DANIEL MONJAS MIGUÉLEZ
70274432-W
18/05/2022

ACAP: PRÁCTICA 4

Actividad 1:

Ancho de banda de memoria

Para medir el ancho de banda de memoria usaremos tanto el ancho de banda pico como el efectivo. El primero de ellos se calcula únicamente teniendo en cuenta las especificaciones del hardware de la gráfica que estemos utilizando. Es más, por lo general en estas mismas especificaciones este viene ya calculado.

$$BW_{Theoretical} = \frac{MemClockRate(Hz) * MemBusWidth(Bytes)}{10^9}$$

Los parámetros utilizados son:

- *MemClockRate(Hz):* frecuencia de la memoria en hertzios.
- *MemBusWidth(Bytes):* ancho del bus de memoria en bytes.
- Aunque no se haya puesto, esto habrá que multiplicarlo por dos o no en función de si la memoria es DDR, es decir *double data rate* (suelen serlo).

Esta fórmula lo que hace es múltiplicar la frecuencia de memoria por los bytes que se transfieren por el bus de memoria por tick de reloj (y por dos en caso de ser *double data rate*, es decir, transfiere dos veces por tick de reloj, una a la subida y otra a la bajada) y pasar el resultado a giga bytes, es decir, divide entre 10^9, dando así el ancho de banda teórico en GB/s.

Para calcular el ancho de banda efectivo debemos tener en cuenta el programa específico que vayamos a utilizar. Para este cálculo usaremos la siguiente ecuación.

$$BW_{Effective} = \frac{(R_B + W_B)}{t(s) * 10^9}$$

Los parámetros utilizados son:

- R_B *:* número de bytes leídos por kernel.
- W_B *:* número de bytes escritos por kernel.
- *t:* tiempo transcurrido en segundos.

Por otro lado, esta fórmula toma los bytes que se transfieren entre lecturas y escrituras en el programa, y los divide entre el tiempo en segundos múltiplicado por 10^9, para así obtener el ancho de banda efectivo en GB/s.

Si nos fijamos en el código utilizado para las mediciones del ancho de banda de memoria efectivo, lo que se hace es medir el tiempo transcurrido durante la ejecución del kernel y ese será el que se va a tener en cuenta como la *t* en el cálculo del ancho de banda efectivo.

Capacidad de cómputo

Para el cálculo de la capacidad de cómputo, aunque no hay un parámetro que sirva como medida universal vamos a usar los *GFLOP/s,* que es una medida común de la capacidad de cómputo. Para esta medida debemos tener en cuenta el programa en concreto que vamos a utilizar como medición.

El programa utilizado tiene una operación de suma y una de multiplicación, dando lugar a dos operaciones flotantes. Con esto utilizamos la siguiente fórmula:

$$GFLOP/s_{Effective} = \frac{NumFLOPS}{t * 10^9}$$

Los parámetros utilizados son:

- *NumFLOPS:* número de operaciones en punto flotante en el programa (o que se tengan en cuenta dentro de la medida del tiempo).
- *t:* tiempo transcurrido en segundos.

En particular, en el programa usado para medir la capacidad de cómputo *NumFLOPS=2*N* donde *N* es el tamaño de los vectores *x* e *y*.

Con estás fórmulas enunciadas vamos a replicar la medición del ancho de banda de memoria y de la capacidad de cálculo tanto en GENMAGIC.UGR.ES como en mi ordenador.

GENMAGIC.UGR.ES

```
estudiante20@genmagic:~/PRACTICA_4$ ./deviceQuery
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA GeForce RTX 2080 Ti"
  CUDA Driver Version / Runtime Version          11.4 / 11.7
  CUDA Capability Major/Minor version number:    7.5
  Total amount of global memory:                 11018 MBytes (11553341440 bytes)
  (068) Multiprocessors, (064) CUDA Cores/MP:    4352 CUDA Cores
  GPU Max Clock rate:                            1545 MHz (1.54 GHz)
  Memory Clock rate:                             7000 Mhz
  Memory Bus Width:                              352-bit
```

Las características que nos interesan son:

- Frecuencia del reloj de memoria: 7000Mhz
- Ancho del bus de memoria: 352-bit
- Aunque ahí no aparece, vemos que la gráfica es una “NVIDIA GeForce RTX 2080 Ti”, y buscando sus especificaciones vemos que utiliza memoria GDDR6, luego usa *double data rate.*

Teniendo en cuenta lo anterior llegamos a que el ancho de banda de memoria pico es:

$$BW_{Theoretical} = 7000 * 10^6 * (352/8) * 2/10^9 = 616 GB/s$$

Ahora para ver cual es el ancho de banda de memoria efectivo ejecutamos el programa 5 veces en GENMAGIC y hacemos la media:

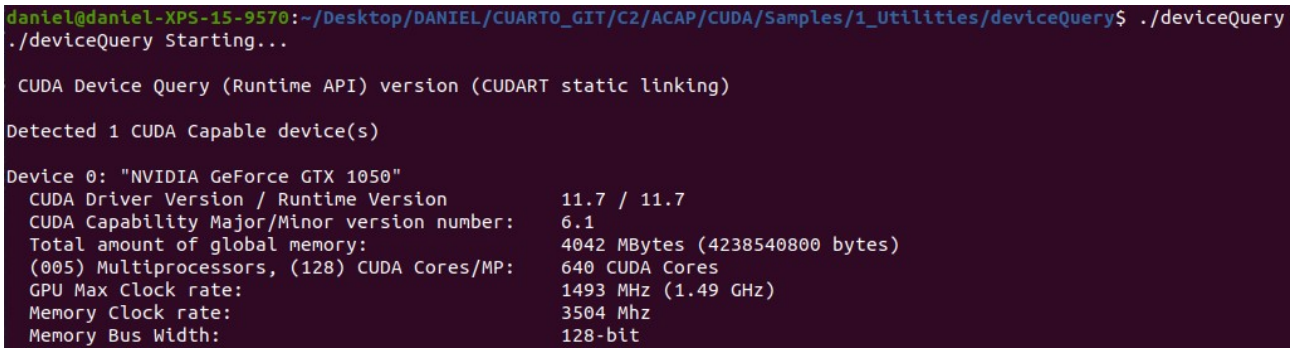
Mediciones	Media
550,954176	552,927552
553,357696	
557,95104	
550,260288	
552,11456	

, donde se ve que el ancho de banda efectivo medio es peor que el pico calculado anteriormente. Para la capacidad de cálculo, simplemente he introducido un nuevo *printf* en el programa que muestre el resultado de aplicar la fórmula vista más arriba a los parámetros *N* y *milliseconds*, teniendo en cuenta que el número de operaciones en punto flotante es *2*N*.

Mediciones	Media
92,87324	92,3624928
92,291232	
92,122576	
92,51924	
92,006176	

Con esto obtenemos una capacidad de cómputo media de 92,3624928 GFLOP/s para el programa de mediciones utilizado.

DELL XPS15



Las características que nos interesan son:

- Frecuencia del reloj de memoria: 3504Mhz
- Ancho del bus de memoria: 128-bit
- Aunque ahí no aparece, vemos que la gráfica es una “NVIDIA GeForce GTX 1050”, y buscando sus especificaciones vemos que utiliza memoria GDDR6, luego usa *double data rate.*

Teniendo en cuenta lo anterior llegamos a que el ancho de banda de memoria pico es:

$$BW_{Theoretical} = 3504 * 10^6 * (128/8) * 2/10^9 \simeq 112\, GB/s$$

Ahora para ver cual es el ancho de banda de memoria efectivo ejecutamos el programa 5 veces en DELL-XPS15 y hacemos la media:

Mediciones	Media
99,26188	99,343432
99,320792	
99,474064	
99,299472	
99,360952	

, de nuevo se ve que el ancho de banda efectivo de memoria es un poco peor que el teórico. Para el cálculo de la capacidad de cómputo seguimos el mismo procedimiento utilizado para GENMAGIC, obteniendo:

Mediciones	Media
16,543646	16,5572388
16,553466	
16,579011	
16,549912	
16,560159	

Con esto obtenemos una capacidad de cómputo media de 16,5572388 GFLOP/s para el programa de mediciones utilizado.

Si comparamos lo obtenido en GENMAGIC con lo obtenido en mi ordenador, vemos que los resultados dados por GENMAGIC son mucho mejores que los dados por mi ordenador, lo cual se debe en gran medida a que la gráfica de GENMAGIC es muy superior a la de mi ordenador.

Actividad 2:

EXPLICACIÓN CÓDIGO:

Lo que he hecho es dividir el vector cuyo máximo quiero calcular en tantas partes como bloques se estén utilizando. El cálculo de la posición inicial a comprobar y el número de posiciones a comprobar lo hace la hebra del bloque con identificador 0. Seguidamente cada bloque busca el máximo dentro del trozo de vector que le corresponde, para ello lo que se hace es que cada hebra recorre las posiciones *threadIdx.x+n*blockDim.x,* donde $n \geq 0$, de forma que cada celda del trozo asignado al bloque es comprobada una única vez por una única hebra.

En el siguiente ejemplo se aprecia como se divide un vector de tamaño 1000 entre 200 bloques, de forma que cada bloque tiene que calcular un máximo local en un subvector de 5 números.

VECTOR										
1	2	3	4	5	….	996	997	998	999	1000
Bloque 1					….	Bloque 200				

En este segundo ejemplo se ve qué posiciones del subvector comprueba cada hebra para 5 hebras por bloque y un subvector de tamaño 1000. De esta forma la hebra una comprueba la posición 1, 6, 11, 16, …, 996.

BLOQUE 1										
1	2	3	4	5	…	996	997	998	999	1000
Hebra 1	Hebra 2	Hebra 3	Hebra 4	Hebra 5	…	Hebra 1	Hebra 2	Hebra 3	Hebra 4	Hebra 5

Cada hebra compara si las posiciones del vector que comprueba son mayores que el máximo local del bloque, en caso de serlo actualiza el máximo local atómicamente. Tras tener calculado el máximo local se comprueba si el máximo local del bloque es mayor que el máximo global (el que tenemos que encontrar, y en este caso, el máximo que hemos encontrado de momento), y en caso de serlo se actualiza el máximo global atómicamente. De esta forma en la variable *max* nos queda el máximo de todos lo máximos locales calculados por los distintos bloques. Para que esta operación no la repitan todas las hebras, se ha puesto una barrera tras el cálculo del máximo local y esta última comprobación solo la realiza la hebra del bloque con identificación 0.

Observación: para esta actividad se debe tener cuidado en la elección del tamaño del vector, ya que si por ejemplo en GENMAGIC tomamos un vector de double de tamaño $1{,}44 * 10^9$ prácticamente ocupamos los 11GB de memoria de los que dispone la gráfica. Luego no podemos elegir tamaños

mucho más grande, pues la gráfica no dispondría de memoria suficiente para alojar el vector, dando lugar a errores.

```
estudiante20@genmagic:~/PRACTICA_4$ ./deviceQuery
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA GeForce RTX 2080 Ti"
  CUDA Driver Version / Runtime Version          11.4 / 11.7
  CUDA Capability Major/Minor version number:    7.5
  Total amount of global memory:                 11018 MBytes (11553341440 bytes)
  (068) Multiprocessors, (064) CUDA Cores/MP:    4352 CUDA Cores
  GPU Max Clock rate:                            1545 MHz (1.54 GHz)
  Memory Clock rate:                             7000 Mhz
  Memory Bus Width:                              352-bit
```

Actividad 3:

EXPLICACIÓN CÓDIGO:

De una forma análoga a lo hecho para la actividad 2, lo que se hace en este producto de matrices es dividir las filas de la matriz resultado entre el número de bloques que realizan el cálculo. De esta forma cada bloque calcula un número determinado de filas del resultado. Dentro del cálculo de cada fila lo que se hace es que cada hebra del bloque calcule las columnas *threadIdx.x+n*blockDim.x* donde $n \geq 0$.

MATRIZ A						MATRIZ B						MATRIZ C				
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	*	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$	$b_{1,4}$	$b_{1,5}$	=	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$		$b_{2,1}$	$b_{2,2}$	$b_{2,3}$	$b_{2,4}$	$b_{2,5}$		$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$
$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$		$b_{3,1}$	$b_{3,2}$	$b_{3,3}$	$b_{3,4}$	$b_{3,5}$		$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$
$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$		$b_{4,1}$	$b_{4,2}$	$b_{4,3}$	$b_{4,4}$	$b_{4,5}$		$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$
$a_{5,1}$	$a_{5,2}$	$a_{5,3}$	$a_{5,4}$	$a_{5,5}$		$b_{5,1}$	$b_{5,2}$	$b_{5,3}$	$b_{5,4}$	$b_{5,5}$		$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$

En este ejemplo tenemos una matriz 5x5 por otra matriz 5x5 y suponemos que tenemos 5 bloques de hebras y 5 hebras por bloque. Si nos fijamos en el esquema de arriba cada color del mismo tipo representa un bloque, es decir, todos los colores rojos son el bloque 1, los verdes el bloque 2 y así sucesivamente. Por otro lado dentro de cada color tenemos varias tonalidades distintas, cinco rojos, cinco verdes, cinco azules, etc, donde cada tonalidad representa una hebra dentro del bloque.

Luego se puede apreciar que en este ejemplo tenemos cinco bloque con cinco hebra cada uno. Cada bloque calcula una fila y cada celda de la fila es calculada por una hebra. Si en vez de cinco hebras tuvieramos solo dos hebras por bloque entonces tendríamos solo dos tonalidades de cada color y se daría lo siguiente:

MATRIZ A						MATRIZ B						MATRIZ C				
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	*	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$	$b_{1,4}$	$b_{1,5}$	=	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$		$b_{2,1}$	$b_{2,2}$	$b_{2,3}$	$b_{2,4}$	$b_{2,5}$		$c_{2,1}$	$c_{2,2}$	$c_{2,3}$	$c_{2,4}$	$c_{2,5}$
$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$		$b_{3,1}$	$b_{3,2}$	$b_{3,3}$	$b_{3,4}$	$b_{3,5}$		$c_{3,1}$	$c_{3,2}$	$c_{3,3}$	$c_{3,4}$	$c_{3,5}$
$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$		$b_{4,1}$	$b_{4,2}$	$b_{4,3}$	$b_{4,4}$	$b_{4,5}$		$c_{4,1}$	$c_{4,2}$	$c_{4,3}$	$c_{4,4}$	$c_{4,5}$
$a_{5,1}$	$a_{5,2}$	$a_{5,3}$	$a_{5,4}$	$a_{5,5}$		$b_{5,1}$	$b_{5,2}$	$b_{5,3}$	$b_{5,4}$	$b_{5,5}$		$c_{5,1}$	$c_{5,2}$	$c_{5,3}$	$c_{5,4}$	$c_{5,5}$

Vemos que solo hay dos tonalidades, es decir, dos hebras y que cada hebra se encarga de las columnas *threadIdx.x+blockDim.x*n,* donde $n \geq 0$.

El número de filas que calcula cada bloque y cual es la primera fila que se calcula, lo calcula le hebra del bloque con identificador 0. Con este método no es necesario realizar operaciones atómicas , pues las escrituras de cada posición de la matriz resultado las hace todas la misma hebra, de forma que lo calculado por una hebra no se va a ver alterado por las restantes.

Destacar que en la comprobación de si el producto es correcto, al comparar con el obtenido por la CPU no se puede hacer directamente la comprobación de si las celdas son iguales una a una, pues las operaciones si bien son asociativas en matemáticas no lo son en operaciones con punto flotante en el ordenador, luego a pesar de que los números obtenidos son prácticamente idénticos fallan en el decimal número 12-13, de ahí que hay que la comprobación sea que cada celda se encuentre extremadamente cerca de lo calculado por la CPU.