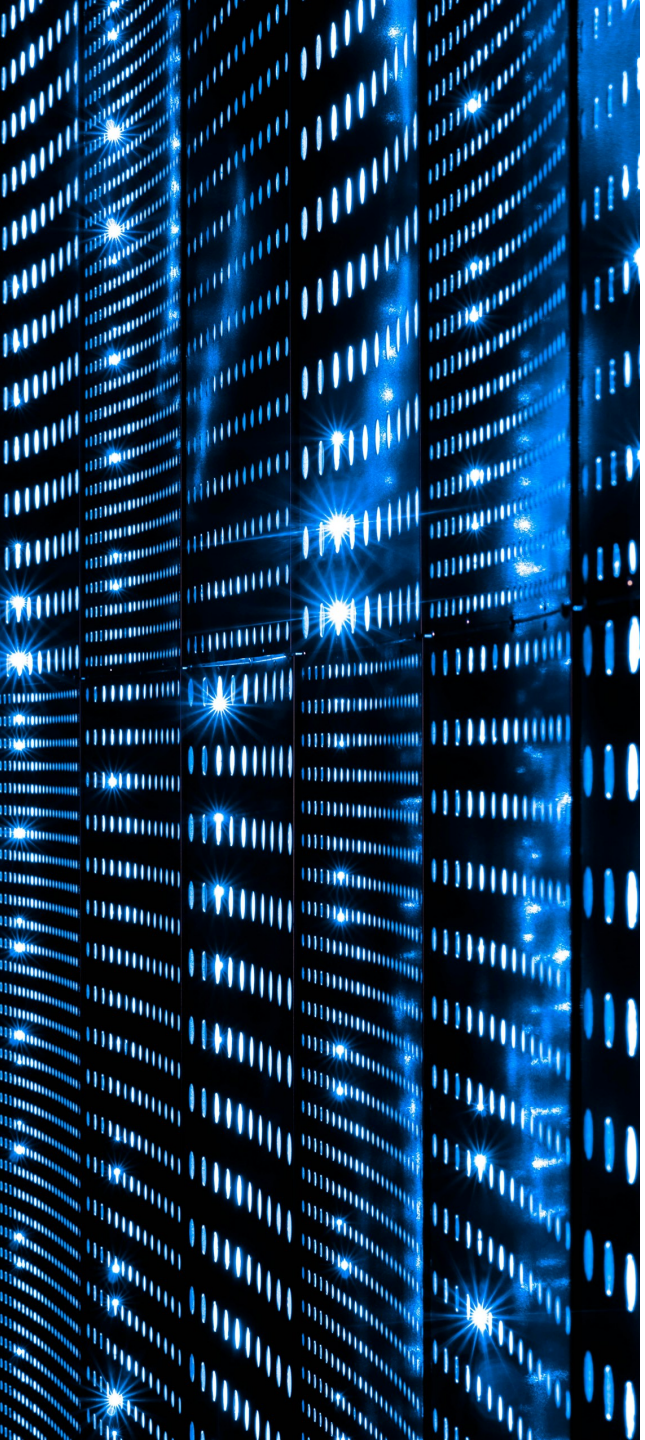


Gerenciamento de Dados na Nuvem

Computação em Nuvem

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Introdução: A Era dos Dados na Nuvem

- **Definição:** Armazenamento em nuvem é a entrega de capacidade de armazenamento sob demanda, gerenciada por um provedor via internet.
- **Contexto Histórico:** O armazenamento de dados foi a primeira manifestação da nuvem pública, com o lançamento do Amazon S3 em 2006.
- **Cenário Atual:** Os sistemas de armazenamento coletivamente suportam uma vasta gama de modelos de dados, desde objetos não estruturados a tabelas relacionais.
- **Importância:** A escolha do modelo depende da natureza, volume, frequência de acesso, escalabilidade e requisitos de latência e segurança dos dados.

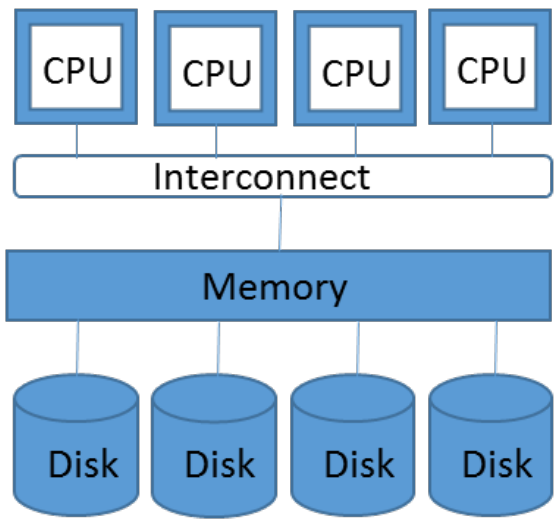
Modelos Fundamentais de Armazenamento

- **Sistemas de Arquivos (File Systems):** Dados organizados em hierarquias de diretórios.
- **Object/Blob Storage:** Modelo de objeto **flat** (plano) para dados binários não estruturados (**Binary Large Objects** - Blobs).
  - Extremamente escalável para petabytes ou exabytes de dados.
  - Objetos são imutáveis após a criação, facilitando implementações altamente escaláveis e confiáveis.
  - Exemplos: Amazon S3, Azure Blob Storage, Google Cloud Storage.
- **Databases (Relacionais e NoSQL):** Coleções de dados estruturados e suas relações.
  - **SQL (Relacional):** Suporta a álgebra relacional e propriedades **ACID** (Atomicidade, Consistência, Isolamento, Durabilidade).
  - **NoSQL (Não-Relacional):** Frequentemente projetados para escalabilidade horizontal, lidando com dados semi-estruturados.

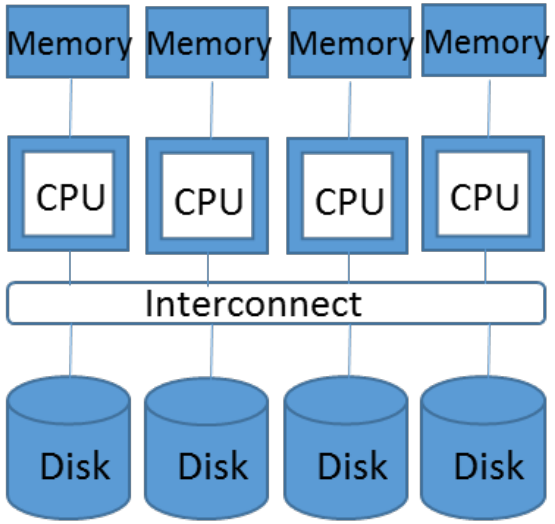
Tipos de Armazenamento em Cloud: Comparativo

Característica	Block Storage (Ex: EBS)	File Storage (Ex: EFS)	Object Storage (Ex: S3)
Estrutura	Blocos de dados com endereço único, tratados como discos.	Arquivos organizados em hierarquias de pastas.	Objetos **flat** com dados, metadados e ID único.
Acesso	Acessado pela VM como disco local (baixa latência).	Protocolos de rede (NFS, SMB).	APIs REST ou HTTP.
Performance	Alta performance, baixa latência. Ideal para I/O frequente (Bancos de Dados Transacionais).	Desempenho moderado. Adequado para compartilhamento.	Otimizado para armazenamento massivo, mas não ideal para leitura/escrita frequente.
Escalabilidade	Limitada ao tamanho do volume.	Limitada pela capacidade do servidor de arquivos.	Altamente Escalável (Petabytes).

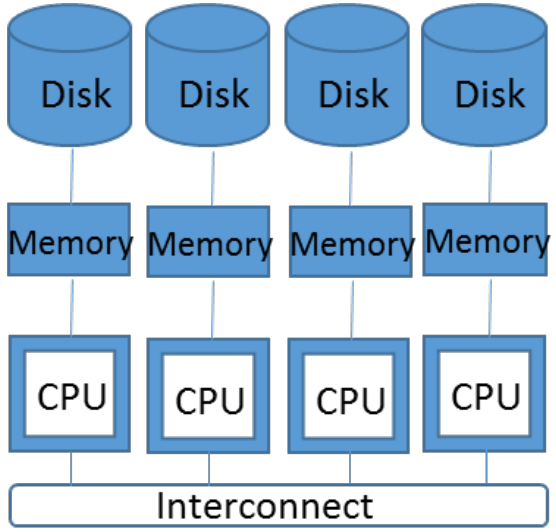
Arquitetura de Bancos de Dados Distribuídos



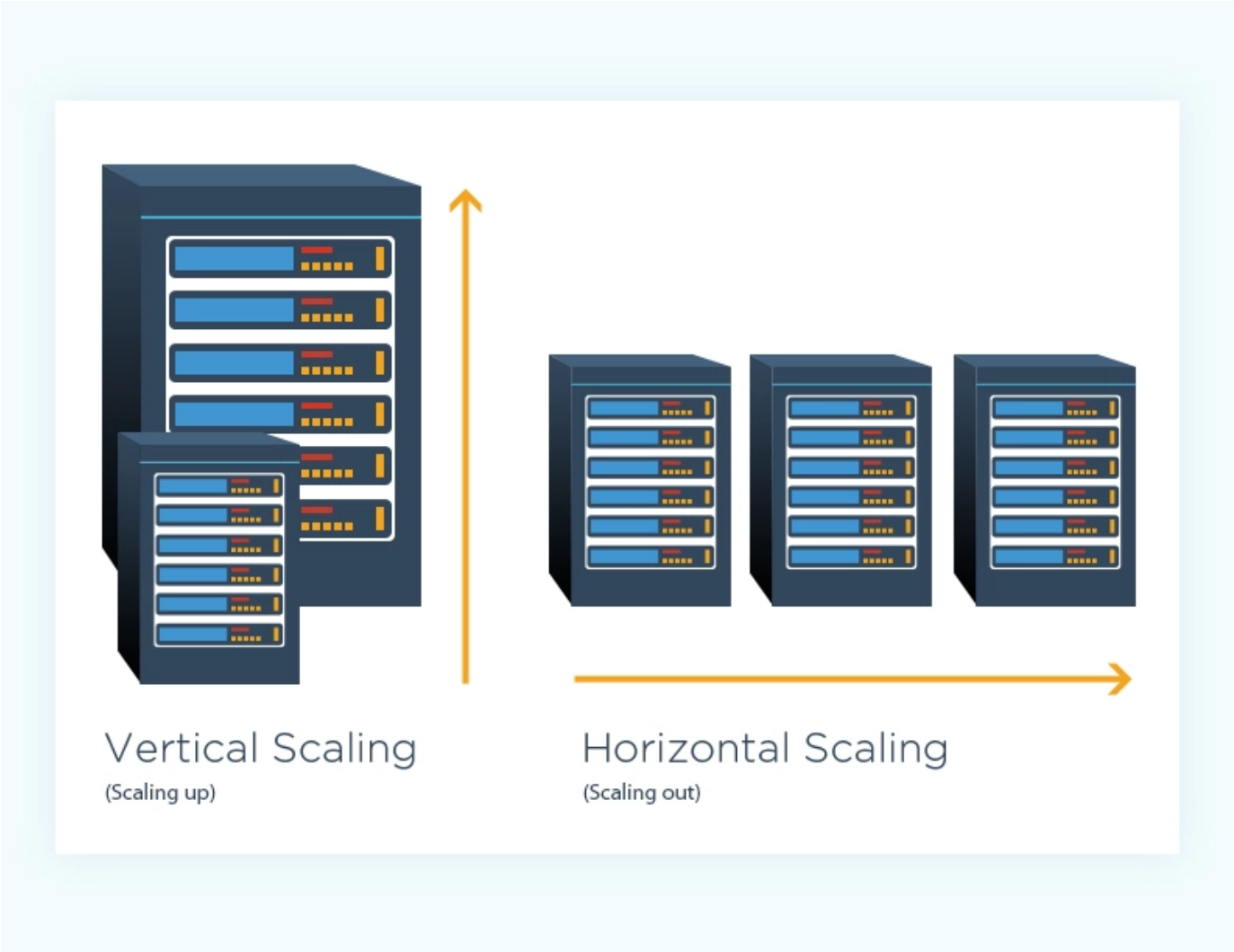
Shared memory architecture



Shared disk architecture



Shared nothing architecture

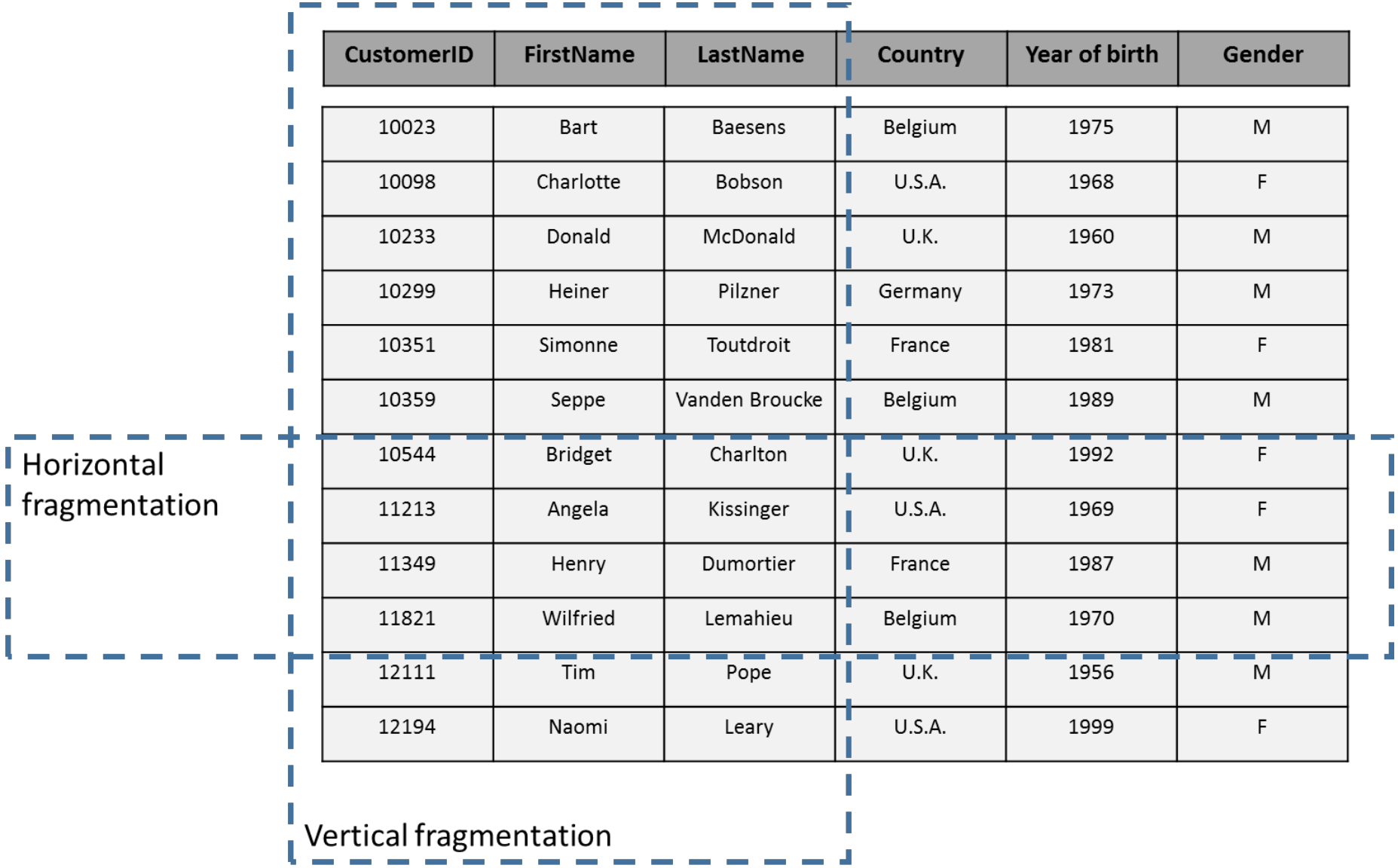


Fonte: CloudZero.

Fragmentação e Alocação de Dados

Fragmentação (Fragmentation): Particionamento de dados em subconjuntos (fragmentos) com base em metas de desempenho, autonomia local e disponibilidade.

- **Fragmentação Vertical:** O fragmento consiste em um subconjunto das colunas da tabela.
  - A visão global é reconstruída com uma operação `JOIN`.
  - Útil se apenas alguns atributos de uma tupla são relevantes para um nó específico.
- **Fragmentação Horizontal (Sharding):** O fragmento consiste em linhas que satisfazem um predicado de consulta.
  - Comum em bancos de dados NoSQL.
  - A visão global é reconstruída com uma operação `UNION`.
- **Fragmentação por Hash (PartitionKey):** Em sistemas NoSQL, o `PartitionKey` é aplicado a um mecanismo de **consistent hashing** para determinar o dispositivo físico de armazenamento.



CustomerID	FirstName	LastName	Country	Year of birth	Gender
10023	Bart	Baesens	Belgium	1975	M
10098	Charlotte	Bobson	U.S.A.	1968	F
10233	Donald	McDonald	U.K.	1960	M
10299	Heiner	Pilzner	Germany	1973	M
10351	Simonne	Toutdroit	France	1981	F
10359	Seppe	Vanden Broucke	Belgium	1989	M
10544	Bridget	Charlton	U.K.	1992	F
11213	Angela	Kissinger	U.S.A.	1969	F
11349	Henry	Dumortier	France	1987	M
11821	Wilfried	Lemahieu	Belgium	1970	M
12111	Tim	Pope	U.K.	1956	M
12194	Naomi	Leary	U.S.A.	1999	F

CustomerID	FirstName	LastName
10023	Bart	Baesens
10098	Charlotte	Bobson
10233	Donald	McDonald
10299	Heiner	Pilzner
10351	Simonne	Toutdroit
10359	Seppe	Vanden Broucke
10544	Bridget	Charlton
11213	Angela	Kissinger
11349	Henry	Dumortier
11821	Wilfried	Lemahieu
12111	Tim	Pope
12194	Naomi	Leary

CustomerID	Country	Year of birth	Gender
10023	Belgium	1975	M
10098	U.S.A.	1968	F
10233	U.K.	1960	M
10299	Germany	1973	M
10351	France	1981	F
10359	Belgium	1989	M
10544	U.K.	1992	F
11213	U.S.A.	1969	F
11349	France	1987	M
11821	Belgium	1970	M
12111	U.K.	1956	M
12194	U.S.A.	1999	F

CustomerID	FirstName	LastName	Country	Year of birth	Gender
10023	Bart	Baesens	Belgium	1975	M
10359	Seppe	Vanden Broucke	Belgium	1989	M
11821	Wilfried	Lemahieu	Belgium	1970	M
10351	Simonne	Toutdroit	France	1981	F
11349	Henry	Dumortier	France	1987	M

CustomerID	FirstName	LastName	Country	Year of birth	Gender
10299	Heiner	Pilzner	Germany	1973	M

CustomerID	FirstName	LastName	Country	Year of birth	Gender
10544	Bridget	Charlton	U.K.	1992	F
10233	Donald	McDonald	U.K.	1960	M
12111	Tim	Pope	U.K.	1956	M

CustomerID	FirstName	LastName	Country	Year of birth	Gender
11213	Angela	Kissinger	U.S.A.	1969	F
10098	Charlotte	Bobson	U.S.A.	1968	F
12194	Naomi	Leary	U.S.A.	1999	F

CustomerID	FirstName	LastName
10023	Bart	Baesens
10359	Seppe	Vanden Broucke
11821	Wilfried	Lemahieu
10351	Simonne	Toutdroit
11349	Henry	Dumortier

CustomerID	FirstName	LastName
10299	Heiner	Pilzner

CustomerID	FirstName	LastName
10544	Bridget	Charlton
10233	Donald	McDonald
12111	Tim	Pope

CustomerID	FirstName	LastName
11213	Angela	Kissinger
10098	Charlotte	Bobson
12194	Naomi	Leary

CustomerID	Country	Year of birth	Gender
10023	Belgium	1975	M
10098	U.S.A.	1968	F
10233	U.K.	1960	M
10299	Germany	1973	M
10351	France	1981	F
10359	Belgium	1989	M
10544	U.K.	1992	F
11213	U.S.A.	1969	F
11349	France	1987	M
11821	Belgium	1970	M
12111	U.K.	1956	M
12194	U.S.A.	1999	F

Replicação

Replicação: Ocorre quando fragmentos se sobrepõem ou se fragmentos idênticos são alocados a diferentes nós.

- **Vantagens da Replicação:** Melhora a autonomia local, performance, escalabilidade, confiabilidade e disponibilidade.
- **Custo:** A replicação induz **overhead** e complexidade adicionais para manter as réplicas consistentes.

Transparência em Bancos de Dados Distribuídos

A Transparência garante que a aplicação e os usuários sejam confrontados com apenas um único banco de dados lógico, sendo isolados das complexidades da distribuição.

- Transparência de Localização (Location Transparency): Usuários não precisam saber onde os dados residem.
- Transparência de Fragmentação (Fragmentation Transparency): Usuários podem executar consultas globais sem se preocupar com a combinação de fragmentos distribuídos.

Transparência em Bancos de Dados Distribuídos (cont.)

- **Transparência de Replicação (Replication Transparency):** Diferentes réplicas são mantidas consistentes, e as atualizações são propagadas transparentemente.
- **Transparência de Acesso (Access Transparency):** O banco de dados distribuído pode ser acessado e consultado uniformemente, independentemente dos diferentes sistemas de banco de dados e APIs utilizados.
- **Transparência de Transação (Transaction Transparency):** O DBMS executa transações distribuídas como se fossem transações em um sistema autônomo.

Processamento Distribuído de Queries

SUPPLIER (SUPNR, SUPNAME, SUPADDRESS, SUPSTATUS)
PURCHASEORDER (PONR, PODATE, *SUPPLIER*)
PRODUCT (PNR, PNAME, PCOLOR, PWEIGHT, WAREHOUSE, STOCK)
…

Location 1:

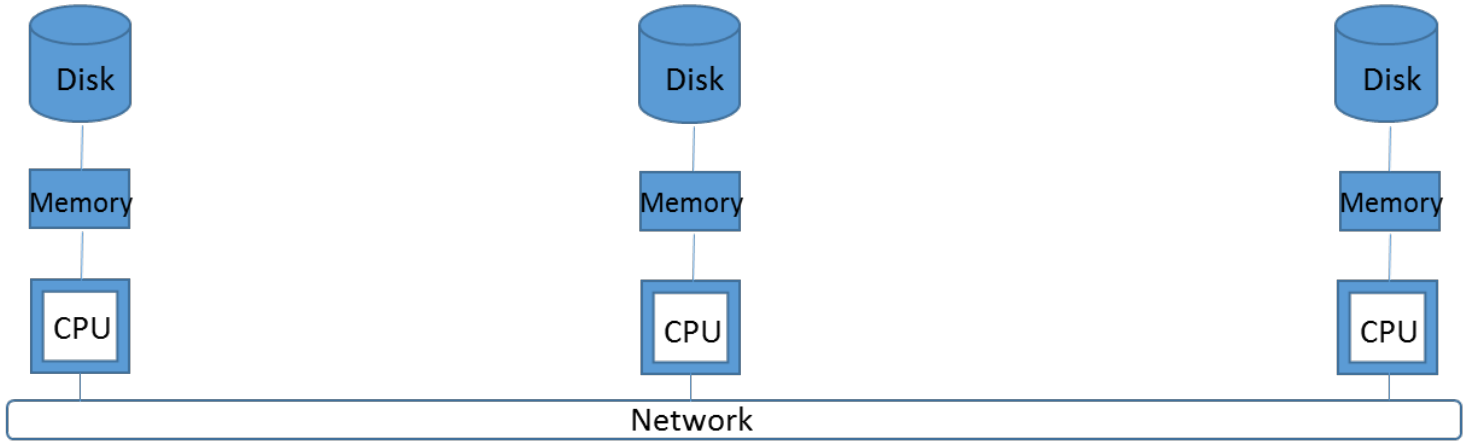
SUPPLIER table
SUPNR: 4 bytes
SUPNAME: 30 bytes
Entire row: 84 bytes
Number of rows: 1000

Location 2:

PURCHASEORDER table
PONR: 6 bytes
SUPPLIER: 4 bytes
Entire row: 16 bytes
Number of rows: 3000
On average, there are 200 suppliers with outstanding purchase orders

Location 3:

Query:
SELECT PONR, SUPNAME
FROM PURCHASEORDER PO, SUPPLIER S
WHERE PO.SUPPLIER = S.SUPNR



Processamento Distribuído de Queries

- Estratégia 1: todas as tabelas copiadas para a localização 3, onde a query será processada.
  - Transporte de dados: (1000 x 84) + (3000 x 16) bytes = 132,000 bytes
- Estratégia 2: Tabela SUPPLIER copiada para localização 2, onde sofrerá join com a tabela PURCHASEORDER. Resultado será enviado para localização 3.
  - Transpore de Dados: (1000 x 84) + (3000 x (6 + 30)) bytes = 192,000 bytes

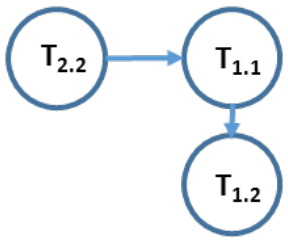
Transaction 1 (T_1)
begin transaction x-lock($account_x$) read($account_x$) $account_x$ = $account_x$ – 50 write($account_x$) x-lock($account_y$) read($account_y$) $account_y$ = $account_y$ + 50 write($account_y$) commit unlock($account_x$, $account_y$)

Transaction 2 (T_2)
begin transaction x-lock($account_y$) read($account_y$) $account_y$ = $account_y$ – 30 write($account_y$) x-lock($account_x$) read($account_x$) $account_x$ = $account_x$ + 30 write($account_x$) commit unlock($account_y$, $account_x$)

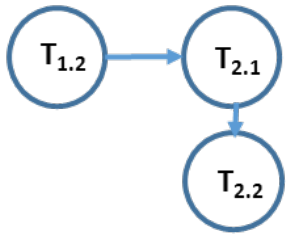
$Account_x$ is stored on Location 1
$Account_y$ is stored on Location 2

	Location 1 (L_1)		Location 2 (L_2)	
Time	$T_{1.1}$	$T_{2.2}$	$T_{2.1}$	$T_{1.2}$
t_1	begin transaction			
t_2	x-lock($account_x$)		begin transaction	
t_3	read($account_x$)		x-lock($account_y$)	
t_4	accountx = $account_x$ – 50		read($account_y$)	
t_5	write($account_x$)		$account_y$ = $account_y$ – 30	
t_6			write($account_y$)	x-lock($account_y$)
t_7		x-lock($account_x$)		wait
t_8		wait		wait

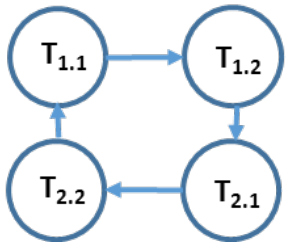
Local wait for graph for L_1



Local wait for graph for L_2



Global wait for graph



Gerenciamento de Transações Distribuídas

- Sistemas Tightly Coupled (Fortemente Acoplados): Interdependência substancial entre participantes. Transações requerem propriedades ACID e tipicamente usam comunicação síncrona.
- Sistemas Loosely Coupled (Frouxamente Acoplados): Interações baseadas em mensagens assíncronas e dados replicados localmente. Frequentemente aplicam **optimistic concurrency** (concorrência otimista) e tipicamente resultam em semântica BASE.

Modelos de Consistência: Fortes e Causais

Definem o contrato entre o armazenamento distribuído e os processos em face da concorrência.

- Consistência Forte (Sequential Consistency): O resultado da execução é o mesmo que se as operações de todos os processos fossem executadas em alguma ordem sequencial. As operações de cada processo individual aparecem nesta sequência na ordem especificada pelo seu programa.
- Consistência Causal: As escritas que são potencialmente causalmente relacionadas devem ser vistas por todos os processos na mesma ordem. Escritas concorrentes podem ser vistas em ordens diferentes por processos distintos.

Consistência Eventual e Transações BASE

Garantir a ordenação global de operações conflitantes é custoso e pode degradar a escalabilidade.

- Consistência Eventual: Na ausência de novas atualizações, todas as atualizações realizadas até aquele ponto são propagadas, de modo que as réplicas eventualmente terão os mesmos dados armazenados.
    - Otimização: Permite evitar a sincronização global, favorecendo a performance.
- Transações BASE (NoSQL): Sistemas NoSQL de larga escala (como DynamoDB ou Bigtable) que abandonam a estrita consistência ACID frequentemente adotam o modelo BASE.

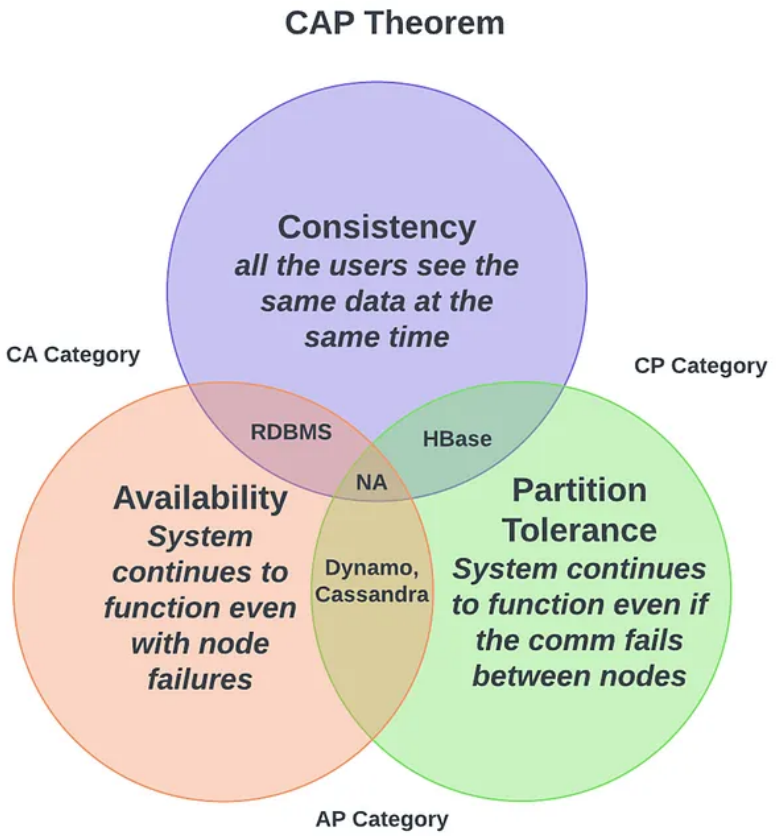
Modelo/Propriedades BASE

- Basically Available: Medidas garantem a disponibilidade sob todas as circunstâncias, se necessário, ao custo da consistência.
- Soft State: O estado pode evoluir (mudar) mesmo sem entrada externa, devido à propagação assíncrona de atualizações.
- Eventually Consistent: O banco de dados se tornará consistente ao longo do tempo, mas pode não ser consistente em qualquer momento, especialmente no commit da transação.

O Teorema CAP

O Teorema CAP afirma que é impossível para um sistema distribuído satisfazer simultaneamente as três propriedades:

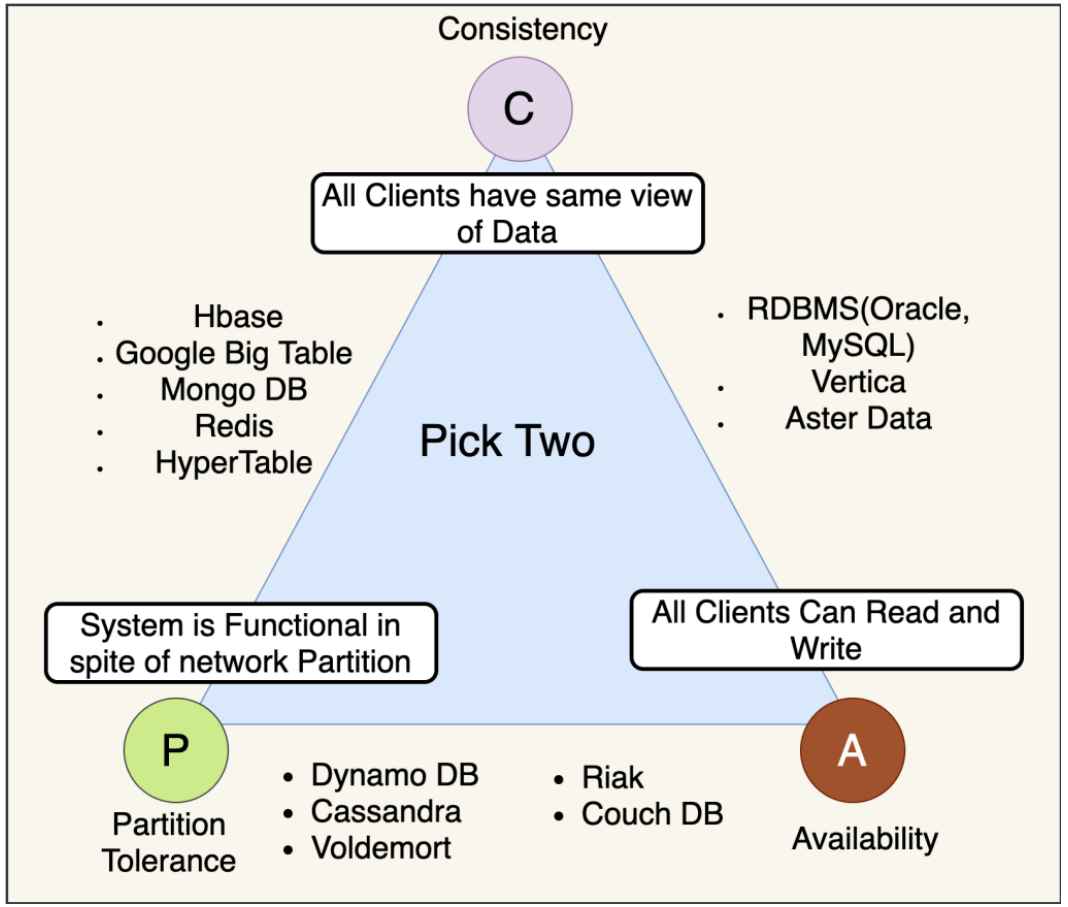
- Consistência (Consistency): Todos os nós veem os mesmos dados ao mesmo tempo.
- Alta Disponibilidade (Availability): Toda requisição recebe uma resposta de sucesso ou falha.
- Tolerância à Partição (Partition tolerance): O sistema continua operando mesmo se falhas de rede impedirem a comunicação entre nós.



O Dilema CAP

1. Escolha CP (Consistência + Tolerância à Partição): Se houver uma partição de rede, o sistema deve garantir a consistência, o que significa que ele pode se tornar indisponível para algumas requisições até que a partição seja resolvida (Ex: Bancos de Dados Relacionais Distribuídos).
2. Escolha AP (Disponibilidade + Tolerância à Partição): Se houver uma partição de rede, o sistema deve permanecer disponível, mas pode retornar dados temporariamente inconsistentes (Ex: A maioria dos sistemas NoSQL de larga escala).
    - Justificativa: Em **big data settings**, a indisponibilidade é frequentemente mais custosa do que a inconsistência temporária, e o **overhead** de **locking** (necessário para consistência) tem um impacto severo na performance.

O Dilema CAP



Fonte: nitendratech.

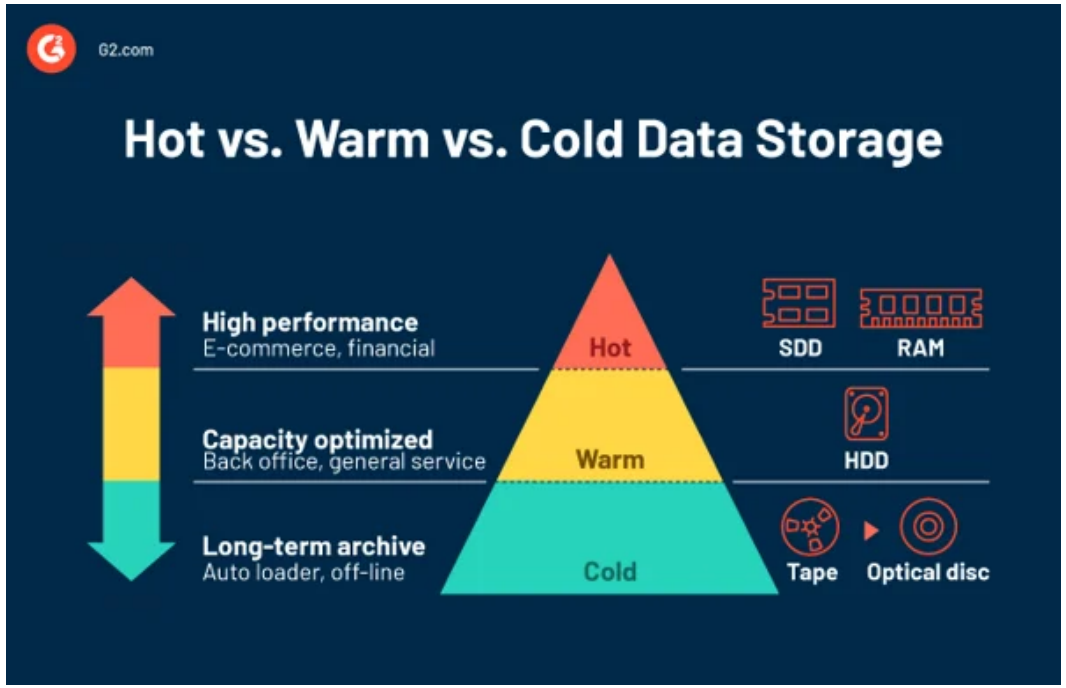
Cenário: Data Pipeline de E-commerce

- **Objetivo:** Arquitetura que suporte análise massiva (Escalabilidade).
- **Armazenamento de Logs Brutos/Mídia:** Utiliza Object Storage (Amazon S3) para escalabilidade massiva e armazenamento de dados não estruturados (e.g., logs de cliques, imagens de produtos).
- **Análise de Dados:** Utiliza um Data Warehouse (como Google BigQuery ou Amazon Redshift) para suportar consultas analíticas complexas sobre dados massivos (petascale).
    - O BigQuery suporta semântica SQL, é totalmente distribuído e replicado.

Otimização de Custo

- A otimização de custo no Object Storage é alcançada pelo Armazenamento em Camadas:

- Camada Hot (Acesso Frequente):: para dados estruturados e acessados com muita frequência.
- Camada Warm (Acesso Infrequente): para dados estruturados e acessados com frequência moderada.
- Camada Cold (Acesso Raro/ Arquivamento): para dados estruturados ou não estruturados que são acessados com pouca frequência.



Fonte da imagem: G2

Conclusão

- A escolha do modelo de armazenamento (Objeto, Bloco, Arquivo, Banco de Dados Relacional ou NoSQL) é a decisão fundamental. Ela depende criticamente da **natureza e do volume dos dados**, dos **requisitos de latência**, do **desempenho** e da **frequência de acesso**.
- **Object Storage (S3/Blobs):** É o modelo preferencial para grandes volumes de dados não estruturados e arquivamento, devido à sua escalabilidade massiva e ao custo-benefício. A imutabilidade dos objetos facilita a construção de implementações altamente escaláveis e confiáveis.
- **Fragmentação (Sharding):** É essencial para distribuir dados e funcionalidade de recuperação sobre múltiplas fontes.
- **Teorema CAP:** Em um sistema distribuído, é impossível satisfazer simultaneamente Consistência (C), Alta Disponibilidade (A) e Tolerância à Partição (P).
- **Modelos BASE:** Sistemas NoSQL de larga escala adotam o modelo **BASE** (Basically Available, Soft State, Eventually consistent), garantindo que os dados eventualmente convergirão, mas podem estar inconsistentemente por um período.