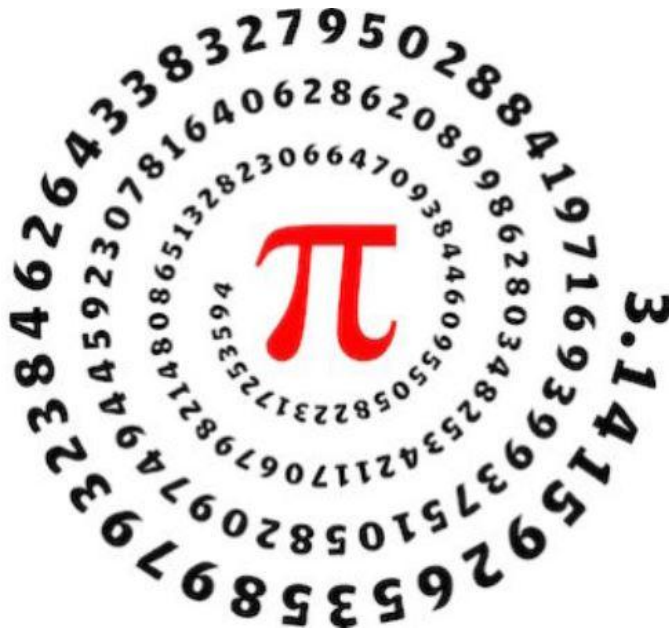


# Práctica 08.

## Cálculo del valor $\pi$ (irracional)



U.A.Q. Fac. de Informática  
**Dra. Sandra Luz Canchola Magdaleno**

Correo: [sandra.canchola@uaq.mx](mailto:sandra.canchola@uaq.mx)

**Dra. Reyna Moreno Beltrán**

Correo: [reyna.moreno@uaq.mx](mailto:reyna.moreno@uaq.mx)



# Valor de PI ( $\pi$ )

$$\frac{\pi^2}{6} = \sum_{i=1}^{n \rightarrow \infty} \frac{1}{i^2}$$

Donde  $n$  es el número de elementos a considerar, mientras se acerca a  $\infty$  es un valor más preciso.

$$\pi^2 = 6 \sum_{i=1}^{n \rightarrow \infty} \frac{1}{i^2}$$

$$\sqrt{\pi^2} = \sqrt{6 \sum_{i=1}^{n \rightarrow \infty} \frac{1}{i^2}}$$

$$\pi = \sqrt{6 \sum_{i=1}^{n \rightarrow \infty} \frac{1}{i^2}}$$

$$\frac{\pi^2}{6} = \sum_{i=1}^{n \rightarrow \infty} \frac{1}{i^2} = \underbrace{\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \frac{1}{6^2} + \dots + \frac{1}{n^2}}_{n \text{ términos}}$$

Términos (n)	Valor de $\pi$
100	3.132076531809105
1,000	3.140638056205995
10,000	3.141497163947215
100,000	3.141583104326456
1'000,000	3.141591698660509
10'000,000	3.141592558095903
100'000,000	3.141592644982390

$$\pi^2 = 6 \sum_{i=1}^{n \rightarrow \infty} \frac{1}{i^2}$$
$$\pi = \sqrt{6 \sum_{i=1}^{n \rightarrow \infty} \frac{1}{i^2}}$$

# Proyecto CUDA

---



## Create a new project

Choose a project template with code scaffolding to get started



### CUDA 11.7 Runtime

A project that uses the CUDA 11.7 runtime

C++

CUDA

Windows

Linux

Cloud

Console

DataScience

Desktop

Machine Learning



### CUDA 12.1 Runtime

A project that uses the CUDA 12.1 runtime

C++

CUDA

Windows

Linux

Cloud

Console

DataScience

Desktop

Machine Learning

# Proyecto CUDA

---

## Configure your new project

CUDA 12.1 Runtime

C++

CUDA

Windows

Linux

Cloud

Console

DataScience

Desktop

Machine Learning

Project name

Prog08\_CalculoPI

Location

C:\TrabajoLaboratorio\CUDATopico2\Projects\Verano2024

...

Solution name ⓘ

Prog08\_CalculoPI

☐ Place solution and project in the same directory

Project will be created in "C:\TrabajoLaboratorio\CUDATopico2\Projects\Verano2024  
\Prog08\_CalculoPI\Prog08\_CalculoPI"

Back

Create

# Operaciones de memoria (CPU)

- **malloc.**- Reserva un bloque de memoria de un tamaño definido de bytes, retornando un apuntador al inicio de dicho bloque. El contenido de dicho bloque no se inicializa por lo que es indeterminado. Ejemplo:

```
void* malloc (size_t size);  
buffer = (char*) malloc (sizeof(char)*100);
```

- **memset.**- asigna valores en secciones de memoria. Ejemplo:  
Memset(variable, valor\_a\_asignar, tamaño\_de\_memoria)  
Donde: tamaño\_de\_memoria se define como n \* sizeof(tipo)

- **memcpy.**- Copia el contenido de un bloque de memoria referenciado por un apuntador a otro apuntador. Ejemplo:

```
void* memcpy( void* dest, const void* src, std::size_t count );  
memcpy (ptrDest, ptrOrigen, sizeof(int)*100);
```

- **free.**- Liberar la memoria reservada con el comando malloc. Ejemplo:  
free(pointerName);  
free(array2);

# Operaciones de memoria (GPU)

- **cudaMalloc**.- asigna una sección de memoria en GPU de acuerdo con el espacio solicitado.

Ejemplo:

```
cudaMalloc((void**) &apuntador, tamaño_de_memoria)
```

Donde: tamaño\_de\_memoria se define como  $n * \text{sizeof}(\text{tipo})$

- **cudaMemset**.- asigna valores en secciones de memoria.

Ejemplo:

```
Memset(apuntador, valor_a_asignar, tamaño_de_memoria)
```

Donde: tamaño\_de\_memoria se define como  $n * \text{sizeof}(\text{tipo})$

- **cudaMemcpy**.- copia memoria hacia y desde el device.

Ejemplo:

```
cudaMemcpy(destino, origen, tamaño_de_memoria, indicador_flujo_de_inf)
```

Donde Indicador= cudaMemcpyHostToDevice, cudaMemcpyDeviceToHost, cudaMemcpyDeviceToDevice

- **cudaFree**.- libera la memoria reservada por un apuntador.

Ejemplo:

```
cudaFree (apuntador)
```

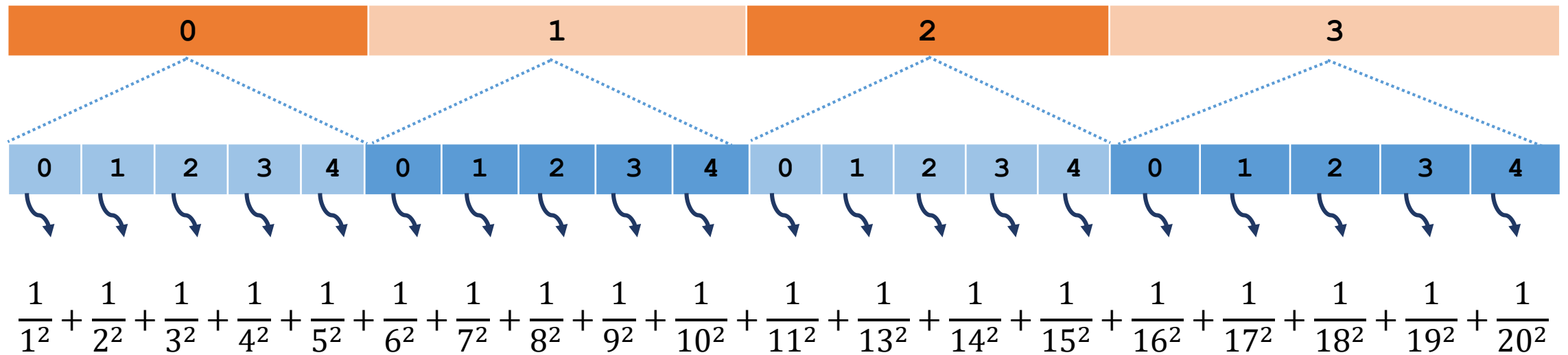
# Objetivo

- Definir al menos tres configuraciones de ejecución paralela para el cálculo cooperativo del valor de  $\pi$ , y comparar la precisión del cálculo con respecto al valor calculado en un proceso secuencias del CPU.

Configuración en GPU		Términos (n)	Valor de $\pi$			Tiempo de ejecución (ms)	
Grid	Block		CPU	GPU	Dif.	CPU	GPU



# Caso 1. X bloques con numHilos c/u



`tid=(blockIdx.x*blockDim.x)+threadIdx.x`

# Caso 1. X bloques con numHilos c/u

```
#define noTerminos 1000000000 // de la formula de PI
...
int numHilos = 1024;
...
int numBloques = divEntera(noTerminos, numHilos);
int numCalculos = noTerminos;
...
__global__ void calculoPI(double* a) {
    double valor = 0;
    int tid = (blockIdx.x*blockDim.x)+ threadIdx.x;
    if (tid < noTerminos) {
        valor = double(tid) + 1;
        a[tid] = 1.0 / (valor*valor);
    }
}
```

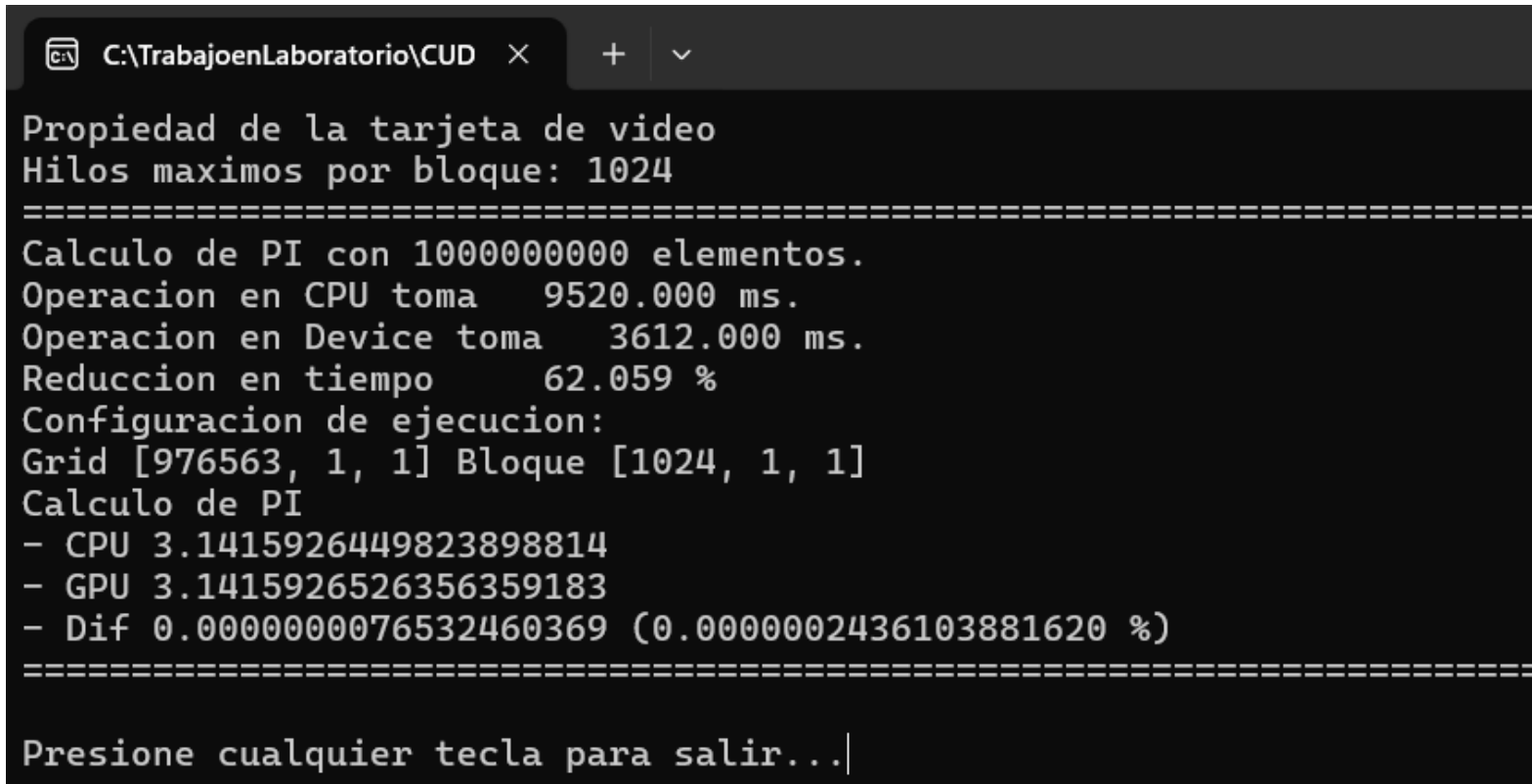
# Caso 1. X bloques con numHilos c/u

```
__global__ void sumarPI(double* a, double* acum, int numElem) {
    int tid = blockIdx.x;
    int inicio = (blockIdx.x * numElem);
    double suma = 0;
    for (int i = 0; i < numElem; i++) {
        suma = suma + a[inicio + i];
    }
    acum[tid] = suma;
}

__global__ void totalPI(double* acum, double* total, int numElem) {
    double suma = 0;
    for (int i = 0; i < numElem; i++) {
        suma = suma + acum[i];
    }
    *total = sqrt(6 * suma);
}
```

# Caso 1. X bloques con numHilos c/u

```
calculoPI << <dimGrid, dimBlock >> > (dElementos);  
sumarPI << < dimGrid, 1 >> > (dElementos, dSumaxBloque, numHilos);  
totalPI << <1, 1 >> > (dSumaxBloque, dTotal, numBloques);
```



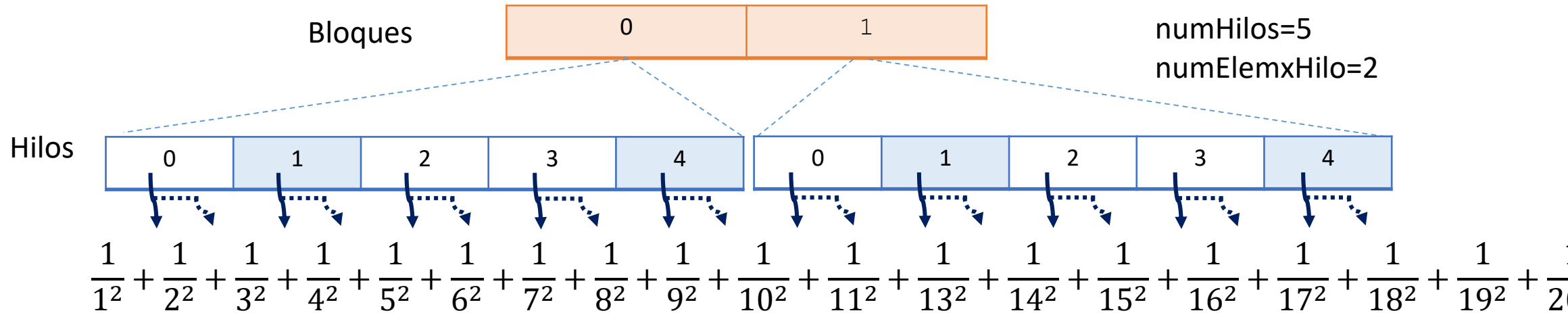
```
C:\TrabajoLaboratorio\CUD >  
Propiedad de la tarjeta de video  
Hilos maximos por bloque: 1024  
=====
```

Calculo de PI con 1000000000 elementos.	
Operacion en CPU toma	9520.000 ms.
Operacion en Device toma	3612.000 ms.
Reduccion en tiempo	62.059 %

```
Configuracion de ejecucion:  
Grid [976563, 1, 1] Bloque [1024, 1, 1]  
Calculo de PI  
- CPU 3.1415926449823898814  
- GPU 3.1415926526356359183  
- Dif 0.0000000076532460369 (0.0000002436103881620 %)  
=====
```

Presione cualquier tecla para salir...|

## Caso 2. X bloques con numHilos que atienden a numElemxHilo c/u



<b>blockIdx.x</b>	<b>threadIdx.x</b>	<b>tid</b>	<b>Terminos calculados</b>
0	0	0	0, 1
0	1	1	2, 3
0	2	2	4, 5
0	3	3	6, 7
0	4	4	8, 9
1	0	5	10, 11
1	1	6	12, 13
1	2	7	14, 15
1	3	8	16, 17
1	4	9	18, 19

$$\text{tid} = (\text{blockIdx.x} * \text{blockDim.x}) + \text{threadIdx.x}$$

$$\text{PrimerElemento} = \text{tid} * \text{numElemxHilo}$$

## Caso 2. X bloques con numHilos que atienden a numElemxHilo c/u

```
#define noTerminos 1000000000 // de la formula de PI
#define elemxHilo 5
...
int numHilos = 1024;
...
int numBloques = divEntera(noTerminos, numHilos * elemxHilo);
int numCalculos = numBloques * numHilos;
...
```

## Caso 2. X bloques con numHilos que atienden a numElemxHilo c/u

```
__global__ void calculoPI(double* a) {  
    double valor = 0;  
    double suma = 0;  
    int tid = (blockIdx.x * blockDim.x) + threadIdx.x;  
    int inicio = tid * elemxHilo;  
    if (inicio < noTerminos) {  
        for (int i = 0; i < elemxHilo; i++) {  
            if ((inicio + i) < noTerminos) {  
                valor = double(inicio + i) + 1;  
                suma = suma + (1.0 / (valor * valor));  
            }  
        }  
    }  
    a[tid] = suma;  
}
```



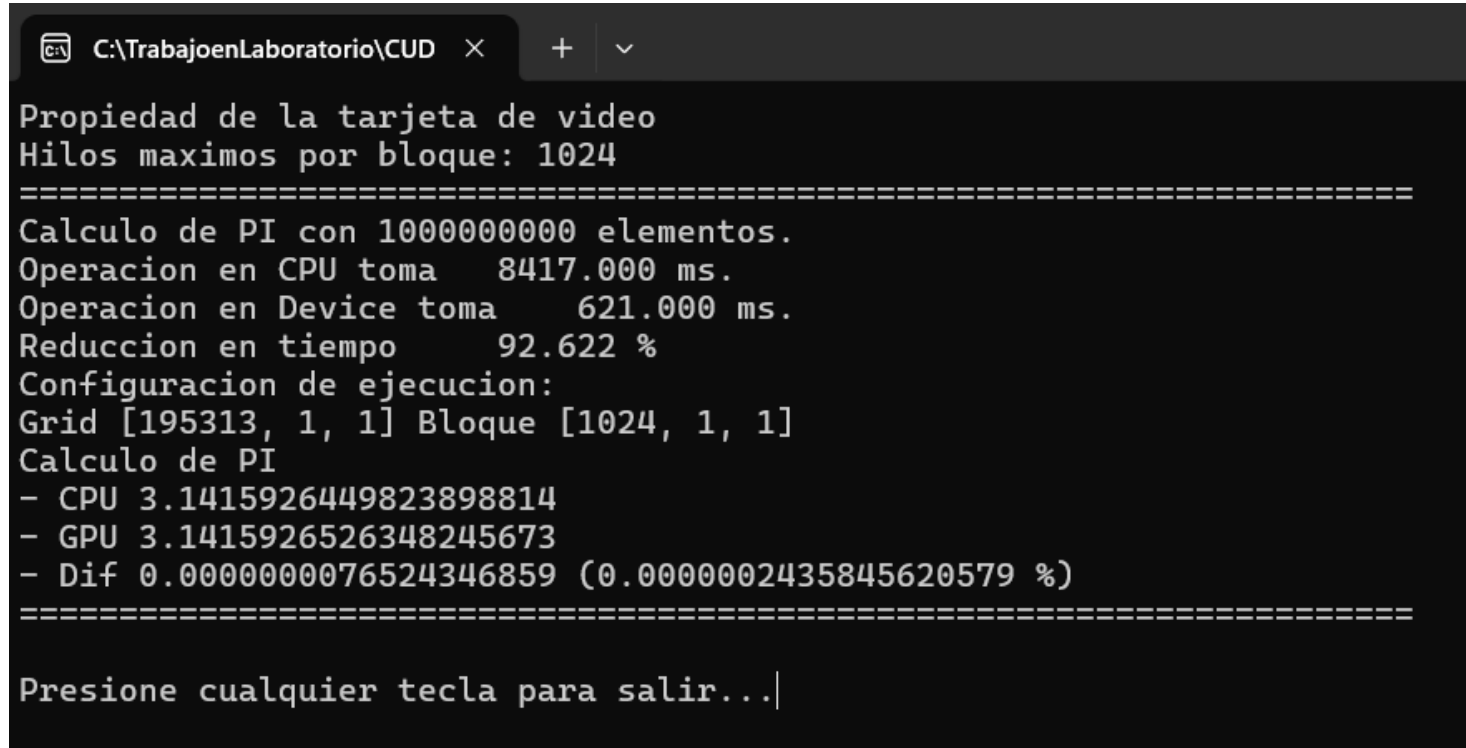
## Caso 2. X bloques con numHilos que atienden a numElemxHilo c/u

```
__global__ void sumarPI(double* a, double* acum, int numElem) {  
    int tid = blockIdx.x;  
    int inicio = (blockIdx.x * numElem);  
    double suma = 0;  
    for (int i = 0; i < numElem; i++) {  
        suma = suma + a[inicio + i];  
    }  
    acum[tid] = suma;  
}
```

```
__global__ void totalPI(double* acum, double* total, int numElem) {  
    double suma = 0;  
    for (int i = 0; i < numElem; i++) {  
        suma = suma + acum[i];  
    }  
    *total = sqrt(6 * suma);  
}
```

# Caso 2. X bloques con numHilos que atienden a numElemxHilo c/u

```
calculoPI << <dimGrid, dimBlock >> > (dElementos);  
sumarPI << < dimGrid, 1 >> > (dElementos, dSumaxBloque, numHilos);  
totalPI << <1, 1 >> > (dSumaxBloque, dTotal, numBloques);
```



```
C:\TrabajoLaboratorio\CUD >  
Propiedad de la tarjeta de video  
Hilos maximos por bloque: 1024  
=====  
Calculo de PI con 10000000000 elementos.  
Operacion en CPU toma 8417.000 ms.  
Operacion en Device toma 621.000 ms.  
Reduccion en tiempo 92.622 %  
Configuracion de ejecucion:  
Grid [195313, 1, 1] Bloque [1024, 1, 1]  
Calculo de PI  
- CPU 3.1415926449823898814  
- GPU 3.1415926526348245673  
- Dif 0.0000000076524346859 (0.0000002435845620579 %)  
=====  
Presione cualquier tecla para salir...|
```

# Bibliografía

---

- Documentación **CUDA C++ Programming Guide** NVIDIA. 2024  
<https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>
- Sitio **CUDA Toolkit Documentation** NVIDIA, 2024.  
<https://docs.nvidia.com/cuda/index.html>
- Storti, Duane; Yurtoglu, Mete. **CUDA for Engineers:An Introduction to High-Performance Parallel Computing**. Addison Wesley. 2015.
- Cheng, John; Grossman, Max; McKercher. **Professional CUDA C Programming**. Edit. Wrox. 2014.
- Sanders, Jason; Kandrot, Edward. **CUDA by Example:An Introduction to General-Purpose GPU Programming**. Addison Wesley. 2011.
- Kirk, David; Hwu, Wen-mei. **Programming Massively Parallel Processors: A Hands-on Approach**. Elsevier. 2010.

---

Gracias por su atención



**U.A.Q. Fac. de Informática  
Campus Juriquilla**

**Dra. Sandra Luz Canchola Magdaleno  
sandra.canchola@uaq.mx  
Cel. 442-1369270**

**Dra. Reyna Moreno Beltrán  
reyna.moreno@uaq.mx**