Université de Pau et des Pays de l'Adour
Département de Mathématiques
2021-2022

M2-BigData : GPGPU
Chapter 9 – Exercice 1

Objectives

Implement an efficient dot product on GPU.
The definition of dot product is :

$$a \cdot b = \sum_{i=0}^{n} a_i b_i$$

where a and b are vectors of length n.

Instructions

Complete the given code to implement a basic reduction kernel.

First, start to compute dot product for $n = 1024$ elements and 512 threads in 1 single block using the naive thread mapping.

Second, adapt your code to handle vector of any length, using an appropriate number of blocks. In this case, the device result is a vector of length equal to the number of blocks. The block size should be the next multiple of 32 greater than $n/2$, with a maximum of 1024. Then it is staightforward to compute the number of blocks needed to handle the entire dot product. To simplify this exercise, we enforce the block size to be equal to 1024, and n to be greater than 2048. The final reduction of the operation is performed on the CPU.

Third improve the second version with the "better" reduction kernel algorithm with the improved data mapping, and reduced divergence.

Questions

1. How many floating operations are being performed in your reduction kernel ? explain.
2. How many global memory reads are being performed by your kernel ? explain.
3. How many global memory writes are being performed by your kernel ? explain.
4. How many times does a single thread block synchronize to reduce its portion of the array to a single value ?
5. Explain why the third version is better than the second one ?
6. Explain how to get rid of the data transfer of partial results and the host-side final reduction.
7. Describe what changes are needed to handle vector length larger than the maximum thread number in the CUDA grid. Explain.
8. Is it possible that the result changes from an execution to an other ? Explain why.