IPS Instituto Politécnico de Setúbal – Escola Superior de Tecnologia de Setúbal	COMPUTAÇÃO PARALELA E DISTRIBUÍDA 2022/2023
	Laboratório 05: Pipelining

Objetivos do laboratório

Pretende-se que o aluno:

1. Entenda o conceito de *pipelining* como forma de paralelismo
2. Conheça alguns mecanismos para processar imagens no Python
3. Aplique o *pipelining* usando *threads* e filas no Python
4. Aplique o *pipelining* usando processos e filas no Python

Pré-requisitos:

- Deverá ter instalado o *Package Python* **Pillow**. Para instalar este package poderá utilizar comando **pip install pillow** ou aba *Python Packages* do Pycharm.
- Deverá descompactar o ficheiro com as imagens e com o código fornecido para a diretoria do projeto.

Pipelining

Processamento em *pipelining* é semelhante a uma linha de montagem: um *worker* (*thread* ou processo) aplica alguma manipulação aos dados e passa-os para o próximo *worker*.

Uma *pipeline* é composta de uma série de elementos de processamento encadeados, onde o output de um elemento é o input do próximo. A Figura 1 ilustra o conceito:

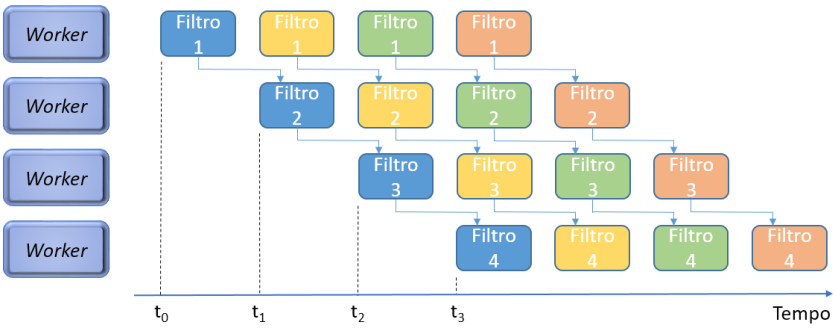

Figura 1 – Processamento em Pipelining

Processamento de Imagens no Python

Neste laboratório vamos utilizar o Pillow para tratar/manipular imagens. A documentação deste package poderá ser consultada no seguinte link:

https://pillow.readthedocs.io/en/stable/reference/index.html

Nível 1

Escreva o código apresentado e responda as questões colocadas.

```
from PIL import Image, ImageFilter, ImageEnhance
import os
from timeit import default_timer as timer


def cinzentos(imagem):
    enh = ImageEnhance.Color(imagem)
    imagem = enh.enhance(0.0)
    return imagem


def contraste(imagem):
    enh = ImageEnhance.Contrast(imagem)
    imagem = enh.enhance(1.8)
    return imagem


if __name__ == '__main__':
    inicio = timer()
    diretoria = "./imagens"  # As imagens devem estar na pasta imagens
    for file in os.listdir(diretoria):
        im = Image.open(diretoria + "/" + str(file))
        im.show()
        im = cinzentos(im)
        im = contraste(im)
        im.show("B&W e mais contraste")
    fim = timer()
    print(f"Tempo para tratamento sequencial: {fim - inicio:.8f}")
```

Explique o que faz o programa.

Nível 2

Escreva o código apresentado e responda as questões colocadas.

```
from threading import Thread
from PIL import Image, ImageEnhance
import os
import queue

from timeit import default_timer as timer


def aplicar_efeito(imagem, efeito):
    if efeito == 'cinzentos':
        enh = ImageEnhance.Color(imagem)
        imagem = enh.enhance(0.0)
    elif efeito == 'contraste':
        enh = ImageEnhance.Contrast(imagem)
        imagem = enh.enhance(1.8)
    # imagem.show()
    return imagem


def trabalhador(tarefa, input_queue, output_queue):
    while True:
        imagem = input_queue.get()
        if imagem is None:
            output_queue.put(None)
            return
        imagem = aplicar_efeito(imagem, tarefa)
        output_queue.put(imagem)



if __name__ == '__main__':
    diretoria = "./imagens"
    inicio = timer()
    queue_inicial = queue.Queue()
    queue_w1_w2 = queue.Queue()
    queue_final = queue.Queue()
    # colocar imagens na queue inicial
    for file in os.listdir(diretoria):
        # Read image
        imagem = Image.open(diretoria + "/" + str(file))
        # Display image
        imagem.show()
        queue_inicial.put(imagem)
    # colocar o terminador na fila de trabalhos
    queue_inicial.put(None)
    trabalhador_1 = Thread(target=trabalhador, args=('cinzentos', queue_inicial, queue_w1_w2))
    trabalhador_1.start()
    trabalhador_2 = Thread(target=trabalhador, args=('contraste', queue_w1_w2, queue_final))
    trabalhador_2.start()
    trabalhador_2.join()
    while not queue_final.empty():
        imagem = queue_final.get()
        if imagem:
            imagem.show()
    fim = timer()
    print(f"Tempo para pipelining: {fim - inicio:.8f}")
```

Explique como é utilizado o *pipelining* no código apresentado realçando o papel de cada uma das *queues* utilizadas.

Nível 3

No código apresentado são definidas duas funções que permitem aplicar novos efeitos.

```
from PIL import Image, ImageFilter, ImageEnhance

def arestas(imagem):
    imagem = imagem.filter(ImageFilter.FIND_EDGES)
    return imagem


def desfocagem(imagem):
    imagem = imagem.filter(ImageFilter.BLUR)
    return imagem
```

Altere o programa do nível 1 para que passem a ser aplicados os 4 efeitos às imagens.

Nível 4

Altere o programa apresentado no nível 2 para que passem a ser aplicados os quatro efeitos às imagens, utilizando *pipelining* com *threads* e *queues.*

Nível 5

Altere o programa apresentado no nível 2 para que passem a ser aplicados os quatro efeitos às imagens, utilizando *pipelining* com processos e *queues.*