Trabalho de Laboratório CUDA

Este trabalho é composto por dois laboratórios.
O primeiro laboratório visa fornecer uma primeira noção sobre como usar CUDA. Isto inclui como compilar um programa CUDA, como lançar um Kernel CUDA, como indexar arrays 1D e muito mais.
O segundo laboratório apresenta um caso de uso para usar GPUs com imagens. O objetivo é fazer você entender como indexar matrizes 2D, enquanto fazendo algo prático.

Se você for executar o código CUDA no Google Colab você precisa executar os seguintes passos :

1) Na sua sessão você deve habilitar a GPU. No menu principal escolha Runtime → Change Runtime Type . Escolha T4 Type e pressione o botão Save.

2) Instale um plugin para habilitar que o código cuda seja executado no notebook, através dos comandos :

```
!pip install git+https://github.com/andreinechaev/nvcc4jupyter.git
%load_ext nvcc_plugin
```

3) No código CUDA, na primeira linha escreva %%cu, para que o notebook saiba que o código a seguir é um código CUDA.

4) Feito isso, você poderá executar o código CUDA.

Laboratório 01

Abra lab01_ex1.cu e encontre a seção comentada “TO-DO #1.2” no código. Ao executar o código veja que somente é impresso o código da CPU. O código da GPU não é impresso. Você precisa introduzir o código de sincronização para aguardar a execução da GPU. Introduza este código de sincronização para fazer o código funcionar conforme o esperado.

Laboratório 02

Agora que você entende como compilar e executar um programa CUDA simples, neste exercício você criará seu primeiro kernel CUDA e introduzirá algumas das funcionalidades necessárias para este kernel funcionar. Isso inclui definir a distribuição dos threads ou adicionar memória nas operações de gerenciamento para transferir dados da memória do host para a GPU e vice-versa. Vamos usar host ou CPU para se referir ao espaço da CPU.
Para isso, implementaremos um programa SAXPY simples. SAXPY é muito adequado para você entender como indexar arrays 1D dentro de um kernel de GPU. O termo significa “Single-Precision A*X Plus Y”, onde A é uma constante e X e Y são matrizes.

Usaremos o arquivo lab01_ex2.cu para resolver o exercício. O código-fonte contém uma função main() que aloca dois arrays, x e y, e inicializa cada elemento do array com 0,1 e 0,2, respectivamente. Espera-se que o usuário forneça o valor da constante “a“ como entrada para o programa. O tamanho de cada array é predefinido com a constante ARRAY_SIZE. No momento, o programa apenas chama cpu_saxpy() para calcular o resultado SAXPY usando a CPU, mas posteriormente você também introduzirá uma chamada para a versão da GPU. Finalmente, o código gera um hash do resultado de ambas as versões. Será utilizado no final da execução para comparar as soluções da versão CPU e da versão GPU.

Você introduzirá pequenas alterações no código fonte de lab01_ex2.cu para permitir uma versão SAXPY na GPU. Você terá que introduzir os seguintes passos :
1)TO-DO #2.1 Declare e implemente o kernel CUDA que você gostaria de executar na GPU.
2)TO-DO #2.2 Defina a distribuição dos threads, em termos da dimensão da grade (grid) e da dimensão de cada bloco (de threads) dentro da grade.
3)TO-DO #2.3.1 e TO-DO #2.3.2 Declare e defina a memória GPU necessária para executar o kernel CUDA. Inclua a transferência dos dados do host para a GPU, se necessário.
4)TO-DO #2.4 Execute o kernel CUDA com os parâmetros correspondentes.
5)TO-DO #2.5.1 Transfira os resultados da GPU para o host. Como alternativa, use uma sincronização para garantir que o host espere que a GPU execute o kernel.