Computación Paralela - Práctica 3

Cálculo del número Pi usando el método de Montecarlo y OpenCL

Esta práctica consiste en implementar el mismo método de la práctica 2 pero utilizando OpenCL. Para ello, se implementará un kernel que se encargará de generar N/Num_Work_Items puntos aleatorios e ir sumando cuántos entran dentro de la circunferencia. A diferencia de OpenMP o MPI, OpenCL no dispone de una operación de reducción entre work items de diferentes work groups (sí entre work items del mismo work group, pero aún no la hemos estudiado aún), por lo que el kernel tendrá un vector de salida de Num_Work_Item elementos en el que cada work item almacenará su suma parcial en la posición correspondiente a su identificador. Será el Host el encargado de realizar la suma final a partir de todas las sumas parciales. Por tanto, el kernel tendrá dos parámetros:

- El número de puntos totales que se deben generar (N). Parámetro de entrada de tipo *int*.
- Vector de tamaño Num_Work_Items donde se almacenarán las sumas parciales de cada work item. Parámetro de salida de tipo *int **.

Para los números aleatorios usaremos una versión reducida de la librería mramd (mramd.cl, disponible en el campus virtual). Para utilizarla basta con añadir *#include "mramd.cl"* al código OpenCL. Esta librería incluye únicamente dos funciones:

- *int ramd(int *status)* → Devuelve un número aleatorio entre 0 y MRAND_MAX. Para generar un número entre 0 y 1 basta con dividir el resultado entre MRAND_MAX. El parámetro *status* sirve para almacenar el estado de la secuencia de forma independiente entre los work items. De esta forma, cada work item podrá generar una secuencia de números aleatorios diferentes.
- *void sramd(int *status, int semilla)* → Inicializa la secuencia de números aleatorios con el parámetro *semilla*. Para que cada work item tenga una secuencia distinta, usaremos su identificador global para generar la semilla (por ejemplo, una constante más el identificador, el propio identificador, …).

Respecto al cálculo de la raíz cuadrada, OpenCL dispone de la función nativa *sqrt* (idéntica a su homónima de C++) que puede usarse en cualquier kernel OpenCL sin necesidad de incluir nada.

El programa recibirá dos parámetros de entrada: el primero indicará el número de iteraciones y el segundo el número de work items que se usarán.

Deberán realizarse pruebas con 4, 8 y 16 work items y comparar éstas con los valores obtenidos por OpenMP en la práctica 2 con 4, 8 y 16 hilos (cada alumno puede acceder a su excel a través del campus virtual). El tamaño de N será el mismo que se usó en la práctica 2. Deberá rellenarse el fichero p3.xlsx (disponible en el Campus Virtual) con los valores de las

pruebas y de los datos correspondientes de la práctica 2 (para facilitar la comparación) así como con las conclusiones obtenidas por el alumno al comparar con OpenMP.

Deberán entregarse los ficheros de código (host y kernel) junto con el fichero de conclusiones en un único archivo comprimido cuyo nombre serán las iniciales del alumno seguido de un guión bajo y P3 (por ejemplo, en mi caso sería JMGC_P3). Dicho archivo se subirá al campus virtual a través de la tarea abierta a tal efecto.