Pràctica 1

Programació paral·lela

Oscar Amoros
2018-2019

Conversió d’espai de color

Les pràctiques 1a i 1b de programació paral·lela, estan ambdues basades en un algorisme molt senzill de transformació d’espai de color, d’una imatge 4K.

L’espai de color, es el format en el que els píxels de la imatge es representen. En aquest cas, transformem una imatge en espai de color bgr (blue, green, red), en una imatge en espai de color rgba (red, green, blue, alpha).

L’espai de color bgr, es composa de tres canals de color. Blau, verd i vermell. Cada canal, pot tenir un nombre de bits que va dels 8 bits als 16 bits, habitualment. El nombre de bits s’anomena profunditat de color, ja que quants més bits tinguem per canal, més resolució de color tindrem. En el nostre cas, cada canal tindrà 8 bits.

Per enmagatzemar aquests 8 bits, utilitzarem el tipus de dada unsigned char. Per enmagatzemar tots els canals d’un pixel, utilitzarem structs. uchar3 per a bgr i uchar4 per a rgba.

El canal “a” de rgba, s’anomena alpha, i habitualment s’utilitza per especificar la transparencia o opacitat del pixel.

Objectiu de la pràctica 1

En la part “a” de la pràctica es proporciona un codi molt basic en C++, que haureu de paral·lelitzar en OpenMP.

En la part “b” de la pràctica, aquest mateix codi s’haurà de paral·lelitzar en CUDA.

En els dos casos, l’objectiu és accelerar la transformació de l’espai de color. Per tant, no només se us demanarà el codi implementat, si no que es demana una memòria on mostrar els resultats de rendiment amb taules i gràfiques, i una discussió del perquè del rendiment de cada versió del codi.

Cada part (a i b) tenen un enunciat, on es demana una serie de tasques a realitzar.

Objectius específics pràctica 1a

1. Mantingueu sempre una copia intacta de la funció “convertBGR2RGBA” per a poder comparar temps d’execució.
2. Experimenteu amb diferents maneres d’accedir a la memòria a la funció “convertBGR2RGBA” per explotar millor la localitat de les dades.
3. Paral·lelitzeu la funció “convertBGR2RGBA” amb openmp.

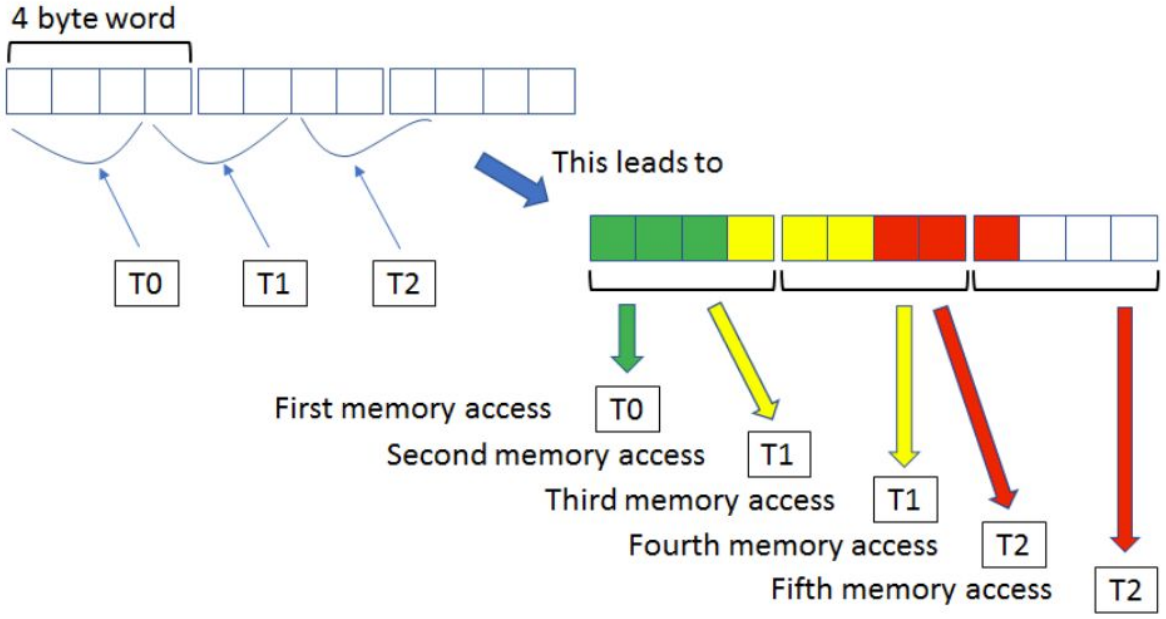
4. Imprimiu de forma correcta l'id de cada un dels threads que executen el codi, utilitzant les funcions proporcionades per OpenMP per coneixer aquest id, i per executar parts de codi de forma ordenada (regiò crítica).
5. Experimenteu amb diferents polítiques d'scheduling. Justifiqueu quina és més eficient i perqué.
6. Comenteu la línea que crida la funció checkResults, i executeu les diferents versions de codi (inclosa la original), amb la comanda time. (punts 2, 3 i 5)
7. Creeu un informe amb aquests resultats en taules i/o gràfiques, explicant perquè creieu que els heu obtingut.

La data d'entrega d'aquest pràctica és el dimecres 13 de Març a les 23h.

Objectius específics pràctica 1b

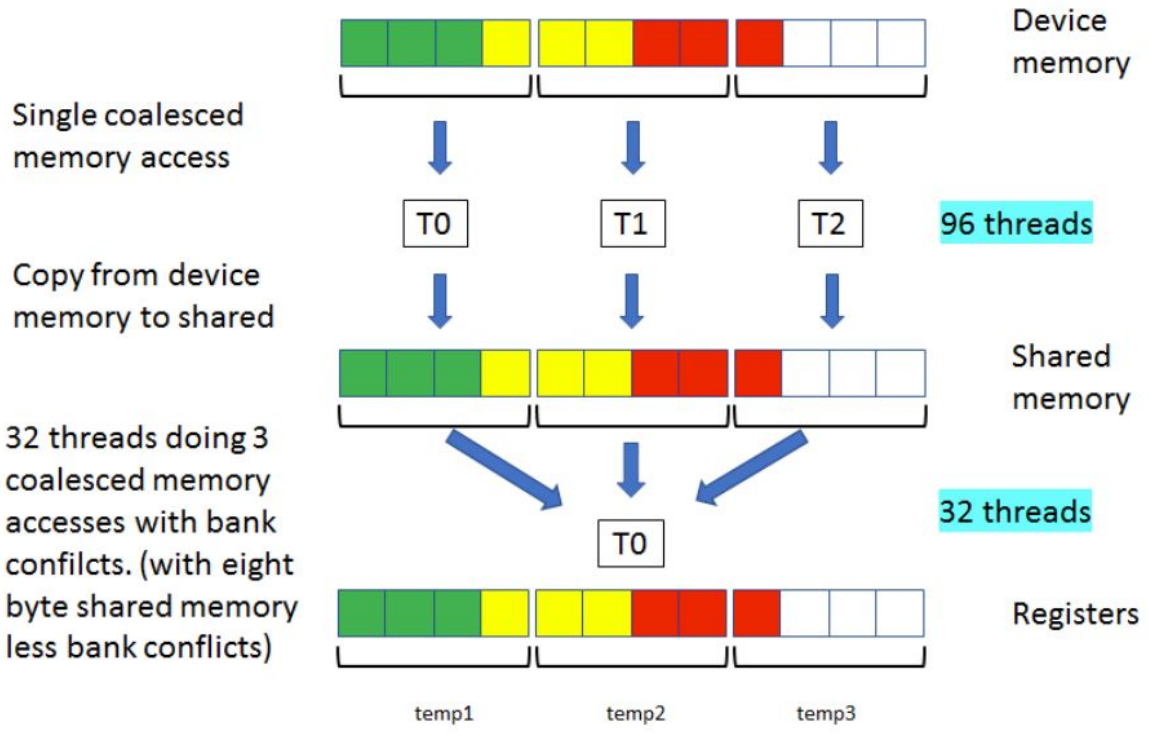
1. Implementeu una versió basica de l'algorisme en CUDA, amb el doble bucle for.
2. Implementeu una versió basica de l'algorisme en CUDA, amb un sol bucle.
3. Intenteu optimitzar les accesos a memoria d'alguna manera basica, sense utilitzar shared memory.
4. Optimitzeu els accesos a memoria utilitzant shared memory.
5. Intenteu aplicar la optimització explicada a continuació.
6. Com a la pràctica anterior, creeu un informe, i en aquest cas, compareu el rendiment de la millor versió en serie i la millor versió amb OpenMP, amb les versions que feu amb CUDA.

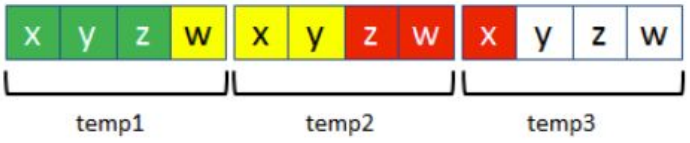
El problema de rendiment d'aquest codi en CUDA és el següent:



Com podeu veure, els threads d'un Warp no estan fent les lectures coalescents, i per tant, el nombre d'accessos a memoria no es un per warp, si no molts més.

A continuació, teniu ilustrada la solució més eficient al problema. La comentarem a classe.

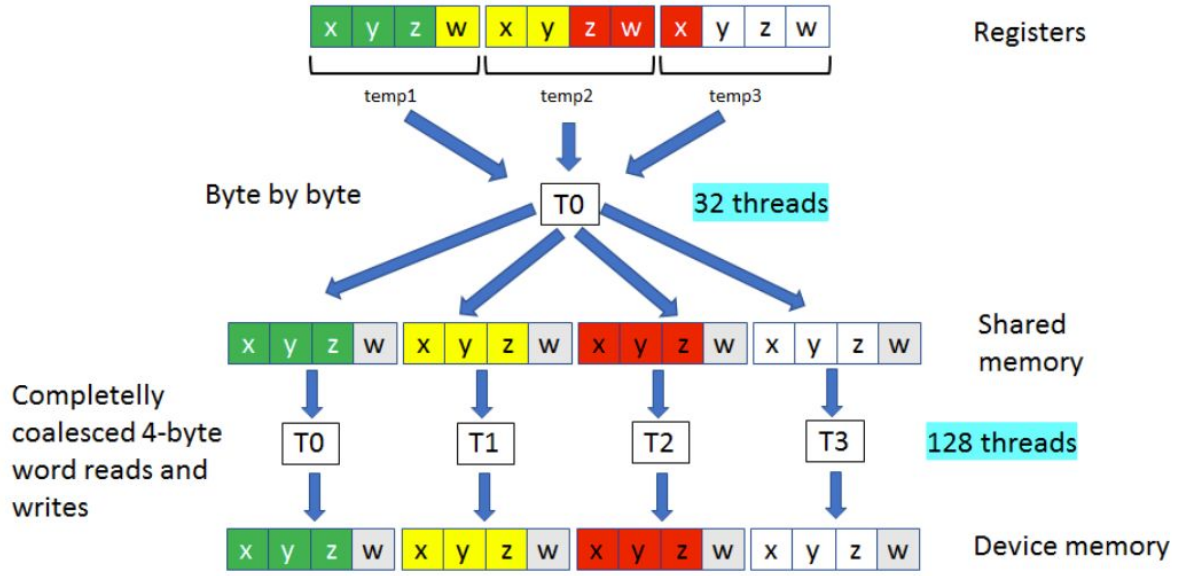




T0

```
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.



La data d'entrega d'aquesta pràctica és el dimecres 8 de Maig.