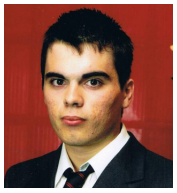
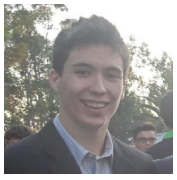
Relatório do Projeto – Parte 2

Sistemas Distribuídos

Grupo 22 – Prof. Miguel Pardal – 2ª feira – 11h00



Miguel Santos
75551



Tiago Rechau
76231

Tomás Alves
75541

Requisito SD-Store A – *Quorum consensus* em Leituras/Escritas

Atulização do domínio e criação de handler

Este requisito implica implementar o protocol *quorum consensus* nas funções *load* e *store* do cliente do SD-Store. Dessa forma, criou-se no domínio tanto do servidor quanto do cliente a classe `Tag`, a qual tem como função identificar o cliente e o número de sequência da ação, dando consistência temporal à execução do programa e permitindo aplicar o dito protocolo.

De forma a não alterar o contrato WSDL, criou-se um *handler* denominado `TagHandler` que tem como função adicionar este objeto `Tag` a pedidos ou respostas que circulem entre os intervenientes da execução.

Implementação do protocolo

Quanto à implementação, foi necessário fazer uma modificação no construtor do Front-End para poder receber como *input* valores *threshold* para leituras e escritas. Este construtor trata também de encontrar todos os servidores de replica que existem pelo UDDI.

Por opção, caso os valores de *threshold* que o construtor recebe, caso qualquer um deles seja 0, ambos terão um valor igual ao do *quorum*, isto é, o valor que é a minima maioria, para assegurar o funcionamento do protocolo. No caso de qualquer dos valores ser maior que o número de servidores existentes, esse valor para a ser igual ao número de réplicas existentes.

É de notar que o desenvolvimento do protocolo foi efetuado com mensagens assíncronas e omissão de mensagens. Assim, não é necessário esperar pelas respostas sequenciais dos servidores, apenas por um número mínimo de mensagens válidas para proceder com a correta execução comportamental do programa.

Para além disso, e tendo em conta que a localização de um gestor de réplica pode variar ao longo do tempo, dá-se primeiramente uma pesquisa no servidor UDDI para actualizar os endereços dos mesmos antes de qualquer das funções de leitura ou escrita ser efetuada.

A implementação correta do protocolo, cada vez que existe uma escrita, essa escrita tem de ser feita numa maioria de servidores e com uma `Tag` mais atualizada de entre todas. Assim sendo, primeiro efetua-se um envio de mensagens de leitura para os servidores de forma a obter um conjunto de `Tags` associadas ao documento alvo. De seguida, procura-se a maior `Tag` de entre as obtidas e, após adquiri-la, cria-se um novo objeto `Tag`, sendo que este é igual à obtida anteriormente, mas com um número de sequência incrementado. Depois, envia-se novamente a todos os servidores uma mensagem de escrita com o conteúdo pretendido e a nova `Tag` e espera-se até que o *threshold* de escritas seja atingido pelo número de respostas recebidas dos gestores de réplicas.

Já no que toca à leitura, tal como numa escrita, faz-se um pedido a cada réplica para devolver o seu conteúdo associado ao documento pretendido e a `Tag` do mesmo. Após obtidas respostas suficientes, isto é, atingido o *threshold* de leituras, é efetuado o procedimento de *writeback*, o qual implica percorrer a lista de respostas chegadas e ver quais é que têm uma `Tag` diferente da `Tag` com o número de sequência mais elevado. A todos os servidores que enviaram as mensagens com as `Tags` diferentes, envia-se uma mensagem de escrita com o conteúdo mais atualizado do documento que está a ser tratado e espera-se por um número de ack que seja igual ao *threshold* de *write*.

Com este protocolo de replicação ativa efetivamente implementado, é possível prestar um melhor serviço ao cliente com melhores disponibilidade, desempenho e escalabilidade, dado a política de tolerância de faltas.

Requisito SD-ID B – Documentos seguros

Atulização do domínio – Autenticidade e integridade

Duas das características de documentos seguros são a autenticidade e a integridade do mesmo, o que implica que ele não foi manipulado por entidades externas. Dessa forma, criou-se a classe `DocumentAuthentication`, a qual trata de garantir estes dois factores.

O algoritmo escolhido foi o MAC. O algoritmo que trata da criação do *digest* é um `SHA-256`, o qual foi escolhido pelo facto de o `SHA-2` ser mais forte e adequado para aplicações sensíveis à segurança como a criação de resumos.

A chave que é usada para cifrar o *digest* é do tipo `DES`, com modo de operação `ECB` e `PKCS5Padding`. A junção destes três atributos resulta numa chave que garante ambas as características pretendidas, dado que apenas a classe é que conhece a chave e é a única com responsabilidade para tratar de documentos e respetivos resumos.

Criação de handlers – Confidencialidade

Outra característica de documentos seguros é a confidencialidade dos mesmos. Esta foi implementada pela criação de dois *handlers*: `DocOwnerHandler` e `CypherHandler`.

A função do primeiro é simples: em qualquer mensagem que um gestor de réplicas enviar do tipo resposta a uma leitura, esta resposta deve incluir também quem é o dono do documento. Esta informação é importante e vai ser necessária para associar a um utilizador a sua chave, tal como vai ser explicado mais à frente. Este *handler* está presente no conjunto de *handlers* dos gestores de réplicas.

Já do lado do cliente do SD-Store, este tem o segundo *handler* que trata de cifrar todas as mensagens que chegam e são enviadas para os gestores de réplicas. De forma a aumentar a transparência da encriptação para o utilizador, esta classe trata igualmente da manutenção de chaves, ou seja, guarda uma instância do tipo `Map` onde associa a cada utilizador uma chave única. Assim, é também possível evitar que um utilizador tenha acesso a documentos de um outro utilizador. Caso o utilizador ainda não exista no mapa de chaves do *handler*, ele trata de o colocar lá. Para além disso, este mapa é guardado de forma persistente num ficheiro

Aquando da receção de uma mensagem, no caso de ser uma escrita nos servidores, o *handler* trata de ir buscar à mensagem SOAP o utilizador que está a efetuar a ação e depois a chave deste para codificar o seu documento. No caso de ser uma leitura, e como o contrato WSDL não permite que seja adicionado novo conteúdo à mensagem, o `DocOwnerHandler` toma a sua importância e permite que o `CypherHandler` possa obter o nome do utilizador em questão de forma a fazer o procedimento inverso e decifrar o conteúdo do documento em causa.

Opções de implementação

Dado que o `BubbleDocs` devia permitir a existência de múltiplos clientes, a manutenção das chaves tornou-se um problema dado que era necessário manter uma chave constante para o mesmo utilizador caso ele acedesse a partir de qualquer uma das aplicações cliente. Esta chave constante permite a que o utlizador, dado que na criação de um novo SD-Store Cliente é também criado um novo `CypherHandler,` mantenha uma chave única e consiga aceder a todos os seus documentos, ao invés da cifra ser diferente de SD-Store Cliente para SD-Store Cliente.

Ao guardar num ficheiro a que qualquer instância de um SD-Store Cliente tem acesso, conseguimos assegurar a consistência das operações do utilizador, apesar do custo de *performance* que é estar a ler um ficheiro e processá-lo.