Informe laboratorio 1 HPC

Integrantes:
Esteban Cruces 20.201.381-3
Esteban Lopez 19.837.971-9

1. Solución planteada

Para la solución de este proyecto, se parte ingresando los datos necesarios mediante getopt, y la ejecución por línea de comandos debe ser con el siguiente formato "*./wave -N tamaño_grilla -x tamaño_bloque_en_X -y tamaño_bloque_en_Y -T n´umero_de_pasos -f archivo_de_salida".*

Dentro del código, inicialmente se crean 3 arreglos del tipo *"float * "*, donde el primer vector representa a la grilla a manejar, mientras que los otros 2 vectores representan a la grilla en el tiempo t-1 y t-2 respectivamente. Para la inicialización de esta grilla existe la función *"initializeGrid"*, la cual retorna un objeto del *"float * ",* con los valores iniciales seteados en 0.

Luego, se crea la función *"schroedinger"* que recibe los vectores creados, en dicha función se encuentra dentro de un *"for"* se recorre la cantidad -T ingresada. Una vez dentro de la función´*"schroedinger"* nos encontramos con que posee 3 casos dentro, en los cuales, si el tiempo T es 0, se setean todos los valores en 0, a excepción de los que se encuentren entre el 40 al 60 % del largo de la grilla, dichos valores en esta iteración se les asigna el valor 20; luego, para cuando el valor T es 1, se ejecuta la siguiente función:

$$H_{i,j}^{t} = H_{i,j}^{t-1} + \frac{c^2}{2}\left(\frac{dt}{dd}\right)^2\left(H_{i+1,j}^{t-1} + H_{i-1,j}^{t-1} + H_{i,j-1}^{t-1} + H_{i,j+1}^{t-1} - 4H_{i,j}^{t-1}\right)$$

Lo que sigue es que para los valores de T desde 2 hasta el valor asignado inicialmente, se ejecuta la siguiente función, que depende de los 2 valores anteriores:

$$H_{i,j}^{t} = 2H_{i,j}^{t-1} - H_{i,j}^{t-2} + c^2\left(\frac{dt}{dd}\right)^2\left(H_{i+1,j}^{t-1} + H_{i-1,j}^{t-1} + H_{i,j-1}^{t-1} + H_{i,j+1}^{t-1} - 4H_{i,j}^{t-1}\right)$$

Finalmente, luego de realizar las funciones mencionadas, se procede a ejecutar la función *"save"* en la cual se recorre el arreglo que representa a la grilla, y esta se procede a escribir en un archivo de salida, con el nombre ingresado en -f de la línea de ejecución. A continuación, se procede a liberar la memoria asignada a esta grilla, mediante la función *"free".*

2. Estrategia de paralelización

Para la paralelización de este proyecto, inicialmente se asignan las dimensiones de la grilla y los bloques ingresadas por línea de comandos en -N, -X e -Y, esto se hace en el *"main"* mediantes las líneas:

```
    dim3 blocksize;
    dim3 gridsize;
    blocksize.x = options.x;
    blocksize.y = options.y;
    gridsize.x = options.grid_size / blocksize.x;
    gridsize.y = options.grid_size / blocksize.y;
```

Luego, se crean los arreglos con los que se trabajarán y se reserva la memoria necesaria en cuda, con:

```
float *grid_0 = initializeGrid(options.grid_size);
float *d_grid_0;
cudaMalloc((void **) &d_grid_0,
options.grid_size*options.grid_size*sizeof(float));
cudaMemcpy(d_grid_0, grid_0, options.grid_size * options.grid_size *
sizeof(float), cudaMemcpyHostToDevice);
```

Con la cual, se llama y se paraleliza la función *"schroedinger"*, en donde por cada hebra se toma las coordenadas de esta, tanto en Y como en X, con las cuales se trabajan las posiciones de la grilla, dichos valores se obtienen de la siguiente forma:

```
int blocksize = blockDim.y * blockDim.x; // number of threads in a TB
int blockId = gridDim.x * blockIdx.y + blockIdx.x; // unique block Id
int i = blockDim.x*blockIdx.x + threadIdx.x;
int j = blockDim.y*blockIdx.y + threadIdx.y;
```

Luego de que se "recorran" todas las posiciones de la grilla, el programa espera para sincronizar, y de esta forma "recorrer" en el siguiente valor de T.

Finalmente se escribe la grilla resultante y a su vez, se libera la memoria asignada tanto para el host como para el device.

3. Rendimiento computacional

Las especificaciones del computador utilizado son las siguientes:

- CPU: AMD® Ryzen 5 5600x 6-core processor × 12
- RAM: 12GB DDR4
- GPU: Nvidia Geforce RTX 3060 12GB

Se realizaron pruebas para diferentes tamaños de grilla de 32x32 y 1000 iteraciones, los resultados fueron los siguientes:

Tamaño Grilla (NxN)	Tiempo de ejecución (wall-clock) (segundos)
512x512	0.125117
1024x1024	0.412683
2048x2048	1.587172
4096x4096	6.277119

Además, se realizaron pruebas con diferentes combinaciones de tamaño de bloques para la grilla de tamaño 2048x2048.

Tamaño Bloque(XxY)	Tiempo de ejecución (wall-clock) (segundos)
16x16	0.857124
32x16	1.703550
32x32	1.593277