

Instrucciones para la realización del proyecto GPU

La tarea final es realizar un histograma de un vector V de un número elevado N de elementos enteros aleatorios. El histograma consiste en un vector H que tiene M elementos que representan "cajas". En cada caja se cuenta el número de veces que ha aparecido un elemento del vector V con el valor adecuado para asignarlo a esa caja (normalmente cada caja representa un rango o intervalo de valores). En nuestro caso, para simplificar la asignación del elemento de V a su caja correspondiente del histograma, vamos a realizar la operación $\text{ValorElementoV} \bmod M$, que directamente nos da el índice de la caja del histograma al que pertenecerá ese elemento y que deberemos incrementar. Se sugiere como N un valor del orden de millones de elementos y como M , 8 cajas.

Como implementación base (que podremos mejorar en tiempo o no) se pide crear tantos hilos como elementos de V para que cada uno se encargue de ir al elemento que le corresponda en V e incremente la caja que corresponda en el vector histograma H (posiblemente de forma atómica).

Como segunda implementación, dividiremos la operación en dos fases. En la primera, en lugar de trabajar sobre un único histograma global, repartiremos el cálculo realizando un cierto número de histogramas (que llamaremos "locales"), cada uno calculado sobre una parte del vector de datos. La idea es reducir el número de hilos escribiendo sobre la misma posición del histograma, ya que dicha operación debe ser atómica y se serializan dichos accesos. La segunda fase realizará la suma de los histogramas locales en un único histograma global final. Se debe intentar llevar a cabo de la forma más paralela posible, utilizando el método de reducción.

Es importante medir adecuadamente el tiempo que conlleva el trabajo realizado por el/los kernels (pueden ser varios). La inicialización del histograma también debe ser incluido en el tiempo medido (en un kernel separado o no, en función del algoritmo). Es importante promediar el tiempo obtenido en varias repeticiones (descartando la primera, que suele ser mayor). Reflejar en las cifras de los tiempos finales solamente las cifras decimales que creamos que son estables y que nos permitan dar conclusiones sobre las pruebas que hagamos y las diferencias que observemos.

Es relevante también discutir el tipo de memoria que podemos utilizar en cada momento y posiblemente realizar pruebas con diferentes posibilidades, comprobando si se obtienen ventajas o no en los tiempos.

El objetivo final es discutir las diferencias en las implementaciones o pruebas que hagamos, intentando explicar (en la medida de lo posible) el porqué de su rendimiento, y qué restricciones imponemos sobre su funcionamiento (número de elementos que usamos, número de histogramas locales, etc.). Es posible (no obligatorio) hacer pruebas "de barrido" seleccionando variar un parámetro del algoritmo y graficar los tiempos obtenidos.

Se valorará la generalidad del algoritmo y su correcto funcionamiento, las pruebas realizadas y su relevancia en el razonamiento. Todo ello se reflejará en un informe y se entregarán además los ficheros con el código que se considere relevante (resultado final, pruebas, implementaciones diferentes, etc.) en un fichero comprimido.

Requisitos

Como mínimo, se espera que se realicen dos implementaciones y que se midan tiempos en ambos casos. Si al realizar la segunda implementación no se llega a realizar una implementación

de la reducción o se hace alguna variación del método, siempre que se discutan los límites y características de lo realizado, el proyecto será válido.

Opcionalmente, se valorará también el haber explorado partes del API de CUDA que por su novedad o complejidad no se han visto en clase como el "Unified Memory Programming" (fácil, simplifica la programación), "Cuda Dynamic Parallelism" (más difícil, permite invocar kernels desde otros kernels) o los "Cooperative Groups" (más difícil, permite sincronizar nº de hilos inferior o superior al nº de hilos del bloque) si con su aplicación se consigue simplificar o mejorar en algún sentido el código del problema.