Trabajo final: Programación en GPU

Pablo Esau Mejía Medina, `Sistemas Concurrentes y Distribuidos`, Pablo Mejía[1,*]

[a] *Universidad Tecnológica Centroamericana*

Bienvenidos a la asignatura final de la asignatura sistema concurrentes y distribuidos. Esta consta de 3 ejercicios, la primera consiste en programar el conjunto de Maldebrot, el segundo la suma de matrices y el tercero el desencriptado de un mensaje.

Objetivo: Al completar esta asignatura usted sera capaz de:

1. Comprender la programación en GPU
2. Comprender y demostrar como alojar y mover memoria de y hacia la GPU
3. Comprender los bloques de hilos, los layout 1D y 2D
4. Comprender la Sincronizacion de hilos

Instrucciones: En orden de mantener un estándar en la presentación del trabajo, seguiremos las siguientes instrucciones:

1. Deberá presentar un informe BREVE de la expliacion de la solución a cada programa hecho en LATEX. Teniendo portada de los nombres de las parejas.
2. Los programas realizados, deberán ir comentados de la mejor manera posible, siendo explicito en su idea.
3. La forma de entrega sera a través de la plataforma, poniendo en una sola carpeta que tendrá como nombre su nombre y en ella subcarpetas con cada programa realizado.La fecha de entrega es para el día

Recomendaciones:

1. Para trabajar en Latex, online, recomiendo Overleaf o sharelatex.
2. El lenguaje de programación que elija deberá tener configurado la programación en pthreads, recomiendo C, c++, java o python.

[*] Correspondencia
Email address: `pablo.mejia@unitec.edu` (Pablo Mejía)

1. Practico:

1. En clase discutimos que era el Conjunto de Julia (para leer un poco mas dar click, note los demás fractales). Para este ejercicio ud deberá programar el conjunto (Fractal)de Mandelbrot (mas detalle, aqui). Para ello debera modificar el programa realizado para la clase del conjunto de Julia. Valor(10 %)
2. Para el problema 2, se requiere hacer la suma de dos Matrices de dimensiones grandes. Elabore una estrategia y programa en GPU para la solución de dicho problema. Valor(10 %)
3. El ejercicio 1 requiere que descifremos algún texto codificado. El texto provisto (en el archivo encrypted01.bin) ha sido codificado utilizando un cifrado afín. El cifrado afín es un tipo de cifrado de sustitución monoalfabético en el que cada carácter numérico del alfabeto se cifra con una función matemática. La función de cifrado se define como:

$$E(x) = (Ax + B) \mod M$$

Donde A y B son claves del cifrado, mod es la operación del módulo y A y M son coprimos. Para este ejercicio, el valor de A es 15, B es 27 y M es 128 (el tamaño del alfabeto ASCII). La función de descifrado afín se define como

$$D(x) = A^{-1}(x - B) \mod M$$

Donde A^{-1} es el inverso multiplicativo modular de A modulo M. Para este ejercicio, A^{-1} tiene un valor de 111.
Nota: La operación modular no es lo mismo que el operador residuo (%) para números negativos. Se ha proporcionado una función mod adecuada para el ejemplo. La función proporcionada toma la forma de $modulo(inta, intb)$ donde a en este caso queda todo lo que queda del operador de mód de funciones de descifrado afín (por ejemplo, $A^{-1}(x - B)$) y b es todo a la derecha del operador de mod (por ejemplo, M).

Como cada uno de los valores de caracteres encriptados es independiente, podemos usar la GPU para descifrarlos en paralelo. Para hacer esto, lanzaremos un hilo para cada uno de los valores de caracteres encriptados y usaremos una función del kernel para realizar el descifrado. A partir del código proporcionado, complete el ejercicio completando lo siguiente (Valor 13 %):

a) Modifique la función de modulo para que el kernel affine_decrypt pueda llamarla al dispositivo.
b) Implementar el kernel de desencriptación para un solo bloque de hilos con una dimensión x de N (1024). La función debe almacenar el resultado en d_output. Puede definir el módulo inverso A, B y M usando una definición de preprocesador.
c) Asigne un poco de memoria en el dispositivo para la entrada (d_input) y la salida (d_output).
d) Copie los valores de entrada del host en h_input en la memoria del dispositivo d_input
e) Configure un solo bloque de N hilos y ejecute el kernel affine_decrypt.
f) Copie los valores de salida del dispositivo en d_output a la memoria del host h_output.
g) Compila y ejecuta tu programa. Si ha realizado el ejercicio correctamente, debería descifrar el texto.
h) Modifique su código para completar el kernel affine_decrypt_multiblock trabaje de tal manera que se usen múltiples bloques de hilos. Cambie su grid y la dimensión del bloque de tal manera que se lancen 8 bloques y 128 hilos por bloque.