- Consistência e Replicação

  - Razões para a replicação

  - Replicação para obter escalabilidade

  - Modelos de consistência centrados nos dados

Razões para a replicação

- **Fiabilidade**

  - Enquanto pelo menos um servidor continuar a funcionar, o serviço continua.

  - Se os dados de um servidor são corrompidos, a probabilidade de uma réplica também ser corrompida é baixa.

- **Performance**

  - A replicação é importante, quando queremos que o sistema seja escalável em termos de número de acessos.

  - Quando queremos ampliar o sistema em termos de área geográfica. Podemos colocar uma cópia perto de cada grande área de acesso. (Diminui o tempo de acesso ao serviço)

Replicação como técnica para obter escalabilidade

Exemplo:

Quando acedemos a páginas web, o browser armazena localmente uma cópia das paginas acedidas previamente.

O utilizador tem acesso imediato à página.

Problemas?

Se a página foi entretanto modificada, a cópia fica desactualizada.

Podemos deixar que o servidor invalide cópias, cujos originais entretanto foram actualizados. Custo??

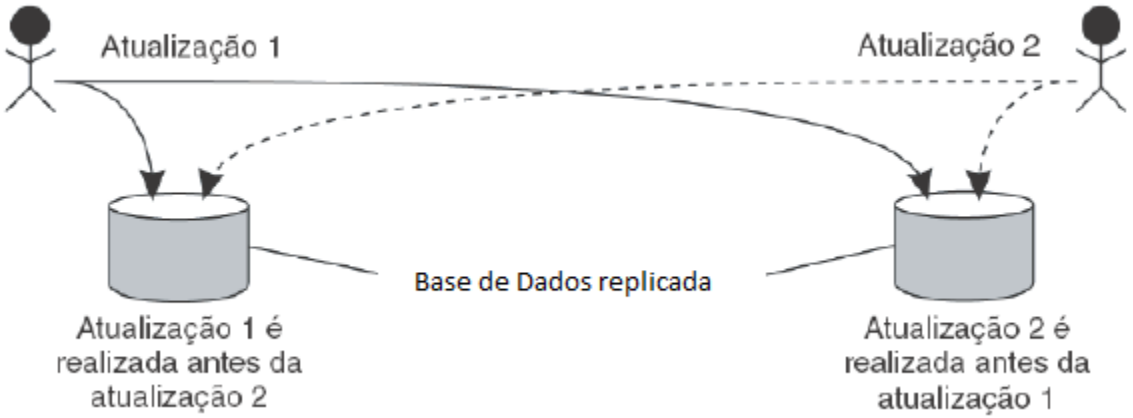
Problema da consistência

Como manter a consistência dos dados entre as várias réplicas?

Suponhamos uma base de dados, replicada em duas cidades, em que:

Actualização1 – um cliente adiciona à sua conta 100€

Actualização 2 - a conta vence juros e é incrementada em 1%



Problema da consistência

Como manter a consistência dos dados entre as várias réplicas?

- Todas as réplicas precisam de chegar a um acordo sobre quando uma actualização deve ser realizada.
  - ordem total usando “Lamport timestamps”?
  - usar um processo coordenador para atribuir a ordem?
  - ??
- Realizar uma actualização dos dados distribuídos, como se fosse uma operação atómica, pode ser tão ou mais custoso em termos de performance como o ganho obtido com a replicação!!!

Modelos de Consistência Centrados nos Dados

A consistência é discutida em termos de operações de leitura e escrita em dados partilhados

Esses dados podem ter a forma de,

- memória partilhada distribuída;
- bases de dados distribuídas;
- sistemas de ficheiros distribuídos

Genericamente designados por **Depósitos de Dados** (*data store*)

Modelos de Consistência Centrados nos Dados

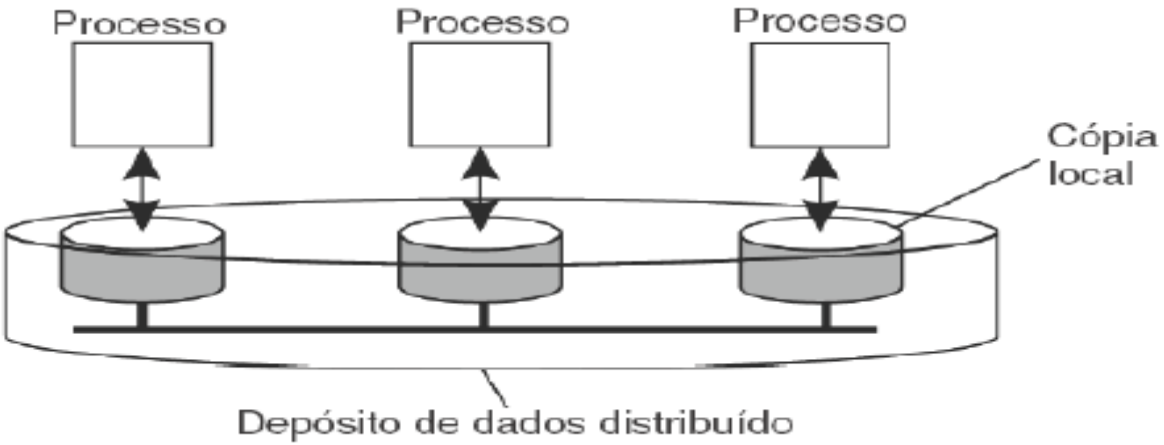
Um depósito de dados,

- Pode estar distribuído por várias máquinas
- Cada processo que pode aceder aos dados tem uma cópia local dos dados
- Operações possíveis:

  write – se altera os dados; read – caso contrário

- Operações de escrita são propagadas para as outras cópias

Modelos de Consistência Centrados nos Dados



Um modelo de consistência é um contracto entre os processos e o depósito de dados.

Se os processos concordam em obedecer a certas regras, o depósito de dados comportar-se-á como esperado

- **Geralmente um processo que faz uma operação de leitura espera obter o valor correspondente à última actualização.**

Modelos de Consistência Centrados nos Dados

Compromisso entre sincronização e performance.

Perder alguma consistência para obter melhor desempenho.

Para obter soluções eficientes é necessário relaxar o conceito de consistência

As inconsistências que um modelo pode aceitar vão depender de cada aplicação.

Consistência Contínua

Consideram-se três eixos independentes para definir inconsistências [Yu and Vahdat 2002]:

- Desvios de valores numéricos entre réplicas
- Desvios na data de actualização
- Desvios em relação à ordem de actualização

=> È definido um valor para o desvio, e considera-se que nesse intervalo existe uma **consistência contínua.**

Consistência Contínua

Exemplos:

- O valor de determinadas acções na bolsa não variar mais de 1% entre duas cópias do depósito de dados.

- Pode considerar-se que para certas aplicações os valores permanecem válidos, durante um certo intervalo de tempo. Uma previsão meteorológica pode ser actualizada para as cópias apenas ao fim de cada dia.

Consistência na Ordem das Operações

- **Consistência Sequencial**
- **Consistência Causal**

Notação:

Wi(x)a → Escrita pelo processo Pi para o item de dados *x do valor a*

Ri(x)b → Leitura pelo processo Pi do item de dados x retornando o valor b

- Quando um processo modifica um item de dados, essa alteração primeiro é feita localmente e depois propagada para as réplicas.

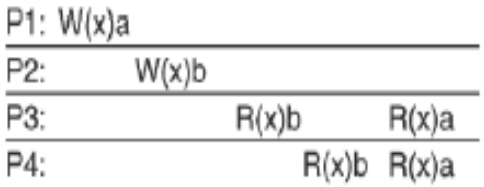
Consistência na Ordem das Operações

- **Consistência Sequencial**

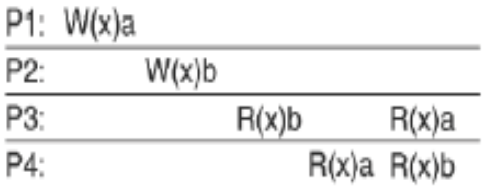
Quando os processos executam concorrentemente em maquinas possivelmente diferentes, qualquer intercalação válida de operações de leitura e escrita é um comportamento aceitável, mas todos os processos devem ver a mesma sequência de operações

Consistência na Ordem das Operações

- **Consistência Sequencial**



(a)



(b)

a) Depósito de dados **com** consistência sequencial

b) Depósito de dados **sem** consistência sequencial

Consistência na Ordem das Operações

- **Consistência Causal**

Se o evento *b é causado ou influenciado por um evento* anterior *a*,

a consistência causal requer que todos vejam primeiro *a e, depois b*

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

Consistência na Ordem das Operações

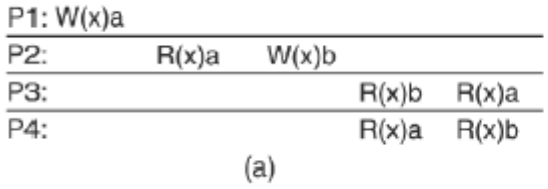
- **Consistência Causal**

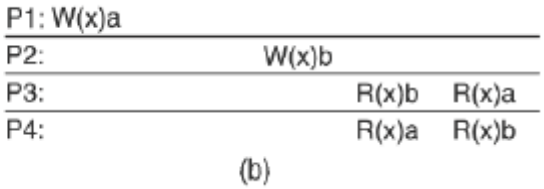
P1:	W(x)a			W(x)c	
P2:		R(x)a	W(x)b		
P3:		R(x)a		R(x)c	R(x)b
P4:		R(x)a		R(x)b	R(x)c

- A sequência respeita a consistência causal, mas não a consistência sequencial.

Consistência na Ordem das Operações

- **Consistência Causal**





a) Violação da consistência causal

b) Sequência correcta num depósito de dados com consistência causal