Ejercicios del Libro Programming Massively Parallel Processors

Capítulo 2

Alumno: Patrick Xavier Márquez Choque

2.8 EJERCICIOS

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

En primer lugar, la solución para una suma de vectores implica la creación de bloques dentro de ambos vectores para dar una organización en cuanto a cada thread realizará una suma de los mismos dentro de un vector resultante C.

Entonces blockIdx.X se refiere a un tipo de dato que cada thread especificará que están utilizando vectores de una sola dimensión; esto al ser multiplicado por blockDim.x que nos permite empezar desde un bloque en común para todos las threads(ya que esta esta estructura especifica cuantas threads existen para cada bloque y nos permite delegar el trabajo para cada thread dentro de cada bloque) y que luego al ser sumado por ThreadIdx.x nos permite encontrar el bloque único para cada thread que tendrá que sumar.

Entonces la definición de este índice i nos permite ubicar específicamente, en que dimensión estamos trabajando(ya sea en vector de 1 dimensión, matrices de 2 dimensiones o de 3 dimensiones) y en que bloque vamos a estar para cada thread ya que cada thread va a trabajar en un bloque único pero esto nos permite desplazarnos entre los bloque y así no ocurran errores ya que cada thread van a tener valores del índice i únicos para acceder a los valores de A y B, sumarlos y colocar la solución dentro del vector resultante C.

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

Como ya se vio en la pregunta anterior donde se utiliza el índice i como único indicador dentro de cada thread en un mismo bloque para saber las posiciones dentro de nuestro vector resultante pero la diferencia es que se le asignará el trabajo de sumar 2 elementos continuos para cada thread, esto reduciendo la cantidad de threads que se lanzarán dentro de cada bloque pero la solución es la multiplicación de este índice por 2 ya que esto nos permitirá saber las posiciones si estuviéramos recorriendo estos elementos de dos en dos. El detalle es que es necesario según está solución que dentro de la propia función de vecAddKernel que se lanzará con cada thread un statement más donde ambos valores entonces dentro del condicional i<n se le tendrá que añadir C[i+1] = A[i+1] + B[i+1] así cada thread tendrá un mayor esfuerzo pero esto permitirá que se reduzcan a la mitad la cantidad de threads necesarias para cada bloque.

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

El detalle que diferencia a este problema es que dentro de cad Thread Block se va a tener 2 secciones entre los elementos de los vectores, entonces el detalle es que cada thread va a calcular la suma de los vectores dentro de un elemento tanto en la primera como en la segunda sección. Este detalle permite sumar "consecutivamente" los elementos entre estas dos secciones. Esta modificación lo podemos ubicarlo multiplicando los valores del BlockDim * 2 ya que, como cada thread sumará dos valores entre ambas secciones, empezando para el primer valor e ir al otro elemento de la otra sección y calcularla dentro del vector resultante.

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

Se divide el número de elementos del tamaño del vector entre la cantidad de threads de cada thread block, el resultado es aproximadamente 7.8 pero se tiene que redondear hacia arriba ya que la cantidad de bloques no puede ser decimal, entonces nos quedarán 8 bloques con un total de 8192 threads

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)

La función de cudaMalloc pide como segundo argumento el tamaño total de memoria en bytes que se está reservando, entonces una manera de realizarlo es multiplicar la cantidad de elementos por cuanto pesa cada elemento, ya que cada elemento es un entero se recurre a la función "sizeof()"

6. 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. (void **) $d_A

La función de cudaMalloc pide como primer argumento un doble puntero a la memoria que se está reservando y entonces como nosotros disponemos de un puntero entonces lo que tenemos que hacer es darle la dirección de memoria de este puntero con el operador & y castearlo para evitar problemas para que sea un puntero al puntero de la memoria.

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)

Esta función de cudaMemcpy tiene estos argumentos: En primer lugar el primer argumento es un puntero al array destino, luego es un puntero al array de fuente de los elementos, luego es la cantidad de elementos que son 3000 bytes y posteriormente es la dirección hacia donde se están copiando los valores, en este caso se trata de copiar los datos desde un Host hacia un device destino por lo tanto utilizaremos la primitiva cudaMemcpyHostToDevice aunque también se puede utilizar el número 1 ya que está sobrecargado en la función de CUDA.

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

Según la propia especificación de CUDA, en la propia documentación se menciona los tipos de error de CUDA y la forma apropiada manera de inicializarlo es cudaError_t err; ya que el tipo de dato se especifica como: typedef enumcudaError cudaError_t.

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?

En primer lugar lo que podría responder es que pueden buscar opciones para estas "dificultades", ya que en posteriores versiones a CUDA 3.0 se puede utilizar una declaración una única función para ambos, tanto para el host como para el device a través de la utilización del comando "__host__ __device__ void function(void *argument)" consecutivamente, esto permite la innecesaria inicialización de dos función es para cada ámbito del usuario. Por lo tanto siempre es bueno revisar las versiones disponibles de CUDA mediante la documentación para saber las distintas manera de hacer los llamados a estas funciones