Sistemas Distribuídos

Aula 4 – Replicação de Dados

Prof. Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

O que é Replicação?

Razões para Replicar

O Problema da Consistência

Cache e Replicação na Web

Crescimento vs. Escalabilidade

Referências

Replicação de Dados

A chave para disponibilidade e tolerância a falhas

O que é Replicação de Dados?

Definição

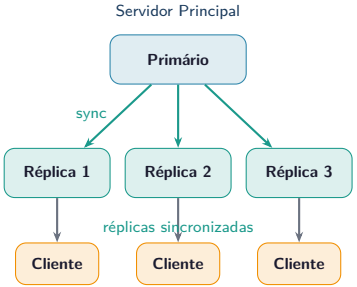
Replicação consiste em manter múltiplas cópias dos dados em locais distintos. Se um local for comprometido, o dado permanece acessível nas demais réplicas. [1]

Por que replicar?

- **Estabilidade** — o sistema continua funcionando com falhas
- **Confiabilidade** — proteção contra corrupção e perda de dados
- **Desempenho** — cópia próxima ao usuário reduz latência
- **Escalabilidade** — distribui carga entre réplicas
- **Tolerância a Falhas** — nenhum ponto único de falha

O desafio central

A replicação resolve disponibilidade, mas cria um novo problema: **como manter todas as cópias consistentes?**



Razões para Replicar

Confiabilidade, desempenho e escalabilidade

Por que Replicar? — Os 3 pilares

▪ Maior Confiabilidade

Se uma réplica se desconecta ou falha, o sistema continua funcionando com as demais cópias.

A consistência entre réplicas é monitorada periodicamente para detectar corrupção de dados. Dados corrompidos em uma réplica são corrigidos pelas demais. [1]

Replicação com tolerância a falhas

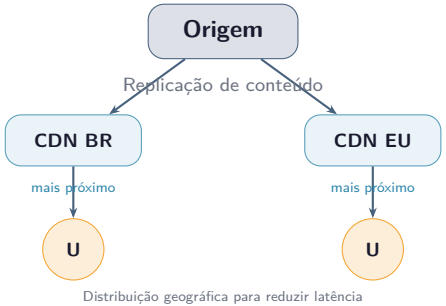
Por que Replicar? — Os 3 pilares

▪ Melhor Desempenho

Cópia dos dados próxima ao processo/usuário que a utiliza **reduz drasticamente a latência** de acesso.

Distribui a carga de leitura entre as réplicas, evitando gargalos no servidor primário.

Ex: CDNs (Cloudflare, Akamai) replicam conteúdo em centenas de pontos pelo mundo.



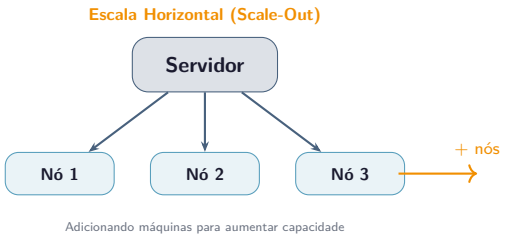
Por que Replicar? — Os 3 pilares

▪ Escalabilidade

À medida que o SD cresce em nós e área geográfica, a replicação distribui os dados onde são necessários.

Crescimento em quantidade: divide o trabalho entre réplicas, aliviando o servidor central.

Crescimento geográfico: cópia local reduz acessos de longa distância e latência.



O Problema da Consistência

O maior desafio da replicação

Consistência — O Preço da Replicação

O problema central

Toda vez que uma réplica é modificada, ela se torna **diferente das demais** — exigindo sincronização. Manter todas as réplicas atualizadas gera **overhead** de rede e processamento.

Quanto **mais réplicas**, maior o custo de manter consistência. Quanto **mais atualizações**, maior o tráfego de sincronização.

O Teorema CAP revisitado

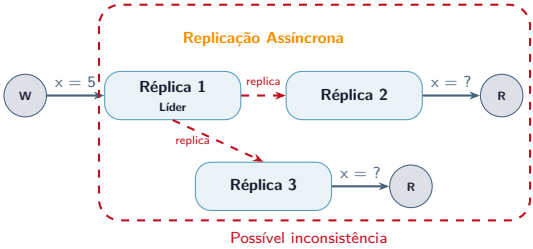
A replicação é o coração do dilema **CAP**:

- **Consistência** forte exige que todas as réplicas vejam os mesmos dados ao mesmo tempo
- **Disponibilidade** exige que qualquer réplica responda sempre
- **Partição** de rede pode impedir a sincronização

Escolher dois deles implica abrir mão do terceiro.

Estratégias de consistência

- **Consistência forte** — todas as réplicas sempre iguais; mais lento
- **Consistência eventual** — réplicas convergem eventualmente; mais rápido
- **Read-your-writes** — você vê suas próprias escritas



Cache e Replicação na Web

Vantagens e Trade-offs

Cache — Replicação Transparente na Web

Como funciona o cache web

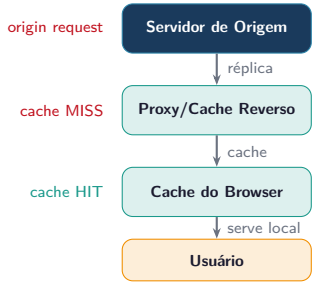
Os navegadores armazenam **cópias locais** das páginas já visitadas. Na próxima visita, servem a cópia local, sem acessar o servidor **latência quase zero**.

O dilema do cache

- Cache local é ótimo para **desempenho**
- Mas pode exibir conteúdo **desatualizado**
- Se o servidor atualizar todos os caches proativamente, pode **degradar o desempenho global**
- Sem cache próximo ao usuário, toda requisição vai ao servidor original — **alta latência**

Solução: TTL e validação

Cache com **TTL** (Time-to-Live) equilibra frescor e desempenho. `Cache-Control` e `ETag` permitem validar o conteúdo sem rebaixar o conteúdo completo.



Crescimento vs. Escalabilidade

O trade-off fundamental da replicação

Crescimento vs. Escalabilidade

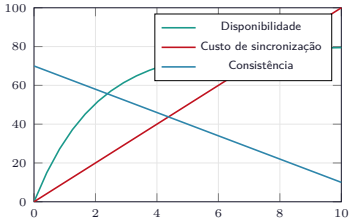
▪ Replicação como técnica de crescimento

Manter cópias próximas dos processos que as utilizam **melhora o desempenho** e resolve problemas de escalabilidade — reduz acessos de longa distância e latência.

▪ O custo escondido

Manter cópias **atualizadas** pode requerer:

- Maior **largura de banda** de rede
- Acessos constantes ao primário
- Sobrecarga na rede — tudo deve ser cuidadosamente **avaliado e dosado**



▪ A equação do equilíbrio

Mais réplicas → maior disponibilidade
Mais réplicas → maior custo de consistência
Solução: replicação *lazy* ou estratégias de consistência eventual.

Resumo — Replicação em Sistemas Distribuídos

Aspecto	Vantagem	Desafio
Disponibilidade	Sistema continua com falhas parciais	Detectar e recuperar réplicas falhas
Desempenho	Cópia local reduz latência	Cache desatualizado (staleness)
Escalabilidade	Distribui carga entre réplicas	Sincronização consume largura de banda
Confiabilidade	Proteção contra perda/corrupção	Gerenciar versões e conflitos
Consistência	Dados corretos em qualquer réplica	CAP: impossível ter tudo ao mesmo tempo

Replicação é imprescindível em SDs modernos — mas exige consistência bem projetada

Referências I

- TANENBAUM, A. S.; VAN STEEN, M. *Distributed Systems: Principles and Paradigms*. 2. ed. Pearson, 2007.
- COULOURIS, G. et al. *Distributed Systems: Concepts and Design*. 5. ed. Addison-Wesley, 2013.