A48 – Segunda Entrega do Projeto de Sistemas Distribuídos

 83420 – Alexandra Figueiredo

 83443 – Denis Voicu

 83520 – Mariana Loureiro

GitHub: https://github.com/tecnico-distsys/A48-SD18Proj

Descrição da solução

A abordagem usada tem base no *Quorum Consensus*, com algumas alterações com o objetivo de melhorar a eficiência do projeto. Nesse sentido, é assumido que existem três réplicas *Station*, e que não pode haver mais que uma falha em simultâneo. As chamadas remotas esperam apenas por duas respostas. Além disso, o servidor *Binas* pode falhar ocasionalmente, o que pode causar a perda ou corrupção na informação dos utilizadores.

Uma das alterações realizadas foi na estrutura da *tag*. Foi decidido que a *tag* seria um número sequencial, que iria ser incrementado apenas quando é realizado a operação *setBalance,* ou seja, quando é realizada a operação de escrita. Isso é possível porque existe apenas um cliente, pelo que não é necessário guardar o *clientId*.

Além disso, a *tag* mais recente é também guardada pelo utilizador, do lado do *Binas*, tal como o saldo correspondente à escrita com essa *tag*. Assim, a leitura realizada antes de se fazer uma escrita, com o objetivo de procurar qual a *tag* mais recente, passa a ser desnecessária, já que pode ser obtida do lado do Binas. Este procedimento funciona apenas quando o servidor *Binas* não falha, ou seja, quando não são perdidas as informações dos utilizadores. Se assim for, há a garantia que o saldo guardado pelo utilizador é o mais atual.

Numa situação de falha do servidor *Binas*, em que os registos de utilizadores são completamente perdidos, a associação de email e saldo de cada utilizador persiste nas réplicas station. Assim sendo, na realização de operações *returnBina* e *getBina* posteriores, admitindo uma verificação da existência do utilizador (via *getBalance*), este voltará a ser instanciado no servidor *Binas* com os valores por omissão.

Quando as leituras na estação são infrutíferas (o utilizador não tem conta instanciada nas *stations*), isto é passado com valores inteiros '-1' na *tag* e saldo, e é tratado do lado do Binas. A *thread* que executa este pedido do lado do *Binas* espera pelas respostas inertemente – é 'adormecida' e verifica se recebeu alguma resposta periodicamente.

No caso específico do *setBalance*, foi implementado de forma a necessitar do *getBalance* para fazer a verificação (e, caso seja necessário, a re-inicialização) dos utilizadores, antes de executar a operação em si. Isto evita uma chamada de leitura remota desnecessária às estações, como explicado em cima, já que o *getBalance* devolve o valor guardado no Binas caso este esteja disponível.

Foi optado por uma estratégia de implementação de *polling*, porque, comparativamente ao *callback*, a atualização da informação aproxima-se de ser em tempo real, e apesar de ser mais custosa, o sistema nunca vai estar numa situação de falta de recursos.

Troca de mensagens

De acordo com o *Quorum Consensus*, o cliente faz a operação para todos os servidores, e recebendo resposta de uma maioria deles, assume sucesso. Considerando qualquer tipo de exceção erro na resposta, e sabendo que o número de *stations* simultâneas é 3, se duas chamadas remotas devolverem corretamente, a operação foi bem-sucedida.

No caso do *getBalance*, a chamada enviará o email do utilizador e devolverá uma estrutura com dois valores – o saldo e a *tag*. Neste caso, é selecionado o valor cuja *tag* seja mais recente (maior *sequence number*). No caso do *setBalance*, é enviado o *email*, o novo saldo, e a *tag*, e o valor de retorno é *void*.

Descrição da figura

Assume-se que:

- *Station* S1 contém informação de 2 clientes: [a.a@b.b, 10, 1] e [c.c@d.d, 10, 1].
- *Station* S2 contém informação de 1 cliente: [a.a@b.b, 10, 1]
- *Station* S3 contém informação de 1 cliente: [c.c@d.d, 10, 1]

No início, o servidor *Binas* sofreu uma falha que resultou na perda da informação dos clientes. De seguida, é recebido um conjunto de instruções que estão numeradas cronologicamente. O cliente 1 envia uma instrução que inclui o *getBalance*(1). Como o *Binas* não tem informação sobre nenhum cliente, o binas vai obter essa informação através do envio de *getBalance*(2) às estações(S1,S2,S3). É obtida a resposta de apenas duas estações, uma vez que uma das chamadas perde-se. O *Binas* ao receber a resposta das 2 estações, compara as *tags* das repostas, escolhe a maior delas e regista essas informações(*email*, saldo, *tag*) no *Binas*(4). Caso o cliente não exista na *station* é retornado [-1,-1] como resposta.

O cliente 2 faz outra instrução que inclui o *setBalance*(5). Como o binas não tem as informações relativamente a esse cliente, é feito o *getBalance*(6) as estações. Pelo menos duas das estações respondem, e registam-se no *Binas* as informações relativas à resposta com a maior *tag*. De seguida, é enviado o *setBalance*(10) a todas as estações com os valores atualizados, ou seja, com o novo saldo e com a *tag* incrementada, e posteriormente, os valores são atualizados no *Binas*(11). Para o caso em que o servidor não tenha o utilizador com o referido *email*, então é criado esse utilizador na estação. Se já existir as informações deste, são atualizadas.