Primeros pasos con CUDA

Programa que eleva un vector al cuadrado

Programa "normal"

```
#include<stdio.h>
#define  N 1000

int main()
{
int       i,N;
float     x;
float     y[N];

for (i=0;i<N;i++)
   {
   x=(float)i;
   y[i]=x*x;
   }
}
```

Compilador gcc

Usamos un compilador *standard*

```
gcc -Wall -lm -o programa.e programa.c
```

... y corremos ... `./programa.e`

Primeros pasos en CUDA

¿Qué modificaciones tenemos que hacerle para CUDA?

Esquema básico

- Inicializar las variables de manera acostumbrada.

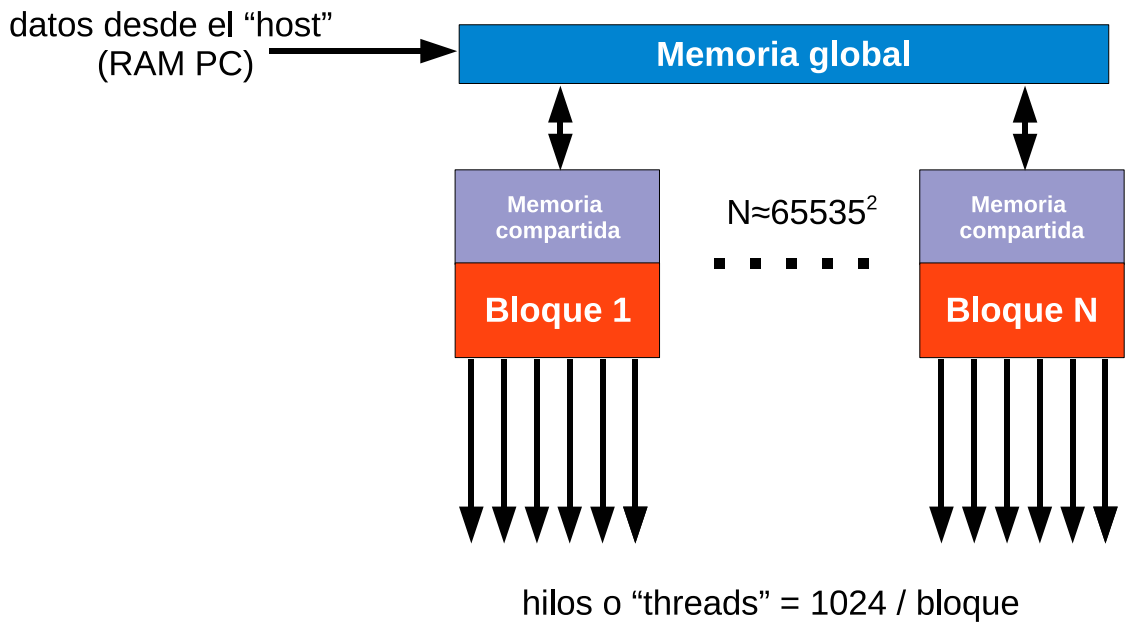
```
#include<stdio.h>
#include<cuda.h> → librería de CUDA (¡obvio!)
#define N 1000

int main()
{
int    i;
size_t s;            → memoria en CUDA
float  *y,*y_cuda;   → variable en CUDA
```

Primeros pasos en CUDA

¿Qué limitaciones de memoria tenemos con CUDA?

- Las funciones de CUDA sólo operan sobre la memoria de la placa gráfica.



La memoria compartida es del orden de 50KBytes.
La memoria global es del orden de 1GByte.

$\Rightarrow$ hay que inicializar la memoria de la placa gráfica.

Primeros pasos en CUDA

¿Qué modificaciones tenemos que hacerle para CUDA?

Programa principal

```
int main()
{
.....
s=N*sizeof(float);
y=(float *)malloc(s);
cudaMalloc((void **)&y_cuda,s);

elevar_cuadrado<<<1,N>>>(y_cuda,N);

cudaMemcpy(y,y_cuda,s,cudaMemcpyDeviceToHost);
cudaFree(y_cuda);
free(y);
}
```

- La función "`elevar_cuadrado`" realiza el cálculo en paralelo y se debe definir en el encabezado del programa.

Primeros pasos en CUDA

Funciones en CUDA

Declaración

```
__global__ salida nombre de la función(argumentos);
{
....
}
```

Invocación

<<<bloques,hilos/bloque>>>

nombre de la función <<<1,N>>> (*argumentos*);

elevar_cuadrado (y_cuda,N)

Primeros pasos en CUDA

Reglas para la asignación de bloques

- La cantidad máxima de hilos/bloque es 1024.
- La memoria compartida dentro de cada bloque tiene menor latencia que la memoria global. Cada hilo accede a la memoria compartida de manera asincrónica. Por lo tanto, puede llegar a ser necesario sinronizar los hilos con llamadas a la función `__syncthreads()`
- Los procesos entre bloques son independientes. La placa gráfica administra la cantidad necesaria de bloques en cada etapa del programa.
- En general, para `<<<n_blocks,block_size>>>`

  se usa la fórmula

  `n_blocks=N/block_size+(N%block_size==0?0:1);`

Primeros pasos en CUDA

Declaración de una función CUDA

```
__global__ void elevar_cuadrado(float *y_cuda,int N)
{
  int i;

  i=threadIdx.x;
  if (i<N)  y_cuda[i]=(float)i*(float)i;
}
```

Observaciones

- Los hilos se numeran de 0 a N.
- También exiten `threadIdx.y` , `threadIdx.z`
- En general, para recorrer los hilos se usa la fórmula:

```
i=blockIdx.x * blockDim.x + threadIdx.x;
```

Primeros pasos en CUDA

Trucos y trampas

- CUDA sólo reconoce archivos fuente con la extensión ".cu"
- La memoria de la placa gráfica es de 32 bits. Por lo tanto, muchos modelos sólo soportan datos "float" (32 bits) en lugar de "double" (64 bits). Nvidia promete solucionar definitivamente este problema.
- Las funciones CUDA deben declararse al comienzo del programa, antes de `main()`.
- CUDA sólo trabaja con "dynamic memory allocation" de punteros.

Primeros pasos en CUDA

Resultados para un único bloque de 1024 hilos (en μseg.)

Operación	secuencial	paralelo	porcentaje
x^2	3.88	1.76	45%
$1/x$	3.92	1.78	45%
$\sqrt{x}$	13.60	1.81	13%
e^{-x}	26.71	2.11	8%
Potenciales:			
$1/x^2$	8.06	1.79	22%
$4(x^{-12} - x^{-6})$	136.3	2.24	2%

Dinámica molecular con CUDA

Un único bloque de 1024 hilos (partículas)

Programa simple

```
int main()
{
 ...
   inicializacion<<<1,N>>>(x,v);

   t=0;
   while (t<T)
         {
         fuerzas<<<1,N>>>(x,f);
         verlet<<<1,N>>>(x,v);
         observables<<<1,N>>>(x,v,f);
         t=t+h;
         }
 ...
}
```