Sistemas Distribuídos

Aula 08 – Replicação e consistência

DCC/IM/UFRRJ
Marcel William Rocha da Silva

Objetivos da aula

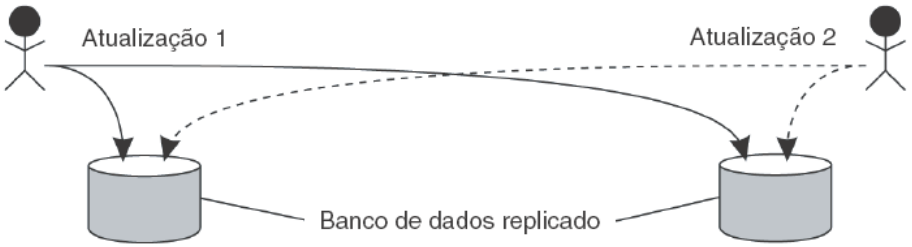
- **Aula anterior**
  - Sincronização
  - Prova
- **Aula de hoje**
  - Razões para a replicação
  - Modelos de consistência
  - Protocolos de consistência
  - Gerenciamento de réplicas

Replicação

- Porque replicar?
  - **Confiabilidade**
    - Trabalho continua mesmo sem alguma réplica
    - Protege contra operações que corrompam dados
  - **Desempenho**
    - Distribuição do trabalho (escalabilidade de tamanho)
    - Reduz o tempo de acesso (escalabilidade geográfica)

Replicação

- Problema:
  - **Consistência**
    - **Exemplo 1**: caches de páginas Web
      - Navegadores armazenam cópia local de páginas acessadas recentemente
      - Tempo de acesso é ótimo
      - Mas cópias locais podem ficar desatualizadas
    - **Exemplo 2**: banco de dados bancários
      - Daria para tolerar cópias desatualizadas? NÃO!

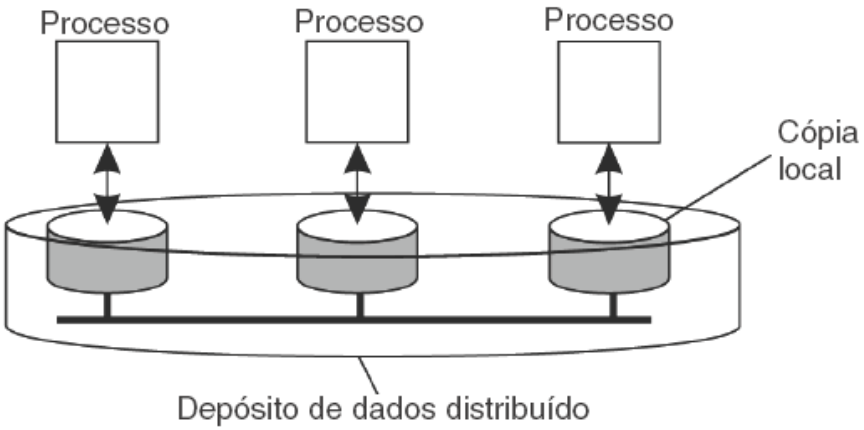


Replicação

- Problema:
  - **Consistência**
    - Uma atualização de um dado em uma réplica deve ser realizada em todas as outras
    - Necessário que as réplicas concordem sobre a ordem com que as atualizações são realizadas
      - Já vimos isso antes → Relógio de Lamport ou vetorial
    - Mas garantir 100% de consistência pode eliminar os ganhos da replicação (ou piorar!)
    - Única solução é relaxar restrições de consistência → permitir que as replicas sejam diferentes em casos específicos

Modelos de consistência centrados em dados

- Consideram que os processos acessam **depósitos de dados**
  - Distribuídos fisicamente
  - Cada processo possui sua cópia local (ou próxima)
  - Operações de **leitura** e **escrita** na cópia local
  - Escritas são propagadas para as outras cópias



Modelos de consistência

- Definição: Contrato entre processos e o depósito de dados
  - Se os processos obedecem certas regras, o depósito de dados funciona corretamente
  - Ex.: processo que lê um item de dados espera obter o valor da última escrita realizada naquele item de dados
- Sem relógios globais é difícil definir a escrita mais recente
  - Modelos de consistência restringem os valores que uma leitura pode retornar

Modelo de consistência contínua

- Define as inconsistências em três eixos independentes (Yu e Vahdat, 2002)
  - Diferenças dos valores numéricos entre réplicas
  - Diferenças das idades entre réplicas
  - Diferença em relação à ordenação de operações de atualização

Modelo de consistência contínua

- **Desvios numéricos** entre réplicas:
  - Dados possuem semântica numérica
  - Ex.1: Registros que contêm preços do mercado de ações
  - Ex.2: Número de atualizações que foram aplicadas a uma determinada réplica, mas que ainda não foram vistas pelas outras

Modelo de consistência contínua

- **Desvios de idade** entre réplicas:
  - Estão relacionados com a última vez que uma réplica foi atualizada
  - Algumas aplicações podem tolerar que uma réplica forneça dados antigos dentro de uma certa tolerância
  - Ex.1: Previsões de tempo, em geral, permanecem razoavelmente exatas durante algum tempo → servidor principal pode receber atualizações em tempos oportunos

Modelo de consistência contínua

- **Desvios de ordenação** entre réplicas:
  - Em algumas aplicações, é permitido que a ordenação das atualizações seja diferente nas várias réplicas, dentro de um limite
  - Atualizações são aplicadas provisoriamente a uma cópia local, à espera de um acordo global de todas as réplicas
  - Em alguns casos, algumas atualizações podem precisar voltar atrás e serem aplicadas em uma ordem diferente antes de se tornarem permanentes

Modelos de consistência **sequencial** e **causal**

- Ampliam os modelos de consistência contínua
  - Replicas devem acordar sobre algum tipo de ordenação global das atualizações
  - Definem diferentes tipos de ordenação consistente de atualizações

Modelos de consistência **sequencial** e **causal**

- **Notação**
  - Operações de um processo representadas ao longo de um eixo de tempo
  - Eixo de tempo sempre representado na horizontal (tempo cresce da esquerda para a direita)
  - **Wi(x)a** → Escrita pelo processo Pi para o item de dados **x** com o valor **a**
  - **Ri(x)a** → Leitura pelo processo Pi do item de dados **x** retornando o valor **b**

Modelos de consistência **sequencial** e **causal**

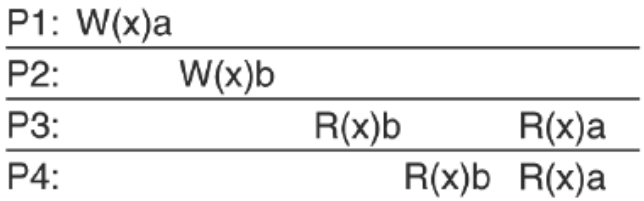
- Exemplo:
  - P_1 executa uma escrita para um item de dados **x**, modificando o seu valor para **a.** Esta operação é feita localmente e depois propagada para os outros processos
  - Mais tarde P_2 lê o valor NIL e, pouco tempo depois, lê **a.** Existe um retardo para propagar a atualização de **x** para P_2

Modelos de consistência **sequencial**

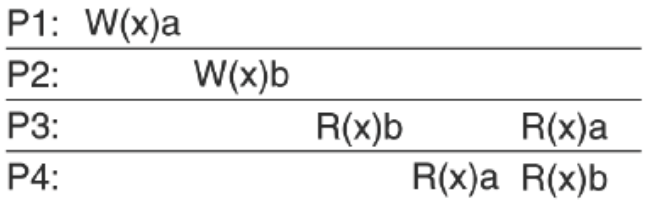
- Definido por Lamport (1979) no contexto de memória compartilhada para sistemas multiprocessadores
  - Quando processos executam concorrentemente em máquinas diferentes, qualquer intercalação válida de operações de leitura e escrita é um comportamento aceitável
  - Todos os processos veem a mesma intercalação de operações

Modelos de consistência **sequencial**

a) P1 executa **W(x)a** para **x**. Mais tarde (tempo absoluto), o processo P2 também executa uma operação de escrita **W(x)b**, ajustando o valor de **x** para **b**. Os processos P3 e P4 primeiro leem o valor **b** e, mais tarde, o valor **a**. A operação de escrita do processo P2 parece ter ocorrido antes da de P1



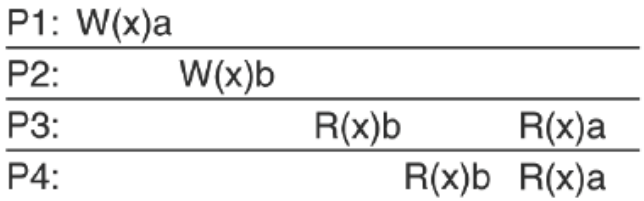
(a)



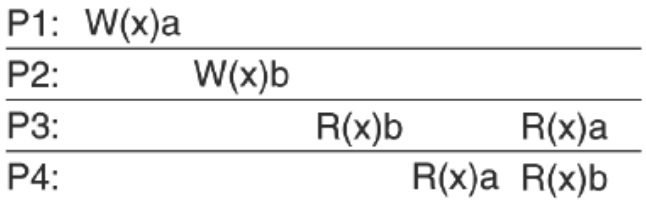
(b)

Modelos de consistência **sequencial**

b) Neste caso é violada a consistência sequencial porque nem todos os processos veem a mesma intercalação de operações de escrita. Para o processo P3 parece que o item de dados foi primeiro alterado para **b,** e mais tarde para **a.** Por outro lado, P4 concluirá que o valor final é **b**



(a)



(b)

Modelos de consistência **sequencial**

- O contrato entre processos e o depósito de dados é que os processos devem aceitar TODOS os resultados válidos como respostas adequadas e devem trabalhar corretamente se qualquer um deles ocorrer. Um programa que funciona somente com um subconjunto de resultados está incorreto

Modelos de consistência **causal**

- Hutto e Ahamad, 1990
- Modelo faz uma distinção entre eventos que são ou não potencialmente relacionados por causalidade
  - Relação *acontece antes*
- Se o evento **b** é causado ou influenciado por um evento anterior **a**, a causalidade requer que todos vejam primeiro **a** e, depois **b**
  - Suponha que o processo P1 escreva um item de dados **x**. Então P2 lê **x** e escreve **y**. A leitura de **x** e a escrita de **y** são potencialmente relacionadas por causalidade porque o cálculo de **y** pode ter dependido do valor de **x** lido por P2, isto é, o valor escrito por P1

Modelos de consistência **causal**

- Consequência:
  - **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 **causal**

- Exemplo:
  - Sequência de eventos permitida quando o depósito é consistente por causalidade (mas incorreta quando o depósito é sequencialmente consistente)
  - Escritas $W_2(x)b$ e $W_1(x)c$ são concorrentes, logo não precisam ser vistas na mesma ordem

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

Modelos de consistência **causal**

a) $W_2(x)b$ é potencialmente dependente de $W_1(x)a$ porque **b** pode ser resultado de um cálculo que envolva o valor lido por $R_2(x)a$. As duas escritas são relacionadas por causalidade, portanto temos uma violação na ordenação das operações

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

(a)

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

(b)

Modelos de consistência **causal**

b) Como a leitura R(x)a foi removida, W_2(x)b e W_1(x)a agora são escritas concorrentes. Um depósito consistente por causalidade não requer que escritas concorrentes sejam ordenadas globalmente

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

(a)

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

(b)

Modelos de consistência **causal**

- Requer monitorar quais processos viram quais escritas
- É preciso construir e manter um gráfico de dependência que mostre qual operação é dependente de quais operações
  - Usar marcas de **tempo vetoriais**

Modelos de consistência **centrados no cliente**

- Modelo de **consistência eventual**
  - Depósitos de dados não sofrem atualizações simultâneas
  - Atualizações simultâneas podem ser resolvidas com facilidade, quando acontecem
  - Aplicações onde a maioria das operações é de leitura
    - Verdade em muitos sistemas de bancos de dados
- Conflitos **escrita-escrita** nunca ocorrerão, mas sim conflitos **leitura-escrita**
- Exemplos:
  - Páginas web atualizadas apenas pelo webmaster
  - Uso de proxies e caches para melhorar eficiência

Modelos de consistência **centrados no cliente**

- Problema
  - Se um mesmo usuário acessa diferentes réplicas de um depósito de consistência eventual, podem haver 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

Relembrando...

- Replicação
  - Vantagens: **confiabilidade** e **desempenho**
  - Problema: **consistência**

- Modelos de consistência
  - Contrato entre **processos** e o "**depósito de dados**" de um SD

Protocolos de consistência

- **Objetivo** → implementar um modelo de consistência específico

- Diferentes tipos de protocolo para os diferentes tipos de modelos de consistência
  - Protocolos de consistência contínua
  - Protocolos baseados em primários (sequencial)
  - Protocolos de escrita replicada (sequencial)
  - Protocolos de coerência de cache (centrado no cliente)

Protocolos de consistência contínua

- Baseados em manter uma métrica local que indica o "tamanho do desvio" entre a réplica local e o que foi percebido nas outras réplicas do sistema
  - Visa garantir que desvio não ultrapasse um **limite máximo**

- Desvio numérico
  - Métrica baseada no "peso" de cada escrita
- Desvio de idade
  - Métrica baseada em estampas de tempo vetoriais
- Desvio de ordenação
  - Uso de uma fila de mensagens de tamanho limitado

Protocolos baseados em primários

- **Objetivo** → implementar modelos de consistência sequencial

- **Ideia geral** → existe uma das réplicas (**primário**) no depósito de dados responsável por coordenar as escritas em cada item de dados **x**

- Duas abordagens:
  1. Escrita remota
  2. Escrita local

Baseados em primários: Escrita remota

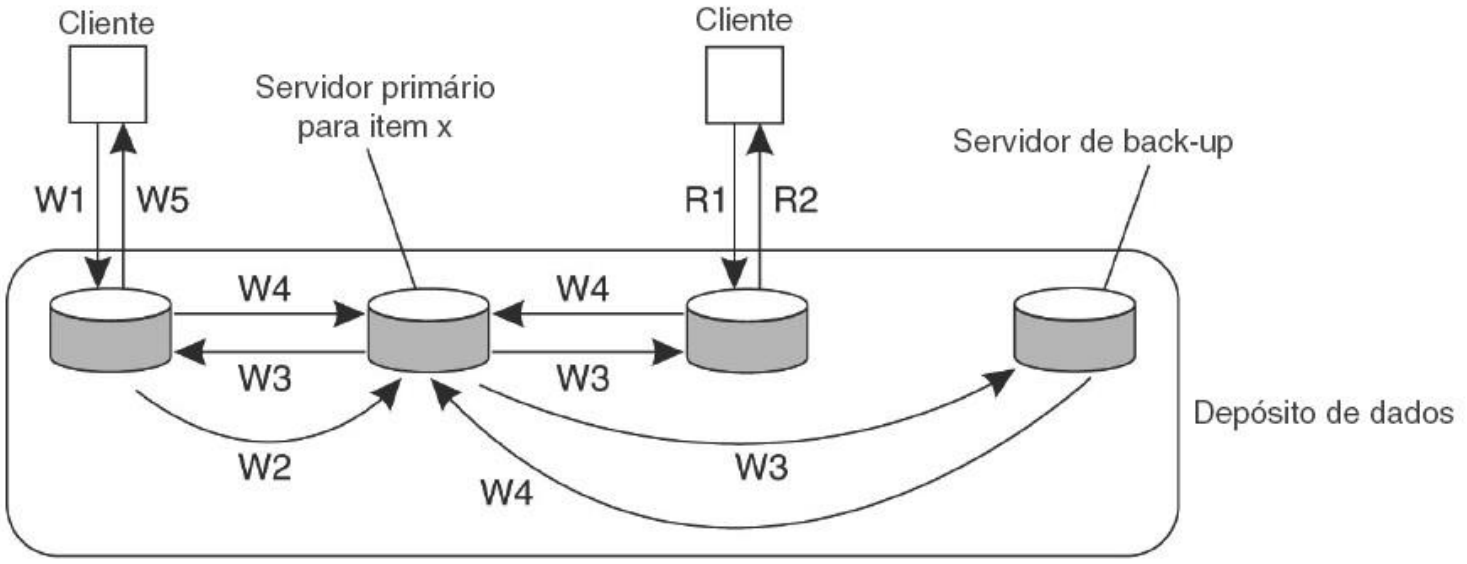


Figura 7.19 Princípio de um protocolo de primário e backup.

Baseados em primários: Escrita remota

- **Problema** → tempo longo até que a escrita seja finalizada
  - Escritas são tarefas bloqueantes
  - Poderia usar escritas não bloqueantes, mas tornaria a solução vulnerável à falhas

- Permitem a implementação direta de consistência sequencial porque o servidor primário pode ordenar todas as escritas em uma ordem temporal globalmente exclusiva

Baseados em primários: Escrita local

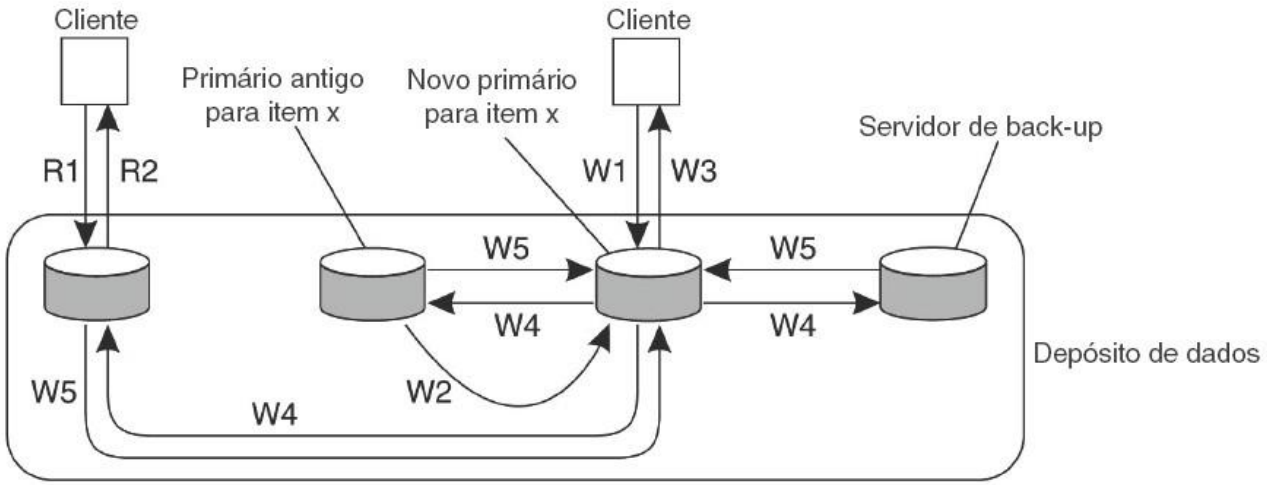


Figura 7.20 Protocolo de primário e backup no qual a cópia primária migra para o processo que quer realizar uma atualização.

Baseados em primários: Escrita local

- **Vantagem** → permite realizar múltiplas escritas em sequência usando a cópia local, quando a escrita é não bloqueante

- Pode ser usado em aplicações móveis
  - Componente móvel torna-se o primário dos itens de dados que utiliza
  - Operações de escrita podem ser realizadas enquanto *offline*
  - Ao reconectar, atualizações locais são enviadas para outras réplicas

Protocolos de escrita replicada

- **Objetivo** → também implementam modelos de consistência sequencial

- **Ideia geral** → operações de escrita são executadas em várias réplicas
  - Diferente do caso de réplicas baseadas em primários

- Dois tipos:
  1. Replicação ativa
  2. Baseados em quórum (voto majoritário)

Escrita replicada: replicação ativa

- Cada réplica possui um processo associado que realiza as operações de atualização
- Operações realizadas localmente devem ser propagadas para as outras réplicas
  - Envio em multicast

- **Problema** → operações devem ser realizadas na mesma ordem em todas as réplicas
  - Multicast ordenado → relógios de Lamport
  - Uso de um coordenador central (sequenciador)

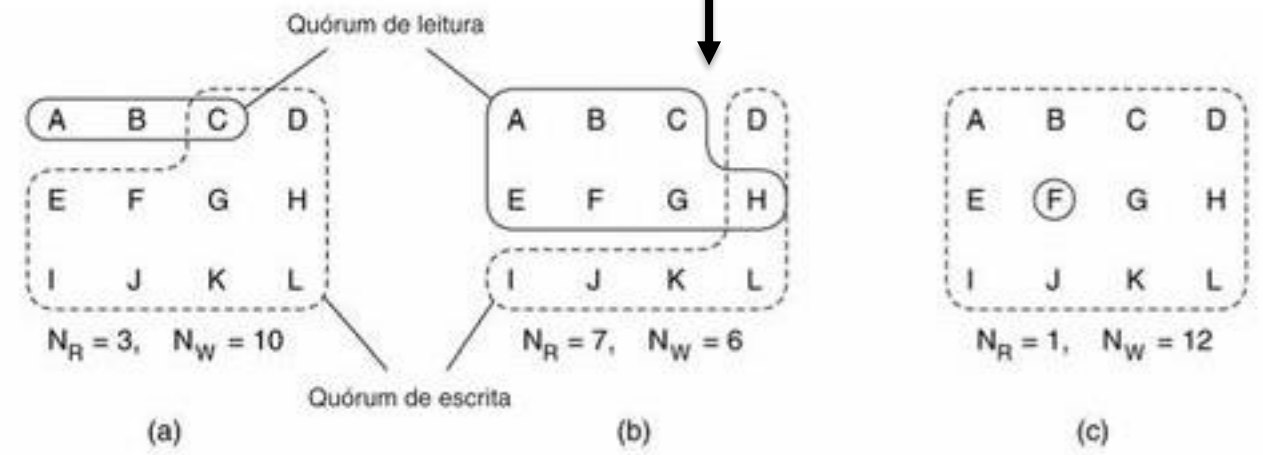
Escrita replicada: quórum

- **Ideia geral** → Clientes devem pedir permissão de vários servidores antes de ler ou escrever um item de dados replicado

- Em um depósito de dados com N réplicas
  - Para escrever um item de dados, cliente deve ter permissão da maioria (N/2 + 1)
  - Para ler um item de dados, o cliente deve ter também a permissão da maioria

Escrita replicada: quórum

- (Gifford, 1979)
- Definição de um quórum de leitura N_R e um quórum de escrita N_W
  - $N_R+N_W > N$
  - $N_W > N/2$
- Exemplos:



Protocolos de coerência de cache

- Caso especial de replicação → réplicas são controladas pelos clientes

- Dois critérios para classificar protocolos de cache
  - Quanto à estratégia de detecção de coerência
    - Detecção estática
    - Detecção dinâmica
  - Quanto à estratégia de imposição de coerência
    - Não permitir dados compartilhados na cache
    - Com dados compartilhados na cache
      - Servidor envia invalidação ou atualização

Protocolos de coerência de cache

- Quando cache é somente de leitura
  - Escritas devem ser realizadas diretamente no depósito de dados
  - Atualizações são propagadas posteriormente para a cache
- Quando cache é leitura/escrita
  - Clientes podem escrever em itens de dados na cache
  - Atualizações são enviadas para os servidores do depósito de dados
  - Caches de escrita **direta** ou **retroativa**
    - Semelhante ao protocolo de escrita local baseados em primários bloqueantes ou não bloqueantes

Gerenciamento de réplicas

- Como posicionar réplicas em um sistema distribuído?

- Posicionar servidores de réplicas
  - Posicionamento de hardware: encontrar as melhores localizações para colocar um servidor que pode hospedar depósito de dados (ou parte dele)
- Posicionar conteúdo
  - Posicionamento dos dados e softwares: encontrar o melhor servidor para colocar conteúdo

Gerenciamento de réplicas: servidores

- Mais uma questão gerencial e comercial do que problema de otimização

- Selecionar as melhores K de N localizações para se instalar servidores de réplicas
  - Uso de heurísticas baseadas na distância (latência, largura de banda) entre clientes e localizações
  - Considerar a Internet como um conjunto de Sistemas Autônomos (AS) e distribuir servidores nos maiores AS primeiro

- Em geral, estes algoritmos apresentam complexidade alta e tempo grande para calcular as localizações → Inaceitável quando há *flash crowds* (multidões instantâneas)

Gerenciamento de réplicas: conteúdo

- Três tipos:
  - **Réplicas permanentes**: conjunto inicial de réplicas que constituem um depósito distribuído
    - Servidores que estão em uma única localização ou espelhamento
  - **Réplicas iniciadas por servidor**: cópias dinâmicas de um depósito de dados para aprimorar desempenho
    - Criadas por iniciativa do (proprietário do) depósito de dados → reduzir carga do servidor
    - Replicação ou migração de arquivos para proximidade de clientes que emitem muitas requisições (*flash crowds*)
  - **Réplicas iniciadas por cliente**: cache na máquina do cliente ou cache em máquina compartilhada por clientes de uma LAN (*proxy*)