Replicação de arquivos

Jonathan de Matos

Sistemas Distribuídos
Prof. Marcos José Santana

Roteiro

- Introdução
- Modelos de consistência
- Protocolos de distribuição
- Propagação de atualizações
- Protocolos de consistência
- Exemplos

Introdução

- Tópico importante
- Técnicas herdadas de outros contextos
- Confiabilidade
  - Proteger contra problemas ou conflitos de gravação
- Desempenho
  - Dividir o tráfego
  - Réplicas mais próximas
- Problemas com consistência
  - Páginas *web*
    - Proximidade e invalidação (cache)

Exemplo em sistemas distribuídos

- Objetos distribuídos
  - Dados
  - Métodos
  - Solução para inconsistência
    - Objeto: medidas específicas para p/ cada cenário
    - Servidor: transparência

Escalabilidade

- Réplicas: reduzir o tempo de acesso
- Problemas para manter réplica atualizadas
  - Ex: acesso muito menos freqüentes que escritas
- Ideal: réplicas sempre atualizadas entre si
  - Taxa de atualização X desempenho
- Métodos não tão exigentes

Modelos de consistência

- Visão de dados
  - Todas as cópias são consideradas
  - Vários clientes
  - A idéia é fornecer cópias idênticas para todos os clientes
- Visão do cliente
  - O cliente “navega” entre as réplicas
  - O cliente sempre deve ver as cópias como a última vista por ele

Modelos de consistência

- Consistência estrita
  - Qualquer leitura em qualquer réplica resulta no valor da última escrita

	T(1)	T(2)	T(3)
P1	Escreve “a”		
P2		Lê “a”	

P1	Escreve “a”		
P2		Lê null	Lê “a”

Modelos de consistência

- Consistência linear
  - A seqüência na qual os processos realizam escritas devem ser vistas em todas as réplicas na mesma ordem, respeitando a linha do tempo (timestamps)

	T(1)	T(2)	T(3)	T(4)	T(5)
P1	Escreve “a”				
P2		Escreve “b”			
P3			Lê “a”		Lê “b”
P4				Lê ”a”	Lê “b”

	T(1)	T(2)	T(3)	T(4)	T(5)
P1	Escreve “a”				
P2		Escreve “b”			
P3			Lê “b”		Lê “a”
P4				Lê ”b”	Lê “a”

Modelos de consistência

- Consistência seqüencial
  - A seqüência na qual os processos realizam escritas devem ser vistas em todas as réplicas na mesma ordem, não necessariamente temporal

	T(1)	T(2)	T(3)	T(4)	T(5)
P1	Escreve “a”				
P2		Escreve “b”			
P3			Lê “b”		Lê “a”
P4				Lê ”b”	Lê “a”

	T(1)	T(2)	T(3)	T(4)	T(5)
P1	Escreve “a”				
P2		Escreve “b”			
P3			Lê “b”		Lê “a”
P4				Lê ”a”	Lê “b”

Modelos de consistência

- Modelo causal
  - Operações que são dependentes devem ser vistas na mesma ordem por todos os processos, operações concorrentes podem ser vistas em diferentes ordens

	T(1)	T(2)	T(3)	T(3)	T(4)	T(5)
P1	Escreve “a”			Escreve “c”		
P2		Lê “a”	Escreve “b”			
P3		Lê “a”			Lê “c”	Lê “b”
P4		Lê “a”			Lê “b”	Lê “c”

	T(1)	T(2)	T(3)	T(4)	T(5)
P1	Escreve “a”				
P2		Lê “a”	Escreve “b”		
P3				Lê “b”	Lê “a”
P4				Lê “a”	Lê “b”

Modelos de consistência

- Consistência FIFO
  - Escritas de um mesmo processo devem ser vistas por todos os processos na seqüência em que foram realizadas, porém escritas entre diferente processos podem ser vista em qualquer ordem

	T(1)	T(2)	T(3)	T(4)	T(5)	T(6)	T(7)
P1	Escreve “a”						
P2		Lê “a”	Escreve “b”	Escreve “c”			
P3					Lê “b”	Lê “a”	Lê “c”
P4					Lê “a”	Lê “b”	Lê “c”

P1	Escreve “a”						
P2		Lê “a”	Escreve “b”	Escreve “c”			
P3					Lê “c”	Lê “a”	Lê “b”
P4					Lê “a”	Lê “b”	Lê “c”

Modelos de consistência

- Consistência fraca
  - Uso de uma variável de sincronização
  - Quando uma sincronização é realizada, todas as réplicas assumem o mesmo estado
- Consistência liberal
  - Uso de uma primitiva “adquirir” e outra “liberar”
  - Imediata ou retardada

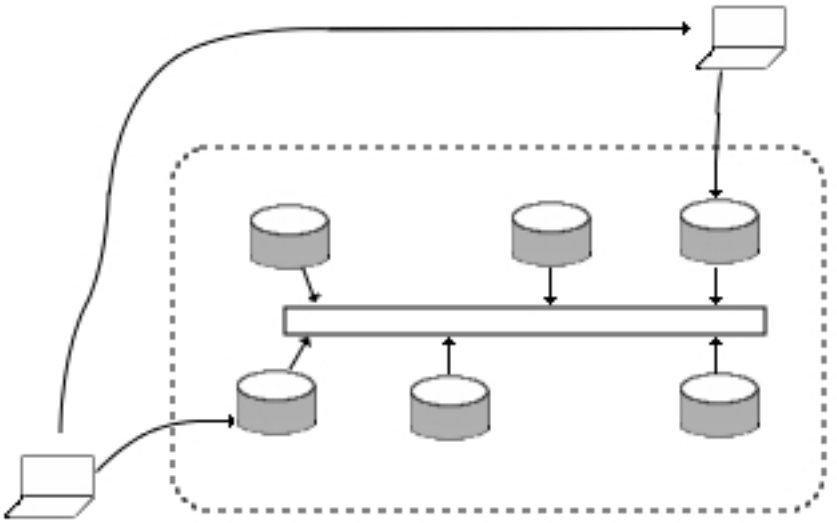
Modelos de consistência

- Consistência de entrada
  - Uso de sincronização
  - Primitivas “adquirir” e “liberar”
  - Não é empregado sobre todo o recurso, apenas uma parte dele

	T(1)	T(2)	T(3)	T(4)	T(5)	T(6)	T(7)	T(8)
P1	Adquirir x	Escrever “a” em x	Adquirir y	Escrever “b” em y	Liberar x	Liberar y		
P2						Adquirir x	Ler “a” de x	Ler null de y
P3							Adquirir y	Ler “b” de y

Modelos de consistência

- Consistência eventual
  - DNS
  - Páginas *web*
  - Consistência atingida com o passar do tempo
  - Tolerância a inconsistência
  - Problema: cliente móvel



Modelos de consistência

- Leituras “monotônicas”
  - Qualquer leitura realizada por um processo retorna sempre o último estado visto por ele ou então um estado mais recente
  - Caixa de email réplicas diferentes
- Gravações “monotônicas”
  - Cada escrita em uma réplica deve ser completada antes que qualquer outra seja efetuada (FIFO para um processo)

Modelos de consistência

- Leitura de suas escritas
  - Sempre que o mesmo processo realizar uma leitura, ele sempre deverá obter o valor de sua última escrita
  - Atualização de páginas *web* e senhas
- Escritas antecedidas por leituras
  - Antes de cada gravação, a réplica é atualizada para o valor mais recente
  - Listas de discussão

Protocolos de distribuição

- Réplicas Permanentes
- Réplicas pelo servidor
- Réplicas pelo cliente

Propagação de atualização

- Estado x operação
  - Notificação
    - Baixa taxa de leitura em relação a escrita
  - Dado modificado
    - Alta taxa de leitura em relação a escrita
  - Transferir apenas as operações
    - Replicação ativa
    - Alto uso de CPU

Propagação de atualização

- Protocolo *push*
  - Baseado no servidor
  - Iniciadas pelo servidor e permanentes
  - Tentativa de obter alto grau de consistência
  - Alta taxa de leitura em relação a escrita
  - *Stateful*
- Protocolo *pull*
  - Baseada no cliente
  - *Cache* de páginas *web*
  - Baixa taxa de leitura em relação a escrita
  - O tempo de acesso aumenta em caso de falta

Propagação de atualização

- Solução híbrida
  - *Leases*
  - Depois de expirado o *lease*, o sistema se torna *pull*
  - Baseado em idade (última modificação)
    - Maior para réplicas mais estáveis, evita-se enviar atualizações que não serão usadas
  - Baseado em freqüência de acesso
    - Mantém-se informação apenas dos cliente mais populares
  - Baseado em sobrecarga
    - Diminui a quantidade de espaço necessário para manter o estado das réplicas

Propagação de atualização

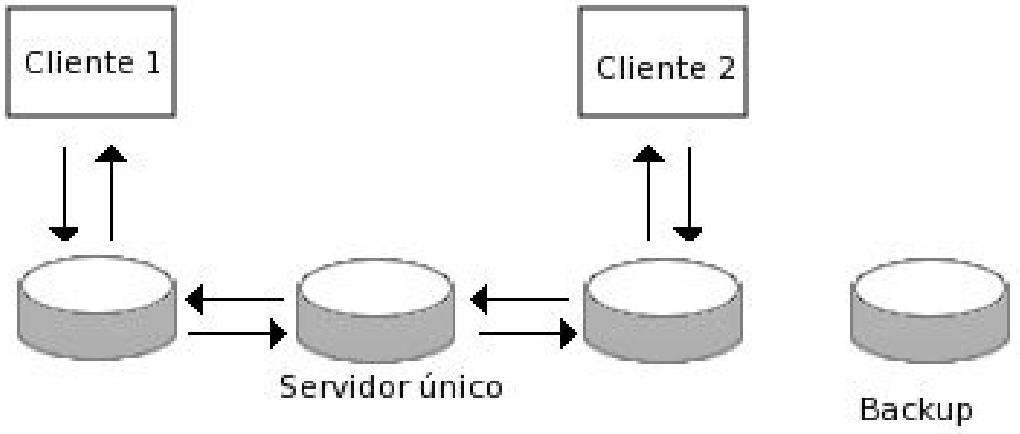
- *Unicast*
  - Melhor para protocolo *pull*
- *Multicast*
  - Melhor para protocolo *push*

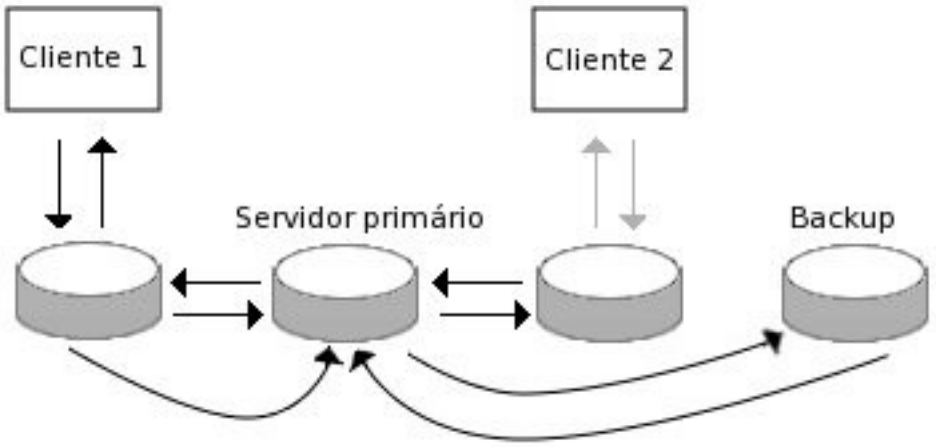
Propagação de atualização

- Protocolos epidêmicos
  - Propagar as atualizações com o mínimo de mensagens possíveis
  - Baseado na mesma idéia da biologia
  - Modelo de anti-entropia
    - Um servidor P escolhe um Q
    - *Push*: ruim quando existem muitos infectivos
    - *Pull*: bom quando existem muitos infectivos
    - Com apenas um infectivo ambos os métodos funcionam
  - Refinamento: propagação de rumores (**Gossip**), análogo a vida real
  - Remoção de dados

Protocolos de consistência

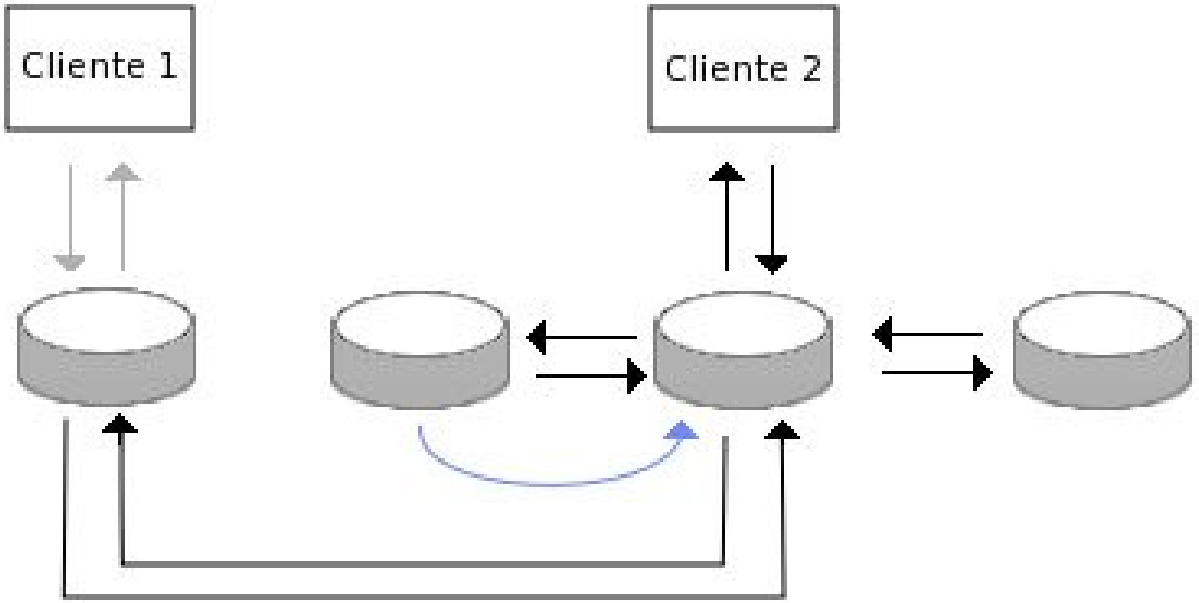
- Baseados em servidor primário (cópia primária)
  - Escrita remota
  - Problemas de tolerância a falhas com chamadas não bloqueantes





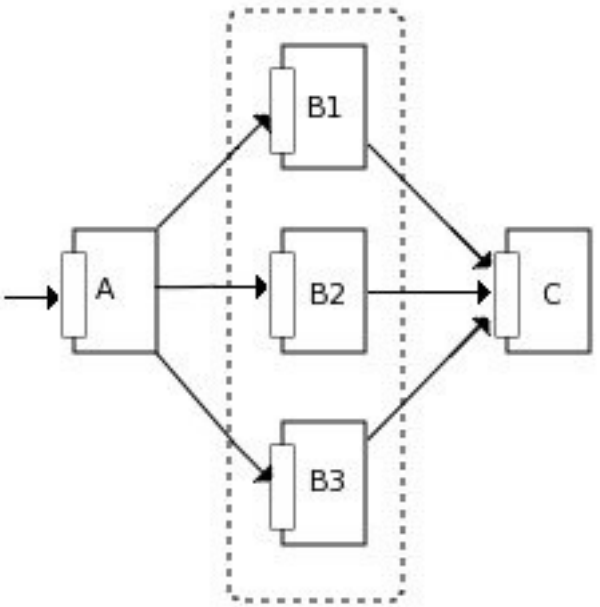
Protocolos de consistência

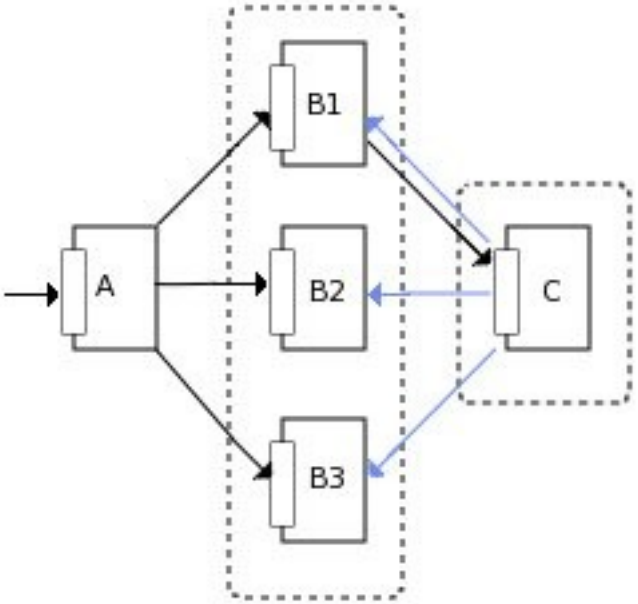
- Baseado em servidor primário
  - Escrita local
  - Bom desempenho para múltiplas escritas
  - Bom desempenho se criado com chamadas não bloqueantes
  - Uso em dispositivos móveis e navegação *offline*



Protocolos de consistência

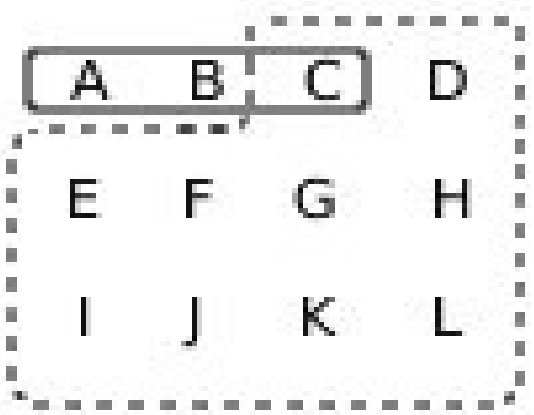
- Escrita replicada
  - Replicação ativa
    - Uso de *timestamps* / *multicast*
    - Baixa escalabilidade
    - Uso de um *sequencer* / centralização
    - Coerência

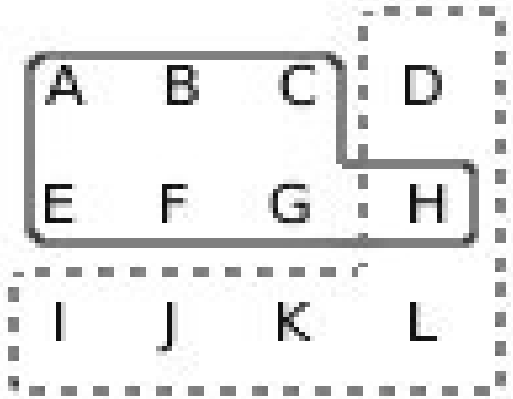




Protocolos de consistência

- Escrita replicada
  - Replicação por votação
  - Escrita em N/2 +1 réplicas
  - Leitura com comparação em N/2
    - $N_w > N/2$ e $N_w + N_r > N$
- Coerência de *cache*





Exemplos

- NFS
  - Deixa bastante aberto para a implementação
  - *Cache – write-back*
  - FS_LOCATIONS
- Plan9
  - Sistema distribuído completo
  - Possui um sistema de arquivo (*diskless*)
  - *Cache – write-through*

Exemplos

- xFS, projeto NOW(Não o XFS, *Silicon Graphics*)
  - *Serverless*
  - *Stripe group / stripe group map*
  - Replicação entre *stripe groups*
  - Consistência por bloco e não por aquivo
  - Modelo de consistência de entrada
  - Recuperação através de *logs*

Exemplos

- CODA
  - Descendente do AFS
  - Manter a semântica do UNIX
  - Computação móvel
  - Exemplo de replicação para desempenho e tolerância a falhas
  - No fechamento do arquivo as modificações são enviadas às réplicas
  - Operação *offline*
  - Uso de *timestamps*
  - Protocolo otimista

Exemplos

- GFS
  - *Google File System*
  - Visa confiabilidade, desempenho e capacidade
  - Uso de metadados
  - Arquivos compostos por *chunks*
  - Os metadados e os *chunks* são replicados
  - *Chunks* são replicados no mínimo 3 vezes
  - Características de *append* nos arquivos
  - Uso de *leases* / *timestamps*
  - Quando um *chunk* é perdido, erros são retornados à aplicação

Conclusão

- Desempenho
- Tolerância a falhas
- Equilíbrio entre ambos
- Modelos difundidos em diversas áreas da computação
- Redes de maior desempenho possibilitam maior evolução da replicação
- Grandes espaços de armazenamento