

Consistência e Replicação





Razões para replicar

- Aumentar a fiabilidade do Sistema
 - Continuidade de operação após uma falha
 - Proteção contra corrupção dos dados
- Performance
 - Cobrir áreas geográficas distantes
 - Atender números elevados de solicitações
- Porque não replicar?
 - Eventual falta de consistência entre replicas

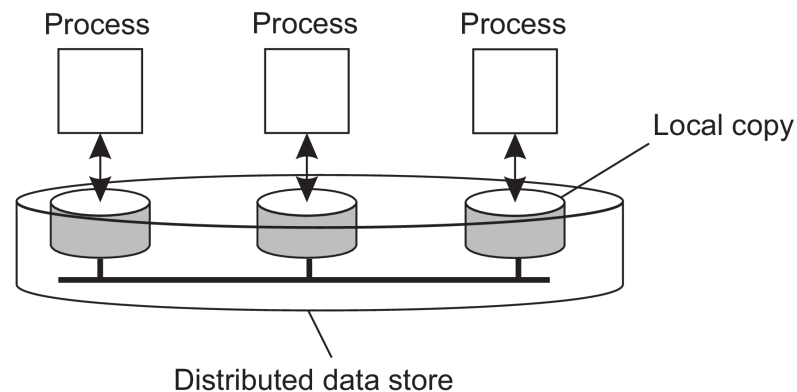


Performance e Escalabilidade

- Para manter consistência entre réplicas, é geralmente necessário garantir que as operações conflitantes são feitas pela mesma ordem.
- Operações que geram conflitos:
 - Read-write: operação de leitura e escrita concorrentes
 - Write-write: duas operações de escrita
- Garantir uma ordem global em operações conflitantes pode ser uma operação onerosa, o que baixa a escalabilidade.
 - Solução: diminuir requisitos de consistência para que a necessidade de sincronismo global seja evitada.

Modelos de consistência Data-centric

- Modelo de consistência
 - Um contrato entre uma data store (distribuída) e processos, em que a data store especifica precisamente qual o resultado de uma operação de leitura e escrita em concorrência.





Consistência contínua

- Podemos falar do grau de consistência:
 - Replicas podem diferir no seu **valor numérico**
 - Replicas podem diferir na sua **imutabilidade** relativa
 - Podem existir diferenças no **número e ordem de operações de actualização (update)**
- Conit
 - Unidade de consistência => especifica a **unidade de dados** sobre a qual deve ser medida a consistência

Exemplo: Conit

- Cada réplica tem um vector clock
- B envia a A operação a cinzento
 - A executou permanentemente essa operação
- Conit:
 - **A** tem 3 operações pendentes:
 - Desvio de **ordem** = 3
 - **A** perdeu 2 operações de **B**:
 - Max diff é $70 + 412 = (2, 482)$
 - Desvio **numérico** de 482

Replica A

Conit	d = 558 // distance	
	g = 95 // gas	
	p = 78 // price	
Operation		Result
< 5, B>	g ← g + 45	[g = 45]
< 8, A>	g ← g + 50	[g = 95]
< 9, A>	p ← p + 78	[p = 78]
<10, A>	d ← d + 558	[d = 558]

Vector clock A = (11, 5)
 Order deviation = 3
 Numerical deviation = (2, 482)

Replica B

Conit	d = 412 // distance	
	g = 45 // gas	
	p = 70 // price	
Operation		Result
< 5, B>	g ← g + 45	[g = 45]
< 6, B>	p ← p + 70	[p = 70]
< 7, B>	d ← d + 412	[d = 412]

Vector clock B = (0, 8)
 Order deviation = 1
 Numerical deviation = (3, 686)

Consistência Sequencial

- Definição

- O resultado de qualquer execução é o mesmo se todas as operações em todos os processos forem executadas na mesma ordem sequencial, e as operações de cada processo individual aparecerem na sequência na ordem especificada pelo programa.

- (a) Armazenamento consistente sequencialmente.
- (b) Armazenamento não consistente sequencialmente

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

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

Consistência causal

- Escritas que estão relacionadas casuisticamente, têm que ser vistas por todos processos com a mesma ordem. Escritas concorrentes podem ser vistas com ordens diferentes por diferentes processos.
- (a) Violação de armazenamento casuístico. (b) Sequência correcta de eventos num armazenamento casuístico.

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



Agrupar Operações

- Definição:
 - Acesso a **locks** são consistentes sequencialmente
 - Não são permitidos acessos a **locks**, antes que todas escritas prévias tenham sido completadas
 - Nenhum acesso a dados é permitido até que todos acessos anteriores a **locks** tenham sido feitos.
- Ideia base
 - Não interessa que as leituras e escritas de uma **série** de operações seja imediatamente conhecida por outros processos. Apenas queremos que o **efeito** da serie seja conhecido.

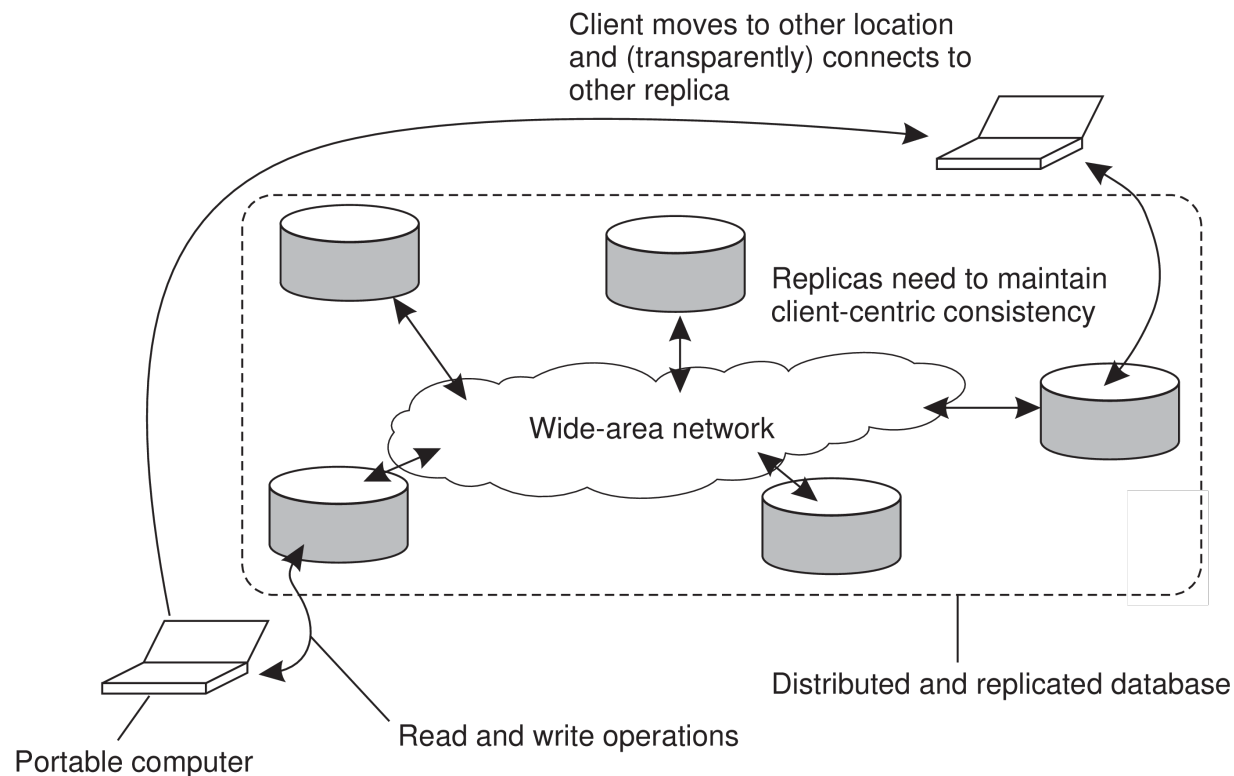
Agrupar Operações (2)

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

Modelos Client centric - Consistência Eventual



Leituras Monotónicas

- Definição
 - Se um processo ler o valor do dado x , qualquer operação de leitura sucessiva sobre x pelo mesmo processo retorna o mesmo valor ou mais recente.
- (a) leitura monotónica (b) leitura não monotónica

L1:	$W_1(x_1)$	$R_1(x_1)$
<hr/>		
L2:	$W_2(x_1; x_2)$	$R_1(x_2)$

L1:	$W_1(x_1)$	$R_1(x_1)$
<hr/>		
L2:	$W_2(x_1 x_2)$	$R_1(x_2)$

Escritas Monotónicas

- Definição
 - Uma operação de escrita por um processo num dado x é completada antes de qualquer operação de escrita sucessiva em x pelo mesmo processo.
- (a) escrita monotónica (b) data store que não disponibiliza consistência por escrita monotónica (c) sem consistência por escrita monotónica (d) Consistente pois $W_1(x_1; x_3)$ apesar de x_1 aparentemente reescrever x_2 .

L1:	$W_1(x_1)$	
<hr/>		
L2:	$W_2(x_1; x_2)$	$W_1(x_2; x_3)$

L1:	$W_1(x_1)$	
<hr/>		
L2:	$W_2(x_1 x_2)$	$W_1(x_1 x_3)$

L1:	$W_1(x_1)$	
<hr/>		
L2:	$W_2(x_1 x_2)$	$W_1(x_2; x_3)$

L1:	$W_1(x_1)$	
<hr/>		
L2:	$W_2(x_1 x_2)$	$W_1(x_1; x_3)$

Leitura após escrita

- Definição
 - O efeito da operação de escrita por um processo nos dados x , será sempre visto pelas operações de leitura sucessivas em x pelo mesmo processo.
- (a) Leitura após escrita consistente (b) Não consistente

L1:	$W_1(x_1)$	
<hr/>		
L2:	$W_2(x_1; x_2)$	$R_1(x_2)$

L1:	$W_1(x_1)$	
<hr/>		
L2:	$W_2(x_1 x_2)$	$R_1(x_2)$

Escrita após leitura


- Definição

- Numa operação de escrita por um processo sobre os dados x após uma operação de leitura prévia sobre x pelo mesmo processo, é garantido que ocorre sobre o mesmo valor de x que foi lido ou mais recente.

- (a) Escrita após leitura consistente (b) não consistente

L1:	$W_1(x_1)$	$R_2(x_1)$
<hr/>		
L2:	$W_3(x_1; x_2)$	$W_2(x_2; x_3)$

L1:	$W_1(x_1)$	$R_2(x_1)$
<hr/>		
L2:	$W_3(x_1 x_2)$	$W_2(x_1 x_3)$



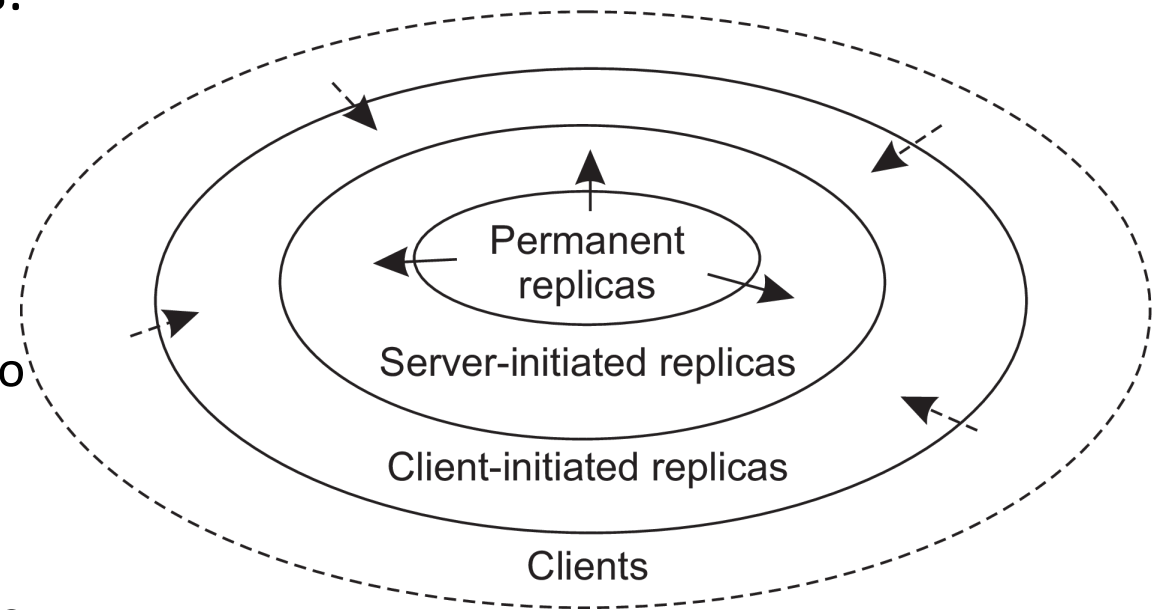
Localização de Replicas

- Descobrir quais os melhores K locais de entre N possíveis localizações
 - Seleccionar a melhor localização de entre N localizações possíveis, para a qual a média da distancia aos clientes é mínima. De seguida escolher o melhor servidor. Repetir o processo K vezes. (caro)
 - Seleccionar os Ks maiores Sistema Autonomo (AS) e colocar um servidor no computador melhor ligado. (caro)
 - Posicionar nós num espaço d-dimensional, onde a distancia reflete a latência. Identificar as K regiões com densidade mais elevada e colocar um servidor em cada uma. (barato)

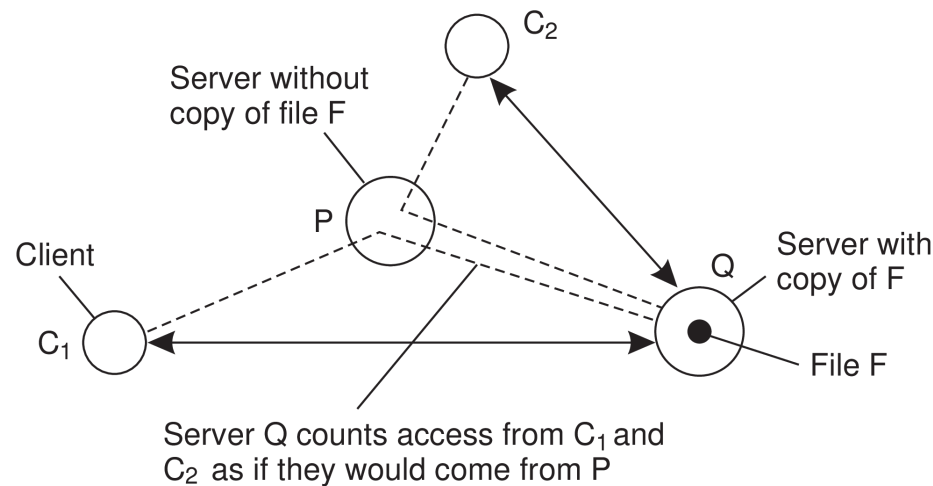
Replicação de conteúdo

- Distinguir os diversos processos:

- Replicas permanentes:
Processo/máquina tem sempre uma réplica
- Replicas iniciadas pelo servidor:
Processo pode dinamicamente hospedar uma réplica a pedido do servidor.
- Replicas iniciada pelo cliente:
Processo pode dinamicamente hospedar uma réplica a pedido do cliente (client cache)



Replicas iniciadas pelo servidor



- Guardar número de acessos por ficheiro, agregar por servidor
- Número de acessos cai abaixo do limite D -> apagar ficheiro
- Número de acessos acima do limite R -> replicar ficheiro
- Número entre D e R -> migrar ficheiro



Distribuição de Conteúdos

- Considerando apenas Cliente-Servidor:
 - Propagar apenas **notificação**/invalidação de uma actualização (caches)
 - Transferir **dados** de uma cópia para outra (DB's distribuídas): **replicação passiva**
 - Propagar a operação de **actualização** para outras cópias: **replicação activa**



Distribuição de Conteúdos: Cliente/Servidor

- Push updates: iniciado pelo servidor, actualização é propagada independentemente do facto do destino ter pedido ou não
- Pulling updates: iniciado pelo cliente, em que o cliente solicita o envio de actualizações
- É possível comutar dinamicamente entre os dois modos através de “leases”: Um contrato no qual o servidor promete fazer o push de updates para o cliente até o contrato expirar

Comparação entre push e pull protocols para replicação

	Push-based	Pull-based
Estado no servidor	Lista de replicas nos clientes e <i>caches</i>	-
Mensagens trocadas	Update (e possivelmente <i>fetch-update</i> se tiver sido uma invalidação)	<i>Poll</i> e <i>Update</i>
Tempo de resposta no cliente	Imediato (ou tempo de <i>fetch-update</i>)	Tempo de <i>Fetch-update</i>



Distribuição de Conteúdos: Leases

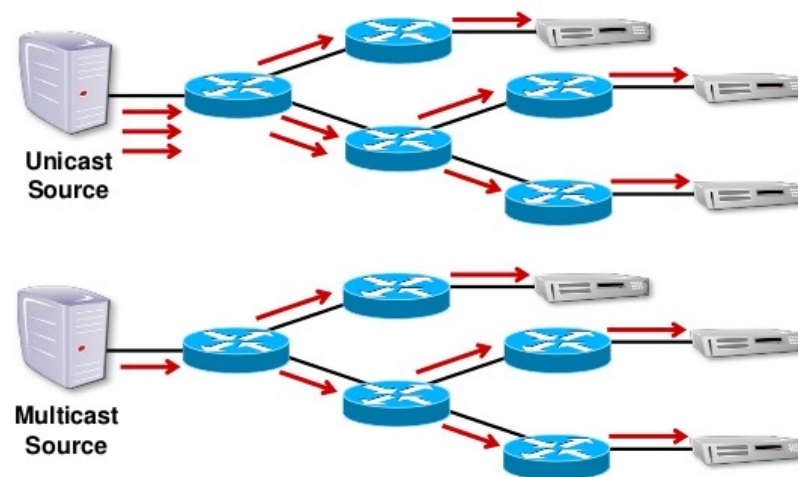
- Tempo de uma "lease" deve ser dependente do comportamento do sistema
 - **Leases baseadas na idade:** De um objecto que não é alterado faz muito tempo, não é esperado que venha a sofrer alteração nos tempos próximos, pelo que o tempo de lease deve ser longo.
 - **Leases com base na frequência de renovação:** Clientes que pedem frequentemente um dado objecto, deverão ter um tempo de expiração mais longo.
 - **Leases baseadas no estado:** Quanto mais carga tiver o servidor, mais curtos devem ser os períodos de lease.

Unicasting vs multicasting

Quando temos disponível um meio de físico de *broadcast*:

Podemos fazer *push* de forma eficiente, com o mesmo custo de ptp

Unicast vs. Multicast



© 2012 Cisco and/or its affiliates. All rights reserved.

Cisco Public

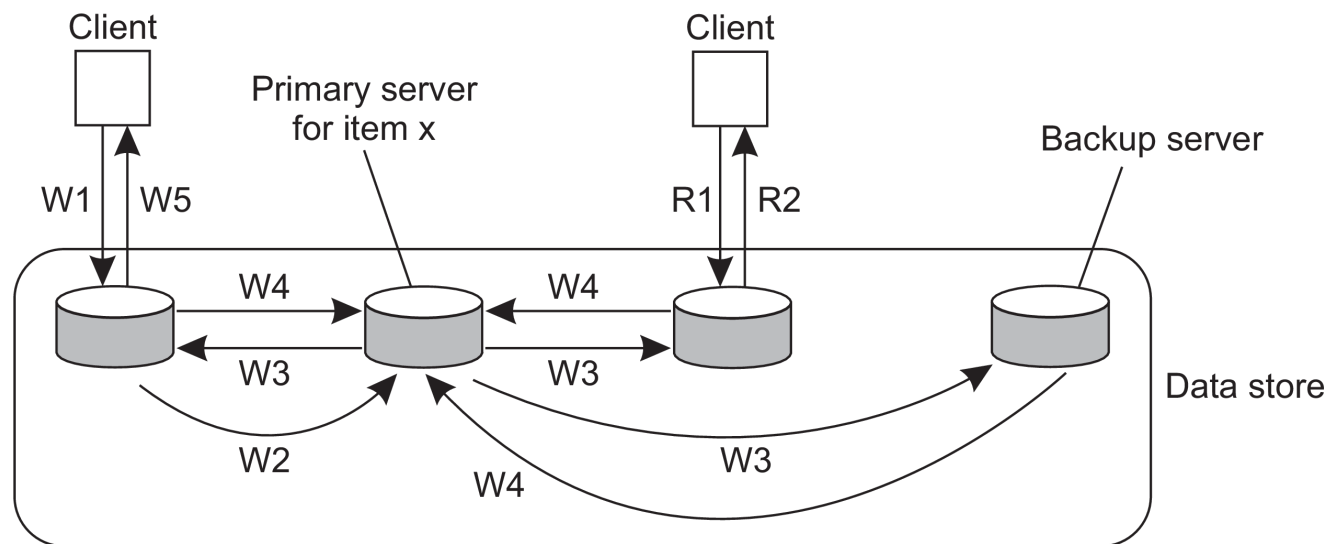
12



Consistência Contínua

- Limitar desvio numérico
 - Garantir que valores não se afastam de um intervalo
- Limitar desvio idade
 - Garantir que não há um atraso de atualizações maior que um intervalo
- Limitar desvio de ordens
 - Garantir um número máximo de ordens em atraso

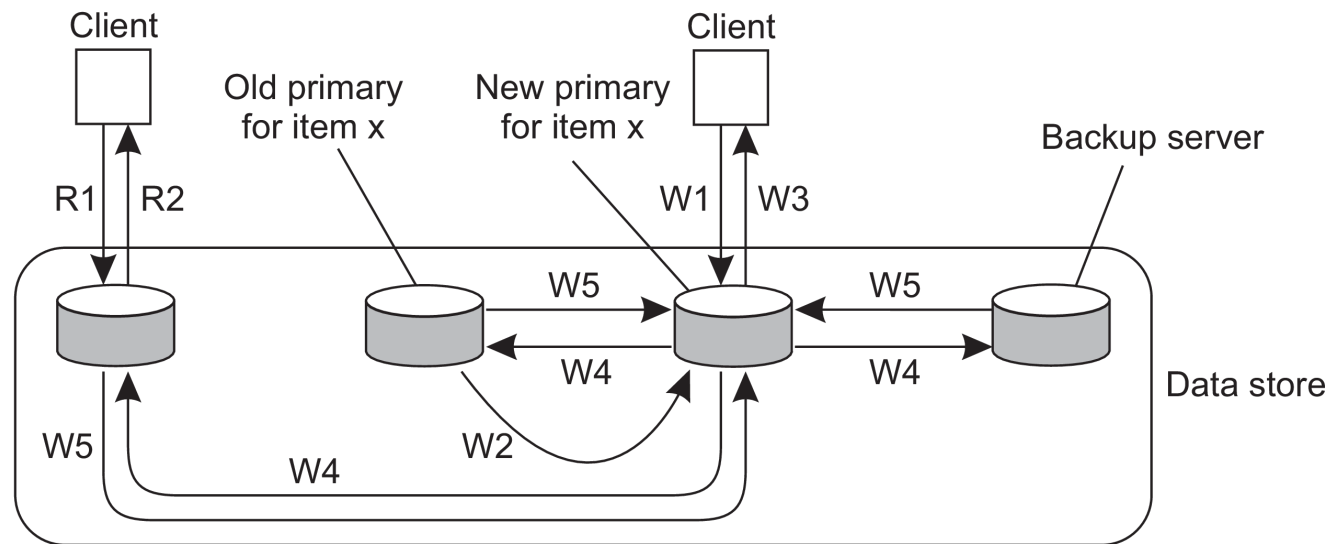
Protocolos baseados num primário



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

Protocolos baseados num primário com escritas locais



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

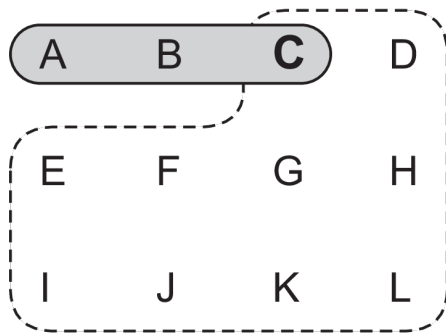
R1. Read request
R2. Response to read

Protocolos escrita replicada

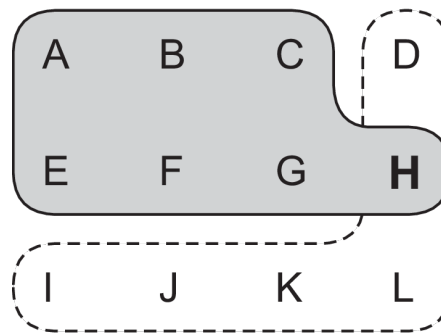
Read Quorum vs Write Quorum

Tem que obedecer as seguintes regras:

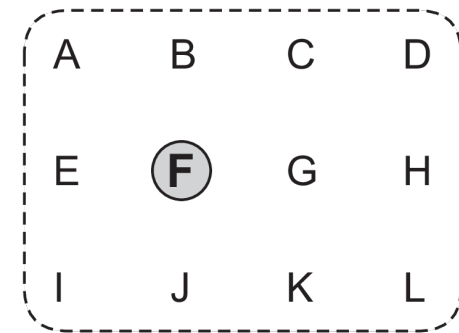
- $N_R + N_W > N$
- $N_W > N/2$



$N_R = 3, \quad N_W = 10$



$N_R = 7, \quad N_W = 6$



$N_R = 1, \quad N_W = 12$

(a) Escolha correcta de read e write (b) Possivel write-write (c) Escolha correcta chamada ROWA (Read One Write All)