Projecto Computação Distribuída

Licenciatura em Engenharia Informática - Computação Distribuída

Distributed Music Editor

Docentes:
- Diogo Gomes (dgomes@ua.pt)
- Nuno Lau (nunolau@ua.pt)
- Mário Antunes (mario.antunes@ua.pt)
- Alfredo Matos (alfredo.matos@ua.pt)

Prazo:
5 de Junho – 24h00

Conceitos a abordar:

- Sockets
- Marshalling
- Message Broker
- Fault Tolerance

Introdução

A firma *Advanced Sound Systems* (ASS) encontra-se a desenvolver uma aplicação de karaoke para músicos. Esta aplicação distingue-se por não só remover a voz das músicas, mas também remover instrumentos individuais, permitindo a um músico substituir a performance do músico original pela sua. Este novo serviço será disponibilizado online através de um portal web em que o músico pode fazer upload de um ficheiro de música, analisar os instrumentos que compõem a música, selecionar vários instrumentos e finalmente receber um ficheiro novo em que a música contém apenas esses instrumentos.

A tarefa de separar uma música em várias pistas por instrumento é intensiva do ponto de vista de processamento pelo que a ASS contratou os alunos de Computação Distribuída para desenvolver um serviço online capaz de atender às necessidades da empresa com uma qualidade de experiência elevada para o utilizador final (identificação rápida dos instrumentos e construção do novo ficheiro sem o(s) instrumento(s) seleccionados).

Para esta tarefa deverão desenvolver todo o código necessário para a criação de um serviço Web que possa servir em paralelo múltiplos clientes de forma rápida e eficiente recorrendo a computação paralela (aqui conseguido através de paralelização em múltiplos processos/workers independentes que podem estar no mesmo computador ou não).

Para esta tarefa é fornecido este guião e um conjunto de pistas de música disponíveis em [4]. Devem criar o vosso projecto no GitHub Classroom usando https://classroom.github.com/a/q9wGcN9U.

O repositório inclui ja código exemplo [1][2][3] de como poderá processar ficheiros MP3 de música e detectar 4 pistas distintas (número que será suficiente para este projecto).

Objectivos

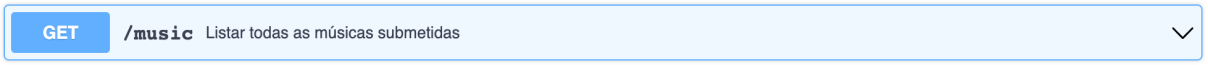
Com este trabalho pretende-se que os alunos desenvolvam um sistema que divida a tarefa de processamento em múltiplas sub-tarefas paralelizáveis com o objectivo de aumentar a performance do sistema.

Funcionamento

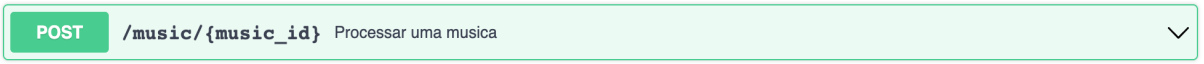
Este é um trabalho aberto em que terão que desenvolver um sistema de distribuição das tarefas de computação (separação dos instrumentos) e o escalonamento de tarefas para servir o backend ao portal da ASS.

O sistema terá como interface uma API REST, através da qual o portal poderá:

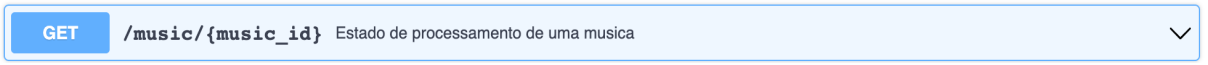
Submeter ficheiros de áudio em formato MP3. Em resposta à submissão do ficheiro de áudio, o serviço deverá retornar um ficheiro JSON contendo um identificador do trabalho (music_id), e uma lista de identificadores das pistas de instrumentos (cada identificador deverá corresponder a um instrumento). Este método apenas armaneza e analisa os metadados da música.



Listar todas as músicas submetidas

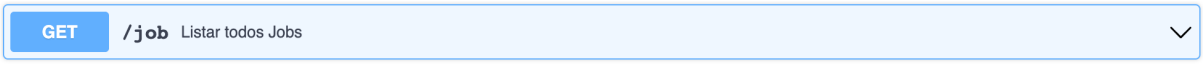


Requisitar o processamento assíncrono (este método deve retornar imediatamente) da música <music_id> com apenas os instrumentos listados através do seu identificador num ficheiro JSON submetido. O sistema deve fazer a separação da música nos seus vários instrumentos, dando origem a um ficheiro por instrumento. No final, o sistema deverá misturar os instrumentos seleccionados num unico ficheiro.

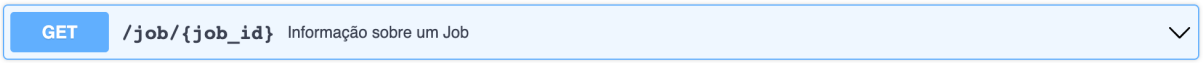


Requisitar o estado de processamento de uma música, quando o estado é 100% a resposta deve incluir o url do ficheiro final. Quando a pista de um instrumento estiver completo deverá retornar o link para o ficheiro de cada instrumento (permitindo ao utilizador fazer o download e reprodução do ficheiro). No final haverá também um link para o ficheiro final misturado (contendo os instrumentos seleccionados).

Para facilitar a utilização deste sistema deverá ainda implementar os endpoints:



Listar todos JOBs (JOB é uma tarefa distribuída a um worker) submetidos e respectivas estatísticas (tamanho do ficheiro, link para os vários ficheiros, tempos de processamento, etc.)



Lista JOB's utilizados para processar, tempo de processamento, número de instrumentos

Apagar todos ficheiros temporários e músicas (sistema limpo), cancelar todos os workers ainda em funcionamento.

A API encontra-se documentada utilizando a especificação OpenAPI 3.0. Pode encontrar tanto o ficheiro fonte como versões PDF e HTML no seu repositório de código (api.yaml, api.pdf, api.html respectivamente). Tenha em atenção que poderá haver necessidade de actualizações antes da data de entrega do projecto, pelo que deverá estar atento aos anúncios no canal #cd do detiuaveiro.slack.com

Deverá ainda criar um portal Web (frontend) **básico** para este projecto para efeitos de **demonstração**. Para efeitos de desenvolvimento e teste tem ficheiros de áudio em [3].

Avaliação

A avaliação deste trabalho será feita através da submissão de código na plataforma GitHub classroom e de um relatório em formato PDF com não mais de 5 páginas colocado no mesmo repositório junto com o código e com o nome relatorio.pdf.

Está em avaliação o protocolo definido e documentado, assim como as *features* implementadas de acordo com os objetivos:

- Serviço web com API REST seguindo os requisitos do trabalho
- Solução distribuída para o processamento de ficheiros de áudio
    - Separação do ficheiro original em múltiplos trechos
    - Identificação dos instrumentos através da separação das sub-pistas
    - Criação de novo ficheiro completo
- Eficiência da solução desenvolvida (pretende-se um sistema capaz de responder de forma mais rápida possível aos pedidos dos clientes finais)
- Atender a múltiplos clientes em simultâneo.
- Atender à possibilidade de ocorrência de falhas

Observações

1. O código exemplo faz uso de todos CPU's fisicos por defeito, para testes locais é necessario limitarmos a 1 CPU. Para este efeito deve acrescentar o seguinte código:

    ```
    import torch
    torch.set_num_threads(1)
    ```
2. Pode utilizar o site https://editor.swagger.io para consultar todos detalhes do ficheiro api.yaml
3. Para obter a melhor performance do seu sistema deve dividir a tarefa assíncrona de processamento da música em pedaços de música mais pequenos e processar os mesmos em paralelo.

Referências

[1] https://github.com/facebookresearch/demucs
[2] https://pytorch.org/
[3] https://torchmetrics.readthedocs.io/en/stable/
[4] https://filesender.fccn.pt/?s=download&token=cd4fcd29-b3f1-4a4d-9da3-50aae00e702d