High Performance Computing
Departamento de Ingeniería en Informática
LAB3 : Paradigma SIMT - Cuda

1 Objetivo

El objetivo de este laboratorio es implementar un simulador paralelo de la difusión de una onda según la equación de Schroendinger, usando CUDA como tecnología de paralelización.

2 La ola

2.1 La ecuación de Schroedinger

Suponga que deseamos simular cómo se difunde una ola en un medio, por ejemplo, acuoso, a partir de un estado inicial. Para simplificar, discretizamos el dominio de difusión en una grilla de $N \times N$ celdas. En particular, la matriz H contiene la altura de la onda en cada posición de la grilla. Además, como el proceso de difusión es dependiente del tiempo, H^t_{ij} es la altura, ya sea positiva o negativa, de la onda en la celda (i, j), en tiempo t. Una solución para la ecuación de Schroendinger se obtiene mediante la siguiente iteración de diferencias finitas:

$$H^t_{i,j} = 2H^{t-1}_{i,j} - H^{t-2}_{i,j} + c^2\Big(\frac{dt}{dd}\Big)^2\Big(H^{t-1}_{i+1,j} + H^{t-1}_{i-1,j} + H^{t-1}_{i,j-1} + H^{t-1}_{i,j+1} - 4H^{t-1}_{i,j}\Big) \tag{1}$$

donde c es la velocidad de la onda en el medio, dt es el intervalo de tiempo con que avanza la simulación y dd es el cambio en la superficie. Para este lab, usaremos $c = 1.0$, $dt = 0.1$, y $dd = 2.0$.

2.2 Condiciones de borde y especiales

La **condición de borde** para este problema es cero, es decir la altura en las celdas de los bordes del dominio permanece siempre con valor cero,

$$H^t_{0,j} = H^t_{i,0} = H^t_{N-1,j} = H^t_{i,N-1} = 0 \quad \forall i, j, t$$

También, la iteración para $t = 1$, se usa la siguiente ecuación, en vez de (1):

$$H^t_{i,j} = H^{t-1}_{i,j} + \frac{c^2}{2}\Big(\frac{dt}{dd}\Big)^2\Big(H^{t-1}_{i+1,j} + H^{t-1}_{i-1,j} + H^{t-1}_{i,j-1} + H^{t-1}_{i,j+1} - 4H^{t-1}_{i,j}\Big) \tag{2}$$

En la ecuación anterior se respeta la condición de borde del problema.

La **condición inicial** del sistema está dada por un impulso en celdas centrales de la grilla. Por ejemplo, para una ola cuadrada inicial, tenemos:

$$H^0_{i,j} = 20 \quad \forall\ 0.4N < i < 0.6N,\ \ 0.4N < j < 0.6N$$

y $H^0_{i,j} = 0$ para todas las otras celdas.

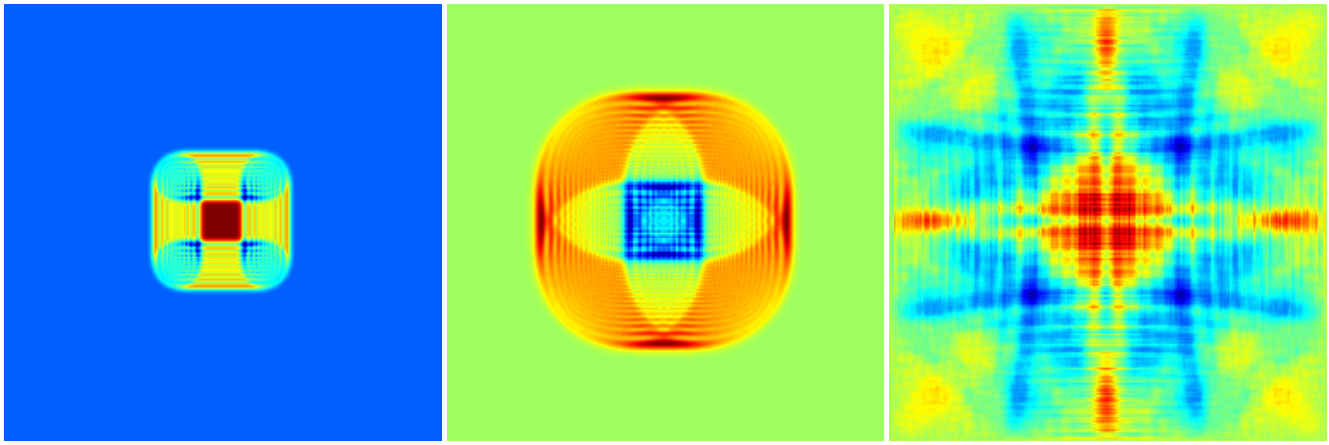

Figure 1: Imágenes de Schroedinger para $t = 300, 1000$, y 10000, para una grilla de 256×256.

3 El programa y la estrategia de paralelización

CUDA es la herramienta perfecta para este lab, pues permite paralelismo a nivel de granularidad muy fino. Basta crear una grilla 2D en la cual la actualización del valor de cada posición de la matriz la realice una hebra. That's it!

```
$ ./wave -N tamaño_grilla -x tamaño_bloque_en_X -y tamaño_bloque_en_Y -T número_de_pasos
          -f archivo_de_salida -t iteracion_de_salida
```

Note que el número de hebras NO necesariamente divide exactamente al tamaño de la grilla. En el archivo de salida se almacenará la grilla de la iteración indicada por la opción `t`, en formato binario, es decir H^t.

El **tipo de dato para la grila** es `float`.

Las imágenes de la Figura 1 muestran ejemplos de salida para una grilla de 256×256, en diversas iteraciones.

4 Rendimiento computacional

Para este programa usted medirá el rendimiento computacional en uno de los computadores del lab. Estos computadores tienen una GPU Quadro M2000.

1. Tiempo de ejecución (wall-clock) para diferente tamaños de grilla: 512x512, 1024x1024, 2048x2048, 4096x4096. En este caso elija para todos un mismo tamaño de bloque.

2. Ocupancia del o los kernels en cada caso.

3. Para la grilla de 2048x2048, medir el tiempo de ejecución para diferentes tamaños de bloques: 16x16, 32x16, 32x32.

5 Matlab

Se recomienda programar este problema, primero en Matlab, y luego en C con CUDA. Esto le ayudará a tener en poco tiempo un prototipo secuencial funcionando, y además usar la facilidades de graficación de Matlab.

Si `A` es una matriz de $N \times N$, entonces el comando en matlab usado para generar las imágenes de ejemplo, es

```
>> imagesc(A); axis('off');axis('square');
```

Si desea escribir la matriz en formato binario y en floats, entonces

```
>> f = fopen('miarchivo.raw', 'w');
>> fwrite(f, A', 'float');
>> fclose(f);
```

Note que la función `fwrite()` recibe como parámetro la traspuesta de la matriz `A`, pues Matlab al igual que Fortran almacena las matrices por columnas, no por filas.

6 Entregables

Tarree, comprima y envíe a `fernando.rannou@usach.cl` al menos los siguientes archivos:

1. `Makefile`: archivo para make que compila los programas
2. `wave.cu`: archivo con el código. Puede incluir otros archivos fuentes.
3. `wave.pdf`: informe breve de su solución. Estrategia de paralelización y rendimiento computacional.

Fecha de entrega:
lunes 25 de noviembre antes de las 23:59 hrs.