Disciplina: Sistemas Distribuídos
Professora: Ana Cristina Barreiras Kochem Vendramin

Avaliação (valor 2,0)
Replicação de dados. gRPC.

Implementar um sistema distribuído simples para **replicação de dados**, utilizando o modelo ***push*** para propagação de atualizações entre os nós.

Considere 5 processos distintos: um **cliente**, um **líder** e **três réplicas.**

Utilize **gRPC** como middleware de comunicação entre esses processos.

1. **(0,3) Cliente**:
   - (0,15) Envia dados para o líder gravar;
   - (0,15) Consulta dados do líder.
2. **(0,9) Líder**:
   - (0,15) Recebe solicitações de gravação de dados do cliente e é responsável pela replicação desses dados. Ele salva os dados em seu log local, contendo época (versão do líder) e *offset* (número sequencial que representa a posição da entrada no log dentro de uma determinada época, indicando a ordem dos registros);
   - (0,15) Envia a nova entrada para as réplicas (modelo *Push*) e aguarda as confirmações (*acks*);
   - (0,15) Após receber a confirmação da **maioria das réplicas**, envia uma **ordem de *commit*** para que as réplicas efetivem a gravação no banco de dados final;
   - (0,15) O líder só marca uma entrada como ***committed*** após receber confirmação da maioria (quórum). Quando isso acontecer, ele deve confirmar a gravação ao cliente;
   - (0,15) Persiste todos os dados (intermediários e finais) com época e *offset*;
   - (0,15) Responde consultas do cliente.
3. **(0,8) Réplicas**:
   - (0,15) Recebe entradas de log do líder e armazena de forma persistente essas entradas localmente como **dados intermediários (*uncommited*)**, que **não podem ser considerados finais e nem serem lidos** até a ordem de *commit* do líder. Deve-se enviar uma confirmação (*ack*) ao líder;
   - (0,35) Verificar se a nova entrada é **consistente** com o log local. Quando uma réplica recebe uma entrada de log do líder, ela espera que essa entrada seja a continuação exata do seu próprio log local,

ou seja, que a época e o ***offset*** estejam alinhados com o que ela já tem.
        - (0,15) Em caso de consistência, deve-se aceitar a nova entrada corretamente;
        - (0,2) Em caso de inconsistência, a réplica deve truncar o log local, ou seja, apagar as entradas a partir do *offset* (índice) conflitante para remover dados inconsistentes ou que não foram confirmados pelo líder atual. Dessa forma, a réplica descarta as entradas divergentes e retorna a um estado consistente em relação ao líder. Em seguida, informa seu estado atual de log ao líder, permitindo que este envie novamente as entradas corretas, a partir do ponto de sincronização, para reconstruir o log na forma correta.
    - (0,15) Ao receber a ordem de *commit* do líder, **efetiva a gravação no banco de dados final**, tornando os dados visíveis e confiáveis para leitura.
    - (0,15) Persiste todos os dados (intermediários e finais) com época e *offset*.

Testes:

- Inclusão de dados corretos e confirmação normal;
- Desligar uma réplica temporariamente e simular a retomada com log desatualizado;
- Criar propositalmente entradas conflitantes em uma réplica e testar se ela corrige o log.
- Testar se as réplicas só aplicam os dados no banco após o comando de *commit* do líder.

Observações:

- Desenvolva uma interface com recursos de interação apropriados.
- É obrigatória a defesa da aplicação para obter a nota.
- O desenvolvimento do sistema pode ser individual ou em dupla.

Profª. Ana Cristina Barreiras Kochem Vendramin
DAINF/UTFPR