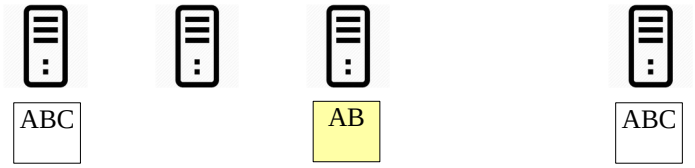
7. Consistência e Replicação



Réplicas e problemas de consistência: Teorema CAP

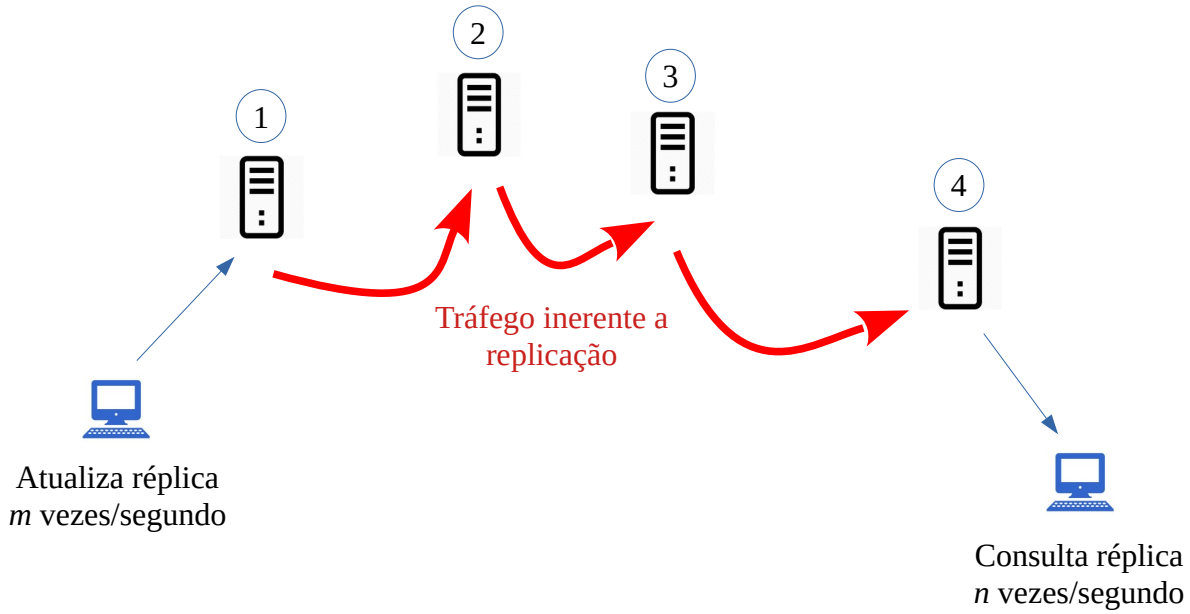
Por que replicar?

- Melhorar o desempenho:
  - Acesso a réplica mais próxima.
  - Balanceamento de carga → Sistema escalável.
- Tolerância a falhas.

Problemas inerentes a replicação:

- Maior custo de atualização.
- Tráfego de rede adicional.
- Consistência.
  - Em larga escala é difícil (“impossível”) manter a consistência total sem abdicar da eficiência. Uma alternativa é flexibilizar o nível de consistência.

Deve-se tomar cuidado para que o próprio tráfego na rede decorrente da existência de réplicas, ou até mesmo a carga de processamento na CPU, não reduzam excessivamente o ganho de desempenho estimado.



Se $n << m$, então muitas versões do dado na réplica 4 nunca serão acessadas. Neste caso, vale a pena tal réplica?

Intuitivamente, a consistência total requer sincronização global entre réplicas e, portanto, dificulta a escalabilidade do sistema. Se o sistema requer uma escrita global atômica, então certamente o desempenho será baixo.

Não há como escapar de um maior custo de atualização, então de que maneira podemos manter a consistência e ao mesmo tempo oferecer eficiência? A resposta encontra-se na flexibilização da consistência mediante o tipo e exigência de cada aplicação.

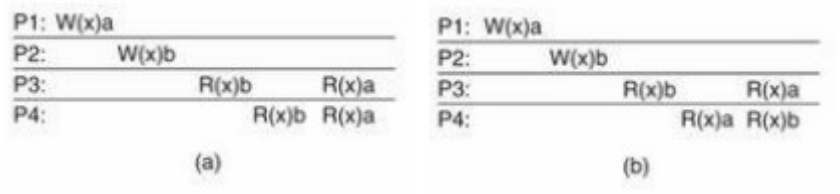
Como regra geral, temos: Quanto mais rígido for o nível de consistência, menos eficiente será o sistema.

1. Consistência contínua

Réplicas não necessariamente iguais…

- Tolerância numérica absoluta (por exemplo, até R$ 0,10) ou relativa (por exemplo, até 1%).
- Tolerância entre idades ou versões das réplicas (por exemplo, réplicas com dados de previsão do tempo).
- Tolerância em relação à ordem de atualização das réplicas.

2. Consistência sequencial



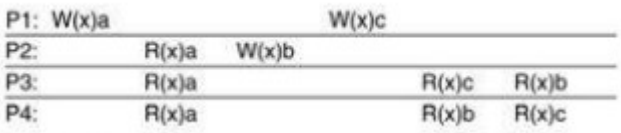
(a) O sistema distribuído oferece consistência sequecial
(b) Não há consistência sequecial

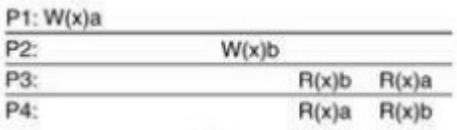
Todos os processos “veem” a mesma intercalação de operações de escrita.

Notar que na consistência sequencial o tempo absoluto em que ocorre a operação de escrita não é importante, apenas garante-se um sequência de atualização única em todas as réplicas.

3. Consistência causal

Trata-se da consistência sequencial aplicada somente a eventos potencialmente relacionados.





4. Consistência eventual

Por um dado momento as réplicas são inconsistentes, mas após um período todas as atualizações se propagam e as réplicas se tornam consistentes.

“Na ausência de atualizações, todas as réplicas convergem a cópias idênticas.”

5. Consistência por leitura monotônica

Leituras subsequentes de um objeto podem retornar a versão mais nova do mesmo, mas nunca uma versão anterior, como por exemplo quando uma réplica desatualizada é acionada.

6. Consistência por escrita monotônica

Operações de escrita realizadas por um processo devem ser propagadas em sua ordem original a todas as réplicas. Notar que operações de escrita concorrentes não estão sujeitas a essa restrição.

7. Consistência leia-suas-escritas

O processo que escreveu um valor, sempre obterá esse valor ou um valor mais recente em uma operação de leitura posterior.

8. Consistência escritas-seguem-leituras

Qualquer operação de escrita realizada por um processo deve ser efetuada sobre uma réplica que contenha um valor do objeto atualizado considerando-se o valor que o processo eventualmente leu antes da escrita propriamente dita.