Olá, aluno(a)!
Seja bem-vindo(a) à aula interativa!

Você entrará na reunião com a câmera e o microfone desligados.

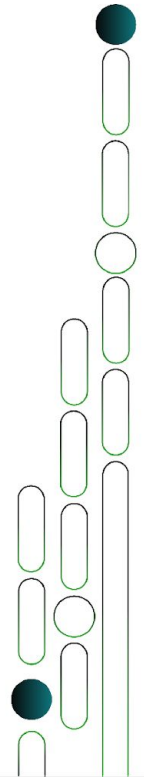
Sua presença será computada através da enquete.
Fique atento(a) e não deixe de respondê-la!

XPe
Aprenda com quem faz

> Bancos de Dados NoSQL

Primeira Aula Interativa

Prof. Ricardo Brito Alves



Nesta aula

- Apresentação do Professor
- Trabalho Prático
- Tópicos da Disciplina e temas interessantes
- Desafio

Apresentação do professor

Ricardo Brito Alves

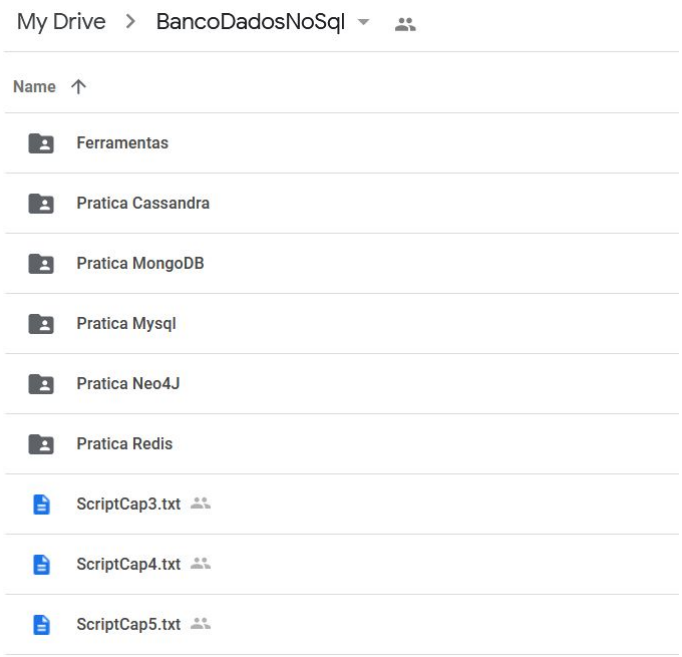
- **Formação Acadêmica:**
- Graduado em Ciência da Computação pela Pontifícia Universidade Católica de Minas Gerais, em 1994.
- MBA em Gestão Estratégica de Projetos pela Una, em 2009.
- Mestrado em Engenharia Elétrica pela Pontifícia Universidade Católica de Minas Gerais, em 2018.
- Doutorando em Ciência da Computação pela Universidade Federal de Minas Gerais, em 2024.
- **Experiência Profissional:**
- Desde 2002, atua com projetos de Data mining e BI.
- Atua há 7 anos na área de Inteligência Artificial.

Trabalho Prático

Tópicos da disciplina e temas interessantes

Material

https://drive.google.com/drive/u/1/folders/176_xlNV4obL2fbVeKFh9oKqpzv6_dfPp



Data Driven

- ***O Data Driven se baseia no uso de ferramentas tecnológicas capazes de coletar e analisar dados diferentes da sua empresa.***
- ***Esses dados, por sua vez, podem ser compilados por meio de BI ou Inteligência Artificial*** e ajudam o gestor a ter uma ideia mais precisa do seu negócio, facilitando a tomada de decisão estratégica.
- O Data Driven surgiu como extensão da Ciência de Dados, campo do conhecimento que utiliza métodos científicos e algoritmos para transformar dados – estruturados e não estruturados – em conhecimento.
- Atualmente, no ambiente corporativo, isso é feito por meio de ferramentas como Big Data, Inteligência Artificial e Machine Learning, para obter insights a partir da coleta, cruzamento e interpretação de dados do mercado.

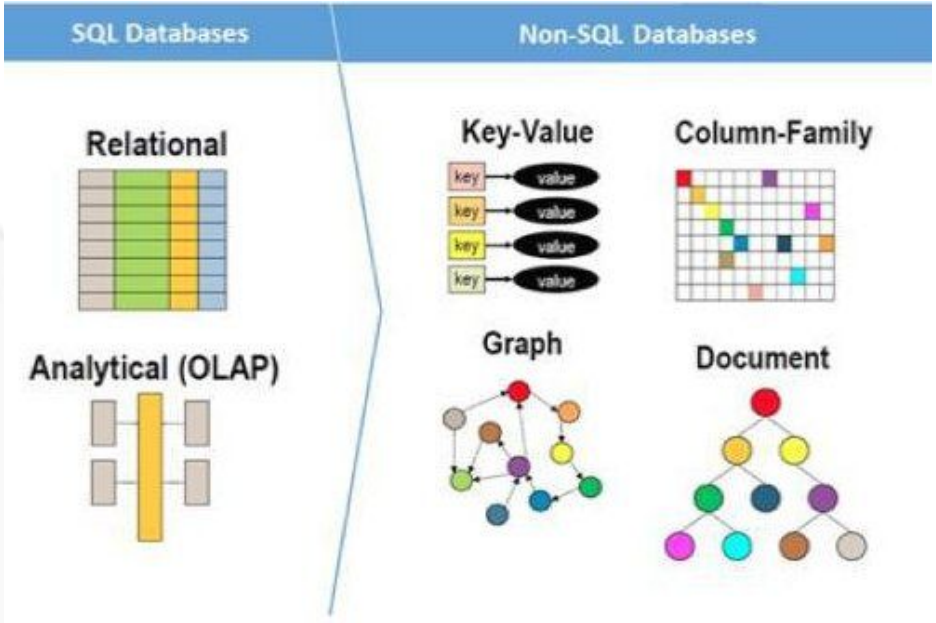
Como adotar o Data Driven?

- O principal objetivo do Data Driven ***é entregar respostas mais precisas e confiáveis por meio de dados.***
- Por isso, o primeiro passo para implementar essa ideia é modificar a postura dos gestores e colaboradores, de forma que eles ***passem a reduzir os "achismos" e entendam o valor desses dados.***
- **Transforme a sua cultura.**
- **Use boas soluções.**
- **Aprenda a entender os dados.**
- **Use os indicadores de performance.**

NoSQL

- ✔ NoSQL (originalmente se referindo a "no SQL": "não SQL" ou "não relacional", posteriormente estendido para ***Not Only SQL*** - Não Somente SQL) é um termo genérico que representa os bancos de dados não relacionais.
- ✔ Bancos de dados NoSQL são cada vez mais usados em ***Big Data*** e ***aplicações web de tempo real***.

Banco de Dados Relacional



Fonte:
https://www.kdnuggets.com/2021/05/nosql-know-it-all-compendium.html

Ranking Banco de Dados

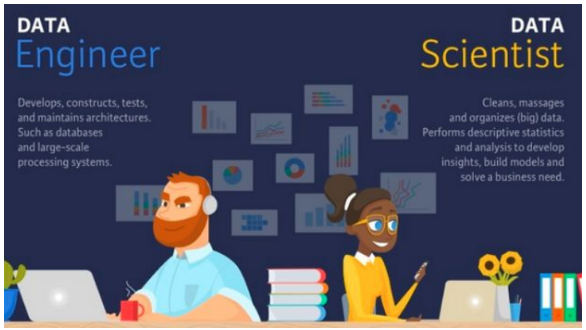
https://db-engines.com/en/ranking

Rank			DBMS	Database Model	Score		
Aug 2020	Jul 2020	Aug 2019			Aug 2020	Jul 2020	Aug 2019
1.	1.	1.	Oracle	Relational, Multi-model	1355.16	+14.90	+15.68
2.	2.	2.	MySQL	Relational, Multi-model	1261.57	-6.93	+7.89
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1075.87	+16.15	-17.30
4.	4.	4.	PostgreSQL	Relational, Multi-model	536.77	+9.76	+55.43
5.	5.	5.	MongoDB	Document, Multi-model	443.56	+0.08	+38.99
6.	6.	6.	IBM Db2	Relational, Multi-model	162.45	-0.72	-10.50
7.	↑ 8.	↑ 8.	Redis	Key-value, Multi-model	152.87	+2.83	+8.79
8.	↓ 7.	↓ 7.	Elasticsearch	Search engine, Multi-model	152.32	+0.73	+3.23
9.	9.	↑ 11.	SQLite	Relational	126.82	-0.64	+4.10
10.	↑ 11.	↓ 9.	Microsoft Access	Relational	119.86	+3.32	-15.47
11.	↓ 10.	↓ 10.	Cassandra	Wide column	119.84	-1.25	-5.37
12.	12.	↑ 13.	MariaDB	Relational, Multi-model	90.92	-0.21	+5.96
13.	13.	↓ 12.	Splunk	Search engine	89.91	+1.64	+4.03
14.	↑ 15.	↑ 15.	Teradata	Relational, Multi-model	76.78	+0.81	+0.14
15.	↓ 14.	↓ 14.	Hive	Relational	75.29	-1.14	-6.51

Fonte: https://db-engines.com/en/ranking

Engenheiro de Dados

- Um Engenheiro de Dados é o profissional dedicado ao ***desenvolvimento, construção, teste e manutenção de arquiteturas, como um sistema de processamento em grande escala.***
- O Engenheiro de Dados é responsável por criar o pipeline dos dados, ***desde a coleta até a entrega*** para análise ou para alimentar um produto ou serviço baseado em análise preditiva já em produção.



Engenheiro de Dados x DBA

- ***As atividades desempenhadas pelo Engenheiro de Dados englobam principalmente a já conhecida rotina de um DBA***, porém acrescidas de muitas outras tarefas, tais como: ***manutenção de sistemas de banco de dados relacionais e não relacionais, ETL (Extração, Transformação e Carga), soluções de Data-Warehouse e Datamart, modelagem de dados e armazenamento em nuvem.***
- Além disso, diversas tecnologias fazem parte do dia a dia desse profissional, tais como: Oracle, MSSQL, MySQL, PostgreSQL, Neo4J, MongoDB, Cassandra, Sqoop, HDFS, Hive e muitas outras.

Cientista de Dados

- Cientistas de Dados ***recebem uma enorme massa de dados (estruturados e não estruturados)*** e usam suas habilidades em Matemática, Estatística, Ciência da Computação e Programação para ***limpar, tratar e organizar os dados.***
- Em seguida, eles aplicam suas capacidades analíticas – Machine Learning, Inteligência Artificial, conhecimento de negócio, ceticismo de suposições existentes – ***para descobrir soluções para os desafios de negócios.***

Business Intelligence x Analytics

Área	Analista de BI	Cientista de Dados
Foco	Relatórios, KPI's, Tendências	Padrões, Correlações, Modelos Preditivos
Processo	Estático, Comparativo	Exploratório, Experimental, Visual
Fontes de Dados	Data Warehouses, Bancos Transacionais	Big Data, Dados Não-Estruturados, Bancos Transacionais e NoSQL, Dados Gerados em Tempo Real
Qualidade dos Dados na Fonte	Alta	Baixa ou Média (requer processo de limpeza e transformação)
Modelo de Dados	Esquema de dados bem definido na fonte	Esquema de dados definido no momento da consulta
Transformações nos Dados	Pouca ou nenhuma (dados já organizados na fonte)	Transformação sob demanda, necessidade de complementar os dados
Análise	Descritiva, Retrospectiva	Preditiva, Prescritiva
Responde à pergunta:	O que aconteceu?	O que pode acontecer?

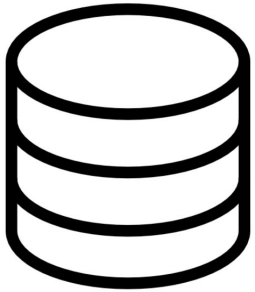
Propriedades ACID

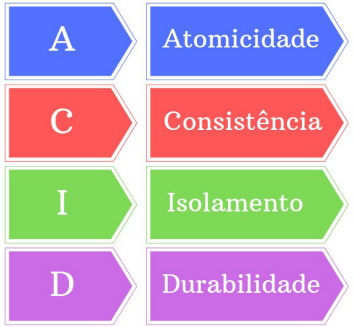
Atomicidade – estado em que as modificações no BD devem ser todas ou nenhuma feita. Cada transação é dita como “atômica”. Se uma parte desta transação falhar, toda transação falhará.

Consistência – estado que garante que todos os dados serão escritos no BD.

Isolamento – requer que múltiplas transações que estejam ocorrendo “ao mesmo tempo” não interfiram nas outras.

Durabilidade – garante que toda transação submetida (commit) pelo BD não será perdida.





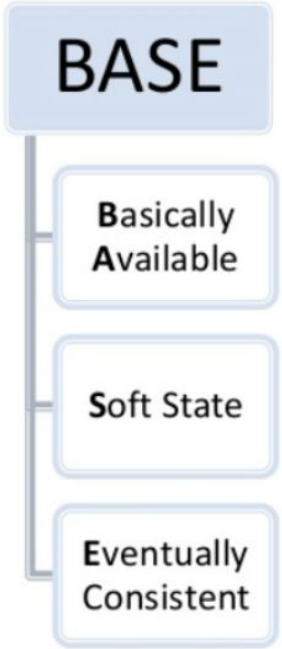
Propriedades BASE

Basically Available – Basicamente Disponível.

Soft-State – Estado Leve.

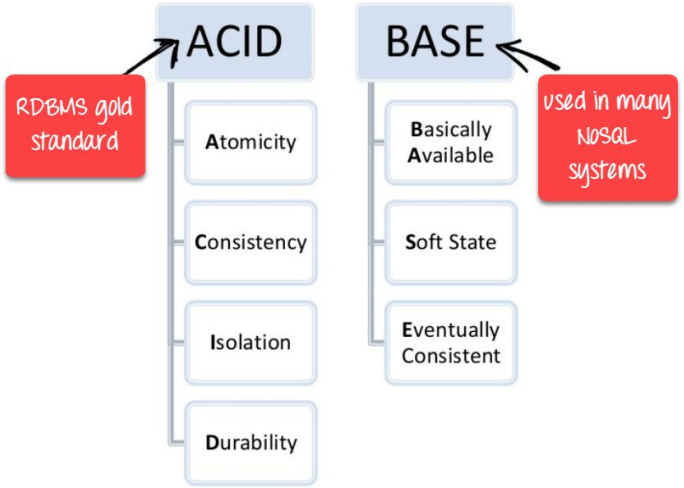
Eventually Consistent – Eventualmente Consistente.

Uma aplicação funciona basicamente todo o tempo (Basicamente Disponível), não tem de ser consistente todo o tempo (Estado Leve) e o sistema torna-se consistente no momento devido (Eventualmente Consistente).



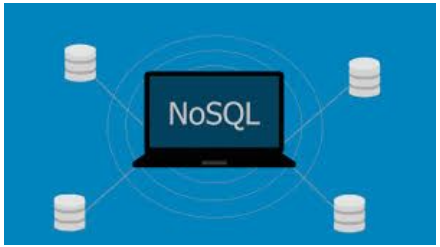
ACID vs BASE

- **BDs Relacionais** trabalham com **ACID** (Atomicidade, Consistência, Isolabilidade, Durabilidade).
- **BDs NoSQL** trabalham com **BASE** (Basicamente Disponível, Estado Leve, Eventualmente consistente).



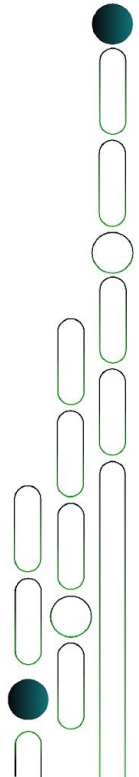
NoSQL - Características

- Escalabilidade horizontal.
- Ausência de esquema ou esquema flexível.
- Suporte a replicação.
- API simples.
- Nem sempre prima pela consistência.



Schemas de Dados

- Uma estrutura de banco de dados funcional não é definível sem entender a estrutura dos dados que devem ser ingeridos no banco de dados.
- Você deve definir o esquema do banco de dados para adequá-lo aos dados.
- Ou ...



Schema-on-Write

- Essa construção está fortemente vinculada ao gerenciamento de Banco de Dados Relacional, incluindo o esquema e a criação de tabelas e a ingestão de dados.
- O problema aqui é que os dados não podem ser carregados nas tabelas sem os esquemas e tabelas **previamente criados e configurados.**

Fonte: https://medium.com/@deepa.account/schema-on-read-and-schema-on-write-ac72524926b

Schema on-Read

- O esquema do banco de dados é criado quando os dados são lidos.
- As estruturas de dados não são aplicadas ou iniciadas antes que os dados sejam ingeridos no banco de dados; eles são criados durante o processo ETL.
- Isso permite que dados não estruturados sejam armazenados no banco de dados.
- A principal razão para desenvolver o princípio do esquema na leitura é o crescimento explosivo dos volumes de dados não estruturados e a alta sobrecarga envolvida durante o processo do esquema na gravação.

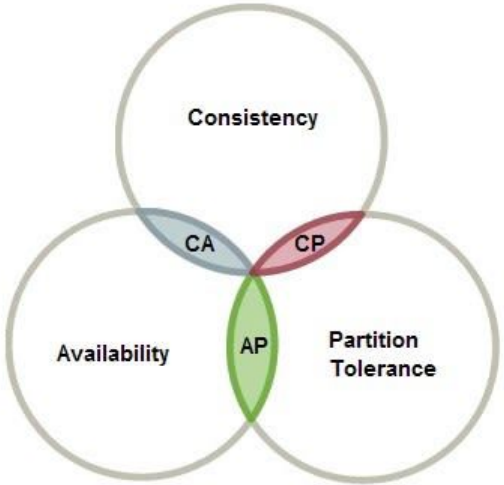
Fonte: https://medium.com/@deepa.account/schema-on-read-and-schema-on-write-ac72524926b

Escalabilidade Horizontal

- ***À medida em que o volume de dados cresce***, aumenta-se a necessidade de escalabilidade e melhoria do desempenho.
- Podemos escalar ***verticalmente*** (adicionar CPU, memória e disco) ou podemos escalar ***horizontalmente*** (adicionar mais nós).

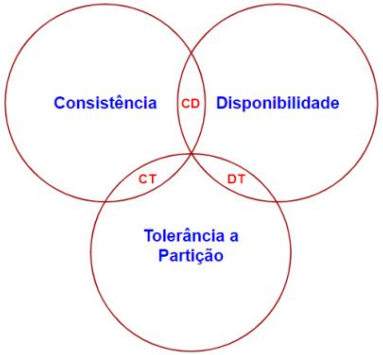
Teorema CAP

- De acordo com o teorema **CAP**, ***um sistema distribuído de bancos de dados somente pode operar com dois desses comportamentos ao mesmo tempo***, mas ***jamais com os três simultaneamente***.

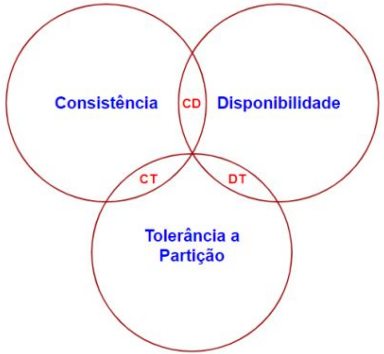


Consistência Eventual

- É um conceito interessante derivado do teorema CAP.
- ***O sistema prioriza as escritas de dados (armazenamento)***, sendo o sincronismo entre os nós do servidor realizado em um momento posterior – o que causa um pequeno intervalo de tempo, no qual o sistema como um todo é inconsistente.
- Para isso, ***são implementadas as propriedades Disponibilidade e Tolerância a Partição***.

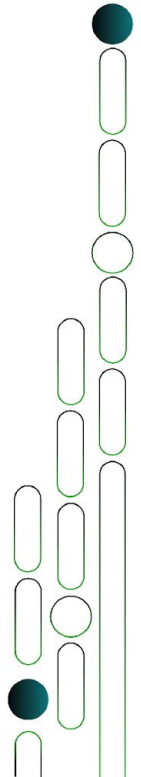


Consistência Eventual

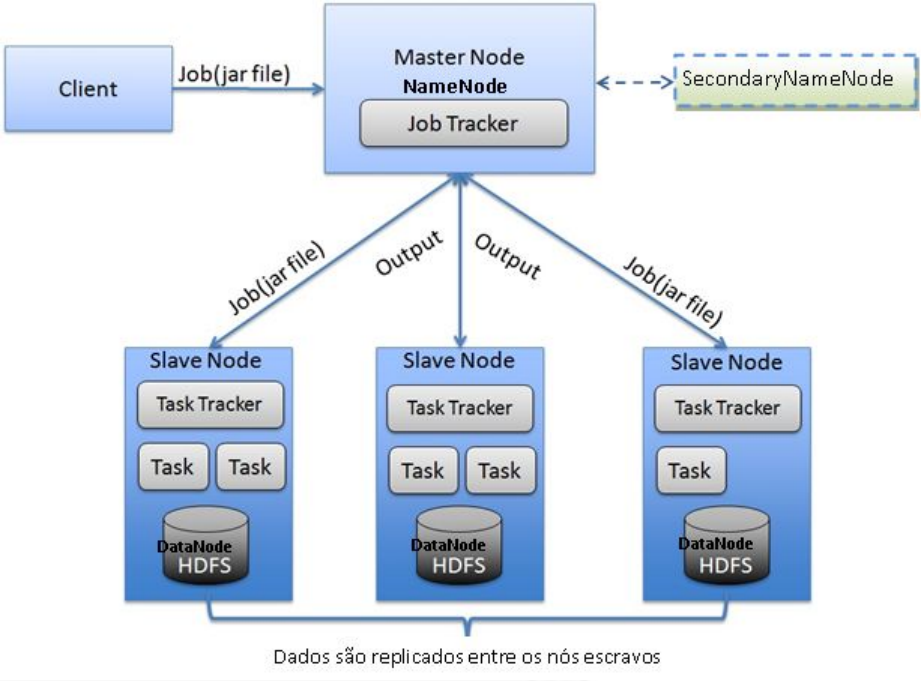


- Exemplos de sistemas de bancos de dados que implementam a consistência eventual são o ***MongoDB, Cassandra e RavenDB (bancos NoSQL),*** entre outros.
- Em Bancos Relacionais, é muito comum implementar as propriedades ***Consistência*** e ***Disponibilidade***. Como exemplos, citamos os SGBDRs ***Oracle, MySQL, PostgreSQL, SQL Server*** e outros.
- Ao criar um banco de dados distribuído, é importante ***ter em mente o teorema CAP***.
- ***Você terá que decidir se o banco será consistente ou disponível, pois bancos de dados distribuídos são sempre tolerantes à partição.***

Apache Hadoop



Hadoop



Fonte:
https://www.devmedia.com.br/hadoop-fundamentos-e-instalacao/29466

Apache Hadoop

Hadoop é uma ***plataforma de software de código aberto para o armazenamento e processamento distribuído de grandes conjuntos de dados, utilizando clusters de computadores com hardware commodity.***

Os serviços do Hadoop fornecem armazenamento , processamento, acesso, governança, segurança e operações de Dados.

Benefícios do Apache Hadoop

Algumas das razões para se usar Hadoop é a sua "capacidade de armazenar, gerenciar e analisar grandes quantidades de dados estruturados e não estruturados de forma rápida, confiável, flexível e de baixo custo."

- **Escalabilidade e desempenho** – distribuídos tratamento de dados local para cada nó em um cluster Hadoop permite armazenar, gerenciar, processar e analisar dados em escala petabyte.

- **Confiabilidade** – clusters de computação de grande porte são propensos a falhas de nós individuais no cluster. Hadoop é fundamentalmente resistente – quando um nó falha de processamento é redirecionado para os nós restantes no cluster e os dados são automaticamente re-replicado em preparação para falhas de nós futuras.

Benefícios do Apache Hadoop

- **Flexibilidade** – ao contrário de sistemas de gerenciamento de banco de dados relacionais tradicionais, você não tem que esquemas estruturados criados antes de armazenar dados. ***Você pode armazenar dados em qualquer formato, incluindo formatos semi-estruturados ou não estruturados***, e em seguida, analisar e aplicar esquema para os dados quando ler.

- **Baixo custo** – ao contrário de software proprietário, ***o Hadoop é open source*** e é executado em hardware commodity de baixo custo.

HDFS

O HDFS (Hadoop Distributed File System) é um sistema de arquivos distribuído, ***projetado para armazenar arquivos muito grandes***, com padrão de acesso aos dados streaming , utilizando clusters de servidores facilmente encontrados no mercado e de baixo ou médio custo.

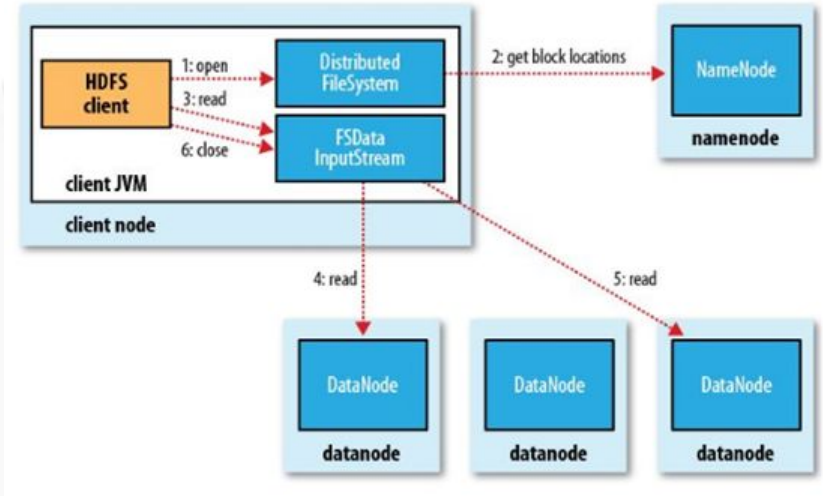
Não deve ser utilizado para aplicações que precisem de acesso rápido a um determinado registro e sim para aplicações nas quais é necessário ler uma quantidade muito grande de dados.

Outra questão que deve ser observada é que ***não deve ser utilizado para ler muitos arquivos pequenos***, tendo em vista o overhead de memória envolvido.

Recursos do HDFS

- O HDFS tem 2 tipos de Nós : **Master** (ou Namenode) e **Worker** (ou Datanode).
    - O **Master** armazena informações da distribuição de arquivos e metadados.
    - Já o **Worker** armazena os dados propriamente ditos. Logo o Master precisa sempre estar disponível. Para garantir a disponibilidade podemos ter um backup (similar ao Cold Failover) ou termos um Master Secundário em um outro servidor. Nesta segunda opção, em caso de falha do primário, o secundário pode assumir o controle muito rapidamente.
- Tal como um sistema Unix, é possível utilizar o HDFS via linha de comando.

HDFS

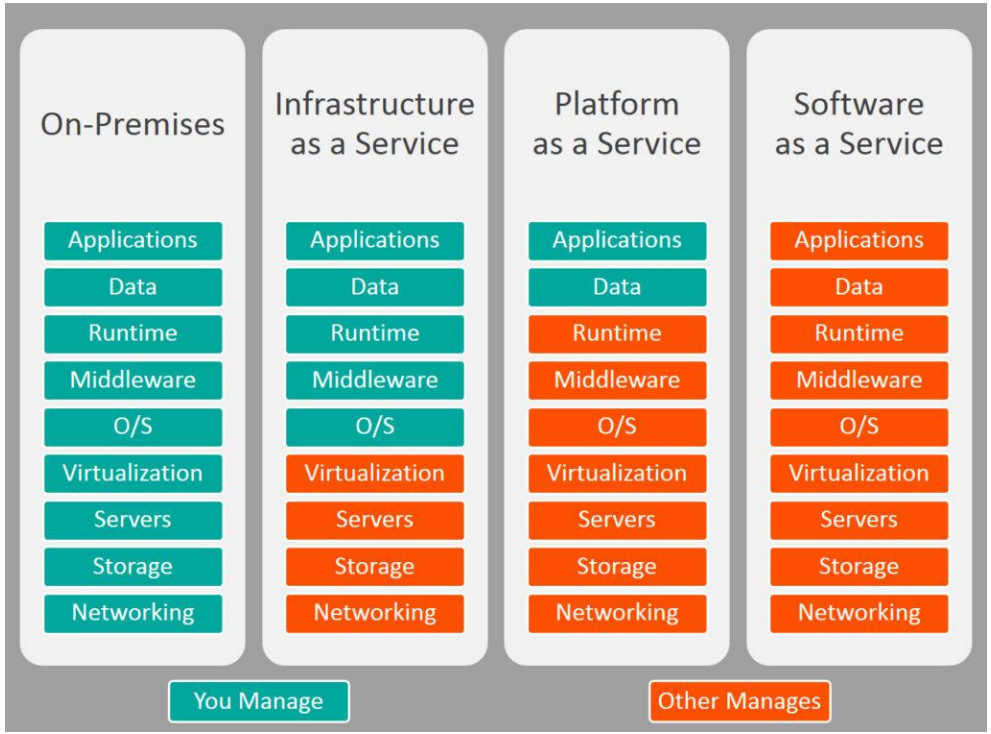


Fonte: http://bigdatabrazil.blogspot.com/2013/07/o-que-e-o-hdfs.html

- Hadoop é uma plataforma de software em Java de computação distribuída voltada para clusters e processamento de grandes volumes de dados, com atenção a tolerância a falhas.
- Foi inspirada no MapReduce (um modelo de programação paralela para processamento largamente distribuído de grandes volumes de dados proposto primeiramente pela empresa Google) e no GoogleFS (Google File System é um sistema de arquivos distribuído proprietário desenvolvido pelo Google para fornecer acesso eficiente e confiável aos dados usando grandes clusters de hardware comum).
- É disponibilizado pela Amazon e IBM em suas plataformas.

- **Hadoop Common** - Contém as bibliotecas e arquivos comuns e necessários para todos os módulos Hadoop.
- **Hadoop Distributed File System (HDFS)** - Sistema de arquivos distribuído que armazena dados em máquinas dentro do cluster, sob demanda, permitindo uma largura de banda muito grande em todo o cluster.
- **Hadoop Yarn** - Trata-se de uma plataforma de gerenciamento de recursos responsável pelo gerenciamento dos recursos computacionais em cluster, assim como pelo agendamento dos recursos.
- **Hadoop MapReduce** - Modelo de programação para processamento em larga escala.

Computação em Nuvem



Computação em Nuvem

Nós temos três principais fornecedores de computação em nuvem, ***AWS, Microsoft Azure e Google Cloud,*** que possuem pontos fortes e fracos que os tornam ideais para diferentes cenários.

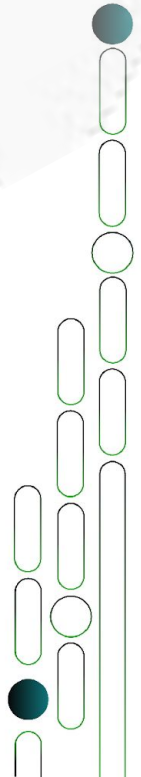
Fonte:
https://allinfo.tech/services/aws-azure-gcp

Arquiteturas Monolíticas

Com as arquiteturas monolíticas, todos os processos são altamente acoplados e executam como um único serviço.

Isso significa que se um processo do aplicativo apresentar um pico de demanda, toda a arquitetura deverá ser escalada. A complexidade da adição ou do aprimoramento de recursos de aplicativos monolíticos aumenta com o crescimento da base de código. Essa complexidade limita a experimentação e dificulta a implementação de novas ideias.

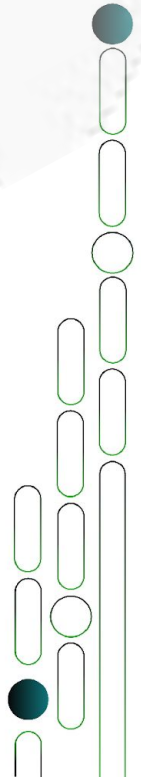
As arquiteturas monolíticas aumentam o risco de disponibilidade de aplicativos, pois muitos processos dependentes e altamente acoplados aumentam o impacto da falha de um único processo.



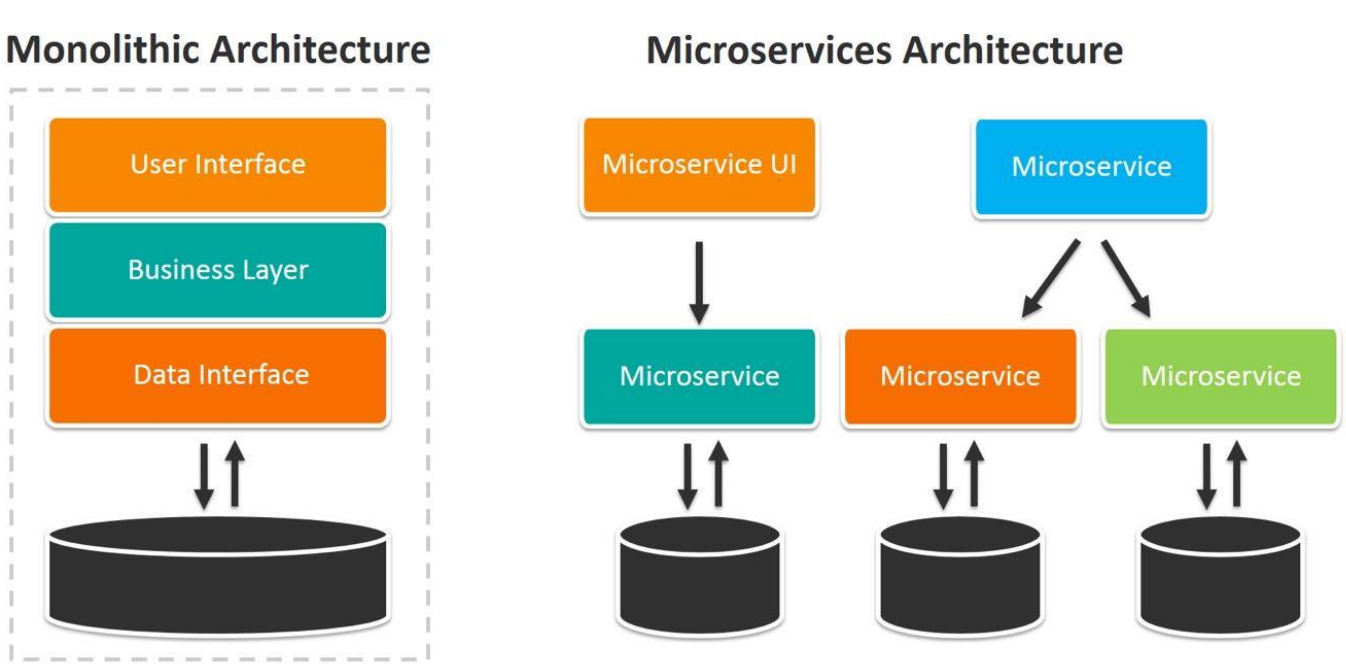
Arquitetura de Microserviços

Com uma arquitetura de microserviços, um aplicativo é criado como componentes independentes que executam cada processo do aplicativo como um serviço.

Esses serviços se comunicam por meio de uma interface bem definida usando APIs leves. Os serviços são criados para recursos empresariais e cada serviço realiza uma única função. Como são executados de forma independente, cada serviço pode ser atualizado, implantado e escalado para atender a demanda de funções específicas de um aplicativo.



Arquitetura



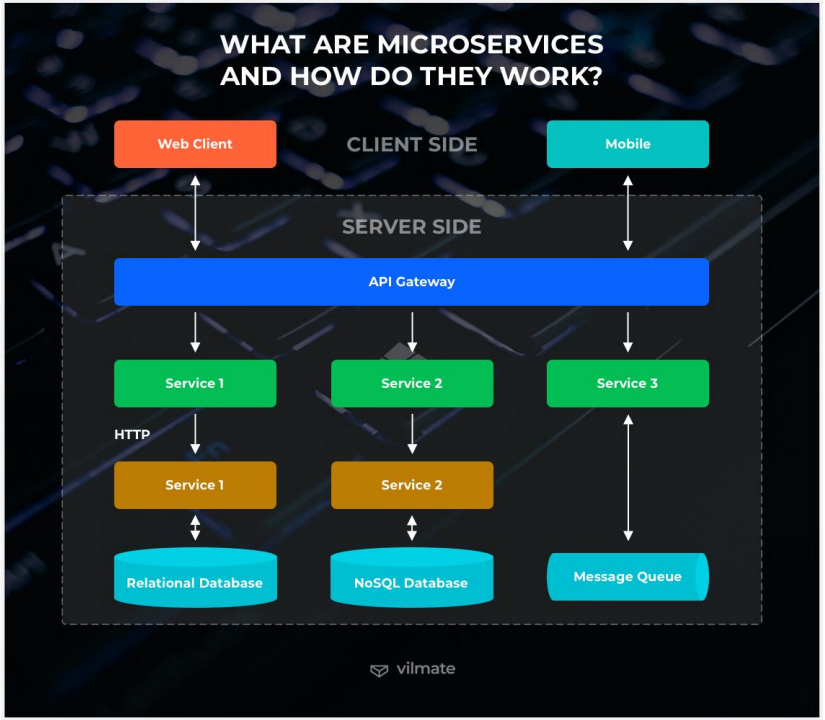
Fonte:
https://www.suse.com/c/rancher_blog/microservices-vs-monolithic-architectures/

Arquitetura de Microserviços

Escalabilidade flexível

Os microserviços permitem que cada serviço seja escalado de forma independente para atender à demanda do recurso de aplicativo oferecido por esse serviço. Isso permite que as equipes dimensionem corretamente as necessidades de infraestrutura, meçam com precisão o custo de um recurso e mantenham a disponibilidade quando um serviço experimenta um pico de demanda.

Arquitetura



Fonte: https://vilmate.com/blog/benefits-of-microservices-architecture/

Data Lake

	Data Warehouse	Data Lake
Dados	· Estruturados · Processados	· Estruturados / Semi-estruturados / Não estruturados · Não processados (em estado bruto)
Processamento	· Esquema de dados gerado no momento da escrita	· Esquema de dados gerado no momento da leitura
Armazenamento	· Alto custo para alto volume de dados	· Criado para ser de baixo custo, independente do volume de dados
Agilidade	· Pouco ágil, configuração fixa	· Bastante ágil, pode ser configurado e reconfigurado conforme necessário
Segurança	· Estratégias de segurança bastante maduras	· Ainda precisa aperfeiçoar o modelo de segurança e acesso aos dados
Usuários	· Analistas de Negócios	· Cientistas e Analistas de Dados

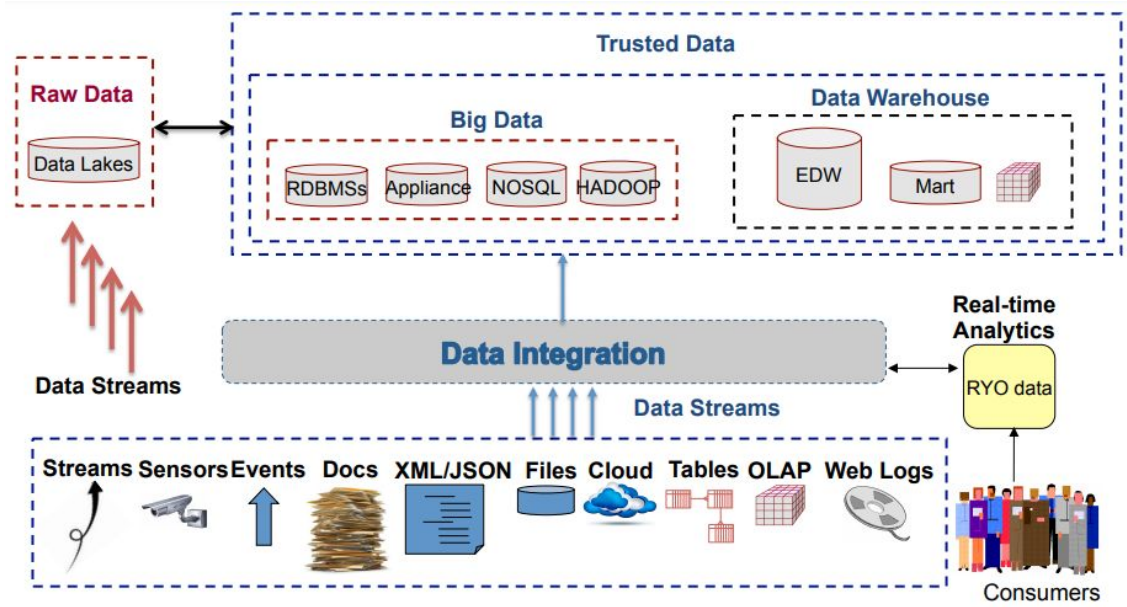
Data Lake NoSQL

- Um Data Lake pode residir em Hadoop, NoSQL, Amazon Simple Storage Service, Banco de Dados Relacional, ou combinações diferentes deles.
- Alimentado por fluxos de dados (data streams).
- Data Lake tem muitos tipos de elementos de dados, estruturas de dados e metadados no HDFS sem levar em conta a importância, IDs ou resumos e agregados.

Data Lake NoSQL

- Importante entender a natureza variada dos dados do Data Lake em relação ao metamodelo do banco de dados perspectiva, NoSQL:
- Semiestruturado.
- Chave: valor (principalmente) com sua estrutura hierárquica.
- A chave e o nome da coluna são partes essenciais da maioria do NoSQL.
- Mais frequentemente, um Data Lake é mantido no Hadoop e alimentado de ou para NoSQL:
- NoSQL é um armazenamento de dados operacional, não analítico.

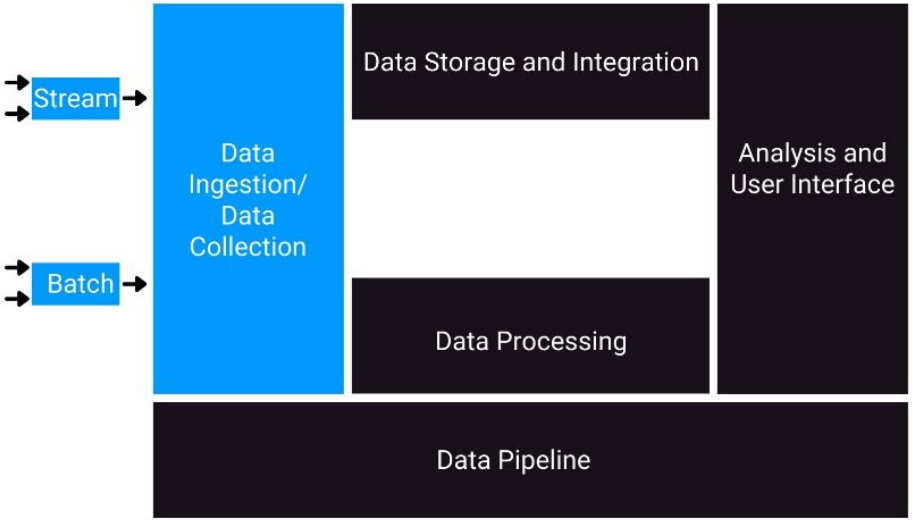
Arquitetura de Dados Moderna



Ingestão / Coleta de Dados

Esta é a primeira camada da arquitetura da plataforma de dados.

A camada de coleta de dados, como o nome sugere, é responsável por conectar-se aos sistemas de origem e trazer dados para a plataforma de dados de maneira periódica.



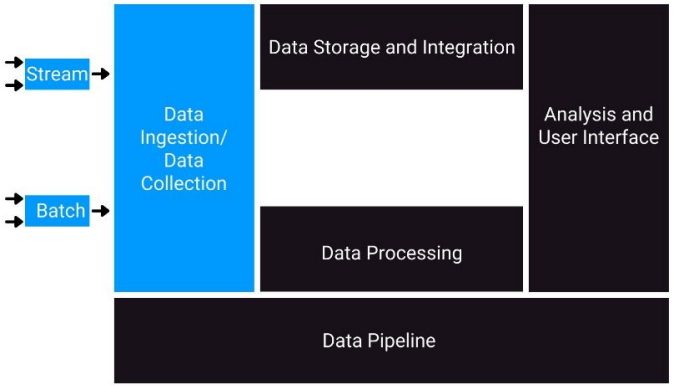
Fonte: https://www.analyticsvidhya.com

Batch e Stream

Qual é a diferença entre batch e stream?

O batch é um lote de pontos de dados que foram agrupados em um intervalo de tempo específico. Outro termo frequentemente usado para isso é uma janela de dados.

Já o processamento de dados em stream lida com dados contínuos e é essencial para transformar de grandes a rápidos.



Fonte: https://www.analyticsvidhya.com

Desafio