LEIC, Sistemas Distribuídos, 2018-2019

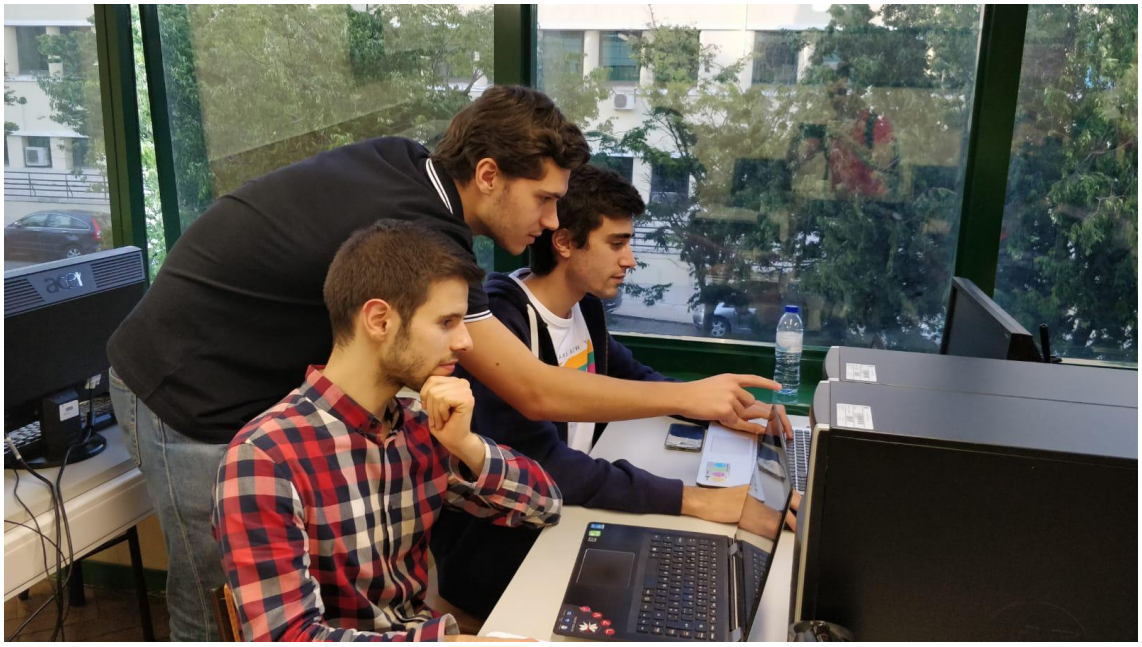
Projeto – Relatório para a 2ª Entrega (E2)

Turno: SDis12645111326L05, Seg, 11:00 – 12:30

Professor: Mauricio Breternitz

Grupo: A18

URL do repositório no GitHub: https://github.com/tecnico-distsys/A18-ForkExec



86443 João Tavares

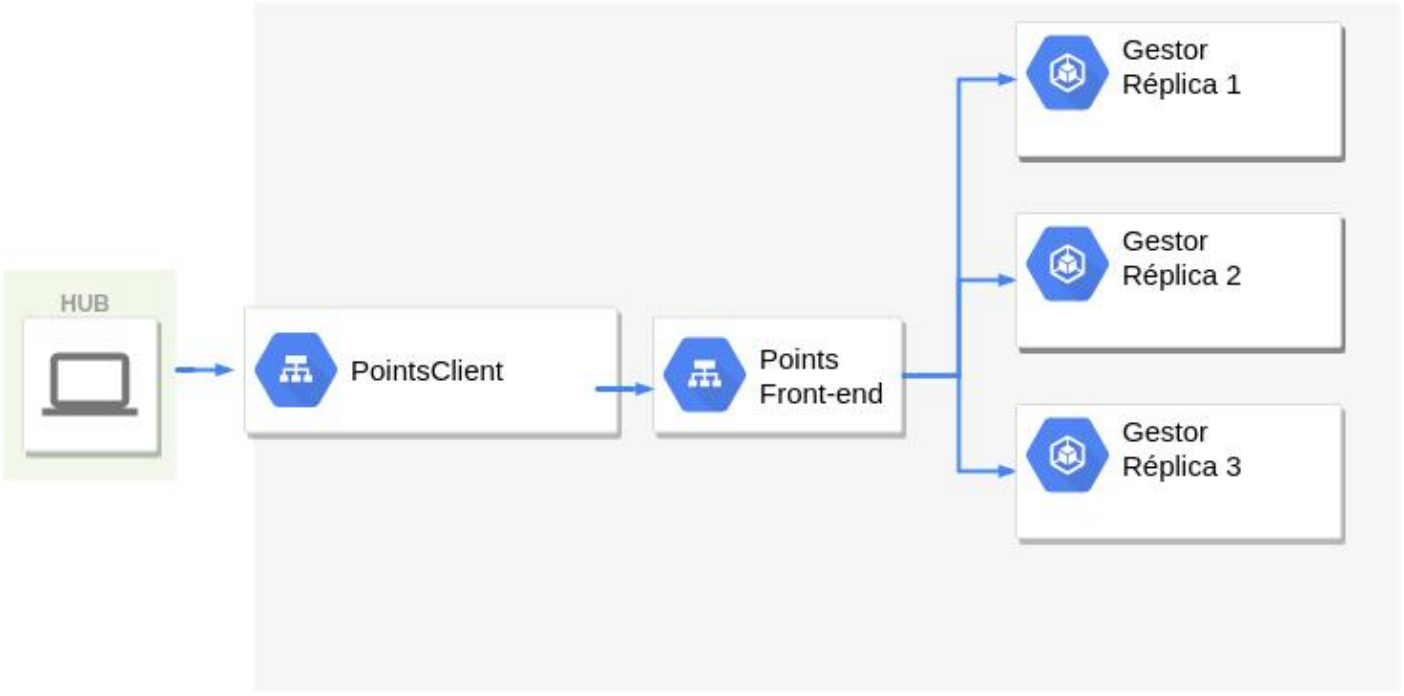
86493 Pedro Antunes

87629 André Leitão

DEFINIÇÃO DO MODELO DE FALTAS

Assumimos que o sistema é assíncrono e a comunicação pode omitir mensagens. Para além disso, que N gestores de réplica podem falhar silenciosamente mas não arbitrariamente, sendo N constante e igual a 3. No máximo, existe uma minoria de gestores de réplica em falha em simultâneo.
O modelo de faltas.
Para o modelo de Faltas considera-se um mecanismo redundante, com políticas de compensação de faltas com replicação ativa. Assim, o Front-End mensagens aos N gestores de réplicas e cada gestor executa a mensagem, read ou write, e envia a resposta.

FIGURA DA SOLUÇÃO DE TOLERÂNCIA A FALTAS



DESCRIÇÃO DA FIGURA E BREVE EXPLICAÇÃO DA SOLUÇÃO

A implementação considera que o cliente, em ligação com o hub, lida com a adaptação das operações antes presentes no servidor, nomeadamente, activateUser, pointsBalance, addPoints e spendPoints, adaptando-as a operações de read e write agora aplicadas no servidor. Desta forma, uma vez que o cliente realiza maior parte dos métodos, as exceções encontram-se maioritariamente no cliente e, portanto, lançadas por este. Para tal, foram criadas classes de exceções do seu lado e removidas do lado do servidor.
O Front-End lida com a existência de réplicas e do protocolo Quorum Consensus, em que, para a implementação, foi criada uma lista de clientes das réplicas. Estas, por sua vez, comunicam com os servidores de réplicas nos métodos de read e write, realizam o UDDI lookup e criam e configuram o stub.

DESCRIÇÃO DE OTIMIZAÇÕES/SIMPLIFICAÇÕES

O protocolo é, conforme supracitado, Quorum Consensus, para que possa ser consistente e, no entanto, tolerar falhas e ser eficiente. Cada réplica guarda o valor do objeto/registo, correspondente ao email e pontos, e o respetivo tag. A tag, de forma a simplificar o protocolo, é composta unicamente pela seq, o número de sequência da escrita que deu origem à versão, sem guardar o cid, o identificador do cliente que escreveu, dado que o modelo pressupõe um único cliente, uma instância de hub, não havendo necessidade de comparar o cid que é sempre igual.

Para otimizar a solução considerámos duas variantes do protocolo:

- Pesos Variáveis: O Front-End determina o peso para cada réplica conforme fiabilidade, conectividade ou poder computacional. Aleatoriamente, assumimos que a primeira metade de réplicas criadas é melhor nesses aspetos obtendo peso 2 e restantes peso 1. O Front-End, verifica pois que o Quórum tem peso superior a metade do peso total do sistema.
- Quóruns de Leitura e Escrita: O peso exigido para operações de read é distinto das de write, em que Read Threshold + Write Threshold > Peso Total do Sistema e WT > Peso Total do Sistema. Estes aspetos garantem que a interseção consecutiva dos quóruns sucessivos é não vazia.

No sistema do ForkExec as leituras são muito mais frequentes que as escritas, uma vez que, da parte do hub, há uma primeira leitura para verificação do saldo disponível, seguido de, nem em todos os casos, de escrita no saldo. Por sua vez, o protocolo Quorum Consensus garante que uma escrita implica sempre uma fase de leitura para determinar o valor de maxTag a que se segue uma fase de escrita com o valor de newTag = maxTag+1. Deste modo, obtemos #leituras >= 2 x #escritas, o que, entendemos como valores de RT = 1/3 Peso Total do Sistema e WT = 2/3 Peso Total do Sistema + 1 => WT = 2xRT + 1.

DETALHE DO PROTOCOLO (TROCA DE MENSAGENS)

As trocas de mensagens correspondem a um Modelo de Interação Assíncrono, em que não limita a latência da rede ou tempo de resposta do servidor. As Invocações Assíncronas são realizadas entre o Front-End e o gestor de cada réplica.