Universidade do Minho

Departamento de Informática

Trabalho Prático

Tolerância a Falhas

Eduardo Jorge a83344
João Vilaça a82339
José Resende A77486

12 de Junho de 2020

Conteúdo

1 Introdução

No âmbito da Unidade Curricular de Tolerância a Falhas, foi-nos pedido para desenvolver um supermercado online tolerante a faltas. Este projecto irá ser elaborado utilizando Java, Spread toolkit e uma base de dados HSQLDB.

O sistema deve guardar um catalogo que contem um descrição de cada produto e a quantidade disponível. Este sistema suporta iniciar uma nova compra, consultar o preço e disponibilidade de um produto, acrescentar um produto à encomenda e confirmar uma encomenda indicando se foi concretizada com sucesso.

Para a implementação será necessário um par cliente servidor capaz de suportar toda a API descrita em cima com a capacidade de replicação para tolerância a faltas, uma base de dados para armazenamento persistente do estado de cada um dos servidores e transferência de estado para permitir a reposição em funcionamento de servidores sem interrupção do serviço

Para alem destes requisitos básicos também foram atacados alguns pontos chaves que tornam este projecto mais relevante como por exemplo separação clara do código entre middleware genérico de replicação e a aplicação, minimizar as encomendas canceladas como consequência de faltas ou do funcionamento do mecanismo de replicação, tirar partido do sistema de bases de dados para actualizar os estado dos servidores nas diversas situações em que isso for necessário, nomeadamente, para diminuir o volume de informação copiada e lidar com partições do grupo no Spread.

2 Modelo

Pretende-se implementar um supermercado online, TNACOP, usando o protocolo de comunicação em grupo Spread, e que seja tolerante a faltas. Este serviço deverá ser capaz de permitir o registo de novos clientes e que os estes criem encomendas que incluam um ou mais produtos. Deverá ainda guardar um catálogo contendo uma descrição de cada produto e a quantidade disponível.

Para um correto funcionamente o serviço deverá disponibilizar então as seguintes operações:

- Criar, indexar e mostrar clientes
- Indexar e mostrar produtos
- Criar e mostrar encomendas
- Adicionar produtos a encomendas
- Fazer submissão de uma encomenda

2.1 Especificação Técnica

- Número finito de processos
- Processos todos interligados com comunicação através do envio de mensagens
- Leituras/Escritas: operação "read" tem de retornar o último valor escrito
- Clientes verem o sistema apenas como um servidor central e sequencial
- A execução de comandos dos clientes é não determinística

2.2 Pressupostos

- Sistema assíncrono
- A rede assegura uma ordem FIFO na comunicação
- Garantia das propriedades de pertença a grupo (Self Inclusion, Local Monotonicity, Agreement e Linear membership)

2.3 Modelo de Faltas

- Faltas por paragem silenciosa (crash)
- Faltas por paragem detectável (failstop failures)

3 Replicação Passiva

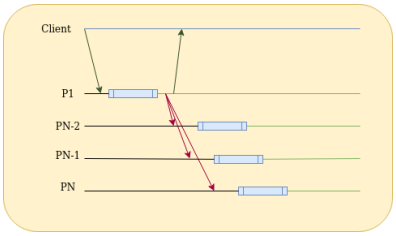
Uma vez que as operações que os utilizadores executam não serem sempre determinísticas, principalmente devido à variabilidade dos stocks dos produtos adicionados às encomendas, as operações de adição e de confirmação de encomenda vão sempre depender do estado atual dos stocks, ou seja, não determinístico. Deste modo, a concorrência que duas operações deste género implicam caso estivéssemos num situação de replicação ativa representaria possíveis inconsistências de estado, daí a decisão de implementar replicação semi-passiva.

3.1 Protocolo de Replicação

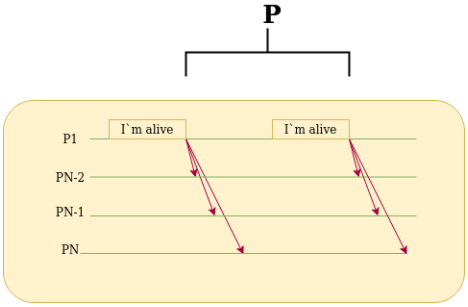
P1: servidor primário, PN: servidores secundários

1) Quando P1 recebe um pedido:

- Processa o pedido
- Caso seja uma operação que atualize ao estado, executa essa atualização e reenvia o pedido aos PN
- Responde ao cliente, sem esperar pela resposta de aos PNs
- PNs processam o pedido e atualizam o seu estado

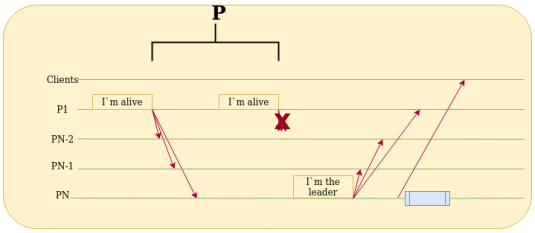


2) P1 envia aos PNs mensagens "I´m alive" cada P unidades de tempo

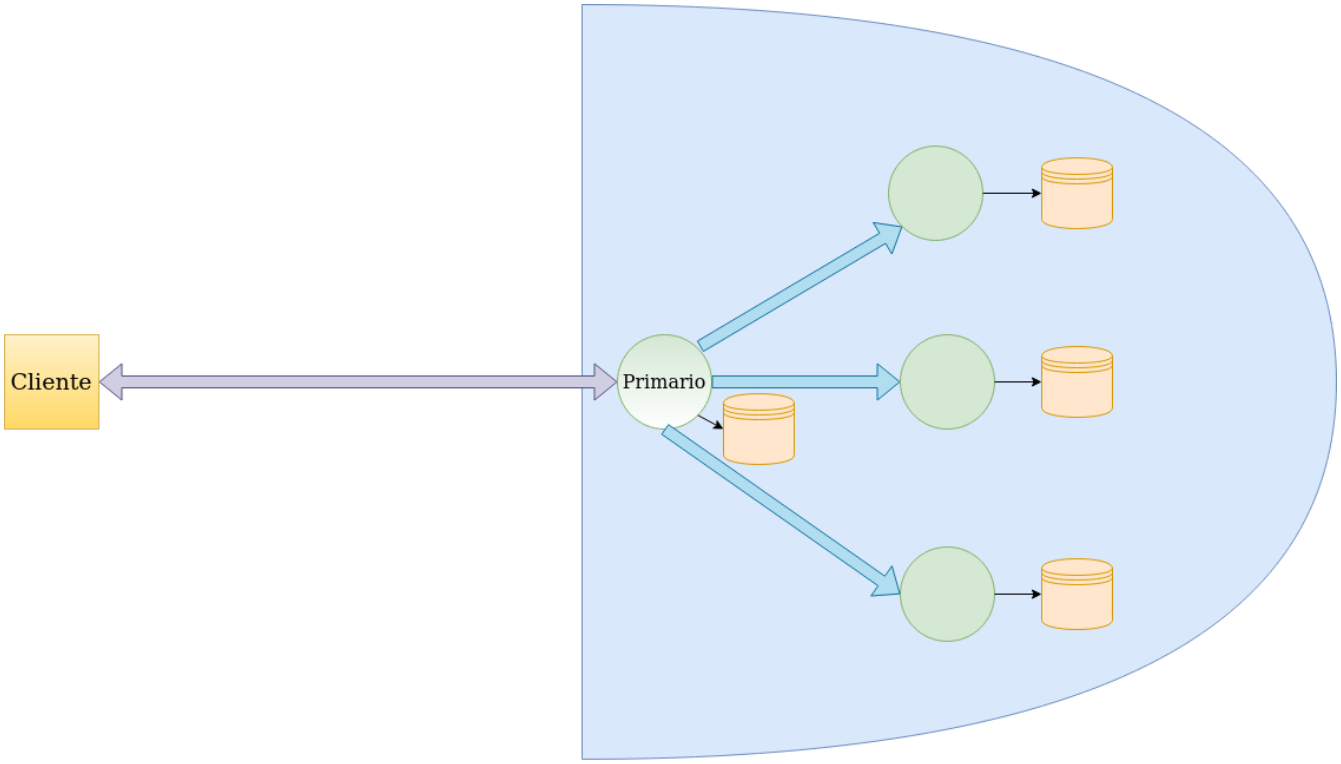


3) Se PN não receber uma mensagem "I´m alive" após expirar um temporizador, torna-se o primário se for o primeiro na lista:

- Avisa os clientes
- Começa a processar os pedidos



4 Arquitetura



5 Servidor

5.1 Base de Dados HSQLDB

Um dos requisitos do projeto era a utilização de um motor de base de dados em memória o HSQLDB, escrito totalmente em Java e que se destaca pelo baixo uso de recursos e pelo seu desempenho.

Apesar de tudo, a grande vantagem do HSQLDB, em termos deste projeto, é a facilidade que apresenta para fazer dump dos dados para backups.

5.1.1 Transferência de estado

Assim, quando o é necessário enviar o estado atual da aplicação para um novo membro, é feito um snapshot da base de dados, este snapshot consiste em fazer um backup dos dados da aplicação do HSQLDB e da tabela incremental de índices para um ficheiro.

Este ficheiro é então lido pela aplicação e enviado para o destino como byte[], destino esse onde depois será feito o unsnap dos dados, o commando de backup do HSQLDB com as flag –extract e –overwrite.

5.2 Spread Toolkit

O Spread Toolkit é a ferramenta usada para controlar o grupo de servidores que compõe o serviço, sendo o responsável pelo gestão e transmissão de mensagens entre eles. Também é através do Spread que é possível gerir as mudanças nos grupo, como a entrada ou saída de nodos, e lidar com problemas na rede, como, por exemplo, partições no grupo.

5.2.1 Adição de um membro

Quando um novo membro se junta ao grupo, adiciona todos os outros elementos do grupo à lista de próximos nodos primários.

Para além disso, o servidor primário, ao saber que um novo membro se juntou ao grupo, executa então então os passado discutidos anteriormente sobre a transferência de estado. Este executa então o snapshot da base de dados e envio esses dados ao novo nodo.

5.2.2 Remoção de um membro

Quando um membro é removido, todos os membros recebem um mensagem de notificação desse evento e imediatamente procedem a remover esse nodo caso esteja na sua lista de candidatos a servidores primários.

Caso o servidor que saiu foi o primário, todos os outros nodos verificam a lista de próximos servidores primários, e quem for o próximo na lista, automaticamente nomeia-se primário e envia uma mensagem de aviso aos outros nodos.

5.2.3 Partições do grupo

Quando ocorre uma partição do grupo, ao receber o evento do Spread, todos os nodos verificam quais os nodos que pertencem ao seu novo grupo, caso não estejam no grupo do servidor primário, eles acabam o seu processamento.

Os servidores restantes mantêm-se ligados. O servidor primário constrói e envia a todos os restantes nodos uma lista de próximos servidores primários construída para manter o flow explicado anteriormente.

5.3 Reencaminhamento de pedidos

Quando o servidor primário recebe de um cliente um pedido que envolve fazer escritas na base de dados, ele reenvia este pedido para os restantes nodos, todos eles executam o pedido normalmente para que o estado entre nodos se possa manter coerente. Para isso é invocado nos controladores o método usado para tratar o pedido como se de um cliente normal viess (handleMessage).

5.4 Heart Beat

Tal como mencionado no protocolo de replicação passiva, a cada X segundo (1 segundo neste caso), o servidor primário envia um mensagem para os outros nodos a indicar que se encontrar vivo e a funcionar corretamente.

Os outros nodos, ao receberem estas mensagens continuam em stand-by à espera de pedidos para atualizar o seu estado,

Caso o servidor primário não envie estas mensagens dentro di período P de timeout (5 segundo neste caso), o servidor que se encontra em primeiro lugar na lista de próximos primários, toma conta da execução e avisa os outros servidores e os clientes.

6 Cliente

O cliente tem conhecimento à priori de quais os endereços onde poderão estar servidores a correr. Quando o cliente se inicia ele tenta criar ligações a todos eles, o servidor primário enviar-lhe-á uma mensagem a informar que é para ele que deverão ser feitos todos os pedidos e, a partir daí, o cliente só comunica com ele. Quando outro servidor se torna primário, deverá informar o cliente que passará a enviar as mensagens para esse.

A implementação do cliente é muito simples, existe um módulo responsável por enviar mensagens aos servidores e ficar à escuta de respostas, que redireciona para os outros módulos os resultados já formatados pelo servidor.

Para além disso, o cliente possui interfaces com todas as operações disponíveis no servidor, de maneira muito simples e a imitar classes de Java sequencial normal, que ocultam os vários servidores distribuídos e a replicação. As implementações destas interfaces são responsáveis pelas chamadas ao módulo de conexão ao servidor das mensagens com o pedido a ser processado.

A interface com o utilizar é feita através da biblioteca Spring Shell, na qual criamos alguns comandos que quando invocados executam os métodos das interfaces.

```
public class ClientCommand implements CommandMarker {
...
    @CliCommand(value = {"sc"}, help = "Show a client by id")
    public Client showClient(@CliOption(key = {"", "id"}) long id) {
        Clients clients = new ClientsImpl();
        return clients.show(id);
    }
```

```
...
}

public interface Clients {
    ...
    Client show(long clientId);
    ...
}

public class ClientsImpl implements Clients {
    ...
    public Client show(long clientId) {
        byte[] data = Connection.sendMessage("client;show-" + clientId);
        market.models.Client client = SerializationUtils.deserialize(data);
        return new ClientImpl(client.getId(), client.getName());
    }
    ...
}
```

6.1 Clientes

6.1.1 Criar

```
spring-shell>cc --name joao
   ClientImpl:
        -> Id: 21
        -> Name: joao
```

6.1.2 Indexar

```
spring-shell>ic
   ClientImpl:
        -> Id: 21
        -> Name: joao

   ClientImpl:
        -> Id: 22
        -> Name: eduardo

   ClientImpl:
        -> Id: 23
        -> Name: jose
```

6.1.3 Mostrar

```
spring-shell>sc 21
   ClientImpl:
        -> Id: 21
        -> Name: joao
```

6.2 Produtos

6.2.1 Indexar

```
spring-shell>ip
   ProductImpl:
        -> Id: 1
        -> Name: Product 1
        -> Price: 1
        -> Stock: 1

   ProductImpl:
        -> Id: 2
        -> Name: Product 2
        -> Price: 2
        -> Stock: 2

   ProductImpl:
        -> Id: 3
        -> Name: Product 3
        -> Price: 3
        -> Stock: 3
```

6.2.2 Mostrar

```
spring-shell>sp 13
   ProductImpl:
        -> Id: 13
        -> Name: Product 13
        -> Price: 13
        -> Stock: 3
```

6.3 Encomendas

```
spring-shell>
spring-shell>
spring-shell># criar encomenda
spring-shell>co --clientId 21
OrderImpl:
   -> Id: 24
   -> Completed: false
   -> Client:
   ClientImpl:
       -> Id: 21
       -> Name: joao

   -> Products:
[]

spring-shell>
spring-shell># adicionar produto
spring-shell>ap --productId 13 --orderId 24
OrderImpl:
   -> Id: 24
   -> Completed: false
   -> Client:
   ClientImpl:
       -> Id: 21
       -> Name: joao

   -> Products:
[   ProductImpl:
       -> Id: 13
       -> Name: Product 13
       -> Price: 13
       -> Stock: 3
]

spring-shell>
spring-shell># confirmar encomenda
spring-shell>cfo 24
OrderImpl:
   -> Id: 24
   -> Completed: true
   -> Client:
   ClientImpl:
       -> Id: 21
       -> Name: joao

   -> Products:
[   ProductImpl:
       -> Id: 13
       -> Name: Product 13
       -> Price: 13
       -> Stock: 3
]

spring-shell>
```

7 Conclusão

Após terminar a implementação, o grupo pensa que todo o projeto foi concluído com sucesso, uma vez que todos os requisitos foram implementados e ainda conseguimos produzir varias valorizações(1,3,4,6).

Este projecto serviu também para aprofundar todo o conhecimento sobre os temas leccionados na Unidade Curricular e perceber a utilidade dos mecanismo e técnicas aprendidas ao longo do semestre. No entanto, também nos apercebemos de todos os problemas e dificuldades na implementação dessas técnicas e mecanismos.

Uma das primeiras dificuldades que enfrentamos esteve relacionada com a transferência de estado, foi preciso uma investigação profunda na documentação do HSQLDB para conseguirmos perceber de que modo era possível gerar um ficheiro com o dump da base de dados, que até era fácil de implementar, mas sobretudo no que dizia respeito à importação dos dados guardados noutra base de dados, este foi um processo demorado, que durante muito tempo não funcionava correctamente e que dava vários erros nos dados carregados.

Outra grande dificuldade, foi em termos da análise do sistema, para tentarmos perceber ao máximo possíveis faltas do mesmo, era essencial esta avaliação para conseguirmos implementar metodologias capazes de lidar com elas e assegurar que o serviço mantinha a melhor robustez possível e tentar que para o cliente esta faltas fossem transparentes.

Em suma, houve bastantes obstáculos, mas foram todos ultrapassados e todas as funcionalidades foram implementadas. Foi um projecto que deu bastante trabalho, tanto na parte do desenvolvimento da arquitectura, como na sua execução, contudo ajudou-nos a consolidar todo o conhecimento leccionado.