2019-2020

El juego de la vida

Práctica 1 CUDA versión por bloques

Javier Muñoz Bulnes, 02744311C

Rubén Calvo Cabello, 09100674B

MEMORIA GLOBAL EN VARIOS BLOQUES

Esta parte de la práctica es similar a la primera, el funcionamiento es el mismo, con la misma matriz original y auxiliar.

El inicio del programa es el mismo, pidiendo la información sobre el tamaño de la matriz y el modo de ejecución, y llenando la matriz de valores de forma aleatoria (entre unos y ceros).

Las diferencias entre esta parte y la anterior surgen debido a la separación de todos los hilos de la matriz (recordemos, uno por celda) en diferentes bloques. Para calcular bien el número de bloques se utiliza la función del host pasarAMult16() que recibe un número y lo agranda hasta llegar al siguiente múltiplo de 16. De esta forma, y dado que los bloques de hilos tienen un tamaño de 16x16, se puede calcular el número de bloques necesarios para la ejecución, dividiendo el nuevo valor del lado y el largo de la matriz entre 16. El nuevo número es siempre mayor o igual al número introducido por el usuario, pero esto no es problema para la ejecución del juego, debido a que sólo los hilos dentro del rango generado por los parámetros introducidos por el usuario realizarán operaciones. El resto de los hilos no realizarán ninguna operación.

El funcionamiento principal del juego es el mismo, con el bucle que imprime la matriz y la llamada al kernel en cada turno, también conocido como generación. Un cambio realizado en la función llamadaCelula() es el método usado para calcular la posición del hilo en la matriz. Esto es debido a que hay que hay que tener en cuenta los diferentes bloques. Esto resulta en unas formulas como las siguientes: threadIdx.x + blockIdx.x \* blockDim.x para el cálculo del número de fila.

threadIdx.y + blockIdx.y \* blockDim.y para el cálculo del número de columna.

También se puede observar un cambio al llamar a la función cambiarEstado(). Ahora, debido a que tenemos hilos que no formaban parte de la matriz creada por el usuario, hay que comprobar que el hilo esté dentro de la matriz original. Si está dentro, es decir, si el valor de su columna es menor al valor del número de columnas propuesto por el usuario, y el valor de su fila es menor que el valor del número de filas propuesto por el usuario, se llama a cambiarEstado(), que a su vez llama a comprobarVecinos() para que le devuelva el número de vecinos vivos alrededor de la célula correspondiente al hilo que ha hecho la llamada. Una vez tiene este valor, cambiarEstado() comprueba el estado actual del hilo y lo modifica según las reglas del juego de la vida, ingresando el nuevo valor en la matriz auxiliar.

Una vez completada la actual generación, el bucle da otra pasada y se repite las acciones mencionadas anteriormente, hasta que el usuario termine el programa.