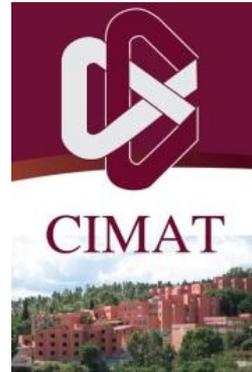


# INTRODUCCIÓN A LA PROGRAMACIÓN EN CUDA

---



Francisco Javier Hernández López

<http://www.cimat.mx/~fcoj23>

Guanajuato, Gto. Noviembre de 2012

# ¿Qué es el Cómputo Paralelo (CP)?

- Ejecución de más de un cómputo (cálculo) al mismo tiempo o “en paralelo”, utilizando más de un procesador.
- Arquitecturas que hay en la actualidad para el cómputo paralelo:



Computadora con múltiples procesadores (Cores)



Cluster



Tarjetas Graficas (GPUs)

# Tipos de CP

- **A nivel de bits**

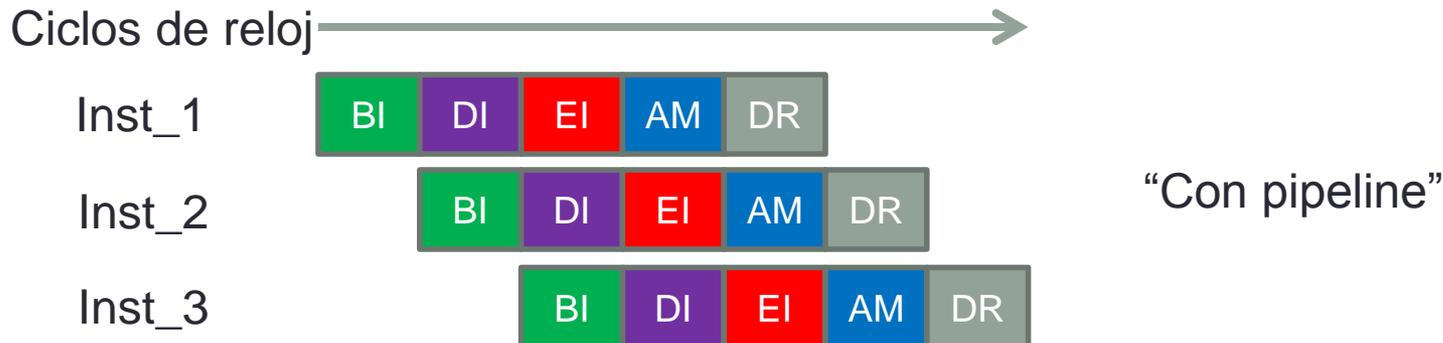
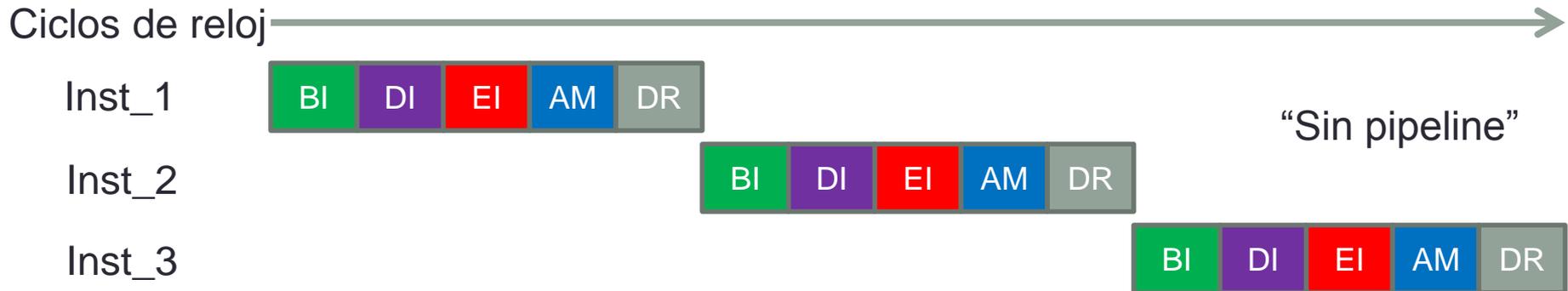
- Basado en incrementar el número de bits de una palabra (*word length, word size o word width*)
- Importante característica de los procesadores (8,16, 32 o 64 bits)
- Ejemplo:
  - Tenemos un procesador de 16 bits y queremos sumar dos enteros (int) de 32 bits
  - Primero el procesador realiza la suma de los primeros 16 bits y después de los últimos 16 bits, completando la suma de los int en dos instrucciones
  - Un procesador mayor o igual a 32 bits realizaría la operación en una sola instrucción.

# Tipos de CP

- **A nivel de instrucción**

- Capacidad de traslapar instrucciones
- Depende del pipeline del procesador

BI → Buscar Instrucción  
DI → Descifrar Instrucción  
EI → Ejecutar Instrucción  
AM → Acceso a Memoria  
DR → Dar Respuesta



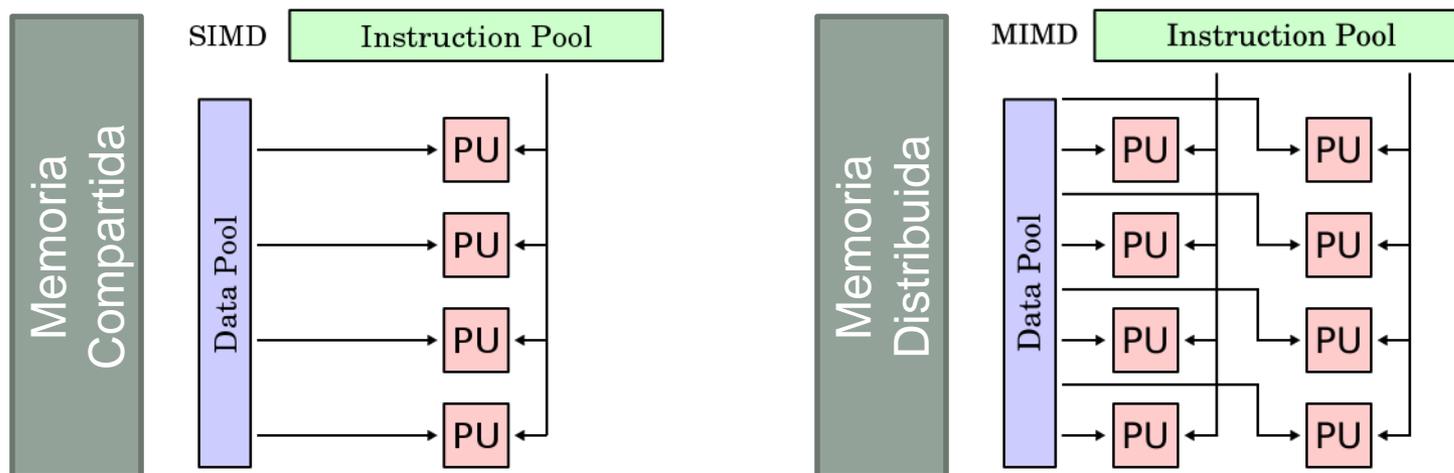
# Tipos de CP

- **A nivel de datos**

- Cada procesador realiza la misma tarea en diferentes partes de los datos (SIMD → Single Instruction Multiple Data)

- **A nivel de tareas**

- Diferentes procesadores pueden estar ejecutando diferentes instrucciones en diferentes partes de los datos (MIMD → Multiple Instruction Multiple Data)



Comparación entre SIMD y MIMD, [wikipedia]

# Lenguajes de programación para el CP



**OpenMP**  
(Memoria Compartida)



**OpenMPI**  
(Memoria Distribuida)



**CUDA** (*Compute Unified Device Architecture*) para GPUs.

**OpenCL** (*Open Computing Language*) para GPUs y CPUs.

## Mezcla de Arquitecturas:

Computadora (o múltiples cores) y un GPU (o un arreglo de GPUs) .

Cluster con computadoras que contienen múltiples cores y a su vez cada maquina tiene un GPU (o un arreglo de GPUs).

Así como mezclamos las arquitecturas también mezclamos los lenguajes:

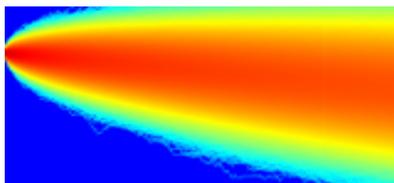
- CUDA con (OpenMP, OpenMPI).
- OpenCL con (OpenMP, OpenMPI).
- PGI Fortran
- OpenACC

# Tarjetas de Video (GPUs)

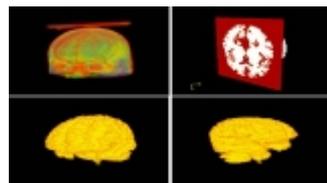


# GPUs

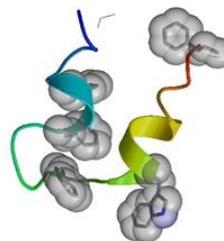
- Procesadores flexibles de procesamiento general
- Se pueden resolver problemas de diversas áreas:
  - Finanzas, Gráficos, Procesamiento de Imágenes y Video, Álgebra Lineal, Física, Química, Biología, etc.



Ecs. Diferenciales



Segmentación de  
Imágenes Medicas



Dinámica Molecular

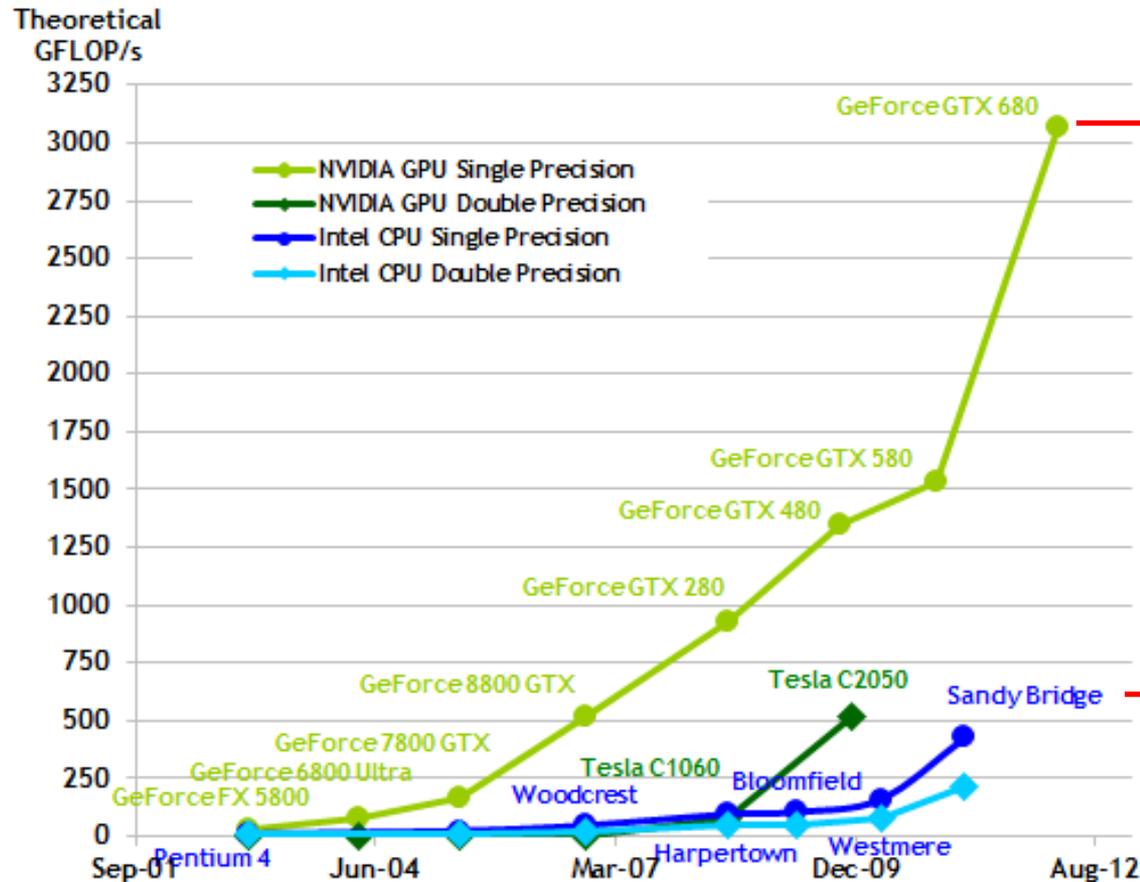


Detección de Objetos

Visitar [CUDA ZONE](#)

# GPUs

- GPUs vs CPUs

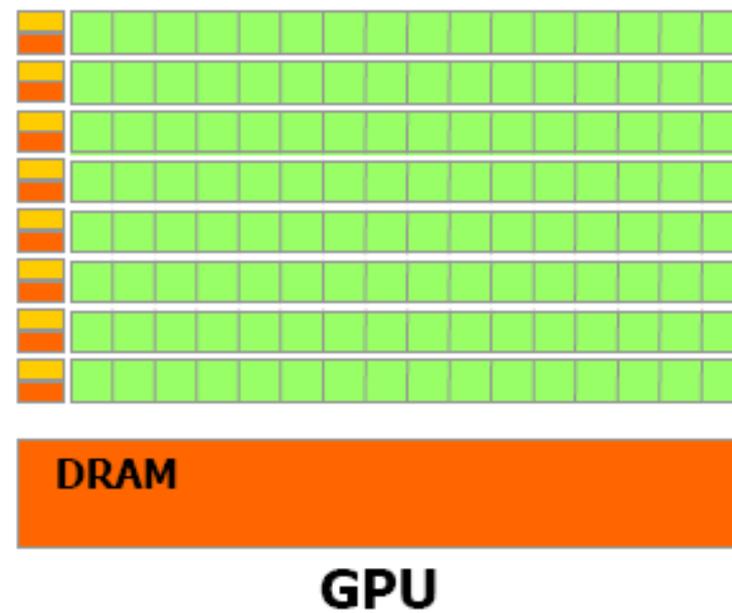
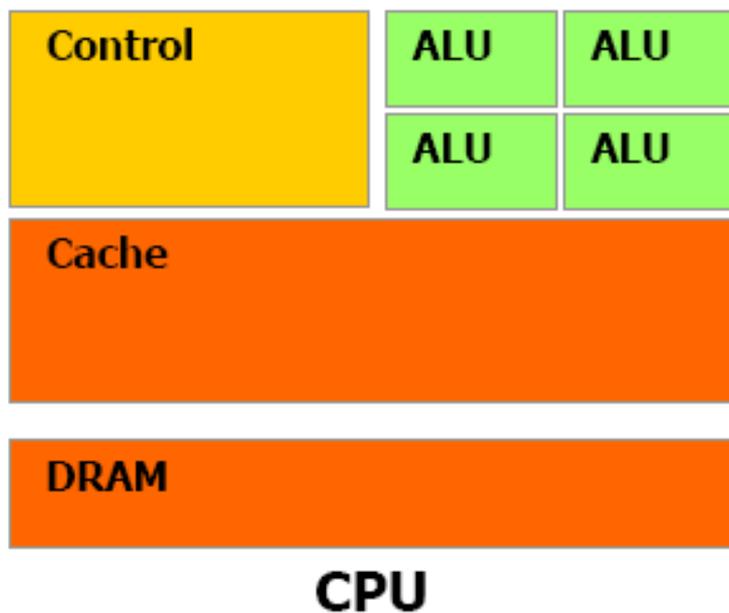


Ver [NVIDIA CUDA C Programming Guide]



# GPUs

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



# CUDA

- Es una tecnología de propósito general que nos permite ejecutar código en GPUs para hacer Cómputo Paralelo
- Desarrollado por NVIDIA en el 2006
- Para utilizar esta arquitectura se requiere tener una tarjeta gráfica GeForce 8 series (o equivalente Quadro)

# Software usando CUDA

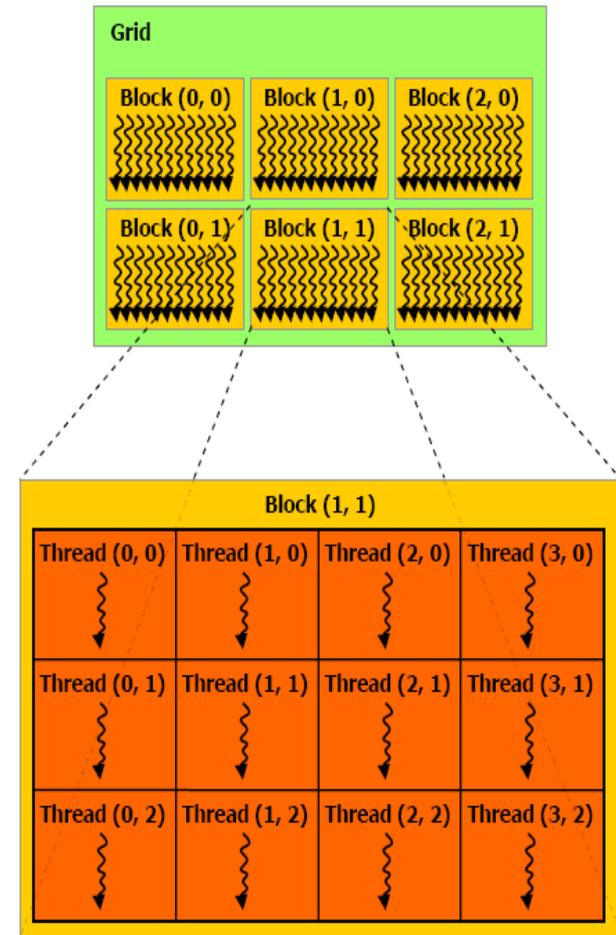


# Características de CUDA

- 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

# Modelo de Programación en CUDA

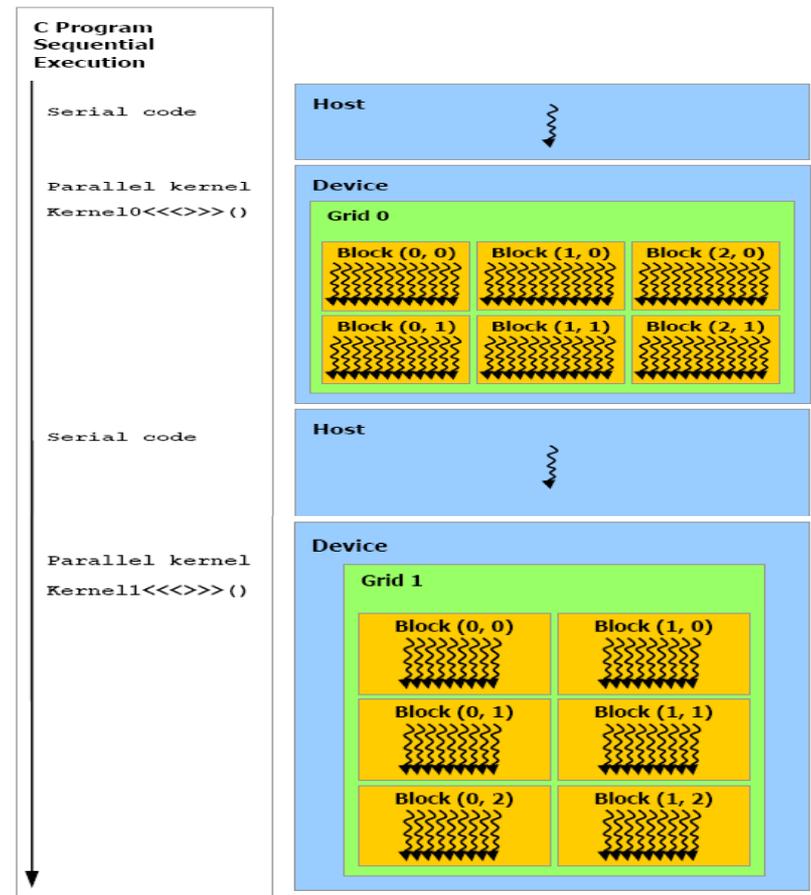
- Un programa que se ejecuta en una tarjeta gráfica se le llama *Kernel*
- El *Kernel* está organizado como una cuadrícula (grid) de bloques de hilos
- Un Bloque de hilos es un conjunto de hilos que pueden cooperar juntos:
  - Con rápido acceso a memoria compartida
  - De forma sincronizada
  - Con un 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 (Capacidad <2.0)
    - 1, 2 o 3 dimensiones (Capacidad >=2.0)



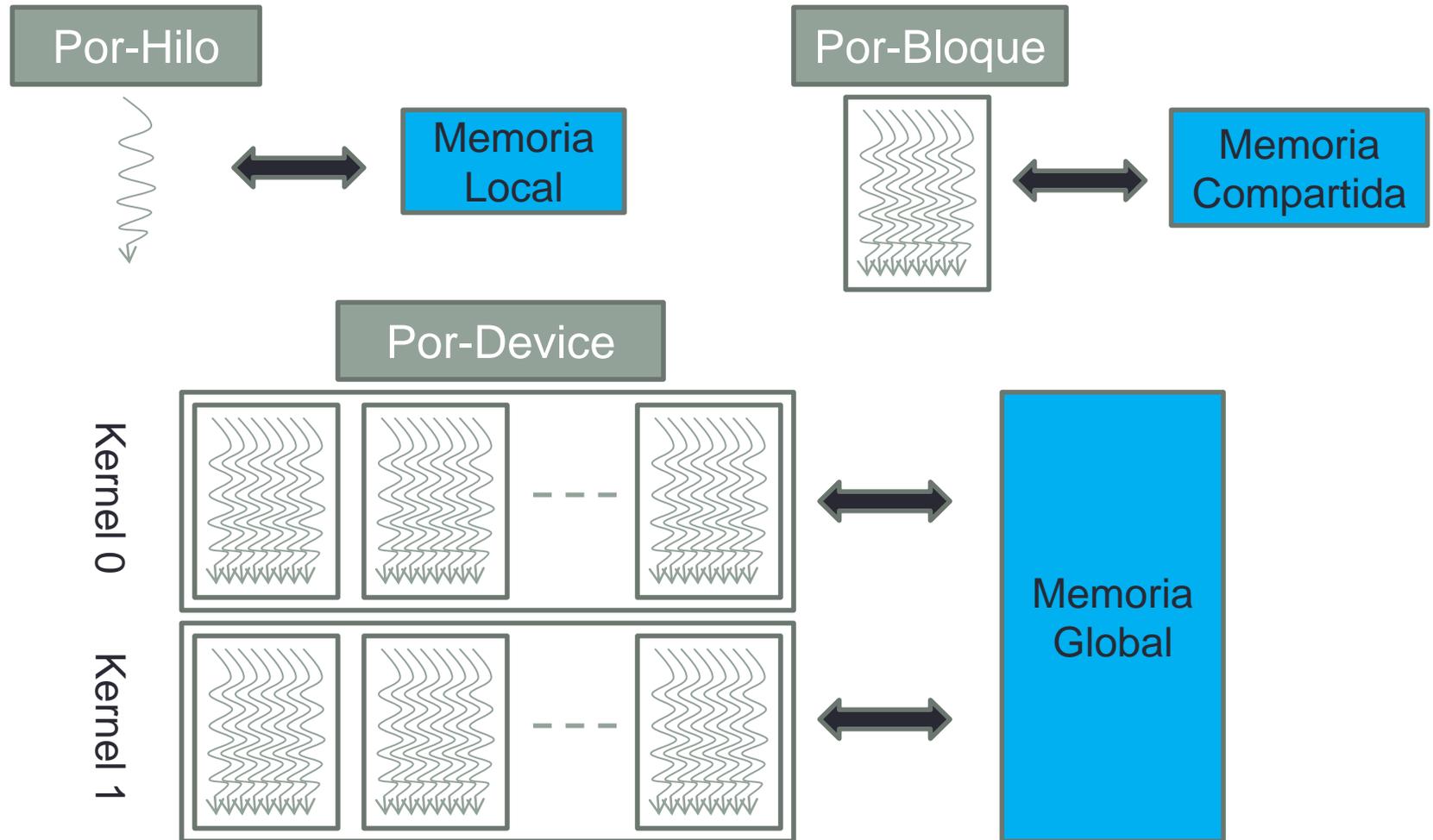
# Modelo de Programación en CUDA

- Ejecución en el Host y Device

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

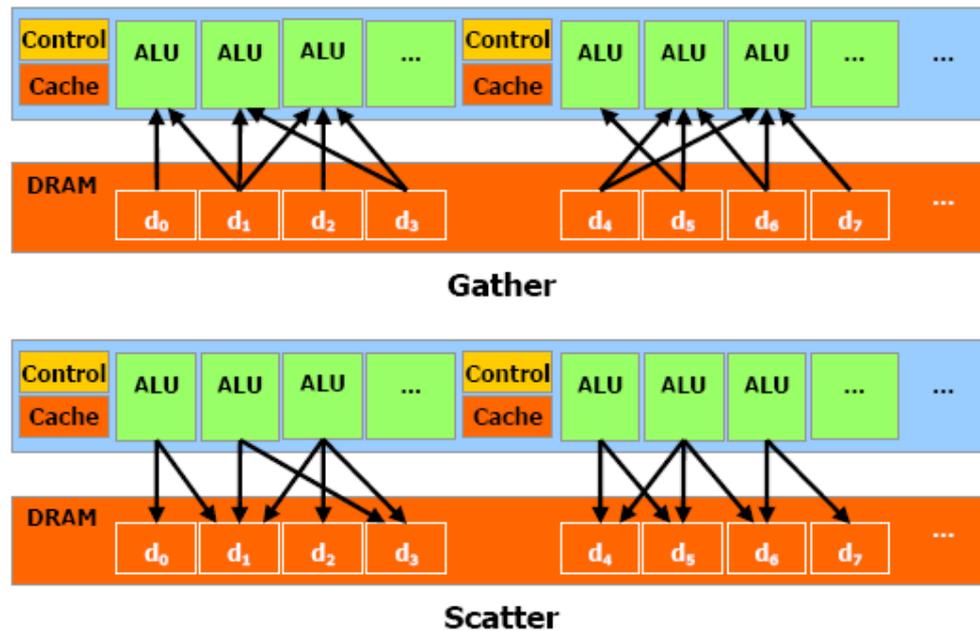


# Modelo de la Memoria en CUDA



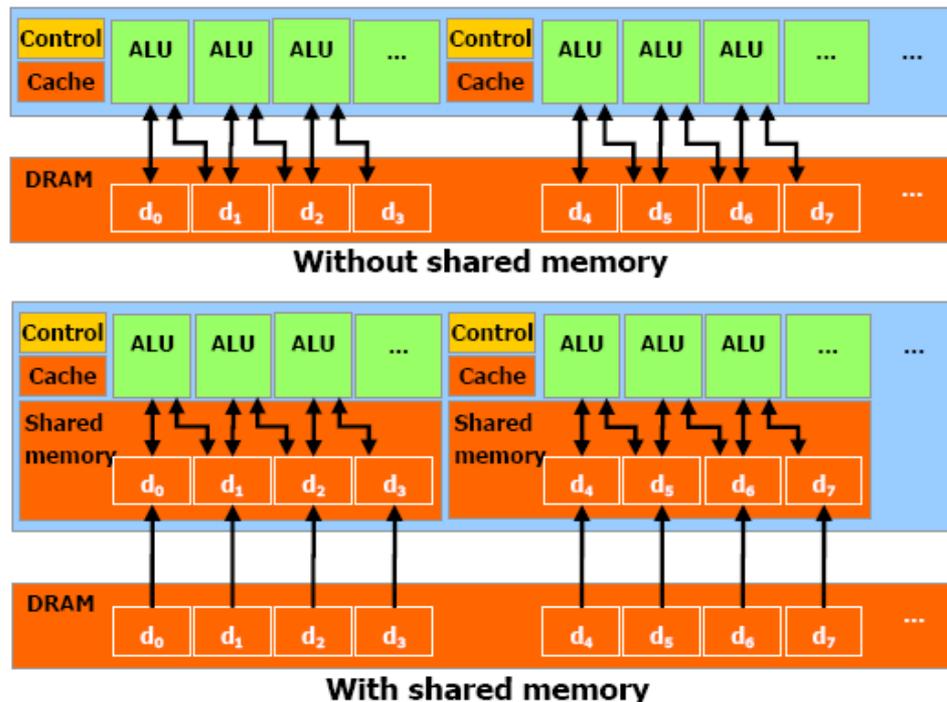
# Modelo de la Memoria en CUDA

- CUDA proporciona un direccionamiento de memoria que permite leer y escribir en cualquier localidad de la memoria DRAM

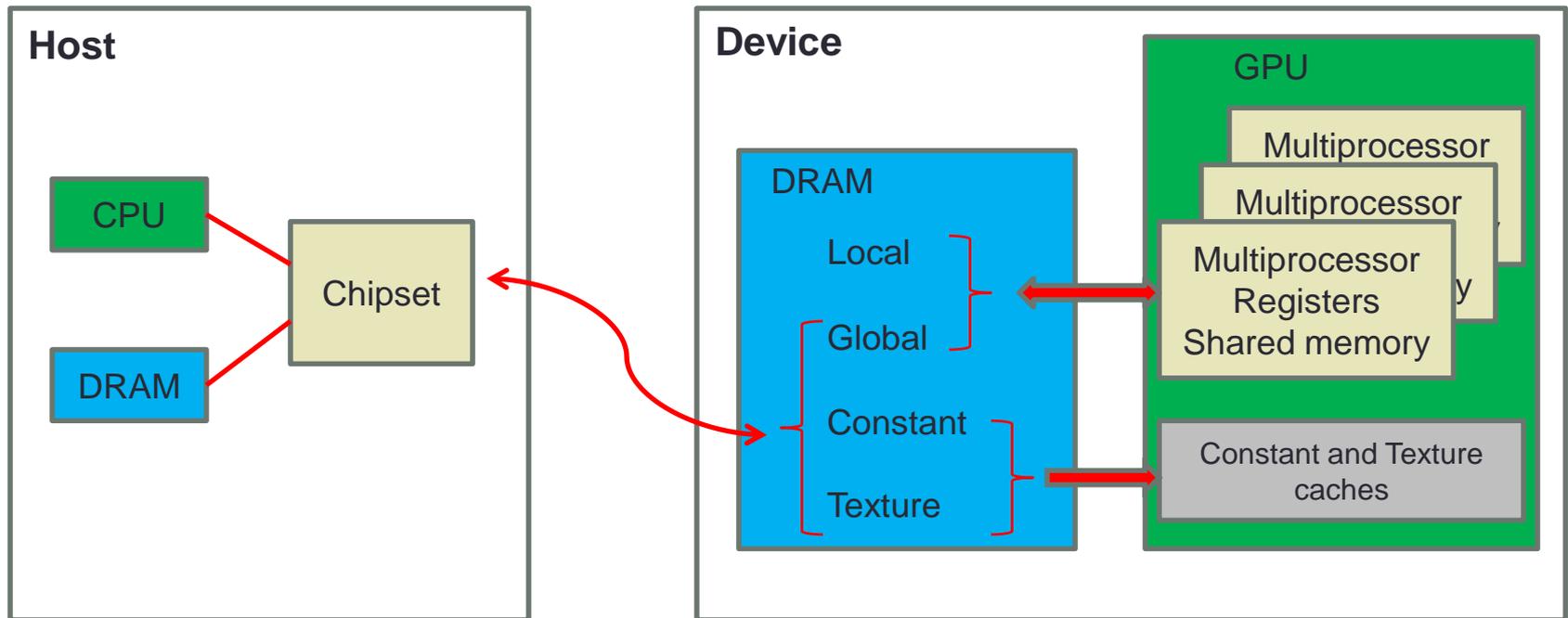


# Modelo de la Memoria en CUDA

- CUDA ofrece una memoria compartida que hace más rápido el acceso de lectura y escritura, permitiendo que varios threads compartan datos entre sí, minimizando los accesos a la memoria DRAM



# Modelo de la Memoria en CUDA



# Instrucciones para el Manejo de Memoria

- Crear memoria

- `cudaMalloc` `((void**) devPtr, size_t size)`
- `cudaMallocHost` `((void**) hostPtr, size_t size)`
- `cudaFree` `(void *devPtr)`
- `cudaFreeHost` `(void *hostPtr)`

# Instrucciones para el Manejo de Memoria

## • 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, cudaMemcpyHostToDevice, cudaMemcpyDeviceToHost, or cudaMemcpyDeviceToDevice.

# 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

- 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

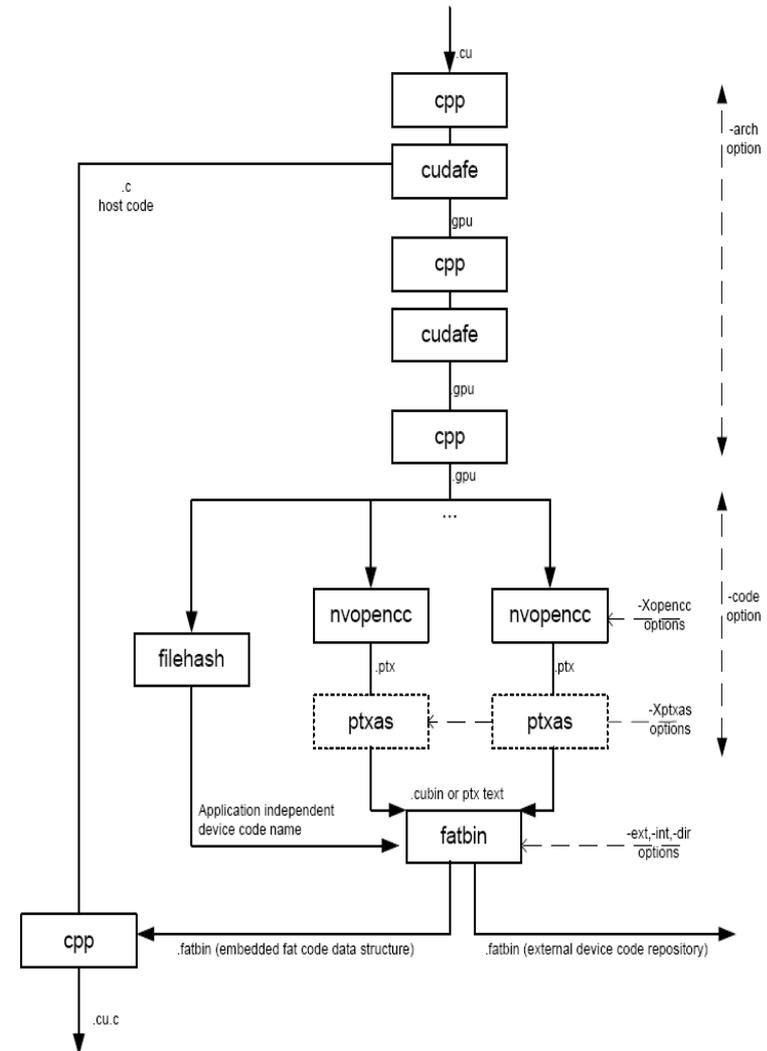
```
float3 temp[10];  
.....  
temp[i].x=0.0; temp[i].y=0.0; temp[i].z=0.0;
```

# 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]

# Tarjetas gráficas compatibles con CUDA



[http://www.nvidia.com/object/cuda\\_gpus.html](http://www.nvidia.com/object/cuda_gpus.html)

Architecturas	Capacidad
8-200 series	1.0-1.3
FERMI (400 series)	2.0-2.1
KEPLER (600 series)	3.0-3.5

Arquitectura y capacidad de las GPUs

# Tarjetas gráficas compatibles con CUDA

GeForce 8400 GS, C=1.1



TESLA C1060, C=1.3



Quadro FX 1700, C=1.1



GeForce GT 640, C=3.0



GeForce 8800 GT, C=1.1



GeForce GTX 480,  
C=2.0



GTX > GTS > GT > GS

# Instalando CUDA

- Instalar: Driver, Toolkit y SDK
  - <http://www.nvidia.com/content/cuda/cuda-downloads.html>

CUDA TOOLKIT 4.2  
CURRENT PRODUCTION RELEASE



**CUDA 4.2 FOR WINDOWS**

<b>1</b> Download Toolkit <a href="#">64bit</a> <a href="#">32bit</a>	<b>2</b> Download Drivers (V 301.32 Or 301.27)			<b>3</b> Download SDK <a href="#">64bit</a> <a href="#">32bit</a>
	Win7/Vista Desktop	Win7/Vista Notebook	Win XP Desktop	
	<a href="#">64bit</a> <a href="#">32bit</a>	<a href="#">64bit</a> <a href="#">32bit</a>	<a href="#">64bit</a> <a href="#">32bit</a>	

**CUDA 4.2 FOR LINUX**

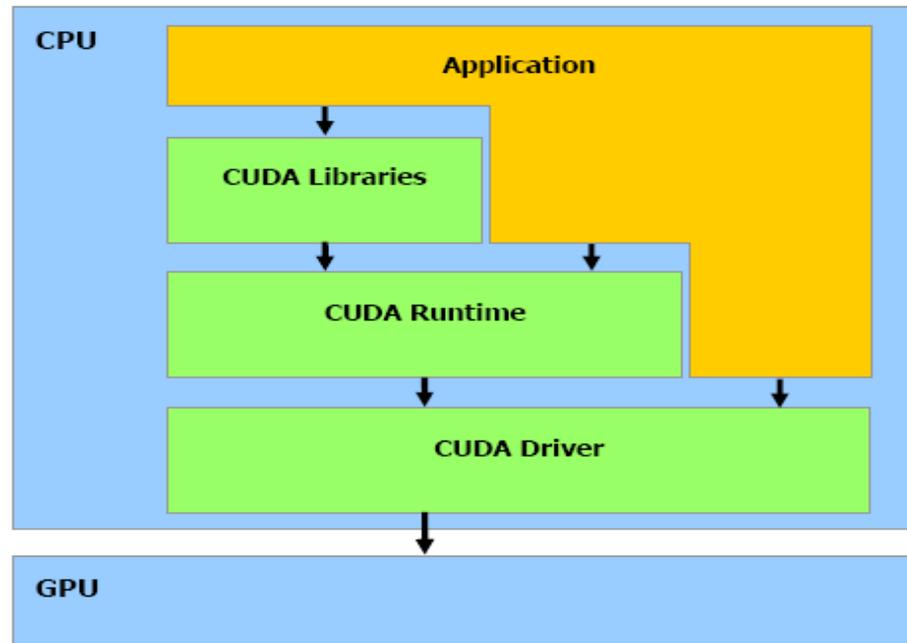
<b>1</b> Download Toolkit <a href="#">64bit</a> <a href="#">32bit</a>	Fedora 14	Redhat 5.5	Redhat 6.0	Ubuntu 11.04	Ubuntu 10.04	OpenSUSE 11.2	SUSE Server 11 SP1
	<a href="#">64bit</a> <a href="#">32bit</a>	<a href="#">64bit</a> <a href="#">32bit</a>	<a href="#">64bit</a>	<a href="#">64bit</a> <a href="#">32bit</a>	<a href="#">64bit</a> <a href="#">32bit</a>	<a href="#">64bit</a> <a href="#">32bit</a>	<a href="#">64bit</a> <a href="#">32bit</a>
	<b>2</b> Download Drivers(ver 295.41) <a href="#">64bit</a> <a href="#">32bit</a>			<b>3</b> Download SDK <a href="#">DOWNLOAD</a>			

**CUDA 4.2 FOR MAC**

<b>1</b> Download Toolkit <a href="#">DOWNLOAD</a>	<b>2</b> Download Drivers (Updated 06.11.12) <a href="#">DOWNLOAD</a>	<b>3</b> Download SDK <a href="#">DOWNLOAD</a>
---	--	---

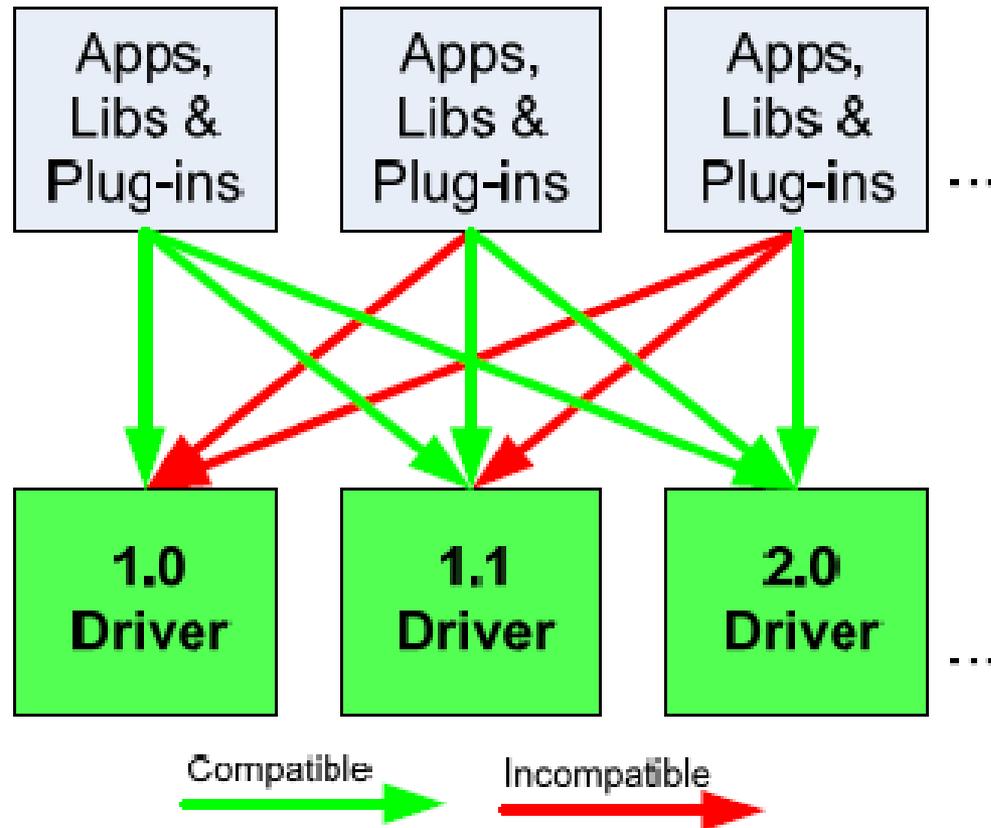
# Software CUDA

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



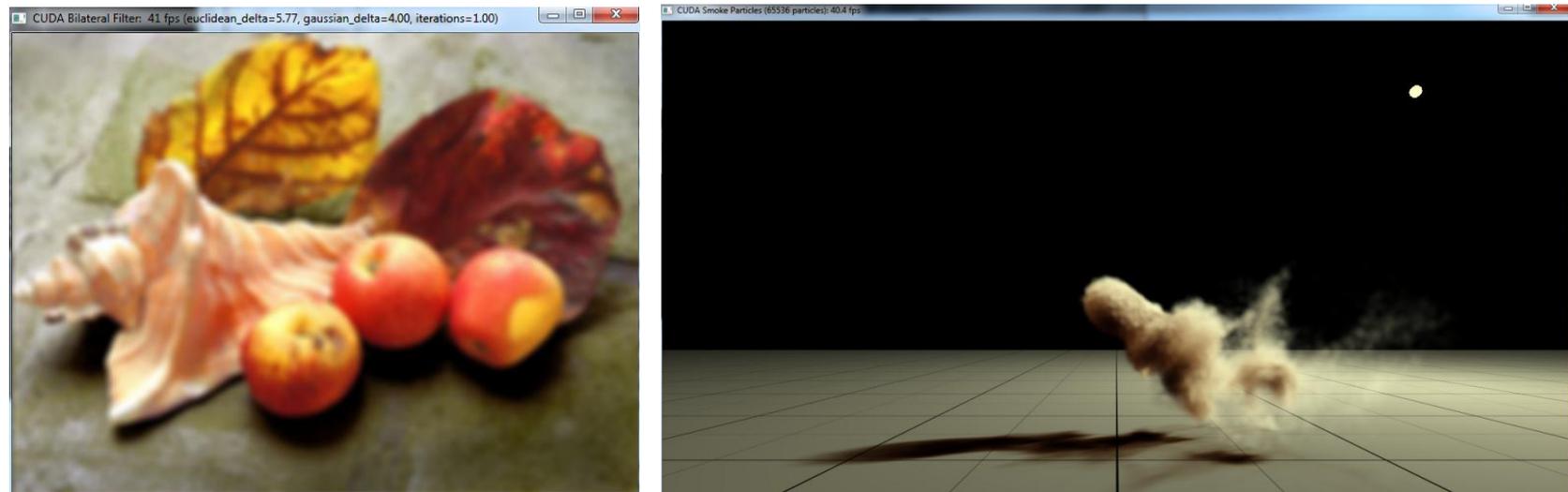
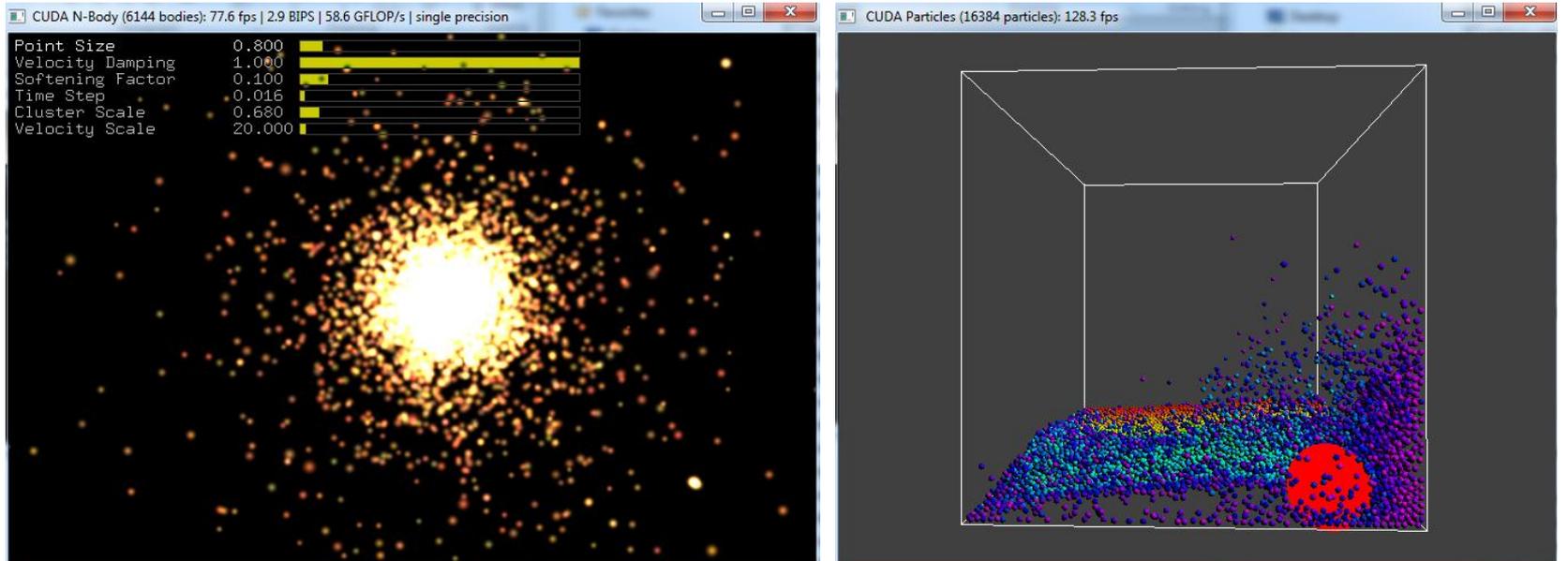
Ver [NVIDIA CUDA C Programming Guide]

# Compatibilidad de versiones



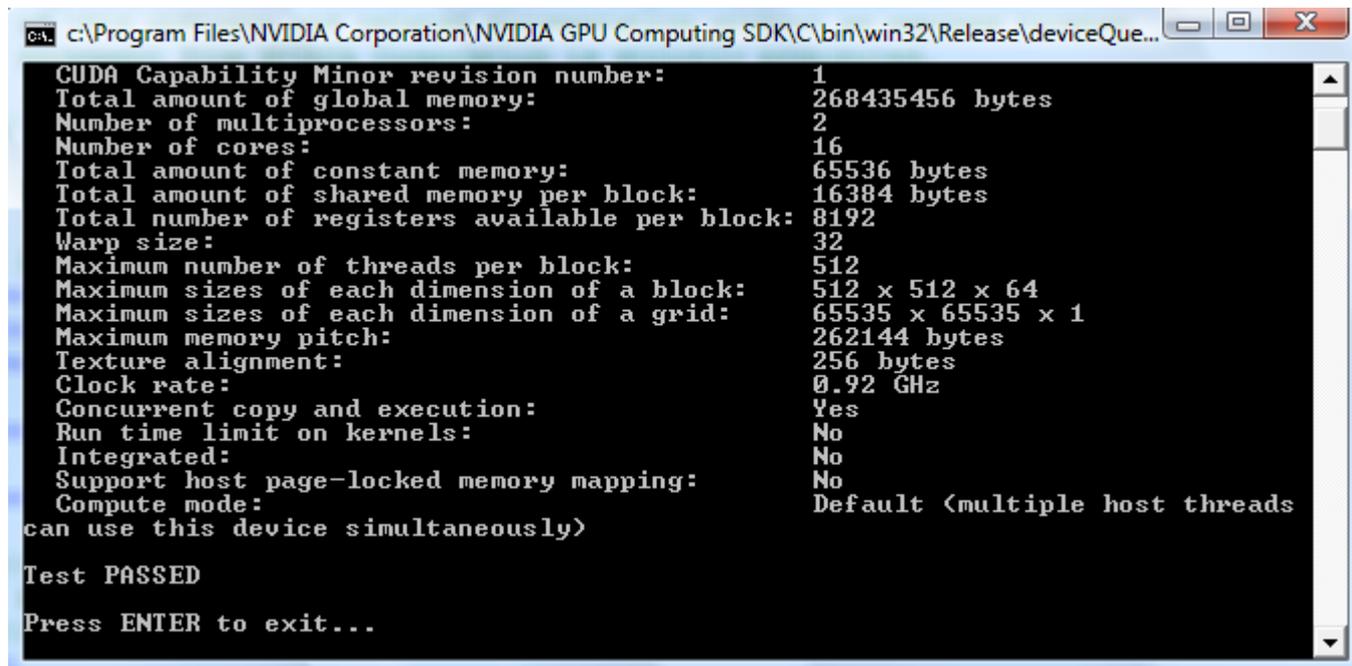
Ver [NVIDIA CUDA C Programming Guide]

# 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>

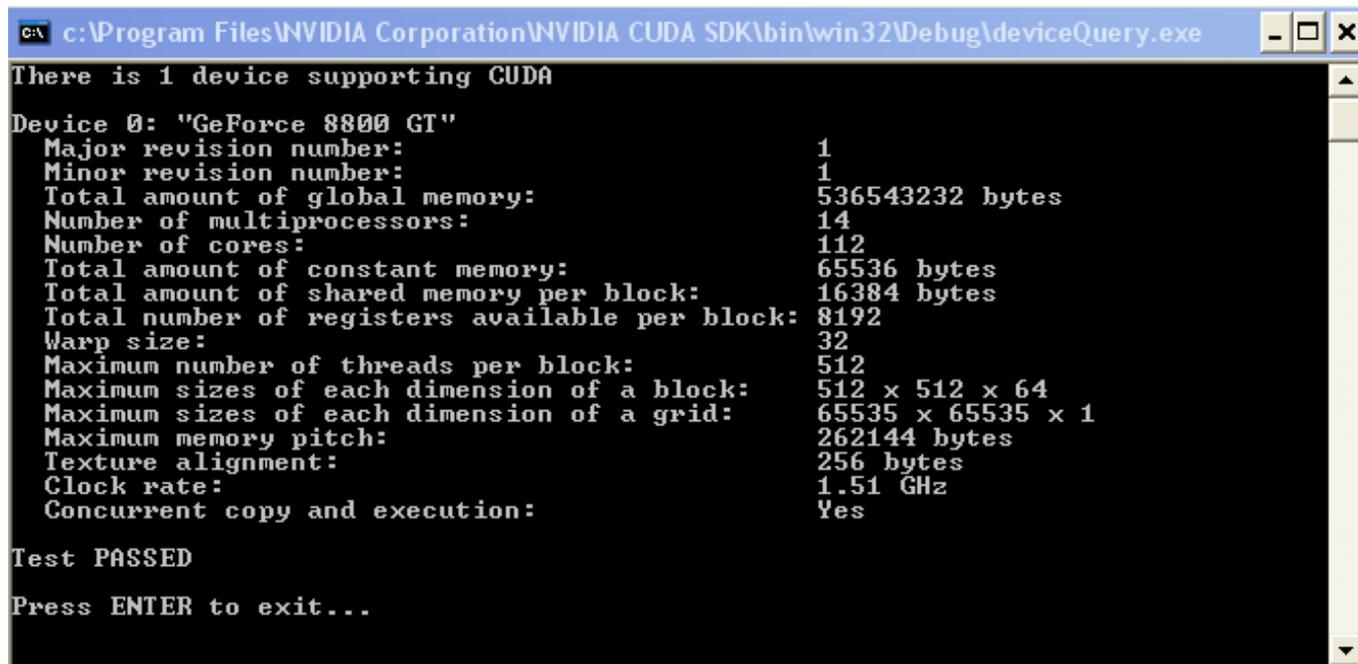
Test PASSED

Press ENTER to exit...
```

Resultado con una tarjeta NVIDIA GeForce 8400 GS

# “deviceQueryDrv”

- Para saber que capacidades tiene nuestra tarjeta de video:

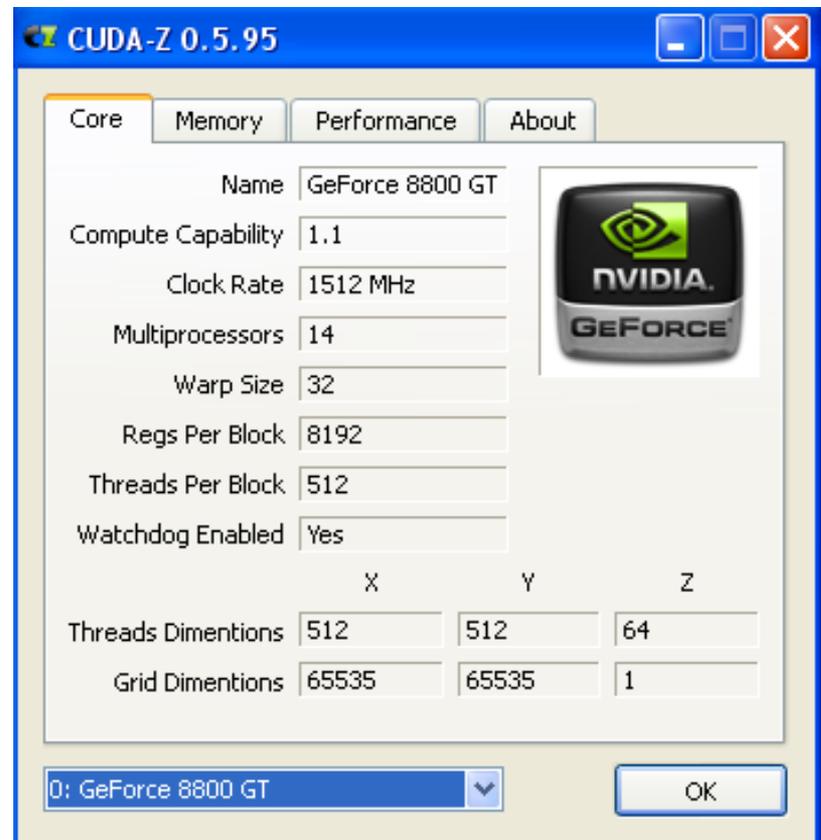
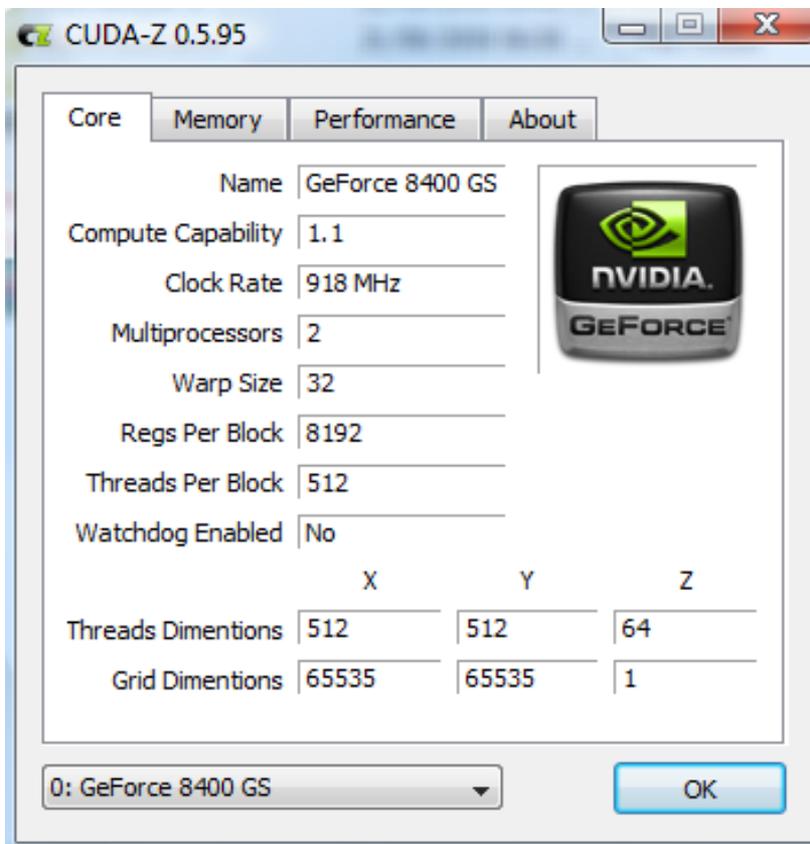


```
c:\Program Files\NVIDIA Corporation\NVIDIA CUDA SDK\bin\win32\Debug\deviceQuery.exe
There is 1 device supporting CUDA
Device 0: "GeForce 8800 GT"
Major revision number: 1
Minor revision number: 1
Total amount of global memory: 536543232 bytes
Number of multiprocessors: 14
Number of cores: 112
Total 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...
```

Resultado con una tarjeta NVIDIA GeForce 8800 GT

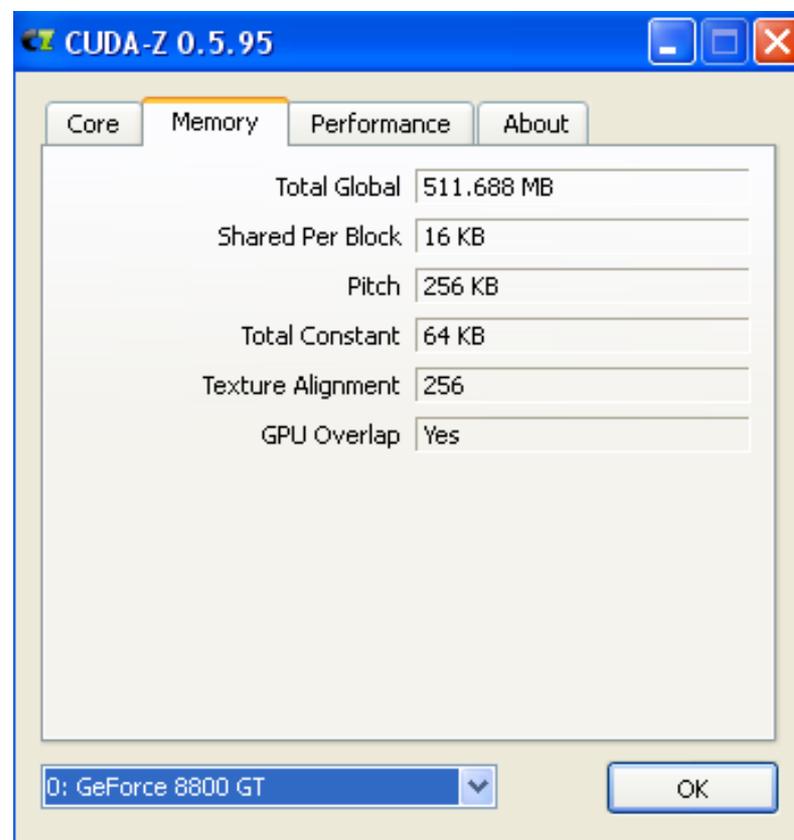
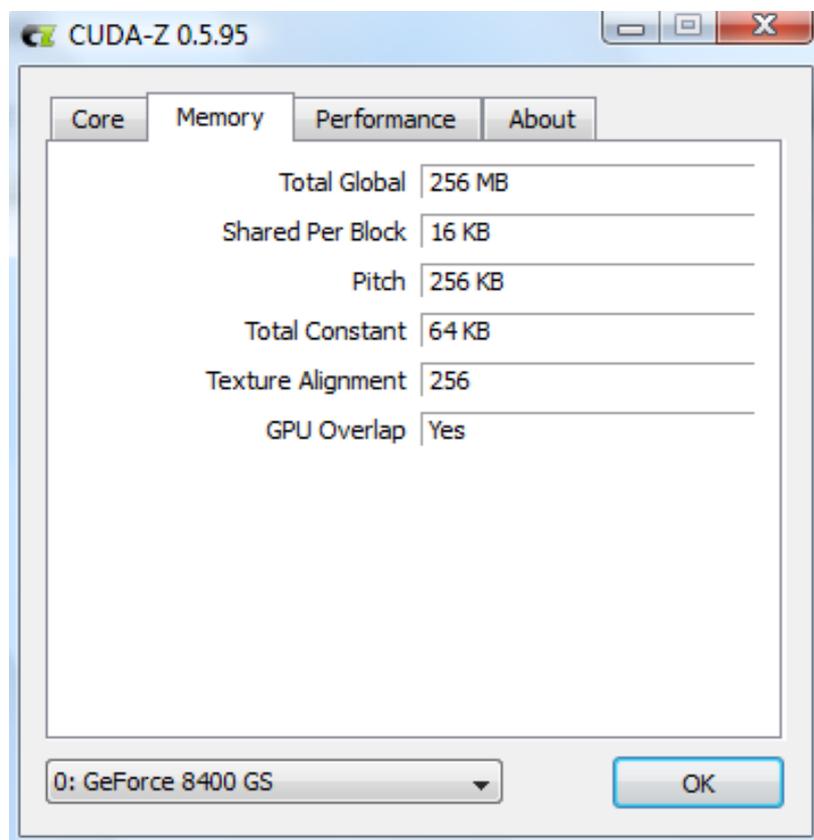
# CUDA-Z

- GeForce 8400 GS & GeForce 8800 GT



# CUDA-Z

- GeForce 8400 GS & GeForce 8800 GT



# CUDA-Z

- GeForce 8400 GS & GeForce 8800 GT

Performance tab for GeForce 8400 GS. The window title is 'CUDA-Z 0.5.95'. The Performance tab is selected. The data is as follows:

	Pinned	Pageable
Memory Copy		
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

Performance tab for GeForce 8800 GT. The window title is 'CUDA-Z 0.5.95'. The Performance tab is selected. The data is as follows:

	Pinned	Pageable
Memory Copy		
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

The screenshot shows the 'Core' tab of the CUDA-Z 0.5.95 application for a Quadro FX 5600 GPU. The interface includes a list of properties and a 3D grid configuration section.

Property	Value
Name	Quadro FX 5600
Compute Capability	1.0
Clock Rate	1350 MHz
Multiprocessors	16
Warp Size	32
Regs Per Block	8192
Threads Per Block	512
Watchdog Enabled	Yes

	X	Y	Z
Threads Dimentions	512	512	64
Grid Dimentions	65535	65535	1

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

The screenshot shows the 'Core' tab of the CUDA-Z 0.5.95 application for a Tesla C1060 GPU. The interface includes a list of properties and a 3D grid configuration section.

Property	Value
Name	Tesla C1060
Compute Capability	1.3
Clock Rate	1296 MHz
Multiprocessors	30
Warp Size	32
Regs Per Block	16384
Threads Per Block	512
Watchdog Enabled	No

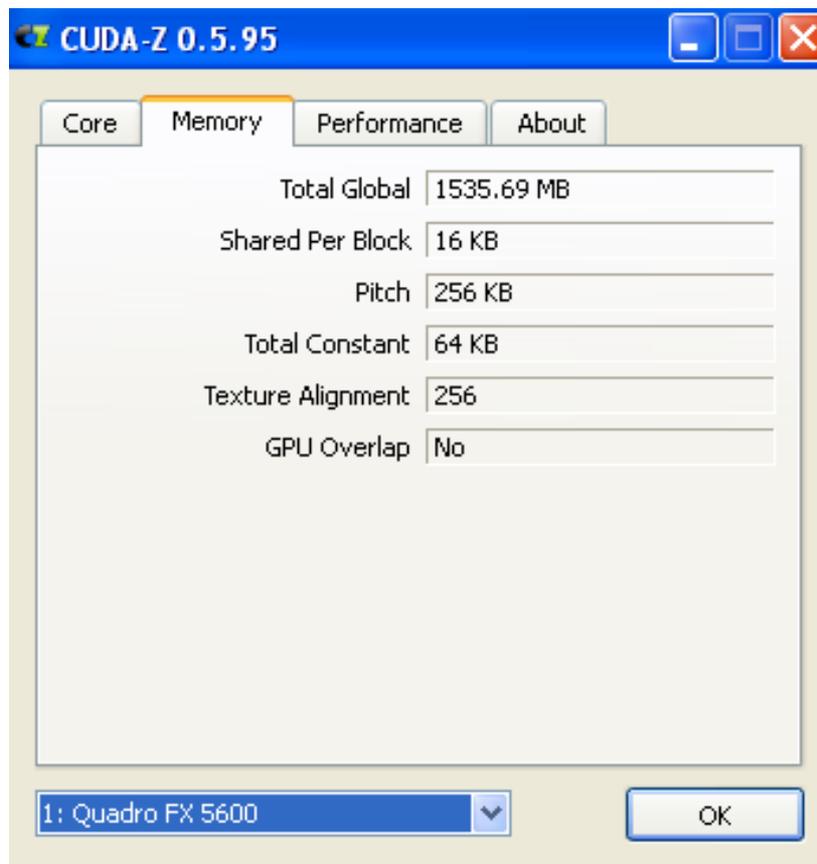
  

	X	Y	Z
Threads Dimentions	512	512	64
Grid Dimentions	65535	65535	1

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

# CUDA-Z

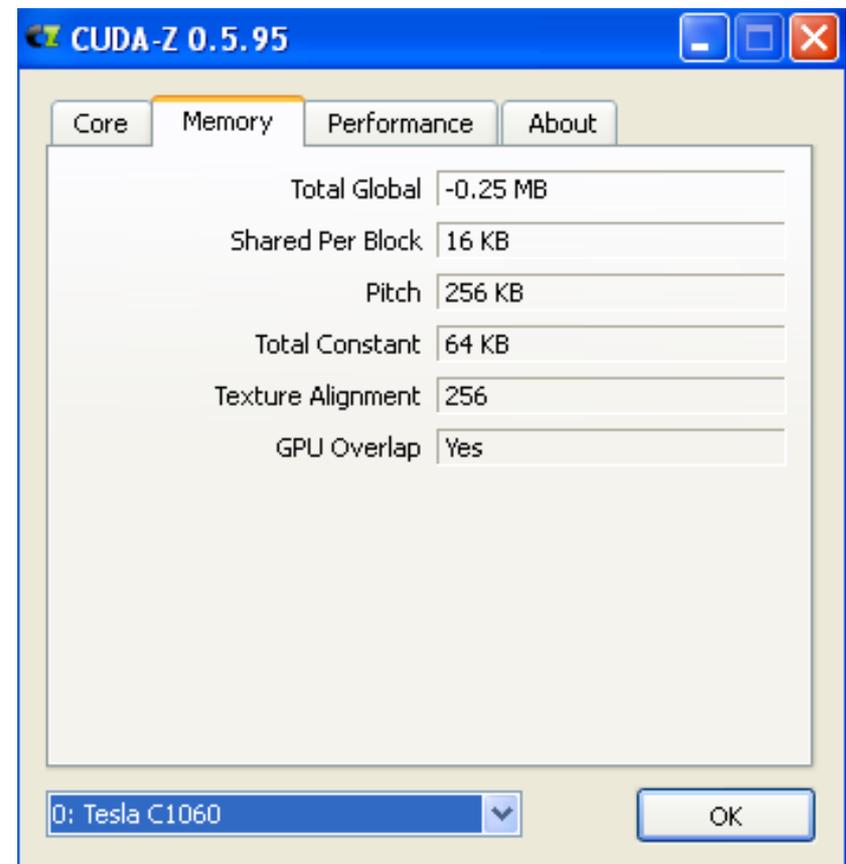
- Quadro FX 5600 & Tesla C1060



Screenshot of the CUDA-Z 0.5.95 application window showing the Memory tab for a Quadro FX 5600 GPU. The window title is "CUDA-Z 0.5.95". The tabs are Core, Memory (selected), Performance, and About. The Memory tab displays the following settings:

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 visible.



Screenshot of the CUDA-Z 0.5.95 application window showing the Memory tab for a Tesla C1060 GPU. The window title is "CUDA-Z 0.5.95". The tabs are Core, Memory (selected), Performance, and About. The Memory tab displays the following settings:

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 visible.

# CUDA-Z

- Quadro FX 5600 & Tesla C1060

The screenshot shows the Performance tab of the CUDA-Z 0.5.95 application for a Quadro FX 5600 GPU. The interface includes tabs for Core, Memory, Performance, and About. The Performance section is divided into Memory Copy (Pinned and Pageable), GPU Core Performance, and a checkbox for 'Update Results in Background'. The data is as follows:

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	Value
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

At the bottom, the device is identified as '1: Quadro FX 5600' and there is an 'OK' button.

The screenshot shows the Performance tab of the CUDA-Z 0.5.95 application for a Tesla C1060 GPU. The interface is identical to the previous one, but the data is for the Tesla C1060. The data is as follows:

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	Value
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

At the bottom, the device is identified as '0: Tesla C1060' and there is an 'OK' button.

# CUDA-Z

- GeForce GTX 480 & GeForce GT 640

CUDA-Z 0.6.163

Core Memory Performance About

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

Watchdog Enabled Yes

Integrated GPU No

Concurrent Kernels Yes

Compute Mode Default

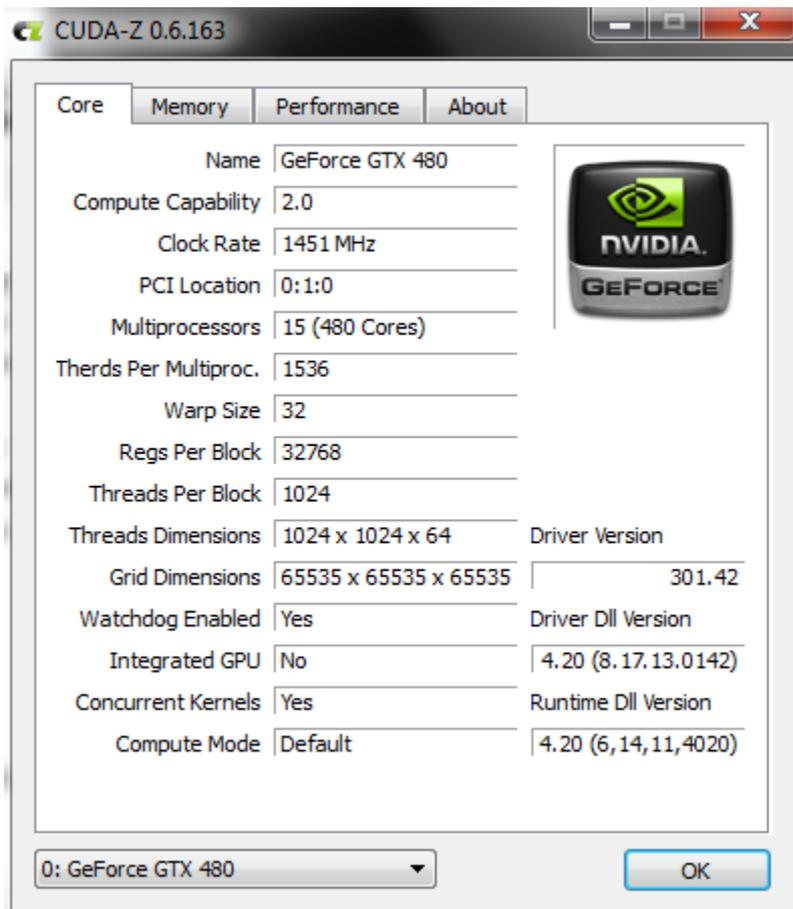
Driver Version 301.42

Driver Dll Version 4.20 (8.17.13.0142)

Runtime Dll Version 4.20 (6,14,11,4020)

0: GeForce GTX 480

OK



CUDA-Z 0.6.163

Core Memory Performance About

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

Watchdog Enabled Yes

Integrated GPU No

Concurrent Kernels Yes

Compute Mode Default

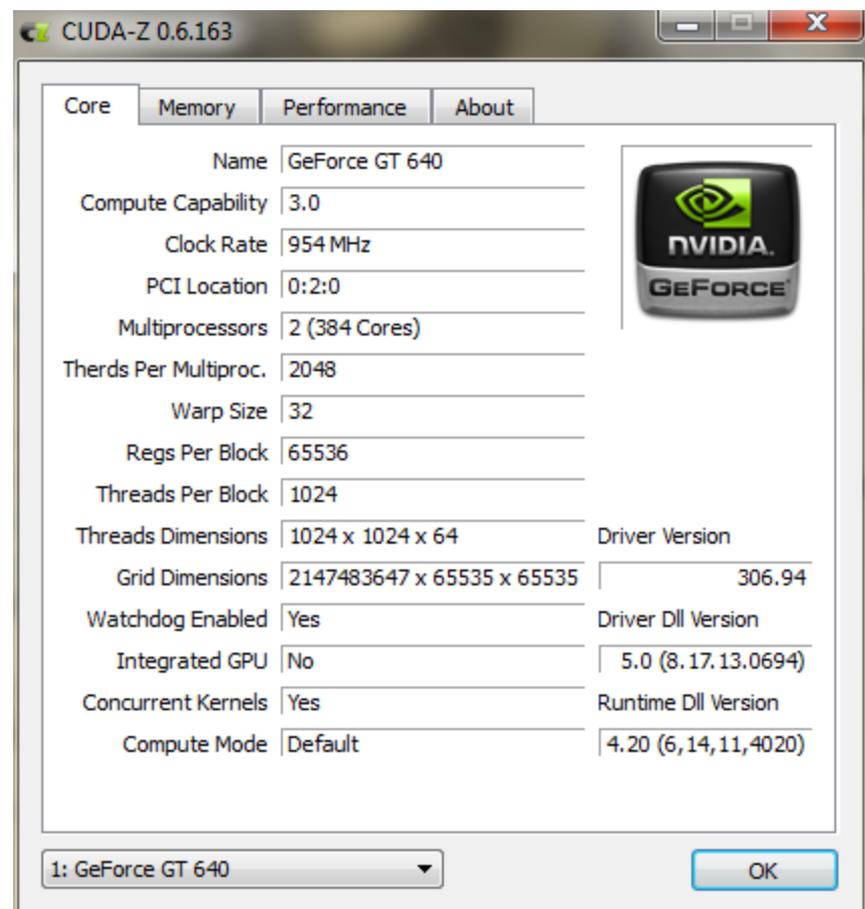
Driver Version 306.94

Driver Dll Version 5.0 (8.17.13.0694)

Runtime Dll Version 4.20 (6,14,11,4020)

1: GeForce GT 640

OK



# CUDA-Z

- GeForce GTX 480 & GeForce GT 640

CUDA-Z 0.6.163

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 65536
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

CUDA-Z 0.6.163

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

0: GeForce GTX 480

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 >> ▼

OK

1: GeForce GT 640

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 >> ▼

OK

# Preguntas



**GRACIAS**

