Distributed Music Editor

Universidade de Aveiro

Licenciatura em Engenharia Informática

Computação Distribuida

Ano Letivo 2022/23

Professores:

Diogo Gomes

Nuno Lau

Trabalho realizado por:

Vasco Faria 107323

Cristiano Nicolau 108536

Introdução

No âmbito da unidade curricular de Computação Distribuída, foi nos proposto um projeto que consiste no desenvolvimento de um sistema distribuído de edição de músicas.

Este deveria ser capaz de, dada uma música, com diversos tamanhos, fosse possível processar e criar outras músicas, juntando os instrumentos escolhidos pelo utilizador.

Bibliotecas Utilizadas

O server e o cliente comunicam através de Flask, que é uma framework de python usada em web. É utilizado um ambiente virtual para trabalhar localmente. Para os pedidos entre o cliente e o server, é utilizada a biblioteca Python Requests, que permite a interação entre os servidores HTTP. Finalmente, a comunicação entre os diferentes workers e o server é feito através de sockets.

Funcionamento do Sistema

O programa contém 3 scripts executáveis. O primeiro é o *server.py,* que recebe as conexões dos clientes através do Flask, e comunica com os workers através de sockets TCP. O segundo é o *worker.py*, que requer um argumento obrigatório- o id do worker. O terceiro script é o *client.py*, que comunica através do flask com o servidor.

O cliente pretende simular um utilizador de sistema, e para alem de submeter novas músicas para o sistema, processa e lista todas as músicas do sistema.

As funcionalidades estão dentro do worker. O cliente especifica o que quer que o sistema faça e o server transmite essa informação aos workers.

Server

O server é a nossa main. É dentro do server que tudo acontece, é aqui que recebemos todos os pedidos do cliente e é transmitida a informação aos workers.

É ainda no server que acontecem coisas mais simples como armazenamento e analisa dos meta dados das músicas e ainda a listagem e todas as musicas já submetidas.

O server só se desconecta quando o desligamos.

Worker

É nos workers que ocorre todo o trabalho mais pesado, desde a separação das músicas nos seus vários instrumentos que são indicados pelo cliente, dando origem a novos ficheiros de áudio. Ocorre ainda a mistura desses mesmo instrumentos em uma *track* final.

Client

O cliente tem uma interface onde pode fazer diversas coisas, recebendo uma resposta do servidor, em relação ao seu pedido, onde o seu pedido consegue ser processado ou uma mensagem de erro.

Submit Music

O cliente submete uma música para o server. Dentro do server ocorre a analise dos meta dados da música e toda a informação é guardada em um dicionário com chaves únicas correspondentes ao id da música. De seguida o server envia toda esta informação para o cliente.

List Musics

O server interage com o dicionário onde estão guardadas todas as músicas e envia para o cliente uma lista com a informação de cada música guardada no server.

Process Music

Quando o cliente pede o processamento de uma música, caso a música ou o id da track escolhida não exista o server envia uma mensagem erro, senão envia uma mensagem de sucesso. O server recebe o pedido de processamento e envia a música e as tracks pedidas pelo cliente para o worker através de sockets.

Nos workers a música é processada de forma assíncrona, deixando o cliente continuar a fazer outros pedidos ao server.

Progress Music

Quando é pedido o progresso de uma música, o worker no caso da música já tiver sido totalmente processada envia os links de cada track das quais foram pedidos o processamento e ainda da track final, numa lista, para que seja possível ocorrer o download. No caso de o processamento ainda não ter ocorrido envia a percentagem de progresso e uma lista vazia.

Jobs e Reset

Sempre que ocorre algum pedido a um worker, é guardado o pedido em um dicionário com a chave única job_id. Ou seja, sempre que algum worker, faz um processamento ou um progresso, é guardada toda a informação no dicionário.

O clientes se quiserem podem aceder a estes dicionários, podendo listar todos os jobs_ids e de seguida aceder a um job específico.

O Cliente pode ainda fazer um pedido ao server, para reiniciar todo o sistema pagando todos os ficheiros guardados, apagando toda a memoria e ainda desligar os workers, sendo necessário voltar a ligá-los caso seja preciso o seu uso.

Protocolo

Todas as mensagens trocadas entre o servidor e o client são enviadas no formato JSON, e as mensagens trocadas entre o server e os workers são no formato Pickle.

O nosso protocolo contém 2 métodos de comunicação, para comunicarmos com o worker (send_message_to_workers e receive_results_from_workers).

Utilizando sockets TCP, as mensagens são enviadas para os workers conectados. A mensagem é serializada através do método *Pickle* e é enviada em 2 partes: os 4 primeiros bytes indicam o tamanho da mensagem permitindo saber quantos bytes são esperados receber.

O server e o cliente trocam várias mensagens de *Post* e *Get*, através de endpoints do flask, de forma que o cliente interaja com o servidor para enviar, processar e obter informações sobre a música e sobre os jobs.

O servidor e os workers utilizam ainda threads para permitir a comunicação e a execução de tarefas assíncronas em segundo plano.

Observações

É possível conectar vários workers ao server, mas não foi implementada a divisão de tarefas de processamento da música em pedaços mais pequenos, sendo processados em paralelo.

Não foi desenvolvido portal web para o projeto, mas o script clients.py corresponde ao mesmo sendo uma interface desenvolvida em python, que obtém todas as respostas e faz todos os pedidos desejados.