ACH 2147 — DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO DISTRIBUÍDOS

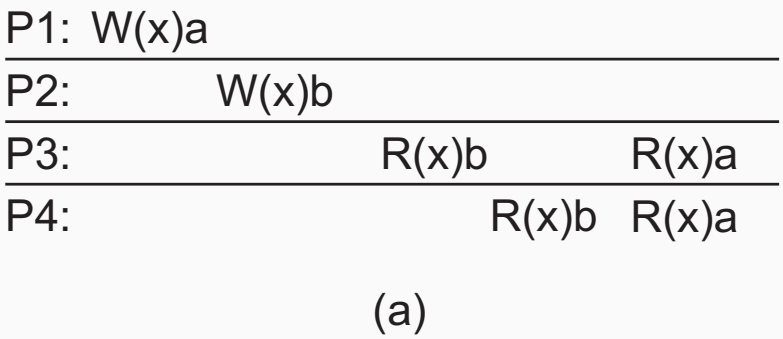
CONSISTÊNCIA E REPLICAÇÃO

Daniel Cordeiro

10 e 17 de junho de 2019

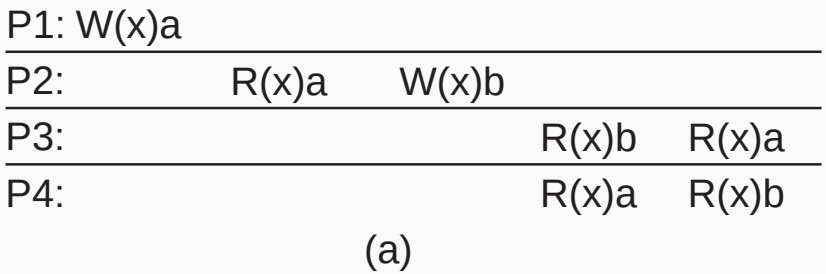
Escola de Artes, Ciências e Humanidades | EACH | USP

Definição
O resultado de qualquer execução é o mesmo, como se as operações de todos os processos fossem executadas na mesma ordem sequencial e as operações de cada processo aparecer nessa sequência na ordem especificada pelo seu programa.

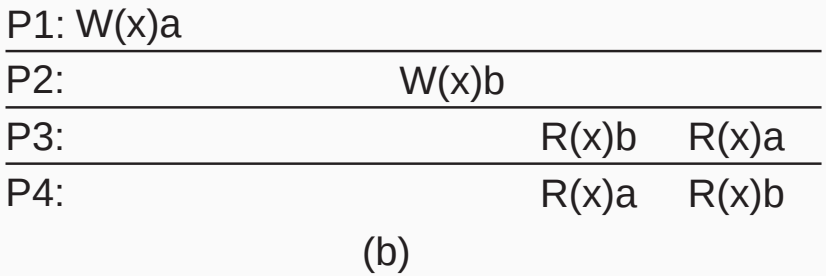


(a) é um *data store* com consistência sequencial; (b) não apresenta consistência sequencial

Definição
Operações de escrita que potencialmente possuem uma relação de causalidade devem ser vistas por todos os processos na mesma ordem. Escritas concorrentes podem ser vistas em uma ordem diferente por processos diferentes.



(a)



(b)

(a) uma violação da consistência causal; (b) uma sequência correta de eventos em um *data store* com consistência causal

Definição

- acessos às variáveis de sincronização (*locks*) possuem consistência sequencial
- o acesso às variáveis de sincronização não é permitido até que todas as escritas anteriores tenham terminado em todos os lugares
- nenhum acesso aos dados é permitido até que todos os acessos às variáveis de sincronização tenham sido feitos

Ideia básica:
Você não precisa se preocupar se as leituras e escritas de uma **série** de operações serão imediatamente do conhecimento de todos os processos. Você só quer que o **efeito** dessa série seja conhecido.

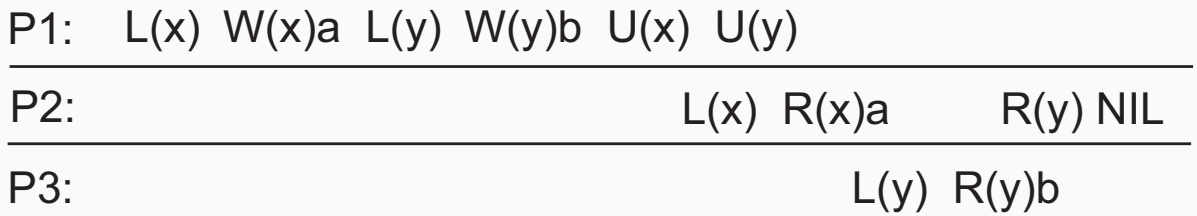


Figura: Um sequência de eventos que respeita a consistência de entrada.

Observação
Consistência de entrada implica a necessidade de proteger os dados com *locks* (implícitos ou não)

- Modelo do sistema
- Leituras monotônicas
- Escritas monotônicas
- *Read-your-writes* (leia-suas-escritas)
- *Write-follows-reads* (escrita-segue-leituras)

Objetivo
Mostrar que talvez manter a consistência em todo o sistema seja desnecessário se nos concentramos no que os **clientes** precisam, ao invés daquilo que deve ser mantido pelos servidores.

Exemplo

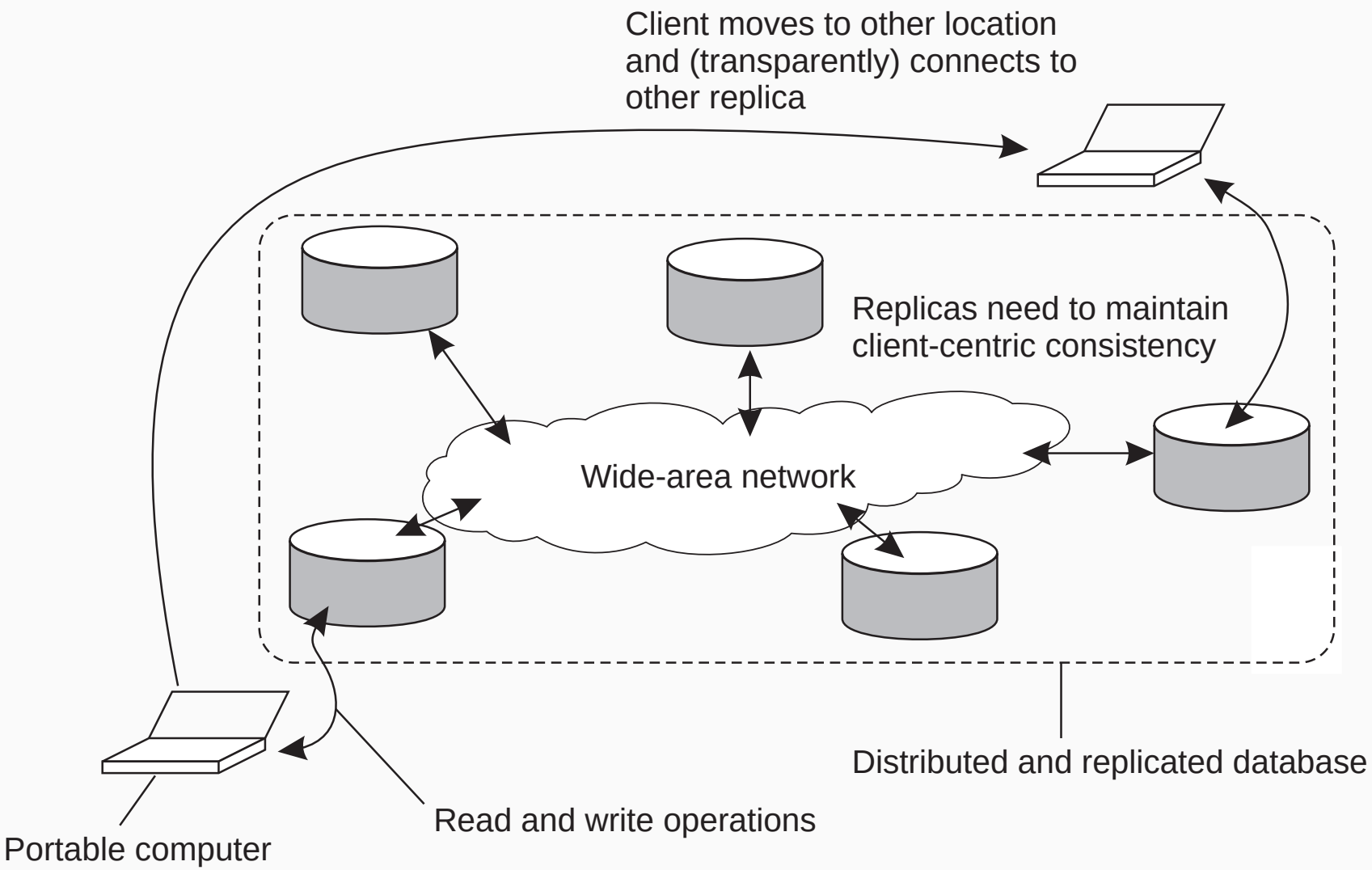
Considere um sistema de banco de dados distribuídos no qual você tem acesso pelo seu notebook. Assuma que seu notebook seja o *front end* do seu banco de dados.

- no local *A* você acessa o banco de dados e realiza leituras e atualizações
- no local *B* você continua seu trabalho, mas, a não ser que você continue acessando o mesmo servidor de antes, você poderá detectar algumas inconsistências:
  - suas atualizações em *A* podem ainda não terem sido propagadas para *B*
  - você pode estar lendo entradas mais novas do que aquelas disponíveis em *A*
  - suas atualizações em *B* podem eventualmente conflitar com àquelas em *A*

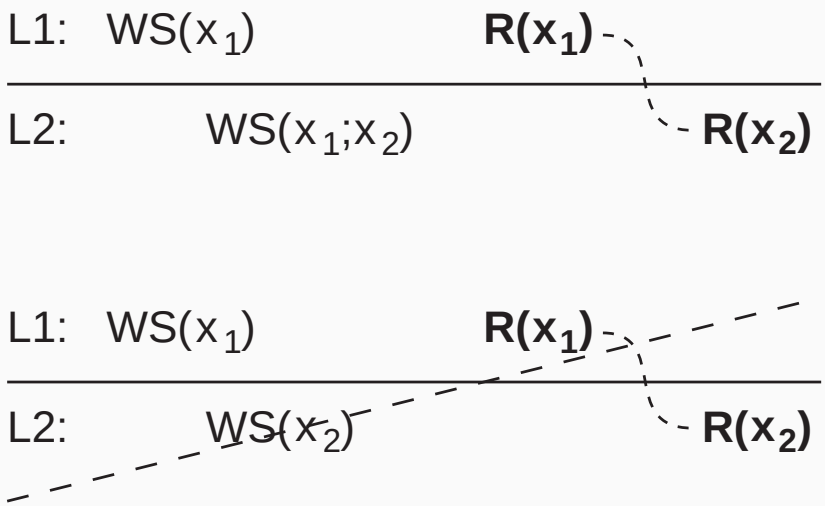
Observação

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

ARQUITETURA BÁSICA



Definição
Se um processo ler o valor de um item *x*, quaisquer leituras sucessivas de *x* feitas por esse processo sempre devolverão o mesmo valor ou um valor mais recente.



Leituras realizadas por um processo *P* em duas cópias locais diferentes do mesmo *data store*. (a) Uma leitura monotônica consistente; (b) um *data store* que não provê leituras monotônicas

Notação

- $W_1(x_2)$ é a operação de escrita feita pelo processo P_1 que leva à versão x_2 de x
- $W_1(x_i; x_j)$ indica que P_1 produziu a versão x_j baseado na versão anterior x_i
- $W_1(x_i|x_j)$ indica que P_1 produziu a versão x_j concorrentemente a versão x_i

Exemplo

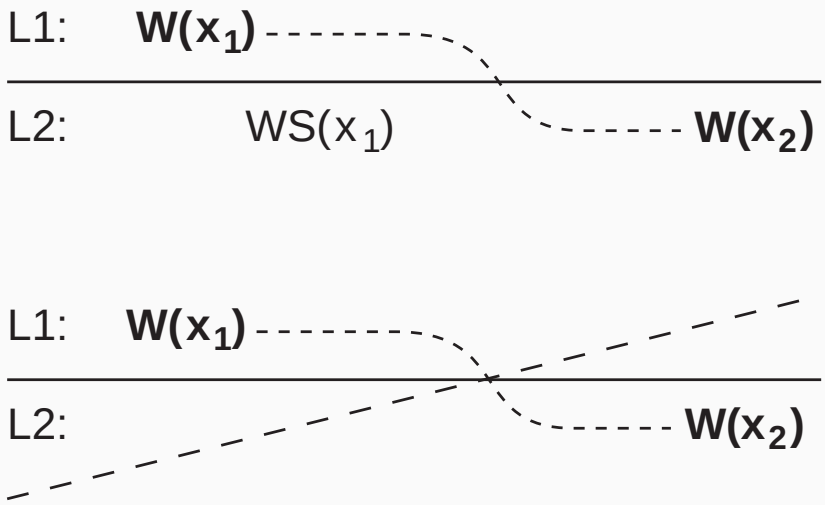
Leituras automáticas das atualizações em seu calendário pessoal vindas de diferentes servidores. Leituras monotônicas garantem que o usuário veja todas as atualizações, independentemente do servidor que originou a leitura

LEITURAS MONOTÔNICAS

Exemplo
Leituras automáticas das atualizações em seu calendário pessoal vindas de diferentes servidores. Leituras monotônicas garantem que o usuário veja todas as atualizações, independentemente do servidor que originou a leitura

Exemplo
Ler (sem modificar) as mensagens enquanto você estiver em movimento. Toda vez que você se conectar a um servidor de e-mails diferente, o servidor irá descarregar (pelo menos) todas as atualizações do servidor que você visitou antes

Definição
Uma escrita monotônica feita por um processo em um dado x é terminada antes de quaisquer operações de escrita sucessivas em x por esse mesmo processo.



Ou seja, se tivermos duas escritas sucessivas $W_k(x_i)$ e $W_k(x_j)$, então não importa onde $W_k(x_j)$ acontece, sempre teremos $W_k(x_i; x_j)$.

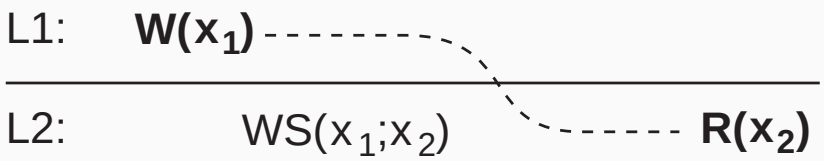
Exemplo

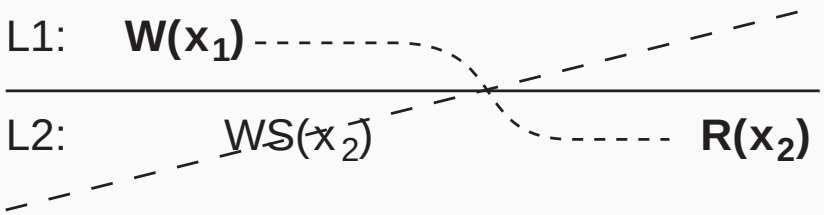
Atualizar um programa no servidor S_2 e garantir que todos os componentes necessários para a compilação também estejam em S_2

Exemplo
Atualizar um programa no servidor S_2 e garantir que todos os componentes necessários para a compilação também estejam em S_2

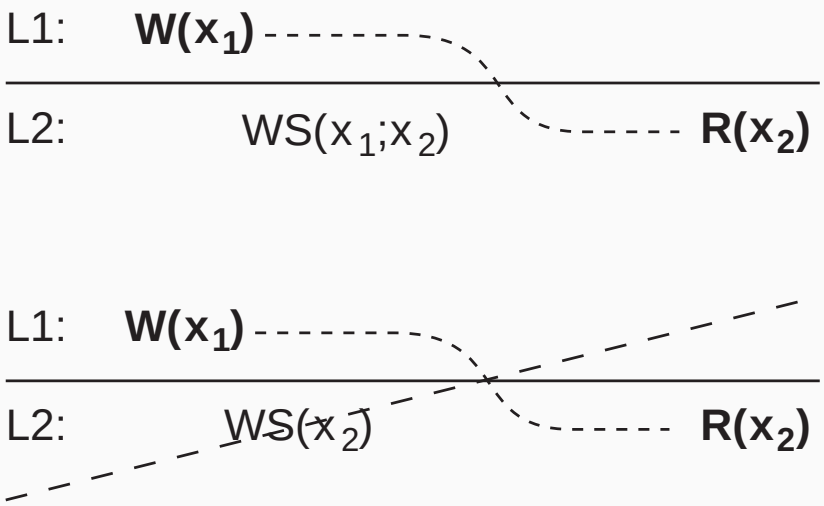
Exemplo
Manter versões de arquivos replicados na ordem correta em todos os lugares (propagando as versões antigas para o servidor onde a versão mais nova está instalada)

Definição
O efeito de uma operação de escrita realizada por um processo no item *x* sempre será visto por operações de leituras de *x* pelo mesmo processo.



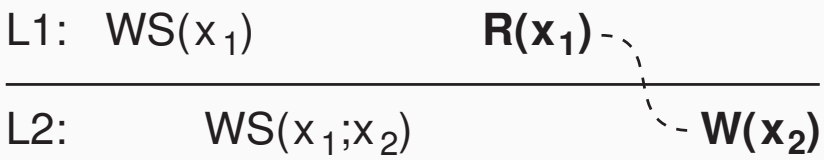


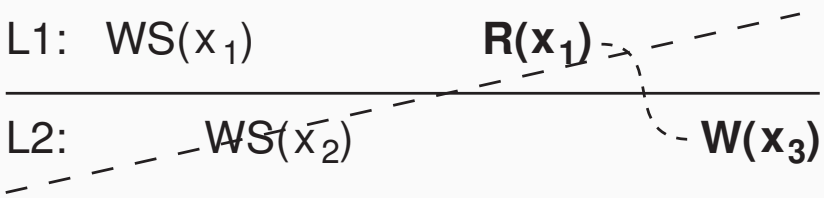
Definição
O efeito de uma operação de escrita realizada por um processo no item *x* sempre será visto por operações de leituras de *x* pelo mesmo processo.



Exemplo
Atualizar sua página web e garantir que o navegador web mostre a versão mais nova ao invés de mostrar a versão em cache

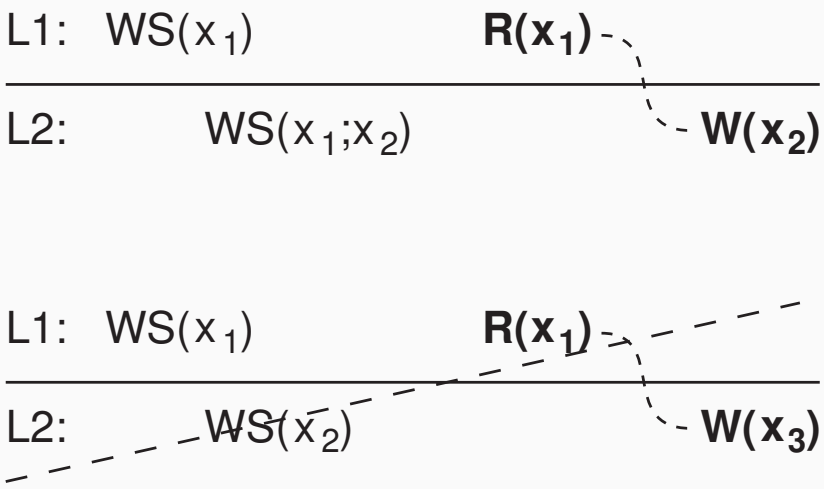
Definição
Uma operação de escrita feita por um processo no item *x* após uma operação de leitura de *x* no mesmo processo é garantidamente realizada no mesmo valor de *x* que foi lido (ou num valor mais novo)





WRITE-FOLLOWS-READ

Definição
Uma operação de escrita feita por um processo no item *x* após uma operação de leitura de *x* no mesmo processo é garantidamente realizada no mesmo valor de *x* que foi lido (ou num valor mais novo)



Exemplo
Ver os comentários a um artigo publicado apenas se você tiver o artigo original (uma leitura "puxa" as operações de escrita correspondentes)

Gerenciamento de réplicas

- posicionamento de servidores de réplicas
- replicação de conteúdo e posicionamento
- distribuição de conteúdo

Ideia
Encontrar as K melhores posições de uma lista de N possibilidades

- iterativamente selecionar as melhores posições de $N - K$ para as quais a distância média até os clientes é mínima e então escolher o próximo melhor servidor (a primeira posição escolhida é a que minimiza a distância média até todos os clientes). **Computacionalmente caro**

Ideia
Encontrar as K melhores posições de uma lista de N possibilidades

- iterativamente selecionar as melhores posições de $N - K$ para as quais a distância média até os clientes é mínima e então escolher o próximo melhor servidor (a primeira posição escolhida é a que minimiza a distância média até todos os clientes). **Computacionalmente caro**
- selecionar o K-ésimo maior **sistema autônomo** e colocar um servidor no host "melhor conectado". **Computacionalmente caro**

Ideia
Encontrar as K melhores posições de uma lista de N possibilidades

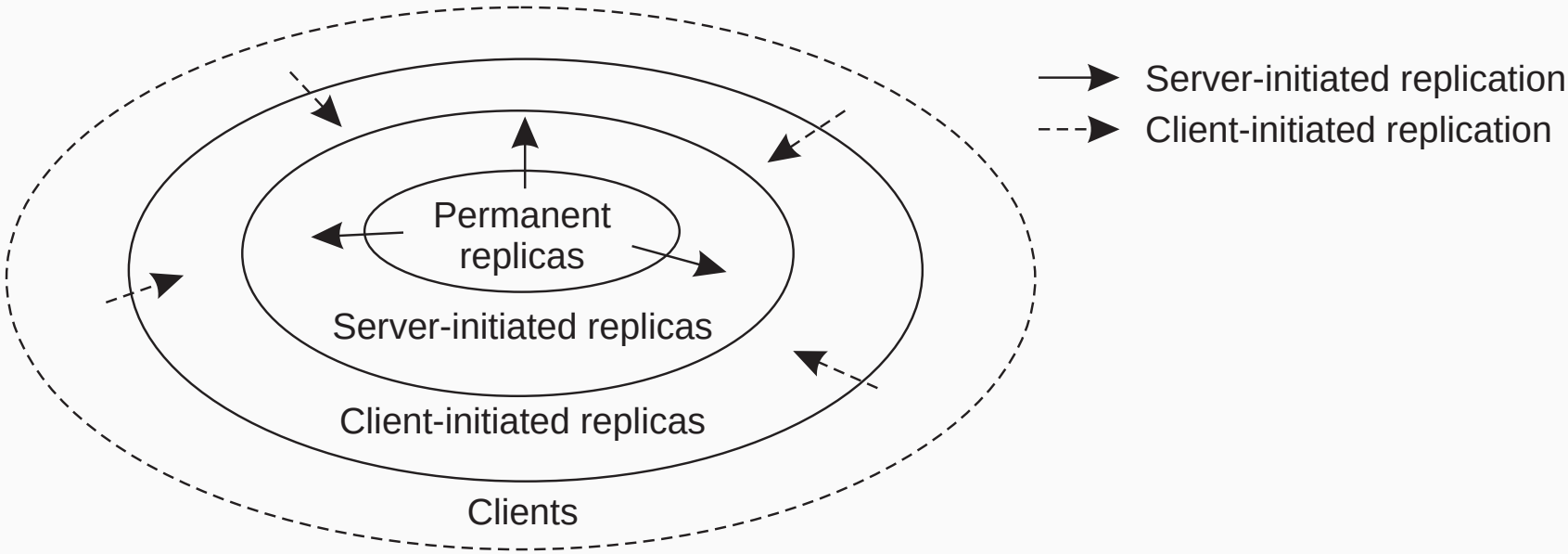
- iterativamente selecionar as melhores posições de $N - K$ para as quais a distância média até os clientes é mínima e então escolher o próximo melhor servidor (a primeira posição escolhida é a que minimiza a distância média até todos os clientes). **Computacionalmente caro**
- selecionar o K-ésimo maior **sistema autônomo** e colocar um servidor no host "melhor conectado". **Computacionalmente caro**
- posicionar os nós em um espaço geométrico d-dimensional, onde a distância reflete a latência. Identificar as K regiões mais densas e colocar um servidor em cada uma delas. **Computacionalmente barato**

Distingue diferentes processos

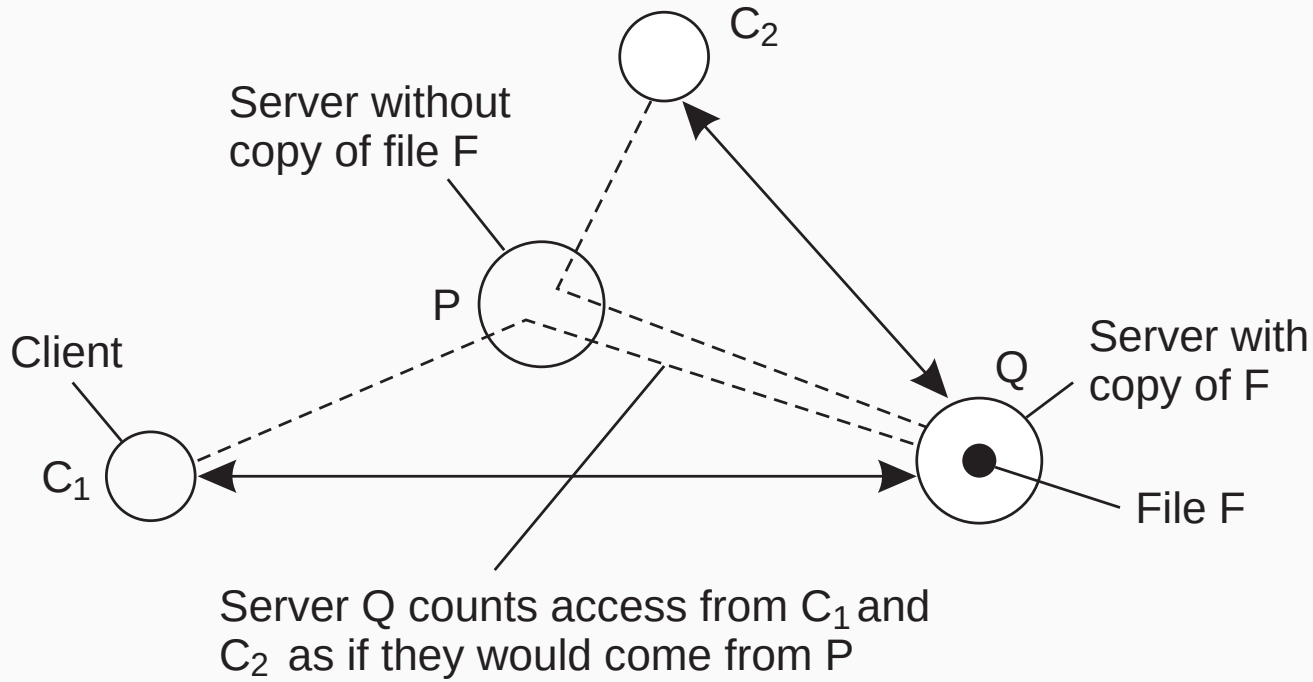
Um processo é capaz de hospedar uma réplica de um objeto ou dado:

- **réplicas permanentes:** processo/máquina sempre tem uma réplica
- **réplica iniciada pelo servidor:** processos que podem hospedar uma réplica dinamicamente, sob demanda de um outro servidor ou *data store*
- **réplica iniciada pelo cliente:** processos que podem hospedar uma réplica dinamicamente, sob demanda de um cliente (cache do cliente)

REPLICAÇÃO DE CONTEÚDO



RÉPLICAS INICIADAS PELO SERVIDOR



- mantenha o número de acessos aos arquivos, agregando-os pelo servidor mais próximo aos clientes que requisitarem o arquivo
- número de acessos cai abaixo de um threshold $D \Rightarrow$ descartar arquivo
- número de acessos acima de um threshold $R \Rightarrow$ replicar arquivo
- número de acessos entre D e $R \Rightarrow$ migrar arquivo