Instituto Superior Técnico, Lisboa
LEIC 2018/2019
Sistemas Distribuídos

Projeto ForkExec – Parte 2
Guião de demonstração
Grupo A40

65284	86448	87658
Rui Alves	João Coelho	Francisco Santos

Nota inicial: esta segunda entrega foi desenvolvida a partir do código fornecido pelos docentes. Não foi alterada a interface do cliente de pts pelo que ainda consegue comunicar com o módulo hub, mas este não estará presente durante a demonstração (assim como o módulo rst). Os pedidos que viriam do hub são simulados na classe PointsClientApp.

Caso F1 – Demonstração do funcionamento normal da aplicação

1. Obter o zip com o código do Fénix e extrair;
2. Abrir quatro terminais:
    - Instalar servidores (réplicas) - no terminal 1, no diretório *pts-ws*, correr o comando **mvn install**
    - Correr Réplica 1 - no terminal 1, no diretório *pts-ws*, correr o comando **mvn compile exec:java**
    - Correr Réplica 2 - no terminal 2, no diretório *pts-ws*, correr o comando **mvn compile exec:java -Dws.i=2**
    - Correr Réplica 3 - no terminal 3, no diretório *pts-ws*, correr o comando **mvn compile exec:java -Dws.i=3**
    - Instalar cliente – no terminal 4, no diretório *pts-ws-cli*, correr o comando **mvn install**
    - Correr Cliente - no terminal 4, no diretório *pts-ws-cli*, correr o comando **mvn compile exec:java -Dexec.cleanupDaemonThreads=false**
3. Quando o Cliente tiver terminado a sua execução, desligar as réplicas, pressionando *Enter* ou *CTRL+C* em cada um dos terminais das réplicas;
4. Não fechar nenhum dos terminais. O terminal 4, do cliente, não vai ser mais usado, apenas servirá para comparar o seu output com o output do caso F2.

Assim, durante a execução do guião, temos as 3 Réplicas a correr e ao executarmos o Cliente, na Classe *PointsClientApp*, após criarmos o objeto *PointsClient* invocamos sobre ele uma série de operações da interface disponível (mantivemos a interface da parte 1 do projeto).

Esta sequência de operações tem associados vários *prints* que permitem confirmar o correto funcionamento da aplicação. Deve agora ser comparado o output da execução do Cliente, após este terminar, com o código no fim do ficheiro PointsClientApp.java assinalado com o comentário **DEMONSTRATION**.

Caso F2 – Demonstração da tolerância a falta

1. Este caso assume que o primeiro já foi corrido;
2. Abrir um novo terminal (terminal 5) e voltar a correr as réplicas:
   - Correr Réplica 1 - no terminal 1, no diretório *pts-ws*, correr o comando **mvn compile exec:java**
   - Correr Réplica 2 - no segundo terminal, no diretório *pts-ws*, correr o comando **mvn compile exec:java -Dws.i=2**
   - Correr Réplica 3 - no terceiro terminal, no diretório *pts-ws*, correr o comando **mvn compile exec:java -Dws.i=3**
   - Correr Cliente - no terminal 5, no diretório *pts-ws-cli*, correr o comando **mvn compile exec:java -Dexec.cleanupDaemonThreads=false -Dcase=2** o parâmetro case=2 indica ao Cliente que estamos no caso 2 e é responsável pela impressão de mensagens extra.
3. Ao longo da execução do Cliente, deve seguir as indicações das mensagens a pedirem-lhe para desligar/ligar alguma réplica. Terá 10 segundos para cada uma delas (o *thread* principal estará adormecido durante este período). Estas mensagens foram colocadas estrategicamente para simular as faltas. Quando lhe for pedido para ligar de novo um servidor, correr o comando **mvn exec:java** com as flags **-Dws.i=2** e **-Dws.i=3** para as réplicas 2 e 3, respetivamente.

Neste segundo caso, as operações são exatamente as mesmas que no primeiro, pelo que, se o sistema é tolerante a faltas, então o output do Cliente deve ser o mesmo em ambos (excluindo as mensagens extra de pedidos para desligar e ligar servidores). A partir do momento em que um dos servidores é desligado, passamos a ter sempre apenas 2 servidores disponíveis. Mas dois são suficientes para obter quórum, mantendo-se assim a consistência dos dados e do serviço, tolerando as faltas de servidores em baixo