Universidad de Alcalá

Escuela Politécnica Superior

Universidad de Alcalá

PRÁCTICA 1 – 16384

Ampliación de programación avanzada

Grado en Ingeniería Informática – Curso 2018/2019

Eduardo Graván Serrano – 03212337L

Marcos Barranquero Fernández – 51129104N

INTRODUCCIÓN

La práctica consiste en desarrollar un juego similar al 2048, aprovechando la capacidad de cómputo de la GPU.

El proyecto entregado puede realizar los cálculos de 2 formas:

- Utilizando 1 sólo bloque con (FILA, COLUMNA) hilos.
- Utilizando FILA bloques con (1, COLUMNA) hilos.

Junto con la práctica se entregan 2 archivos: cada uno hace los cálculos de una forma u otra.

QUÉ SE ENTREGA

Entregado:

Memoria global y 1 bloque	Entregado
Memoria global y varios bloques	Entregado
Memoria compartida	No entregado
Librería gráfica	No entregado
Teselación	No entregado

Otros aspectos a valorar:

Implementación de niveles de dificultad	Si
Lectura de características y ajuste de programa	Si
Ejecución manual y automática	Si
Inicialización aleatoria de matrices	Si
Documentación y comentarios	Si

Respecto a memoria global, aunque no lo hemos implementado, sabríamos hacerlo en caso de tener más tiempo. Presentamos el algoritmo que habría que realizar. Podría hacerse para cualquier desplazamiento, utilizaré como ejemplo el **desplazamiento a la derecha**.

Lanzamiento del kernel de suma y desplazamientos:

1. El bloque contiene COLUMNAS hilos.
2. Para cada hilo, tendrá acceso a la posición [FILA][COLUMNA], donde FILA sería blockIdx y COLUMNA threadId. De esta forma, cada hilo del mismo bloque tendrá acceso a la misma fila.

Ejecución del kernel de suma y desplazamiento:

1. Traer **fila** a memoria compartida. (Si fuese desp. Vertical, habría que traer columna).
2. Sincronizar para que se rellene toda la fila.
3. Comprobar para cada elemento si se debe sumar, sobre memoria compartida. De la misma forma que en global, pero en lugar de comprobar toda la matriz, solo debería comprobar los elementos que estén a su derecha.

4. Sincronizar para que comprueben el resto de hilos del mismo bloque.
5. Realizar multiplicaciones y ponerse a 0, o quedarse igual.
6. Copiar fila de memoria compartida sobre un vector auxiliar de mismo tamaño, también en memoria compartida.
7. Para el desplazamiento:
    a. Ver nº de 0s que tengo a la derecha.
    b. Escribir valor en fila auxiliar[columna + nº de 0s].
8. Copiar el resultado (fila auxiliar) en la fila de la matriz.

CARACTERÍSTICAS COMUNES A CADA ARCHIVO

El juego puede lanzarse con 4 argumentos:

- Juego manual (jugará el usuario) o automático (jugará la máquina)
- Juego fácil o difícil, generando menos o más seeds.
- Nº de filas y Nº de columnas.

Para cada juego, se cargan las siguientes características:

- Movimientos: nº de movimientos realizados.
- Puntuación: puntuación por partida.

La matriz empieza con 15 semillas si está en modo fácil, o 8 en modo difícil.

Se considera fin de juego cuando una seed llega tras sumarse a OBJETIVO, que es 16384.

EJECUCIÓN SECUENCIAL

El programa se ejecuta siempre de la siguiente forma secuencial:

1. Se leen los argumentos de lanzamiento. SI son incorrectos se muestra mensaje de error, en caso contrario se ajustan los parámetros del juego.
2. Se muestra ventana de bienvenida, y las características de tu gráfica.
3. Se pregunta por tu nombre.
4. Se pregunta si se quiere cargar partida. De ser así, se carga.
5. En caso contrario, se genera una matriz aleatoria.
6. Se muestran instrucciones, y se lanza la partida.
7. Se realizan desplazamientos hacia arriba, abajo, izquierda, o derecha. Buscando sumar las seed de valor idéntico.
8. Para cada movimiento, se muestra el número de movimientos y su puntuación.
9. Si alguna de esta seeds llega a objetivo, se muestra mensaje de fin de partida y pregunta si se quiere echar otra.
10. Si se pierde, se resta una vida y se pregunta si se quiere volver a jugar.

EXPLICACIÓN MODULAR

El programa podría dividirse en los siguientes módulos:

GUARDAR Y CARGAR PARTIDA

Se tienen dos métodos para guardar y cargar partida.

- Guardado: consiste en escribir la matriz en un archivo .txt con el nombre del jugador pedido.
- Cargado: consiste en leer la matriz de un fichero .txt con el nombre del jugador.

Además, se guarda la puntuación, el número de vidas restantes y el número de movimientos.

Si se intenta cargar una matriz con filas y columnas distintas a las especificadas en los argumentos, se muestra un mensaje de error.

MOSTRAR CARACTERÍSTICAS DE LA GRÁFICA

Se consigue imprimiendo por pantalla las líneas de cudaDeviceProp, asociado a las propiedades de la gráfica del PC.

GET ELEMENTO Y SET ELEMENTO

Permiten acceder a elementos de la matriz especificando la matriz, la fila y la columna.

FUNCIONES AUXILIARES DEL HOST

El host puede lanzar las siguientes funciones:

- Rellenar matriz con 0s: rellena una matriz con 0s.
- Inicializar matriz: dada una matriz de 0s, escribe las semillas aleatoriamente, en función de la dificultad de juego establecida.
- Nueva semilla: genera y añade una nueva semilla a la matriz dada.
- estaLlena: devuelve true si la matriz está llena.
- finJuego: devuelve true si la matriz que recibe ha llegado a OBJETIVO.
- Pintar Matriz: imprime la matriz por pantalla.
- Copiar matriz: copia el contenido de una matriz a otra.

DESPLAZAR MATRIZ (HOST)

La función de desplazar matriz contiene la lógica para sumar en una dirección y desplazar todo en esa fila o columna a esa dirección.

Su ejecución podría resumirse en el siguiente algoritmo:

1. En el main se reserva espacio en DEVICE para las matrices.
2. Se copia la matriz actual en DEVICE, junto a una matriz vacía donde se escribirá el resultado.
3. Reservar espacio en memoria para las variables de desplazamiento y puntuación.
4. Crear matrices de bloques y malla de bloques.
5. Llamada al kernel de sumas en la dirección indicada.
6. SYNCRONIZE
7. Copiar matriz de salida en matriz de entrada.
8. Establecer matriz de salida a 0s.
9. Mientras, matriz de salida sea distinta a matriz de entrada.
    a. Desplazar matriz de entrada sobre matriz de salida.
    b. Comprobar si son iguales.
    c. Si no lo son, copiar matriz de salida en entrada.
    d. Establecer matriz de salida a 0s.
10. Comprobar si está llena.
11. Si no está llena, se añade una nueva semilla.
12. Comprobar si se ha perdido la partida. SI es así, mostrar mensaje de perdedor y salir.
13. Mover matriz final al host y esperar a siguiente movimiento.

KERNELS AUXILIARES

- Kernel Copiar matriz: copia una matriz a otra desde device de forma eficiente.
- Kernel set matriz 0: establece todas las posiciones de una matriz a 0.

Los siguientes Kernels reciben un booleano por referencia y las matrices que necesiten. Comprueban el estado de la matriz, sobreescribiendo ese booleano en función de la comprobación que hagan.

- Kernel comprobar iguales: recibiendo dos matrices y un booleano, lo pone a false si encuentra un elemento en la misma posición en ambas matrices que sean distintos entre sí.
- Kernel comprobar llena: devuelve true sobre un booleano si una matriz está llena, es decir, si todos los elementos son distintos de 0.
- Kernel comprobar movimientos posibles: comprueba que aún se puede realizar alguna suma. Solo es llamado si la matriz está llena.
- Kernel comprobar si ha ganado: devuelve true sobre un booleano si alguna seed contiene el valor de OBJETIVO.

KERNELS PRINCIPALES

Se tienen dos kernels principales: el que realiza las posibles sumas en una dirección, y el que realiza los desplazamientos en una dirección.

KERNEL DE SUMAS

Recibe:

- *Matriz de entrada en DEVICE
- *Matriz de salida en DEVICE
- *Puntuación en DEVICE
- *Desplazamiento horizontal y vertical.
- *Dimensiones de matriz (filas y columnas).

Se basa en el siguiente algoritmo:

1. Identificar la fila, columna, y elemento de su posición.
2. Analizar el movimiento que se está realizando.
3. Para ese movimiento:
4. Si soy un 0, no hago nada.
5. Si estoy en el límite de la matriz en función del movimiento, me escribo a mi mismo en la misma posición en la matriz de salida.
6. Si no, compruebo la paridad de elementos en la dirección en la que me muevo:
    a. Si hay un número par de elementos iguales que yo incluyéndome a mi mismo marco como que multiplicaré.
    b. Espero a que marquen el resto de hilos.
    c. Si he marcado. multiplico el elemento inmediatamente cercano * 2 y me escribió en la matriz de salida a 0.
    d. Si hay un número impar de elemento iguales incluyéndome a mi mismo o un elemento distinto que yo, me escribiré en la matriz de salida con el mismo valor.

KERNEL DE DESPLAZAMIENTOS

Recibe:

- *Matriz de entrada en DEVICE
- *Matriz de salida en DEVICE
- *Puntuación en DEVICE
- *Desplazamiento horizontal y vertical.
- *Dimensiones de matriz (filas y columnas).

Se basa en el siguiente algoritmo:

1. Identifica fila, columna, y elemento de mi posición.
2. Analiza movimiento.
3. Si no estoy en el límite ni soy un 0, compruebo la casilla inmediatamente cerca a la mía en el movimiento en el que estoy.
4. Si la casilla vecina es un 0, significa que puedo desplazar.
5. Espero a que todos los kernel marquen.
6. Si he marcado que desplazaré, escribo en la matriz de salida mi valor en la posición + 1 en el movimiento elegido.
7. Si no me desplazo, escribo mi valor en la misma posición en la matriz de salida.
8. Espero a que terminen el resto de hilos.

JUEGO AUTOMÁTICO Y MANUAL

El juego soporta 2 tipos de jugadores:

JUEGO MANUAL

Este método es el encargado de las llamadas a kernels y funciones auxiliares para el funcionamiento del juego.

Sigue la siguiente sucesión de pasos:

1. Muestra mensaje de bienvenida y características de la tarjeta gráfica.
2. Establece movimientos y puntuación a 0.
3. Se pregunta si se quiere cargar partida.
    a. En caso afirmativo, se lee la matriz guardada, a menos que no exista, en cuyo caso muestra error.
    b. En caso negativo, se rellena la matriz con seeds de inicio, en base a la dificultad establecida.
4. Se explican instrucciones de juego.
5. Mientras no se gane o se pierda, o se salga:
    a. Se lee la entrada del usuario. Si es algo incorrecto, se muestra por pantalla.
    b. Si es correcto, se procesa el movimiento llamando a desplazarMatriz.
    c. Después, se actualiza el estado del juego (si ha ganado o perdido).
    d. Tras esto, se copia matriz de salida en entrada, para la siguiente iteración.
    e. Se imprime la matriz por pantalla, junto con la puntuación y número de movimientos.
    f. Se estudia el estado de la matriz:
        i. Si se gana, se muestra mensaje de victoria, y se pregunta si se quiere seguir jugando.
        ii. Si se pierde, se resta una vida y, si aún quedan vidas, se pregunta si se quiere volver a jugar. Si no quedan, fin de juego.
        iii. Si se acaban los movimientos tras haber ganado, no se resta vida, y se pregunta si se quiere volver a jugar.
6. Si se introduce una s, asociada a salir, se pregunta antes de salir si se quiere guardar la partida. De responder si, se escribe la matriz en un archivo de texto con el nombre de usuario introducido.

JUEGO AUTOMÁTICO

Esencialmente es idéntico al juego manual, pero sin la posibilidad de guardar y cargar partidas. Los movimientos son aleatorios para cada iteración, eligiéndolos de un array de posibles movimientos.

Si se pierde, la máquina también pierde vidas. Si se queda sin vidas o se elige salir tras perder, se termina el programa.