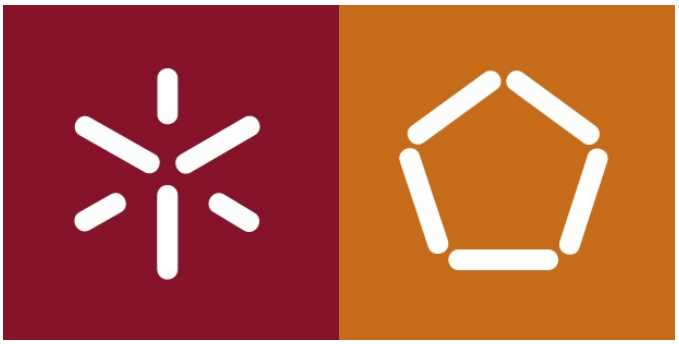
UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA



Alocação de Servidores na Nuvem

SISTEMAS DISTRIBUIDOS


Davide Matos A80970


Francisco Freitas A81580


Maria Dias A81611


Pedro Freitas A80975

7 de Janeiro de 2019

Conteúdo

INTRODUÇÃO

No âmbito da cadeira de Sistemas Distribuıdos foi-nos proposto a elaboração de um trabalho prático cujo o tema é Alocação de servidores na nuvem. Desta forma, este projeto tem como objetivo desenvolver um serviço de alocação de servidores na nuvem e de contabilização do custo incorrido pelos utilizadores, podendo a reserva de um servidor ser feita a pedido ou em leilão. Na reserva a pedido, um servidor fica atribuído até ser libertado pelo utilizador,já na reserva em leilão, um utilizador propõe o preço que está disposto a pagar por o servidor de determinado tipo.A este só lhe é atribuído uma reserva do servidor quando o preço horário proposto for o maior entre os licitantes para esse tipo de servidor e para além disso,a reserva em leilão pode ser cancelada pela nuvem de forma a satisfazer uma reserva a pedido.

O nosso projeto será implementado usando um modelo cliente-servidor escrito em JAVA, no qual os utilizadores podem interagir com a plataforma intermediados por um servidor multi-threaded, e recorrendo a comunicação via sockets TCP.

IMPLEMENTAÇÃO

Utilizador

A classe Utilizador é responsável por representar um utilizador da nossa aplicação que pretenda reservar e licitar servidores.Para além disso pode consultar os seus servidores e libertá-los.

BufferServidor

A classe BufferServidor faz a comunicaçao entre o servidor e o cliente, enviando-lhes as mensagens baseadas nas ações que o mesmo despoleta.

Gestao

A classe Gestao é responsável por armazenar toda a informação presente na aplicação num determinado momento. Nesta classe estão armazenados todos os servidores, utilizadores e mensagens. Por último também tem armazena o leilão.

Leilao

A classe Leilao representa um leilão a decorrer. Nesta classe está guardada a lista de todas as licitações.

Licitacao

A classe Licitacao é aquela que sustenta a funcionalidade do leilão. Esta classe guarda informação como o nome do utilizador que a faz, o nome do tipo de servidor o qual pretende licitar e o valor da licitação.

Menu

Esta classe é a responsável por apresentar os diferentes menus consoante as ações desejadas pelo utilizador.

Servidor

Sendo este um problema de alocação de servidores na nuvem, esta classe é aquela que representa um servidor em si. Esta classe guarda a informação respetiva a um servidor: id, o tipo , o preço, se está disponível ou não, se está reservado ou não, o valor ao qual foi leiloado, o nome do utilizador a qual o servidor pertence.

ThreadClienteInput

Esta classe é a responsável pela transmissão da informação entre cada utilizador e o server, noemadamente, as ações que cada um deseja fazer.

ThreadClienteOutput

É responsável por receber do server, a informação direcionada para o utilizador.

ThreadServidorRead

Esta classe é responsável por intrepretar a informação recebida do utilizador, fazendo a respetiva ação.

ThreadServidorWrite

Esta classe é responsável por transmitir o resultado para o utilizador após ter realizado a ação desejado por este.

Cliente

A classe Cliente representa o executável que qualquer utilizador da aplicação irá usar para tirar proveito das suas funcionalidades.

Server

Esta é a classe que representa o executável que mantém o servidor ligado e que permite aos utilizadores desfrutarem da aplicação.

Exceptions

Como existem várias exceções possíveis, ou seja, situações indesejadas à nossa aplicação, foi necessária a criação de classes Exceptions:

1 LoginInvalidoException

2 RegistoInvalidoException

3 ServidorIndisponivelException

PROTOCOLO CLIENTE-SERVIDOR

Para ir de encontro ao que era pedido no enunciado, implementamos um servidor e um cliente que comunicam via sockets(TCP). Quando o Servidor aceita um Cliente, são criadas quatro threads. Do lado do servidor são criadas duas threads,uma que é responsável por ler do socket do cliente e outra que lhe envia mensagens. No cliente podemos observar um comportamento análogo, onde uma das suas threads envia mensagens e outra recebe mensagens por parte do Servidor. As mensagens são orientadas à linha, o que significa que a comunicação é feita através de linhas de texto que equivalem às ditas mensagens.

Cada cliente tem a si associado uma instância da classe BufferServidor, que funciona como uma espécie de buffer, ou seja, envia para o cliente todas as mensagens que o servidor encaminha pela sua socket, servindo de meio de comunicação entre os dois. Do lado do

servidor esta classe desempenha também o papel de comunicação entre a sua thread de escrita e a sua thread de leitura. Para além disso, para que a ThreadWrite saiba quando enviar uma mensagem, esta possui uma Condition, que faz com que fique parada (await()), até que receba um sinal para acordar (signal()) por parte da ThreadRead, que quer enviar uma mensagem ao utilizador com que comunica.

Escolhemos esta forma de implementação para facilitar a notificação de vários utilizadores quando são afetados pela ação de outro cliente. Por exemplo, quando um utilizador é ultrapassado num leilão, esta BufferServidor facilita a passagem da mensagem para o utilizador afetado. Outro exemplo concreto que é possível observar neste projeto é que todos os utilizadores que licitam num leilão, mas não efetuam a licitação mais alta, recebem a mensagem que acabamos de referir.

CONTROLO DE CONCORÊNCIA

O principal objetivo deste trabalho é controlar o acesso ao conjunto de leilões e pedidos de reserva de servidores, onde existe partilha de memória por todas as threads do Servidor, de modo a que o programa consiga lidar com a informação de vários clientes em simultâneo. Para esse efeito, foram utilizados mecanismos de controlo de concorrência.

No código desenvolvido existem as classes Utilizador , BufferServidor, Gestor, Licitacao e Leilao, onde podemos verificar que existe partilha, por todas as threads do Servidor. O Gestor possui um conjunto de Servidores, um conjunto de Utilizadores e um conjunto de BufferServidor, o que torna possıvel haver comunicação entre os diferentes clientes. Para garantir que esta informação partilhada por todos esteja sempre atualizada quando é pretendida por cada utilizador, implementamos um sistema que faz com que só um utilizador possa aceder à informação partilhada por todos os utilizadores de cada vez, uma vez que esta se trata da secção crıtica do nosso código. Sendo assim, na classe Gestao, temos três locks, um para cada conjunto referido anteriormente, permitindo assim bloquear o acesso de vários utilizadores, enquanto um está a aceder aos objectos, e libertar o mesmo quando este acaba. Para além disso, sempre que um utilizador acede a um leilão, também bloqueamos o acesso aos outros utilizadores, no entanto, o lock é referente a um leilão concreto, permitindo assim que dois clientes possam aceder a dois leilões diferentes, e não ao mesmo leilão em simultâneo. O mesmo ocorre com a classe BufferServidor, que bloqueia o acesso simultâneo dos utilizadores, perimitindo que utilizadores adicionem uma ou um conjunto de mensagens em diferentes BufferServidor, mas não no mesmo, em simultâneo.

INTERFACE

Inicialmente, é apresentado ao cliente um menu que lhe permite registar-se na aplicação, iniciar sessão ou sair.

```
************* MENU ****************
* 1 - Iniciar Sessao              *
* 2 - Registar                    *
* 0 - Sair                        *
***********************************
```

Figura 1: Menu Inicial

Assim que o utilizador inicia sessão, surge o menu da área do cliente, no qual é possível licitar ou efetuar a reserva de um servidor, bem como consultar e libertar servidores em uso. Para além disso, o cliente pode consultar o valor da dívida acumulada até ao momento. Neste menu, o cliente também tem a opção de terminar sessão.

```
*********** ÁREA CLIENTE **********
* 1 - Licitar um Servidor         *
* 2 - Efetuar pedido de Servidor  *
* 3 - Servidores em uso           *
* 4 - Libertar Servidores         *
* 5 - Consultar valor em dívida   *
* 0 - Terminar Sessao             *
***********************************
```

Figura 2: Menu Área do Cliente

O catálogo de servidores é apresentado sempre que o cliente pretende reservar um servidor a pedido ou através de leilão.

```
******** Catálogo Servidores ******
* 1 - t3.micro                    *
* 2 - m5.large .                  *
* 0 - voltar                      *
***********************************
```

Figura 3: Catálogo de Servidores

CONCLUSÃO

Com a realização deste trabalho pudemos asim aplicar e consolidar os variados tópicos abordados durante este semestre.

Sistemas Distribuídos têm como objetivo a distribuição das várias tarefas, podendo assim executa-las de paralelamente e consequentemente reduzir o tempo de execução.

Assim de forma a ter um maior proveitamento, tentamos promover a concorrência usando para isso, a implementação de Threads. Além desta concorrência é necessário assegurar a fiabilidade dos dados ao longo da utilização da aplicação, tendo sido necessário aplicar exclusão mútua com os respetivos mecanismos(como locks e e variáveis de condição).

Por fim, após uma revisão ao projeto, achamos que conseguimos alacançar os objetivos a qual fomos propostos, quer ao nível das funcionalidades, quer ao nível da arquitetura da aplicação quer à implementação de um sistema dinâmico.