 ISEL <small>INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA</small>	Departamento de Engenharia de Eletrónica de Telecomunicações e Computadores Mestrado em Engenharia Informática e de Computadores (MEIC) Mestrado em Engenharia Informática e Multimédia (MEIM)
20-11-2024	Computação Distribuída (Inverno 24/25)

Trabalho Prático de Avaliação Final

Objetivo: Implementação de um sistema distribuído englobando os diferentes paradigmas de interação e *middleware* estudados e já utilizados nos Laboratórios das aulas práticas

Notas prévias:

- **Embora possam já existir períodos de dúvidas nas aulas da semana de 18/Nov, as aulas das semanas de 25/Nov, 02/Dez e 09/Dez de 2024,** serão essencialmente alocadas para apoio à realização do trabalho. No entanto, é pressuposto, e faz parte dos ECTS da Unidade Curricular, que cada grupo de alunos terá de dedicar horas de trabalho fora das aulas. Para eventual apoio e esclarecimento de dúvidas fora das aulas devem agendar com os professores o pedido de ajuda que poderá ser feito presencial ou remoto via Zoom. (nos *links* disponíveis no Moodle de cada turma);
- De acordo com as regras de avaliação definidas no slide 5 do conjunto *CD-01 Apresentação.pdf*, este trabalho tem um peso de 30% na avaliação final e é de entrega obrigatória, com avaliação de nota mínima de 9.5 valores;
- A entrega será realizada em Moodle com um ficheiro Zip, incluindo os projetos desenvolvidos (*src*, *pom.xml* e *assembly.xml*, sem incluir os artefactos JAR), bem como outros ficheiros que considerem pertinentes para valorizar a avaliação do trabalho. **É obrigatório a entrega de documento PDF como um relatório técnico** que descreve o sistema implementado, permitindo a um leitor compreender o objetivo, pressupostos, a arquitetura, a configuração para execução do sistema e as conclusões. A qualidade deste relatório terá peso significativo na avaliação final do trabalho;
- **Na penúltima e última semana do semestre (16 a 20 de dezembro de 2024)** cada grupo terá de apresentar e demonstrar, durante 15 minutos para toda a turma, a funcionalidade e operacionalidade do trabalho realizado, sendo a calendarização em cada turma comunicada posteriormente;
- **A entrega limite no Moodle será 14 de dezembro de 2024 até às 23:59h.**

Considere um cenário onde se pretende que múltiplos utilizadores possam submeter fotos (formatos jpg, png) e um conjunto de palavras para serem marcadas numa nova foto como se ilustra na Figura 1.



Figura 1 – Foto original (esq.) e nova foto (dir.) marcada com lista de palavras

Para que múltiplos utilizadores possam realizar a marcação das suas fotos, usando recursos remotos na Cloud, foi decidido implementar um sistema informático com uma aplicação cliente que os utilizadores podem executar nas suas máquinas locais. Para suportar muitos utilizadores em simultâneo e para garantir distribuição de carga, a aplicação cliente pode aceder a múltiplos servidores que se executam em múltiplas máquinas virtuais (VM) na *Google Cloud Platform*, configuradas com um sistema de ficheiros distribuídos (*Gluster distributed file System*), que permite que qualquer ficheiro criado numa VM seja replicado em múltiplas VM. Assim, a aplicação cliente pode fazer o *upload* de fotos em *streaming* através de um servidor e posteriormente realizar o *download*, também em *streaming*, das fotos marcadas através de outros servidores. Na Figura 2, ilustra-se o diagrama geral do sistema a implementar, cuja descrição é apresentada a seguir.

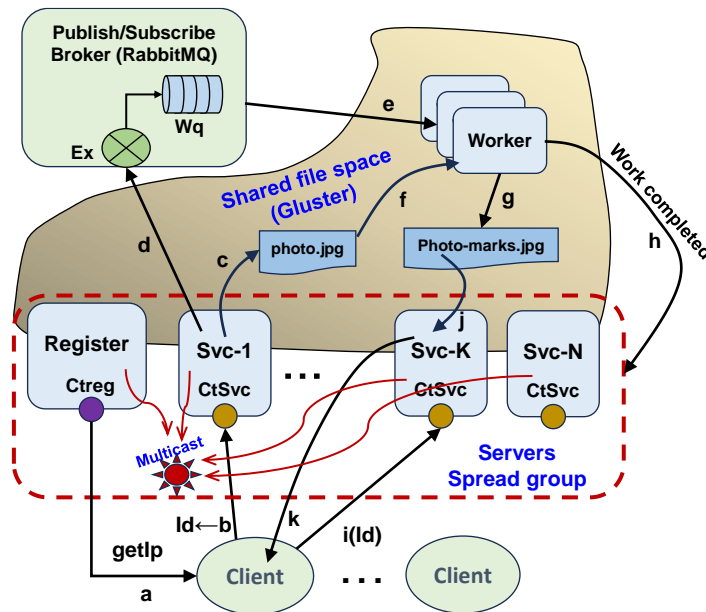



Figura 2 - Diagrama geral do sistema

Requisitos funcionais


- Todos os servidores (*Register*, *Svc-1*, ..., *Svc-N*) pertencem a um grupo Spread de nome *Servers* com suporte para comunicação *multicast*;
- Os servidores (*Register*, *Svc-1*, ..., *Svc-N*) disponibilizam contratos gRPC, adequados a serem usados pela aplicação *Client*;
- O servidor *Register* é o único que tem um *endpoint* (IP, port) conhecido previamente e disponibiliza um contrato (**Fig. 2 - Ctreg**) que permite às aplicações *Client*, em qualquer momento obter o *endpoint* de um dos servidores (*Svc-1*, ..., *Svc-N*), para que possam submeter pedidos;
- As decisões internas ao grupo de servidores (*Servers Spread Group*) são tomadas por um algoritmo de consenso baseado na eleição de um *Leader*, utilizando comunicação *multicast*. O servidor *Register* deve devolver (**Fig 2 - a**) aos *Client* o *endpoint* de um dos servidores (*Svc-1*, ..., *Svc-N*) que, por consenso, tenha o menor número de aplicações *Client* conectadas. Note que face à existência de comunicação por grupos com *multicast* e garantia de ordenação total de mensagens (SAFE) no Spread, o algoritmo a implementar não requer a complexidade de implementação do algoritmo de Consenso Raft;
- Assume-se que o servidor *Register* é sempre o primeiro do grupo (lançado na inicialização do sistema) e que nunca falha. Contrariamente os servidores (*Svc-1*, ..., *Svc-N*) que vão entrando dinamicamente (*Join*), podem sair (*Leave*) ou mesmo falhar;
- Os clientes fazem pedidos de *upload* de fotos e *download* de fotos marcadas, usando *streaming* de ficheiros (bloco a bloco), a qualquer um dos servidores, através do contrato gRPC (**Fig. 2 - CtSvc**);
- O pedido de *upload* (**Fig 2 - b**), feito num servidor, devolve um identificador (*Id*), permitindo que o pedido de *download* possa ser realizado posteriormente noutro servidor (**Fig 2 - i(Id)**);
- Quando um novo servidor *Svc-K* entra no grupo, irá receber, do *Leader* atual, o estado global dos pedidos em curso existentes;
- Os ficheiros com fotos, são armazenados (**Fig. 2 - c**) num espaço partilhado (**Fig.2 - Shared file space**) entre as VM utilizadas, suportado pelo sistema de ficheiros distribuídos Gluster (ver anexo 1);
- Após a conclusão do *upload* das fotos e respetivo armazenamento no Gluster os servidores (*Svc-1*, ..., *Svc-N*) publicam uma mensagem (**Fig. 2 - d**) num *exchange* (**Fig. 2 - Ex**) do *middleware* RabbitMQ, provocando que uma das múltiplas instâncias *Worker* (**Fig. 2 - e**) processe a mensagem, usando o padrão *work-queue* associado a uma fila de mensagens (**fig. 2 - Wq**). A mensagem deve transportar o nome do ficheiro existente no espaço de armazenamento partilhado, bem como as palavras a marcar na imagem;

 ISEL <small>INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA</small>	Departamento de Engenharia de Eletrónica de Telecomunicações e Computadores Mestrado em Engenharia Informática e de Computadores (MEIC) Mestrado em Engenharia Informática e Multimédia (MEIM)
20-11-2024	Computação Distribuída (Inverno 24/25)

- A aplicação *Worker*, faz a marcação das imagens, carregando, em memória, o ficheiro (**Fig.2 - f**) existente no espaço de armazenamento partilhado e, após marcação da foto com as palavras, armazena no espaço partilhado um novo ficheiro com o mesmo nome e um sufixo *marks* (**Fig.2 - g**), enviando de seguida uma mensagem *multicast* (**Fig. 2 - h**) para o grupo Spread de nome *Servers* indicando que finalizou o trabalho. No **anexo 2**, apresenta-se o código de uma aplicação *standalone* que ilustra como marcar uma imagem em formato JPG ou PNG;
- Em qualquer altura, usando qualquer um dos servidores (Svc-1, ..., Svc-N) a aplicação *Client*, pode pedir o *download* em *streaming* de uma foto marcada (**Fig. 2 - i(l), j, k**);

Requisitos não funcionais


- A definição dos contratos gRPC devem seguir princípios de simplicidade e funcionalidade adequada ao problema, sendo da total liberdade de cada grupo de alunos;
- As estruturas de dados internas dos servidores (*Register*, Svc-1, ..., Svc-N) para manutenção de estado dos pedidos é da inteira responsabilidade de cada grupo de alunos;
- O algoritmo de eleição deve ser o mais simples possível e que tire o máximo de partido da existência de mensagens *multicast* e de *Membership* no Spread. O algoritmo de eleição é um aspeto importante na avaliação do trabalho pelo que a sua descrição e demonstração de funcionalidade deve ser cuidadosamente detalhada no relatório final;
- Assuma que a demonstração da funcionalidade final do sistema vai unicamente necessitar de 3 nós computacionais (3 VM) pelo que pode considerar que todos os nós computacionais (VM) têm a mesma configuração;
- Utilize VM *instances* GCP com sistema operativo Ubuntu versão 20.04 LTS, tal como fez durante os Laboratórios e no trabalho prático TPA1;
- Utilize os guias disponibilizados ao longo dos laboratórios com as instruções e os comandos para instalar/configurar as VMs com os *middleware* e *runtimes* necessários para o sistema operativo Ubuntu 20.04 LTS:
 - ✓ Java Open JDK 11;
 - ✓ Docker runtime;
 - ✓ Compilador GCC e outras *tools* necessárias para compilar as *sources* do Spread *toolkit*;
 - ✓ Instalação/configuração do Spread Toolkit de acordo com Laboratório 04;
 - ✓ Para instalação/configuração do Gluster File System (mais informação em <https://www.gluster.org/>), siga as instruções no **anexo 1**;
- A atribuição às 3 VM dos vários componentes (*RabbitMQ*, *Register Server*, (Svc-1, ..., Svc-N) *servers* e *Workers*) do sistema a desenvolver, é definida por cada grupo seguindo uma estratégia que considere adequada;
- Tanto na plataforma RabbitMQ como no Spread *toolkit* os dados das mensagens são normalmente um *array* de bytes (`byte[]`). No entanto, para maior flexibilidade, devem nas várias aplicações usar classes Java para definir as mensagens a transferir entre os diversos intervenientes. Sugere-se a utilização da biblioteca Gson para a serialização de objetos no formato JSON, disponível no repositório central Maven (<https://mvnrepository.com/artifact/com.google.code.gson/gson/>), que de forma simples e flexível permite fazer conversões de objetos para `byte[]` e de `byte[]` para objetos. No **anexo 3**, apresenta-se um exemplo de uso da referida biblioteca (Gson);

 ISEL <small>INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA</small>	Departamento de Engenharia de Eletrónica de Telecomunicações e Computadores Mestrado em Engenharia Informática e de Computadores (MEIC) Mestrado em Engenharia Informática e Multimédia (MEIM)
20-11-2024	Computação Distribuída (Inverno 24/25)

Sugestões Gerais

- Qualquer questão ou dúvida sobre os requisitos deve ser discutida com o professor;
- Antes de começar a escrever código desenhe a arquitetura do sistema, os contratos de interação bem como os diagramas de interação mais importantes, nomeadamente as mensagens necessárias ao algoritmo de eleição;
- Quando tiver dúvidas sobre os requisitos, verifique no site *Moodle* se existem "*Frequently Asked Questions*" com esclarecimentos sobre o trabalho;
- Parametrize todas as aplicações por forma a ser possível fazer *deployment* do sistema em múltiplas VM, sem portos e endereços TCP/IP *hard-coded*. Sugere-se a utilização nas várias aplicações de argumentos na linha de comando, por exemplo:

```
java -jar worker.jar <ipRabbitMQ> <portRabbitMQ> <workQueue> <Spread Group> <...>
```
- Inscreva no código comentários ou em ficheiros *readme.txt* descrições sucintas e justificativas das partes mais relevantes;
- Não esqueça que o relatório é parte importante e terá peso na avaliação final do trabalho.

 ISEL <small>INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA</small>	Departamento de Engenharia de Eletrónica de Telecomunicações e Computadores Mestrado em Engenharia Informática e de Computadores (MEIC) Mestrado em Engenharia Informática e Multimédia (MEIM)
20-11-2024	Computação Distribuída (Inverno 24/25)

Anexo 1: Instalação do Sistema de ficheiros distribuídos Gluster

Numa Vm base com sistema operativo Ubuntu, para além da instalação dos ambientes já usados nos laboratórios (Java, Docker e Spread Toolkit) deve agora instalar o sistema de ficheiros distribuídos Gluster (<https://www.gluster.org/>).

Instalação do Gluster

```
sudo add-apt-repository ppa:gluster/glusterfs-7
```

```
sudo apt update
```

```
sudo apt install glusterfs-server
```

Lançar o daemon do Gluster e verificar que está em execução

```
sudo service glusterd start
```

Observar o estado do serviço gluster

```
sudo service glusterd status
```

Criar diretoria com permissões totais para ser um Brick de um gluster volume

```
sudo mkdir -p /var/gluster/brick
```

```
sudo chmod 777 /var/gluster/brick
```

Criar diretoria para armazenar ficheiros das aplicações que criam/usam ficheiros no sistema de ficheiros distribuídos (gluster). Esta diretoria será Mounted em cada nó computacional formando um volume gluster

```
sudo mkdir /var/sharedfiles
```

```
sudo chmod 777 /var/sharedfiles
```

As instruções seguintes assumem um cenário em que foram criados 3 nós computacionais (3 VM) com os nomes tpa2-node1, tpa2-node2 e tpa2-node3

Em cada VM lançar o daemon do Gluster e verificar que está em execução

```
sudo service glusterd start
```

Observar o estado do serviço gluster em execução através do comando:

```
sudo service glusterd status
```

Colocar no ficheiro /etc/hosts das 3 Vm a associação de nomes dos nodes e

os IP internos dos 3 nós computacionais

```
10.128.0.8 tpa2-node1
```

```
10.128.0.10 tpa2-node2
```

```
10.128.0.11 tpa2-node3
```

(ATENÇÃO: Só executar num node, ex: tpa2-node1): Definir os nós peers

```
sudo gluster peer probe tpa2-node2
```

```
sudo gluster peer probe tpa2-node3
```

Em cada nó verificar que já são gluster peers dos outros nodes

```
sudo gluster peer status
```

(ATENÇÃO: Só executar num node, ex: tpa2-node1): Criar um Gluster volume

```
sudo gluster volume create glustervol replica 3 tpa2-node1:/var/gluster/brick \
tpa2-node2:/var/gluster/brick tpa2-node3:/var/gluster/brick force
```

No mesmo node onde se criou o volume, por exemplo no tpa2-node1

```
sudo gluster volume start glustervol
```

Em cada node associar (mount) a diretoria /var/sharedfiles para o gluster volume

Por exemplo no tpa2-node1

```
sudo mount -t glusterfs tpa2-node1:/glustervol /var/sharedfiles
```

Note que não é possível fazer mount do gluster brick como diretoria partilhada

```
sudo mount -t glusterfs tpa2-node1:/glustervol /var/gluster/brick
```


Para testar que todos os nós partilham um volume com réplicas de ficheiros

execute numa das Vms o comando linux

```
date > /var/sharedfiles/date.txt
```

verifique que o ficheiro está replicado nas 3 Vms na diretoria /var/sharedFiles

Altere o ficheiro num dos nodes e verifique que a alteração existe nas réplicas

 ISEL <small>INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA</small>	Departamento de Engenharia de Eletrónica de Telecomunicações e Computadores Mestrado em Engenharia Informática e de Computadores (MEIC) Mestrado em Engenharia Informática e Multimédia (MEIM)
20-11-2024	Computação Distribuída (Inverno 24/25)

Anexo 2: Aplicação de marcação de imagens JPG ou PNG


```
package markimage;

import javax.imageio.ImageIO;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;

public class MarkApp {

    public static void main(String[] args) {
        // args[0] - image pathname; args[1] - image result pathname
        // args[2]...args[n] keywords to mark image
        String inputPath=args[0];
        String outputPath=args[1];
        ArrayList<String> keywords=new ArrayList<>();
        for (int i=2; i < args.length;i++) keywords.add(args[i]);
        BufferedImage img = null;
        try {
            img = ImageIO.read(Path.of(inputPath).toFile());
            annotateImage(img, keywords);
            int lastIndex = inputPath.lastIndexOf('.');
            String format=inputPath.substring(lastIndex+1);
            ImageIO.write(img, format, Path.of(outputPath).toFile());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void annotateImage(BufferedImage img, ArrayList<String> keywords) {
        Graphics2D gfx = img.createGraphics();
        gfx.setFont(new Font("Arial", Font.PLAIN, 22));
        gfx.setColor(new Color(0x0000ff));
        String sentence="";
        for (String s : keywords) sentence+=s+" ";
        gfx.drawString(sentence, 10, 20);
        Polygon poly = new Polygon();
        poly.addPoint(3, 3);
        poly.addPoint(10*sentence.length(), 3);
        poly.addPoint(10*sentence.length(), 25);
        poly.addPoint(3, 25);
        poly.addPoint(3, 3);
        gfx.setColor(new Color(0xff0000));
        gfx.draw(poly);
    }
}
```

 ISEL <small>INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA</small>	Departamento de Engenharia de Eletrónica de Telecomunicações e Computadores Mestrado em Engenharia Informática e de Computadores (MEIC) Mestrado em Engenharia Informática e Multimédia (MEIM)
20-11-2024	Computação Distribuída (Inverno 24/25)

Anexo 3: Exemplo de conversão: Objeto -> byte[] -> Objeto

```

public static void main(String[] args) {
    SomeClass someObject=new SomeClass();
    someObject.setId(5); someObject.setName("ABCD");
    someObject.setResults(new String[]{"abc","def"});
    System.out.println(someObject.toString());
    // converter objeto em string JSON
    Gson js=new GsonBuilder().create();
    String jsonString=js.toJson(someObject);
    System.out.println(jsonString);
    // converter string em byte[]
    byte[] binData=jsonString.getBytes(StandardCharsets.UTF_8);
    for (byte b : binData) System.out.print(b+" ");
    System.out.println();
    // binData pode ser enviado como mensagem em binário
    // em qualquer plataforma por exemplo RabbitMQ ou Spread,...
    // Receção de binData e deserialização para objeto
    String newJsonString=new String(binData, StandardCharsets.UTF_8);
    SomeClass newSomeObject=js.fromJson(newJsonString,SomeClass.class);
    System.out.println(newSomeObject.toString());
}

```

```

public class SomeClass {
    private int id;
    private String name;
    private String[] results;

    public SomeClass(){}

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String[] getResults() { return results; }
    public void setResults(String[] results) { this.results = results; }

    @Override
    public String toString() {
        String strResults="["; boolean first=true;
        for (String s : getResults()) {
            strResults+= first? "\"" +s+"\"": ","+"\"" +s+"\""; first=false;
        }
        strResults+="]";
        return "SomeClass("+getId()+","+getName()+","+strResults+")";
    }
}

```

```

<dependency>
<groupId>com.google.code.gson</groupId>
<artifactId>gson</artifactId>
<version>2.10.1</version>
</dependency>

```