UNIVERSIDADE da MADEIRA

2023/2024

Programação Avançada – Projecto prático 1

1. Introdução:

Com este projecto espera-se que os alunos demonstrem o conhecimento adquirido no primeiro terço da disciplina de Programação Avançada. Assim, espera-se a entrega de um projecto em JAVA que resolva o problema descrito abaixo de uma forma paralela. De uma forma mais específica, espera-se a utilização de:

- *Threads*;
- Semáforos;
- Trancas;
- Padrões arquitecturais para o desenvolvimento paralelo.

2. Descrição do problema:

Num cenário real, o processamento de grandes volumes de dados, como imagens de alta resolução, pode ser bastante exigente em termos de recursos computacionais. A utilização de um sistema de processamento distribuído permite optimizar este processo, dividindo a carga por vários servidores de processamento. Neste trabalho, os grupos deverão implementar um sistema que simule este cenário.

O projecto proposto, doravante referido como "**pa-distributed-img**", tem como objectivo desenvolver um sistema distribuído para processamento de imagens, no qual múltiplos clientes e servidores de processamento interagem entre si, tal como mostrado na Figura 1. O ponto de partida para este projecto será o código partilhado no repositório da disciplina[1].

Quando um cliente recebe uma imagem que necessita de processamento, o primeiro passo é dividir essa imagem em várias partes iguais. O número de partes em que a imagem será dividida é determinado pelo utilizador e deverá ser especificado no ficheiro de configuração do projecto. Após a divisão de imagem, cada segmento resultante será enviado para um servidor diferente para ser processado. O total de servidores disponíveis para processamento também é uma informação que deverá ser configurada pelo utilizador e armazenada no ficheiro de configuração do projecto.

A atribuição das partes de imagem aos servidores deverá ter em conta a carga de trabalho actual de cada servidor. Isso significa que, antes de enviar uma parte de imagem para um servidor, cada cliente deverá avaliar a capacidade de processamento disponível em cada um servidor de processamento, com o objectivo de distribuir a carga de maneira equilibrada.

Para implementar o balanceamento de carga, será necessário utilizar um ficheiro denominado `load.info`, que regista o número de imagens que cada servidor de processamento está a gerir no momento. Apenas os servidores de processamento têm permissão para actualizar este ficheiro, reflectindo as alterações na sua carga de trabalho sempre que, entre outros, iniciam ou concluem o processamento de uma imagem. Por outro lado, os clientes apenas podem ler o conteúdo do `load.info` para determinar qual servidor que, no momento, deverá processar uma parte de imagem. Assim, quando um cliente precisa processar uma parte

[1] Note-se que terá de fazer *download* do repositório (ou seja, menu *Code* > *Download ZIP*). Não faça *clone* ou *fork* do mesmo.

de uma imagem, ele consulta este ficheiro e escolhe enviar essa parte de imagem para o servidor com a menor carga de trabalho registada, assegurando uma distribuição equilibrada do processamento e evitando a sobrecarga de qualquer servidor específico.

É fundamental prevenir a sobrecarga nos servidores de processamento. Desta forma, é necessário ter em conta que cada servidor de processamento tem uma capacidade máxima definida para processar partes de imagens em simultâneo. Esta limitação de capacidade deverá ser especificada no ficheiro de configuração do projecto (poderá ser igual para todos os servidores de processamento). Caso esta capacidade seja ultrapassada, as partes de imagens adicionais deverão ser colocadas numa fila de espera, aguardando pelo seu processamento. Mesmo para as imagens em fila de espera, o ficheiro `load.info` necessita de ser actualizado, reflectindo o acréscimo de carga no servidor correspondente, ao incrementar o número de imagens que cada servidor de processamento está a gerir no momento.

O código fornecido no repositório da disciplina implementa uma funcionalidade de processamento de imagens que remove a componente vermelha de uma imagem. No entanto, os alunos têm a liberdade de escolher e implementar um tipo diferente de processamento de imagens, caso prefiram. Adicionalmente, não foram estabelecidas restrições quanto aos formatos de imagem que o sistema deve suportar; os alunos podem decidir quais formatos de imagem o projecto irá processar.

Todas as configurações que o grupo considere necessárias para o projecto deverão ser lidas do ficheiro `project.config`, que deverá ser incluído no repositório do projecto. Toda a parametrização adicional que cada grupo ache necessária deverá ser adicionada a este ficheiro de configuração.

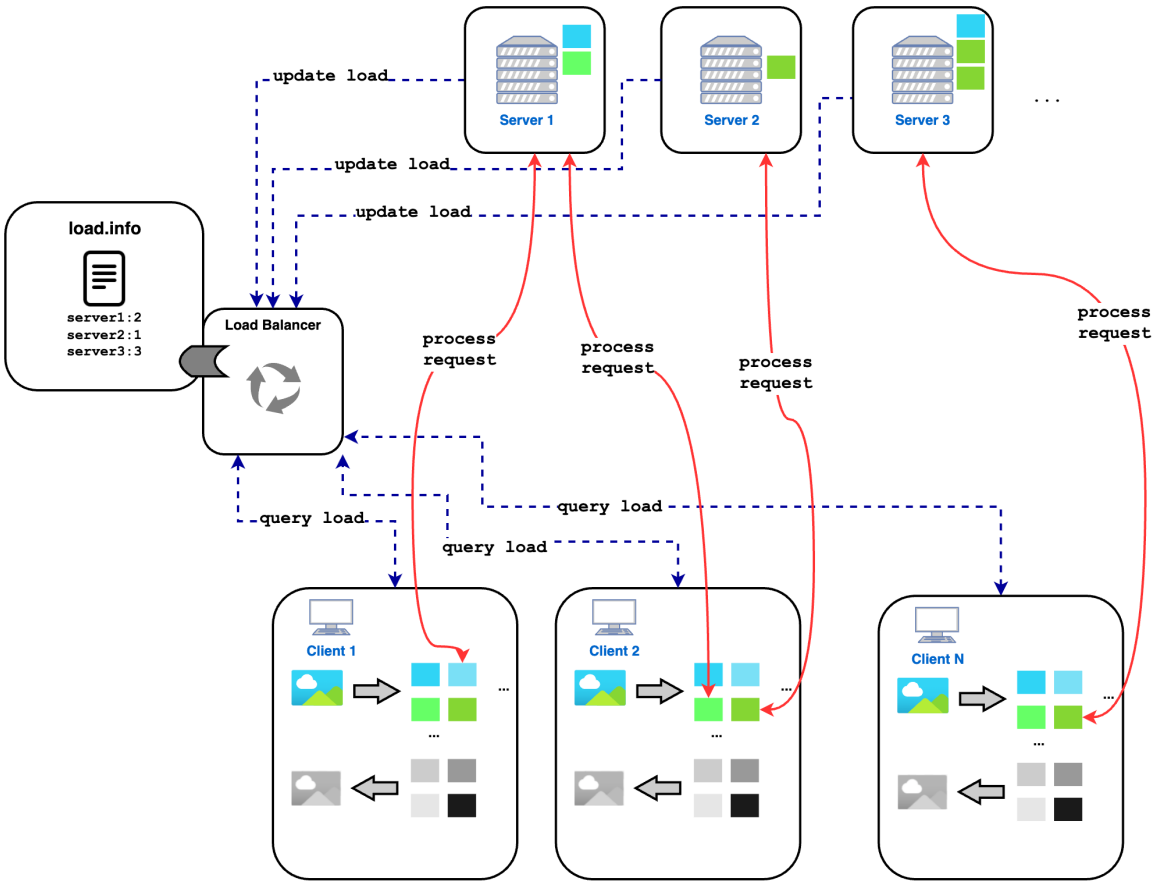

Figura 1: Arquitetura conceptual do projeto

Na prática, serão avaliados os seguintes requisitos de implementação. Estes requisitos terão de ser cumpridos para a aprovação do projecto:

2.1 Requisitos de implementação:

1. Terão de ser integrados **pelo menos 2 padrões de desenvolvimento** paralelo discutidos nas aulas para a resolução do problema acima.
2. Terão de ser utilizadas *threads*/*runnables*, trancas e semáforos.
3. É da responsabilidade de cada servidor de processamento a escrita do número de imagens que está a gerir no momento.
    a. O programa tem de garantir a integridade do ficheiro `load.info`. Apenas uma *thread* (quer seja dos clientes [leitura], quer dos servidores de processamento [escrita]) pode aceder (e escrever) ao mesmo de cada vez.
4. Quanto a capacidade máxima de processamento simultâneo for excedida, as partes de imagens subsequentes deverão entrar em fila de espera. Quando a capacidade ficar novamente disponível, as partes das imagens deverão ser processadas por ordem de chegada ao servidor de processamento (FIFO).
5. A comunicação entre os clientes e os servidores de processamento deverá ser feita usando *sockets* TCP/IP (código já disponibilizado).
6. Após o processamento de todas as partes, o sistema deverá automaticamente juntar partes processadas para recriar a imagem original na sua forma editada. A junção deverá ser feita de forma que a imagem final seja indistinguível da original, excepto pelas modificações resultantes do processamento. As imagens reconstituídas deverão ser armazenadas no directório denominado `results`. O nome de cada imagem processada deverá seguir o formato `<nome-antigo>_edited`, mantendo a sua extensão original (por exemplo, `.png`, `.jpg`, etc.).
7. Cada cliente deverá possuir uma interface de utilizador simples. Esta interface tem como objectivo principal facilitar duas acções essenciais: o envio de imagens para processamento e a visualização de mensagem resultantes da execução. Para o envio de imagens, a interface deverá permitir aos utilizadores indicar as imagens a serem processadas, através da especificação do caminho no sistema de ficheiros onde as imagens estão armazenadas.
8. A utilização do Git deverá observar as boas práticas apresentadas em aula, nomeadamente, o desenvolvimento colaborativo (por exemplo, através da divisão dos requisitos pelos elementos do grupo), os *commits* semânticos, e a criação e a fusão de *branches* (onde cada *branch* deverá possuir um nome apropriado).

2.2 Requisitos extra (2 valores):

1. O sistema deverá ser facilmente escalado, adicionando mais servidores de processamento sem necessidade de grandes alterações na arquitectura ou na configuração, e sem interrupções no sistema.
2. O sistema deverá possuir um mecanismos de tolerâncias a falhas, garantindo que, no caso de um servidor de processamento se tornar inacessível, as novas partes sejam automaticamente realocadas para outros servidores disponíveis sem interrupção significativa do serviço.

2.3 Requisitos de entrega:

1. O projecto deverá ser entregue até ao dia 23 de Março de 2024.
2. A entrega deverá ser realizada através do repositório da disciplina, através da criação de uma *branch* com o nome `final`.

3. Juntamente com a implementação, deverão ser entregues testes de integração e o relatório de *code coverage* gerado pela biblioteca JaCoCo (já incluída no `pom.xml` do projecto partilhado).
4. Projectos que não compilem, e cuja sua execução seja, assim, impossível não serão considerados para avaliação.
5. Todas as classes e os métodos do programa deverão apresentar documentação de suporte de acordo com as práticas de documentação JAVA (Javadoc) discutidas em aula.

De forma a garantir os requisitos n.º 2, 3 e 4, é obrigatório que os alunos implementem GitHub Actions para a execução dos testes de integração, criação do relatório de *code coverage* e criação da documentação de suporte. Poderão seguir os exemplos criados durante as aulas práticas.

3. Avaliação individual:

A contribuição individual de cada aluno será avaliada tendo em conta o histórico de actividade do mesmo no repositório do projecto. Antes de iniciar a implementação, *todos* os grupos deverão criar os seus repositórios privados e partilhá-los com os docentes da disciplina (Lipegno e Dntfreitas).

4. Código de ética e de honestidade académica:

Nesta disciplina, espera-se que cada aluno subscreva os mais altos padrões de honestidade académica. Isto significa que cada ideia que não seja do aluno deverá ser explicitamente creditada ao(s) respectivo(s) autor(es). O não cumprimento do disposto constitui uma prática de plágio. O plágio inclui a utilização de ideias, código ou conjuntos de soluções de outros alunos ou indivíduos, ou qualquer outra fonte para além dos textos de apoio à disciplina, sem dar o respectivo crédito a essas fontes. A menção das fontes não altera a classificação, mas os alunos não devem copiar código de outros colegas, ou dar o seu próprio código a outros colegas em qualquer circunstância. De notar que a responsabilidade de manter o acesso ao código somente para os colegas de grupo é de todos os elementos.