PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA.
SEGUNDO SEMESTRE 2020

TAREA 4
IMT2112-ALGORITMOS PARALELOS EN COMPUTACIÓN CIENTÍFICA

Vicente Hojas

1. Explicación del código:

   a) Los puntos c se eligen de forma aleatoria dentro del cubo de lado 4 y centro en el origen, esto es, si $c = a + ib$ entonces $-2 \leq a \leq 2$ y $-2 \leq b \leq 2$. De esta forma, como se tiene que este cuadrado contiene a la circunferencia de radio 2, esto es, $\{c \in \mathbb{C} : |c| \leq 2\}$, entonces para los c elegidos se tiene que $|c| \leq 2$. Esto es útil porque sabemos que el radio de escape del set de Mandelbrot es 2, o sea si $|x_n| > 2$ para algún n, entonces la serie diverge. Esto implica también que si $|c| > 2$ entonces la serie diverge porque $x_1 = c$, luego $|x_1| > 2$ y la serie diverge. Entonces sabemos que no hay puntos fuera de nuestro cuadrado que pertenezcan al conjunto de Mandelbrot. Luego para calcular el área se calcula la proporción de los puntos que pertenecen al conjunto y se multiplica esa proporción por 16.

   b) Se realizan *n_iter* iteraciones de la serie $x_n = x_{n-1}^2 + c$. Si luego de *n_iter* iteraciones se tiene que $|x_{n\_iter}| < 2$, entonces se considera que c pertenece al conjunto de Mandelbrot. La demostración de por qué el radio de escape es 2 se encuentra en el anexo (1)

   c) En este caso se utiliza un máximo de 50.000 iteraciones para comprobar la convergencia.

2. Creación de números aleatorios: Se crean dos vectores de números aleatorios, a y b, que corresponden a la parte real e imaginaria de los números aleatorios creados. Para la paralelización, el problema es que como son números *pseudo* aleatorios, para una misma semilla se genera el mismo número, entonces si se ocupa como semilla el tiempo de reloj, por ejemplo, entonces al correr el código en paralelo habrá muchos números que serán iguales.

3. Algoritmo en OpenCL: Se ocupa en parte el código sum-2 disponible en Canvas, la idea es que se les envían los vectores a y b a la GPU, donde luego cada thread se encarga de un chunck de los vectores y de revisar si el número pertenece o no al conjunto, luego se calcula la proporción de números del chunck que le correspondía al vector que cayeron dentro del conjunto (es en esta parte donde en cada thread se hace un if-else statement por cada número c, pero luego de correr todas las iteraciones), luego el thread 0 suma las proporciones locales y calcula a partir de ellas una proporción de ese grupo de trabajo. Sería ineficiente hacer el if-else statement en cada iteración de la serie porque se produciría divergencia de trabajo (instrucciones serían distintas dependiendo de si la serie converge o no, por lo que el paradigma SIMD se volvería mucho más lento y no sería eficiente).

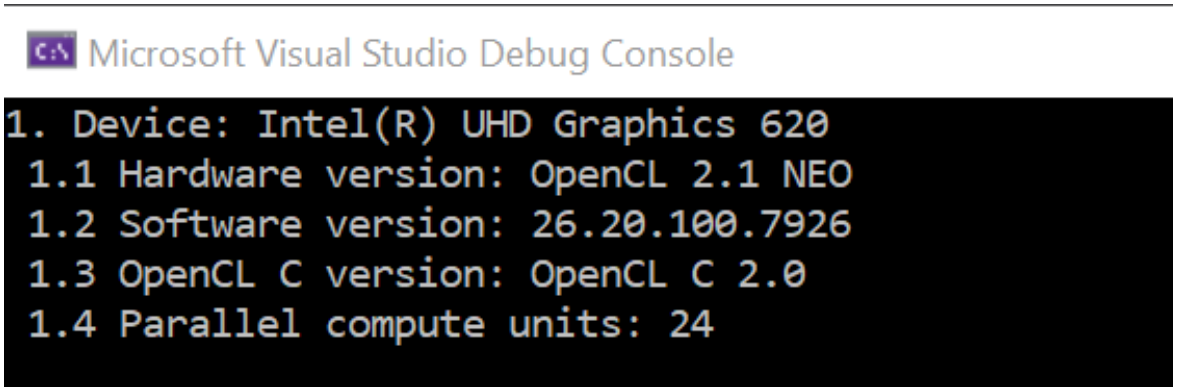

Figura 1: Resultado de opencl-devices.cpp

4. Resultados: En la figura (1) se muestra el resultado de correr opencl-devices.cpp (disponible en Canvas) en el computador en el cual se obtuvieron los Resultados

   En la figura (2) se muestra el resultado para un vector aleatorio de largo 100.000:

```
Size of sum: 100000
Size of work group: 8
Size of chunks: 128
Number of groups: 98
Size of vector: 100352
Device name: Intel(R) UHD Graphics 620
Global size of kernel: 100352
Local size of kernel: 8
Local sums: 0.094727  0.098633  0.096680  0.091797  0.112305  0.101563  0.080078  0.118164  0.095703  0.097656  0.086914
  0.103516  0.097656  0.088867  0.090820  0.103516  0.104492  0.093750  0.094727  0.087891  0.103516  0.103516  0.100586
  0.092773  0.095703  0.080078  0.090820  0.081055  0.102539  0.082031  0.092773  0.084961  0.119141  0.086914  0.077148
  0.082031  0.097656  0.086914  0.098633  0.094727  0.084961  0.094727  0.093750  0.108398  0.105469  0.091797  0.089844
  0.096680  0.083984  0.085938  0.085938  0.106445  0.109375  0.100586  0.076172  0.091797  0.102539  0.091797  0.098633
  0.086914  0.086914  0.088867  0.085938  0.084961  0.088867  0.097656  0.093750  0.094727  0.095703  0.100586  0.107422
  0.091797  0.096680  0.103516  0.098633  0.083008  0.095703  0.100586  0.087891  0.091797  0.094727  0.091797  0.090820
  0.084961  0.097656  0.085938  0.098633  0.067383  0.111328  0.103516  0.080078  0.080078  0.120117  0.091797  0.092773
  0.102539  0.095703  0.415039
Calculated sum: 1.509760
```

Figura 2: Resultado

Cada suma en cada grupo corresponde a la proporción de números que le correspondían al grupo que cayeron dentro del set, mientras que la suma total corresponde a la aproximación del área calculada (1.509760), vemos que el error es de 0.00316823.

Anexo

1. Si $\exists n \in \mathbb{N}$ tal que $|x_n| > 2$, entonces la serie definida por:

$$x_0 = 0$$
$$x_n = x_{n-1}^2 + c \quad n = 1, 2, ...$$

   diverge. Primero, para $|c| \leq 2$, suponemos que $|x_n| > 2$ para algún n. Se tiene que:

$$\begin{aligned} |x_{n+1}| &= |x_n^2 + c| \\ &\geq |x_n|^2 - |c| \\ &\geq 2|x_n| - |c| \\ |x_{n+1}| - |x_n| &\geq |x_n| - |c| \\ &\geq |x_n| - 2 \\ &> 0 \end{aligned}$$

Luego vemos que $|x_{n+1}| > |x_n|$, luego $|x_n| \to \infty$. Para $c > 2$, veremos por inducción que para todo $n \in \mathbb{N}$, $n \geq 2 \to |x_n| > |c|$. Para el caso base:

$$\begin{aligned} |x_2| &= |c^2 + c| \\ &\geq |c|^2 - |c| \\ &= |c|(|c| - 1) \\ &> |c| && (|c| > 2) \end{aligned}$$

Asumimos que $|x_n| > 2$, veremos que para x_{n+1} se tiene que:

$$\begin{aligned} |x_{n+1}| &\geq |x_n|^2 - |c| \\ &\geq |c||x_n| - |c| && \text{(por HI)} \\ &= |x_n| + |x_n|(|c| - 1) - |c| \\ |x_{n+1}| - |x_n| &\geq |x_n|(|c| - 1) - |c| \\ &\geq |c|(|c| - 1) - |c| \\ &= |c|(|c| - 2) \\ &> 0 && (|c| > 2) \end{aligned}$$

Luego $|x_{n+1}| - |x_n| > 0$, luego la hipótesis de inducción se cumple y además se ve que $|x_{n+1}| - |x_n| > 0$, por lo que la serie es monótona creciente (en valor absoluto) y por lo tanto diverge.