Instituto Superior de Engenharia de Lisboa
Área Departamental de Engenharia Eletrónica, Telecomunicações e Computadores (ADEETC)
Mestrado em Engenharia Informática e Computadores

Infraestruturas de Sistemas Distribuídos (IESD)

Semestre de verão 2021/2022

Professor: Eng.º Luís Osório

Relatório do Trabalho Prático 1

Elaborado por:
Tiago Conceição, Nº47611
Ricardo Candeias, Nº42087
André Mendes, Nº30569
Grupo 08

Índice

1.Introdução

1.1 Apresentação sumária do problema

Pretende-se implementar/concretizar um cenário de validação de conceitos de coordenação no acesso concorrente a elementos serviço, conforme proposto na figura 1 do enunciado deste trabalho.

1.2 Apresentação sumária da abordagem

O foco principal da abordagem é a discussão e o desenvolvimento de estratégias tendo em conta a coordenação de acesso concorrente a elementos serviço distribuídos.

1.3 Estrutura do documento

Este documento é estruturado em 4 tópicos: Introdução; Estado do Conhecimento e Análise e Discussão do Problema; Demonstrador Centrado na Coordenação; e Conclusões.

No tópico da Introdução, será apresentado o problema, a abordagem elaborada, e a estrutura do relatório.

No tópico Estado do Conhecimento e Análise e Discussão do Problema, será descrito o desenvolvimento de sistemas informáticos na base de elementos *Service*[1]. Será enunciada e justificada a coordenação de transações distribuídas. Por último, serão descritos os quadros tecnológicos utilizados.

No tópico Demonstrador Centrado na Coordenação, será descrita a forma como foi desenvolvido o demonstrador, tendo em conta os desafios da **estratégia de implementação do serviço de coordenação de transações e da concorrência.**

Por fim, no tópico Conclusões, será apresentado de forma resumida o que foi discutido e realizado, e serão enunciadas as dificuldades e aspetos a melhorar neste trabalho prático.

2. Estado do Conhecimento e Análise e Discussão do Problema

2.1 Desenvolvimento de sistemas informáticos na base de elementos Service (SOA)

O desenvolvimento deste sistema informático na base de elementos Service (SOA - Arquitectura Orientada a Serviços)[2], permitiu-nos abordar cada elemento do sistema como sendo um serviço.

Concretização de uma Arquitectura Orientada a Serviços:
Segundo o modelo ISoS, é um sistema informático (ISystem), composto por CES (Cooperation Enabled Services), e cada CES irá ter elementos que se designam por Service.
Entende-se por Serviço, uma abstração que permite a qualquer dispositivo ligar-se a ele.

Tendo em conta este modelo, foi desenvolvido um sistema informático composto por 3 CES: CesVector, CesTC (CesTransanctionCoordination) e CesRegistry. O CES CesVector irá conter 2 serviços, estes responsáveis pelo servidor vector (SerVector) e o respetivo cliente (SerVectorCli). O CES CesTC é constituído por 2 serviços: um serviço destinado a coordenar as transações (SerTM) e um segundo serviço responsável pela coordenação do acesso a recursos partilhados, através de uma abordagem 2Phase-Lock. Por último, o CesRegistry contém um único serviço, SerRegistry que é destino a possibilitar transparência à localização para os restantes serviços do sistema informático.

2.2 Coordenação de transações distribuídas

Identificamos 2 grandes sub-problemas nas propriedades ACID[3], no contexto da coordenação transacional: o problema da atomicidade e o problema da concorrência.

Consideramos o conceito de transação num contexto de serviços – entidades computacionais autónomas – em que, cada serviço invoca um conjunto de operações que têm de ser executadas de forma atómica. Temos ainda o problema acrescido de serem transações distribuídas.
Identificámos a necessidade de ter um mediador/intermediário, o qual será (também) um *Service*, com a responsabilidade de coordenar as transações.

Quando um serviço recebe (do serviço cliente) uma operação no contexto de uma transação, ele tem de se registar no "mediador" como participante. Esse mediador será o **Transaction Manager**.

O **Transaction Manager (TM)** implementa o protocolo **Two-Phase Commit**[4]. Este modelo funciona como um processo de votação, e consiste, como o próprio nome diz, em executar o processo de commit de uma transação em duas fases:

1ª - fase de *Prepare*, na qual o TM vai percorrer todos os serviços que se registaram como participantes naquela transação, e vai "avisá-los" que se devem "preparar"

para o Commit. De certa forma, é uma maneira do mediador perguntar aos serviços: “Correu tudo bem? Preparar para o Commit.”
Todos aqueles serviços vão responder/votar.

2ª – fase de *Commit*. Se todos os serviços responderem *Ready-to-commit*, então isso significa que correu tudo bem, e o TM vai percorrer novamente os serviços para dar ordem de Commit.
Se algum serviço responder *abort*, ou nem sequer responder, então a transação deve ser abortada. O TM vai percorrer novamente os serviços para dar ordem de Rollback à transação.

As transações são identificadas por um ID único, gerido pelo TM, o qual é fornecido ao serviço cliente. O objetivo do ID único é identificar o contexto das operações.

O Transaction Manager vem resolver a necessidade de atomicidade – ou a transação é realizada do início até ao fim, ou não é realizada.

Existe a necessidade de coordenar a concorrência no acesso. Essencialmente, a estratégia consiste em serializar as transações. O algoritmo é o **TPLM (Two-Phase Lock Manager)**[5], e o seu funcionamento é explicado no tópico mais à frente.

2.3 Descrição de quadros tecnológicos utilizados

Utilizámos o quadro tecnológico gRPC[6], o qual é a implementação da Google do modelo RPC (Remote Procedure Call).
Desta forma, optamos assim por um modelo de comunicação directa, em alternativa ao modelo de comunicação indirecta – MOM (Message Oriented Midleware), na interação entre as entidades.

A utilização do gRPC permite que os serviços possam ser implementados em linguagens de programação diferentes, ou seja, o quadro tecnológico não é uma dependência para realizar uma integração entre serviços deste sistema informático, ou seja, numa abordagem Collaborative Mobility Services Provider (CMSP)[1], serviços integrantes noutros sistemas informáticos, apenas têm de conhecer os contratos de comunicação protobuf, que poderão estar acessíveis num repositório remoto. E conhecer o endereço IP e o porto do serviço de diretoria (SerRegistry) já enunciado no ponto 2.1.

3. Demonstrador Centrado na Coordenação

3.1 Descrição do demonstrador

Implementou-se um Serviço adicional (SerRegistry) com a função de registar a participação dos elementos do sistema distribuído (TM, TPLM e Vector Services). Através deste serviço é possível adquirir ou registar outros serviços de forma genérica e parametrizável.

Assim existe um certo nível de transparência à localização[7] pois cada serviço que tente aceder às funcionalidades do sistema, apenas precisa de comunicar com este serviço para que este lhe forneça a informação de comunicação com os outros serviços do sistema distribuído.

Contudo, este serviço facilita uma abordagem Collaborative Mobility Services Provider (CMSP), de modo que para uma integração entre um serviço do sistema e um serviço externo fosse realizada com sucesso apenas seria necessário que o componente externo em questão tivesse posse do contrato de comunicação, do endereço IP e porto de uma instância do servidor de diretoria.

Conceção da comunicação entre serviços:

A comunicação entre os serviços TM, VectorClient e Vector seguem os standards referidos no de modelo X/OPEN[8], no qual foram implementadas as especificações TX e XA.

A especificação TX foi concretizada na comunicação entre o serviço VectorClient e o serviço TM e foram implementados os métodos: tx_begin, tx_commit e tx_rollback.

A especificação XA foi associada na comunicação entre o serviço TM e o serviço Vector no qual foram implementados os métodos: ax_reg, xa_prepare, xa_commit e xa_rollback.

Estratégia de implementação do serviço de coordenação de transações e da concorrência:

A responsabilidade de coordenação de transações e da concorrência foi estruturada enquanto abstração CES (Cooperation Enabled Services), composto por dois elementos serviço:

- **TM (Transaction Manager)** – implementação do conceito de transação – tem a responsabilidade de coordenação de transações distribuídas. Foi desenvolvido na base do modelo X/Open. Todos os serviços que são gestores de recursos (neste caso, o recurso é o vector), vão participar numa transação. Formalmente, no modelo X/Open, esses serviços gestores de recursos chamam-se **Resourse Manager (RM)**.

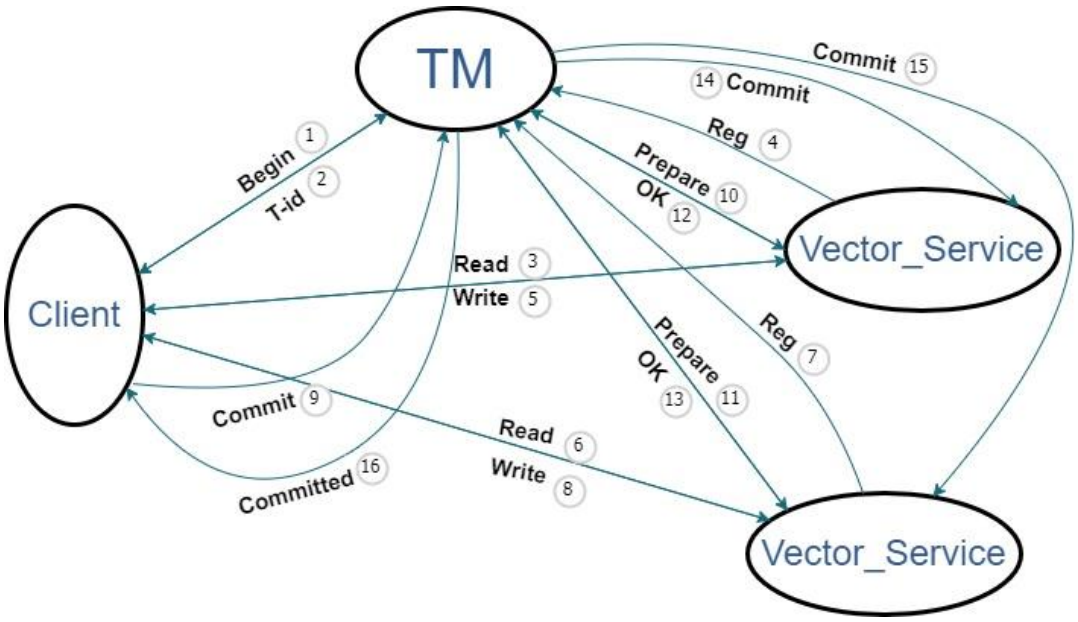

Figura 1 – esquema ilustrativo do algoritmo do TM

- **TPLM (Two-Phase Lock Manager)** – implementação da coordenação da concorrência – tem a responsabilidade de garantir o isolamento no acesso aos recursos.

  Foi tida em conta a importância da granularidade do *Lock[9]*, que deverá ser aquela que maximiza a concorrência. Caso contrário, vamos introduzir latência. Desta forma, nós definimos que o Lock bloqueia apenas uma posição no recurso (vector), e não o recurso inteiro.
  Num cenário de escalabilidade, no qual sejam instanciados um número elevado de serviços cliente e vector, podendo até o recurso (vector) ter uma dimensão elevada, a granularidade do *Lock* é extremamente importante.
  Por exemplo: se tivermos um recurso com dimensão elevada, e um (serviço) cliente pretender aceder apenas a duas ou três posições, então não queremos que o recurso fique todo bloqueado para aquele cliente, deixando todos os outros clientes em espera, quando provavelmente as posições pretendidas pelos outros clientes nem são as mesmas.

  O serviço cliente, após obter o ID da transação, vai solicitar ao TPLM os Locks que necessita para a transação. Os Locks são caracterizados/identificados da seguinte forma:
  - [Resourse Manager].[posição específica no vector]
  - Tipo de Lock:
    - Lock para leitura (lock partilhado)
    - Lock para escrita (lock exclusivo)

  Foi usado o modelo **Two-Phase Locking**, cujo princípio é este: uma transação deve obter todos os Locks antes de liberá-los.
  Existem, portanto, duas fases bem definidas:

  1ª fase – (fase ascendente) aquisição de todos os Locks necessários à transação. O TPLM apenas atribui os Locks ao (serviço) cliente quando todos estiverem

disponíveis. Se algum Lock não estiver disponível, então o cliente ficará em espera.

2ª fase – libertação de todos os Locks. O serviço cliente liberta todos os Locks após concluída a transação.

Com esta estratégia, conseguimos evitar o problema do *deadlock[10]*, uma vez que não existe aquisição de Locks durante as operações da transação. Por outro lado, esta estratégia não favorece a concorrência, porque há Locks que a determinado momento já não são necessários, mas só serão libertados no final.

Implementação:

Em relação ao modelo de two-phase locking implementado criou-se um elemento que representa o estado de disponibilidade, quer seja para escrita quer seja para leitura, de cada elemento individual de todos os Vector Services. O acesso à estrutura de dados que contém esta informação é mediada por um lock, ou seja, apenas um pedido de cada vez poderá verificar a disponibilidade de locks.

Um pedido de lock de um determinado elemento para leitura, irá deixar outras leituras disponíveis, mas impedirá escritas para o mesmo elemento.

Um pedido de lock de um determinado elemento para escrita, impedirá quaisquer leituras ou escritas adicionais para o mesmo elemento.

Qualquer pedido que não lhe seja concedido os locks aos elementos que pretendia ficará à espera sobre o lock. Quando uma transação libertar os locks dos elementos que precisou para a sua operação, todos os outros pedidos à espera de locks serão notificados e tentaram novamente obter os locks para os seus elementos.

[Qualquer espera sobre qualquer lock neste trabalho tem um timeout. Podendo falhar desse modo.]

Validação:

A validação consiste em garantir o invariante, conforme explicado no enunciado do trabalho.
Esta operação é totalmente independente ao funcionamento do sistema. A verificação do invariante só é feita quando todas as transações estão *commited[11]*, de forma a ser garantida consistência no resultado.

4. Conclusões

4.1 Resumo do que foi discutido e realizado

Os conceitos e as estratégias mais importantes que foram analisados, discutidos, e implementados, estiveram relacionados com a coordenação de transações distribuídas e da concorrência.

Conforme foi discutido anteriormente, identificamos 2 grandes sub-problemas nas propriedades ACID, no contexto da coordenação transacional: o problema da atomicidade e o problema da concorrência.

Foi também constatada a importância da granularidade do *Lock*, que deverá ser aquela que maximiza a concorrência. Caso contrário, estaremos a introduzir latência, especialmente em recursos de elevada dimensão.

4.2 Dificuldades e aspetos a melhorar

Ficaram claras as dificuldades e os desafios no desenvolvimento de sistemas informáticos na base de elementos (Service) autónomos, com interdependências.

Foram discutidos cenários de escalabilidade, com um número elevado de serviços cliente e vector.
Com um número elevado de serviços vector, o modelo de comunicação teria de ser repensado para uma abordagem mais dinâmica.
Com um número elevado de serviços cliente, iriamos ter um aumento da latência como consequência direta do aumento da concorrência. Lembremos que as transações correm de forma serializada.
Talvez fosse necessário distribuir a responsabilidade do TM e do TPLM, para distribuição de carga. Nesse caso, teríamos de assegurar o sincronismo em relação à responsabilidade desses sistemas.

Como aspeto a melhorar, deveremos implementar alguma estratégia de tolerância a falhas para a responsabilidade de coordenação, uma que se trata de um elemento crítico do sistema informático. Para tal, poderemos considerar o sistema ou elemento de sistema Zookeeper[12] (projeto Apache Zookeeper) estudado. O Zookeeper é, basicamente, um servidor, que permite manter uma árvore de nós – raiz e o conceito de znode – com uma API simples baseada em diretorias. Esta ferramenta permite lidar com a tolerância a falhas de forma simples e robusta (e económica).

5. Referências

[1] - **On Reliable Collaborative Mobility Services: 19th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2018, Cardiff, UK, September 17-19, 2018, Proceedings**

[2] – What is Service-Oriented Architecture?, 2019, Software Development Community

[3] – ACID, 2021, Wikipedia

[4] – Two-phase commit protocol, 2022 , Wikipedia

[5] – Two-phase locking, 2021, Wikipedia

[6] - gRPC, 2022

[7] – Location transparency, 2021, Wikipedia

[8] – X/Open, 2021, IBM

[9] – Lock, En-Academic

[10] – DeadLock, En-Academic

[11] – Commit, 2011, Techopedia

[12] - ZooKeeper