, como filtragem, agregação, ordenação, junção e transformações de valores.

Operações do Spark

O Spark oferece um conjunto completo de operações para trabalhar com dados, incluindo transformações, ações e operações de window.



Extract Transform Load



Exemplo

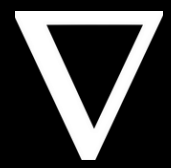
Extração

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, upper

# 1. Extração
# Criando uma sessão Spark
spark = SparkSession.builder \
    .appName("Exemplo ETL com Spark") \
    .getOrCreate()

# Lendo dados de um arquivo CSV
input_path = "dados_entrada.csv" # caminho para o arquivo de entrada
df = spark.read.csv(input_path, header=True, inferSchema=True)

# Exibindo os dados extraídos
print("Dados Extraídos:")
df.show()
```

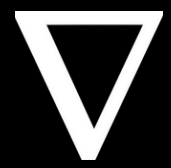


Exemplo

Transformação

```
# 2. Transformação
# Realizando algumas transformações
# Exemplo: Converter o nome para maiúsculas e filtrar por uma condição
df_transformado = df.select(
    col("id"),
    upper(col("nome")).alias("nome_maiusculo"),
    col("idade")
).filter(col("idade") > 18) # Filtrando apenas maiores de 18 anos

# Exibindo os dados transformados
print("Dados Transformados:")
df_transformado.show()
```

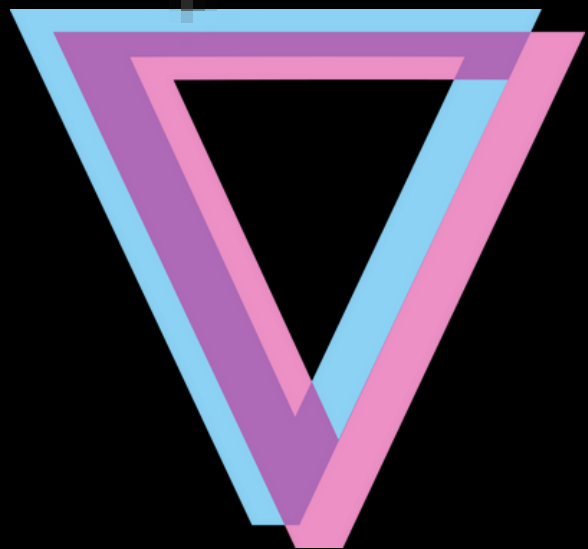


Exemplo

Carga

```
# 3. Carga
# Salvando os dados transformados em um novo arquivo CSV
output_path = "dados_saida.csv" # caminho para o arquivo de saída
df_transformado.write.csv(output_path, header=True)

# Encerrando a sessão Spark
spark.stop()
```

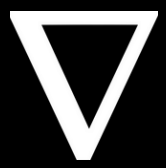


Prática com PySparky



Google Colab


 [google.com](https://colab.google.com)



 data@icmc.usp.br

 @data.icmc

 /c/DataICMC

 /icmc-data

 data.icmc.usp.br

OBRIGADO!