Paradigmas para Gestão da C onsistency (Consistência), A vailability (Disponibilidade) e P artitioning (Distribuição de Dados)

98H00-04
Infraestrutura para Gestão de Dados

Prof. Eduardo Arruda, MSc.

PUCRS | ESCOLA POLITÉCNICA

Requisitos de Bancos de Dados Distribuídos

C*onsistency* (Consistência)

Cada leitura recebe a gravação mais recente ou um erro.

A*vailability* (Disponibilidade)

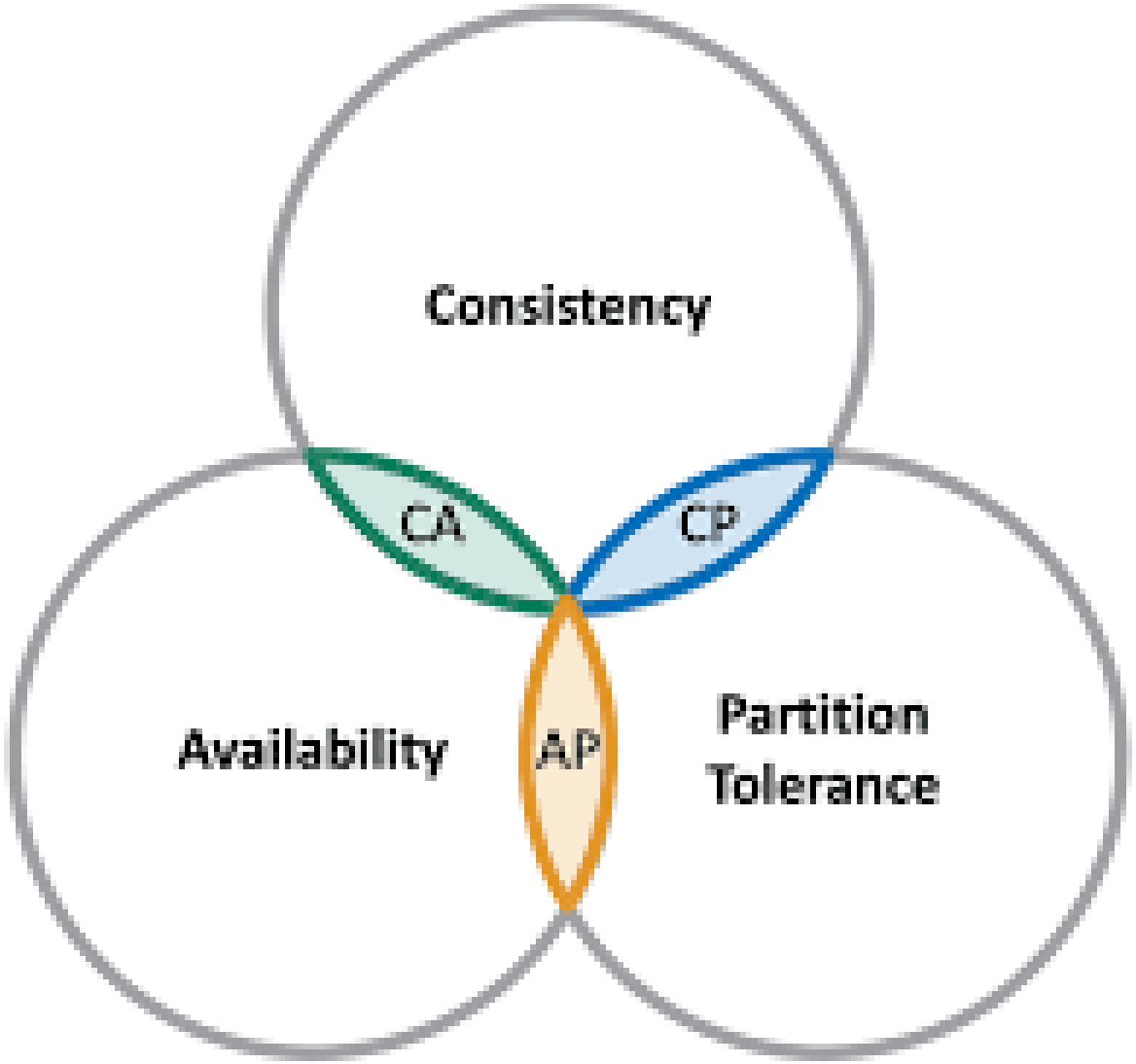
Cada solicitação recebe uma resposta (sem erro), sem a garantia de que contém a gravação mais recente.

P*artition Tolerancy* (Tolerância de partição)

O sistema continua a operar apesar de um número arbitrário de mensagens serem descartadas (ou atrasadas) pela rede entre os nós

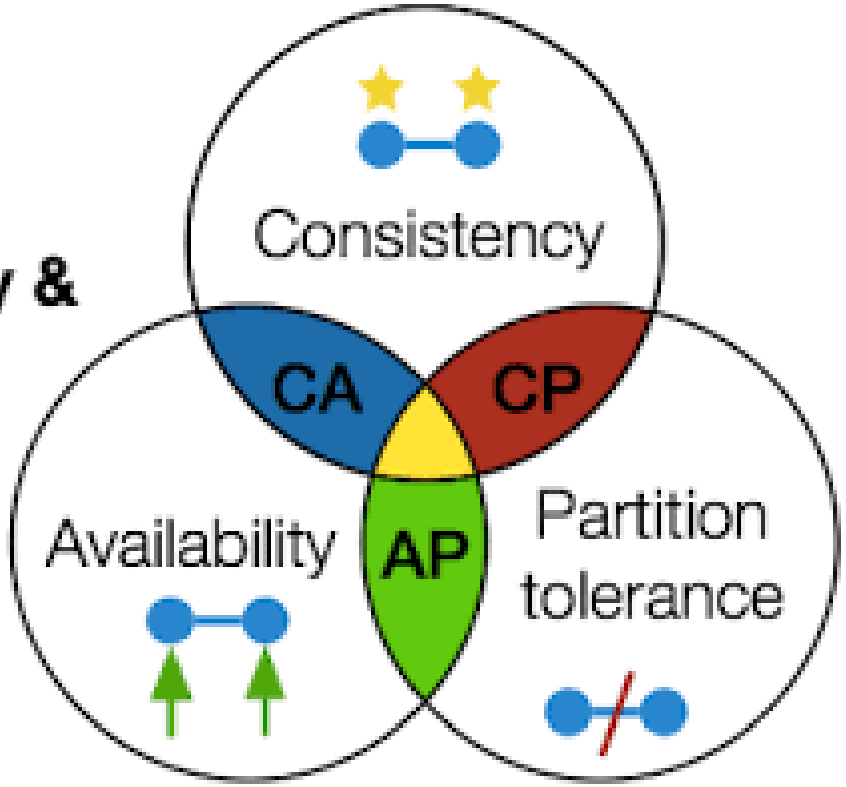
Teorema CAP

- Proposto por Eric Brewer em 1998
- Quando ocorre uma falha de partição (P), deve-se decidir se a operação será cancelada e, portanto, diminuirá a disponibilidade (A), mas garantirá a consistência (C), ou prosseguirá com a operação e, assim, fornecerá disponibilidade (A), mas arriscará a inconsistência (~C).
- Assim, se houver um particionamento de dados, deve-se escolher entre consistência (C) ou disponibilidade (A).



- Eric Brewer argumenta que o conceito "dois de três" pode ser um pouco enganador, porque os projetistas só precisam sacrificar a consistência (C) ou disponibilidade (A) na presença de partições (P), mas naquela época o particionamento era pouco frequente.

Consistency & Availability
- MySQL
- PostgreSQL



Consistency & Partition tolerance
- HBase
- MongoDB
- Redis
- Memcache

Availability & Partition tolerance
- Riak
- Cassandra
- CouchDB
- DynamoDB


ATOMIC
CONSISTENT
ACID
ISOLATED
DURABLE


BASICALLY AVAILABLE
SOFT STATE
BASE
EVENTUALLY CONSISTENT

Requisitos ACID

Atomicidade

- Cada transação é executada corretamente ou o processo é interrompido e o banco de dados volta ao estado anterior ao início da transação.
- Isso garante que todos os dados no banco de dados sejam válidos.

Consistência

- Uma transação processada nunca colocará em risco a integridade estrutural do banco de dados.

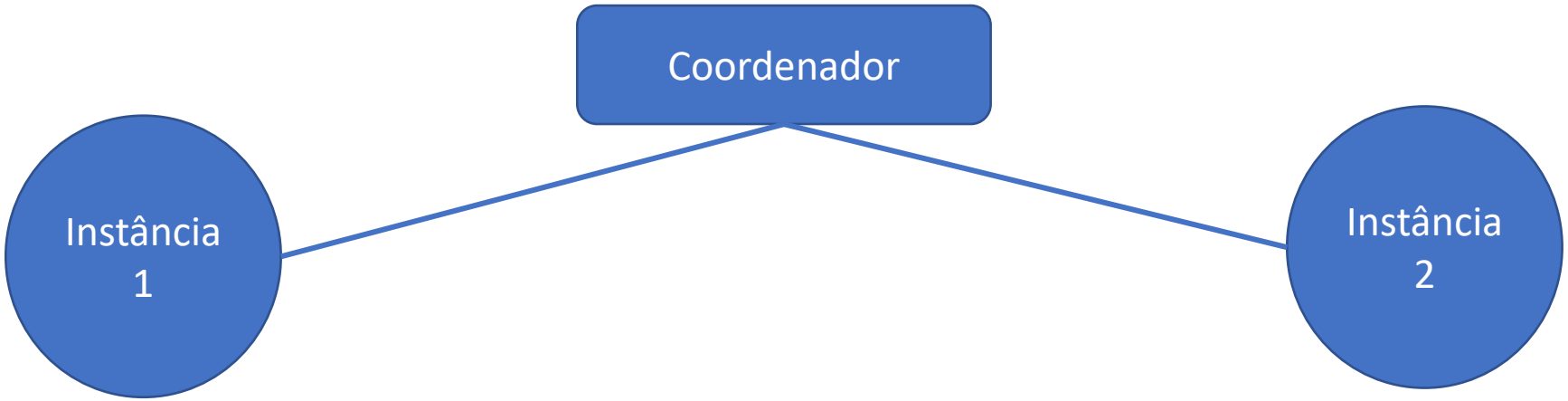
Isolamento

- As transações não podem comprometer a integridade de outras transações interagindo com elas enquanto ainda estão em andamento.

Durabilidade

- Os dados relacionados à transação concluída persistirão mesmo nos casos de queda de rede ou energia. Se uma transação falhar, ela não afetará os dados manipulados.

Replicação de Dados



Two-Phase Commit

- Durante a execução da transação, antes do início do protocolo de confirmação de duas fases
  - Quando a aplicação chama tx_begin para iniciar a transação, o coordenador cria um registro de transação em sua memória volátil
  - Cada vez que um gerenciador de recursos chama xa_reg para ingressar na transação como um instância, o coordenador anexa a identidade da instância ao registro da transação

Fase 1: Preparação

- Quando o aplicativo invoca tx_commit
- O coordenador envia mensagem de preparação (coordenação para todas as instâncias):
  - Se a instância quiser abortar a qualquer momento antes ou no recebimento da mensagem, ela aborta e libera bloqueios
  - Se a instância quiser confirmar, ela move todos os registros de atualização para o armazenamento em massa, forçando um registro de preparação para seu log
  - Garante que a instância será capaz de confirmar (apesar de falhas) se o coordenador decidir confirmar (já que os registros de atualização são duráveis)
  - Instância entra no estado preparado
- A instância envia uma mensagem de voto (“pronto” ou “abortando”)
  - não pode mudar de ideia
  - retém todos os bloqueios se o voto estiver "pronto"
  - entra em período incerto (não pode prever o resultado final)

Fase 1: Preparação

- Mensagem de votação (instância para coordenador): instância indica que está “pronto” para se comprometer ou está “abortando”
  - Coordenador registra voto no registro de transação
  - Se algum voto estiver “abortando”, o coordenador decide abortar e apaga o registro da transação
  - Se todos estiverem “prontos”, o coordenador decide o commit, força o registro do commit (contendo o registro da transação) para o seu log (fim da fase 1)
  - Transação confirmada quando o registro de confirmação é durável
  - Como todas as instâncias estão no estado preparado, a transação pode ser confirmada apesar de quaisquer falhas
  - O coordenador envia uma mensagem de confirmação ou aborto para todas as instâncias

Fase 2: Commit

- Envio de mensagem de commit ou abort (coordenador para instância)
  - Se enviar mensagem de commit
    - instância confirma localmente forçando um registro de confirmação em seu log
    - coorte envia mensagem de concluída ao coordenador
  - Se enviar a mensagem de abort, aborta
  - Em ambos os casos, os bloqueios são liberados e o período incerto termina
- Envio de mensagem de conclusão (da instância para coordenador)
  - Quando o coordenador recebe uma mensagem de conclusão de cada instância
    - ele grava um registro completo em seu log e
    - exclui o registro de transação do armazenamento volátil


ATOMIC
CONSISTENT
ACID
ISOLATED
DURABLE


BASICALLY AVAILABLE
SOFT STATE
BASE
EVENTUALLY CONSISTENT

Requisitos BASE

BAsicamente disponível

- Em vez de impor consistência imediata, os bancos de dados NoSQL modelados em BASE garantirão a disponibilidade dos dados, espalhando-os e replicando-os pelos nós do cluster de banco de dados.

Soft state

- Devido à falta de consistência imediata, os valores dos dados podem mudar ao longo do tempo.
- O modelo BASE rompe com o conceito de banco de dados que impõe sua própria consistência, delegando essa responsabilidade aos desenvolvedores.

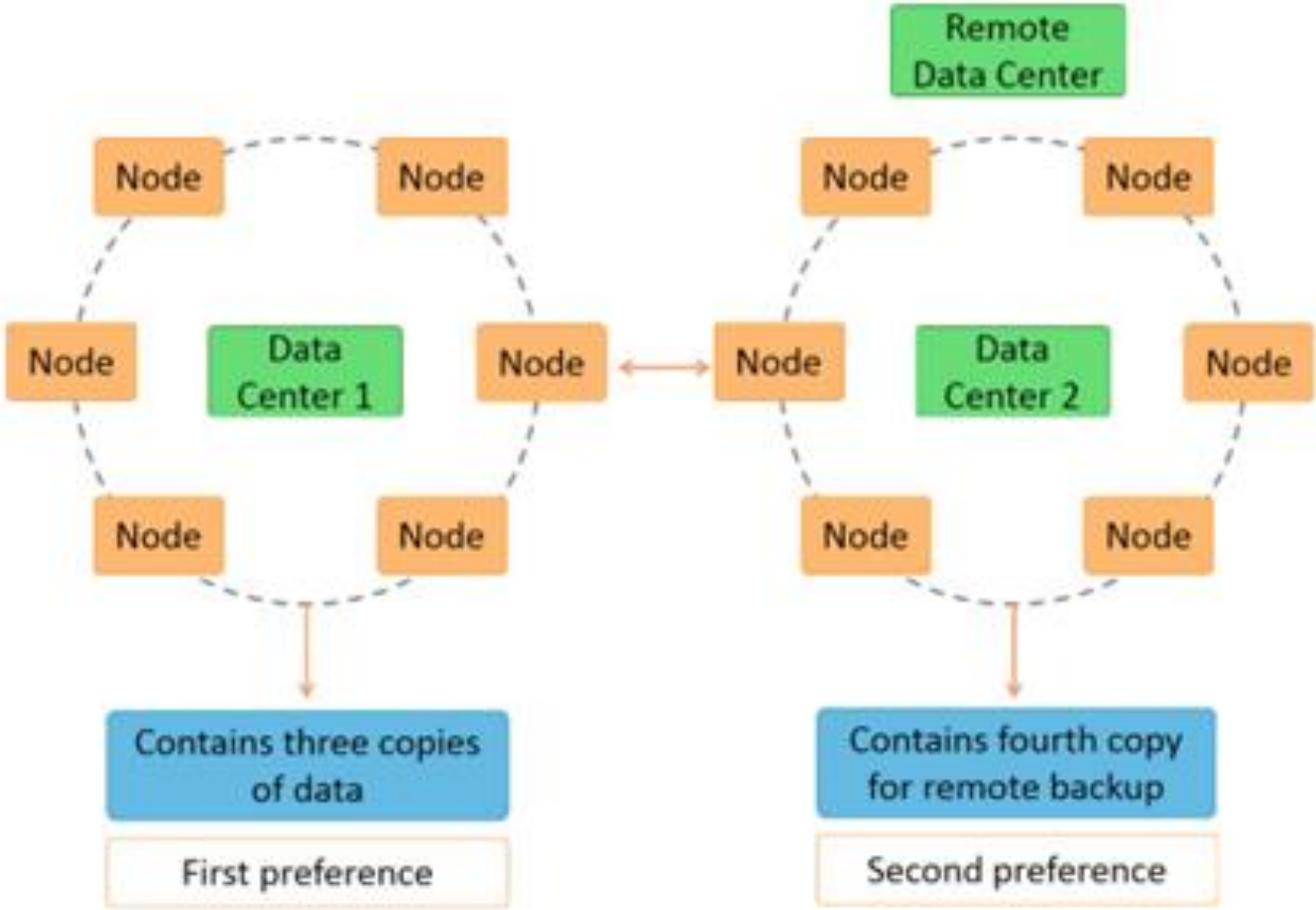
Eventualmente consistente

- O fato de o BASE não impor consistência imediata não significa que nunca a alcance.
- No entanto, até que isso aconteça, as leituras de dados ainda são possíveis (mesmo que não reflitam a realidade).

- Apache Cassandra é um SGBD NoSQL de código aberto
- Baseado em modelo de armazenamento de colunas amplas (*wide columns*) particionadas.
- Baseado em requisitos BASE.
- Inicialmente projetado no Facebook
  - Arquitetura orientada a eventos (SEDA)
  - Baseado
    - Nas técnicas de armazenamento e replicação do Amazon Dynamo
    - No modelo de mecanismo de armazenamento e dados Google Bigtable do Google

Organização dos Dados nos Nós

- Keyspace: define como um conjunto de dados é replicado entre datacenters. Replicação é o número de cópias de um mesmo dado salvas por cluster. Os keyspaces contêm tabelas.
- Tabela: define o esquema tipado para uma coleção de partições. As tabelas contêm partições, que contêm linhas, que contêm colunas. As tabelas do Cassandra podem adicionar novas colunas de forma flexível às tabelas, com tempo de inatividade zero.
- Partição: Define a parte obrigatória da chave primária que todas as linhas do Cassandra devem ter para identificar o nó em um cluster onde a linha está armazenada. Todas as consultas de alto desempenho fornecem a chave de partição na consulta.
- Linha: contém uma coleção de colunas identificadas por uma chave primária exclusiva composta pela chave de partição e, opcionalmente, chaves de cluster adicionais.
- Coluna: Um único dado com um tipo que pertence a uma linha.


Remote
Data Center
Node
Node
Node
Node
Node
Node
Node
Node
Data
Center 1
Data
Center 2
Contains three copies
of data
First preference
Contains fourth copy
for remote backup
Second preference

Principais objetivos

- Replicação completa de banco de dados multi-master
- Disponibilidade global com baixa latência
- Escalonamento em *hardware* comum
- Aumento da taxa de transferência de forma linear com o acréscimo de processadores adicionais
- Balanceamento de carga e crescimento de cluster *online*
- Consultas orientadas por chave particionada
- Esquema flexível
- ***Cassandra Query Language* (CQL), linguagem semelhante ao SQL**

Modelagem de Dados

- Sem joins
  - Joins devem ser implementados no cliente ou pela desnormalização dos dados
- Sem integridade referencial
  - Cassandra não implementa
- Desnormalização
  - Cassandra porque ele funciona melhor quando o modelo de dados é desnormalizado
  - Muitas vezes bancos de dados relacionais também são desnormalizados
    - Melhorar o desempenho
    - Manter valores históricos

Modelagem de Dados

- Projetando para armazenamento ideal
  - Manter as colunas relacionadas definidas juntas na mesma tabela
  - Minimizar o número de partições que devem ser pesquisadas para satisfazer uma determinada consulta
- Classificação é uma decisão de projeto
  - A ordem de classificação nas consultas é fixa e determinada inteiramente pela seleção de colunas de cluster que você fornece no comando CREATE TABLE
  - CQL SELECT oferece suporte a ORDER BY mas somente na ordem especificada pelas colunas de *clustering*