Relatório Processamento Distribuído de Imagens (TPA1)

Autores: Afonso Bento, Pedro Silva, Leonil Sulude

Docentes: Luís Assunção, José Simão

Unidade Curricular: Computação Distribuída
Data: 08 de Novembro 2025

Descrição Geral

O sistema implementado tem como objetivo demonstrar o funcionamento de uma arquitetura distribuída para processamento automático de imagens, desenvolvida em Java e executada em ambiente Google Cloud Platform (GCP).

A solução utiliza gRPC para comunicação entre componentes, Docker para processamento isolado e Redis como sistema de sincronização e partilha de estado entre servidores.

Composição da Solução

O projeto é composto por oito módulos Java Maven, cada um com uma responsabilidade bem definida:

- ManagerServerContract – Define o contrato gRPC (Protocol Buffers) entre ManagerServer e ImgServers.
- ManagerClientContract – Define o contrato entre ManagerServer e clientes.
- ImgClientContract – Define o contrato entre clientes e ImgServers.
- ManagerServerApp – Implementa o ManagerServer, que coordena os ImgServers e gere o Redis.
- ImgServerApp – Implementa os servidores de imagem responsáveis por receber, processar e armazenar as imagens.
- ImageProcessorApp – Aplicação executada em containers Docker que realiza o

redimensionamento das imagens.

- ClientApp – Aplicação cliente interativa que permite upload/download de imagens.

- DemoApp – Aplicação para testes de carga. Simula vários clientes concorrentes, onde cada cliente envia múltiplos pedidos de upload utilizando automaticamente a imagem incluída dentro do próprio JAR.

Contratos (Interfaces gRPC)

1. ManagerServerClientService – fornece o ImgServer disponível para o cliente.

Método: GetImgServer(GetImgServerRequest) → GetImgServerResponse Retorna o endereço (IP e porta) de um servidor de imagens disponível para processamento.

2. ManagerServerRegistrationService – permite o registo dos ImgServers no ManagerServer.

Método: RegisterImgServer(RegisterImgServerRequest) → RegisterImgServerResponse Permite que cada ImgServer se registe, informando o seu IP, porta e obtendo a configuração do Redis.

3. ImgServerClientService – gere o upload e download de imagens em streaming, com suporte a estados e redirecionamentos.

Métodos:

– UploadImage(stream ImageChunk) → UploadImageResponse — recebe imagens enviadas pelo cliente em fluxo (streaming).

– DownloadImage(DownloadImageRequest) → stream DownloadImageChunk — envia imagens processadas ou redireciona o pedido conforme o estado.

Principais Erros Encontrados e Soluções

1. Erro: “Address types of NameResolver 'unix' not supported by transport”

Causa: resolvers unix/npipe não suportados em conexões TCP/IP.

Solução: uso exclusivo do DNS NameResolver.

2. Erro: “Could not find policy 'pick_first' in LoadBalancerRegistry”
Causa: O gRPC não encontrou a política de balanceamento padrão (pick_first), responsável por definir como o cliente escolhe o servidor a contactar, mesmo quando existe apenas um.

Solução: Foi registado manualmente o PickFirstLoadBalancerProvider, restaurando o comportamento padrão e permitindo a criação correta dos canais de comunicação.

Considerações Técnicas

• Docker é usado para isolar o processamento das imagens.
• ManagerServer usa round-robin para distribuir pedidos.
• DemoApp simula carga concorrente com múltiplos clientes.

Passos para Execução

Após a inicialização das máquinas virtuais, aceda ao terminal de cada uma e execute os comandos indicados.

Assume-se que os ficheiros JAR de cada aplicação já se encontram no diretório HOME das respetivas máquinas e que a imagem Docker da aplicação de processamento de imagens já foi previamente criada a partir do *Dockerfile*.

1. VM1 (ubuntu20lts-base) - ManagerServer (IP Público: 35.187.4.84)

// Lançar container do Redis
docker run -d --name redis-server -p 6379:6379 redis
OU
docker start redis-server (Se container já estiver criado)

// Iniciar a aplicação do ManagerServer
java -DmanagerIp=35.187.4.84 -DmanagerPort=8000 -DredisPort=6379 -jar ManagerServerApp-1.0-SNAPSHOT-jar-with-dependencies.jar

2. VM2 (ubuntu20lts-base2) - ImgServer1 (IP Público: 35.205.152.215)

java -Dip=35.205.152.215 -Dport=8000 -DmanagerIp=35.187.4.84 -DmanagerPort=8000 -jar ImgServerApp-1.0-SNAPSHOT-jar-with-dependencies.jar

3. VM3 (ubuntu20lts-base3) - ImgServer2 (IP Público: 104.155.1.232)

java -Dip=104.155.1.232 -Dport=8000 -DmanagerIp=35.187.4.84 -DmanagerPort=8000 -jar ImgServerApp-1.0-SNAPSHOT-jar-with-dependencies.jar

// Iniciar o ImgServer3 (na mesma VM opcionalmente)
java -Dip=104.155.1.232 -Dport=8001 -DmanagerIp=35.187.4.84 -DmanagerPort=8000 -jar ImgServerApp-1.0-SNAPSHOT-jar-with-dependencies.jar

4. Cliente (na máquina local ou remota)

// Execução manual
java -DmanagerIp=35.187.4.84 -DmanagerPort=8000 -jar ClientApp-1.0-SNAPSHOT-jar-with-dependencies.jar

// Execução do teste de carga
java -DmanagerIp=35.187.4.84 -DmanagerPort=8000 -Dclients=5 -DuploadsPerClient=3 -jar DemoApp-1.0-SNAPSHOT-jar-with-dependencies.jar