ACH 2147 — Desenvolvimento de Sistemas de Informação Distribuídos

Aula 21: Consistência e Replicação (parte 2)

Prof. Renan Alves

Escola de Artes, Ciências e Humanidades — EACH — USP

20/05/2024

Consistência centrada no cliente: usuários móveis

Exemplo

Considere um banco de dados distribuído ao qual você tem acesso por meio do seu notebook. Suponha que seu notebook funcione como um frontend para o banco de dados.

- Na localização *A*, você acessa o banco de dados fazendo leituras e atualizações.
- Na localização *B*, você continua seu trabalho, mas, a menos que você acesse o mesmo servidor acessado na localização *A*, você pode detectar inconsistências:
  - suas atualizações em *A* podem não ter sido propagadas para *B*
  - você pode estar lendo entradas mais recentes do que as de *A*
  - suas atualizações em *B* podem eventualmente conflitar com as de *A*

Consistência centrada no cliente: usuários móveis

Exemplo

Considere um banco de dados distribuído ao qual você tem acesso por meio do seu notebook. Suponha que seu notebook funcione como um frontend para o banco de dados.

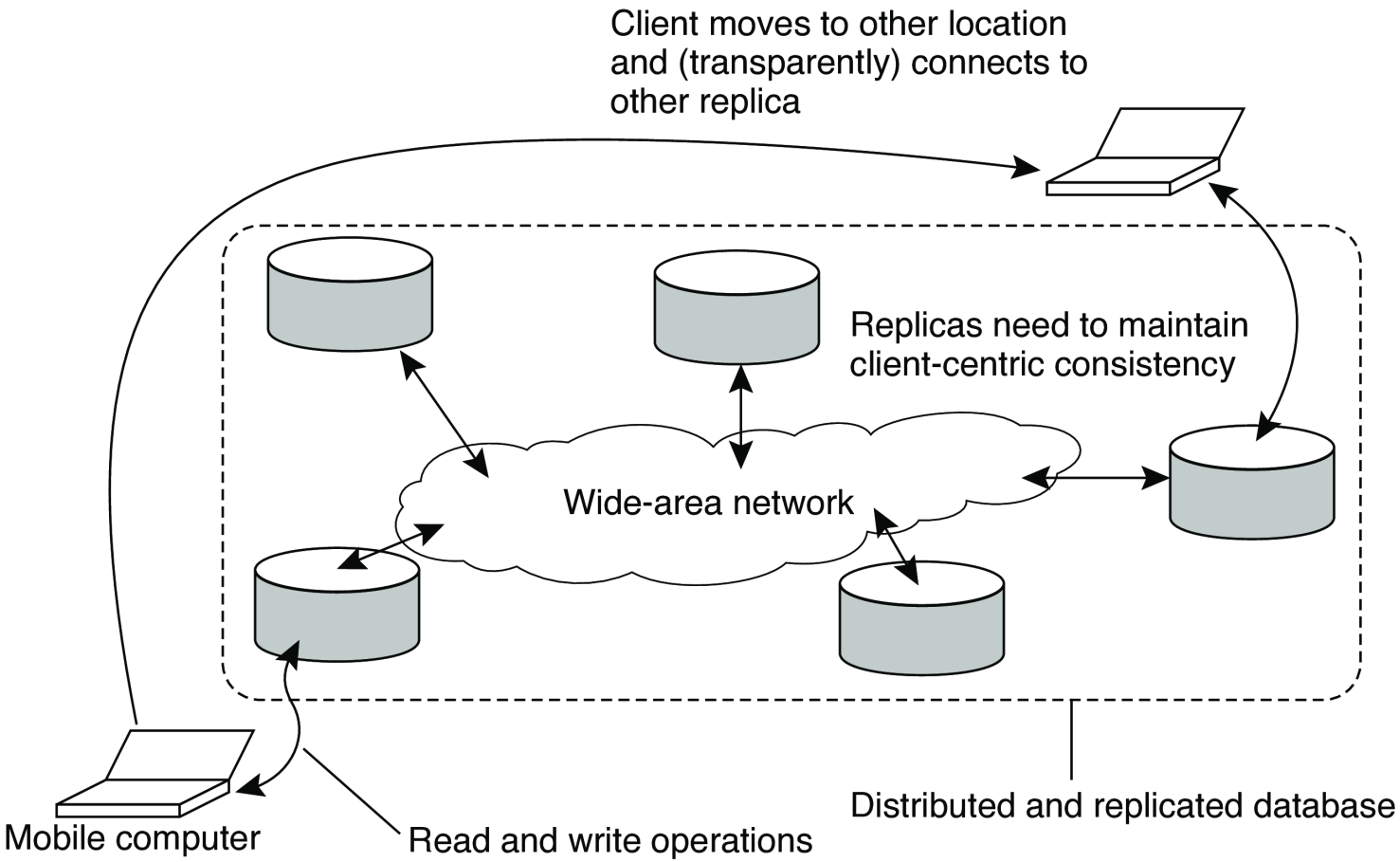
- Na localização *A*, você acessa o banco de dados fazendo leituras e atualizações.
- Na localização *B*, você continua seu trabalho, mas, a menos que você acesse o mesmo servidor acessado na localização *A*, você pode detectar inconsistências:
  - suas atualizações em *A* podem não ter sido propagadas para *B*
  - você pode estar lendo entradas mais recentes do que as de *A*
  - suas atualizações em *B* podem eventualmente conflitar com as de *A*

Observação

A única coisa que você realmente quer é que as entradas que você atualizou e/ou leu em *A*, estejam em *B*. Nesse caso, o banco de dados parecerá ser consistente para você.

Consistência centrada no cliente: arquitetura básica

Usuário móvel acessando diferentes réplicas de um banco de dados distribuído



Consistência centrada no cliente: notação

Notações

- $W_1(x_2)$ é a operação de escrita pelo processo P_1 que leva à versão x_2 de x
- $W_1(x_i; x_j)$ indica que P_1 produz a versão x_j com base em uma versão anterior x_i.
- $W_1(x_i|x_j)$ indica que P_1 produz a versão x_j concorrentemente com a versão x_i.

Leituras monotônicas

Exemplo 1

Leitura automática das atualizações do seu calendário pessoal a partir diferentes servidores. Leituras monotônicas garantem que o usuário veja todas as atualizações, não importa de qual servidor a leitura automática ocorra.

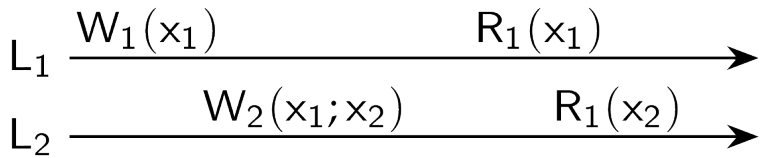
Exemplo 2

Lendo (não modificando) e-mails recebidos enquanto está se deslocando. Cada vez que você se conecta a um servidor de e-mail diferente, esse servidor busca todas as atualizações do servidor que você visitou anteriormente (pelo menos).

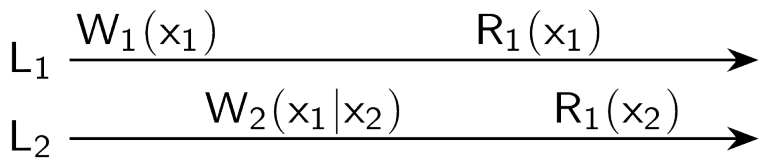
Leituras monotônicas

Definição

Se um processo lê o valor de um item de dados x, qualquer operação de leitura sucessiva em x por esse processo sempre retornará esse mesmo ou um valor mais recente.



Um armazenamento de dados consistente com leituras monotônicas



Um armazenamento de dados que não fornece leituras monotônicas

Escritas monotônicas

Exemplo 1

Atualização de um programa no servidor S, garantindo que todos os componentes em que a compilação e a linkagem dependem, também estejam atualizados em S.

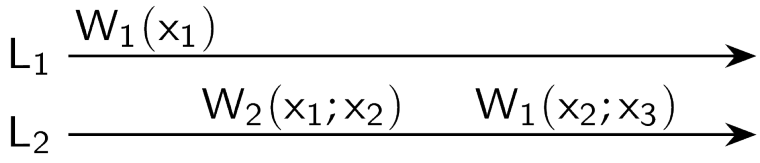
Exemplo 2

Manter versões de arquivos replicados na ordem correta em todos os lugares (propagar a versão anterior para o servidor onde a versão mais recente está instalada).

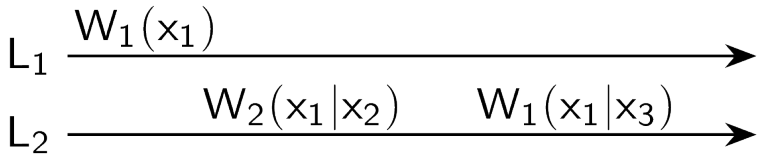
Escritas monotônicas

Definição

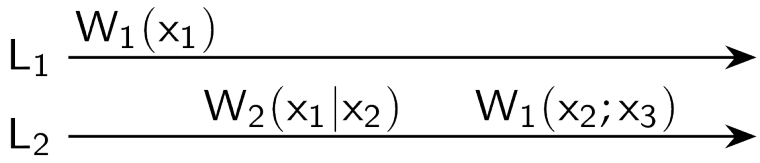
Uma operação de escrita por um processo em um item de dados *x* é concluída antes de qualquer operação de escrita sucessiva em *x* pelo mesmo processo.



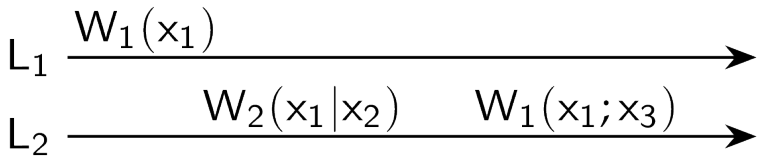
OK



Não OK



Não OK

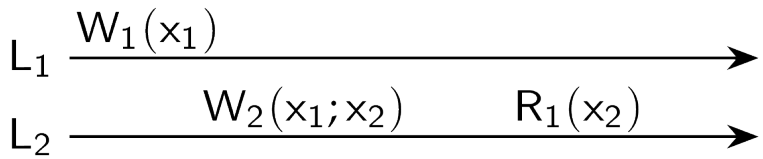


OK

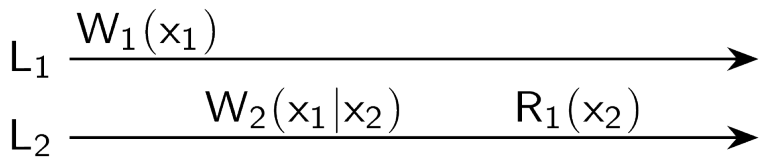
Leia suas escritas

Definição

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



OK

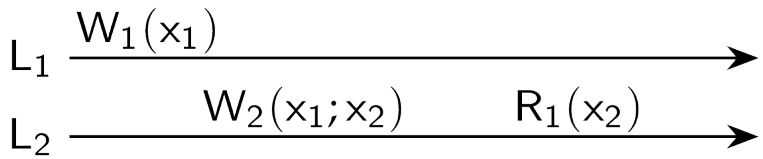


Não OK

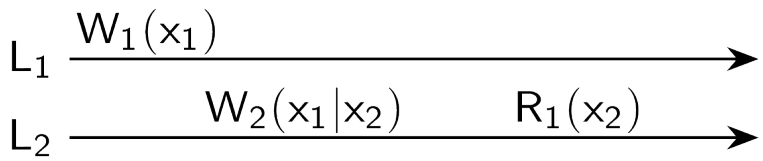
Leia suas escritas

Definição

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



OK



Não OK

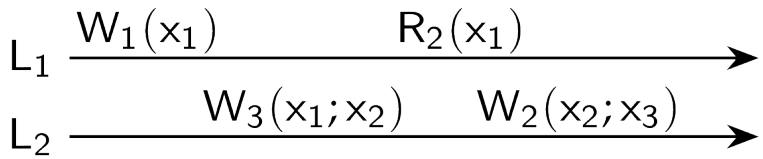
Exemplo

Atualizar sua página da Web e garantir que seu navegador da Web mostre a versão mais recente em vez de sua cópia em cache.

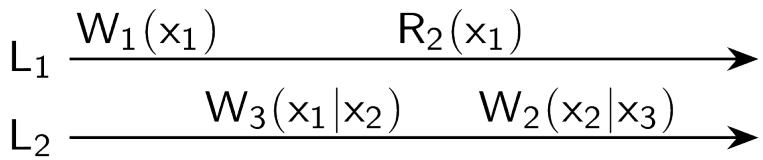
Escritas seguem leituras

Definição

Uma operação de escrita por um processo em um item de dados x após uma operação de leitura anterior em x pelo mesmo processo, é garantido que ocorra no mesmo ou em um valor mais recente de x que foi lido.



OK

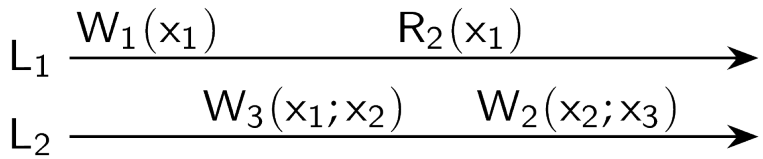


Não OK

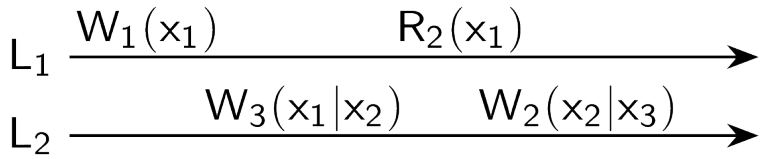
Escritas seguem leituras

Definição

Uma operação de escrita por um processo em um item de dados x após uma operação de leitura anterior em x pelo mesmo processo, é garantido que ocorra no mesmo ou em um valor mais recente de x que foi lido.



OK



Não OK

Exemplo

Ver reações a artigos publicados apenas se você tiver a postagem original (uma leitura “traz” a operação de escrita correspondente).

Gerenciamento de réplicas: posicionamento

Essência

Encontrar quais são os melhores K lugares entre N locais possíveis.

Gerenciamento de réplicas: posicionamento

Essência

Encontrar quais são os melhores K lugares entre N locais possíveis.

- Selecione o melhor local entre $N - K$ para o qual a distância média para os clientes é mínima. Em seguida, escolha o próximo melhor servidor. (Nota: O primeiro local escolhido minimiza a distância média para todos os clientes.) Computacionalmente caro.

Gerenciamento de réplicas: posicionamento

Essência

Encontrar quais são os melhores K lugares entre N locais possíveis.

- Selecione o melhor local entre $N - K$ para o qual a distância média para os clientes é mínima. Em seguida, escolha o próximo melhor servidor. (Nota: O primeiro local escolhido minimiza a distância média para todos os clientes.) Computacionalmente caro.
- Selecione a K-ésima maior organização e coloque um servidor no host mais conectado. Pode ser financeiramente caro.

Gerenciamento de réplicas: posicionamento

Essência

Encontrar quais são os melhores K lugares entre N locais possíveis.

- Selecione o melhor local entre $N - K$ para o qual a distância média para os clientes é mínima. Em seguida, escolha o próximo melhor servidor. (Nota: O primeiro local escolhido minimiza a distância média para todos os clientes.) Computacionalmente caro.
- Selecione a K-ésima maior organização e coloque um servidor no host mais conectado. Pode ser financeiramente caro.
- Posicione nós em um espaço geométrico d-dimensional, onde a distância reflete a latência. Identifique as K regiões com maior densidade e coloque um servidor em cada uma. Computacionalmente barato.

Replicação de conteúdo

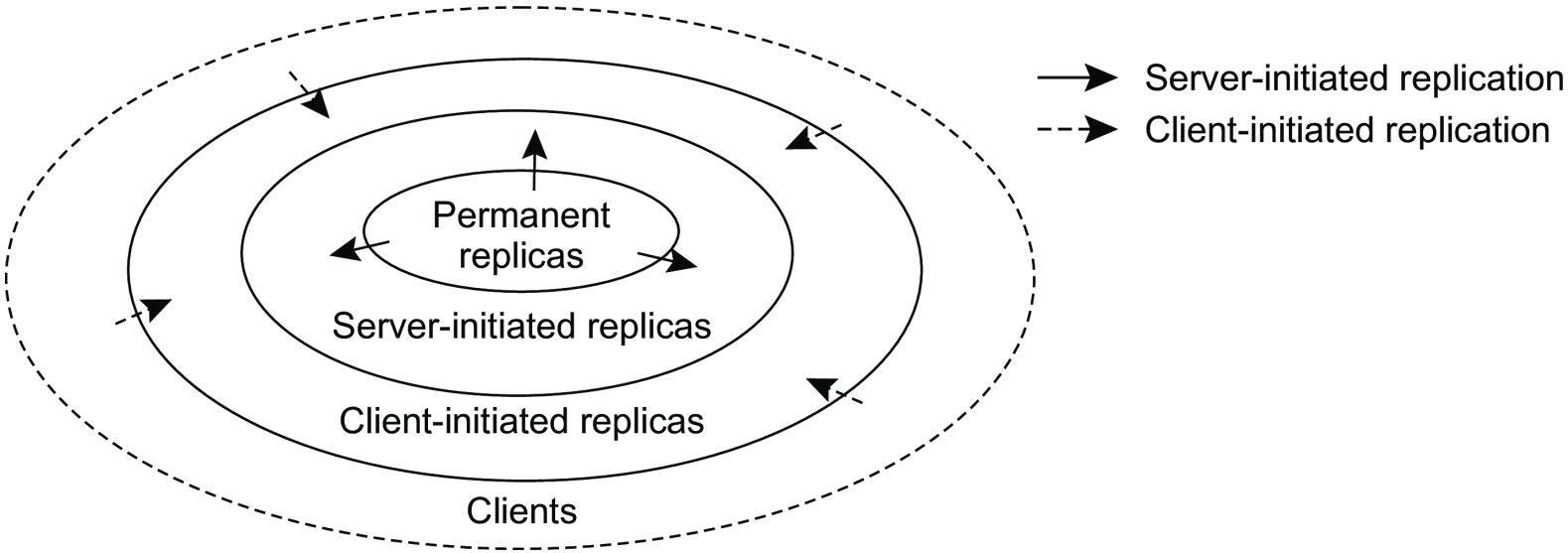
Distinguir diferentes processos

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

- Réplicas permanentes: Processo/máquina sempre tendo uma réplica
- Réplica iniciada pelo servidor: Processo que pode hospedar dinamicamente uma réplica a pedido de outro servidor no armazenamento de dados
- Réplica iniciada pelo cliente: Processo que pode hospedar dinamicamente uma réplica a pedido de um cliente (cache do cliente)

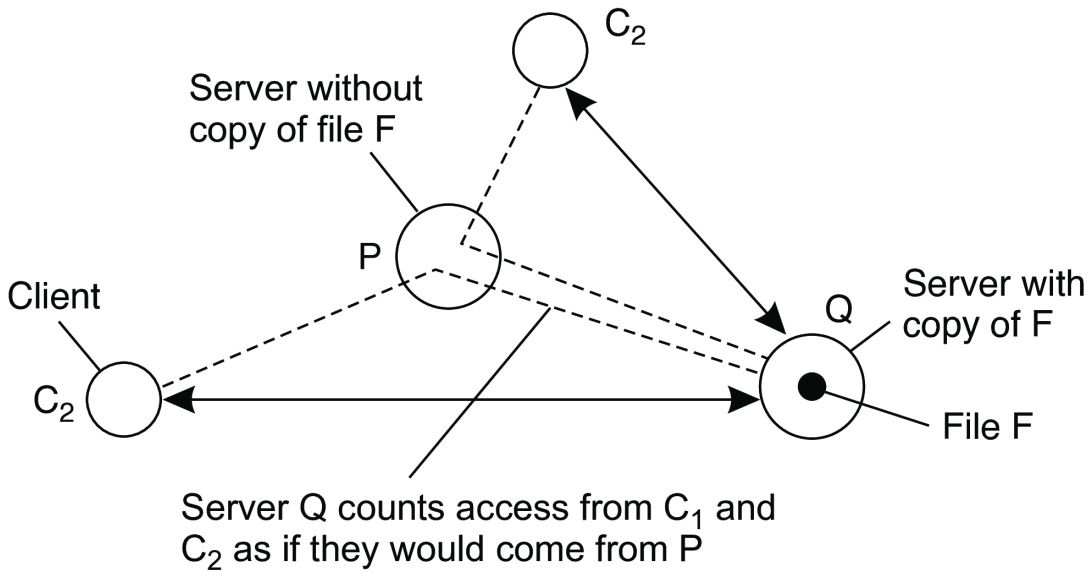
Replicação de conteúdo

A organização lógica de diferentes tipos de cópias de um armazenamento de dados em três anéis concêntricos



Réplicas iniciadas pelo servidor

Contagem de solicitações de acesso de diferentes clientes



- Manter as contagens de acesso por arquivo, agregadas de acordo com o servidor mais próximo dos clientes solicitantes
- Número de acessos cai abaixo do limite $D \Rightarrow$ excluir arquivo
- Número de acessos excede o limite $R \Rightarrow$ replicar arquivo
- Número de acessos entre D e $R \Rightarrow$ migrar arquivo

Distribuição de conteúdo

Considere apenas a combinação de cliente/servidor

- Propagar apenas notificação/invalição da atualização (frequentemente usado para caches)
- Transferir dados de uma cópia para outra (bancos de dados distribuídos): replicação passiva
- Propagar a operação de atualização para outras cópias: replicação ativa

Nota

Não há uma única abordagem melhor, mas depende muito da largura de banda disponível e da relação leitura/escrita nas réplicas.

Distribuição de conteúdo: sistema cliente/servidor

Uma comparação entre protocolos baseados em push e pull no caso de sistemas de vários clientes, único servidor

- Push de atualizações: abordagem iniciada pelo servidor, na qual a atualização é propagada independentemente se o destino a solicitou ou não.
- Pull de atualizações: abordagem iniciada pelo cliente, na qual o cliente solicita ser atualizado.

Questão	Baseado em push	Baseado em pull
1:	Lista de caches do cliente	Nenhum
2:	Atualização (e talvez busca de atualização)	Poll e atualização
3:	Imediato (ou tempo de busca-atualização)	Tempo de busca-atualização

1: Estado no servidor

2: Mensagens a serem trocadas

3: Tempo de resposta no cliente

Distribuição de conteúdo

Observação

Podemos alternar dinamicamente entre push e pull usando leases: Um contrato no qual o servidor promete empurrar (push) atualizações para o cliente até que o lease expire.

Tornando o tempo de expiração do lease adaptativo

Distribuição de conteúdo

Observação

Podemos alternar dinamicamente entre push e pull usando leases: Um contrato no qual o servidor promete empurrar (push) atualizações para o cliente até que o lease expire.

Tornando o tempo de expiração do lease adaptativo

- Lease baseados em idade: Um objeto que não mudou por muito tempo, não mudará no futuro próximo, portanto, forneça um lease de longa duração

Distribuição de conteúdo

Observação

Podemos alternar dinamicamente entre push e pull usando leases: Um contrato no qual o servidor promete empurrar (push) atualizações para o cliente até que o lease expire.

Tornando o tempo de expiração do lease adaptativo

- Lease baseados na frequência de renovação: Quanto mais frequentemente um cliente solicita um objeto específico, mais longo será o tempo de expiração para esse cliente (para esse objeto)

Distribuição de conteúdo

Observação

Podemos alternar dinamicamente entre push e pull usando leases: Um contrato no qual o servidor promete empurrar (push) atualizações para o cliente até que o lease expire.

Tornando o tempo de expiração do lease adaptativo

- Lease baseados em estado: Quanto mais carregado um servidor estiver, mais curtos se tornarão os tempos de expiração

Distribuição de conteúdo

Observação

Podemos alternar dinamicamente entre push e pull usando leases: Um contrato no qual o servidor promete empurrar (push) atualizações para o cliente até que o lease expire.

Tornando o tempo de expiração do lease adaptativo

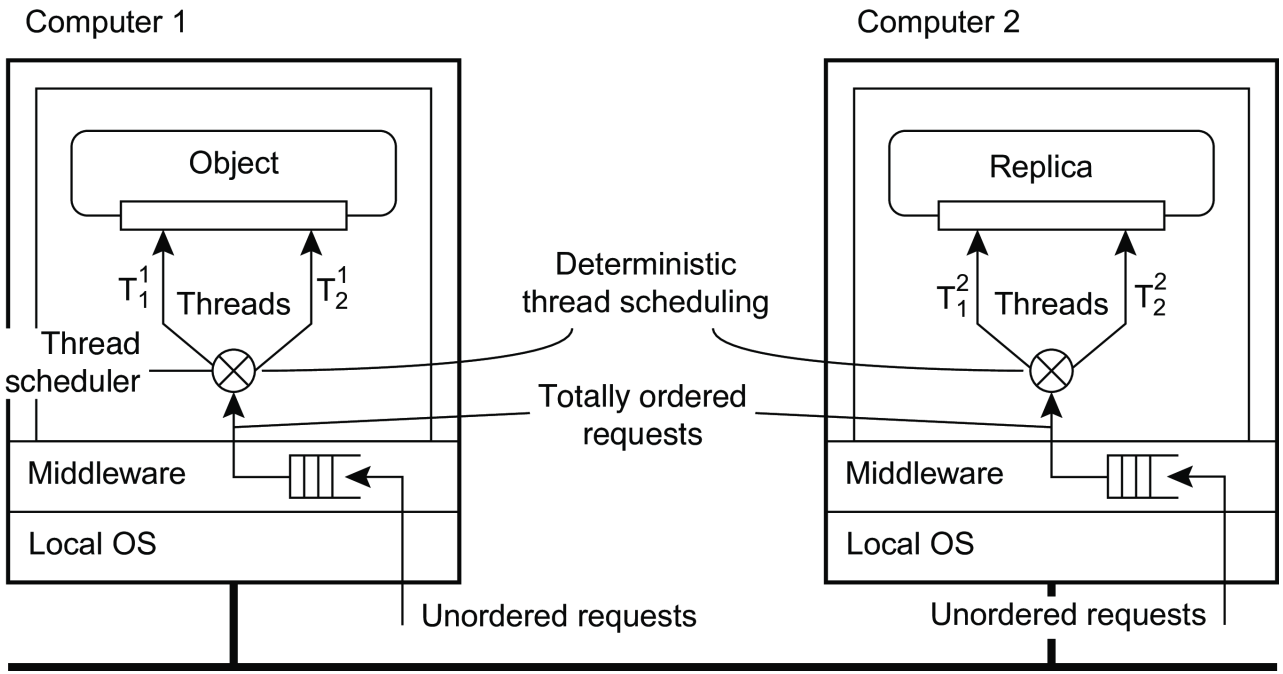
- Lease baseados em idade: Um objeto que não mudou por muito tempo, não mudará no futuro próximo, portanto, forneça um lease de longa duração
- Lease baseados na frequência de renovação: Quanto mais frequentemente um cliente solicita um objeto específico, mais longo será o tempo de expiração para esse cliente (para esse objeto)
- Lease baseados em estado: Quanto mais carregado um servidor estiver, mais curtos se tornarão os tempos de expiração

Questão

Por que ter todo este trabalho?

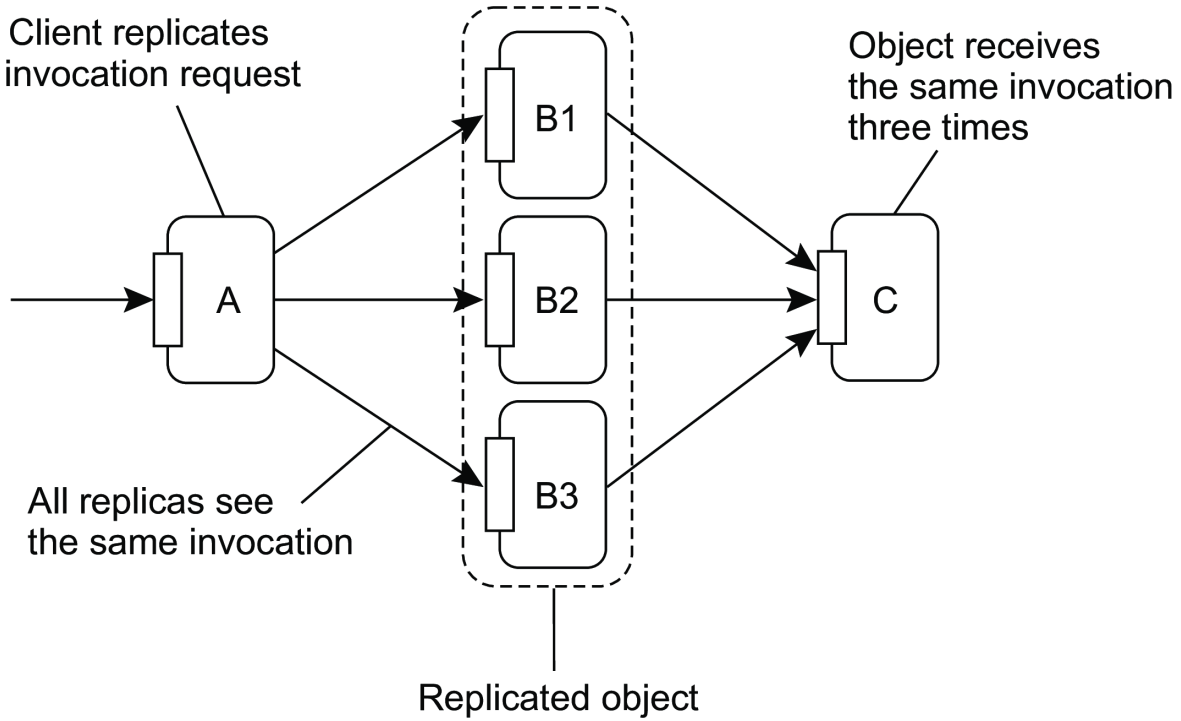
Gerenciando objetos replicados

- Prevenir a execução simultânea de múltiplas invocações no mesmo objeto: o acesso aos dados internos de um objeto deve ser serializado. O uso de mecanismos de bloqueio local é suficiente.
- Garantir que todas as alterações no estado replicado do objeto sejam as mesmas: duas invocações de método independentes em réplicas diferentes não devem ocorrer ao mesmo tempo: precisamos de escalonamento de thread determinístico.

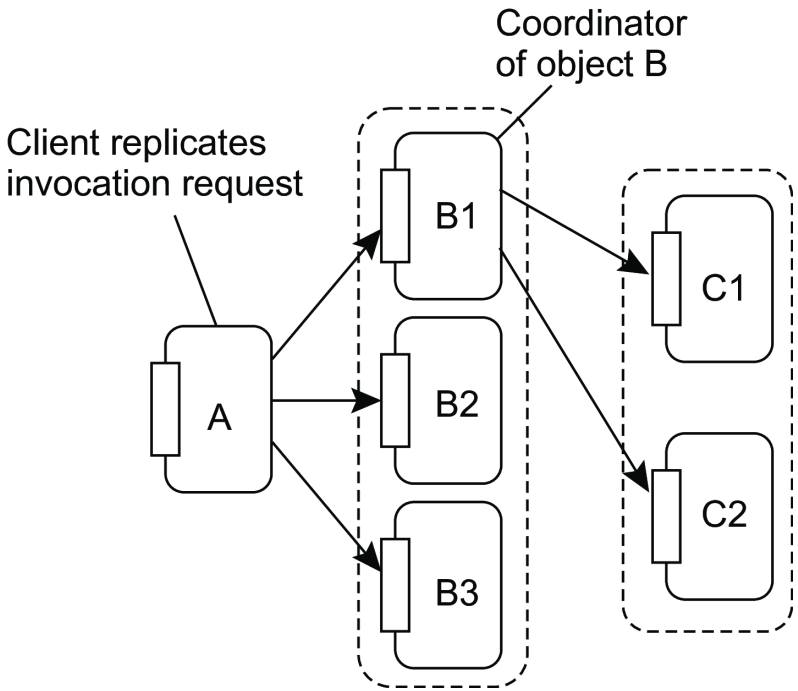


Invocações de objeto replicado

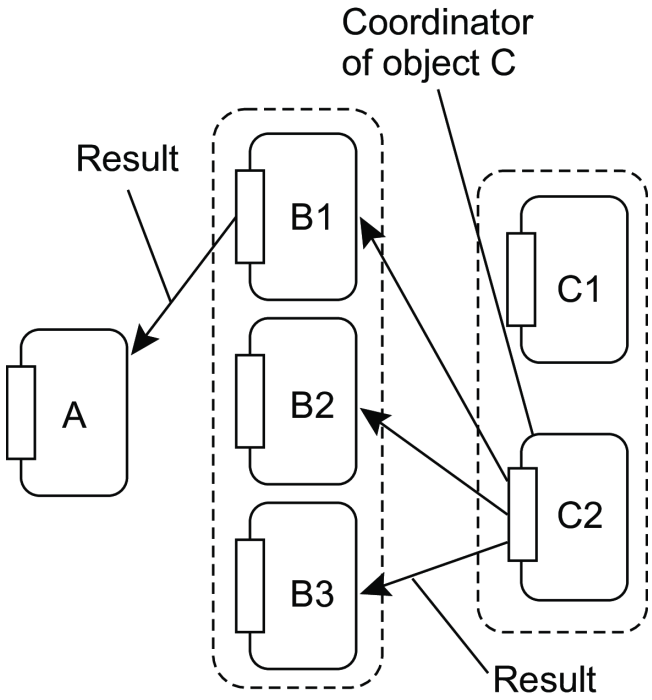
Problema ao invocar um objeto replicado



Invocações de objeto replicado



Encaminhando uma requisição



Retornando a resposta