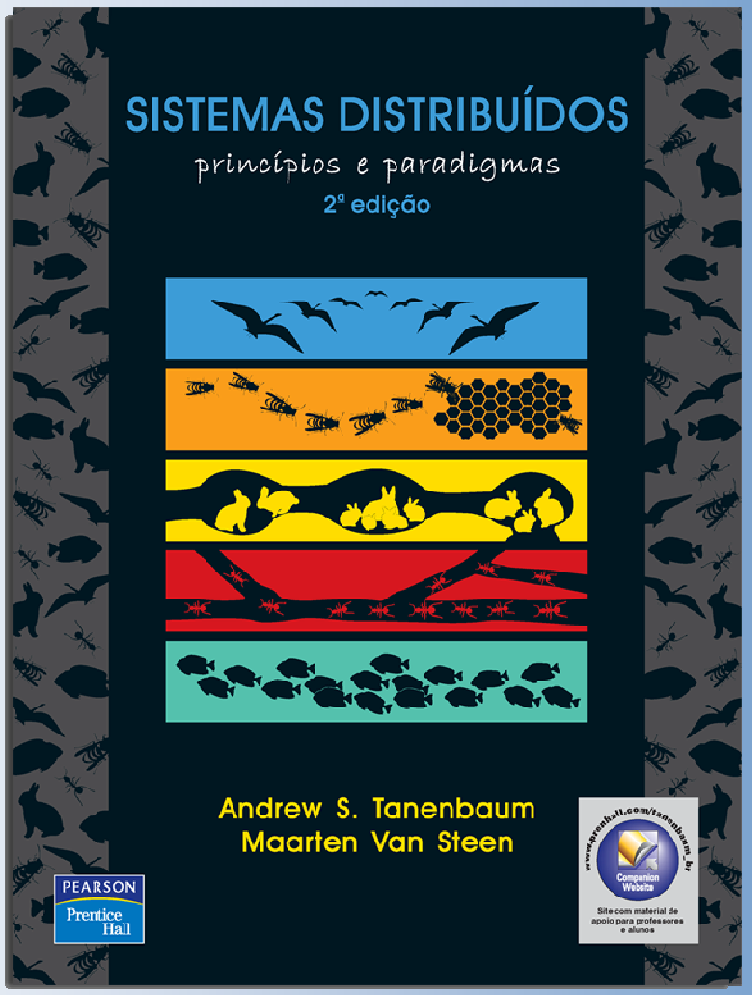
Consistência e replicação



capítulo 7

Consistência e Replicação

1. Introdução
2. Modelos de Consistência Centrados em Dados
3. Modelos de Consistência Centrados no Cliente
4. Gerenciamento de Réplicas
5. Protocolos de Consistência

Introdução
Razões para Replicação

- ☺ Confiabilidade
  - ☺ É possível continuar trabalho mesmo que uma das réplicas caia;
  - ☺ Maior proteção de dados corrompidos;
- ☺ Desempenho
  - ☺ Ampliação em quantidade para dividir trabalho de servidor centralizado, diminuindo o esforço para cada servidor;
  - ☺ Ampliação geográfica para diminuir tempo de acesso a dados, aumentando o desempenho dos clientes;
- ☹ Consistência
  - ☹ Sempre que uma cópia é modificada, é necessário atualizar todas as réplicas, o que pode gerar, no mínimo, em maior largura de banda para manter consistência dos dados

Introdução
Replicação como técnica de crescimento

- Questões de escalabilidade aparecem sob a forma de problemas de desempenho.
  - ☺ Cópias de dados mais próximas dos processos que as estão usando pode melhorar o desempenho pela redução do tempo de acesso, resolvendo o problema de escalabilidade!
  - 😐 Manter cópias atualizadas pode requerer mais largura de banda de rede...
    - ☹ Supondo que *P* acessa uma réplica *N* vezes por segundo, e essa é atualizada *M* vezes por segundo. Se *N* << *M*, muitas versões da réplica nem serão vistas por *P*!
    - ☹ Manter réplicas atualizadas pode estar sujeito a problemas de escalabilidade, principalmente quando a mesma requer atomicidade na atualização.
  - ☹ A cura saiu pior que a doença?
    - ☺ Não se pudermos relaxar as restrições de consistência!

Consistência e Replicação

1. Introdução
2. Modelos de Consistência Centrados em Dados
3. Modelos de Consistência Centrados no Cliente
4. Gerenciamento de Réplicas
5. Protocolos de Consistência

Modelos de Consistência Centrados em Dados

- A consistência tem sido discutida no contexto de operações de **leitura** e **escrita** em dados compartilhados por meio de:
  - Memória compartilhada (distribuída);
  - Banco de dados (distribuído);
  - Sistema de arquivos (distribuído);
- Os termos listados anteriormente serão chamados indistintamente de **depósito de dados**.
- Cada processo que pode acessar dados do depósito tem uma cópia local (ou próxima) disponível do depósito inteiro.
- Operações de escrita são propagadas para outras cópias.
  - Alterou dados? → Escrita. Não alterou dados? → Leitura.

Modelos de Consistência Centrados em Dados
Propagação de operações de escrita

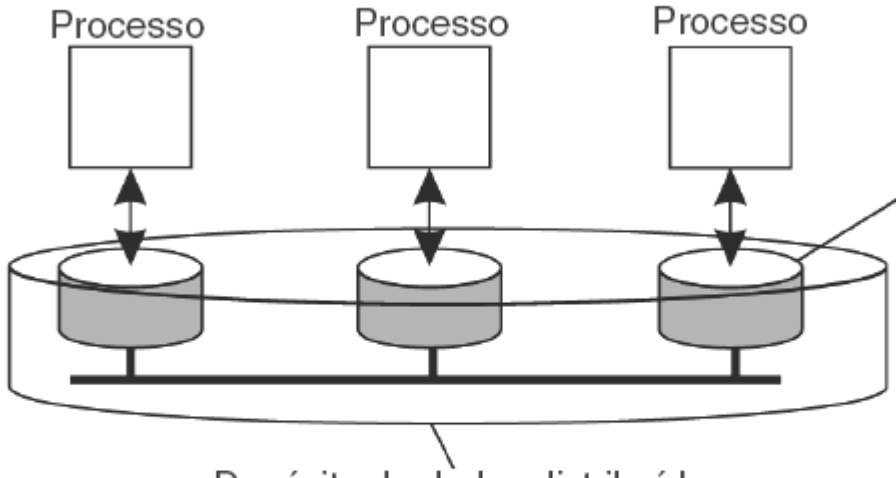


Figura 7.1 Organização geral de um depósito de dados lógico, fisicamente distribuído e replicado por vários processos.

- **Modelo de Consistência** – contrato entre processos e depósitos que firma que o depósito funcionará de maneira correta se os processos concordarem em obedecer certas regras.

Modelos de Consistência Centrados em Dados
Consistência Contínua

- Quais inconsistências são toleráveis?
  - Definidas a partir de três eixos, que formam faixas de **consistência contínua** (Yu e Vahdat):
    1. Desvio em valores numéricos entre réplicas;
       - Cópias não podem variar por mais que dado valor absoluto ou relativo;
       - Número de atualizações não vistas por outras réplicas;
    2. Desvio em idade entre réplicas;
       - Última vez que réplica foi atualizada;
    3. Desvio em relação à ordenação de operações de atualização.
       - Tolera ordenação diferente temporariamente aplicadas provisoriamente a uma cópia local temporária, mas que devem ser refeitas na ordem correta antes de se tornarem permanentes.

Modelos de Consistência Centrados em Dados
Consistência Contínua – Noção de uma Conit

- Proposta uma unidade de consistência: *conit* (consistency + unit).

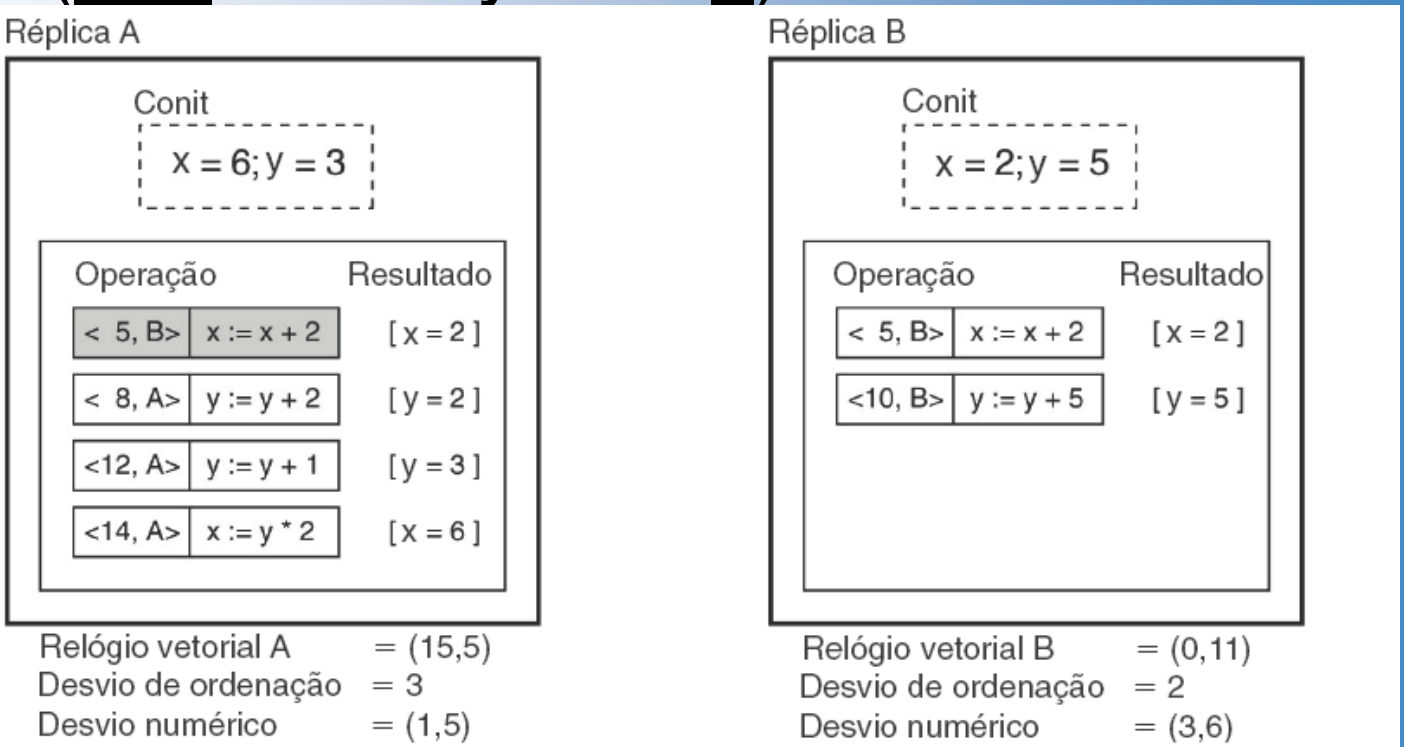


Figura 7.2 Exemplo de monitoração de desvios de consistência (adaptado de Yu e Vahdat, 2002).

Modelos de Consistência Centrados em Dados
Consistência Contínua – Granularidade de Conit

- Necessário compromisso para manter conits de granularidade grossa e conits de granularidade fina.
  - Ex.: diferença de duas réplicas não pode ter mais que uma atualização pendente:

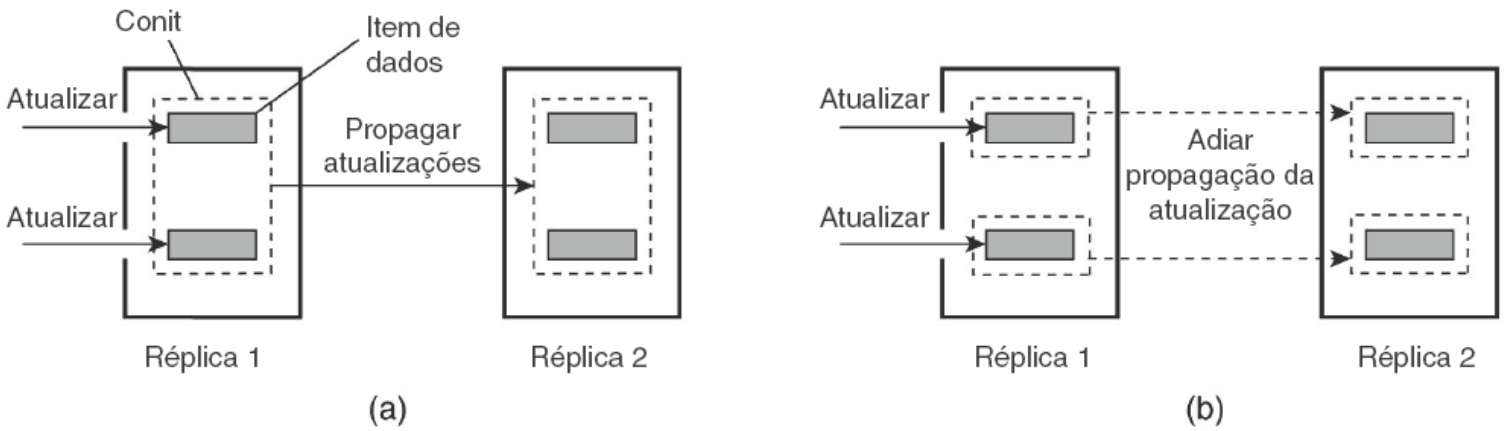


Figura 7.3 Escolha da granularidade adequada para uma conit. (a) Duas atualizações resultam em propagação da atualização. (b) Nenhuma propagação de atualização é necessária (ainda).

Modelos de Consistência Centrados em Dados
Consistência Contínua – Protocolos

- Necessário tratar duas questões para se por as *conits* em prática:
  - É necessário impor protocolos para impor consistência;
  - Programadores devem especificar os requisitos de consistência para aplicações.
- Pode ser implementada como um conjunto de ferramentas parecido com bibliotecas.
  - Conits são declaradas junto com a atualização de um item de dados (semelhante a semáforos).

```
AffectsConit(ConitQ, 1, 1);
Inclua mensagem m na fila Q;
..
DependsOnConit(ConitQ, 4, 0, 60);
Leia mensagem m da frente da fila Q;
```

Operações não vistas;

Atualizações provisórias

Idade da cópia (segundos)

Modelos de Consistência Centrados em Dados
Ordenação consistente de operações

- Modelos de programação concorrente que tratam de ordenar operações consistentemente em dados compartilhados replicados:
  - Consistência seqüencial;
  - Consistência causal.
- Ampliam os modelos de consistência contínua no sentido de que, quando for preciso comprometer atualizações provisórias em réplicas, estas terão de chegar a um acordo sobre uma ordenação global dessas atualizações.

Modelos de Consistência Centrados em Dados
Consistência seqüencial

- Definido em 1979 por Lamport. Diz-se que um depósito de dados é seqüencialmente consistente quando satisfaz a seguinte condição:
  - O resultado de qualquer execução é o mesmo que seria se as operações (de leitura e escrita) realizadas por todos os processos no depósito de dados fossem executadas na mesma ordem seqüencial e as operações de cada processo individual aparecessem nessa seqüência na ordem especificada por seu programa.
- Logo, qualquer intercalação válida de operações é aceitável, mas *todos os processos vêem a mesma intercalação de operações*.
  - Nada é dito sobre o tempo;
  - O processo "vê" escritas de todos, mas apenas as suas próprias leituras.

Modelos de Consistência Centrados em Dados
Consistência seqüencial

Notação adotada:

$W_i(x)a$ → Processo *i* Escreve valor *a* em item de dados *x*

$R_i(x)b$ → Processo *i* Lê valor b em item de dados *x*

P1:	W(x)a		
P2:		R(x)NIL	R(x)a

Figura 7.4 Comportamento de dois processos que operam sobre o mesmo item de dados. O eixo horizontal representa o tempo.

P1:	W(x)a			
P2:		W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)b	R(x)a

(a)

P1:	W(x)a			
P2:		W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

(b)

Figura 7.5 (a) Depósito de dados seqüencialmente consistente. (b) Depósito de dados que não é seqüencialmente consistente.

Modelos de Consistência Centrados em Dados
Consistência seqüencial

Processo P1	Processo P2	Processo P3
x ← 1;	y ← 1;	z ← 1;
print(y, z);	print(x, z);	print(x, y);

Figura 7.6 Três processos que executam concorrentemente.

x ← 1;	x ← 1;	y ← 1;	y ← 1;
print(y, z);	y ← 1;	z ← 1;	x ← 1;
y ← 1;	print(x, z);	print(x, y);	z ← 1;
print(x, z);	print(y, z);	print(x, z);	print(x, z);
z ← 1;	z ← 1;	x ← 1;	print(y, z);
print(x, y);	print(x, y);	print(y, z);	print(x, y);
Impressões: 001011	Impressões: 101011	Impressões: 010111	Impressões: 111111
Assinatura: 001011	Assinatura: 101011	Assinatura: 110101	Assinatura: 111111
(a)	(b)	(c)	(d)

Figura 7.7 Quatro seqüências de execução válidas para os processos da Figura 7.6. O eixo vertical é o tempo.

Modelos de Consistência Centrados em Dados
Consistência Causal

- Representa um enfraquecimento da consistência seqüencial no sentido que faz distinção entre eventos que são potencialmente relacionados por causalidade e os que não são.
  - Se o evento *b* é causado ou influenciado por um evento anterior *a*, então é necessário que todos vejam **primeiro** *a* e, **depois**, *b.*
    Se o processo P1 escreve *x*, em seguida P2 lê *x* e escreve *y*, os dois primeiros são **causais**, porém as duas escritas são **concorrentes**.

Modelos de Consistência Centrados em Dados
Consistência Causal

- Para um depósito de dados ser considerado consistente por causalidade, é necessário que obedeça à seguinte condição:

  Escritas que são potencialmente relacionadas por causalidade devem ser vistas por todos os processos na mesma ordem. Escritas concorrentes podem ser vistas em ordem diferente em máquinas diferentes.

Modelos de Consistência Centrados em Dados
Consistência Causal

P1:	W(x)a			W(x)c	
P2:		R(x)a	W(x)b		
P3:		R(x)a		R(x)c	R(x)b
P4:		R(x)a		R(x)b	R(x)c

P1: W(x)a				
P2:	R(x)a	W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

P1: W(x)a				
P2:		W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

Implementar consistência causal requer monitorar quais processos viram quais escritas.

Modelos de Consistência Centrados em Dados
Operações de agrupamento

- Consistências seqüencial e causal são definidas no nível de operação de leitura e escrita.
  - Granularidade fina (baseada em memória compartilhada em um mesmo hardware)
- Algumas vezes é incompatível com a granularidade fornecida pelas aplicações.
  - Granularidade grossa, mantida por meio de mecanismos de sincronização para exclusão mútua e transações.

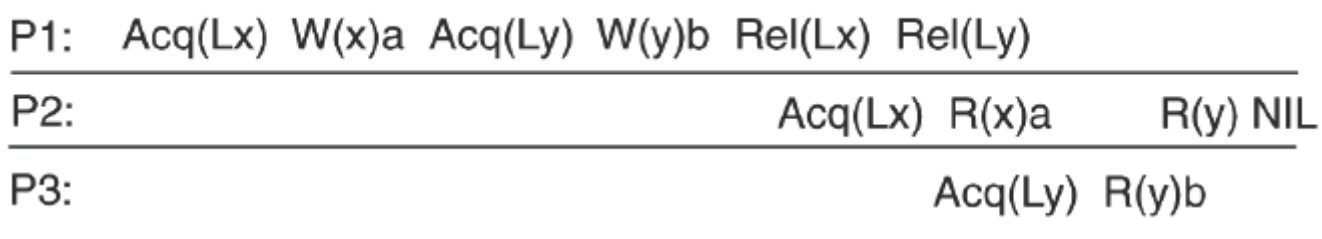


Figura 7.10 Seqüência válida de eventos para consistência de entrada.

Modelos de Consistência Centrados em Dados
Consistência vs. Coerência

- **Modelo de Consistência** descreve o que pode ser esperado com relação ao conjunto de itens de dados quando vários processos operam concorrentemente sobre aqueles dados.
  - Um conjunto é consistente se adere às regras descritas pelo modelo.
- **Modelos de Coerência** descreve o que pode ser esperado para só um item de dados.
  - Um item replicado em vários lugares adere às regras definidas pelo seu modelo de coerência associado.
  - Modelo popular: consistência seqüencial, mas aplicado a só um item de dados.
    - No caso de escritas concorrentes, a certa altura todos os processos verão ocorrer a mesma ordem de atualizações.

Consistência e Replicação

1. Introdução
2. Modelos de Consistência Centrados em Dados
3. Modelos de Consistência Centrados no Cliente
4. Gerenciamento de Réplicas
5. Protocolos de Consistência

Modelos de consistência Centrados no cliente
Consistência eventual

- Um modelo de consistência muito fraca, no qual depósitos de dados são caracterizados pela ausência de atualizações simultâneas ou, quando tais atualizações acontecem, elas podem ser resolvidas com facilidade.
- A maioria das operações envolve ler dados.
  - Em muitos sistemas de bancos de dados, a maioria dos processos dificilmente executa operações de atualização;
  - No DNS, apenas a autoridade de nomeação pode atualizar sua porção do espaço de nomes;
    - Conflitos **escrita-escrita** nunca ocorrerão, mas sim conflitos **leitura-escrita**.
  - Páginas WWW, que são atualizadas apenas pelo webmaster;
    - Uso de proxies e caches para melhorar eficiência.

Modelos de consistência Centrados no cliente
Consistência eventual

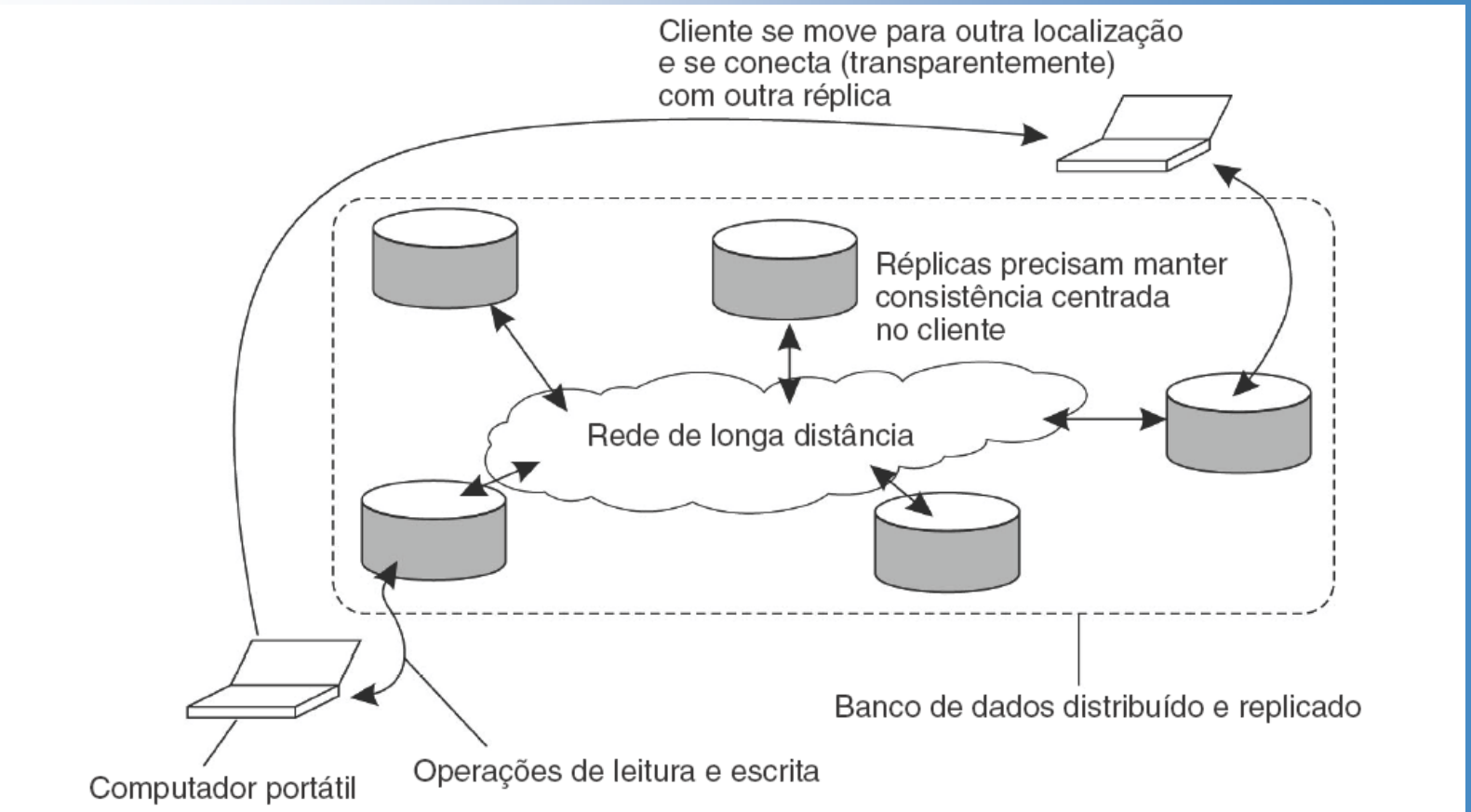


Figura 7.11 O princípio de um usuário móvel que acessa réplicas diferentes de um banco de dados distribuído.

Modelos de consistência Centrados no cliente
Consistência centrada no cliente

- Quando se usa depósitos de dados de consistência eventual e um mesmo usuário pode operar sobre réplicas diferentes, pode-se obter inconsistências.
- Resolvidas a partir de consistência centrada no cliente:
  - Dá garantia *a um único cliente* de consistência de acesso a um depósito de dados por esse cliente;
  - Não há garantia para acessos concorrentes por clientes diferentes.
- Dividido em 4 modelos:
  - Leituras monotônicas;
  - Escritas monotônicas;
  - Leia-suas-escritas;
  - Escritas-seguem-leituras.

Modelos de consistência Centrados no cliente
Leituras monotônicas

- Se um processo ler o valor de um item de dados x, qualquer operação de leitura sucessiva de x executada por esse processo sempre retornará o mesmo valor ou um valor mais recente.
  - Leitura de E-mails “com cache”.

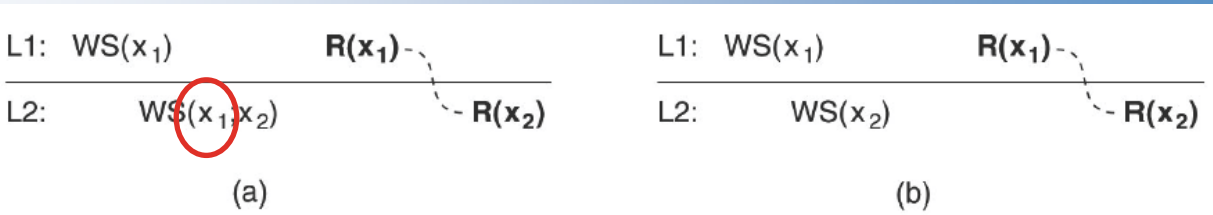


Figura 7.12 Operações de leitura executadas por um único processo *P* em duas cópias locais diferentes do mesmo depósito de dados. (a) Depósito de dados que oferece consistência de leitura monotônica. (b) Depósito de dados que não oferece consistência de leitura monotônica.

Modelos de consistência Centrados no cliente
Escritas monotônicas

- Uma operação de escrita executada por um processo em um item de dados x é concluída antes de qualquer operação de escrita sucessiva em x pelo mesmo processo.
  - Atualização versões de bibliotecas substituindo uma ou mais funções

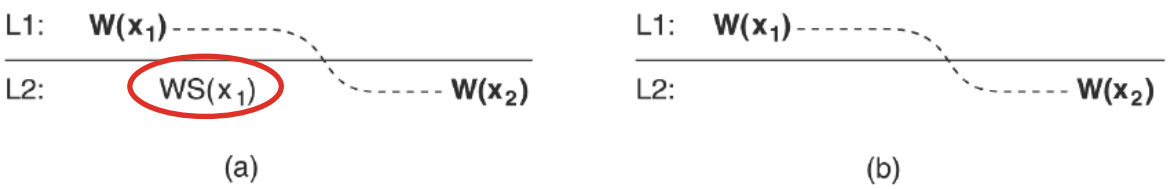


Figura 7.13 Operações de escrita executadas por um único processo *P* em duas cópias locais diferentes do mesmo depósito de dados. (a) Depósito de dados consistente por escrita monotônica. (b) Depósito de dados que não oferece consistência por escrita monotônica.

Modelos de consistência Centrados no cliente
Leia-suas-escritas

- O efeito de uma operação de escrita por um processo no item de dados x sempre será visto por uma operação de leitura sucessiva em x pelo mesmo processo.
  - Contra-exemplo: Atualizamos uma página no servidor, mas quando abrimos a página vemos a versão antiga (problema com o cache).

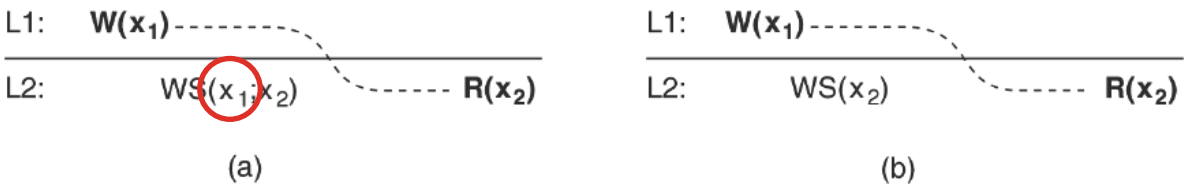


Figura 7.14 (a) Depósito de dados que oferece consistência leia-suas-escritas. (b) Depósito de dados que não fornece tal consistência.

Modelos de consistência Centrados no cliente
Escritas-seguem-leituras

- Garante-se que uma operação de escrita por um processo em um item de dados x em seguida a uma operação de leitura anterior em x pelo mesmo processo ocorre sobre o mesmo valor, ou sobre o valor mais recente de x que foi lido.
  - Garantir que usuários de um grupo de discussão em rede vejam a apresentação de uma reação a um artigo somente depois de terem visto o artigo original.

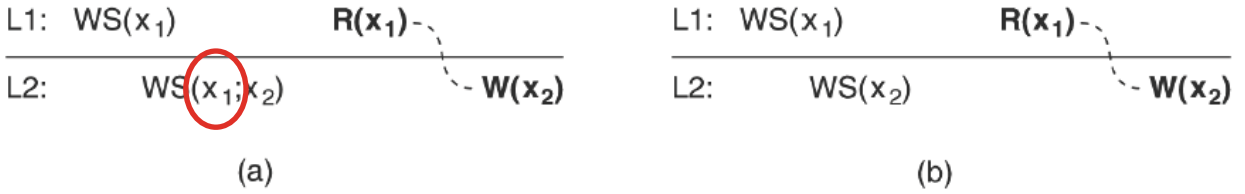


Figura 7.15 (a) Depósito de dados consistente por escritas-seguem-leituras. (b) Depósito de dados que não provê consistência escritas-seguem-leituras.

Consistência e Replicação

1. Introdução
2. Modelos de Consistência Centrados em Dados
3. Modelos de Consistência Centrados no Cliente
4. Gerenciamento de Réplicas
5. Protocolos de Consistência

Gerenciamento de Réplicas

- Onde, quando e por quem réplicas devem ser posicionadas?
  - Posicionar *servidores de réplicas*;
    - É o posicionamento de hardware: encontrar as melhores localizações para colocar um servidor que pode hospedar depósito de dados (ou parte dele);
  - Posicionar *conteúdo*;
    - É o posicionamento dos dados e softwares: encontrar o melhor servidor para colocar conteúdo.

Gerenciamento de Réplicas
Posicionamento do servidor de réplicas

- Questão gerencial e comercial mais do que problema de otimização, e, portanto, pouco estutado;
- Consiste em selecionar as melhores *K* de *N* localizações para se instalar servidores de réplicas;
  - Resolvidos por heurísticas baseadas na distância (latência, largura de banda) entre clientes e localizações;
  - Ignorar posições de clientes considerando a Internet como um conjunto de Sistemas Autônomos (AS) e distribuir replicações nos AS que possuem maior número de enlaces;
- Estes algoritmos apresentam complexidade mais alta que $O(N^2)$, sendo que a demora para o cálculo mesmo para poucos milhares de localizações leva algumas dezenas de minutos, podendo ser inaceitável quando há *flash crowds* (multidões instantâneas).

Gerenciamento de Réplicas
Posicionamento do servidor de réplicas

- Identificar regiões para posicionamento de réplicas contendo nós que acessam o mesmo conteúdo.
- Necessário determinar o tamanho das células, feito pela função da distância média entre dois nós e do número de réplicas requeridas.

- Funciona tão bem quanto os anteriores, mas com complexidade O(*N**MAX(log(N),K))
- 20 réplicas para 64000 nós = 50000 vezes mais rápido.
- Pode ser feito em tempo real!

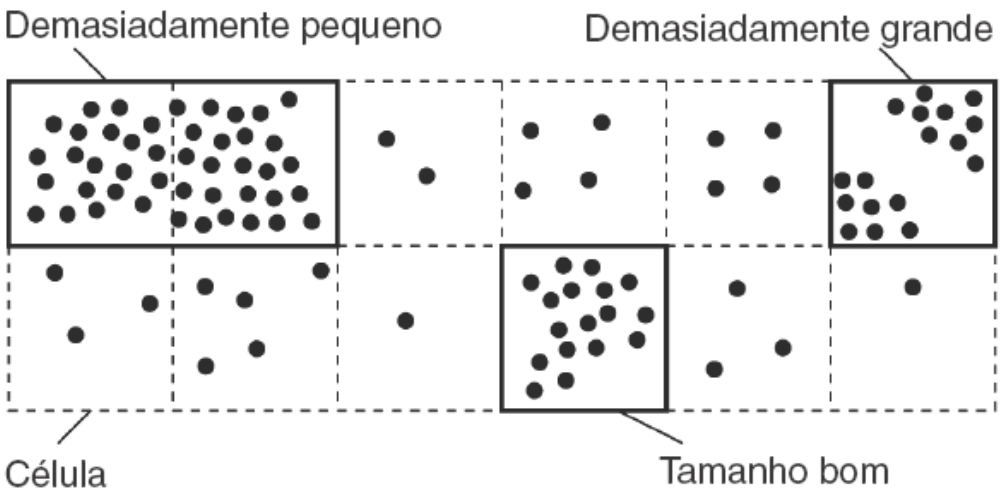


Figura 7.16 Escolha de um tamanho adequado de célula para posicionamento de servidor.

Gerenciamento de Réplicas
Replicação e posicionamento de conteúdo

- São distinguidos três tipos de réplicas organizadas logicamente:
  - **Réplicas permanentes**: conjunto inicial de réplicas que constituem um depósito distribuído;
    - Servidores que estão em uma única localização;
    - **Espelhamento** (servidores geograficamente espalhados pela Internet).
  - **Réplicas iniciadas por servidor**: cópias de um depósito de dados para aprimorar desempenho e criadas por iniciativa do (proprietário do) depósito de dados.
    - Para reduzir carga do servidor;
    - Replicação ou migração de arquivos para proximidade de clientes que emitem muitas requisições;
  - Réplicas iniciadas por cliente;
    - Cache na máquina do cliente;
    - Cache em máquina compartilhada por clientes de uma LAN.

Gerenciamento de Réplicas
Replicação e posicionamento de conteúdo

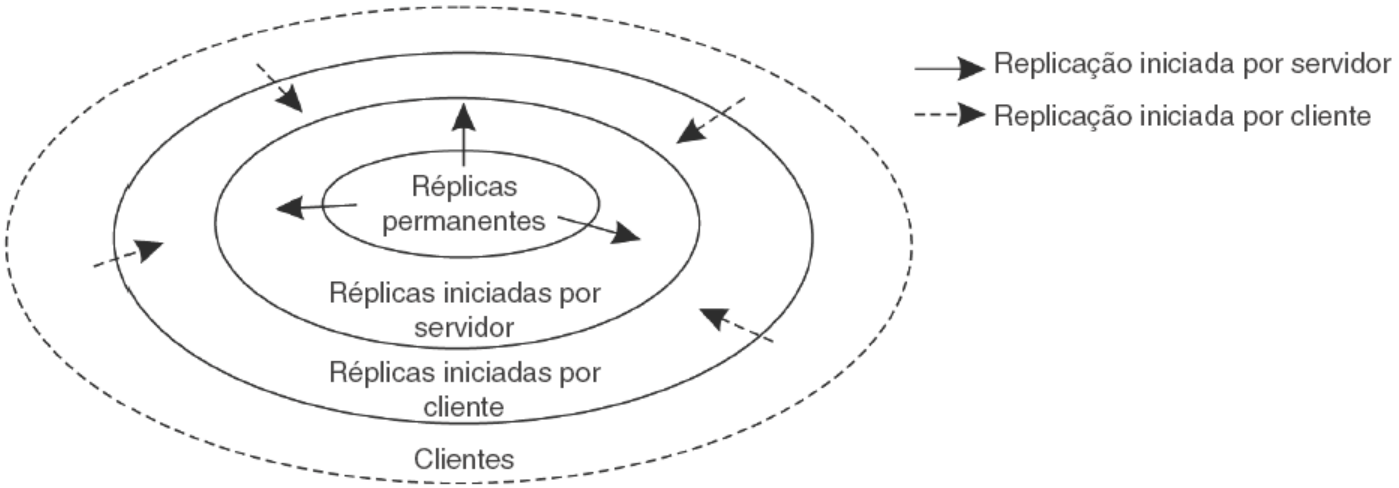


Figura 7.17 Organização lógica de diferentes tipos de cópias de um depósito de dados em três anéis concêntricos.

Gerenciamento de Réplicas
Réplicas iniciadas por servidor

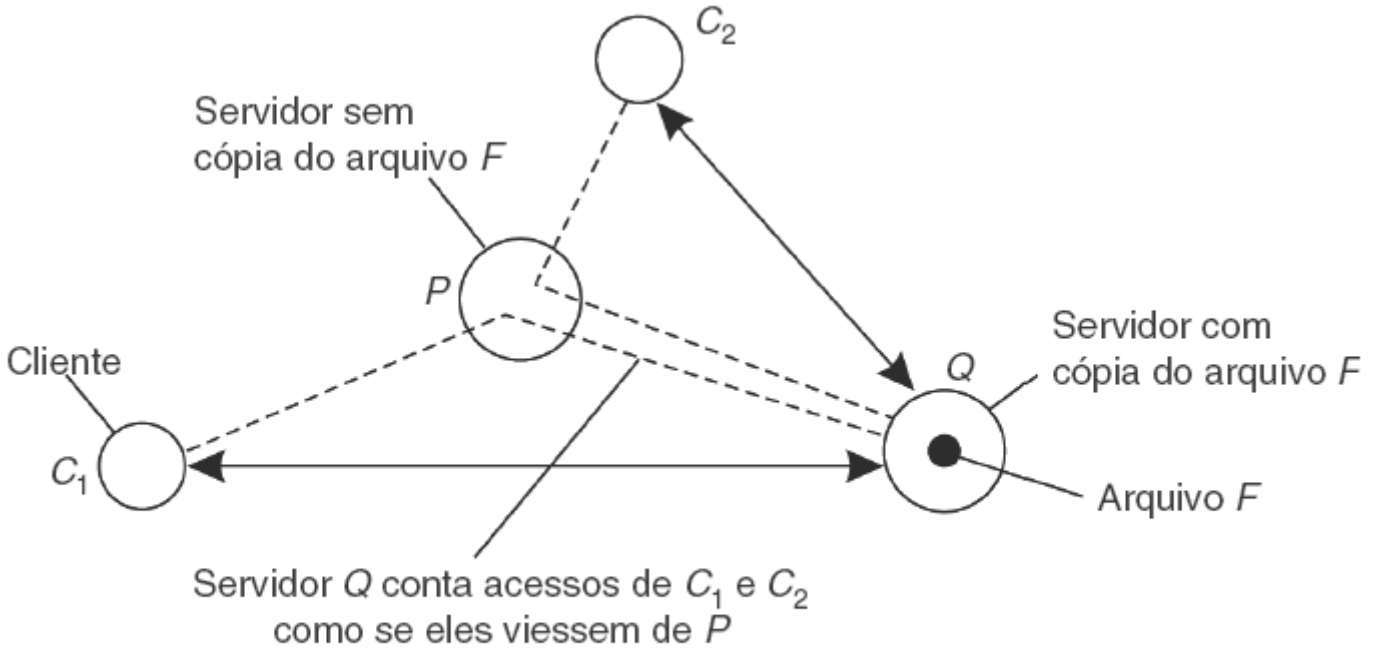


Figura 7.18 Contagem de requisições de acesso de clientes diferentes.

Gerenciamento de Réplicas
Distribuição de conteúdo

- O gerenciamento de réplicas também trata da propagação de conteúdo atualizado para servidores de réplicas relevantes, seguindo os compromissos:
  - Estado vs. operações
  - Protocolos de recuperação de atualização vs. protocolos de envio de atualizações
  - Unicast vs. multicast

Gerenciamento de Réplicas
Estado vs. Operações

- O que deve ser propagado?
  1. Somente uma notificação de uma atualização;
     - Protocolos de invalidação: informa cópias que dados foram modificados, podendo especificar qual parte está inválida;
     - Usam pouca largura de banda. Útil quando relação leitura/escrita é relativamente pequena.
  2. Transferir dados de uma cópia para outra;
     - Várias modificações empacotadas em uma única mensagem;
     - Útil quando a taxa leitura/escrita é relativamente alta;
  3. Propagar a operação de atualização para outras cópias;
     - Não transferir dados, mas informar a cada réplica qual operação de atualização ela deve realizar (replicação ativa).

Gerenciamento de Réplicas
Recuperação vs. envio de atualizações

- Atualizações são recuperadas ou enviadas?
  - **Abordagem baseada em envio**: atualizações são propagadas para outras réplicas sem que essas réplicas tenham solicitado essas atualizações;
    - Usadas quando é necessário alto grau de consistência
    - Dados consistentes estão disponíveis imediatamente quando solicitados.
  - **Abordagem baseada em recuperação de atualizações**: um servidor ou cliente requisita que um outro servidor lhe envie quaisquer atualizações que ele tiver no momento em questão.
    - Usados por cache Web: verificam se itens estão atualizados, atualizando caso necessário.
    - Tempo de resposta aumenta em *ausência* na cache.

Gerenciamento de Réplicas
Comparação entre os protocolos

Assunto	Baseadas em envio	Baseadas em recuperação
Estado no servidor	Lista de réplicas e caches de clientes	Nenhum
Mensagens enviadas	Atualizar (e possivelmente buscar atualização mais tarde)	Sondar e atualizar
Tempo de resposta no cliente	Imediato (ou tempo de busca–atualização)	Tempo de busca–atualização

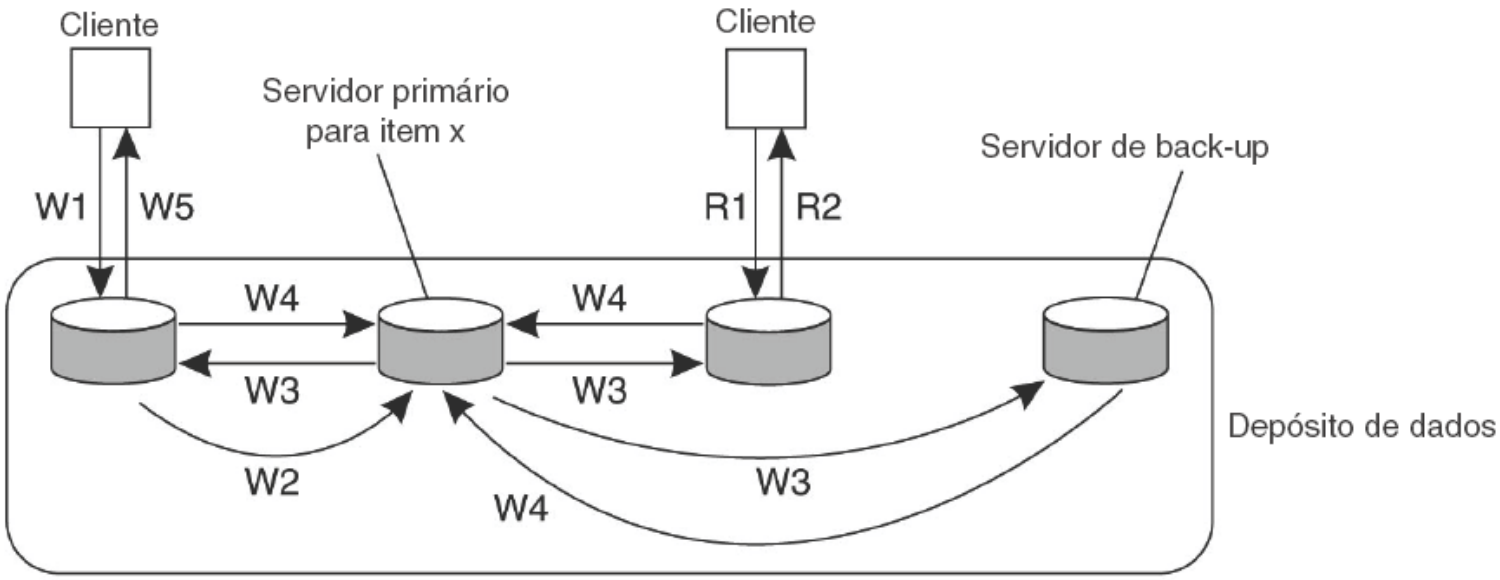
Tabela 7.1 Comparação entre protocolos baseados em recuperação de atualizações e envio de atualizações no caso de sistemas com múltiplos clientes e com um único servidor.

Gerenciamento de Réplicas
Unicast vs. Multicast

- Para enviar ou recuperar atualizações, é necessário decidir por unicast ou multicast:
  - Redes locais? Broadcast com custo de unicast, sendo melhor que multicast;
  - Para propagar atualizações, o envio de atualizações cuidadosamente integradas podem ser enviadas a um único grupo multicast;
  - Para abordagem baseada em recuperação, geralmente é apenas 1 cliente ou servidor que requisita a atualização, sendo o unicast mais eficiente.

Consistência e Replicação

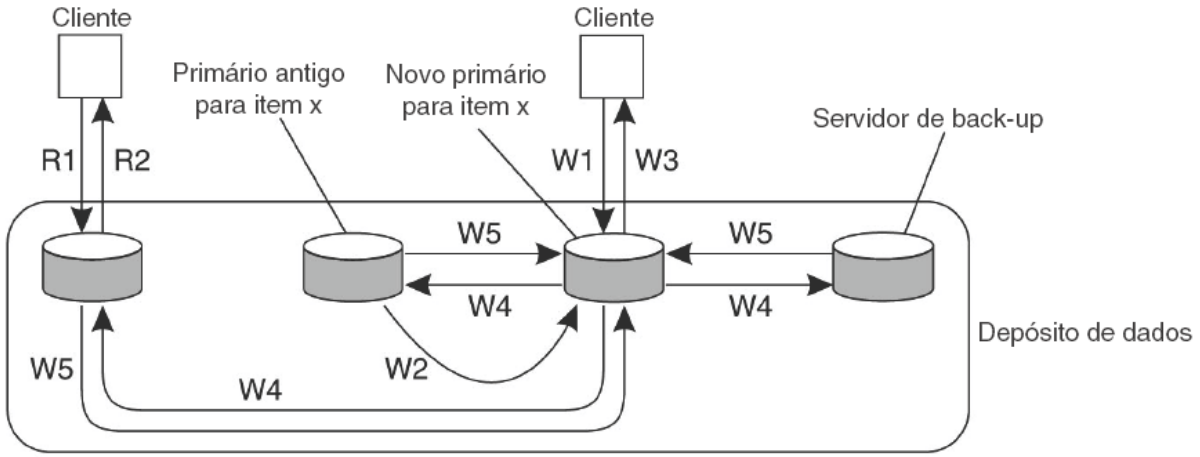
1. Introdução
2. Modelos de Consistência Centrados em Dados
3. Modelos de Consistência Centrados no Cliente
4. Gerenciamento de Réplicas
5. Protocolos de Consistência



W1. Requisição de escrita
W2. Repassa requisição ao primário
W3. Diz aos back-ups para atualizar
W4. Reconhece atualização
W5. Reconhece escrita concluída

R1. Requisição de leitura
R2. Resposta à leitura

Figura 7.19 Princípio de um protocolo de primário e backup.



W1. Requisição de escrita
W2. Move item x para novo primário
W3. Reconhece escrita concluída
W4. Diz aos back-ups para atualizar
W5. Reconhece atualização

R1. Requisição de leitura
R2. Resposta à leitura

Figura 7.20 Protocolo de primário e backup no qual a cópia primária migra para o processo que quer realizar uma atualização.

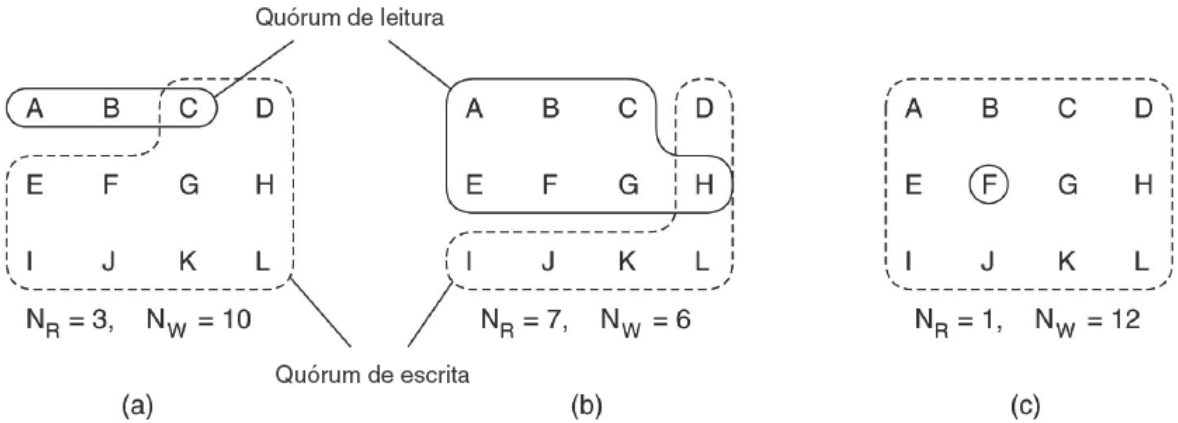


Figura 7.21 Três exemplos do algoritmo de votação. (a) Escolha correta de conjunto de leitura e de escrita. (b) Escolha que pode levar a conflitos escrita–escrita. (c) Escolha correta, conhecida como ROWA (lê uma, escreve todas).