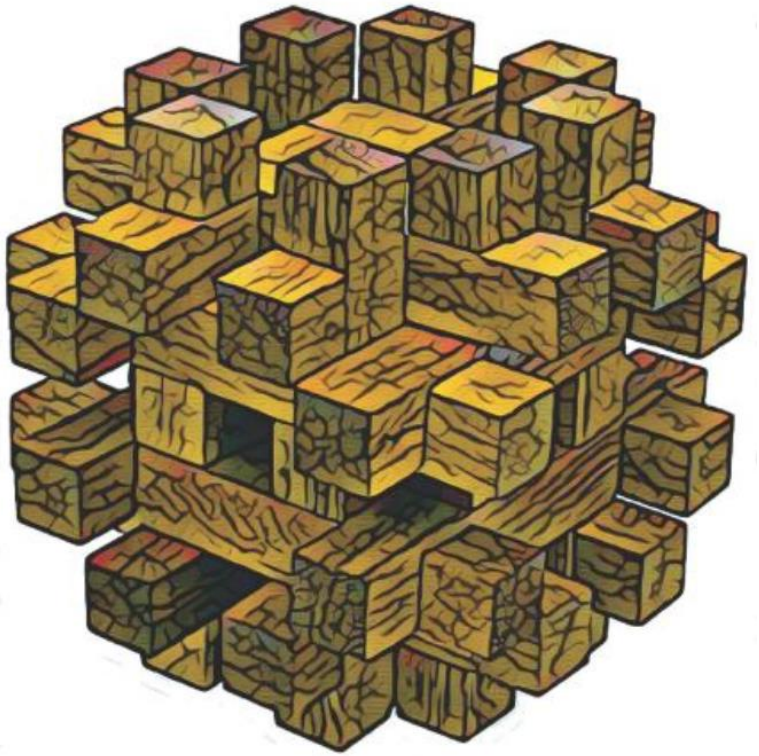
Distributed Systems

Maarten Van Steen & Andrew S. Tanenbaum



3th Edition – Version 3.03 - 2020

Capítulo 7

Consistência e Replicação

Quinta-feira, 25 de Agosto de 2022

DESEMPENHO E ESCALABILIDADE

QUESTÃO PRINCIPAL

Para manter replicas consistentes, geralmente precisamos garantir que todas operações conflitantes sejam feitas na mesma ordem em todos lugares.

OPERAÇÕES CONFLITANTES: DO MUNDO DAS TRANSAÇÕES

- Conflito READ-WRITE: uma operação de leitura e uma de escrita concorrentemente
- Conflito WRITE-WRITE: duas operações de escrita concorrentes

QUESTÃO

- A garantia de ordenação global em operações conflitantes pode ser uma operação custosa, que degrada escalabilidade. Solução: enfraquecimento dos requisitos de consistência de forma que sincronização global possa ser evitada.

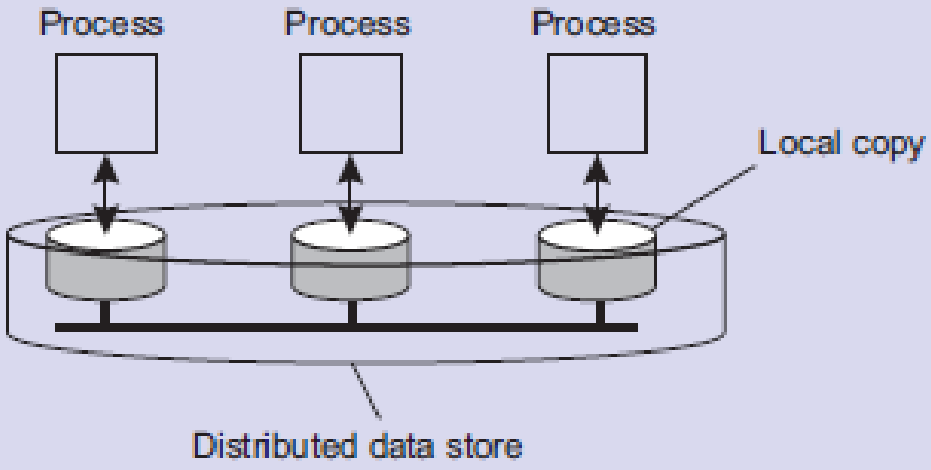
MODELOS DE CONSISTÊNCIA CENTRADO EM DADOS

MODELO DE CONSISTÊNCIA

- Um contrato entre repositórios de dados distribuídos e processos, no qual o repositório especifica precisamente quais são os resultados de operações de leitura e escrita são em caso de concorrência.

ESSENCIAL

Um repositório de dados é uma coleção de armazenadores:



CONSITÊNCIA CONTÍNUA

PODEMOS FALAR SOBRE GRAU DE CONSISTÊNCIA

- Replicas podem ser diferentes em seus valores numéricos
- Replicas podem diferir em seu envelhecimento relativo
- Pode existir diferenças em relação ao número e ordem das operações de atualização realizadas

CONIT

Unidade de consistência => especifica a unidade de dados sobre o qual consistência deve ser medida

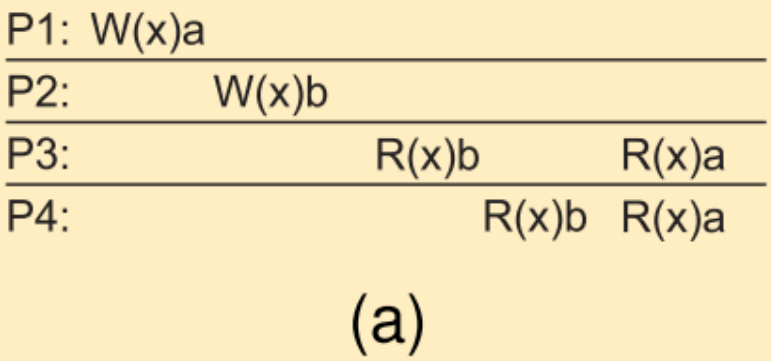
CONSISTÊNCIA SEQUÊNCIAL

DEFINIÇÃO

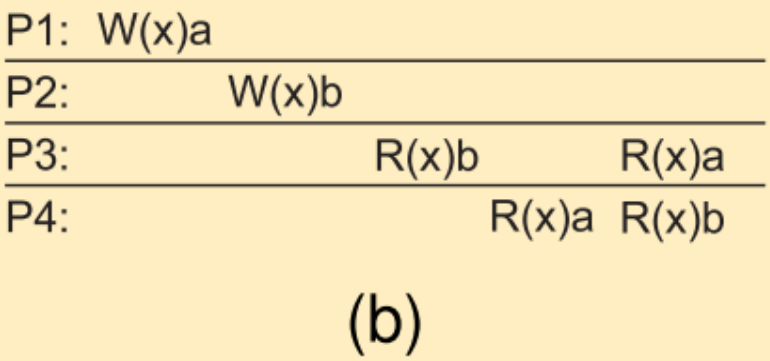
- O resultado de uma execução é o mesmo que o de operações de todos os processos executados em uma ordem sequencial, e as operações de cada processo individual aparecem nesta sequência na ordem especificada pelo programa.

(a) UM ARMAZENAMENTO SEQUENCIAL CONSISTENTE

(b) UM ARMAZENAMENTO NÃO SEQUENCIALMENTE CONSISTENTE



(a)



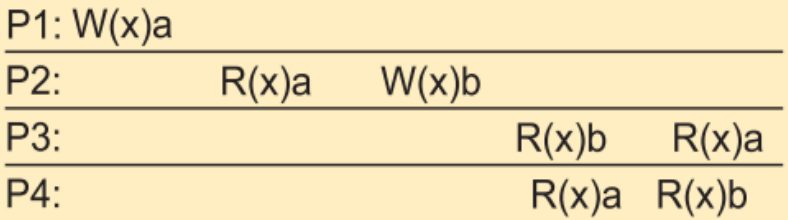
(b)

CONSISTÊNCIA CAUSAL

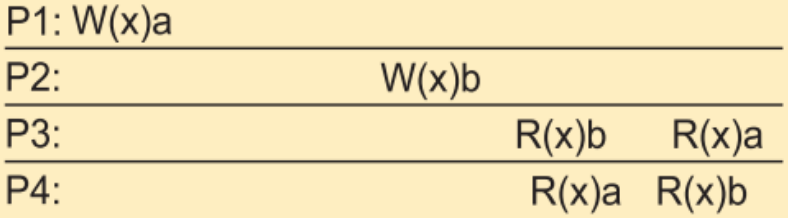
DEFINIÇÃO

- Escritas que são causalmente relacionadas devem ser vistas por todos processos na mesma ordem. Escritas concorrentes podem ser vistas em ordem diferente por diferentes processos

(a) UMA VIOLAÇÃO DE UM ARMAZENAMENTO CAUSAL-CONSISTENTE (b) UMA SEQUENCIA CORRETA DE EVENTOS EM UM ARMAZENAMENTO CAUSAL-CONSISTENTE



(a)



(b)

OPERAÇÕES EM GRUPO

DEFINIÇÃO

- Acessos a “locks” são sequenciais e consistentes
- Nenhum acesso a um “lock” é permitido ser desempenhado até que todas escritas prévias tenham sido completadas em todos lugares
- Nenhum acesso a dados é permitido ser desempenhado até que todos acessos a “locks” tenham sido desempenhados

IDÉIA BÁSICA

Não se preocupe com leituras e escritas de uma série de operações serem imediatamente conhecidas por outros processos. Você quer somente que o efeito da série seja conhecido.

OPERAÇÕES EM GRUPO

UMA SEQUÊNCIA VÁLIDA DE EVENTOS PARA CONSISTÊNCIA DE ENTRADA

P1:	L(x)	W(x)a	L(y)	W(y)b	U(x)	U(y)					
P2:							L(x)	R(x)a		R(y)	NIL
P3:									L(y)	R(y)b	

OBSERVAÇÃO

- Consistência de entrada implica que precisamos de “lock” e “unlock” de dados (implícito ou não)

QUESTÃO

- Qual é uma forma conveniente para tornar consistência mais transparente para programadores ?

CONSITÊNCIA PARA USUÁRIOS MÓVEIS

EXEMPLO

Considere uma base de dados distribuída para a qual você tem acesso pelo seu notebook, que age como *front-end* da base de dados:

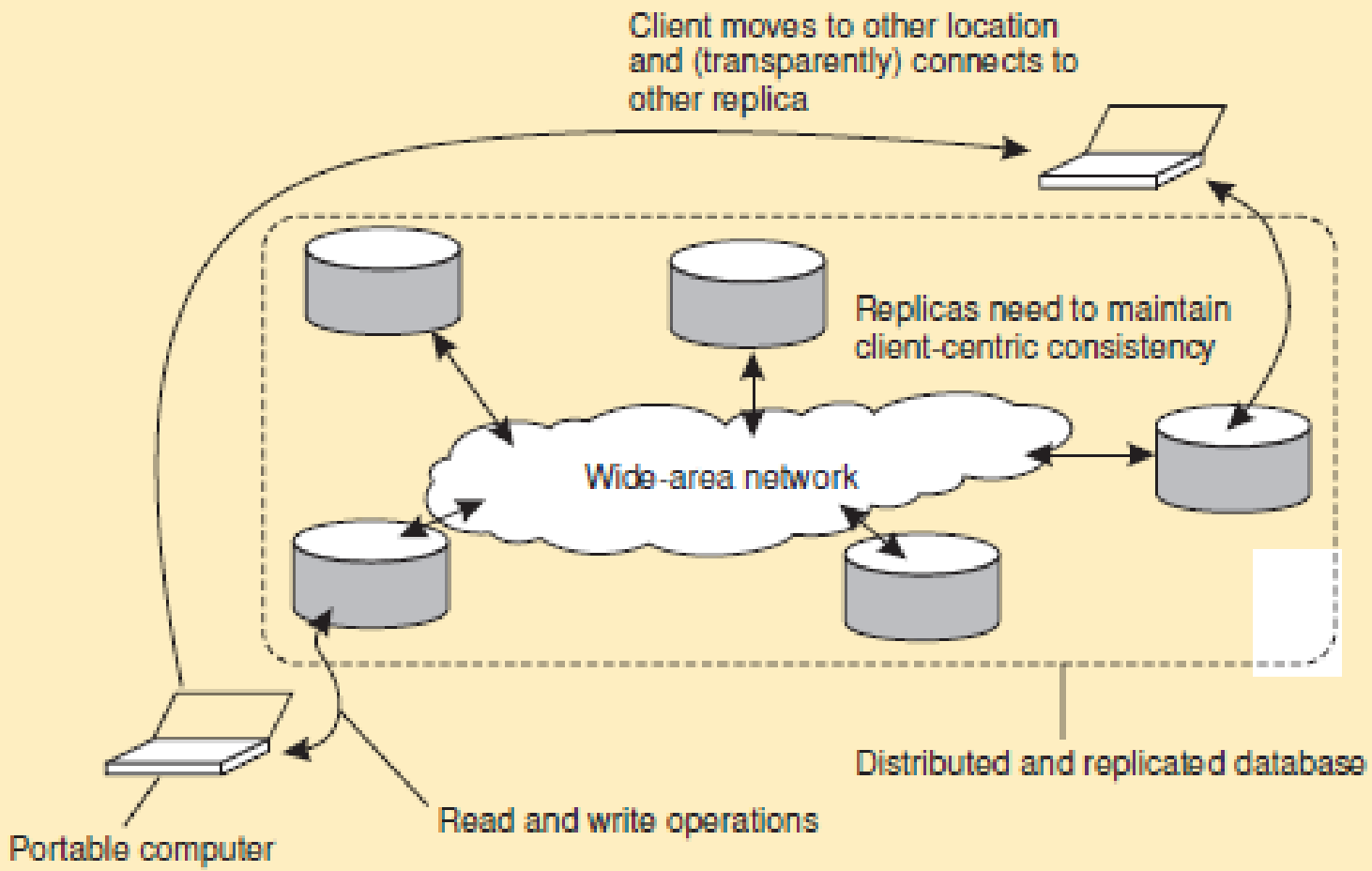
- Na localidade A você acessa a base de dados fazendo leituras e atualizações
- Na localidade B você continua seu trabalho, mas ao menos que acesse o mesmo servidor da localidade A, pode-se detectar inconsistências:
  - Suas atualizações em A podem não ter sido propagadas para B
  - Vc pode estar lendo novas entradas além daquelas disponíveis em A
  - Suas atualizações em B podem eventualmente conflitar com aquelas em A

NOTA

- A única coisa que você quer é que as entradas que você atualiza e/ou lê em A estão em B do jeito que vc deixou em A. Neste caso, a base de dados aparecerá consistente pra você.

ARQUITETURA BÁSICA

O PRINCÍPIO DE UM USUÁRIO MÓVEL ACESSANDO DIFERENTES RÉPLICAS DE UMA BASE DE DADOS DISTRIBUÍDA



LEITURAS MONOTÔNICAS

DEFINIÇÃO

- Se um processo lê o valor de um item dado x, qualquer operação de leitura sucessiva em x pelo processo vai sempre retornar o mesmo ou um valor mais recente

OPERAÇÕES DE LEITURA REALIZADAS POR UM ÚNICO PROCESSO P EM DUAS DIFERENTES CÓPIAS DA MESMA BASE DE DADOS. (a) UMA LEITURA MONOTÔNICA CONSISTENTE NA BASE DE DADOS (b) UMA BASE DE DADOS QUE NÃO PROVÊ LEITURAS MONOTÔNICAS

L1:	$W_1(x_1)$	$R_1(x_1)$
L2:	$W_2(x_1;x_2)$	$R_1(x_2)$

L1:	$W_1(x_1)$	$R_1(x_1)$
L2:	$W_2(x_1\|x_2)$	$R_1(x_2)$

CONSISTÊNCIA CENTRADA NO CLIENTE: NOTAÇÃO

NOTAÇÃO

- $W_1(x_2)$ é a *operação de leitura pelo processo* P_1 *que leva a versão* x_2 *de* x
- $W_1(x_i;x_j)$ indica P_1 produziu versão x_j baseada em uma versão prévia de x_i
- $W_1(x_i|x_j)$ indica P_1 produziu versão x_j concorrente a versão x_i

LEITURAS MONOTÔNICAS

EXEMPLO

Lendo automaticamente atualizações de seu calendário a partir de servidores diferentes. Leituras monotônicas garantem que um usuário vê todas atualizações, independente de qual servidor foi usado para leitura automática.

EXEMPLO

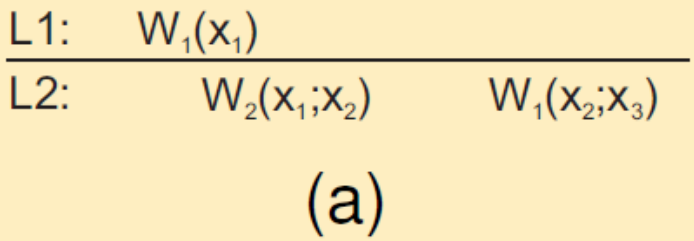
Lendo (não modificando) email chegando enquanto você está em movimento. Cada vez que vc conecta a um servidor diferente, o servidor busca (pelo menos) todas atualizações do servidor visitado previamente

ESCRITAS MONOTÔNICAS

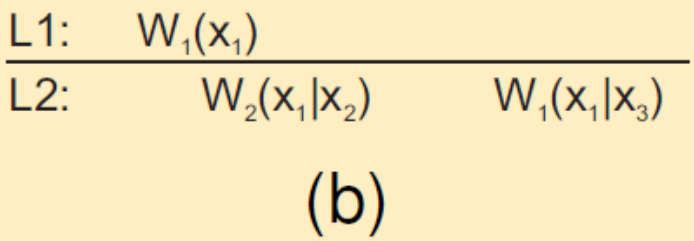
DEFINIÇÃO

- Uma operação escrita (write) por um processo em item dado x é completada antes que qualquer operação escrita sucessiva em x pelo mesmo processo.

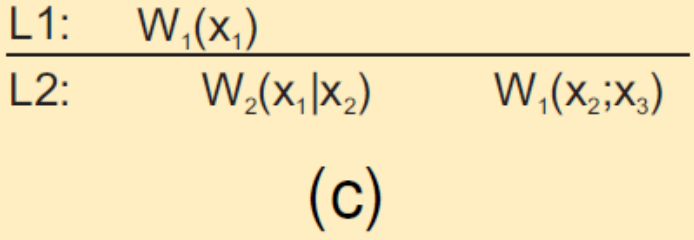
(a) UMA ESCRITA MONOTÔNICA CONSISTENTE NA BASE DE DADOS (b) UMA BASE DE DADOS QUE NÃO PROVÊ ESCRITAS COM CONSISTÊNCIA MONOTÔNICAS (c) SEM CONSISTÊNCIA POIS WS(X1|X2) E ASSIM TAMBÉM WS(X1|X3). (d) CONSISTÊNCIA POIS WS(X1;X3) EMBORA X1 TENHA APARENTEMENTE SOBRESCRITO X2.



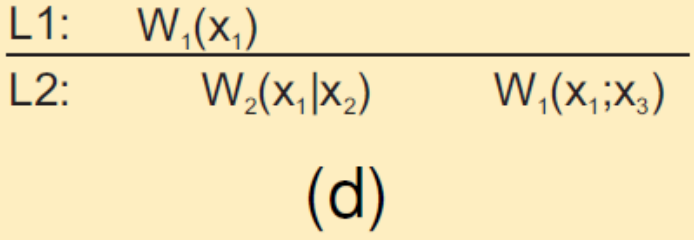
(a)



(b)



(c)



(d)

ESCRITAS MONOTÔNICAS

EXEMPLO

Atualizando um programa no servidor S_2, e garantindo que todos componentes nos quais a compilação e linkagem dependem, também acontecem em S_2

EXEMPLO

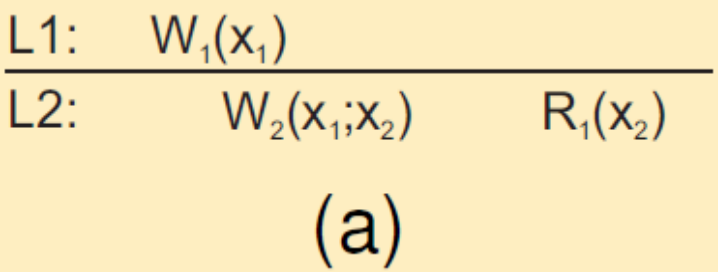
Mantendo versões de arquivos replicados na ordem correta em todos lugares (propagam a versão prévia para o servidor onde a nova versão está instalada)

LEIA SUAS ESCRITAS (read your writes)

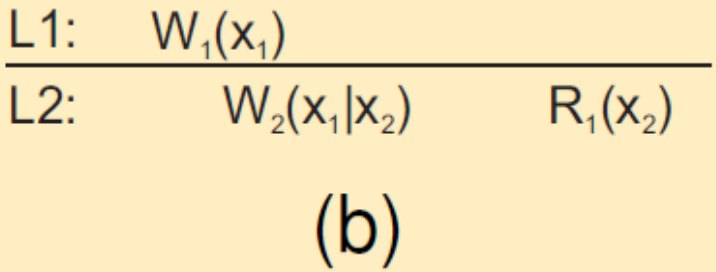
DEFINIÇÃO

- O efeito de uma operação escrita por um processo em item dado x, sempre será visto por uma operação de leitura sucessiva em x pelo mesmo processo

(a) UMA BASE DE DADOS QUE PROVÊ CONSISTÊNCIA READ-YOUR-WRITES. (b) UMA BASE DE DADOS QUE NÃO PROVÊ



(a)



(b)

EXEMPLO

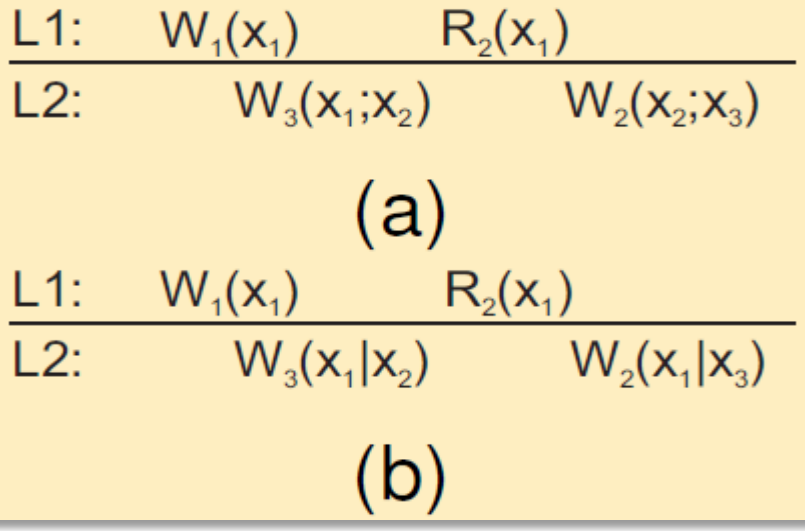
Atualizando sua página Web e garantindo que seu Web Browser mostre a versão mais nova ao invés da cache

ESCRITAS SEGUEM LEITURAS

DEFINIÇÃO

- Uma operação escrita por um processo em um item dado x seguindo uma operação de leitura prévia em x pelo mesmo processo é garantido que vá acontecer no mesmo ou no valor de x mais recente que foi lido.

(a) UMA BASE DE DADOS COM CONSISTÊNCIA WRITE-FOLLOW-READ (b) UMA BASE DE DADOS QUE NÃO PROVÊ CONSISTÊNCIA WRITE-FOLLOW-READS



EXEMPLO

Veja as reações de artigos postados somente se vc tem a postagem original (uma leitura "pulls in" a operação escrita correspondente

LOCALIZAÇÃO DE RÉPLICAS

ESSÊNCIA

Descubra qual as melhores k localidades de N possíveis locais.

- Selecione a melhor localização em N – K para a qual a distância média para os clientes é mínima. Então escolha o melhor próximo servidor. (Nota: a primeira localidade escolhida minimiza a distância média para todos clientes) computacionalmente caro.
- Selecione o maior K-ésimo sistema autônomo e coloque o servidor no melhor hospedeiro conectado. computacionalmente caro
- Posicione nós em uma geometria espacial d-dimensional, onde a distância reflete a latência. Identifique as K regiões com a maior densidade e coloque um servidor em cada uma. Computacionalmente barato.

REPLICAÇÃO DE CONTEÚDO

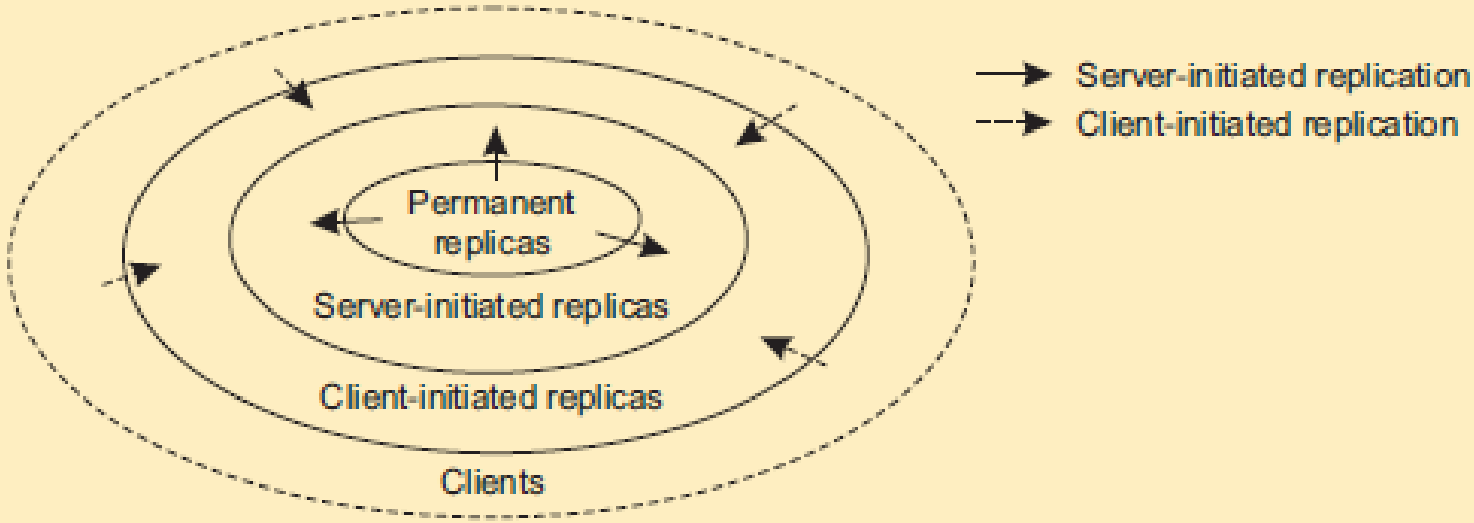
DISTINÇÃO DE PROCESSOS DIFERENTES

Um processo é capaz de hospedar uma réplica de um objeto ou dados:

- Réplicas permanentes: processo/máquina sempre tendo uma réplica
- Réplica iniciada por servidor: processo que pode dinamicamente hospedar uma réplica sob requisição de outro servidor no armazém de dados
- Réplica iniciada pelo cliente: processo que pode dinamicamente hospedar uma réplica sob requisição de um cliente (cache no cliente)

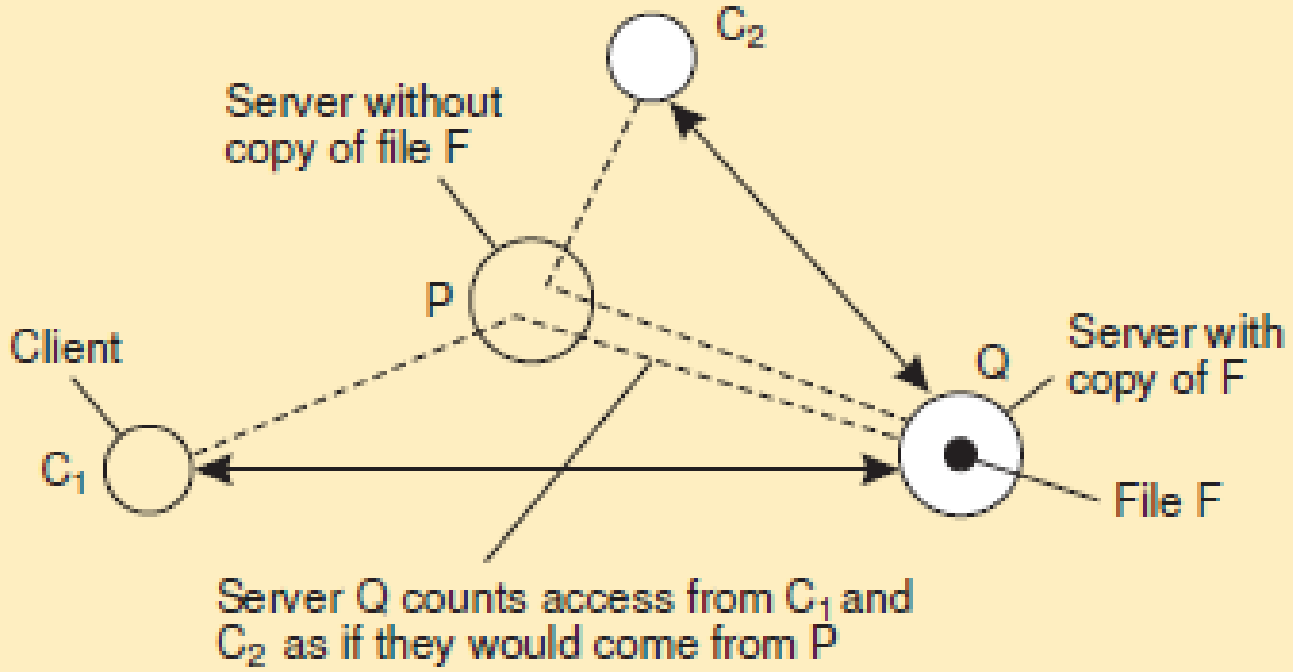
REPLICAÇÃO DE CONTEÚDO

A ORGANIZAÇÃO LÓGICA DE DIFERENTES TIPOS DE CÓPIAS DO ARMAZÉM DE DADOS EM TRÊS ANÉIS CONCENTRICOS



REPLICAS INICIADAS PELO SERVIDOR

CONTANDO REQUISIÇÕES DE ACESSO DE DIFERENTES CLIENTES



- Mantém contagem de acesso por arquivo, agregado por consideração de servidor mais próximo por clientes requisitantes
- Número de acessos cai pra baixo de margem D => descarta arquivo
- Número de acessos excedem margem R => replicar arquivo
- Número de acessos entre D e R => migrar arquivo

DISTRIBUIÇÃO DE CONTEÚDO

CONSIDERE SOMENTE UMA COMBINAÇÃO CLIENTE-SERVIDOR

- Propague somente notificação/invalidação da atualização (frequentemente usado para caches)
- Transfira dados de uma cópia para outra (bases distribuídas): replicação passiva
- Propague a operação atualização para outras cópias: replicação ativa

NOTA

Não há uma estratégia melhor que a outra, mas todas são dependentes na largura de banda disponível e razão read-to-write em réplicas

DISTRIBUIÇÃO DE CONTEÚDO

OBSERVAÇÃO

Podemos chavear dinamicamente entre arrendamentos (leases) de dados em modelo PULL e PUSH: um contrato no qual o servidor promete atualizar dados (PUSH) para o cliente até que o arrendamento expire.

FAZER O TEMPO DE ARRENDAMENTO EXPIRAR EM FUNÇÃO DO TEMPO E DO COMPORTAMENTO DO SISTEMA (LEASE ADAPTATIVO)

- Age-based leases (idade): um objeto que não foi mudado por um longo período, não será mudado no futuro próximo, assim provenha um arrendamento de longa duração
- Arrendamento baseado na frequência de renovação: quanto mais frequente um cliente requisite um objeto, mais longo será o tempo de expiração para aquele cliente (para aquele objeto) será.
- Arrendamento baseado em estado: quanto maior a carga do servidor, menor fica o tempo de expiração,

QUESTÃO

Por que estamos fazendo isto ?

DISTRIBUIÇÃO DE CONTEÚDO: sistema cliente/servidor

UMA COMPARAÇÃO ENTRE PROTOCOLOS PUSH E PULL NO CASO DE MULTIPLOS CLIENTES, E SERVIDOR ÚNICO

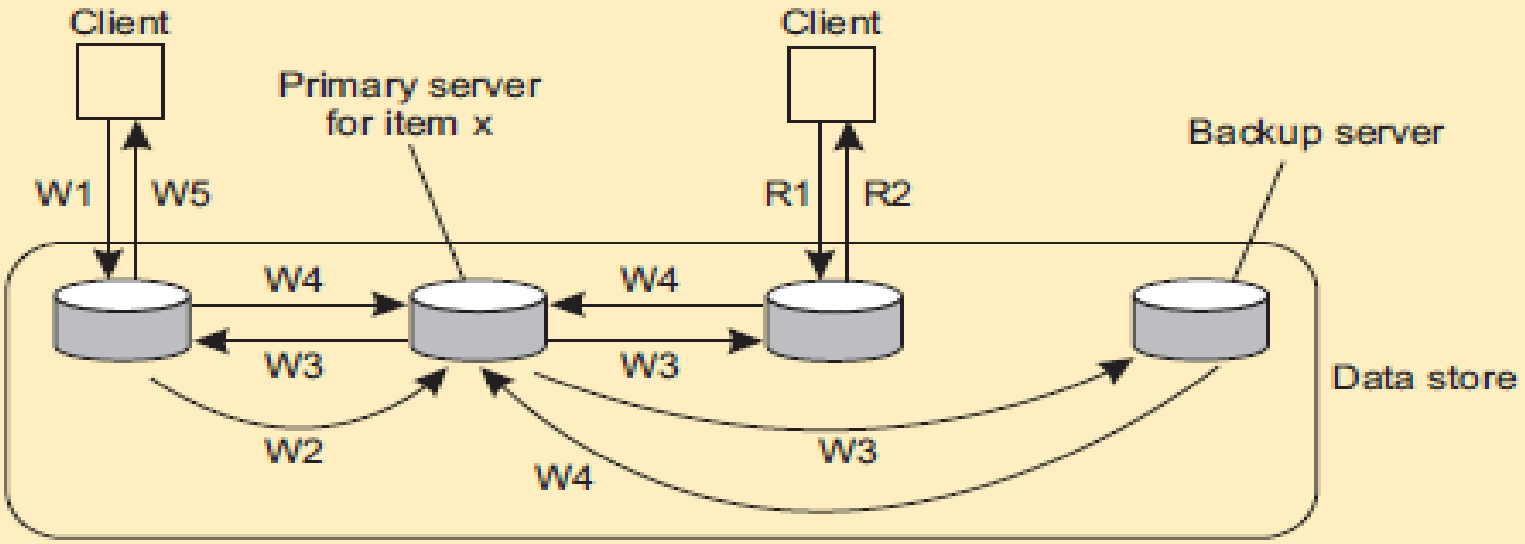
- Pushing updates: server-initiated approach, in which update is propagated regardless whether target asked for it.
- Pulling updates: client-initiated approach, in which client requests to be updated.

Issue	Push-based	Pull-based
1:	List of client caches	None
2:	Update (and possibly fetch update)	Poll and update
3:	Immediate (or fetch-update time)	Fetch-update time

1: State at server
2: Messages to be exchanged
3: Response time at the client

PROTOCOLO DE BASE PRIMÁRIA

PRIMARY-BACKUP PROTOCOL



W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

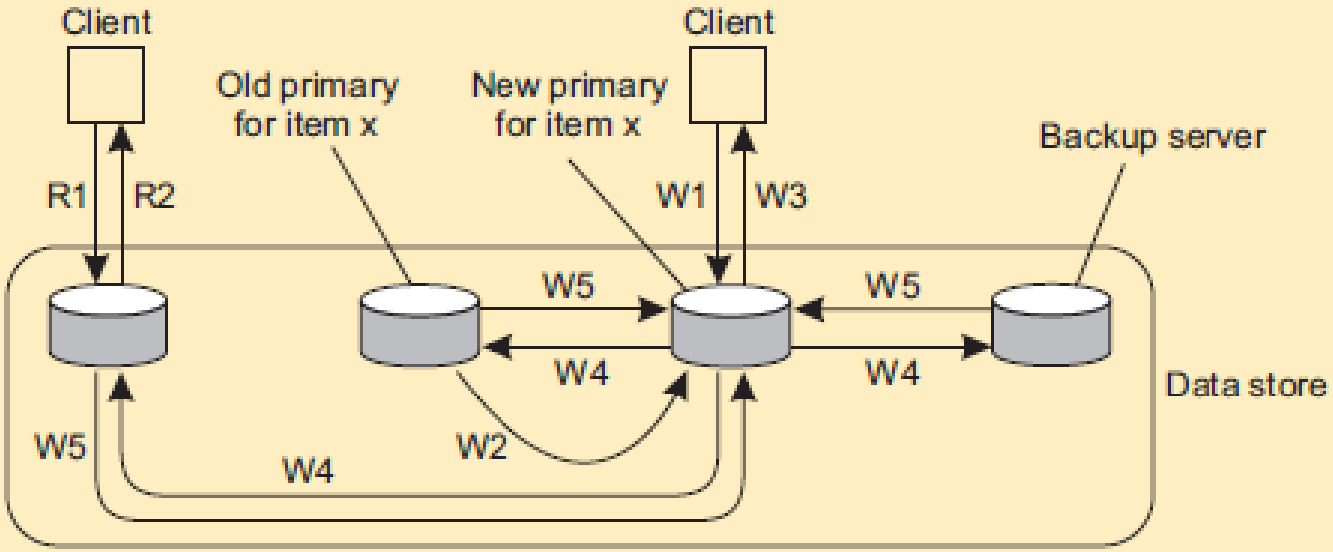
R1. Read request
R2. Response to read

EXEMPLO primary-backup protocol

Tradicionalmente aplicado em base de dados distribuída e sistemas de arquivos que demandam alto grau de tolerância a falhas. Réplicas normalmente estão na mesma LAN

PROTOCOLO DE BASE PRIMÁRIA

PRIMARY-BACKUP PROTOCOL COM WRITE LOCAIS



W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

EXEMPLO primary-backup protocol

Computação móvel em modo desconectado (envie todos arquivos relevantes para usuário antes de desconectar, e atualiza mais tarde).