FIAP

# Stream Processing Pipelines

MBA+

## Plataforma Apache Spark

**Prof. Rafael Pereira**

# Ementa

- Stream Processing.

- Apache Spark.
- Apache Flink.

- Amazon Kinesis.
- Azure Stream Analytics.
- Google Cloud Dataflow .

- Event-Based vs. Micro-Batching.

- CI/CD: Continuous Integration / Continuous Delivery.

# Roteiro

## Stream Processing Solutions

- Stream Architecture
- Stream Processing vs. Batch Processing
- Características
- Componentes
- Ferramentas
- Aplicações
- Desafios

## Plataforma Apache Spark

- Características
- Componentes
- Arquitetura
- Funcionamento
- Casos de Uso

# Quem sou eu?

- BACHAREL EM SISTEMAS DE INFORMAÇÃO - IMPACTA TECNOLOGIA
- LICENCIADO EM MATEMÁTICA - UNIVERSIDADE CRUZEIRO DO SUL
- PÓS-GRADUAÇÃO EM TECNOLOGIA EM SOFTWARE - USP
- PÓS-GRADUAÇÃO EM DIVERSIDADE E GÊNERO - UNIVERSIDADE DE SANTOS
- PÓS-GRADUAÇÃO EM GESTÃO DE PROJETOS - FECAP
- MESTRANDO EM CIÊNCIA DA COMPUTAÇÃO - UNIVERSIDADE DO CAMPO LIMPO

17 ANOS DE EXPERIÊNCIA EM TECNOLOGIA (IA, SOFTWARE E GESTÃO)

- MAKRO
- SODEXO
- BAYER
- CARREFOUR
- ITAÚ

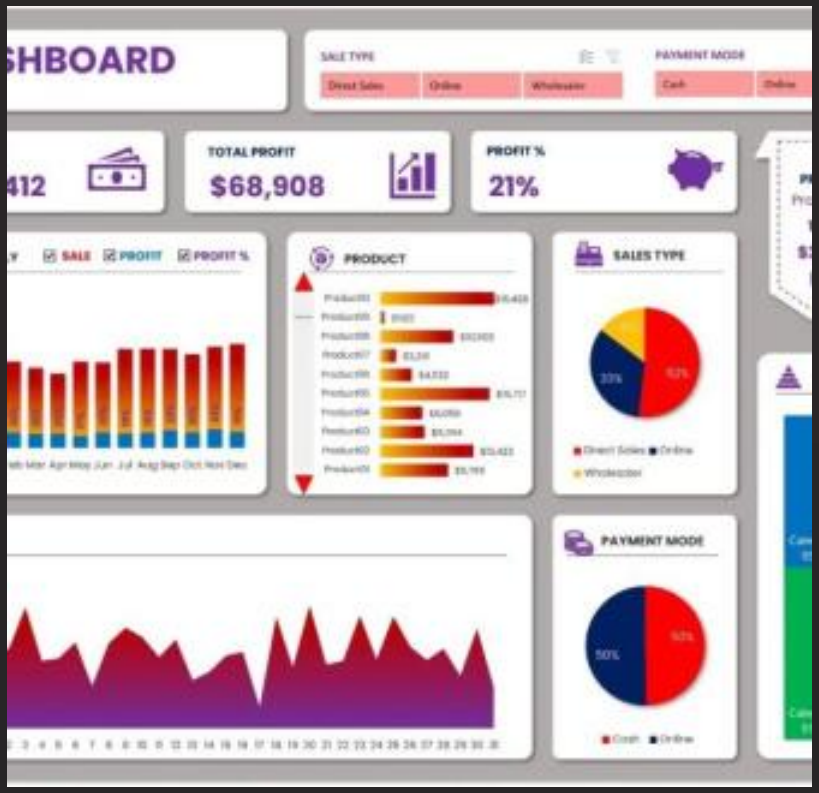
10 ANOS COMO DOCENTE

- FIAP
- SENAC
- CENTRO PAULA SOUZA
- FAMESP - MBA EM DATA SCIENCE

# Por que Spark?

## O Dilema dos Dados Modernos

Executivos lidam com volumes massivos de dados, mas Excel e SQL Server são limitados.



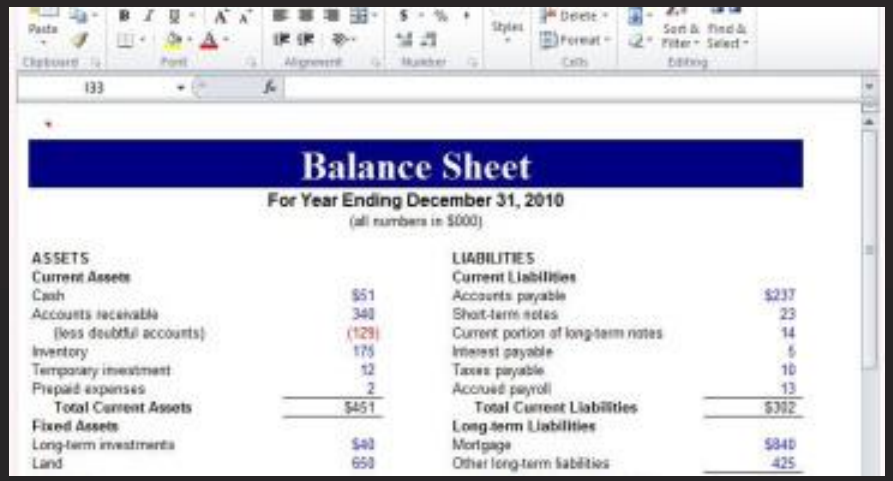

## Onde o Spark Entra

Apache Spark processa **terabytes de dados rapidamente**, integrando-se a análises executivas.

## Quando Usar (e Quando Evitar)

- Use Spark para grandes volumes, streaming e machine learning.
- Evite para relatórios simples ou pequenos datasets.

# Spark vs Ferramentas Tradicionais

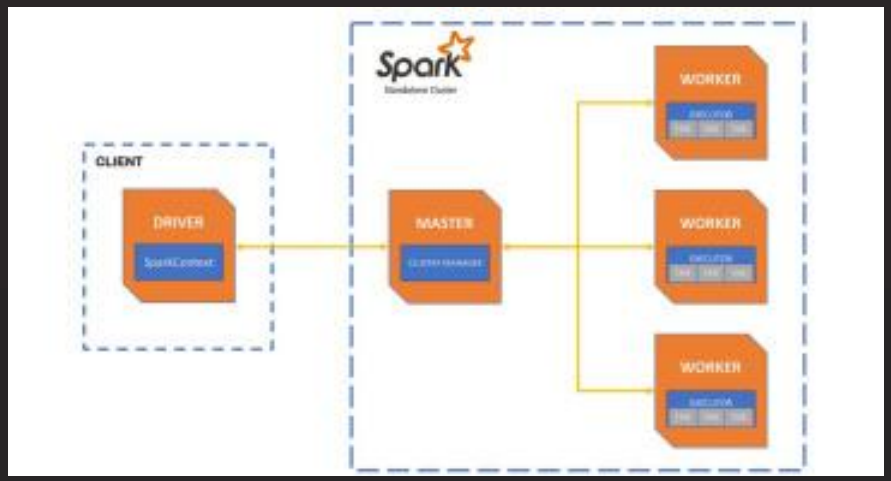


Balance Sheet

For Year Ending December 31, 2010

(all numbers in $000)

| ASSETS | | LIABILITIES | |
|---|---|---|---|
| **Current Assets** | | **Current Liabilities** | |
| Cash | $51 | Accounts payable | $237 |
| Accounts receivable | 340 | Short-term notes | 23 |
| (less doubtful accounts) | (129) | Current portion of long-term notes | 14 |
| Inventory | 175 | Interest payable | 5 |
| Temporary investment | 12 | Taxes payable | 10 |
| Prepaid expenses | 2 | Accrued payroll | 13 |
| **Total Current Assets** | $451 | **Total Current Liabilities** | $302 |
| **Fixed Assets** | | **Long-term Liabilities** | |
| Long-term investments | $40 | Mortgage | $840 |
| Land | 650 | Other long-term liabilities | 425 |




**Excel / SQL Server**

- Ótimo para relatórios operacionais
- Fácil de aprender
- Limite de GBs de dados
- Lento com grandes volumes

**Apache Spark**

Escala para terabytes, processamento paralelo em memória, suporta batch/streaming, integra IA e analytics.

# Casos de Uso no Brasil

FIAP

**Nubank**

Detecção de fraude em tempo real com Spark no Databricks usando streaming de transações.

**iFood**

Otimização de entregas com análise de tráfego e demanda em tempo real usando janelas temporais.


**Magalu**

Personalização de ofertas com processamento batch de milhões de pedidos diários.

# Batch vs Streaming


## Processamento Batch: Relatório Mensal

Como um CFO que recebe um balanço fechado ao final do mês.

## Processamento Streaming: Painel em Tempo Real

Como um COO monitorando vendas minuto a minuto durante uma Black Friday.

Ambos são essenciais — o primeiro para estratégia, o segundo para ação imediata.

# Arquitetura Spark

## Driver

Como um CEO: coordena todas as tarefas e toma decisões centrais.

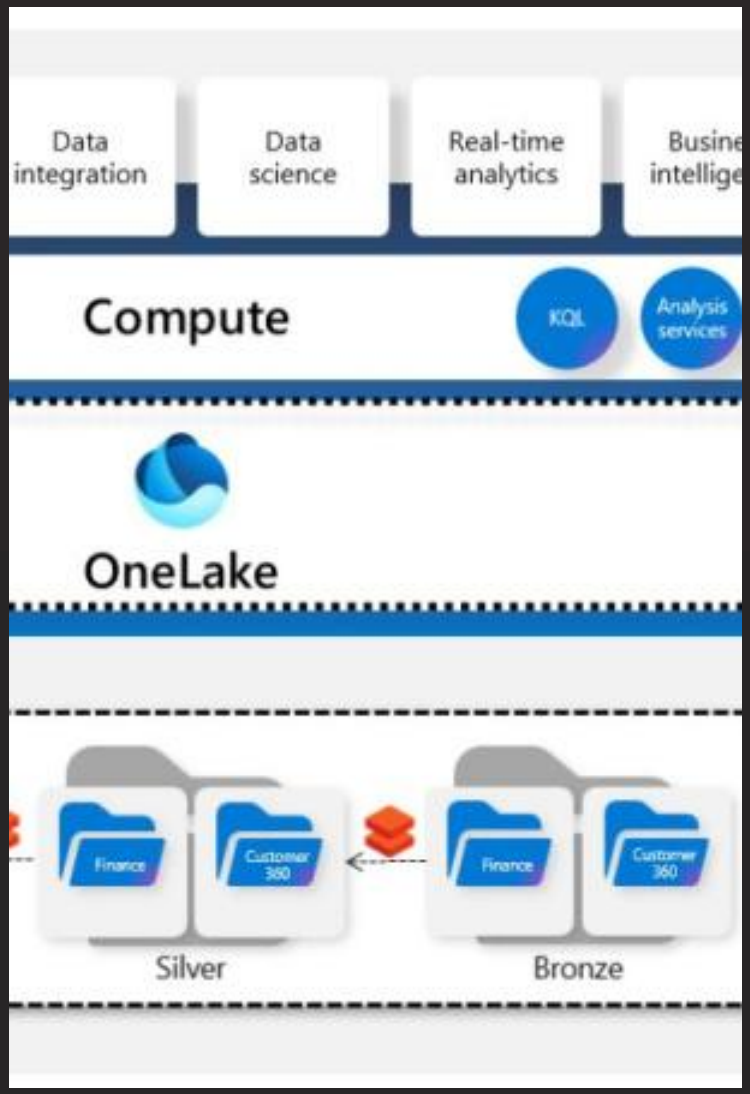
## Executor

Como gerentes regionais: executam tarefas locais com autonomia.

## Cluster

Como uma rede de lojas: escala conforme a demanda, integrando recursos.

# Principais Benefícios do Databricks




databricks

## 1 - Velocidade

Spark otimizado para nuvem: processamento acelerado.

## 2 - Colaboração

Notebooks compartilhados: equipes colaboram em tempo real.

## 3 - Escalabilidade

Infraestrutura elástica que cresce com a necessidade de dados.

## 4 - Segurança

Governança e conformidade regulatória: controles avançados.

# O Conceito de Lakehouse

## O que é um Lakehouse?

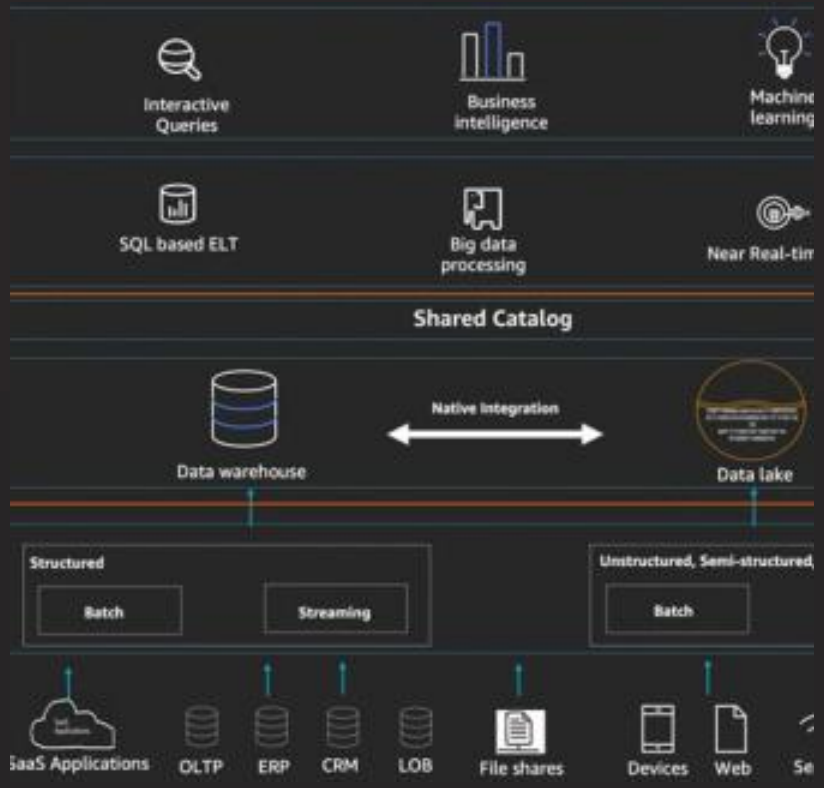
Um Lakehouse une as vantagens de um data lake (baixo custo, escalabilidade) com as funcionalidades de um data warehouse (estrutura, qualidade, desempenho).

## Como Funciona

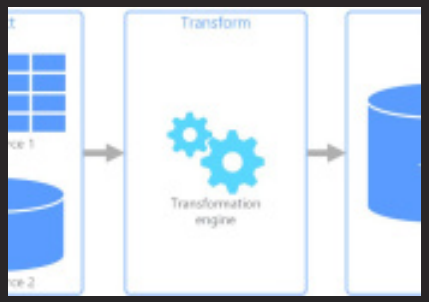
Dados brutos entram, são organizados em camadas (bruta, enriquecida, analítica), permitindo consultas SQL rápidas e machine learning.

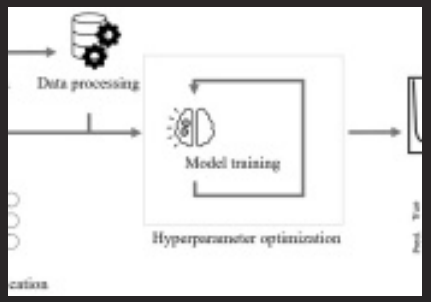
## Impacto nos Negócios

Elimina a movimentação de dados entre sistemas, reduzindo complexidade e custos.




# Fluxo de Trabalho no Databricks


**Ingestão**
Coleta de dados de APIs, bancos e arquivos.

**Transformação**
Limpeza e estruturação com Spark SQL ou Python.

**Modelagem**
Criação de modelos

**Visualização**
Exportação para ferramentas de BI ou relatórios internos.

# Governança de Dados

## Por Que é Importante?

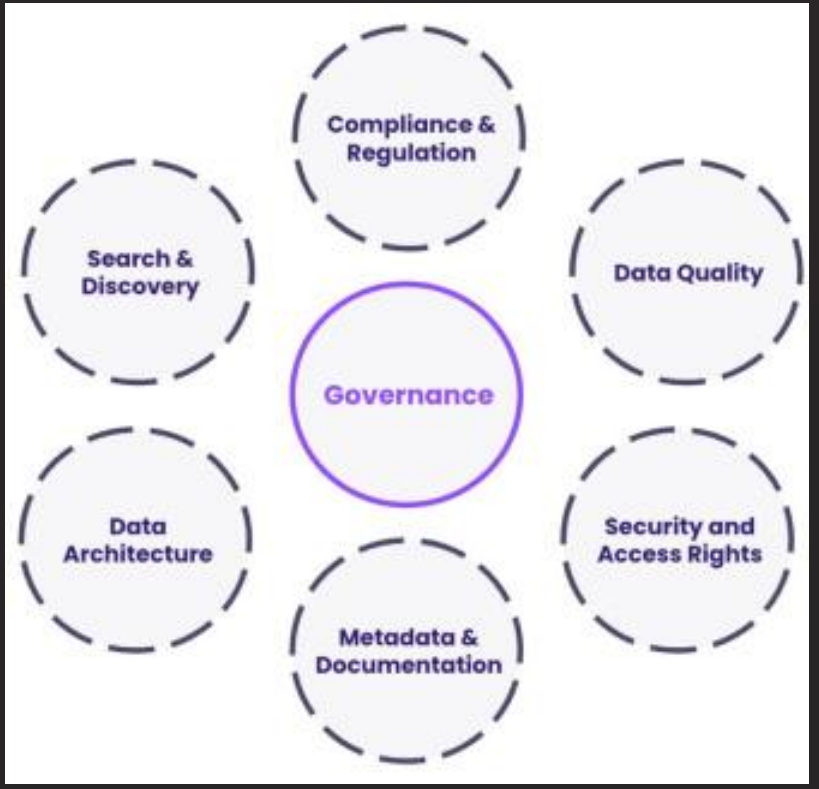
Em um mundo regulado (LGPD), a governança garante acesso ético e seguro aos dados.

## Como o Databricks Ajuda

- Controle de acesso granular (usuário, grupo, função)
- Auditoria de acesso a dados
- Classificação automática de dados sensíveis

## Aplicação Estratégica

Líderes tomam decisões confiantes, com dados confiáveis e em conformidade.



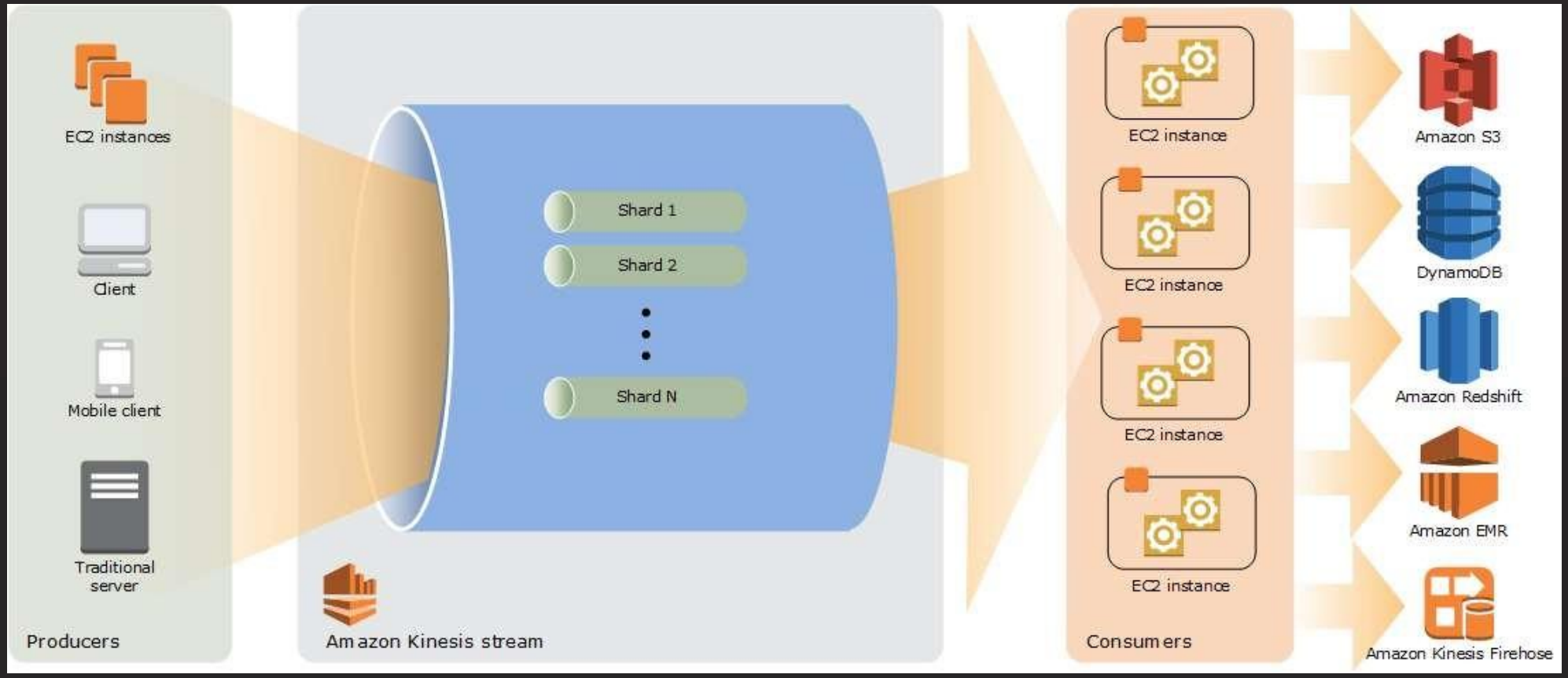
# Stream Processing

- Conceito: É um modelo de processamento de dados que realiza a transformação das informações em lotes menores (*micro-batching*) ou *on-demand*.

- Basicamente é o modelo oposto ao processo em batch tradicional, onde os dados são processados “de uma vez só” em lotes grandes de informações.

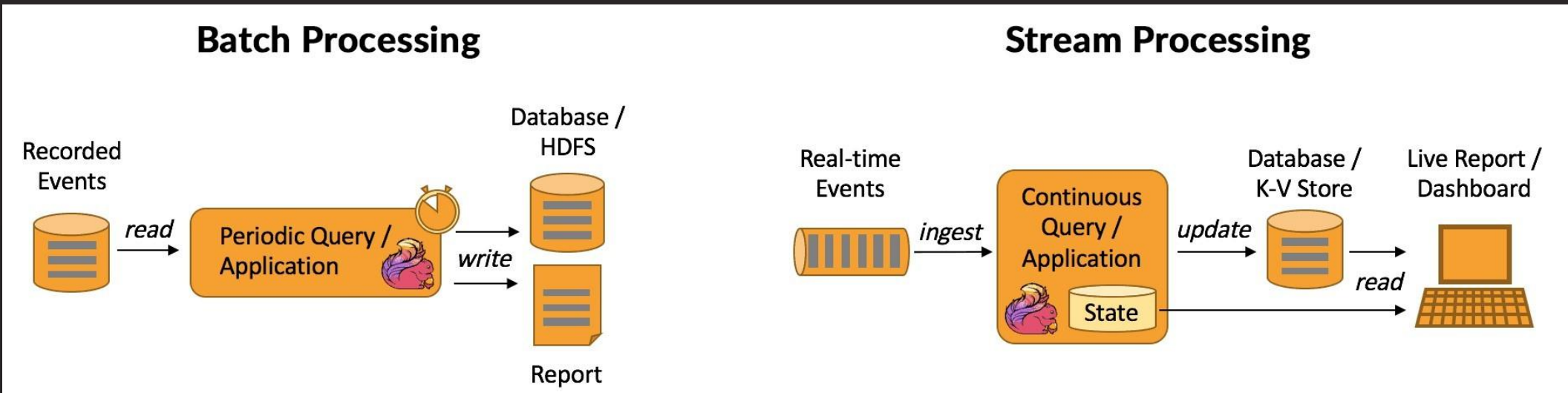
- Veremos nos próximos slides as diferenças arquiteturais e de implementação nas duas abordagens.

# Stream Processing - Arquitetura




Fonte: Amazon

# Stream Processing vs. Batch Processing




Fonte: K21Academy

# Stream Processing vs. Batch Processing

| Característica | Batch | Stream |
|---|---|---|
| **Natureza do Dado** | Lotes Grandes | Lotes Pequenos / Fluxo |
| **Latência** | Elevada | Baixa |
| **Escalabilidade** | Vertical | Horizontal |
| **Complexidade do Código** | Baixa | Alta |
| **Uso de Memória** | Elevada | Baixa |
| **Tolerância a Falhas** | Implementação Simples | Implementação Complexa |

- Dica: Lembre-se que a tabela possui comparativos de alto-nível e genéricos, o caso concreto pode mudar as características e a performance do processo!

# Stream Processing – Características

- O processamento e análise de dados em tempo-real habilita cenários de negócio e de tecnologia na qual se precisa de um tempo de reação reduzido.

- Especialmente em empresas *data-driven* ou *event-driven*, ter um tempo de resposta menor se torna um diferencial de mercado.

# Stream Processing – Componentes

- Ingestão de Dados: Coleta e pré-processamento dos dados, que podem vir de forma tanto estruturada como não-estruturada.

- Processamento em Tempo Real: Transformação, Agregação e Filtragem dos Dados.

- Saída de Resultados: Geração de Insights em Tempo Real para outras Aplicações ou Analistas.

# Stream Processing – Ferramentas

- Apache Kafka: Plataforma de streaming distribuída para ingestão e transmissão de dados.
- Apache Flink: Motor de processamento de dados em streaming e em lote de alto desempenho.
- Apache Spark Streaming: Processamento de streaming no ecossistema Apache Spark.
- Dentre outras soluções em nuvem: Amazon Kinesis, Azure Stream Analytics, Google Cloud Dataflow, etc.

# Stream Processing – Aplicações

- Real-Time Social Media Monitoring.
- Network Traffic Analytics (Security).
- Credit Card Fraud Detection and Analysis.
- Server Monitoring (Observability).
- IoT Systems Monitoring.

# Stream Processing – Desafios

- Escalabilidade: Gerenciar os picos de carga de dados.
- Gerenciamento de Estado: Manter o estado do sistema em tempo real.
- Tolerância a Falhas: Construir pipelines robustos à interrupções de rede e erros de processamento.
- Complexidade: Streams são bem mais complexos de se construir e manter.

# Stream Processing – Desafios

- Não há ferramenta que cuida de todos os cenários de processamento de maneira performática.

- Comparativo Interessante entre Ferramentas: https://www.upsolver.com/blog/popular-stream-processing-frameworks-compared

# Plataforma Apache Spark

FIAP

- Definição: O Apache Spark possui um motor de processamento de dados com capacidade de processar grandes volumes de dados de forma rápida e de fácil gerenciamento.

- Criado em 2013 e permite a codificação de pipelines de processamento em múltiplas linguagens de programação, mais notadamente Scala, Java, SQL, Python, R e C#.

# Apache Spark - Características

- Velocidade: Processamento *in-memory* (mais detalhes a seguir) para cargas de trabalho em batch e/ou em streaming.

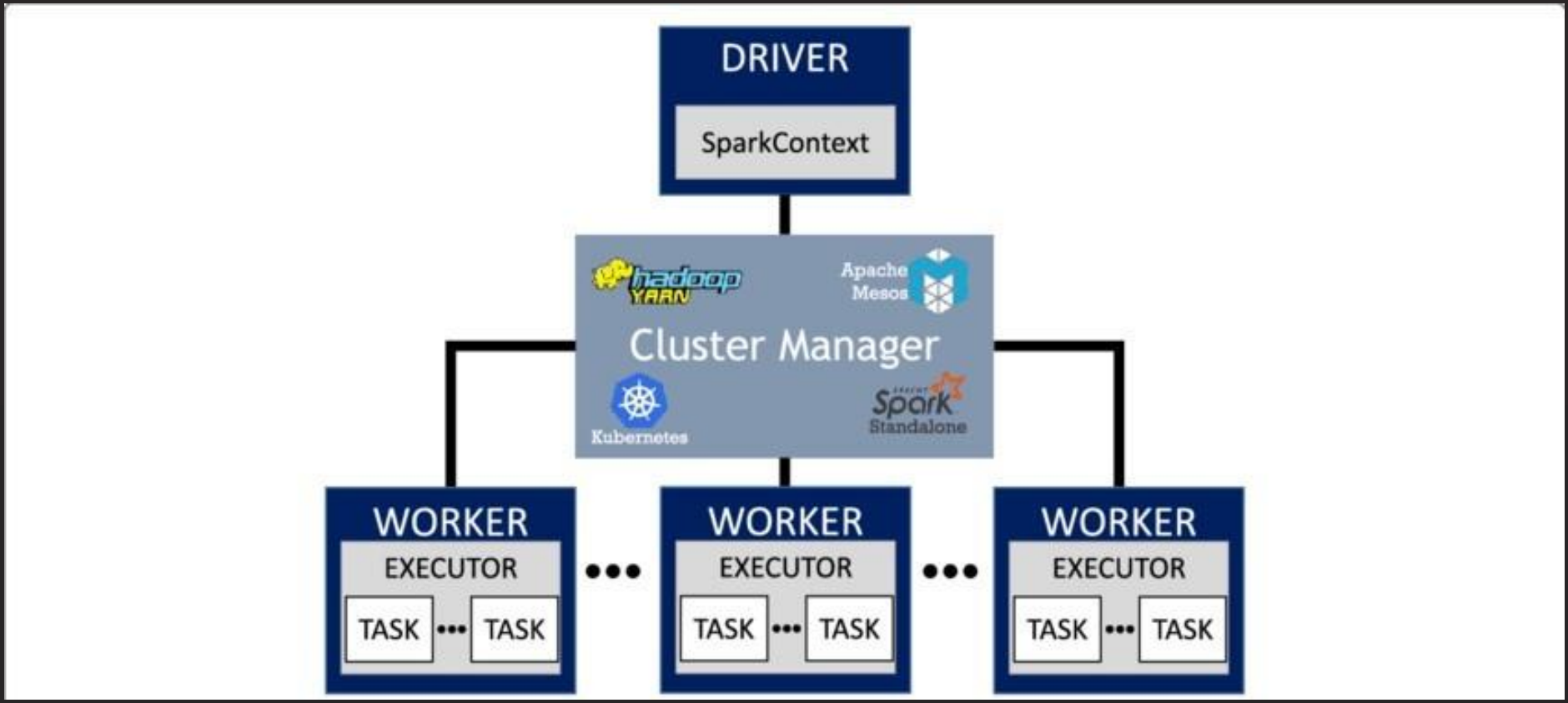
- Escalabilidade: Escala horizontalmente para lidar com volumes de dados massivos (Ideal para cenários de Cloud e Big Data).

- Ecossistema: Integração com Hadoop, MLlib, GraphX, etc.

# Apache Spark - Componentes

- Apache Spark Core: Motor de execução.
- Spark SQL: Módulo de Processamento SQL.
- Spark Streaming: Módulo de Processamento em Tempo Real.
- MLlib: Biblioteca de Machine Learning.
- GraphX: Processamento de Gráficos.

# Apache Spark - Arquitetura




Fonte: ResearchGate

# Apache Spark - Funcionamento

- Processamento em Memória: Minimiza o uso de I/O de disco, aproveitando a memória para maior velocidade e *throughput*.

- Resiliência: Tolerância a falhas por meio do uso do DAG (Directed Acyclic Graph) e recuperação de RDDs.

- API: Oferece suporte a várias linguagens e diversos formatos de data sources.

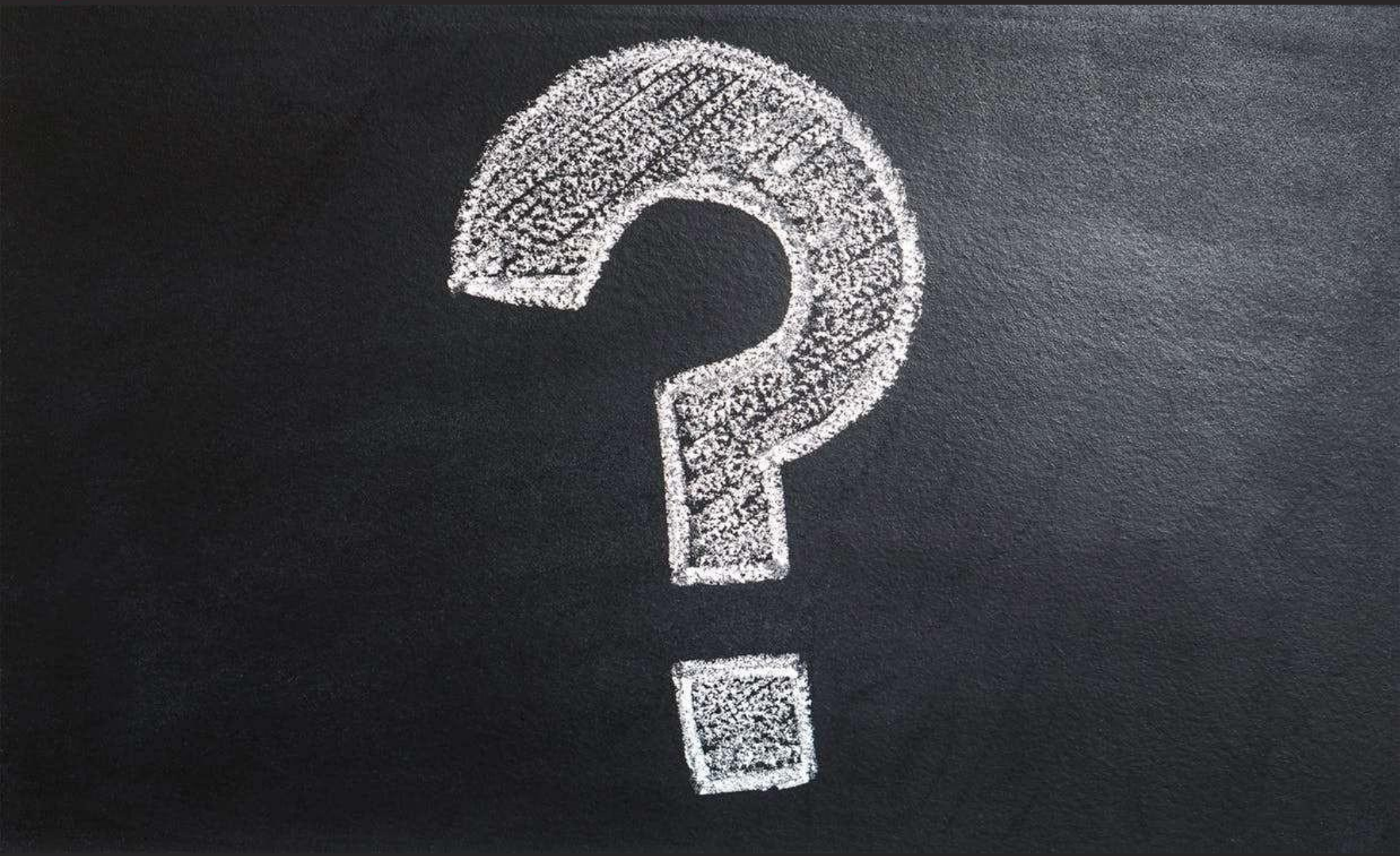
# Apache Spark – Casos de Uso

- Real-Time Data Analytics.
- Processamento de ETL com Carga Massiva de Dados.
- Machine Learning.
- Graph Analytics.
- Processamento de Logs e Eventos.

# Lab Time

- Vamos conhecer um pouco do Apache Spark e suas funcionalidades principais.
- Criar uma carga de dados fake.
- Realizar a ingestão de dados.
- Visualização da carga de dados.

# Dúvidas?



FIAP

MBA+