Data Parallel Computing

Norman Patrick Harvey Arce

June 12, 2017

Ejercicios Capítulo 2
Libro: Programming Massively Parallel Processors 3rd Edition

1 If we want to use each thread to calculate one output element of a vector addition, what would be the expression for mapping the thread/block indices to data index?

C. i=blockIdx.x*blockDim.x + threadIdx.x;

Necesitamos saber la dimensión del bloque

2 Assume that we want to use each thread to calculate two (adjacent) elements of a vector addition. What would be the expression for mapping the thread/block indices to i, the data index of the first element to be processed by a thread?

C. i=(blockIdx.x*blockDim.x + threadIdx.x)*2

Cada thread cubre 2 elementos consecutivos. El índice de datos inicial es simplemente el doble del índice de subprocesos globales, es decir, que todos los bloques anteriores cubren (blockIdx.x * blockDim.x) * 2. Dentro del bloque, cada thread cubre 2 elementos consecutivos de modo que la posición inicial de un thread es 2 * threadIdx.x

3 We want to use each thread to calculate two elements of a vector addition. Each thread block processes 2*blockDim.x consecutive elements that form two sections. All threads in each block will first process a section first, each processing one element. They will then all move to the next section, each processing one element. Assume that variable i should be the index for the first element to be processed by a thread. What would be the expression for mapping the thread/block indices to data index of the first element?

D. i=blockIdx.x*blockDim.x*2 + threadIdx.x;

Cada bloque anterior cubre (blockIdx.x * blockDim.x) * 2. El elemento inicial es consecutivo en este caso así que simplemente se añade threadIdx.x

4 For a vector addition, assume that the vector length is 8000, each thread calculates one output element, and the thread block size is 1024 threads. The programmer configures the kernel launch to have a minimal number of thread blocks to cover all output elements. How many threads will be in the grid?

C. 8192

Techo(8000 / 1024.0) * 1024 = 8 * 1024 = 8192. Múltiplo mínimo de 1024 para cubrir 8000 es 1024 * 8 = 8192.

5 If we want to allocate an array of v integer elements in CUDA device global memory, what would be an appropriate expression for the second argument of the cudaMalloc call?

D. v * sizeof(int)

Simplemente multiplicamos el número de elementos (v) por el tamaño que necesita un int para ser almacenado.

6 If we want to allocate an array of n floating-point elements and have a floating-point pointer variable d_A to point to the allocated memory, what would be an appropriate expression for the first argument of the cudaMalloc() call?

D. (void **) &d_A

&d_A es un puntero a un puntero float. Para convertirlo en un puntero para cudaMalloc() se debería usar (void **) para "castearlo" como un puntero doble.

7 If we want to copy 3000 bytes of data from host array h_A (h_A is a pointer to element 0 of the source array) to device array d_A (d_A is a pointer to element 0 of the destination array), what would be an appropriate API call for this data copy in CUDA?

C. cudaMemcpy(d_A, h_A, 3000, cudaMemcpyHostToDevice);

La sintaxis de la función cudaMemcpy indica que los parámetros que debemos indicar, son en primer lugar, array origen, array destino, tamaño, host a dispotivo, en ese orden.

8 How would one declare a variable err that can appropriately receive returned value of a CUDA API call?

C. cudaError_t err;

Sintaxis (typdef enum).

9 A new summer intern was frustrated with CUDA. He has been complaining that CUDA is very tedious: he had to declare many functions that he plans to execute on both the host and the device twice, once as a host function and once as a device function. What is your response?

Hay funciones que probablemente sólo se deban ejecutar en el Host ya que no es necesario declararlas en el Dispositivo.