Sistemas Distribuídos Confiáveis

Tolerância a Faltas

Relatório de Desenvolvimento

António Anjo
a67660

João Monteiro
a67740

Ricardo Fernandes
pg32986

1 de Maio de 2017

Resumo

Relatório de desenvolvimento do primeiro trabalho prático da Unidade Curricular de Sistemas Distribuídos Confiáveis: Tolerância a Faltas. Neste documento apresentar-se-ão algumas das decisões tomadas durante o processo de desenvolvimento, assim como eventuais dificuldades encontradas.

Conteúdo

Capítulo 1

Introdução

O trabalho consiste na implementação em Java, usando o protocolo de comunicação em grupo Spread, de um par cliente/servidor tolerante a faltas.

A aplicação a desenvolver é um sistema de controlo de tráfego ferroviário. Considerando linhas ferroviárias divididas em segmentos numerados, o sistema deve garantir que nunca há duas composições num mesmo segmento assim como a existência de um segmento de linha livre entre duas composições.

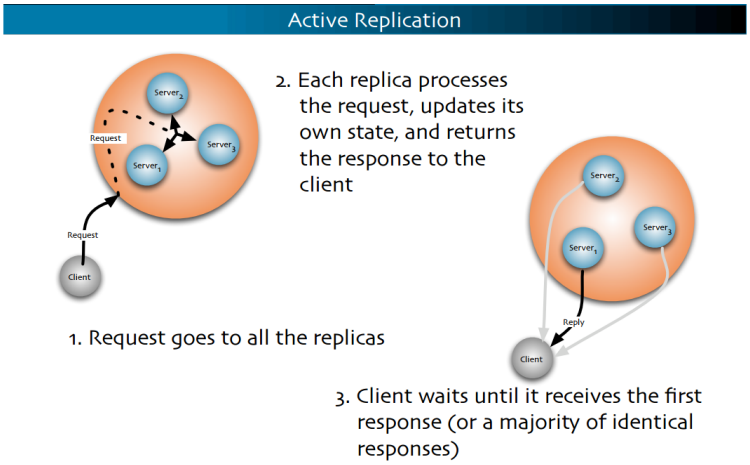
É esperado que composições possam efetuar pedidos de acesso a segmentos da linha, assim como enviar notificações de entrada ou saída de um segmento. O servidor deve poder manter uma lista de posições atuais das composições assim como um histórico de alarmes correspondentes a notificações de situações de perigo.

Capítulo 2

Tolerância a faltas

2.1 Replicação

O aspeto deste sistema que o torna tolerante a faltas é a utilização de um protocolo de **replicação ativa**, sendo a ideia deste protocolo que exista um grupo de no mínimo 2 servidores e, quando o cliente pretende fazer um pedido, este seja enviado a todas as réplicas.



Cada uma das réplicas processa então o pedido, atualiza o seu próprio estado e responde ao cliente. Fica assim ao cargo do cliente esperar pelas respostas que receber e tratá-las como entender.

Isto significa então que, em caso de falta de um servidor, os outros continuam a funcionar normalmente.

2.1.1 Mensagens Duplicadas

É importante notar que, dado que os pedidos chegam a múltiplos servidores e todos eles vão responder, o cliente vai receber múltiplas respostas para cada pedido.

Se esta situação não for devidamente tratada, isto pode levar a problemas no sistema, como o cliente ler a resposta ao pedido errado.

Como solução a este problema, cada pedido é etiquetado com um número de sequência, que é copiado para a resposta pelo servidor. Assim, o cliente ignora todas as respostas cuja etiqueta não correspondam ao pedido esperado.

2.1.2 Replicação Passiva

Uma alternativa ao protocolo de replicação utilizado seria a **replicação passiva**. Este tipo de replicação introduz o conceito de servidor primário.

Neste tipo de replicação, o pedido do cliente chega apenas ao servidor primário, sendo aqui processado. O servidor atualiza o seu estado e envia depois uma mensagem de atualização de estado às réplicas. Cada réplica atualiza o seu estado e envia uma mensagem de acknowledgment ao servidor primário. Só nesta fase é que este envia a resposta ao cliente. Em caso de falha do servidor primário, uma das réplicas toma o seu lugar.

Este tipo de replicação tem a vantagem de ser transparente para o cliente e resolve o problema mencionado anteriormente das mensagens replicadas. No entanto, a sua dificuldade acrescida de implementação levou nos a manter o uso da replicação ativa.

Capítulo 3

Implementação e Funcionamento

3.1 Ponto de Partida

Como ponto de partida para a implementação do sistema pedido, utilizou-se o código resultante das primeiras aulas práticas da Unidade Curricular, onde se utilizam protocolos de multicasting e replicação no contexto de um sistema de gestão de contas bancárias tolerante a faltas.

3.2 Troca de Mensagens

A comunicação entre as composições e o sistema de controlo tráfego é feita utilizando o protocolo de comunicação em grupo Spread.

O Spread é um toolkit open source destinado à utilização em aplicações distribuídas que requerem alta disponibilidade, performance e robustez de comunicação entre grupos de intervenientes.

Na aplicação desenvolvida, os vários servidores constituem um SpreadGroup, isto é, um grupo para o qual uma mensagem pode ser multicast para todos os seus elementos simultaneamente. Cada composição tem a ela associada também o seu próprio SpreadGroup privado, o qual vai estar especificado nas mensagens que envia.

Assim, quando uma composição pretende comunicar com os servidores, faz multicast da sua mensagem para todas as réplicas, as quais vão processar o pedido, obter o grupo privado da composição que o enviou através do método `getSender` e transmitir para esse grupo a resposta ao pedido.

3.3 Linhas e Segmentos

Como sugerido no enunciado do trabalho, o sistema está pré-configurado com um conjunto de linhas e respetivos segmentos.

Construíram-se 3 linhas unidireccionais, sendo que os seus segmentos são numerados por ordem crescente no sentido da linha.

- Linha 1 - 6 Segmentos
- Linha 2 - 8 Segmentos
- Linha 3 - 11 Segmentos

Ao entrar no sistema, cada composição escolhe uma linha, um segmento de entrada e um segmento de saída, utilizando o sistema de controlo de tráfego para entrar na linha e navegar até ao seu destino.

Cada segmento tem a ele associado um valor binário que indica se está, em dado momento, ocupado ou livre.

3.4 Teste e Navegação

3.4.1 Execução

Para executar o teste, deve-se começar por compilar o código na diretoria do projeto, com o comando `$ mvn clean compile`.

De seguida, iniciam-se os servidores, com o comando `$ ./run Servers`.

Iniciados os servidores, o sistema pode ser testado com o comando `$ ./run TrainsTest X` sendo X o número de composições que vão navegar o sistema durante o teste. Se este parâmetro não for fornecido, existirá apenas uma composição.

3.5 Navegação

- Cada composição escolhe aleatoriamente uma linha, um segmento de entrada e um segmento de saída. Durante a simulação, esta vai navegar a linha no sentido crescente entre estes dois pontos. Por exemplo, Linha 1 - Segmento 1 - Segmento 6.
- Para cada segmento do caminho a navegar, incluindo o de entrada, a composição envia ao servidor um pedido de entrada (Entry Request). O servidor responderá com a autorização para entrar quando e só se houver condições para tal. Aquando da receção da resposta, a composição avança para esse segmento.
- É enviada uma notificação para sinalizar que a composição está a entrar no segmento.
- Quando uma composição sai de um segmento que navegou, esta envia uma notificação ao servidor para que o estado do segmento em questão possa ser atualizado.
- Neste ponto, o servidor já atualizou o seu estado, refletindo a ocupação ou não dos segmentos envolvidos na operação.
- O processo repete-se até que todas as composições terminem a sua viagem.

3.5.1 Entry Request

Cada vez que é feito um Entry Request, isto é, que uma composição pede autorização para entrar num determinado segmento, o sistema faz a verificação de disponibilidade desse segmento.

Para uma composição obter permissão de entrada é necessário que tanto o segmento em questão como o segmento seguinte estejam disponíveis. Por exemplo, se uma composição quiser entrar no segmento 3, é necessário que os segmentos 4 e 5 estejam livres.

Se estas condições se verificarem, o servidor responde ao cliente que fez o pedido com a devida permissão.

Caso contrário, o servidor aguarda até que se verifiquem e só aí responde ao cliente. Assim, não existem respostas negativas a este pedido. As composições devem esperar até receberem uma resposta para avançar para o segmento.

3.6 Desafios na implementação

Embora a solução implementada funcione como previsto na grande maioria dos casos, existe um problema que causa que por vezes, quando se utiliza um número de composições superior a 1, o teste bloqueie. O grupo não conseguiu resolver este problema nem encontrar a sua causa.

Quando isto acontece, é necessário reiniciar os servidores.

Capítulo 4

Conclusões

Este projeto serviu para o grupo consolidar e pôr em prática os conhecimentos obtidos sobre replicação de servidores e transferência de estado, assim como multicasting e comunicação em grupo através do uso do protocolo Spread.

Foram atingidos os requisitos mínimos do trabalho, tendo sido desenvolvido um par cliente-servidor com replicação ativa e com transferência de estado, assim como uma aplicação de teste do sistema.

Ficam no entanto abertas vias para evolução, como a obtenção de medidas de desempenho e a realização de testes em multiplas máquinas com simulação de falhas.

Para além disso, foram encontradas algumas dificuldades na implementação do sistema, nomeadamente relativamente à operação concorrente de várias composições e servidores.