ARQUITECTURA DE COMPUTADORS
PRÀCTIQUES DE SUPERCOMPUTACIÓ

PRÀCTICA 3: PARAL·LELITZACIÓ D'ALGORISMES

Objectius

- Distribució de l'execució d'un programa a la grid.
- Paral·lelització d'algorismes d'ordenació i cerca.
- Generació d'imatges a la GPU.
- Avançar en els conceptes i aplicacions de la programació en CUDA.

Requisits:

Per a poder realitzar aquesta pràctica, cal disposar d'una gràfica NVIDIA que suporti CUDA.

Actualment podeu comprovar si la vostra NVIDIA és compatible a:

https://developer.nvidia.com/cuda-gpus

La pràctica es farà amb Linux.

Es pot consultar la documentació de CUDA a http://docs.nvidia.com/cuda/index.html

1) Paral·lelització a la grid

Quan executem un programa a la GPU, idealment, volem que la distribució de l'execució sigui el més uniforme possible, a fi d'evitar la creació de threads de més, i la copia innecessaria del kernel com a blocks. També és important saber que és el que volem executar en paral·lel, ja que no tots els algorismes resulten eficients a la GPU, sinó que una implementació a la CPU pot ser més que suficient. Recordem l'exemple de l'imatge, on cada píxel es tracta en paral·lel, en una grid organitzada en blocks i threads en funció de les dimensions de l'imatge. En aquests casos, la paral·elització és recomenable per a reduir els temps d'execució.

EXERCICI 1:

Obriu el fitxer organitza_grid.cu i mireu les instruccions presents com comentaris al codi. El que heu de fer en aquest exercici és el següent: Donada una estructura de *NxN* elements, organitzeu al vostre grid mitjançant una distribució uniforme en blocks i threads, de mode que cada element de l'estructura NxN sigui tractada per un sol thread, distribuits uniformament a un determinat nombre de blocks. Heu de treballar en 2 dimensions, ja que l'estructura, com podria ser una imatge, és de *NxN*.

a) Printeu per pantalla com quedaría la grid, de mode que cada block tindrà un valor númeric, que representarà l'index del block, i cada block tindrà un nombre de threads on cada thread del block tindrà el mateix valor númeric que el block. És a dir:

0 0 1 1
0 0 1 1
2 2 3 3
2 2 3 3

Representa una grid amb 4 blocks de 4 threads cada block.

b) Modifiqueu el valor de *N* amb diferents valors, i comenteu quines diferencies hi han.

EXERCICI 2:

Obriu el fitxer cerca.cu.

a) Heu d'implementar un senzill programa al host i al device que, donat un array de valors random, i un valor *v* introduit al terminal, cerqui al array si el valor *v* és present a l'array. El vostre programa ha de retornar per pantalla si el valor es present a l'array random.

b) Observeu els temps d'execució de la cerca al host i al device. Per què creieu que retorna aquests valors?

c) Ara fem una modificació al nostre programa. Afegiu tant a la funció de cerca del host com a la funció de cerca del device, després del càlcul de cada índex, un bucle amb un conjunt d'operacions. Exemple:

```
for(int i = 0; i <= 1000000; i++)
        b = (b*70)/3;
```

O qualsevol operació amb un cert pes. Executeu de nou el vostre programa i observeu els temps. Hi ha alguna diferència? Si és així, per què ho creieu?

EXERCICI 3:

Obriu el fitxer bubblesort.cu.

a) Heu d'implementar l'algorisme d'ordenació Bubble Sort que vam fer a la primera pràctica, però en aquest cas a la GPU.

b) Observeu que la crida al kernel està inclosa a un bucle *for*. Per què ho creieu? Que passa si eliminem aquest bucle? Per què el bucle itera fins *N*2*?

c) (**Opcional**) Podeu comprovar les diferències en el temps d'execució entre la versió seqüencial a la CPU que vau implementar a la primera pràctica, i la versió en paral·el implementada en aquesta. Proveu per diferents tamanys del vector i comenteu.

2) Conjunt de Mandelbrot

El conjunt de Mandelbrot representa un dels fractals més coneguts.

Consulteu sobre els fractals a http://en.wikipedia.org/wiki/Fractal i sobre el conjunt de Mandelbrot a http://en.wikipedia.org/wiki/Mandelbrot_set

Llegiu la documentació sobre els fractals i sobre el conjunt de Mandelbrot.

EXERCICI 4:

Obriu el fitxer mandel.cu que conté el codi per calcular els llindars del conjunt de Mandelbrot al host i també té l'esquelet del programa equivalent per al dispositiu CUDA. A la pàgina de la wikipedia hi podeu trobar el pseudocodi en què està basat aquest programa.

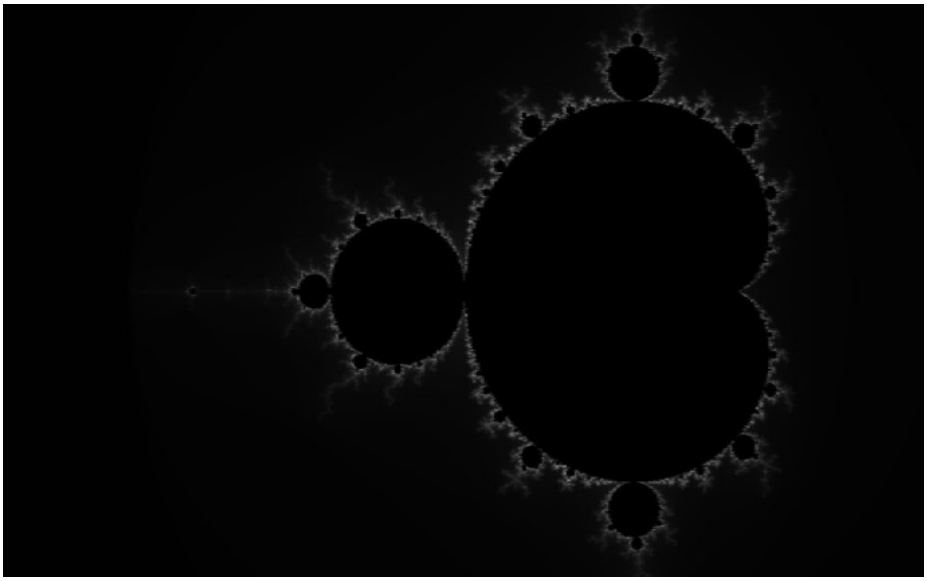
a) Complementeu el programa anterior amb la funció que s'executarà al dispositiu i la funció que configura i executa el grid al dispositiu. La configuració del grid inicial serà de 16x16 threads a cada bloc i tants blocs com sigui necessari per a completar la dimensió de la imatge, que en el cas inicial, serà de 5120x5120 píxels i 320x320 blocs per grid.

b) Afegiu el codi que verifiqui que el resultat al host i al device són idèntics.

c) Proveu diferents valors de dimensió del grid (canvieu el número de threads per bloc i de blocs per grid) i analitzeu el temps que tarda en executar-se la funció per a cada configuració de grid. Executeu diverses vegades la funció i feu la mitjana del temps per minimitzar errors.

Sortida del programa:

La figura següent representa la sortida del programa:



ENTREGA DE LA PRÀCTICA

Les pràctiques es realitzaran en grups de 3 persones.

La pràctica s'haurà de lliurar a l'aula global i haureu d'entregar un fitxer comprimit que contingui els fitxers de codi demanats amb el codi que heu afegit i, amb els fitxers que heu creat i una memòria amb les vostres respostes en format pdf. Només un dels tres membres del grup ha de pujar la pràctica.

El nom del fitxer que pengeu a l'aula global:

GP_p3_NIA1_NIA2_NIA3.zip

La vostra memòria ha de contenir:

- Nom, cognoms i NIA dels 3 membres del grup.
- Les respostes a les preguntes formulades .
- Les sortides de les vostres execucions (si és necessari) .

Es valorarà positivament la claredat i precisió de les respostes així com la qualitat del codi.