Lab 1

Programació paral·lela

Oscar Amoros
2019-2020

Conversió d’espai de color

Les pràctiques 1 i 2 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 brg (blue, red, green), en una imatge en espai de color rgba (red, green, blue, alpha).

L’espai de color brg, es composa de tres canals de color. Blau, vermell i verd. 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 emmagatzemar tots els canals d’un pixel, utilitzarem structs. uchar3 per a brg 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 les pràctiques 1 i 2

A la pràctica 1 es proporciona un codi molt basic en C++, on haureu de treballar els conceptes de: flags de compilació, alineació de les dades, i localitat de dades i paral·lelitzar el codi amb OpenMP.

A la pràctica 2, 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.

Es puntuaran principalment els **raonaments sobre cada un dels resultats demanats**, i no només el resultat més ràpid. **SUSPENDREU** la práctica si només presenteu la versiò de codi més ràpida. Per tant, poseu cura en entregar el codi corresponent a cada punt, junt amb explicació de resultats. Podeu preguntar a classe.

Objectius específics pràctica 1

Utilitzeu les màquines de l'aula per a fer la pràctica. Si voleu afegir experiments amb altres màquines, ho podeu fer, però heu de executar les proves a l'aula en qualsevol cas.

Entregueu el codi amb la funció convertBRG2RGBA original inclosa.

1. Compileu i executeu el codi tal com se os entrega.
    a. Compileu-lo amb la comanda: make
    b. Executeu-lo: ./main
    c. Anoteu el temps en micro segons "us"
    d. Mireu el codi i enteneu-lo.
2. Modifiqueu el Makefile, i experimenteu amb els flags d'optimització del compilador. Aquests flags son -O, -O2, -O3 i -Ofast.
    a. Llegiu la documentació sobre aquests flags.
    b. Hi ha algún risc en utilitzar aquests flags? Busqueu a Google i/o pregunteu a classe.
    c. Utilitzant -Ofast, compileu: make
    d. Executeu: ./main
    e. Anoteu el temps i calculeu l'Speedup respecte al punt 1. Expliqueu, segons el que hem vist a teoria, que pot estar passant.
3. Mantenint el flag -Ofast:
    a. Creeu una funció convertBRG2RGBA_2.
    b. Aquesta funció ha de contenir modificacions que millorin la localitat de les dades. Expliqueu els canvis i els speedups obtinguts.
    c. Si els temps obtinguts son menors que 1s, augmenteu EXPERIMENT_ITERATIONS a 1000.
4. Mantenint les optimitzacions del punt 3:
    a. Segons el que hem vist a teoria, quin problema pot tenir l'struct uchar3?
    b. Canvieu la definició de uchar3 per aquesta:

```
struct _uchar3 {
   uchar x;
   uchar y;
   uchar z;
} __attribute__ ((aligned (4)));
```

    c. Torneu a executar i mesureu el temps.
    d. Busqueu a google que es __attribute__ i que fa concretament aligned(4). Pregunteu a classe. Expliqueu a la memoria que heu entès.
5. Mantenint les optimitzacions anteriors, creeu una funció "convertBRG2RGBA_3" i paral·lelitzeu-la amb OpenMP:
    a. Quin bucle va millor paral·lelitzar?
    b. Quin scheduling es millor? Feu taules amb resultats, modificant la mida de la imatge, el numero de iteracions, i el chunk size del scheduling. Justifiqueu el raonament amb aquestes taules.

    c. Canvia alguna cosa definir variables private i shared?
6. Paral·lelitzeu el bucle que itera EXPERIMENT_ITERATIONS, executant la funció “convertBRG2RGBA_2”. Com compara amb el punt 5? Perquè creieu que passa?
7. Imprimiu l’id de cada un dels threads que executen el codi, utilitzant les funcions proporcionades per OpenMP per conèixer aquest id. Assegureu-vos que el codi és correcte i assegureu que executeu les parts de codi de forma ordenada (regiò crítica).
8. Creeu un informe amb aquests resultats en taules i/o gràfiques, i responent a totes les preguntes.

La data d’entrega d’aquest pràctica és el dimecres 11 de Març a les 23h.