Parallel and distributed algorithms – Examination, session 2 (2 hours)
September 2020

Exercise 1: Basic knowledge (4 points, 15 minutes)

You have to answer as shortly as possible to the following questions.

1. What is the fusion mode in a PRAM CRCW machine?
2. How can you transform the following PRAM CRCW algorithm into a PRAM EREW algorithm?

```
INPUT: T an Array[1...N] of Integer
FOR each PE i∈[2...N] in parallel:
    T[ i ]⟵T[ i ]*T[ i-1 ]
```

3. Is it possible to synchronize threads in CUDA? In which circonstances it works, or doesn't work?
4. Assuming $(1024, 1024, 64)$ being the maximum numbers of threads a given CUDA device supports for dimensions (x, y, z), what is its maximum number of threads per block?

Exercise 2: Design Patterns for HPC (6 points, 25 minutes)

In this exercise we assume the following vectors exist:

- $A = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]$
- $key = [0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3]$
- $map = [14, 12, 10, 8, 6, 4, 2, 0, 1, 3, 5, 7, 9, 11, 13, 15]$

Moreover, the used binary operator is always the integer addition, and we consider only *inclusive SCAN*. What are the results of the following parallel patterns applied to these vectors?

1. GATHER(A, map)
2. SCATTER(A, map)
3. REDUCE(A)
4. SEGMENTED-REDUCE(A, key)
5. SCAN(A)
6. SEGMENTED-SCAN(A, key)

Exercise 3: MLP in Cuda ($\sim$ 10 points, 80 minutes)

A MLP (Multi-Layer Perceptron) is a neural network made with some layers of neurons. The input of the MLP is a vector of scalars, and the output generally a scalar (if the output layer contains one neuron only), or a vector of scalars. In this exercise we consider that the last layer contains one neuron only. Both in the learning step or in decision process, the MLP computes the scalar output of this graph of neurons for a given input.
The layers are fully connected two-by-two: this means that the output of the first (input) layer forms a vector of scalar which becomes the input of each neuron of the next layer. Then, a neuron contains two parts: the first take the vector made by the ouput values of the previous layer (or the input vector for the first layer), and produces a scalar value using a dot product with a weight vector ; then, each neuron contains a given fonction φ, that transforms the scalar resulting from the dot product, producing a scalar as output. Each neuron has a specific vector of weights $\{w^i_{kj}\}$, that are computed during the learning step (with a stochastic gradient descent for instance). By denoting φ the activation function, the neuron number k of layer i computes the following scalar value x^i_k:

$$x^i_k = \varphi\left(\sum_{j=1}^{m} w^i_{kj} x^{i-1}_j\right),$$

where m is the size of the previous layer.

In the following, we assume that the size of the input X is n, the scalar values are `float` values, and the number of layers (hidden layers plus input and output ones) is l (so we consider $l-2$ hidden layers). Each neuron in this exercise is made with the same function for each neuron of each layer, called f.
Each layer has it is own number of neurons. Let us take an example, with $n=2$, $l=4$, layers' size are 2, 3, 2 and obviously 1 for the last one (because it is always the case in this exercise). Here, the neurons of the first layer have a weight vector of size $n=2$, corresponding to the input size. The second layer consists of neurons having a weight vector of size 2, because first layer contains 2 neurons. The layer 3 have weight vectors of size 3, because second layer is made of 3 neurons. And last layer has one neuron wit 2 weights, because layer 3 has 2 neurons.

1. (2 points) Draw the example of the MLP proposed above ($n=2$, $l=4$, size of neurons layers are $2,3,2,1$).
2. (4 points) Write a CUDA kernel that computes the output of a given MLP. In this question we consider that the layers' sizes are less of equal to 32. The inputs of the kernel are the input vector of size n, the weights of each layers' neuron, the number of neurons, the pointer to the result ...
3. (2 points) Modify the previous CUDA kernel considering here that the layers' sizes are less or equal to 256. Which part needs to be modified? How?
4. (2 points) During the learning step, we can compute many values in parallel. In this case, we may compute one value per thread block, or do a fully parallel work. Discuss the two possibilities in terms of efficiency, and implementation (using Thrust for instance). Full implementations are not required here.

And that's all, folks!