ACH 2147 — Desenvolvimento de Sistemas de Informação Distribuídos

Aula 22: Consistência e Replicação (parte 3)

Prof. Renan Alves

Escola de Artes, Ciências e Humanidades — EACH — USP

24/05/2024

Protocolos primary-based

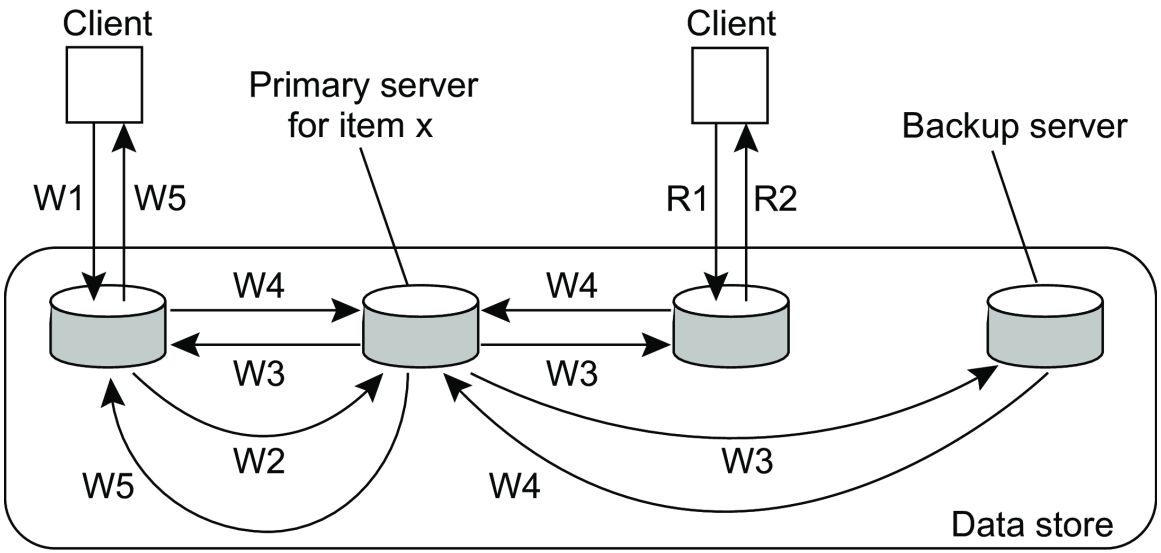
Implementação de consistência sequencial.

Ideia básica

- Cada item de dados x tem um `primário` associado
- O primário de x é responsável por coordenar as operações de escrita em x
- O primário pode ser um servidor remoto fixo
- Alternativamente, as operações de escrita podem ser realizadas localmente, após mover o primário para o processo onde a operação de gravação é iniciada.

Protocolos primary-based

Protocolos primary-based de backup

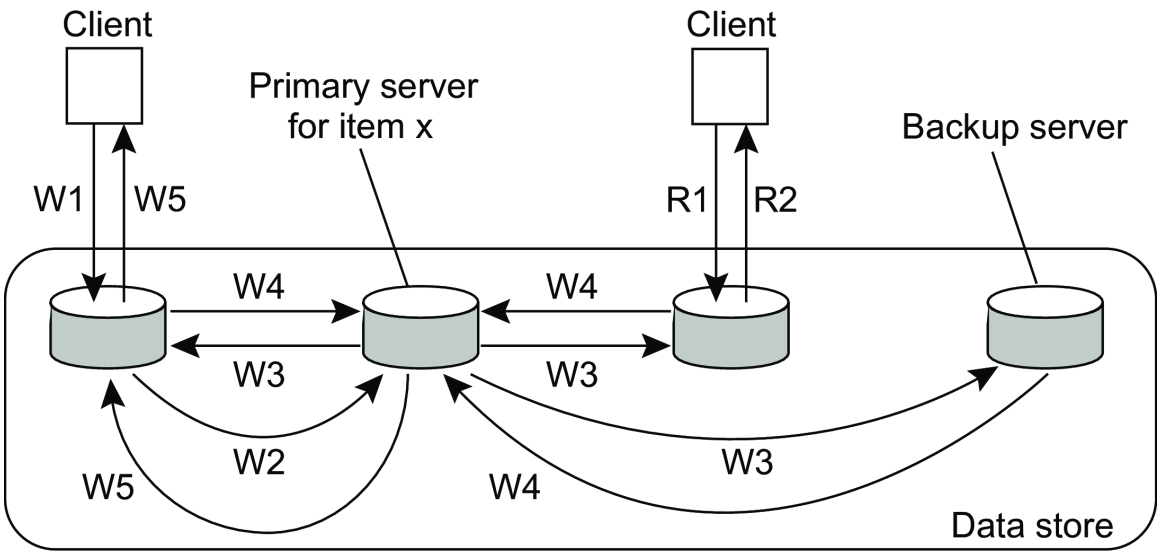


W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

Protocolos primary-based

Protocolos primary-based de backup



W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

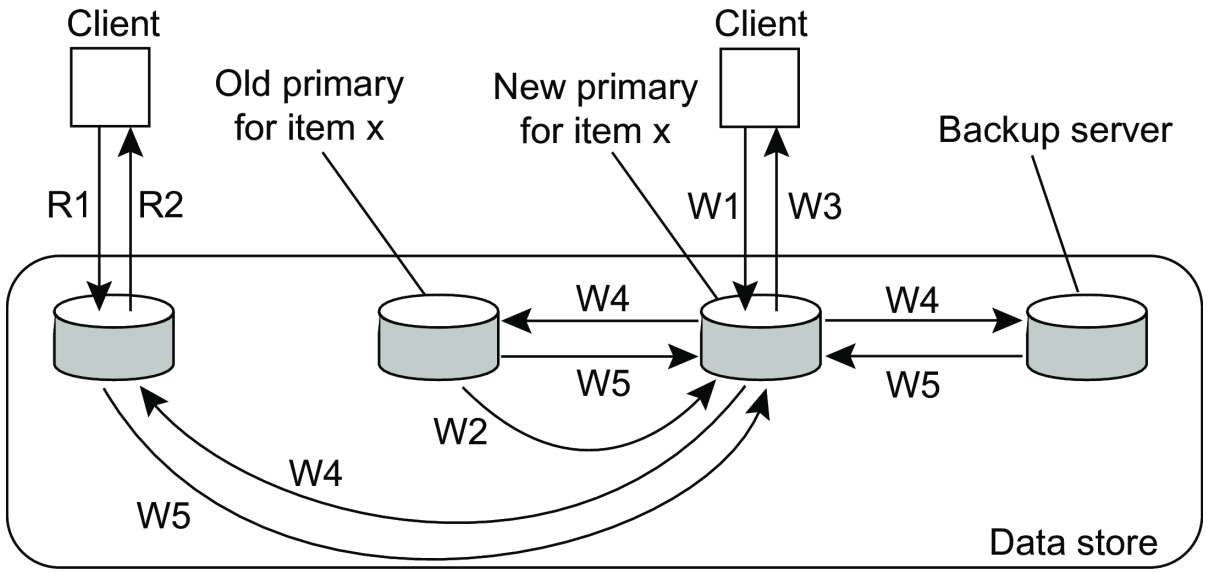
R1. Read request
R2. Response to read

Exemplo de protocolo primary-based de backup

Tradicionalmente aplicado em bancos de dados distribuídos e sistemas de arquivos que requerem um alto grau de tolerância a falhas. As réplicas muitas vezes são colocadas na mesma LAN.

Protocolos primary-based

Protocolo primary-based com escritas locais

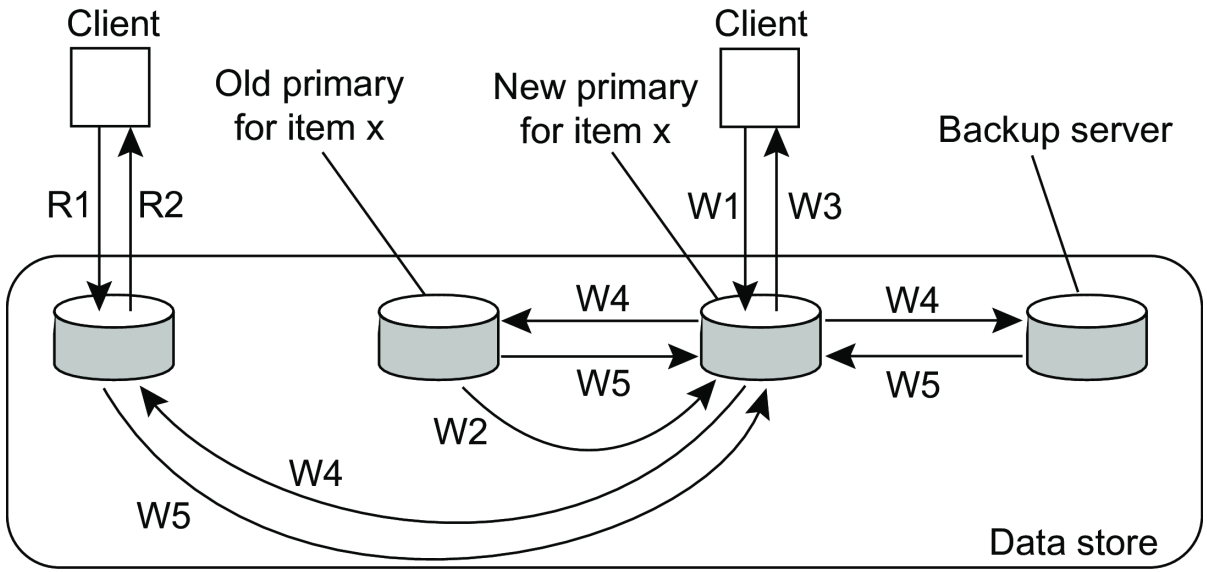


W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

Protocolos primary-based

Protocolo primary-based com escritas locais



W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

Exemplo de protocolo primary-based para backup com escritas locais

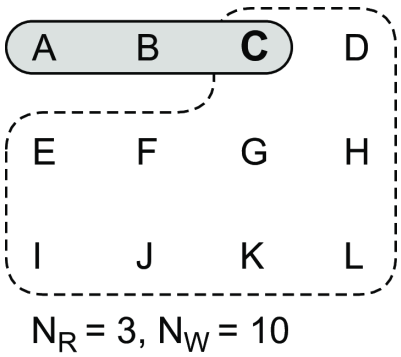
Computação móvel em modo desconectado (envio de todos os arquivos relevantes para o usuário antes da desconexão, atualizando posteriormente).

Protocolos de escrita replicada

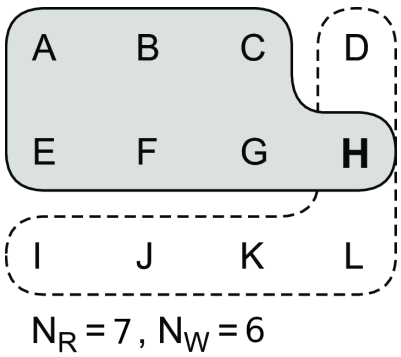
Protocolos baseados em quórum (quorum-based)

Assuma N réplicas. Garante que cada operação seja realizada de tal forma que um voto majoritário seja estabelecido: distinção entre quórum de leitura N_R e quórum de escrita N_W.
É preciso garantir que:

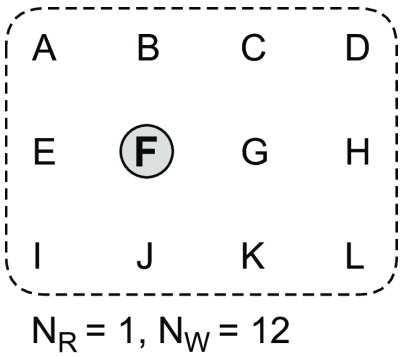
1. $N_R + N_W > N$ (evitar conflitos de leitura-escrita)
2. $N_W > N/2$ (evitar conflitos de escrita-escrita)



Correto



Pode ter conflito de W-W



Correto (ROWA)

Consistência contínua: erros numéricos

Funcionamento

- Cada servidor S_i tem um log, denominado L_i.
- Considere um item de dados x, com $val(W)$ denotando a mudança numérica em seu valor após uma operação de escrita W.
  Por simplicidade, assuma que

$$\forall W : val(W) > 0$$

- W é inicialmente encaminhado para uma das N réplicas, denotada como $origin(W)$.
  $TW[i,j]$ são as escritas executadas no servidor S_i que se originaram de S_j:

$$TW[i,j] = \sum\{val(W) | origin(W) = S_j \ \& \ W \in L_i\}$$

- $T[i,i]$ representa as operações de escrita submetidas em S_i

Consistência contínua: erros numéricos

Nota

Valor real $v(t)$ de x:

$$v(t) = v_{init} + \sum_{k=1}^{N} TW[k,k]$$

valor v_i de x no servidor S_i:

$$v_i = v_{init} + \sum_{k=1}^{N} TW[i,k]$$

Consistência contínua: Erros numéricos

Problema

Precisamos garantir que $v(t) - v_i < \delta_i$ para cada servidor S_i.

Consistência contínua: Erros numéricos

Problema

Precisamos garantir que $v(t) - v_i < \delta_i$ para cada servidor S_i.

Abordagem

- Cada servidor S_k mantém uma visão $TW_k[i,j]$ do que ele acredita ser o valor de $TW[i,j]$.
- Essas informações podem ser disseminadas por gossip quando uma atualização é propagada.

Consistência contínua: Erros numéricos

Problema

Precisamos garantir que $v(t) - v_i < \delta_i$ para cada servidor S_i.

Abordagem

- Cada servidor S_k mantém uma visão $TW_k[i,j]$ do que ele acredita ser o valor de $TW[i,j]$.
- Essas informações podem ser disseminadas por gossip quando uma atualização é propagada.

Nota

$$0 \leq TW_k[i,j] \leq TW[i,j] \leq TW[j,j]$$

Consistência contínua: erros numéricos

Solução

S_k envia operações de seu log para S_i quando vê que $TW_k[i,k]$ está se distanciando demais de $TW[k,k]$, em particular, quando

$$TW[k,k] - TW_k[i,k] > \delta_i/(N-1)$$

Consistência contínua: erros numéricos

Solução

S_k envia operações de seu log para S_i quando vê que $TW_k[i,k]$ está se distanciando demais de $TW[k,k]$, em particular, quando

$$TW[k,k] - TW_k[i,k] > \delta_i/(N-1)$$

Pergunta

Até que ponto estamos sendo pessimistas: de onde vem $\delta_i/(N-1)$?

Consistência contínua: erros numéricos

Solução

S_k envia operações de seu log para S_i quando vê que $TW_k[i,k]$ está se distanciando demais de $TW[k,k]$, em particular, quando

$$TW[k,k] - TW_k[i,k] > \delta_i/(N-1)$$

Pergunta

Até que ponto estamos sendo pessimistas: de onde vem $\delta_i/(N-1)$?

Nota

A consistência por antiguidade relativa (staleness) pode ser feita de forma análoga, essencialmente acompanhando o que foi visto por último em S_i (por exemplo, através de relógios vetoriais).

Implementando consistência centrada no cliente

Resumidamente

Para cada operação de escrita *W* é atribuído um identificador globalmente único pelo seu servidor de origem. Para cada cliente, mantemos dois conjuntos de escritas:

- Conjunto de leitura: contém (identificadores das) escritas relevantes para as operações de leitura desse cliente
- Conjunto de escrita: contém (identificadores das) operações de escrita do cliente.

Implementando consistência centrada no cliente

Consistência de leitura monotônica

Quando o cliente *C* deseja ler no servidor *S*, *C* passa seu conjunto de leitura. *S* traz quaisquer atualizações antes de executar a operação de leitura, após o que o conjunto de leitura é atualizado.

Implementando consistência centrada no cliente

Consistência de leitura monotônica

Quando o cliente *C* deseja ler no servidor *S*, *C* passa seu conjunto de leitura. *S* traz quaisquer atualizações antes de executar a operação de leitura, após o que o conjunto de leitura é atualizado.

Consistência de escrita monotônica

Quando o cliente *C* deseja escrever no servidor *S*, *C* passa seu conjunto de escrita. *S* traz quaisquer atualizações, executá-as na ordem correta e, em seguida, executa a operação de escrita, após o qual o conjunto de escrita é atualizado.

Implementando consistência centrada no cliente

Consistência leia-suas-escritas (read-your-writes)

Quando o cliente *C* deseja ler no servidor *S*, *C* passa seu conjunto de escrita. *S* traz quaisquer atualizações antes de executar a operação de leitura, após o que o conjunto de leitura é atualizado.

Implementando consistência centrada no cliente

Consistência leia-suas-escritas (read-your-writes)

Quando o cliente *C* deseja ler no servidor *S*, *C* passa seu conjunto de escrita. *S* traz quaisquer atualizações antes de executar a operação de leitura, após o que o conjunto de leitura é atualizado.

Consistência escritas-seguem-leituras (writes-follows-reads)

Quando o cliente *C* deseja escrever no servidor *S*, *C* passa seu conjunto de leitura. *S* traz quaisquer atualizações, executa-as na ordem correta e, em seguida, executa a operação de escrita, após o qual o conjunto de escrita é atualizado.

Implementando consistência centrada no cliente

Problema

Desempenho:

- Os conjuntos de leitura e escrita podem ficar grandes demais
- O atraso para que o servidor se atualize pode ser proibitivo

Exemplo: replicação na Web

Cache no lado do cliente

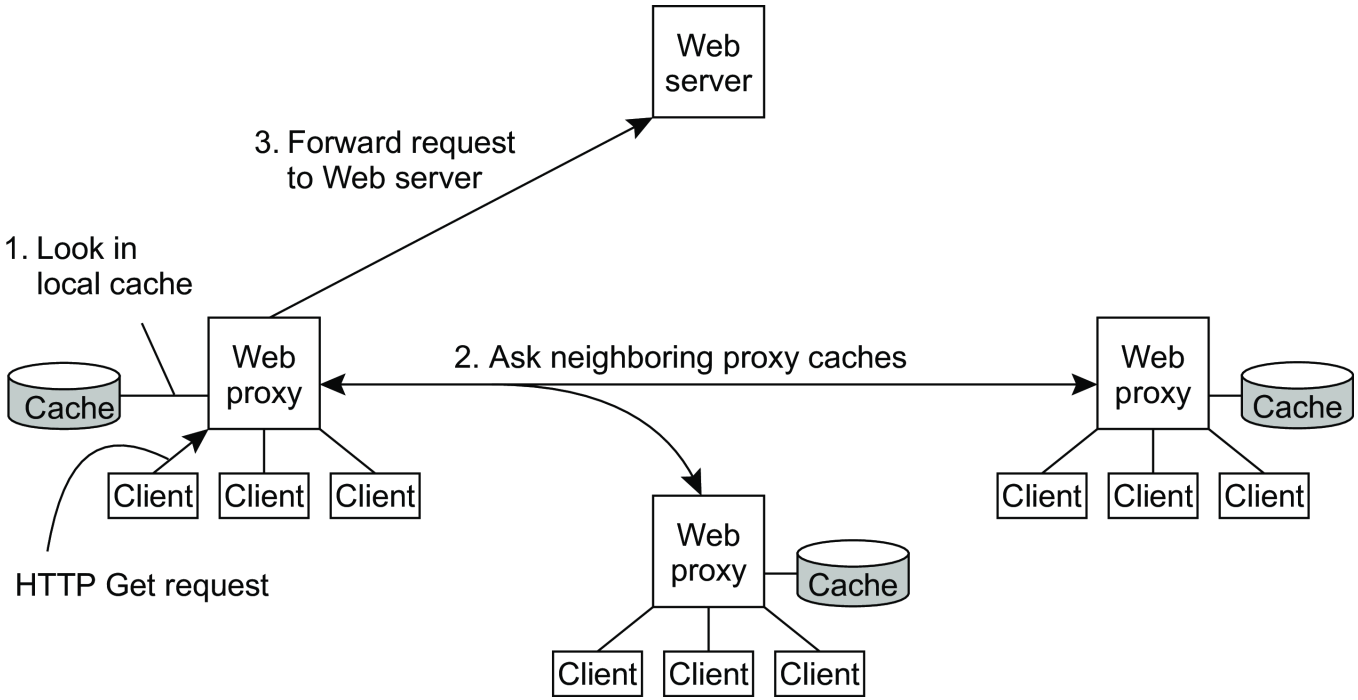
- No navegador
- Na localidade de um cliente, geralmente através de um proxy Web

Caches nos ISPs

Os Provedores de Internet também colocam caches para reduzir o tráfego entre ISPs e melhorar o desempenho do lado do cliente.

Cache cooperativo

Alternativa ao modelo puramente hierárquico.



Consistência do cache da Web

Como garantir informações recentes?

Para evitar que informações obsoletas sejam retornadas a um cliente:

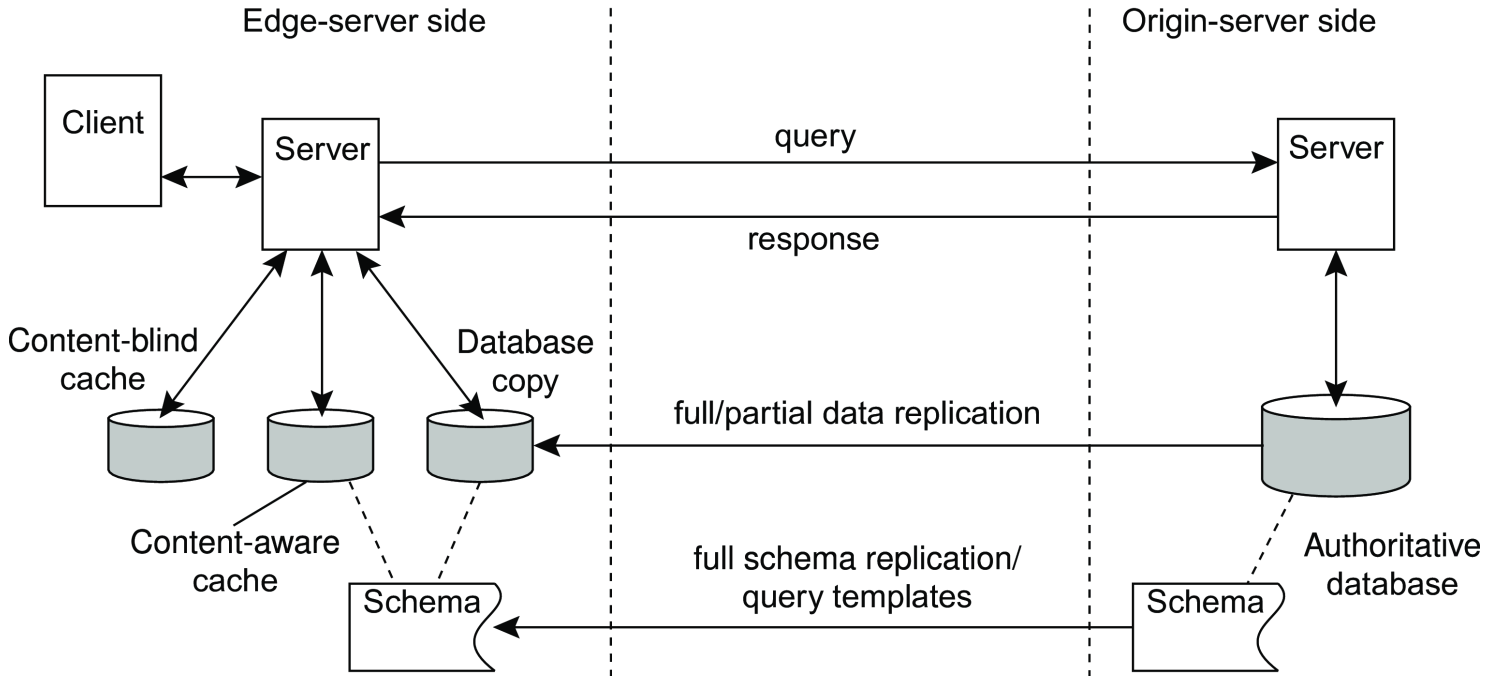
- Opção 1: permitir que o cache entre em contato com o servidor original para verificar se o conteúdo ainda está atualizado.
- Opção 2: Atribuir um instante de expiração T_{expire} que depende de quanto tempo atrás o documento foi modificado pela última vez quando foi armazenado em cache. Se $T_{last\_modified}$ é o último instante de modificação de um documento (registrado pelo seu proprietário), e T_{cached} é o tempo em que foi armazenado em cache, então

$$T_{expire} = \alpha(T_{cached} - T_{last\_modified}) + T_{cached}$$

geralmente com $\alpha = 0.2$ (valor empírico).

Até T_{expire}, o documento é considerado válido.

Tipos de caching e replicação



- Cópia completa do banco de dados: a borda (edge) tem o mesmo que o servidor de origem
- Cache consciente de conteúdo: verifica se uma consulta normal pode ser respondida com dados em cache. Requer que o servidor saiba quais dados estão em cache na borda.
- Cache sem considerar o conteúdo: armazena (o hash de) uma consulta e seu resultado. Quando uma consulta exatamente igual é feita novamente, retorna o resultado do cache.