Relatório da 2ª entrega do Projeto de Sistemas Distribuídos

2 de Maio de 2018

80832
Margarida Ferreira

81805
Duarte David

83557
Ricardo Brancas

Repositório com o projeto:
https://github.com/tecnico-distsys/A60-SD18Proj.git

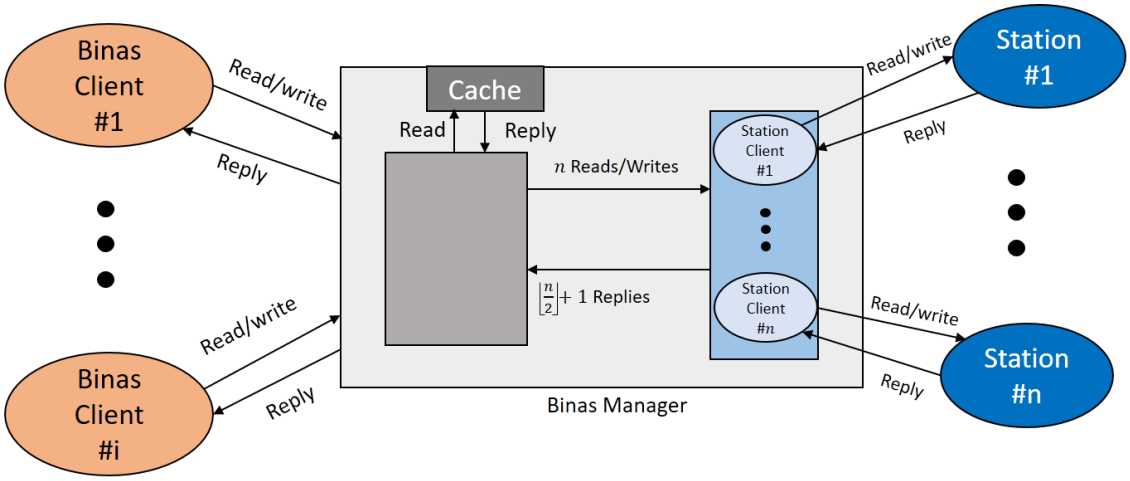

Figura 1: Funcionamento do Binas *Manager* com o protocolo de ***Quorum Consensus***, com n réplicas. Mantemos uma *cache* para evitar leituras às réplicas desnecessárias (se a leitura na *cache* for bem sucedida, não se tenta ler das réplicas).

O protocolo de ***Quorum Consensus*** consiste numa técnica de replicação ativa que depende da existência de um *quorum* (uma maioria) para garantir a correção dos valores guardados no sistema. Na situação particular do Binas, é possível fazer várias simplificações dado que existe apenas um *client front-end* (o binas-ws).

Em primeiro lugar, podemos descartar a parte da *tag* correspondente ao `client-id`, pois só existe um cliente, bastando então um número de sequência. Como tal, é possível assumir que o servidor Binas sabe (quase) sempre qual a última *tag* utilizada (uma exceção é, por exemplo, a situação em que o servidor acabou de iniciar). Para além disso, como não existem mais clientes, o último saldo escrito (com sucesso) para um determinado *user* é sempre o valor que seria lido em seguida; é assim possível omitir as leituras em quase todos os casos.

Passamos então a descrever o protocolo implementado:

- Quando o binas-ws é iniciado, é criada uma cache (inicialmente vazia) de utilizadores.
- Quando chega um pedido de leitura de um *user*, é primeiro verificado se o *user* existe na cache. Se existir, então o valor é devolvido imediatamente (pois temos a garantia que é o mais recente). Se não existir, são consultadas as estações (segundo o protocolo de *Quorum Consensus*) e o *user*, caso exista, é trazido para cache. Caso não exista, é devolvido um erro.
- Quando chega um pedido de escrita, é primeiro verificado se o *user* já existe em cache. Caso exista, é lançado um pedido de escrita para as estações (em *QC*) e, apenas caso tenha sucesso, é feita a escrita na cache também. Caso o *user* não esteja em cache, é feito um pedido de leitura e de seguida procede-se como descrito acima.

Caso em qualquer um dos pedidos relacionados com a replicação o *Quorum Consensus* não seja atingido, o pedido é repetido até que tenha sucesso.

Troca de Mensagens

Para realizar o protocolo descrito acima, definimos dois novos tipos de mensagens trocadas entre os *StationClients* e as *Stations*.

- **getBalance**, envia um e-mail (*String*) e recebe o crédito do cliente (*Integer*) e a *tag* associada(*Integer*). Enviamos esta mensagem sempre que não temos dados associados a um cliente na *cache* do Binas Manager.
- **setBalance**, envia um e-mail (*String*), o novo crédito do cliente (*Integer*) e a tag associada(*Integer*), para que a *Station* guarde. Recebe uma confirmação da *Station*.

A *tag* é um inteiro guardado no Binas Manager, incrementado de um a cada mensagem enviada por este.

Assincronismo

As mensagens entre os *StationClient*s e as *Stations* são trocadas de forma assíncrona, de forma a que a execução possa prosseguir sem esperar por todas as respostas (como o protocolo do *Quorum Consensus* permite). A nossa implementação, suportada por boas práticas de Programação Orientada a Objetos, cria um objeto do tipo `QuorumConsensus`. Este objeto tem um método `run()` , que envia as mensagens para as *Stations* e associa a cada envio uma função de *Callback* que incrementa atomicamente um contador nele próprio. O método `get()` retorna o resultado das mensagens quando obteve o número necessário de respostas (fazendo *Polling* ao contador) para atingir o *Quorum Consensus*, retornando o valor com a *tag* mais recente (se houver). Se obtiver erros em respostas suficientes, lança uma exceção `QuorumConsensusNotReached`.