João Scaravonatti - 8ª fase

Processamento de *array* 2D: aplicação de máscara de convolução

Para este trabalho final, foi escolhido o problema de processamento de um *array* 2D em paralelo. Na prática, foi aplicada uma máscara de convolução (*kernel*) sobre uma imagem de entrada, gerando como saída a imagem após o processamento e aplicação do *kernel*.

O desenvolvimento prático deste trabalho foi feito utilizando a linguagem de programação *Python.* Para trabalhar com processamento paralelo, foi utilizado o módulo *multiprocessing.* Os cálculos e manipulações de matrizes foram realizados utilizando a biblioteca *numpy*, pela facilidade que ela traz para este tipo de procedimento. Com relação à manipulação de imagens, foi utilizada a biblioteca *cv2* para a leitura das entradas, e a biblioteca *PIL* para a visualização. Por fim, foi utilizado o módulo *time* para realizar comparações entre os experimentos com execução em uma única tarefa e a execução com várias tarefas paralelas.

Para gerar o *array* 2D, utilizou-se a função *imread* do *cv2* para ler a imagem de entrada em escalas de cinza. Após isso, a entrada foi dividida em quatro tiras verticais, o mesmo número de tarefas que foram selecionadas para serem executadas paralelamente. Com essa divisão, foi criado um objeto *Pool* do módulo *multiprocessing*, sendo passado 4 para o valor de *processes* no construtor da classe, ou seja, o número de processos *workers*.

Utilizando o objeto *Pool* criado, é chamada a função *starmap*, passando como argumento a função que realiza a convolução 2D, e um vetor de tuplas, onde cada um contém uma tira da imagem e o *kernel*. A função *starmap* é responsável por enviar os parâmetros (divididos igualmente visando o balanceamento de carga) para as tarefas que serão executadas paralelamente, e ao final, a *starmap* retorna um vetor contendo os resultados retornados em cada tarefa.

Para a condução dos experimentos, foi utilizada uma imagem com resolução 225x255 (Figura 1) e *kernel* de detecção de bordas (Figura 2).

Figura 1: Imagem de entrada.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figura 2: Detecção de bordas.

O primeiro experimento foi realizado sem a utilização dos recursos de paralelismo e realizou a operação de convolução em aproximadamente 0.2 segundos. Já o segundo experimento, utilizando 4 tarefas paralelas, obteve o resultado em aproximadamente 0.04 segundos. Como foi utilizado o mesmo *kernel*, a saída dos dois experimentos pode ser observada na Figura 3:

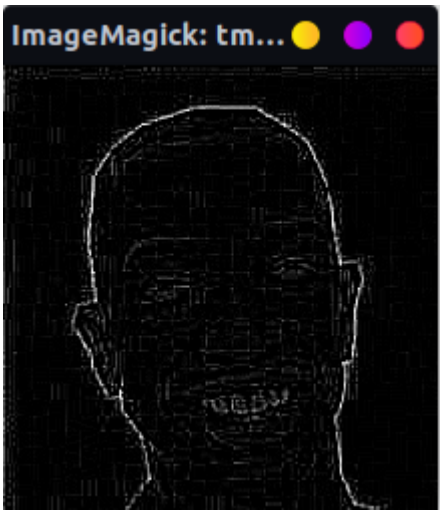

Figura 3: Imagem com *kernel* de detecção de bordas aplicado

Como visto nos resultados de performance, o processamento do *array* foi executado em menos tempo no experimento implementado com paralelismo, superando em aproximadamente 5 vezes o experimento com apenas um *worker*. Deste modo, destaca-se a importância do processamento paralelo neste tipo de cenário onde o trabalho pode ser dividido e realizado em tempos menores.