INF14006 — Programação Distribuída e Concorrente

Servidores Replicados usando Filas MQTT
Trabalho 4 — 2021.2

apresentações de andamento: 29/11, 1/12 e 6/12
entrega: 12/12
demonstrações: 13 e 15/12

Profa. Noemi Rodriguez
Departamento de Informática, PUC-Rio

O objetivo do trabalho é implementar um grupo de servidores que utiliza um serviço de tópicos MQTT para atender a pedidos de clientes e também para comunicação interna no grupo na implementação de tolerância a falhas.

Cada servidor, como no trabalho 3, manterá um repositório de duplas (chave, valor) que pode ser consultado ou atualizado pelos clientes. Os clientes enviam requisições ao serviço através de um tópico MQTT de nome "inf1406-reqs". A mensagem enviada deve ser uma string json. Como no trabalho 3, existem dois tipos de requisição, consultas e inserções. Agora *todos os servidores* estarão escutando no tópico "inf1406-reqs"e todos manterão uma réplica de todos os dados. As operações de inserção (atualização) serão executadas em todas as réplicas do serviço. As requisições de consulta (leitura) serão executadas por apenas um dos servidores.

O sistema deverá incluir um grupo de processos servidores e um processo separado que fará o papel de monitor. (Como veremos, o monitor cria um ponto de falha, mas não vamos tratar desse problema no trabalho — numa solução real, precisaríamos também de um grupo de monitores, ou os próprios servidores poderiam fazer o papel de monitores.)

O trabalho pode ser desenvolvido em linguagem de sua escolha entre Lua, C, Rust, Java ou Go.

Funcionamento Normal

Para fazer uma requisição, o cliente envia uma msg para o tópico "inf1406-reqs"contendo um *json*. Esse json deve sempre ter campos *tipomsg*, *chave*, *topico-resp* e *idpedido*. No caso de um pedido de inserção, um campo *novovalor* deve existir também.

Ao serem disparados, cada um dos processos servidores recebe um identificador de 0 a $n-1$, onde n é o número total de servidores.

Cada pedido de inserção deve ser executado por todos os servidores. O serviço de filas entregará as requisições na mesma ordem para todos os servidores, garantindo sua consistência.

No caso de consulta, apenas um dos servidores executará a requisição e enviará a resposta para o *ropico-resp* indicado nela. Para determinar que servidor atenderá a requisição, vamos usar a mesma função de hash do trabalho 3:

$$\left(\sum_{i=1}^{\#\text{chave}} \text{i-ésimo byte da chave}\right) \bmod n$$

Vamos supor que chave e valor são sempre strings.

Tolerância a Falhas

Um processo monitor deverá ficar escutando no canal "inf1406-monitor". Cada um dos processos servidores deve mandar uma mensagem de *heartbeat* periodicamente. Essa mensagem será uma string json com um campo *idServ*. O monitor deve estabelecer um *timeout* para recebimento de *heartbeats* de cada servidor. Caso um deles expire, o monitor conclui que esse servidor falhou (vamos considerar que as únicas falhas possíveis são *crash* de servidores, e que a rede e os demais componentes permanecem sempre operantes.).

Quando o monitor concluir que um servidor falhou, ele deve enviar para o tópico "inf1406-reqs"uma mensagem com campos *tipomsg*, com valor "falhaserv", *idserv*, indicando o id do servidor que falhou, e *vistoem*, com a hora do último *heartbeat* recebido desse servidor.

Ao receber essa mensagem, o servidor de id $(idserv + 1) mod\ n$, que chamaremos de servidor substituto, deve responder as requisições de leitura destinadas ao servidor que falhou recebidas depois de *vistoem*. Para que isso seja possível, cada servidor deve manter um *log* de todas as requisições recebidas, com expiração de um tempo X (experimente com X em torno de dez vezes maior que o tempo para detecção de falha usado pelo monitor). Observe que um cliente pode receber uma resposta duplicada, pois não sabemos se o servidor *idserv* respondeu ou não essas requisições antes de falhar.

O servidor substituto deve continuar respondendo a novas requisições de leitura que seriam responsabilidade de *idserv* até que o novo servidor esteja pronto. Ao entrar em atividade, o servidor novo deve enviar uma mensagem para o tópico "inf1406-reqs"com campos *tipomsg*, com valor "novoserv", e *topico-resp*. O seu servidor substituto deve então enviar para este *topico-resp* todo o estado atual, um conjunto de duplas (chave, valor).

A Figura 1 mostra a arquitetura de tópicos do sistema.

Resta um problema ainda, que é sincronizar o servidor substituto com o novo servidor, determinando em que momento requisições de leitura voltam a ser atendidas por *idserv*. Proponha uma solução para essa sincronização.

Testes

Na entrega, documente como vc testou o seu programa e o que funcionou ou não.

Seus testes devem se dividir em (a) funcionamento normal e (b) funcionamento com falhas nos servidores. Para o caso (b), procure criar scripts que depois de decorrido determinado tempo matem um processo servidor. Além disso, ou se não fizer scripts assim, documente a sequência de passos que realizou para testar a resistência de seus servidores a falhas.

Entrega

A entrega deve conter os programas cliente, servidor e monitor, os scripts de teste e um arquivo (texto ou pdf) relatando as decisões de implementação e os testes realizados.

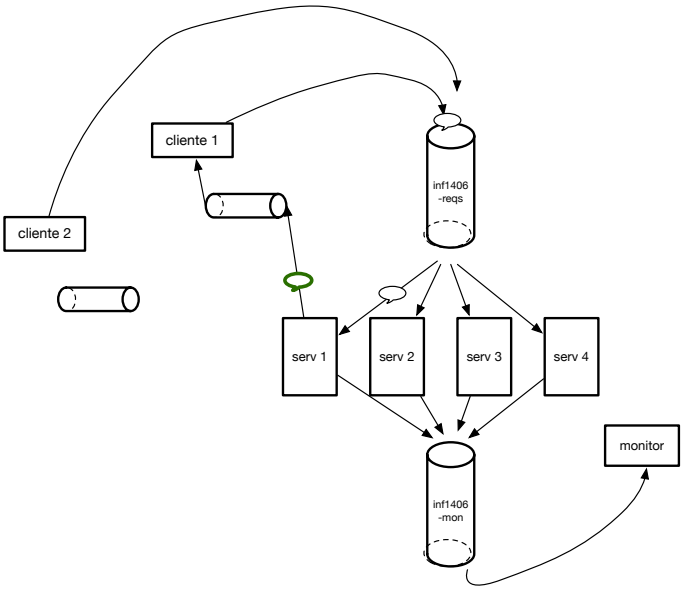

Figura 1: Arquitetura básica do sistema.