

# COMPUTE UNIFIED DEVICE ARCHITECTURE (CUDA)

Francisco J. Hernández López

fcoj23@cimat.mx



# TARJETA DE VIDEO O UNIDAD DE PROCESAMIENTO GRÁFICO (GPU)

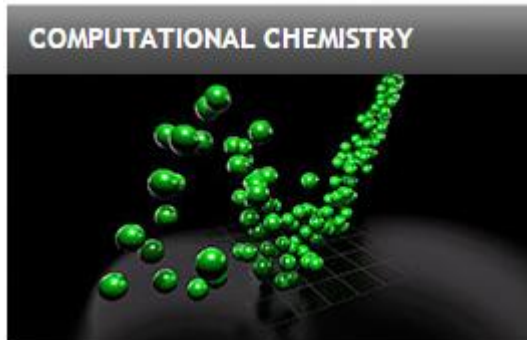




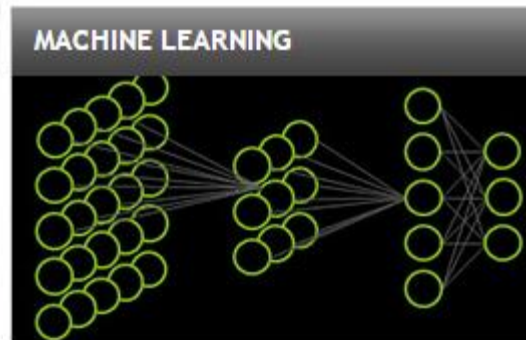
# GPUS



- Procesadores flexibles de procesamiento general
- Se pueden resolver problemas de diversas áreas:
  - Finanzas, Gráficos, Procesamiento de Imágenes y Video, Álgebra Lineal, Física, Química, Biología, etc.



COMPUTATIONAL CHEMISTRY



MACHINE LEARNING



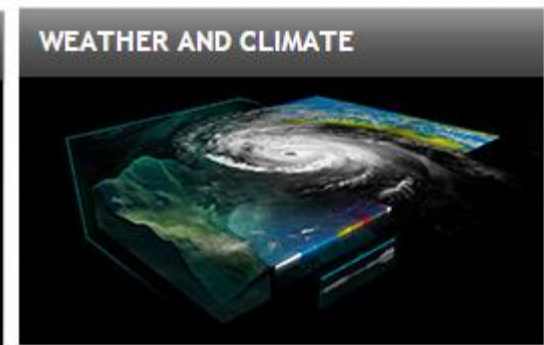
DATA SCIENCE



BIOINFORMATICS



COMPUTATIONAL FLUID DYNAMICS

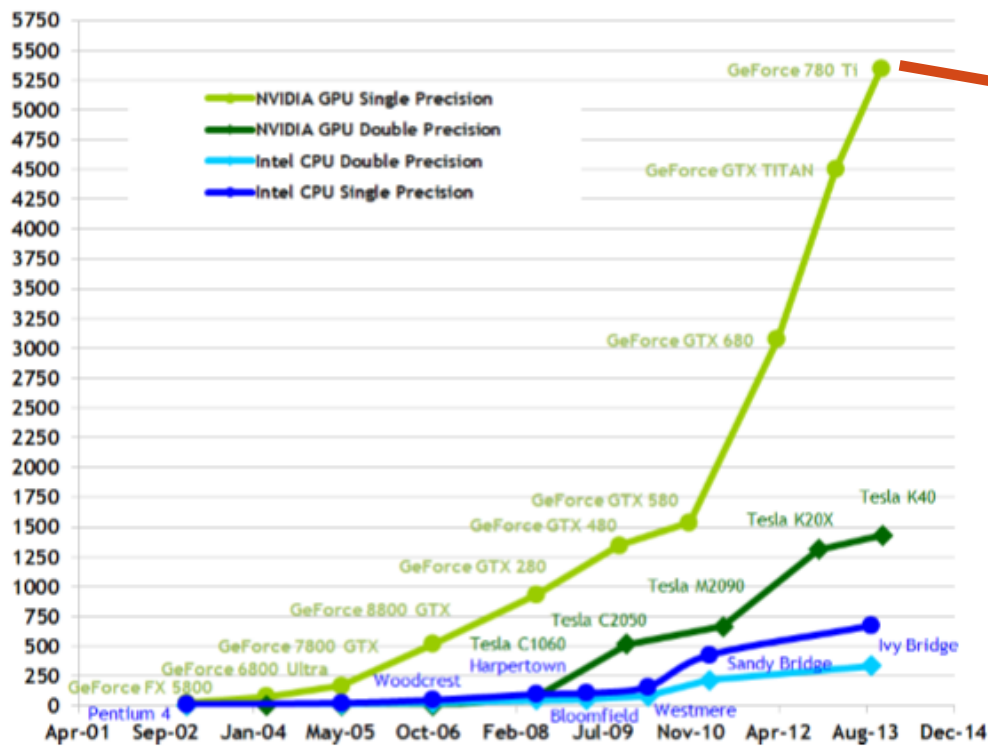


WEATHER AND CLIMATE

Visitar CUDA ZONE: <https://developer.nvidia.com/cuda-zone>

# GPUS VS CPUS

Theoretical GFLOP/s

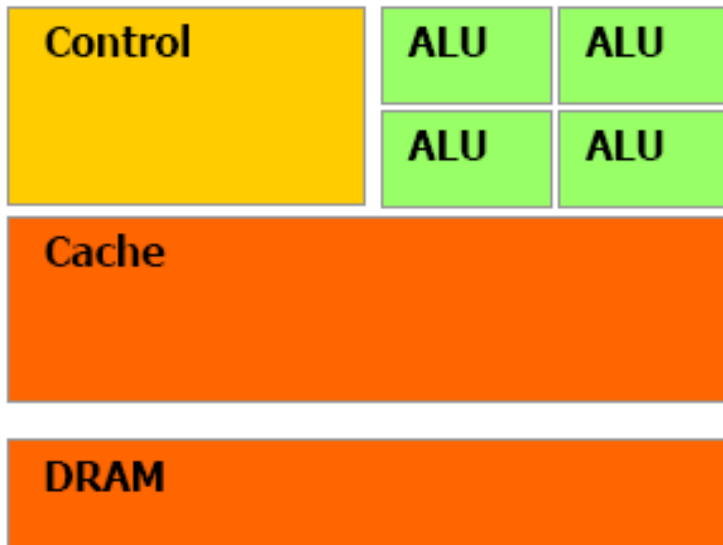


GeForce GTX 780Ti

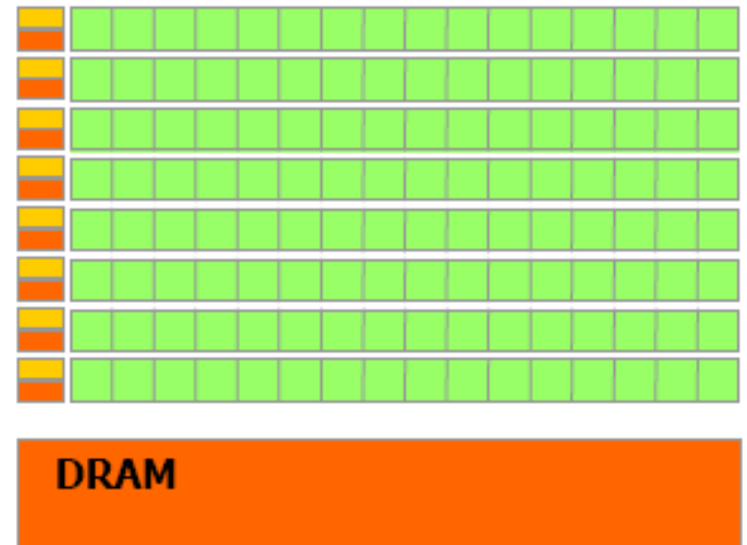
CUDA\_C\_Programming\_Guide.pdf

# GPUS VS CPUS (C1)

- Las GPUs cuentan con mayor número de transistores para procesar los datos



**CPU**



**GPU**

# HERRAMIENTAS PARA CÓMPUTO PARALELO USANDO LA GPU

- **CUDA** (**C**ompute **U**nified **D**evice **A**rchitecture). Desarrollado por NVIDIA en el 2006, como uno de los primeros sistemas de programación en GPU que se liberaron de la forma que había para programar en una GPU (code-it-as-graphics, Cg). Compatible con GPUs Nvidia.
- **OpenCL** (**O**pen **C**omputing **L**anguage). Es un estándar abierto para desarrollar programas que pueden ejecutarse en plataformas heterogéneas, incluyendo GPUs (Nvidia o AMD), CPU, DSPs (Digital Signal Processors). Su modelo de programación es muy parecido al de CUDA.
- **OpenACC**. Permite el uso de directivas para el compilador, para mapear automáticamente cálculos a la GPU o a un multicore.

# HERRAMIENTAS PARA CÓMPUTO PARALELO USANDO LA GPU (C1)

- **Thrust.** Es una librería de plantillas en C++ que acelera el desarrollo de programas en GPU, utilizando un conjunto de clases y un conjunto de algoritmos que automáticamente se ejecutan en la GPU. Desde la versión 1.6, puede lanzar ejecuciones a la GPU o a la CPU. Está incorporado en el SDK de CUDA desde la versión 4 de CUDA.
- **ArrayFire.** Es una librería completa de funciones para el GPU que cubre: Matemáticas, Procesamiento de imágenes y señales, Estadística, y otros dominios científicos. Opera en arreglo de datos de forma similar que Thrust.
- **C++ AMP (C++ Accelerated Massive Parallelism).** Tecnología de Microsoft basado en DirectX 11. Permite la ejecución transparente del código C++ en una CPU o una GPU con base en un conjunto de directivas o extensiones del lenguaje. El modelo de programación es similar al de OpenMP.

Barlas, G. (2014). *Multicore and GPU Programming: An integrated approach*. Elsevier.

# CUDA

- Es una tecnología de propósito general que nos permite ejecutar código en GPUs para hacer Cómputo Paralelo
- Desarrollado por NVIDIA en el 2006
- Soporta los lenguajes de programación C/C++, Fortran, Matlab, Python, LabView, etc.
- Soporte de datos en paralelo y manejador de hilos.
- Librerías:
  - FFT (Fast Fourier Transform)
  - BLAS (Basic Linear Algebra Subroutines)
  - CURAND (Generar números aleatorios)
  - CUSPARSE (Subrutinas de algebra lineal para operar matrices ralas)
  - NPP (NVIDIA Performance Primitives)...
- Opera internamente con OpenGL y DirectX.
- Soporta los sistemas operativos:
  - Windows XP 32/64-bit, Windows Vista 32/64-bit, Windows 7 32/64-bit, Linux 32/64-bit y Mac OS.



## GPU Computing Applications

### Libraries and Middleware

cuDNN TensorRT	cuFFT, cuBLAS, cuRAND, cuSPARSE	CULA MAGMA	Thrust NPP	VSIPL, SVM, OpenCurrent	PhysX, OptiX, iRay	MATLAB Mathematica
-------------------	------------------------------------	------------	---------------	----------------------------	-----------------------	-----------------------

### Programming Languages

C	C++	Fortran	Java, Python, Wrappers	DirectCompute	Directives (e.g., OpenACC)
---	-----	---------	---------------------------	---------------	-------------------------------

### CUDA-enabled NVIDIA GPUs

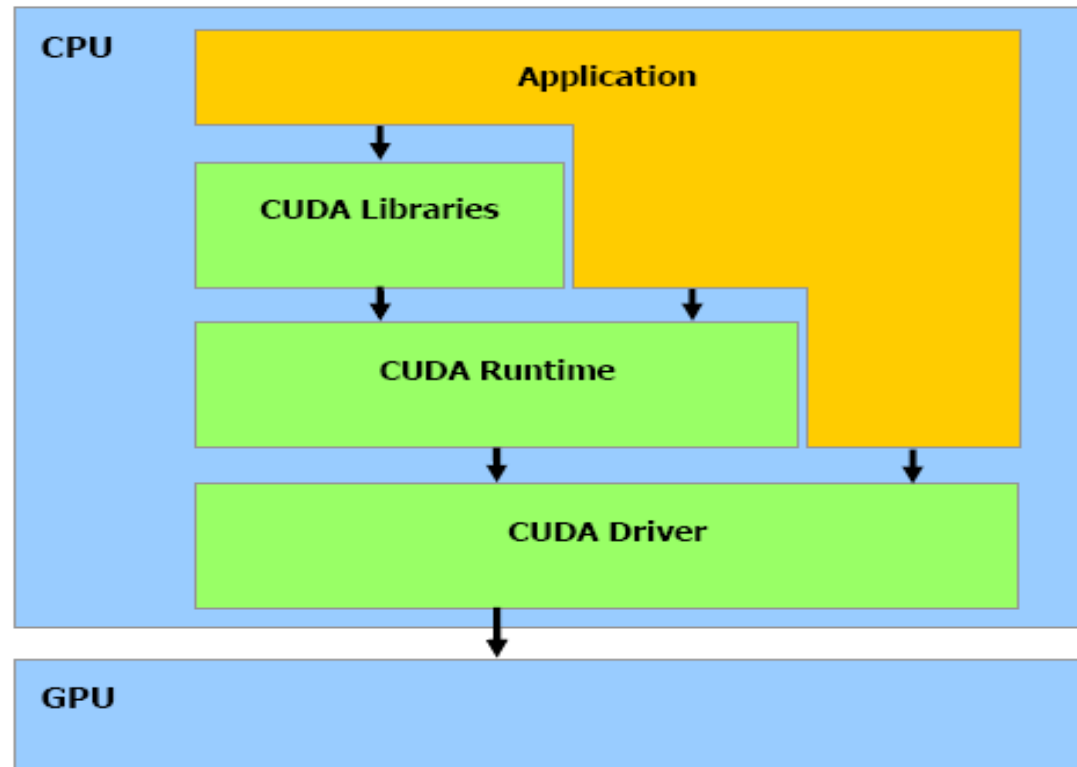
Turing Architecture (Compute capabilities 7.x)	DRIVE/JETSON AGX Xavier	GeForce 2000 Series	Quadro RTX Series	Tesla T Series
Volta Architecture (Compute capabilities 7.x)	DRIVE/JETSON AGX Xavier			Tesla V Series
Pascal Architecture (Compute capabilities 6.x)	Tegra X2	GeForce 1000 Series	Quadro P Series	Tesla P Series
Maxwell Architecture (Compute capabilities 5.x)	Tegra X1	GeForce 900 Series	Quadro M Series	Tesla M Series
Kepler Architecture (Compute capabilities 3.x)	Tegra K1	GeForce 700 Series GeForce 600 Series	Quadro K Series	Tesla K Series
	EMBEDDED	CONSUMER DESKTOP, LAPTOP	PROFESSIONAL WORKSTATION	DATA CENTER

# SOFTWARE USANDO CUDA



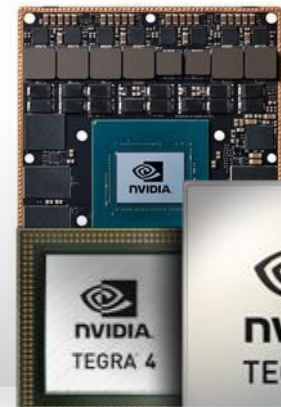
# SOFTWARE CUDA

- El software CUDA esta compuesto por:
  - Hardware driver
  - Runtime
  - Libraries



CUDA\_C\_Programming\_Guide.pdf

# GPUS COMPATIBLES CON CUDA



Arquitectura	Capacidad	Ejemplos	Año
Tesla	1.0 - 1.3	GeForce 8800 GT, Quadro FX 370	2006
Fermi	2.0 - 2.1	GeForce GTX 480, Quadro 2000	2010
Kepler	3.0 - 3.5	Tesla K20, NVS 510, Tegra K1	2012
Maxwell	5.0 - 5.2	GeForce GTX 980M, Quadro M6000	2014
Pascal	6.0 - 6.1	GeForce GTX 1080, Quadro P6000	2016
Volta	7.0	NVIDIA Titan V, Tesla V100	2017
Turing	7.5	Quadro RTX 8000, Tesla T4	2018
Ampere	8.0 - 8.6	GeForce RTX 3070-3090, NVIDIA A100	2020
Lovelace, Hopper, Blackwell	8.9	NVIDIA RTX 4090 (Video juegos) NVIDIA H100 (Centro de datos)	2022- 2023



### Compute Capability (CUDA SDK support vs. Microarchitecture)

CUDA SDK version(s)	Tesla	Fermi	Kepler (early)	Kepler (late)	Maxwell	Pascal	Volta	Turing	Ampere	Ada Lovelace	Hopper
1.0 <sup>[32]</sup>	1.0 – 1.1										
1.1	1.0 – 1.1+x										
2.0	1.0 – 1.1+x										
2.1 - 2.3.1 <sup>[33][34][35][36]</sup>	1.0 – 1.3										
3.0 - 3.1 <sup>[37][38]</sup>	1.0 –	2.0									
3.2 <sup>[39]</sup>	1.0 –	2.1									
4.0 - 4.2	1.0 –	2.1+x									
5.0 - 5.5	1.0 –			3.5							
6.0	1.0 –			3.5							
6.5	1.1 –				5.x						
7.0 - 7.5		2.0 –			5.x						
8.0		2.0 –				6.x					
9.0 - 9.2			3.0 –				7.0				
10.0 - 10.2			3.0 –					7.5			
11.0 <sup>[40]</sup>				3.5 –					8.0		
11.1 - 11.4 <sup>[41]</sup>				3.5 –					8.6		
11.5 - 11.7.1 <sup>[42]</sup>				3.5 –					8.7		
11.8 <sup>[43]</sup>				3.5 –							9.0
12.0 - 12.2					5.0 –						9.0

<https://en.wikipedia.org/wiki/CUDA>

CUDA\_Intro. Francisco J. Hernández-López

Ene-Jun 2024

Feature support (unlisted features are supported for all compute capabilities)	Compute capability (version)									
	1.0, 1.1	1.2, 1.3	2.x	3.0	3.2	3.5, 3.7, 5.x, 6.x, 7.0, 7.2	7.5	8.x	9.0	
Warp vote functions ( <code>__all()</code> , <code>__any()</code> )	No	Yes								
Warp vote functions ( <code>__ballot()</code> )	No		Yes							
Memory fence functions ( <code>__threadfence_system()</code> )										
Synchronization functions ( <code>__syncthreads_count()</code> , <code>__syncthreads_and()</code> , <code>__syncthreads_or()</code> )										
Surface functions										
3D grid of thread blocks	No		Yes							
Warp shuffle functions										
Unified memory programming										
Funnel shift	No			Yes						
Dynamic parallelism	No				Yes					
Uniform Datapath <sup>[50]</sup>	No						Yes			
Hardware-accelerated async-copy	No								Yes	
Hardware-accelerated <i>split arrive/wait barrier</i>										
Warp-level support for reduction ops										
L2 cache residency management										
DPX instructions for accelerated dynamic programming	No								Yes	
Distributed shared memory										
Thread block cluster										
Tensor memory accelerator (TMA) unit										
Feature support (unlisted features are supported for all compute capabilities)	1.0,1.1	1.2,1.3	2.x	3.0	3.2	3.5, 3.7, 5.x, 6.x, 7.0, 7.2	7.5	8.x	9.0	
	Compute capability (version)									

<https://en.wikipedia.org/wiki/CUDA>

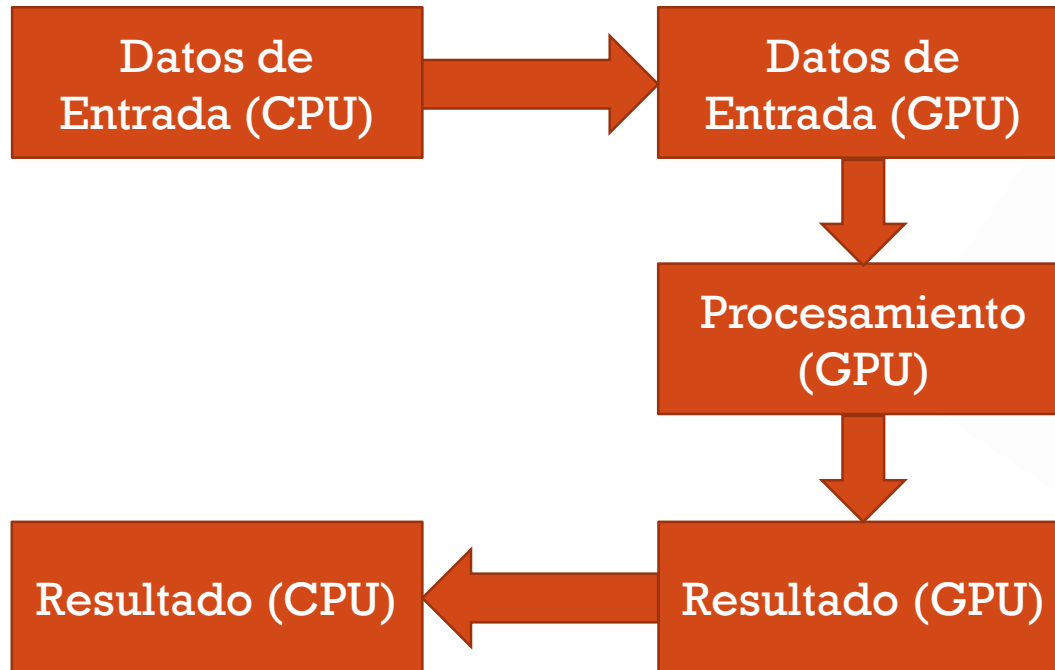
CUDA\_Intro. Francisco J. Hernández-López

Ene-Jun 2024

# MODELO TRADICIONAL DE PROGRAMACIÓN EN CUDA



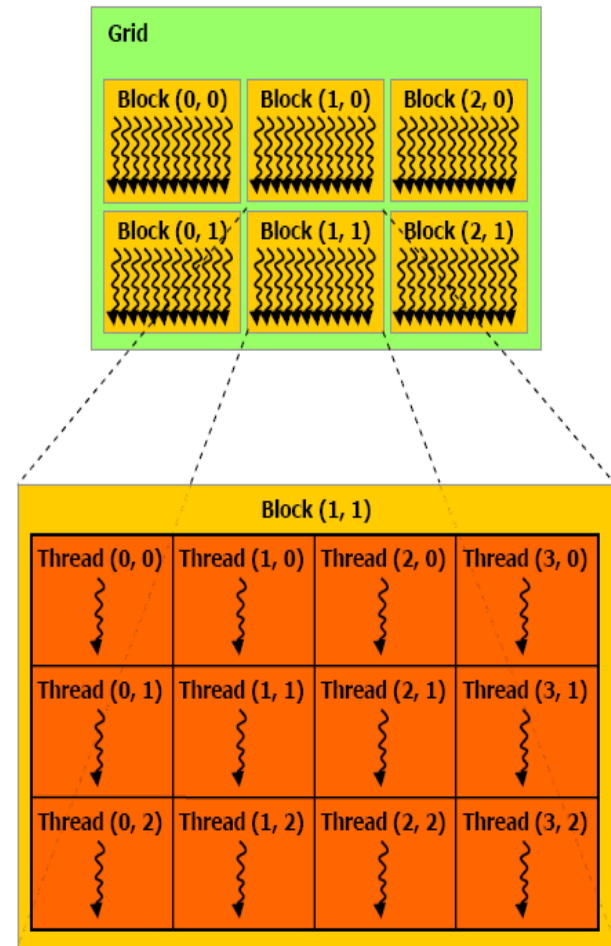
CPU



GPU

# CONFIGURACIÓN DE LOS HILOS

- Un programa que se compila para ejecutarse en una tarjeta gráfica se le llama *Kernel*.
- El conjunto de hilos que ejecuta un *Kernel* están organizados como una cuadrícula o malla (gríd) de bloques de hilos.
- Un Bloque de hilos es un conjunto de hilos que pueden cooperar juntos:
  - Con rápido acceso a memoria compartida.
  - De forma sincronizada.
  - Con un identificador de hilos ID.
  - Los Bloques pueden ser arreglos de 1, 2 o 3 dimensiones.
- Un Grid de bloques de hilos:
  - Tiene un número limitado de hilos en un bloque.
  - Los bloques se identifican mediante un ID.
  - Pueden ser arreglos de 1 o 2 dimensiones. Hasta 3 en GPUs con Capacidad  $\geq 2$



CUDA\_C\_Programming\_Guide.pdf

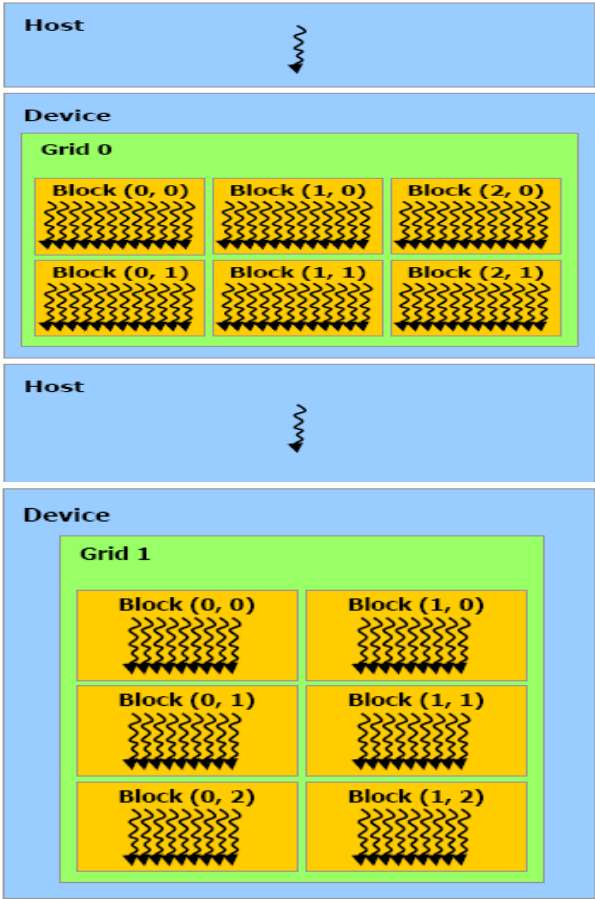
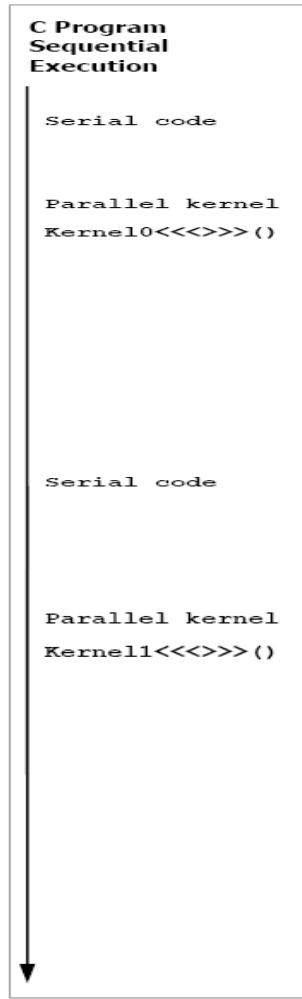


# TAMAÑOS DE LOS BLOQUES Y MALLAS

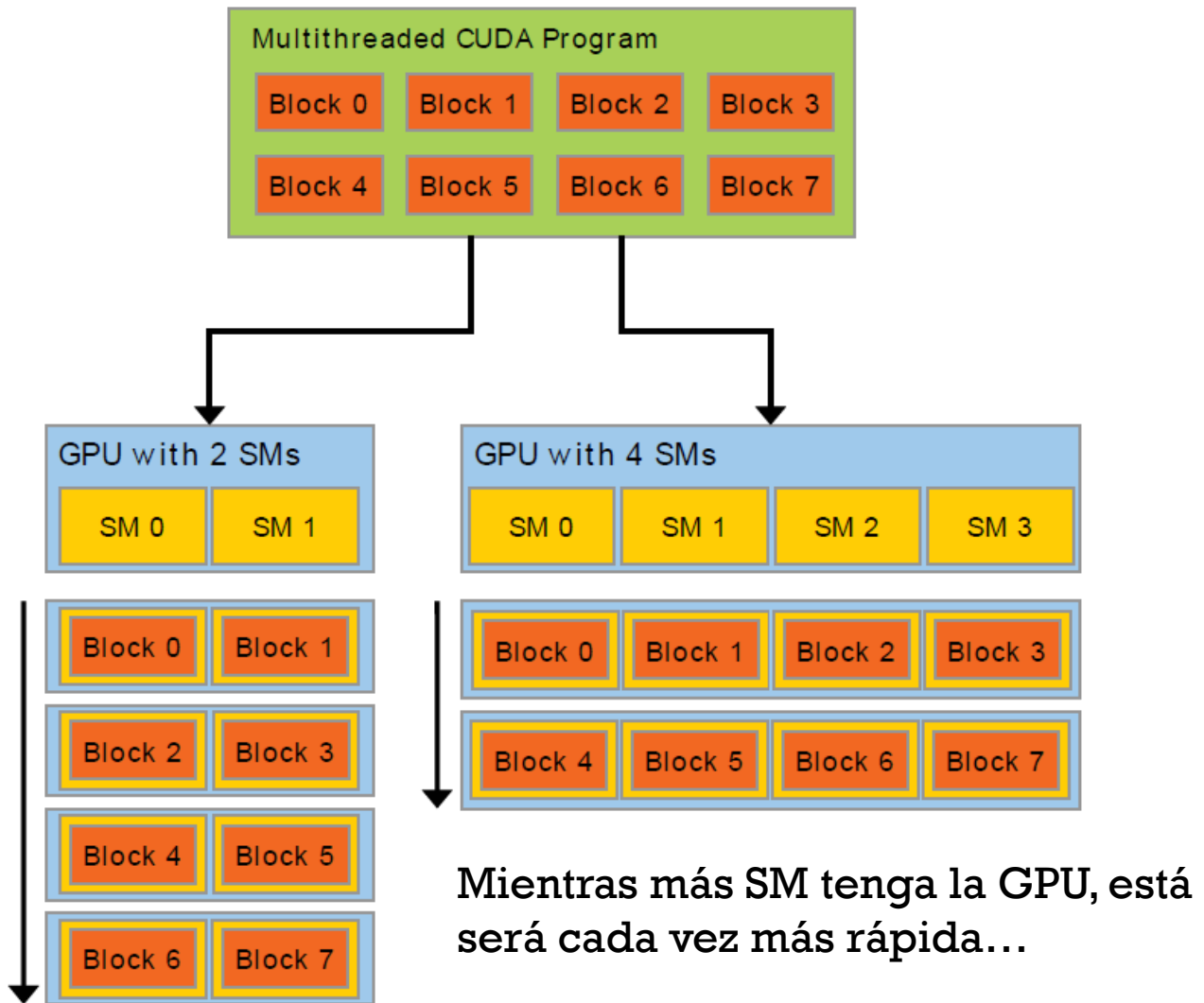
Item	Compute Capability			
	1.x	2.x	3.x	5.x
Max. number of grid dimensions	2	3		
Grid maximum x-dimension	$2^{16} - 1$		$2^{31} - 1$	
Grid maximum y/z-dimension	$2^{16} - 1$			
Max. number of block dimensions	3			
Block max. x/y-dimension	512	1024		
Block max. z-dimension	64			
Max. threads per block	512	1024		
GPU example (GTX family chips)	8800	480	780	980

# EJECUCIÓN DE UN PROGRAMA EN GPU

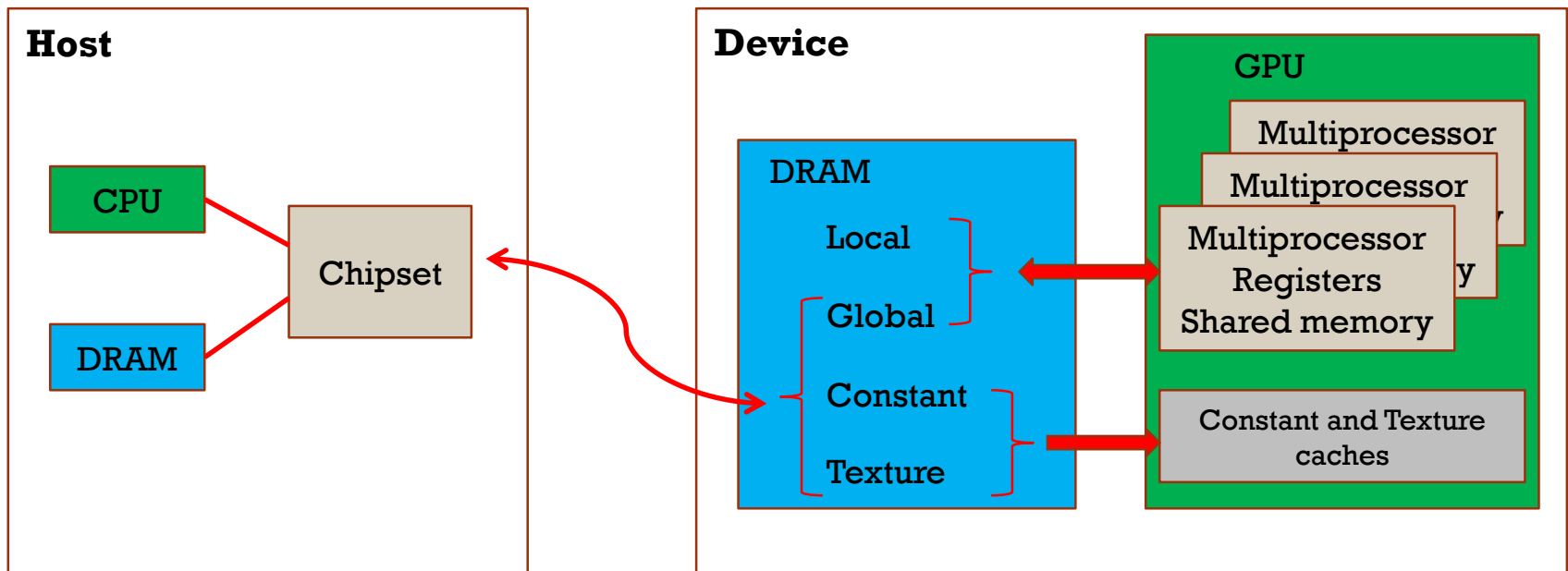
Host = CPU  
Device = GPU  
Kernel = Conjunto de instrucciones que se ejecutan en el device.



# ESCALABILIDAD AUTOMÁTICA



# MODELO DE LA MEMORIA EN CUDA





# INSTRUCCIONES PARA CREAR MEMORIA

- **cudaMalloc** ((void\*\*) devPtr, size\_t size)
- **cudaMallocHost** ((void\*\*) hostPtr, size\_t size)
- **cudaFree** (void \*devPtr)
- **cudaFreeHost** (void \*hostPtr)

# INSTRUCCIONES PARA COPIAR MEMORIA

- **cudaMemcpy** (void \*dst, const void \*src, size\_t count, enum cudaMemcpyKind kind)
- **cudaMemcpy2D** (void \*dst, size\_t dpitch, const void \*src, size\_t spitch, size\_t width, size\_t height, enum cudaMemcpyKind kind)
- **cudaMemcpyToSymbol** (const char \*symbol, const void \*src, size\_t count, size\_t offset, enum cudaMemcpyKind kind) H→D D→D
- **cudaMemcpyFromSymbol** (void \*dst, const char \*symbol, size\_t count, size\_t offset, enum cudaMemcpyKind kind) D→H D→D

## **Kind =**

cudaMemcpyHostToHost = 0, cudaMemcpyHostToDevice = 1, cudaMemcpyDeviceToHost = 2, cudaMemcpyDeviceToDevice = 3. cudaMemcpyDefault = 4 (Unified Virtual Address)

# CALIFICADORES DE UNA FUNCIÓN

## \_\_device\_\_

- Se ejecuta en el dispositivo
- Llamada solamente desde el dispositivo

## \_\_global\_\_

- Se ejecuta en el dispositivo
- Llamada solamente desde el host

## \_\_host\_\_

- Se ejecuta en el host
- Llamada solamente desde el host

# CALIFICADORES DE UNA VARIABLE

## device

- Reside en el espacio de la memoria global
- Tiene el tiempo de vida de una aplicación
- Es accesible a partir de todos los hilos dentro del grid, y a partir del host a través de la biblioteca en tiempo de ejecución

## constant (Opcionalmente se utiliza junto con device)

- Reside en el espacio de la memoria constante
- Tiene el tiempo de vida de una aplicación
- Es accesible a partir de todos los hilos dentro del grid, y a partir del host a través de la biblioteca en tiempo de ejecución

## shared (Opcionalmente se utiliza junto con device)

- Reside en el espacio de memoria compartida de un bloque de hilos
- Tiene el tiempo de vida de un bloque
- Solamente accesible a partir de los hilos que están dentro del bloque



# LLAMADA A UNA FUNCIÓN KERNEL

Una función, por ejemplo:

```
__global__ void NameFunc(float *parametro);
```

debe ser llamada como sigue:

```
NameFunc <<< Dg, Db, Ns, St >>> (parametro);
```

**Dg:** Es de tipo *dim3* dimensión y tamaño del grid

**Db:** Es de tipo *dim3* dimensión y tamaño de cada bloque

**Ns:** Es de tipo *size\_t* número de bytes en memoria compartida

**St:** Es de tipo *cudaStream\_t* el cuál indica que stream va a utilizar la función kernel

(Ns y St son argumentos opcionales)

# VARIABLES DEFINIDAS AUTOMÁTICAMENTE

Todas las funciones `__global__` y `__device__` tienen acceso a las siguientes variables:

- **gridDim** es de tipo `dim3`, indica la dimensión del grid
- **blockIdx** es de tipo `uint3`, indica el índice del bloque dentro del grid
- **blockDim** es de tipo `dim3`, indica la dimensión del bloque
- **threadIdx** es de tipo `uint3`, indica el índice del hilo dentro del bloque

# TIPOS DE DATOS

**char1, uchar1, char2, uchar2, char3, char3, char4, uchar4, short1, ushort1, short2, ushort2, short3, ushort3, short4, ushort4, int1, uint1, int2, uint2, int3, uint3, int4, int4, long1, ulong1, long2, ulong2, long3, ulong3, long4, ulong4, longlong1, longlong2, float1, float2, float3, float4, double1, double2**

La 1ra, 2da, 3ra, and 4ta componentes se acceden a través de los campos x, y, z y w respectivamente

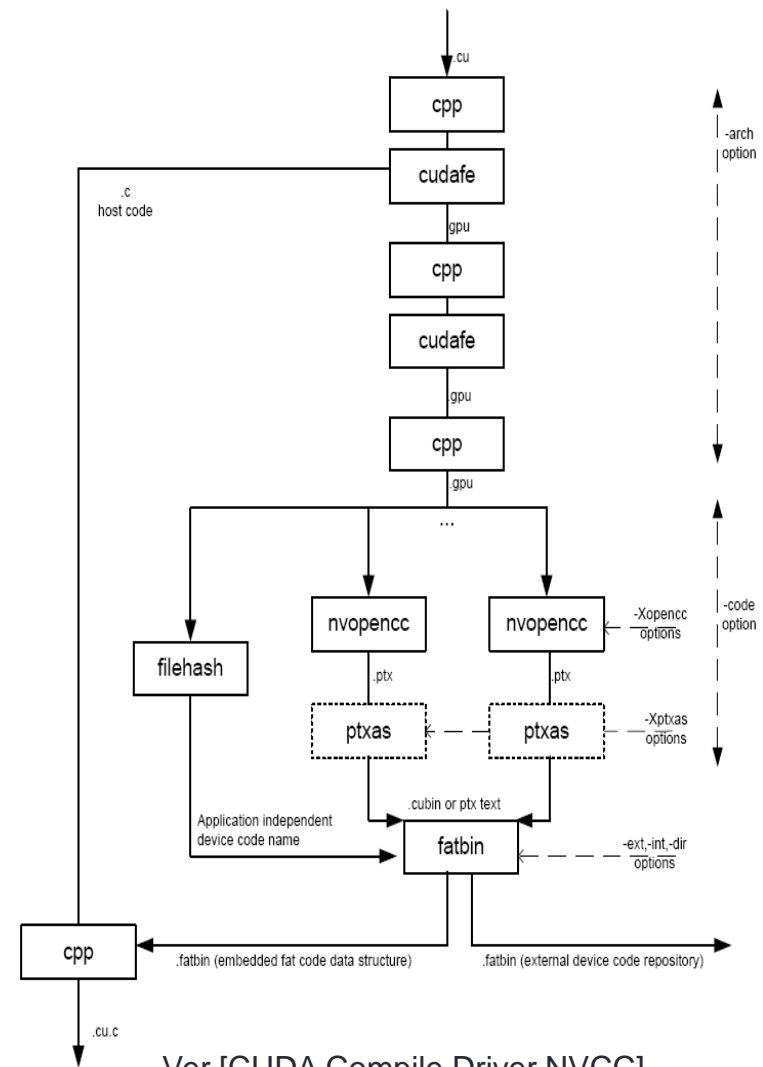
```
float3 temp[10];  
.....  
temp[i].x=0.0; temp[i].y=0.0; temp[i].z=0.0;
```

# FUNCIONES MATEMÁTICAS

- `__NombreFuncion()`
  - A nivel de hardware
  - Mayor velocidad pero menor precisión
  - Ejemplos: `__sinf(x)`, `__expf(x)`, `__logf(x)`,...
- `NombreFuncion()`
  - Menor velocidad pero mayor precisión
  - Ejemplos: `sinf(x)`, `expf(x)`, `logf(x)`,...
- `-use_fast_math`: Opción del compilador `nvcc`

# COMPILACIÓN CON NVCC

- El *nvcc*, es el encargado de compilar el código CUDA
- Soporta C/C++
- El *nvcc* utiliza los siguientes compiladores para el código *host*:
  - Linux: gcc, g++
  - Windows: Microsoft VS C/C++
  - Mac: Xcode



Ver [CUDA Compile Driver NVCC]

# INSTALANDO CUDA

<https://developer.nvidia.com/cuda-downloads>

## CUDA Toolkit 10.2 Download

### Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

#### Operating System

Windows

Linux

Mac OSX

## CUDA Toolkit 11.7 Downloads

[Home](#)

### Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown. By downloading and using the software, you agree to fully comply with the terms and conditions of the [CUDA EULA](#).

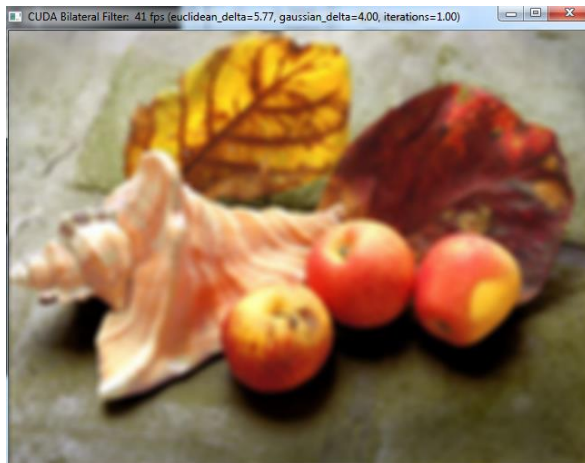
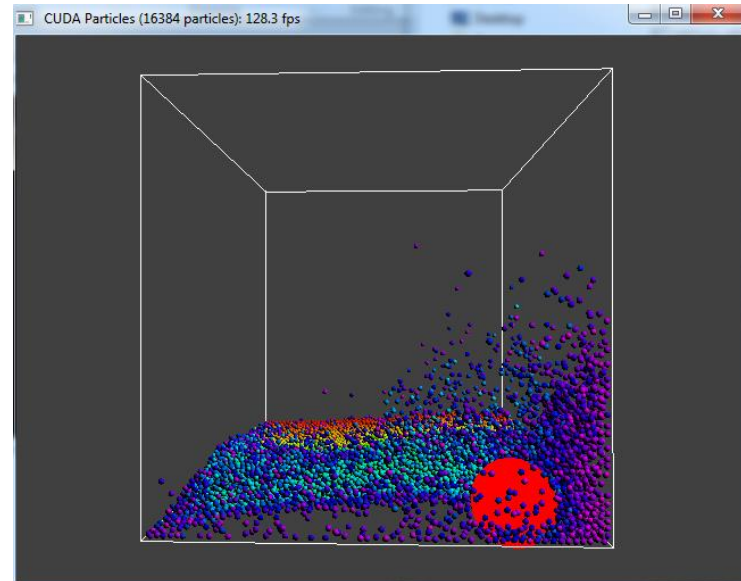
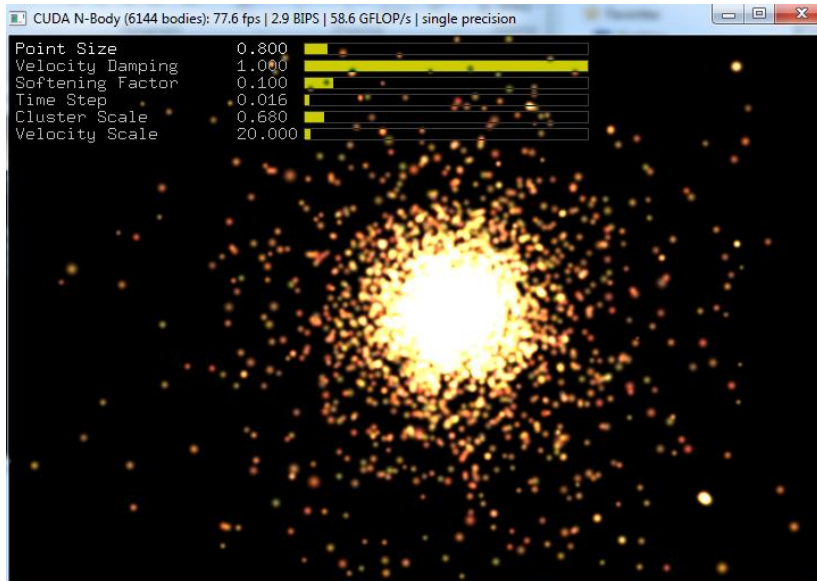
#### Operating System

Linux

Windows

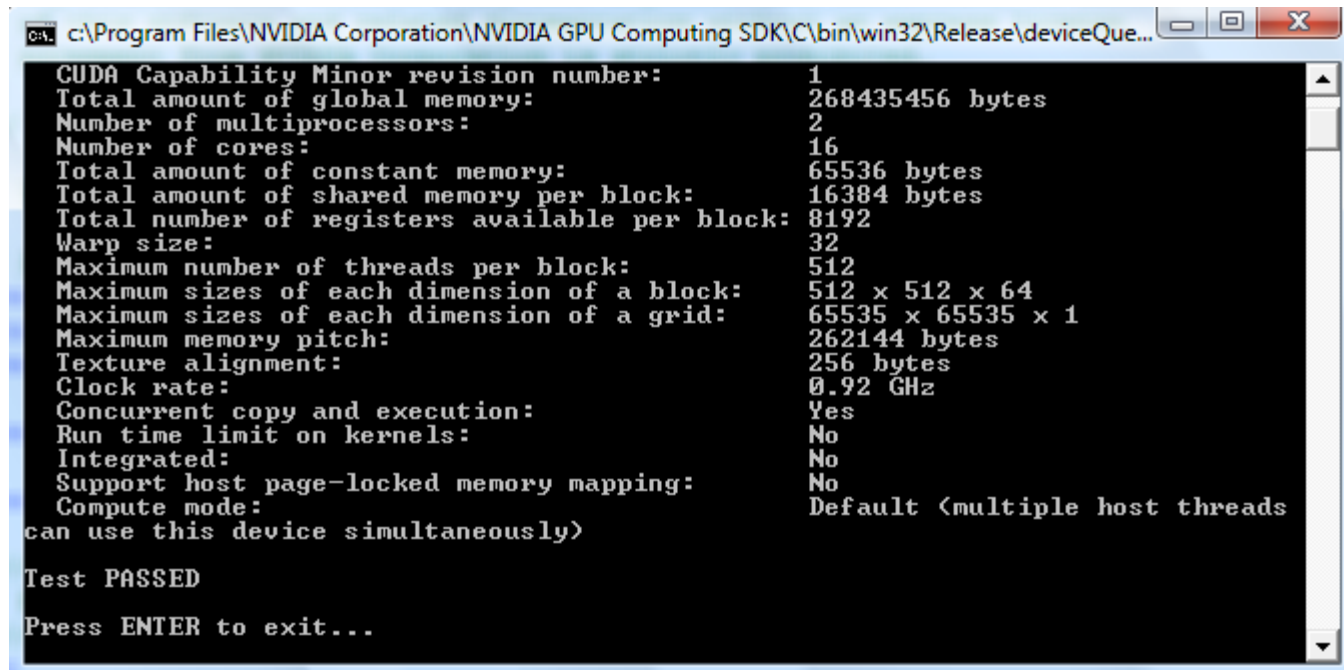


# EJEMPLOS DEL SDK



# “deviceQueryDrv”

- Para saber que capacidades tiene nuestra tarjeta de video:



```
c:\Program Files\NVIDIA Corporation\NVIDIA GPU Computing SDK\C\bin\win32\Release\deviceQue...
CUDA Capability Minor revision number: 1
Total amount of global memory: 268435456 bytes
Number of multiprocessors: 2
Number of cores: 16
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 16384 bytes
Total number of registers available per block: 8192
Warp size: 32
Maximum number of threads per block: 512
Maximum sizes of each dimension of a block: 512 x 512 x 64
Maximum sizes of each dimension of a grid: 65535 x 65535 x 1
Maximum memory pitch: 262144 bytes
Texture alignment: 256 bytes
Clock rate: 0.92 GHz
Concurrent copy and execution: Yes
Run time limit on kernels: No
Integrated: No
Support host page-locked memory mapping: No
Compute mode: Default <multiple host threads
can use this device simultaneously>

Test PASSED

Press ENTER to exit...
```

Resultado con una tarjeta NVIDIA GeForce 8400 GS

# “deviceQueryDrv”

```
c:\Program Files\NVIDIA Corporation\NVIDIA CUDA SDK\bin\win32\Debug\deviceQuery.exe
There is 1 device supporting CUDA
Device 0: "GeForce 8800 GT"
  Major revision number:          1
  Minor revision number:          1
  Total amount of global memory:  536543232 bytes
  Number of multiprocessors:      14
  Number of cores:                112
  Total amount of constant memory: 65536 bytes
  Total amount of shared memory per block: 16384 bytes
  Total number of registers available per block: 8192
  Warp size:                      32
  Maximum number of threads per block: 512
  Maximum sizes of each dimension of a block: 512 x 512 x 64
  Maximum sizes of each dimension of a grid: 65535 x 65535 x 1
  Maximum memory pitch:           262144 bytes
  Texture alignment:              256 bytes
  Clock rate:                     1.51 GHz
  Concurrent copy and execution:  Yes
Test PASSED
Press ENTER to exit...
```

NVIDIA GeForce 8800 GT

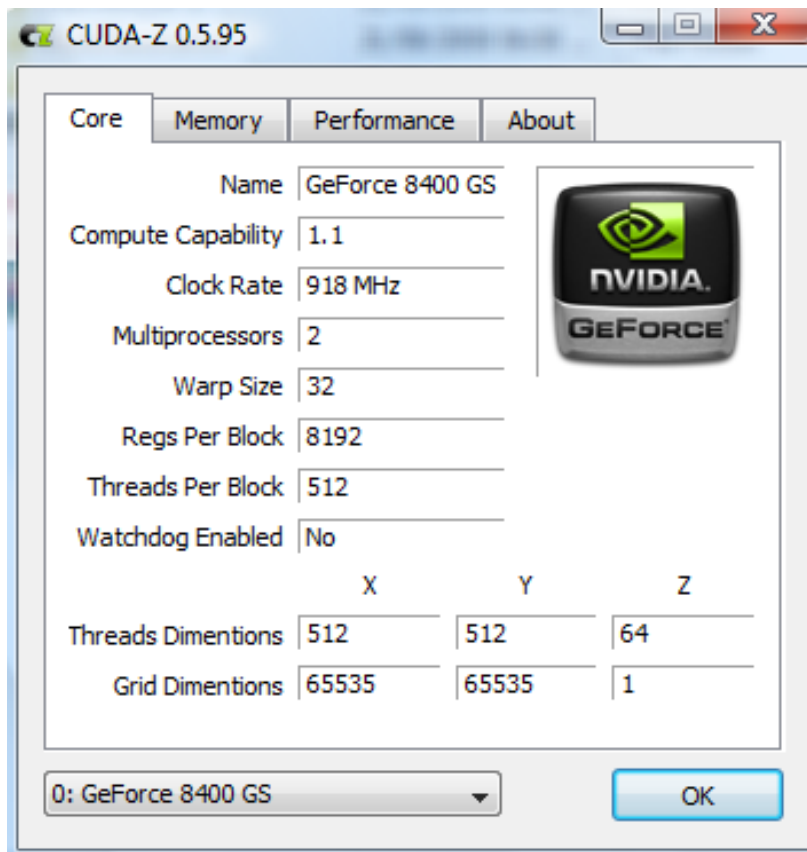
# “deviceQueryDrv”

```
c:\Program Files\NVIDIA Corporation\NVIDIA CUDA SDK\bin\win32\Debug\deviceQuery.exe
There is 1 device supporting CUDA
Device 0: "GeForce 8800 GT"
  Major revision number:          1
  Minor revision number:         1
  Total amount of global memory: 536543232 bytes
  Number of multiprocessors:     14
  Number of cores:               112
  Total amount of constant memory: 65536 bytes
  Total amount of shared memory per block: 16384 bytes
  Total number of registers available per block: 8192
  Warp size:                     32
  Maximum number of threads per block: 512
  Maximum sizes of each dimension of a block: 512 x 512 x 64
  Maximum sizes of each dimension of a grid: 65535 x 65535 x 1
  Maximum memory pitch:         262144 bytes
  Texture alignment:            256 bytes
  Clock rate:                   1.51 GHz
  Concurrent copy and execution: Yes
Test PASSED
Press ENTER to exit...
```

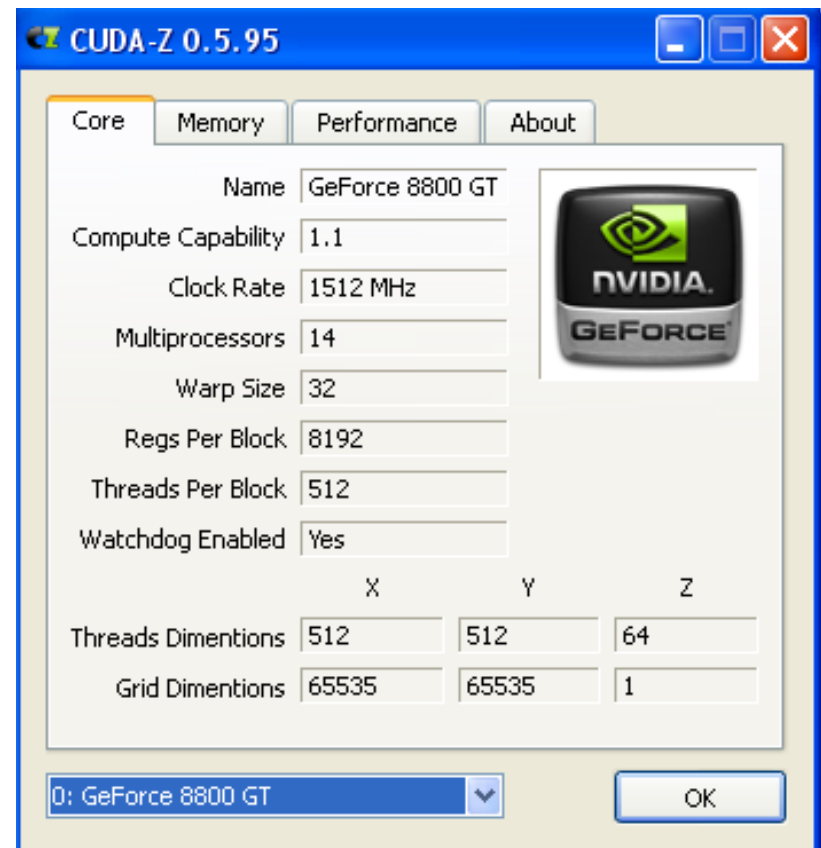
NVIDIA GeForce 8800 GT

# CUDA-Z

- GeForce 8400 GS & GeForce 8800 GT

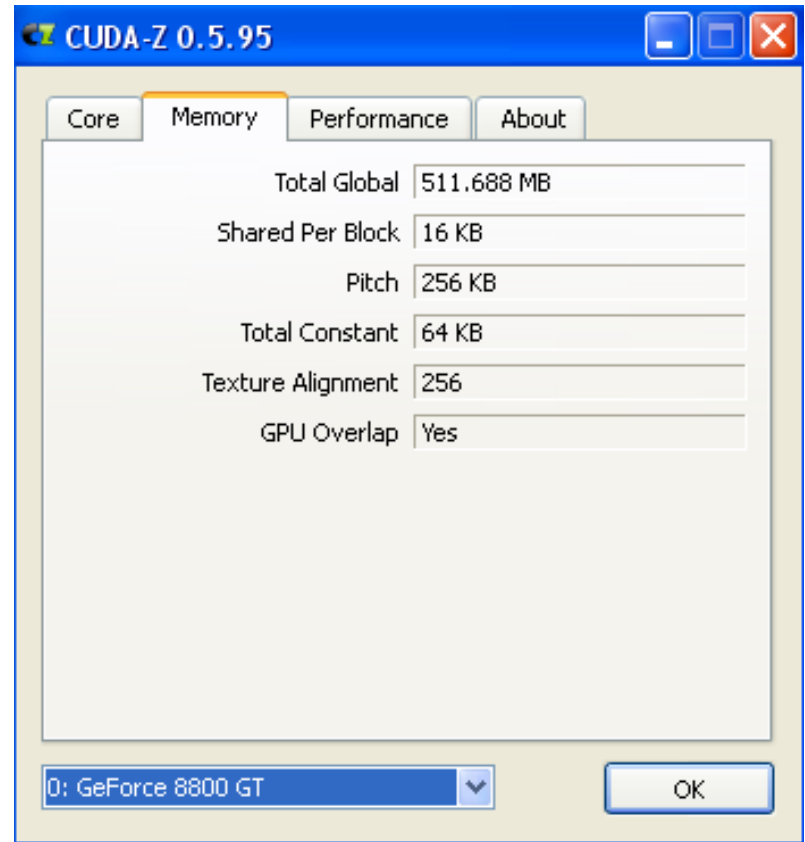
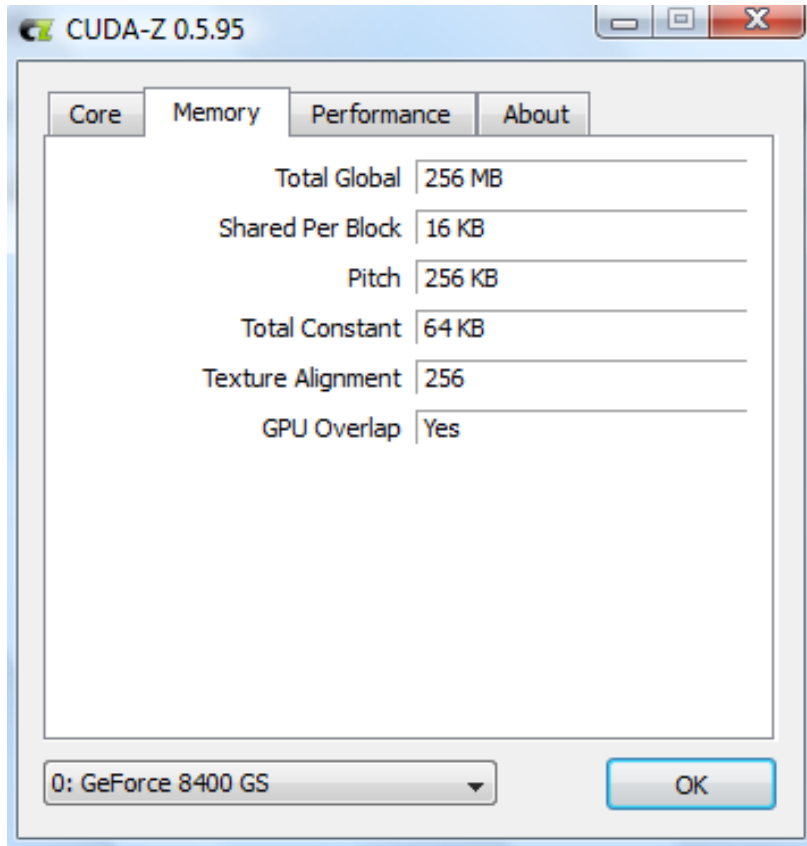


[<http://cuda-z.sourceforge.net/>]



# CUDA-Z

- GeForce 8400 GS & GeForce 8800 GT





# CUDA-Z

- GeForce 8400 GS & GeForce 8800 GT

The screenshot shows the Performance tab of the CUDA-Z 0.5.95 application for a GeForce 8400 GS. The interface includes tabs for Core, Memory, Performance, and About. The Performance section is divided into Memory Copy, GPU Core Performance, and a checkbox for 'Update Results in Background'. The Memory Copy section shows Host to Device, Device to Host, and Device to Device transfer rates. The GPU Core Performance section shows Single-precision Float, Double-precision Float (Not Supported), 32-bit Integer, and 24-bit Integer performance metrics. An 'Export >>' button is located at the bottom right of the main data area. A dropdown menu at the bottom left shows '0: GeForce 8400 GS' and an 'OK' button is at the bottom right.

Memory Copy	Pinned	Pageable
Host to Device	2442.7 MB/s	1818.96 MB/s
Device to Host	2425.35 MB/s	1848.57 MB/s
Device to Device	2075.73 MB/s	

GPU Core Performance

Single-precision Float	29266.9 Mflop/s
Double-precision Float	<i>Not Supported</i>
32-bit Integer	5860.85 Miop/s
24-bit Integer	29258.6 Miop/s

Update Results in Background      Export >> ▾

0: GeForce 8400 GS      OK

The screenshot shows the Performance tab of the CUDA-Z 0.5.95 application for a GeForce 8800 GT. The interface is similar to the GeForce 8400 GS screenshot, showing Memory Copy and GPU Core Performance metrics. The GPU Core Performance section shows significantly higher values for Single-precision Float, 32-bit Integer, and 24-bit Integer. The Double-precision Float remains 'Not Supported'. An 'Export >>' button is located at the bottom right of the main data area. A dropdown menu at the bottom left shows '0: GeForce 8800 GT' and an 'OK' button is at the bottom right.

Memory Copy	Pinned	Pageable
Host to Device	1918.1 MB/s	968.283 MB/s
Device to Host	1916.61 MB/s	978.734 MB/s
Device to Device	22583.3 MB/s	

GPU Core Performance

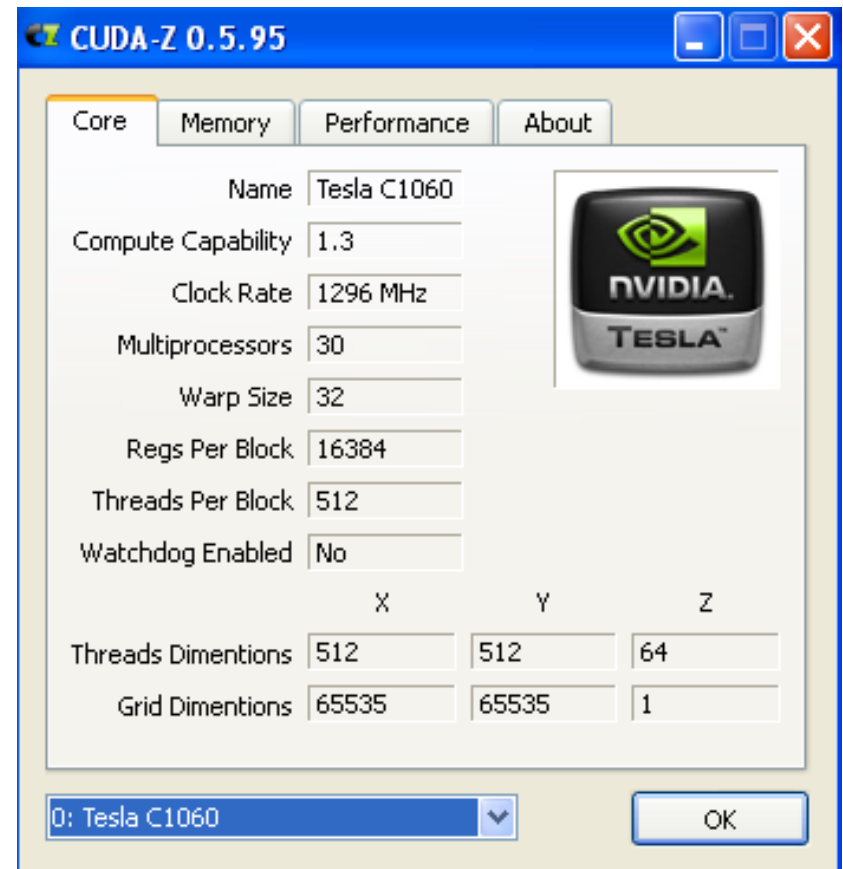
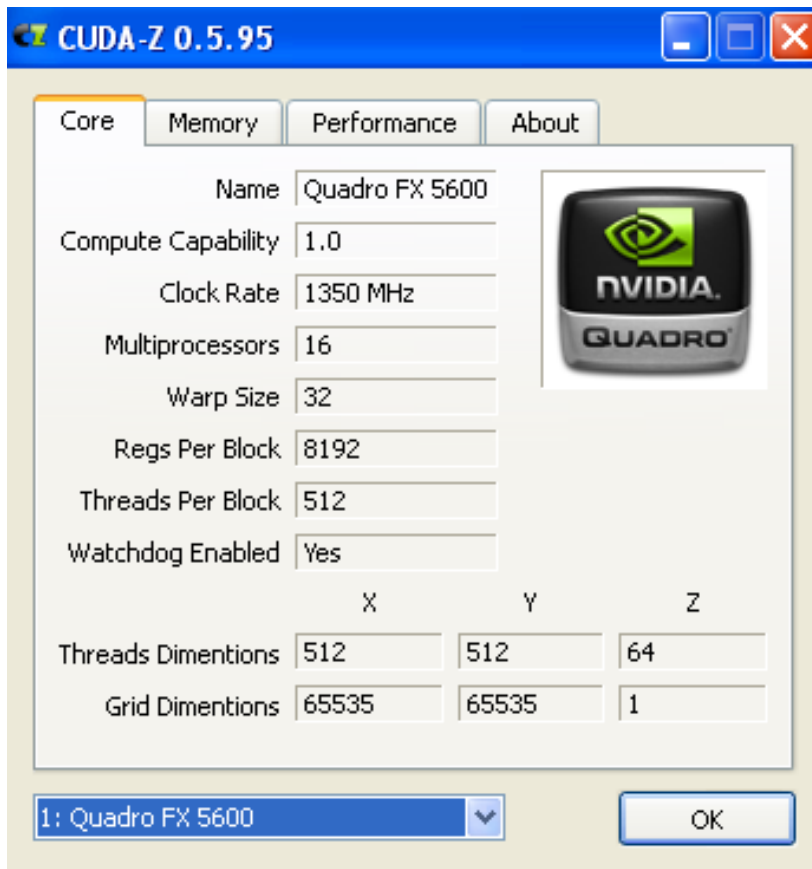
Single-precision Float	336450 Mflop/s
Double-precision Float	<i>Not Supported</i>
32-bit Integer	67353.1 Miop/s
24-bit Integer	336339 Miop/s

Update Results in Background      Export >> ▾

0: GeForce 8800 GT      OK

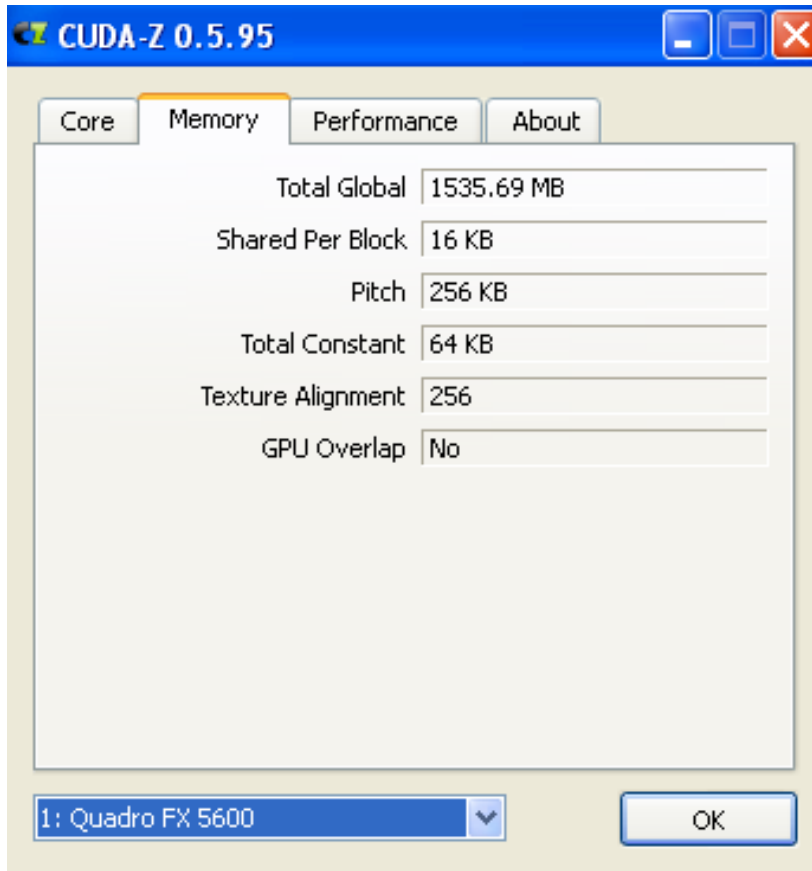
# CUDA-Z

- Quadro FX 5600 & Tesla C1060



# CUDA-Z

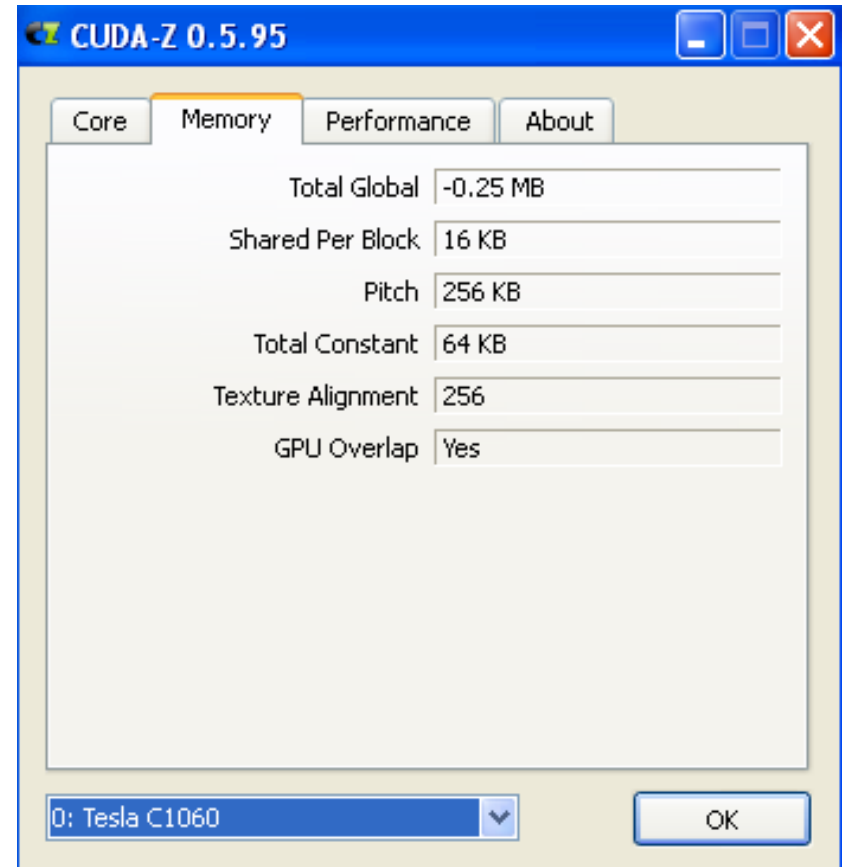
- Quadro FX 5600 & Tesla C1060



The screenshot shows the 'Memory' tab of the CUDA-Z 0.5.95 application for a Quadro FX 5600 GPU. The window title is 'CUDA-Z 0.5.95'. The 'Memory' tab is selected, and the 'About' tab is also visible. The memory configuration is as follows:

Total Global	1535.69 MB
Shared Per Block	16 KB
Pitch	256 KB
Total Constant	64 KB
Texture Alignment	256
GPU Overlap	No

At the bottom, a dropdown menu shows '1: Quadro FX 5600' and an 'OK' button is present.



The screenshot shows the 'Memory' tab of the CUDA-Z 0.5.95 application for a Tesla C1060 GPU. The window title is 'CUDA-Z 0.5.95'. The 'Memory' tab is selected, and the 'About' tab is also visible. The memory configuration is as follows:

Total Global	-0.25 MB
Shared Per Block	16 KB
Pitch	256 KB
Total Constant	64 KB
Texture Alignment	256
GPU Overlap	Yes

At the bottom, a dropdown menu shows '0: Tesla C1060' and an 'OK' button is present.

# CUDA-Z

- Quadro FX 5600 & Tesla C1060

Performance metrics for Quadro FX 5600:

Memory Copy	Pinned	Pageable
Host to Device	2909.03 MB/s	2800.47 MB/s
Device to Host	2909.57 MB/s	2800.51 MB/s
Device to Device	31037.3 MB/s	

GPU Core Performance:

Single-precision Float	343201 Mflop/s
Double-precision Float	<i>Not Supported</i>
32-bit Integer	68886.4 Miop/s
24-bit Integer	343180 Miop/s

Update Results in Background      Export >>

1: Quadro FX 5600      OK

Performance metrics for Tesla C1060:

Memory Copy	Pinned	Pageable
Host to Device	5684.15 MB/s	5218.91 MB/s
Device to Host	5684.56 MB/s	5220.9 MB/s
Device to Device	36259.5 MB/s	

GPU Core Performance:

Single-precision Float	618980 Mflop/s
Double-precision Float	75354.1 Mflop/s
32-bit Integer	124282 Miop/s
24-bit Integer	619094 Miop/s

Update Results in Background      Export >>

0: Tesla C1060      OK

# CUDA-Z

## •GeForce GTX 480 & GeForce GT 640

The screenshot shows the 'Core' tab of the CUDA-Z 0.6.163 application for a GeForce GTX 480. The window title is 'CUDA-Z 0.6.163'. The 'Core' tab is selected, and the 'Name' field is 'GeForce GTX 480'. The 'NVIDIA GeForce' logo is displayed on the right. The 'Grid Dimensions' are 65535 x 65535 x 65535. The 'Driver Version' is 301.42. The 'Driver Dll Version' is 4.20 (8.17.13.0142). The 'Runtime Dll Version' is 4.20 (6,14,11,4020). The 'Compute Mode' is 'Default'. The 'Watchdog Enabled' checkbox is checked. The 'Integrated GPU' checkbox is unchecked. The 'Concurrent Kernels' checkbox is checked. The 'Threads Per Block' is 1024. The 'Regs Per Block' is 32768. The 'Warp Size' is 32. The 'Therds Per Multiproc.' is 1536. The 'Multiprocessors' are 15 (480 Cores). The 'PCI Location' is 0:1:0. The 'Clock Rate' is 1451 MHz. The 'Compute Capability' is 2.0.

Property	Value
Name	GeForce GTX 480
Compute Capability	2.0
Clock Rate	1451 MHz
PCI Location	0:1:0
Multiprocessors	15 (480 Cores)
Therds Per Multiproc.	1536
Warp Size	32
Regs Per Block	32768
Threads Per Block	1024
Threads Dimensions	1024 x 1024 x 64
Grid Dimensions	65535 x 65535 x 65535
Driver Version	301.42
Driver Dll Version	4.20 (8.17.13.0142)
Runtime Dll Version	4.20 (6,14,11,4020)
Compute Mode	Default
Watchdog Enabled	Yes
Integrated GPU	No
Concurrent Kernels	Yes

The screenshot shows the 'Core' tab of the CUDA-Z 0.6.163 application for a GeForce GT 640. The window title is 'CUDA-Z 0.6.163'. The 'Core' tab is selected, and the 'Name' field is 'GeForce GT 640'. The 'NVIDIA GeForce' logo is displayed on the right. The 'Grid Dimensions' are 2147483647 x 65535 x 65535. The 'Driver Version' is 306.94. The 'Driver Dll Version' is 5.0 (8.17.13.0694). The 'Runtime Dll Version' is 4.20 (6,14,11,4020). The 'Compute Mode' is 'Default'. The 'Watchdog Enabled' checkbox is checked. The 'Integrated GPU' checkbox is unchecked. The 'Concurrent Kernels' checkbox is checked. The 'Threads Per Block' is 1024. The 'Regs Per Block' is 65536. The 'Warp Size' is 32. The 'Therds Per Multiproc.' is 2048. The 'Multiprocessors' are 2 (384 Cores). The 'PCI Location' is 0:2:0. The 'Clock Rate' is 954 MHz. The 'Compute Capability' is 3.0.

Property	Value
Name	GeForce GT 640
Compute Capability	3.0
Clock Rate	954 MHz
PCI Location	0:2:0
Multiprocessors	2 (384 Cores)
Therds Per Multiproc.	2048
Warp Size	32
Regs Per Block	65536
Threads Per Block	1024
Threads Dimensions	1024 x 1024 x 64
Grid Dimensions	2147483647 x 65535 x 65535
Driver Version	306.94
Driver Dll Version	5.0 (8.17.13.0694)
Runtime Dll Version	4.20 (6,14,11,4020)
Compute Mode	Default
Watchdog Enabled	Yes
Integrated GPU	No
Concurrent Kernels	Yes

# CUDA-Z

## •GeForce GTX 480 & GeForce GT 640

Core	Memory	Performance	About
Total Global	1535.69 MiB		
Bus Width	384 bits		
Clock Rate	1900 MHz		
Error Correction	No		
L2 Cache Size	48 KiB		
Shared Per Block	48 KiB		
Pitch	2048 MiB		
Total Constant	64 KiB		
Texture Alignment	512 B		
Texture 1D Size	65536		
Texture 2D Size	65536 x 65535		
Texture 3D Size	2048 x 2048 x 2048		
GPU Overlap	Yes		
Map Host Memory	Yes		
Unified Addressing	No		
Async Engine	Yes, Unidirectional		

0: GeForce GTX 480

OK

Core	Memory	Performance	About
Total Global	1024 MiB		
Bus Width	128 bits		
Clock Rate	2500 MHz		
Error Correction	No		
L2 Cache Size	48 KiB		
Shared Per Block	48 KiB		
Pitch	2048 MiB		
Total Constant	64 KiB		
Texture Alignment	512 B		
Texture 1D Size	65536		
Texture 2D Size	65536 x 65536		
Texture 3D Size	4096 x 4096 x 4096		
GPU Overlap	Yes		
Map Host Memory	Yes		
Unified Addressing	No		
Async Engine	Yes, Unidirectional		

1: GeForce GT 640

OK



# CUDA-Z

## •GeForce GTX 480 & GeForce GT 640

The screenshot shows the Performance tab for a GeForce GTX 480. The window title is 'CUDA-Z 0.6.163'. The Performance tab is selected, showing memory copy and GPU core performance metrics. The device is identified as '0: GeForce GTX 480' in the dropdown at the bottom.

Memory Copy	Pinned	Pageable
Host to Device	6180.17 MiB/s	3018.5 MiB/s
Device to Host	6212.32 MiB/s	4220.21 MiB/s
Device to Device	73.6236 GiB/s	

GPU Core Performance

Single-precision Float	1386.26 Gflop/s
Double-precision Float	174.092 Gflop/s
32-bit Integer	695.33 Giop/s
24-bit Integer	694.529 Giop/s

Update Results in Background  
 Heavy Load Test Mode

Export >> ▾

0: GeForce GTX 480

OK

The screenshot shows the Performance tab for a GeForce GT 640. The window title is 'CUDA-Z 0.6.163'. The Performance tab is selected, showing memory copy and GPU core performance metrics. The device is identified as '1: GeForce GT 640' in the dropdown at the bottom.

Memory Copy	Pinned	Pageable
Host to Device	5798.5 MiB/s	1765.2 MiB/s
Device to Host	6254.84 MiB/s	1804.55 MiB/s
Device to Device	26.0472 GiB/s	

GPU Core Performance

Single-precision Float	425.788 Gflop/s
Double-precision Float	30.4833 Gflop/s
32-bit Integer	121.56 Giop/s
24-bit Integer	121.296 Giop/s

Update Results in Background  
 Heavy Load Test Mode

Export >> ▾

1: GeForce GT 640

OK

# CUDA-Z

## •GeForce GTX 980M

**Core** Memory Performance About

Name GeForce GTX 980M

Compute Capability 5.2

Clock Rate 1126.5 MHz

PCI Location 0:1:0

Multiprocessors 12 (1536 Cores)

Threads Per Multiproc. 2048

Warp Size 32

Regs Per Block 65536

Threads Per Block 1024

Threads Dimensions 1024 x 1024 x 64

Grid Dimensions 2147483647 x 65535 x 65535

Watchdog Enabled Yes

Integrated GPU No

Concurrent Kernels Yes

Compute Mode Default

Stream Priorities Yes

0: GeForce GTX 980M

**Core** Memory Performance About

Memory Copy Pinned Pageable

Host to Device 5772.19 MiB/s 3381.27 MiB/s

Device to Host 6109.27 MiB/s 3532.94 MiB/s

Device to Device 62.537 GiB/s

GPU Core Performance

Single-precision Float 3285.41 Gflop/s

Double-precision Float 104.779 Gflop/s

64-bit Integer 180.854 Giop/s

32-bit Integer 967.637 Giop/s

24-bit Integer 746.964 Giop/s

Update Results in Background

Heavy Load Test Mode

Export >>

0: GeForce GTX 980M

**Core** Memory Performance About

Total Global 4096 MiB

Bus Width 256 bits

Clock Rate 2505 MHz

Error Correction No

L2 Cache Size 48 KiB

Shared Per Block 48 KiB

Pitch 2048 MiB

Total Constant 64 KiB

Texture Alignment 512.B

Texture 1D Size 65536

Texture 2D Size 65536 x 65536

Texture 3D Size 4096 x 4096 x 4096

GPU Overlap Yes

Map Host Memory Yes

Unified Addressing Yes

Async Engine No

M

OK

# “deviceQueryDrv”

```
Device 0: "Tesla K40c"
  CUDA Driver Version / Runtime Version      11.2 / 11.2
  CUDA Capability Major/Minor version number: 3.5
  Total amount of global memory:             11441 MBytes (11996954624 bytes)
  (15) Multiprocessors, (192) CUDA Cores/MP: 2880 CUDA Cores
  GPU Max Clock rate:                       745 MHz (0.75 GHz)
  Memory Clock rate:                        3004 Mhz
  Memory Bus Width:                         384-bit
  L2 Cache Size:                            1572864 bytes
  Maximum Texture Dimension Size (x,y,z)    1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers 1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(16384, 16384), 2048 layers
  Total amount of constant memory:          65536 bytes
  Total amount of shared memory per block:   49152 bytes
  Total shared memory per multiprocessor:    49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                32
  Maximum number of threads per multiprocessor: 2048
  Maximum number of threads per block:      1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                    2147483647 bytes
  Texture alignment:                       512 bytes
  Concurrent copy and kernel execution:     Yes with 2 copy engine(s)
  Run time limit on kernels:                No
  Integrated GPU sharing Host Memory:       No
  Support host page-locked memory mapping:  Yes
  Alignment requirement for Surfaces:       Yes
  Device has ECC support:                   Enabled
  Device supports Unified Addressing (UVA): Yes
  Device supports Managed Memory:          Yes
  Device supports Compute Preemption:      No
  Supports Cooperative Kernel Launch:      No
  Supports MultiDevice Co-op Kernel Launch: No
  Device PCI Domain ID / Bus ID / location ID: 0 / 8 / 0
  Compute Mode:
    < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >
```

# “deviceQueryDrv”

```
Device 0: "TITAN RTX"
  CUDA Driver Version / Runtime Version      11.0 / 11.0
  CUDA Capability Major/Minor version number: 7.5
  Total amount of global memory:            24220 MBytes (25396838400 bytes)
  (72) Multiprocessors, ( 64) CUDA Cores/MP: 4608 CUDA Cores
  GPU Max Clock rate:                       1770 MHz (1.77 GHz)
  Memory Clock rate:                        7001 Mhz
  Memory Bus Width:                         384-bit
  L2 Cache Size:                             6291456 bytes
  Maximum Texture Dimension Size (x,y,z)    1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers
  Total amount of constant memory:          65536 bytes
  Total amount of shared memory per block:   49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                 32
  Maximum number of threads per multiprocessor: 1024
  Maximum number of threads per block:      1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                     2147483647 bytes
  Texture alignment:                         512 bytes
  Concurrent copy and kernel execution:     Yes with 3 copy engine(s)
  Run time limit on kernels:                 No
  Integrated GPU sharing Host Memory:        No
  Support host page-locked memory mapping:  Yes
  Alignment requirement for Surfaces:       Yes
  Device has ECC support:                    Disabled
  Device supports Unified Addressing (UVA):  Yes
  Device supports Managed Memory:           Yes
  Device supports Compute Preemption:       Yes
  Supports Cooperative Kernel Launch:       Yes
  Supports MultiDevice Co-op Kernel Launch: Yes
  Device PCI Domain ID / Bus ID / location ID: 0 / 175 / 0
  Compute Mode:
    < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >
```

# “deviceQueryDrv”

```
Device 1: "Quadro RTX 8000"
  CUDA Driver Version / Runtime Version      11.0 / 11.0
  CUDA Capability Major/Minor version number: 7.5
  Total amount of global memory:             48601 MBytes (50962169856 bytes)
  (72) Multiprocessors, ( 64) CUDA Cores/MP: 4608 CUDA Cores
  GPU Max Clock rate:                       1770 MHz (1.77 GHz)
  Memory Clock rate:                        7001 Mhz
  Memory Bus Width:                         384-bit
  L2 Cache Size:                            6291456 bytes
  Maximum Texture Dimension Size (x,y,z)    1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers
  Total amount of constant memory:          65536 bytes
  Total amount of shared memory per block:   49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                32
  Maximum number of threads per multiprocessor: 1024
  Maximum number of threads per block:      1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                     2147483647 bytes
  Texture alignment:                        512 bytes
  Concurrent copy and kernel execution:     Yes with 3 copy engine(s)
  Run time limit on kernels:                No
  Integrated GPU sharing Host Memory:       No
  Support host page-locked memory mapping:  Yes
  Alignment requirement for Surfaces:       Yes
  Device has ECC support:                   Disabled
  Device supports Unified Addressing (UVA):  Yes
  Device supports Managed Memory:          Yes
  Device supports Compute Preemption:      Yes
  Supports Cooperative Kernel Launch:      Yes
  Supports MultiDevice Co-op Kernel Launch: Yes
  Device PCI Domain ID / Bus ID / location ID: 0 / 216 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >
```

# “deviceQueryDrv”

```
Command Prompt
CUDA Device Query (Runtime API) version (CUDA static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Quadro T1000"
  CUDA Driver Version / Runtime Version      11.4 / 11.2
  CUDA Capability Major/Minor version number: 7.5
  Total amount of global memory:             4096 MBytes (4294967296 bytes)
  (14) Multiprocessors, ( 64) CUDA Cores/MP: 896 CUDA Cores
  GPU Max Clock rate:                       1455 MHz (1.46 GHz)
  Memory Clock rate:                        4001 Mhz
  Memory Bus Width:                         128-bit
  L2 Cache Size:                            1048576 bytes
  Maximum Texture Dimension Size (x,y,z)    1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers
  Total amount of constant memory:          65536 bytes
  Total amount of shared memory per block:   49152 bytes
  Total shared memory per multiprocessor:    65536 bytes
  Total number of registers available per block: 65536
  Warp size:                                32
  Maximum number of threads per multiprocessor: 1024
  Maximum number of threads per block:      1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                     2147483647 bytes
  Texture alignment:                        512 bytes
  Concurrent copy and kernel execution:     Yes with 6 copy engine(s)
  Run time limit on kernels:                 Yes
  Integrated GPU sharing Host Memory:        No
  Support host page-locked memory mapping:   Yes
  Alignment requirement for Surfaces:        Yes
  Device has ECC support:                    Disabled
  CUDA Device Driver Mode (TCC or WDDM):     WDDM (Windows Display Driver Model)
  Device supports Unified Addressing (UVA):   Yes
  Device supports Managed Memory:            Yes
  Device supports Compute Preemption:        Yes
  Supports Cooperative Kernel Launch:        Yes
  Supports MultiDevice Co-op Kernel Launch: No
  Device PCI Domain ID / Bus ID / location ID: 0 / 1 / 0
  Compute Mode:
    < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 11.4, CUDA Runtime Version = 11.2, NumDevs = 1
```

# GRACIAS POR SU ATENCIÓN

Francisco J. Hernández-López

[fcoj23@ciimat.mx](mailto:fcoj23@ciimat.mx)

WebPage:

[www.ciimat.mx/~fcoj23](http://www.ciimat.mx/~fcoj23)

