Lab 2

Programació paral·lela

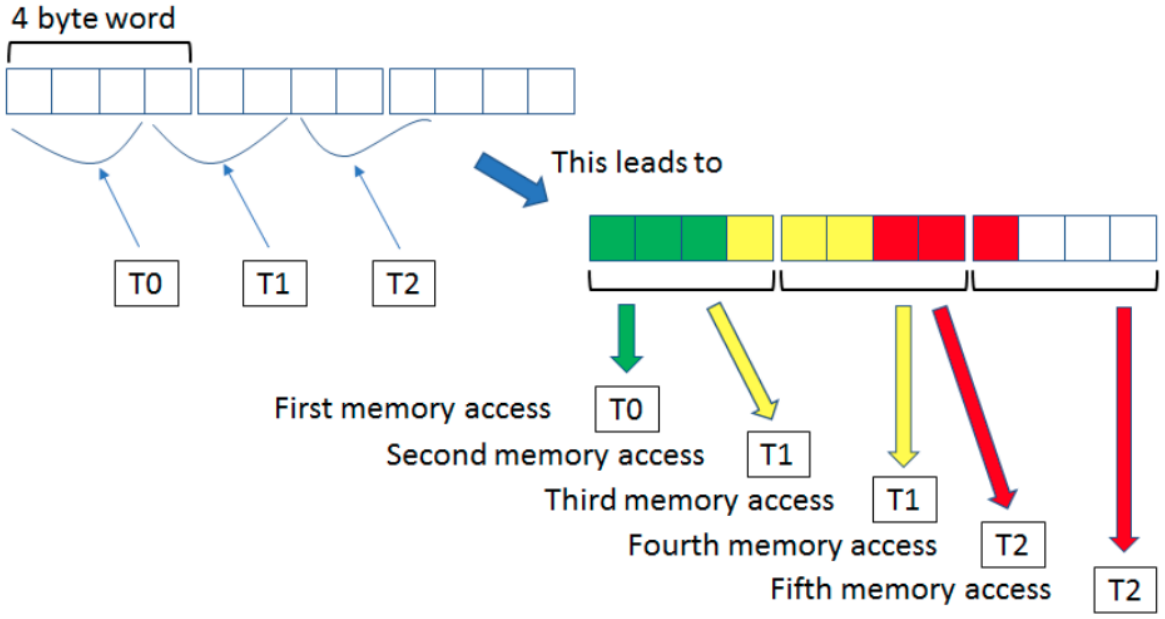
Oscar Amoros
2020-2021

Color space conversion

The description of the lab, and the tasks to perform are all in the google colab document provided along this one. You need to use any Google account in order to open it, or install your own Jupiter Notebook environment with support for CUDA.

Here we give some diagrams to explain what we expect of section 5.

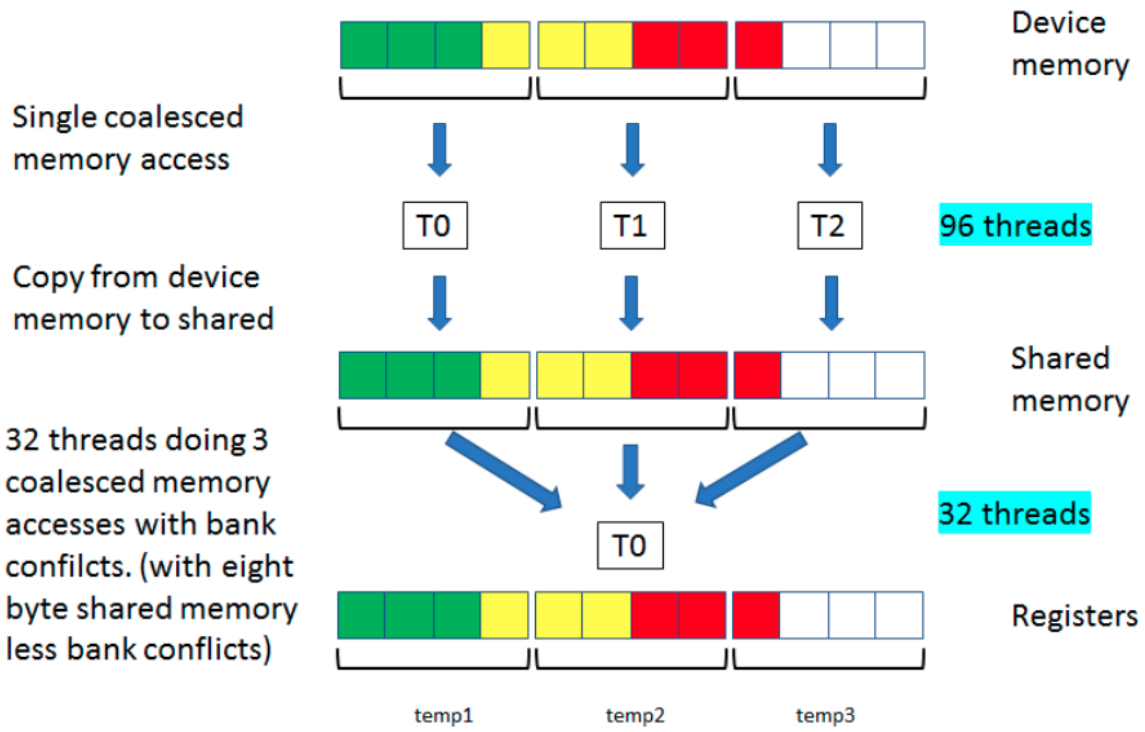
The performance issues in this code for CUDA are the following:

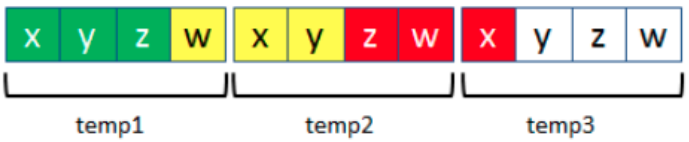


As you can see, the threads from a warp are not doing the coalesced readings, therefore, the number of accesses to the memory is not one per warp, but a lot more.

As a reminder, a warp can perform a single memory access instruction to access up to 32 data elements, at least of 4 bytes each.

Next you have a better solution of this algorithm in CUDA.

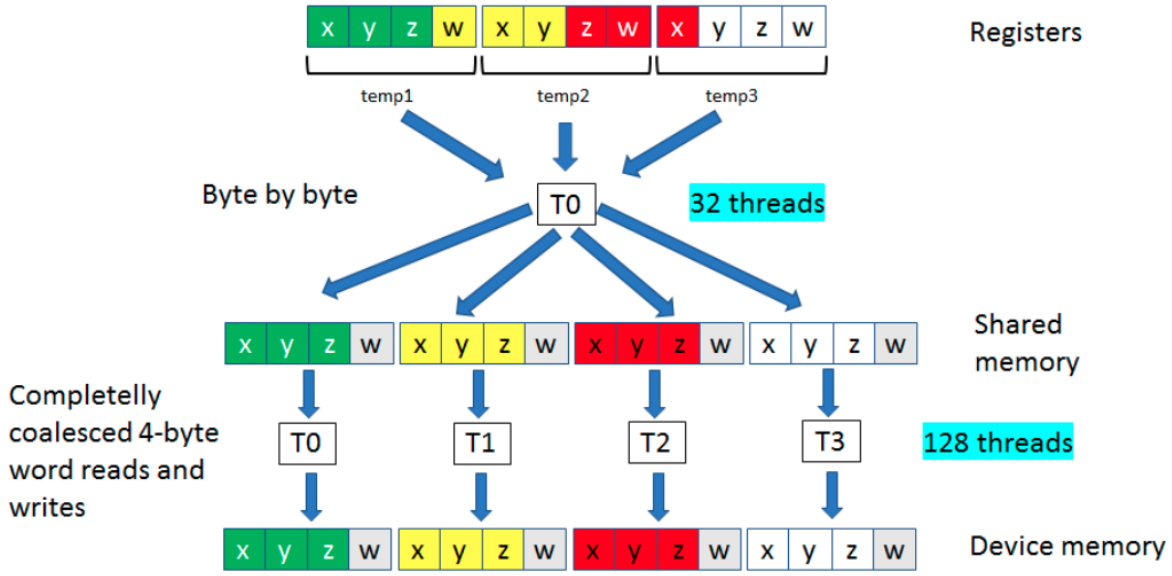




T0

```
pix_write[threadIdx.x * 4]       = make_uchar4(temp1.z, temp1.y, temp1.x, (uchar)0);
pix_write[(threadIdx.x * 4) + 1] = make_uchar4(temp2.y, temp2.x, temp1.w, (uchar)0);
pix_write[(threadIdx.x * 4) + 2] = make_uchar4(temp3.x, temp2.w, temp2.z, (uchar)0);
pix_write[(threadIdx.x * 4) + 3] = make_uchar4(temp3.w, temp3.z, temp3.y, (uchar)0);
```

32 threads generate 128 uchar4 values and write them in shared memory again.



Note that the order of x, y, z and w change with respect to the code provided. You have to use the x, y, z, w ordering corresponding to the code in the google colab.

The delivery date of this lab is the 7th of May.