ACAP

Práctica 4

Antonio Priego Raya

1. Recoger cudaGetDeviceProperties()

Alojo los datos en data/ejecución/InfoGPUs. Archivo que se genera con cada ejecución de bin/cuda. En mi caso, arroja esta información:

```
./bin/cuda se ejecutara en
  ||
  |└Device Number: 0
  |  ├Device name: GeForce GTX 950M
  |  ├Memory Clock Rate (KHz): 900000
  |  ├Memory Bus Width (bits): 128
  |  ├Peak Memory Bandwidth (GB/s): 28.800000
  |  └Max threads per block: 1024
```

2. Código

El procedimiento a implementar en código, va a ser bien sencillo, tanto para GPU como para CPU. Puede consultarse en src/ .

1)Leer ficheros para obtención de vectores A y B
2)Reservar memoria
3)Hacer operaciones
4)Salvar fichero (vector C)

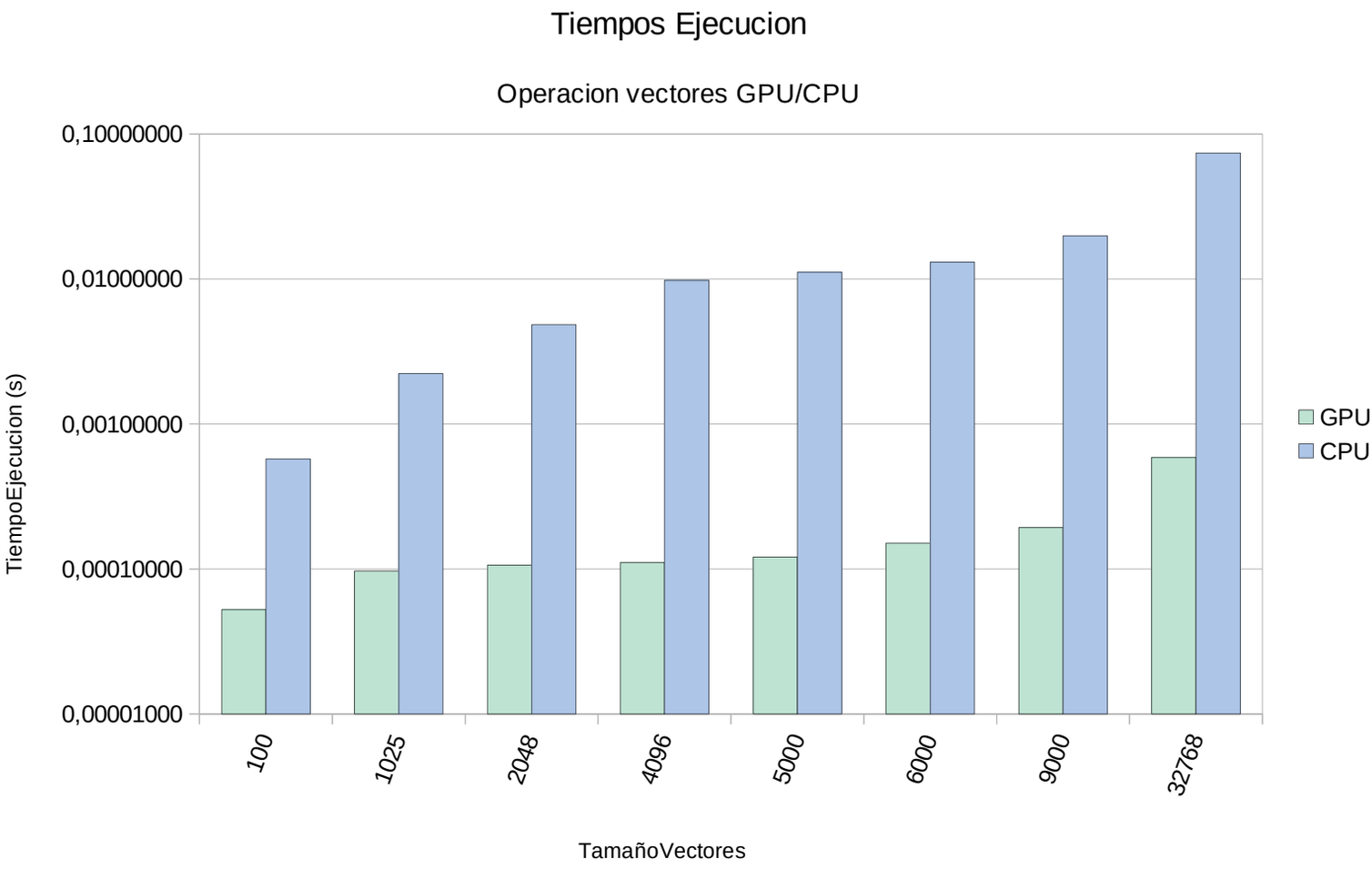
3. Obtención de tiempos

He medido los tiempos, en ambos casos, de gestión de memoria y de operación de vectores. Analizando en última instancia la suma de ambos. Estos pueden consultarse en data/ejecucion/CUDA y data/ejecucion/secuencial, y sintetizados en data/Resultados.ods.

4. Automatización

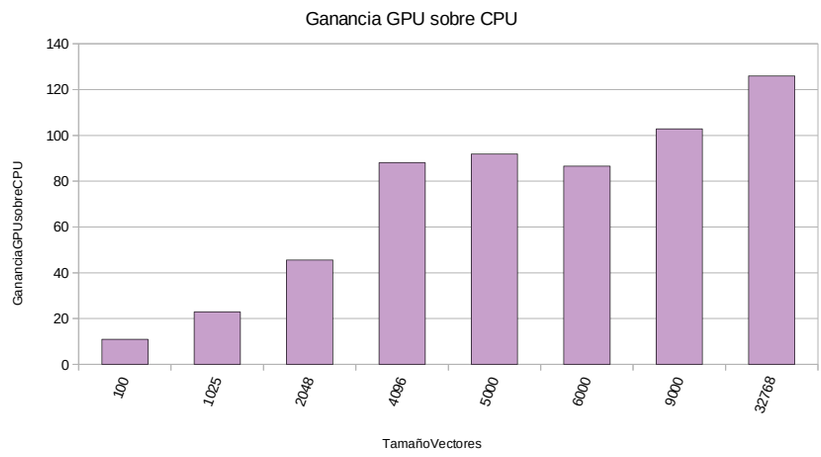
Ejecutando el script en ./recop_tiempos_ejec realizamos la compilación y ejecución para todas las muestras de ambos programas. Resultados de operación en data/vectorC y el resto de información de la ejecución en data/ejecucion/.

5. Análisis de resultados



Antes de comenzar, aclarar que el *eje Y* se visualiza con una escala logarítmica, para poder observar con claridad ambos comportamientos.

Los tiempos obtenidos cuadran con lo que cabría esperar, y es que, para una muestra pequeña, vemos que la disparidad de tiempos entre CPU y GPU no es tan grande. Esta se va evidenciando a medida que trabajamos con tamaños de vectores mayores. Como vemos de forma más remarcada en este gráfico de la derecha, siempre obtenemos una mejora, si bien las ganancias para tamaños mayores son aun mayores; llegando a ser unas 130 veces mejor para vectores de 32769 valores.



Otra cosa que se llega a apreciar, es que la ganancia no para de crecer. Y la hipótesis que podemos extraer es que lo seguirá haciendo hasta llegar a un punto en el que se estabilice y permanezca constante,

al saturar el uso de la GPU. Si bien ese límite está fuera de las muestras que estamos estudiando.

Percibimos observando los datos crudos obtenidos, que en la GPU, los tiempos de ejecución de operación con vectores prácticamente permanecen impasibles sin depender del tamaño del vector. Lo que resulta determinante para que aumente el tiempo que se emplea en el proceso, es la gestión de memoria, el trasladar los datos de los ficheros de entrada a memoria para trabajar con ellos.
No ocurre lo mismo con la CPU, donde el tiempo de operaciones aumenta su protagonismo en el tiempo total a medida que el tamaño del vector crece.

Un elemento a estudiar es a partir de qué tamaño de muestra sería más conveniente pasar a hacer uso de secuencial a CUDA.
En nuestro caso, de todos los tamaños de muestra extraemos mejores resultados con CUDA, por lo tanto, como mínimo, CUDA es mejor hasta muestras de 100 elementos.

TABLA DE DATOS EXTENDIDA (PROMEDIO MUESTRAS)			
GPU		CPU	
Tamanio vector: 100			
Tiempo ejecucion:	**0,00001186**	Tiempo ejecucion:	**0,00004072**
Tiempo R/W memoria:	**0,00004062**	Tiempo R/W memoria:	**0,00053345**
Tiempo total:	**0,00005248**	Tiempo total:	**0,00057417**
Tamanio vector: 4096			
Tiempo ejecucion:	**0,00001249**	Tiempo ejecucion:	**0,00073075**
Tiempo R/W memoria:	**0,00009829**	Tiempo R/W memoria:	**0,00903448**
Tiempo total:	**0,00011078**	Tiempo total:	**0,00976523**
Tamanio vector: 32768			
Tiempo ejecucion:	**0,00001287**	Tiempo ejecucion:	**0,00526379**
Tiempo R/W memoria:	**0,00057425**	Tiempo R/W memoria:	**0,06872597**
Tiempo total:	**0,00058712**	Tiempo total:	**0,07398976**
Tamanio vector: 100			
Tiempo ejecucion:	**0,00001161**	Tiempo ejecucion:	**0,00005372**
Tiempo R/W memoria:	**0,00004366**	Tiempo R/W memoria:	**0,00049403**
Tiempo total:	**0,00005527**	Tiempo total:	**0,00054775**
Tamanio vector: 9000			
Tiempo ejecucion:	**0,00001440**	Tiempo ejecucion:	**0,00153835**
Tiempo R/W memoria:	**0,00018920**	Tiempo R/W memoria:	**0,02005889**
Tiempo total:	**0,00020361**	Tiempo total:	**0,02159724**
Tamanio vector: 1025			
Tiempo ejecucion:	**0,00001265**	Tiempo ejecucion:	**0,00017635**
Tiempo R/W memoria:	**0,00008424**	Tiempo R/W memoria:	**0,00205009**
Tiempo total:	**0,00009689**	Tiempo total:	**0,00222644**
Tamanio vector: 2048			
Tiempo ejecucion:	**0,00001181**	Tiempo ejecucion:	**0,00037452**
Tiempo R/W memoria:	**0,00009427**	Tiempo R/W memoria:	**0,00446759**
Tiempo total:	**0,00010608**	Tiempo total:	**0,00484211**
Tamanio vector: 5000			
Tiempo ejecucion:	**0,00001193**	Tiempo ejecucion:	**0,00079639**
Tiempo R/W memoria:	**0,00010903**	Tiempo R/W memoria:	**0,01032401**
Tiempo total:	**0,00012096**	Tiempo total:	**0,01112041**
Tamanio vector: 6000			
Tiempo ejecucion:	**0,00001226**	Tiempo ejecucion:	**0,00096553**
Tiempo R/W memoria:	**0,00015070**	Tiempo R/W memoria:	**0,01210230**
Tiempo total:	**0,00016296**	Tiempo total:	**0,01306783**
Tamanio vector: 9000			
Tiempo ejecucion:	**0,00001247**	Tiempo ejecucion:	**0,00146397**
Tiempo R/W memoria:	**0,00018094**	Tiempo R/W memoria:	**0,01842230**
Tiempo total:	**0,00019341**	Tiempo total:	**0,01988627**

6. Conclusiones

- El resultado de la práctica es que obtenemos unos muy buenos tiempos en CUDA, ya que el cómputo que se está realizando es suficientemente simple.
- La GPU con unos tamaños de vector pequeños, aunque se sigue desenvolviendo mejor que mi CPU, está siendo desaprovechada. Esto se entiende con claridad al comprobar que los tiempos de ejecución (operación de vectores, no gestión de memoria), son prácticamente idénticos del primer al último tamaño de muestra estudiado. Tendríamos que salir de estos tamaños para realmente aprovechar todo su potencial.
- La ganancia de GPU respecto de CPU no deja de crecer hasta un tamaño de muestra que sature la GPU y esta comience a permanecer prácticamente constante.
- Sabemos que con valores pequeños de muestra, la CPU debería superar a la GPU, en un contexto donde las prestaciones entre ambas son proporcionales. Pero no hemos estudiado un tamaño de vector suficientemente pequeño para que se de este suceso.