Exámenes

CUDAtest

Volver a la Lista de Exámenes

Parte 1 de 1 - / 1.0 Puntos

Preguntas 1 de 10

0.1 Puntos. Puntos descontados por fallo: 0.025

El runtime de CUDA sirve, entre otras cosas, para

- ✔ ○ A. Que un programa pueda gestionar varias GPUs y lanzar kernels diferentes a cada una de ellas
- ✔ ○ B. Transformar instrucciones de OpenCL a CUDA
- ✔ ○ C. Asignar las iteraciones del bucle a los hilos disponibles
- ✔ ○ D. Generar el código máquina a partir del código PTX

Respuesta correcta: A

Preguntas 2 de 10

0.1 Puntos. Puntos descontados por fallo: 0.025

Indica la afirmación FALSA en relación a la arquitectura de una GPU

- ✔ ○ A. La GPU consta de N multiprocesadores, cada uno de ellos con M núcleos, compartiéndose la unidad de instrucción entre los M núcleos
- ✔ ○ B. La memoria global es más lenta que el resto de memorias, y está situada fuera del chip de la GPU
- ✔ ○ C. Es posible implementar kernels muy eficientes aprovechando la memoria compartida, que es rapidísima, la cual permite que dos hilos cualesquiera puedan comunicarse mediante variables en esta memoria, aunque pertenezcan a diferente bloque de hilos
- ✖ ○ D. En comparación con la memoria global, las memorias constante y de texturas son más rápidas, pero únicamente puede escribir en ellas el *host*

Respuesta correcta: C

Preguntas 3 de 10

0.1 Puntos. Puntos descontados por fallo: 0.025

Indica la opción FALSA en relación a la definición de un kernel de CUDA

- ✔ ○ A. Es una función que se ejecuta en la GPU de forma replicada por tantos hilos como se haya indicado en la invocación
- ✖ ○ B. Es una función declarada con los modificadores `__global__` o `__device__`
- ✔ ○ C. Es una función que se ejecuta en la GPU pero se invoca habitualmente desde el *host*
- ✔ ○ D. Es una función para facilitar el cálculo de los índices globales de cada hilo a partir de `threadIdx` y `blockIdx`

Respuesta correcta: D

Preguntas 4 de 10

0.1 Puntos. Puntos descontados por fallo: 0.025

Si consideramos las diferencias entre los hilos de CUDA y los de OpenMP, ¿cuál de las siguientes opciones NO es correcta?

- ✔ ○ A. En CUDA los hilos se ejecutan en grupos de 32 síncronamente, todos la misma instrucción, mientras que en OpenMP cada hilo tiene su propio contexto de ejecución y, por tanto, se ejecutan de forma independiente
- ✖ ○ B. En OpenMP los hilos se comunican a través de variables en memoria compartida, y en CUDA también, con la salvedad de que si las variables están en la memoria especial `__shared__` solo se pueden comunicar los hilos del mismo bloque
- ✔ ○ C. En CUDA el identificador de hilo se usa para que cada hilo realice un cálculo distinto, mientras que en OpenMP no
- ✔ ○ D. La eficiencia en CUDA se basa en realizar una descomposición de grano fino para poder lanzar miles de hilos a la vez, mientras que en OpenMP el número de hilos no suele ser tan alto y, por tanto, debe hacerse una descomposición en tareas más gruesas

Respuesta correcta: C

Preguntas 5 de 10

0.1 Puntos. Puntos descontados por fallo: 0.025

¿Cómo se establece el número de hilos y su estructura jerárquica en CUDA?

- ✔ ○ A. El número de bloques de hilos e hilos dentro de cada bloque se va ajustando dinámicamente durante la ejecución según el número de núcleos libres que tenga la GPU

- ✔ ◯ B. Se usan dos argumentos especiales con la sintaxis `<<<bloq,hilos>>>` al invocar el kernel, que indican el número de bloques de hilos e hilos dentro de cada bloque, respectivamente
- ✔ ◯ C. El sistema elige el número de hilos apropiado a partir del tamaño de la matriz indicado al invocar el kernel
- ✔ ◯ D. Las variables `gridDim` y `blockDim` deben definirse al principio del código del kernel

Respuesta correcta: B

Preguntas 6 de 10

0.1 Puntos. Puntos descontados por fallo: 0.025

Supongamos que se invoca un kernel de la siguiente forma:

```
dim3 bsize(10,10);
dim3 gsize(8,8);
MatAdd <<<gsize, bsize>>> (A, B, C);
```

- ✔ ◯ A. Se lanzan un total de 36 hilos
- ✔ ◯ B. Se lanzan un total de 80 hilos
- ✔ ◯ C. Se lanzan un total de 6400 hilos
- ✔ ◯ D. Se lanzan un total de 800 hilos

Respuesta correcta: C

Preguntas 7 de 10

0.1 Puntos. Puntos descontados por fallo: 0.025

La llamada

```
cudaMemcpy(a,b,10*sizeof(double),cudaMemcpyDeviceToDevice)
```

- ✖ ◯ A. Realiza la copia de 10 bytes de b a a
- ✔ ◯ B. Será correcta siempre y cuando los punteros a y b sean direcciones de memoria de la GPU
- ✔ ◯ C. Realiza la copia de 80 bytes de a a b
- ✔ ◯ D. No se puede realizar desde código de *host*, únicamente desde código de un kernel

Respuesta correcta: B

Preguntas 8 de 10

0.1 Puntos. Puntos descontados por fallo: 0.025

La llamada `__syncthreads()` ...

- ✔ ◯ A. Se pone opcionalmente después de invocar un kernel para que la ejecución en la CPU se espere a que el kernel haya terminado de ejecutarse
- ✔ ◯ B. Hace que el contenido de la memoria compartida se sincronice (se haga copia) con memoria global
- ✔ ◯ C. Fuerza la sincronización de los hilos, pero afecta únicamente a los hilos de un mismo bloque de hilos
- ✔ ◯ D. Hace que se ejecute un bloque de hilos a continuación de otro, aunque haya en la GPU más núcleos disponibles

Respuesta correcta: C

Preguntas 9 de 10

0.1 Puntos. Puntos descontados por fallo: 0.025

Supongamos que se lanza un kernel con bloques de hilos bidimensionales. El kernel opera con matrices, y queremos que cada hilo realice el trabajo asociado al elemento `i,j` de las matrices. ¿Cuál sería una forma correcta de calcular los índices `i,j`?

- ✔ ◯ A.
```
int i = threadIdx.x + gridDim.x*blockIdx.x;
int j = threadIdx.y + gridDim.y*blockIdx.y;
```
- ✔ ◯ B.
```
int i = threadIdx.x + blockDim.x*blockIdx.x;
int j = threadIdx.y + blockDim.y*blockIdx.y;
```
- ✔ ◯ C.
```
int i = threadIdx + blockDim*blockIdx;
int j = i%gridDim;
```
- ✔ ◯ D.
```
int i = blockIdx.x + blockDim.x*threadIdx.x;
int j = blockIdx.y + blockDim.y*threadIdx.y;
```

Respuesta correcta: B

Preguntas 10 de 10

0.1 Puntos. Puntos descontados por fallo: 0.025

Un hilo que está ejecutando un *kernel* tiene las siguientes variables predefinidas:

```
gridDim={2,5}, blockIdx={0,0}, blockDim={5,5}, threadIdx={3,1}
```

- ✔ ○ A. Se trata del hilo con coordenadas (0,0) dentro del bloque (3,1). El kernel se está ejecutando con un total de 250 hilos.
- ✔ ○ B. Se trata del hilo con coordenadas (0,0) dentro del bloque (3,1). El kernel se está ejecutando con un total de 35 hilos.
- ✔ ○ C. Se trata del hilo con coordenadas (3,1) dentro del bloque (0,0). El kernel se está ejecutando con un total de 35 hilos.
- ✔ ○ D. Se trata del hilo con coordenadas (3,1) dentro del bloque (0,0). El kernel se está ejecutando con un total de 250 hilos.

Respuesta correcta: D