Micro investigación

1. ¿Qué es CUDA?
CUDA es una arquitectura de cálculo paralelo de NVIDIA que aprovecha la gran potencia de la GPU (unidad de procesamiento gráfico) para proporcionar un incremento extraordinario del rendimiento del sistema [1].

2. ¿Qué es un kernel en CUDA y cómo se define?
Un kernel en C para CUDA, es una función la cual al ejecutarse, lo hará en n distintos hilos en lugar de en secuencial. Se define incluyendo __global__ en la declaración, de la siguiente forma:

```
__global__ void f(int a, int b, int c)
{
}
```

3. ¿Cómo se maneja el trabajo a procesar en CUDA? ¿Cómo se asignan los hilos y bloques?
Los hilos, en las GPU, se organizan en bloques. Los bloques son ejecutados por una unidad multiprocesamiento. Los hilos pueden tener índices de hasta 3 dimensiones al igual que los bloques. Existe una lista de bloques esperando ser procesados por el GPU.

4.Investigue sobre la plataforma Jetson TX2 ¿Cómo está compuesta la arquitectura de la plataforma a nivel de hardware?
Jetson TX" es un sistema embebido en un módulo. Posee un procesador NVIDIA Denver2 de doble núcleo y un procesador ARM Cortex-A57 de cuatro núcleos. Posee 8GB de memoria RAM y un GPU Pascal de 256 núcleos de procesamiento paralelo [2].

Ejemplo CUDA

1. Los pasos para generar una aplicación en CUDA son:
    a. Inicializar la memoria del GPU con los valores necesarios (no siempre se requiere).
    b. Configurar la cantidad de bloques y la cantidad de hilos por bloque.
    c. Llamar al kernel para que ejecute la función en el GPU.
    d. Sincronizar los hilos de ser necesario.

2. Este código suma cada uno de los elementos de los vectores. Cada elemento procesado se identifica con un número de identificación dentro de los bloques que se envían al CPU. Dentro del kernel, cada "iteración" se determina con estos identificadores.

3. Efectivamente suma cada elemento de los vectores entrantes tanto en GPU como en CPU. Para finalizar compara los tiempos de cada ejecución, como se aprecia en la siguiente imagen:

```
block size = 100
n = 1000        GPU time = 0.000156s    CPU time = 0.000013s
n = 10000       GPU time = 0.000120s    CPU time = 0.000177s
n = 100000      GPU time = 0.000151s    CPU time = 0.001569s
n = 1000000     GPU time = 0.000225s    CPU time = 0.015663s

block size = 250
n = 1000        GPU time = 0.000126s    CPU time = 0.000012s
n = 10000       GPU time = 0.000142s    CPU time = 0.000390s
n = 100000      GPU time = 0.000152s    CPU time = 0.002173s
n = 1000000     GPU time = 0.000290s    CPU time = 0.011250s
n = 10000000    GPU time = 0.000297s    CPU time = 0.089702s

block size = 400
n = 1000        GPU time = 0.000124s    CPU time = 0.000011s
n = 10000       GPU time = 0.000119s    CPU time = 0.000162s
n = 100000      GPU time = 0.000158s    CPU time = 0.001379s
n = 1000000     GPU time = 0.000183s    CPU time = 0.011636s
n = 10000000    GPU time = 0.000275s    CPU time = 0.089416s

block size = 550
n = 1000        GPU time = 0.000146s    CPU time = 0.000009s
n = 10000       GPU time = 0.000122s    CPU time = 0.000143s
n = 100000      GPU time = 0.000161s    CPU time = 0.001352s
n = 1000000     GPU time = 0.000200s    CPU time = 0.014925s
n = 10000000    GPU time = 0.000321s    CPU time = 0.092201s
```

Referencias

[1] NVIDIA Corporation. (2018). Accelerated Computing - Training. 12 de mayo de 2018, de NVIDIA Corporation Sitio web: https://developer.nvidia.com/accelerated-computing-training

[2] NVIDIA Corporation. (2018). Jetson TX2 Module. 12 de mayo de 2018, de NVIDIA Corporation Sitio web: https://developer.nvidia.com/embedded/buy/jetson-tx2