COMPUTE UNIFIED DEVICE ARCHITECTURE (CUDA)

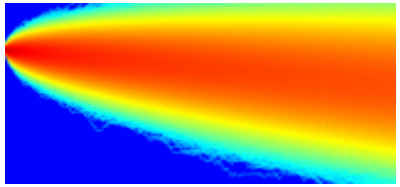
Francisco J. Hernández López

fcoj23@cimat.mx

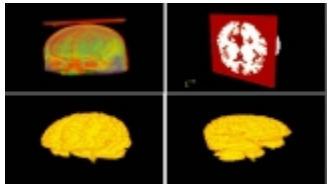
TARJETAS DE VIDEO (GPUS)

GPUS

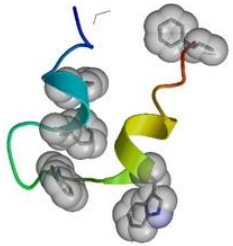
- Procesadores flexibles de procesamiento general
- Se pueden resolver problemas de diversas áreas:
  - Finanzas, Gráficos, Procesamiento de Imágenes y Video, Algebra Lineal, Física, Química, Biología, etc.



Ecs. Diferenciales



Segmentación de Imágenes Medicas

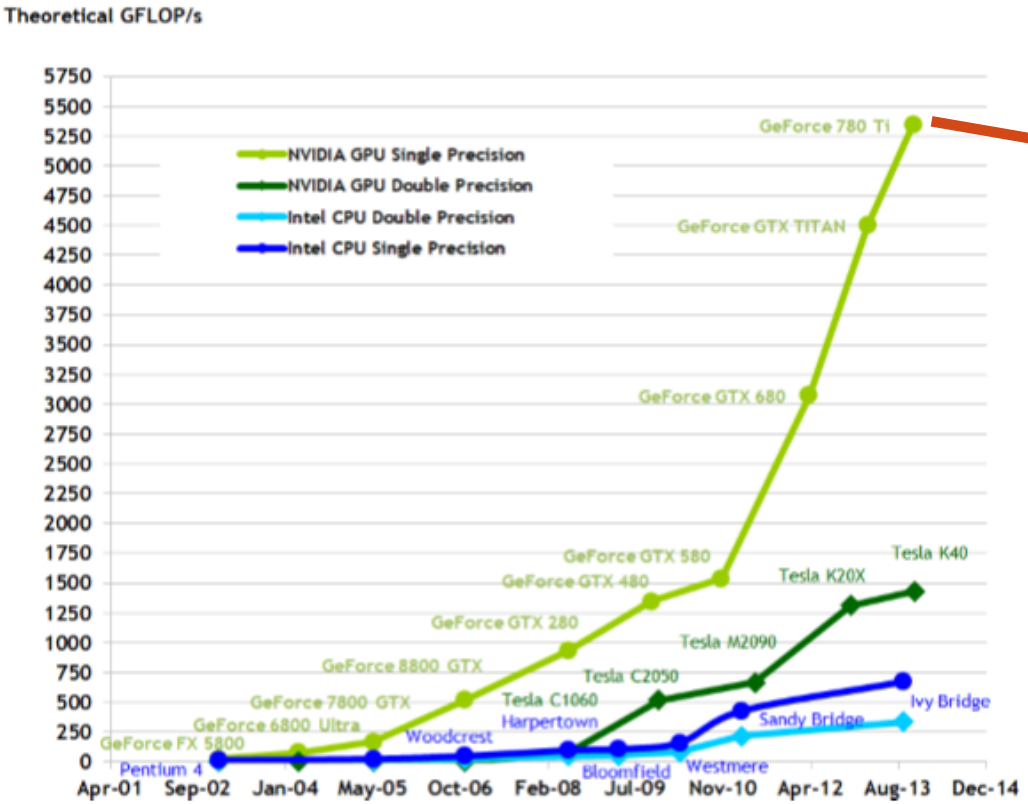


Dinámica Molecular

Detección de Objetos

Visitar CUDA ZONE

GPUS VS CPUS

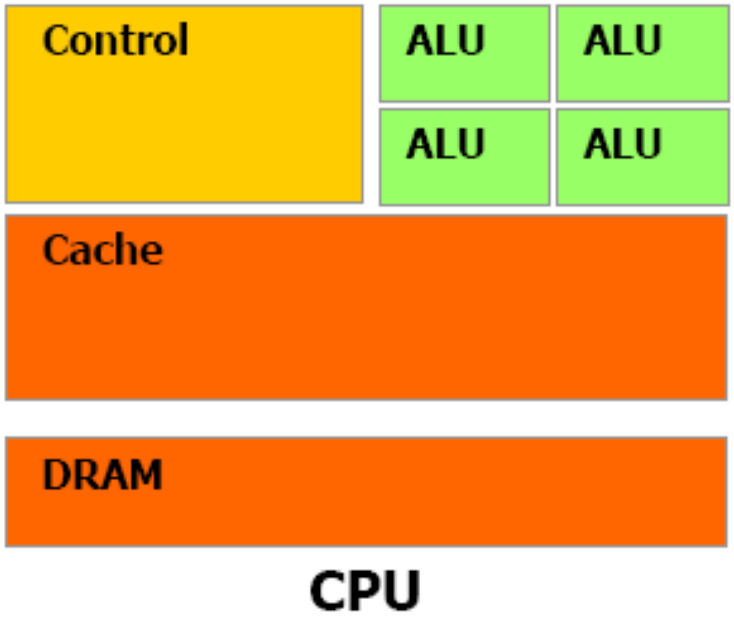


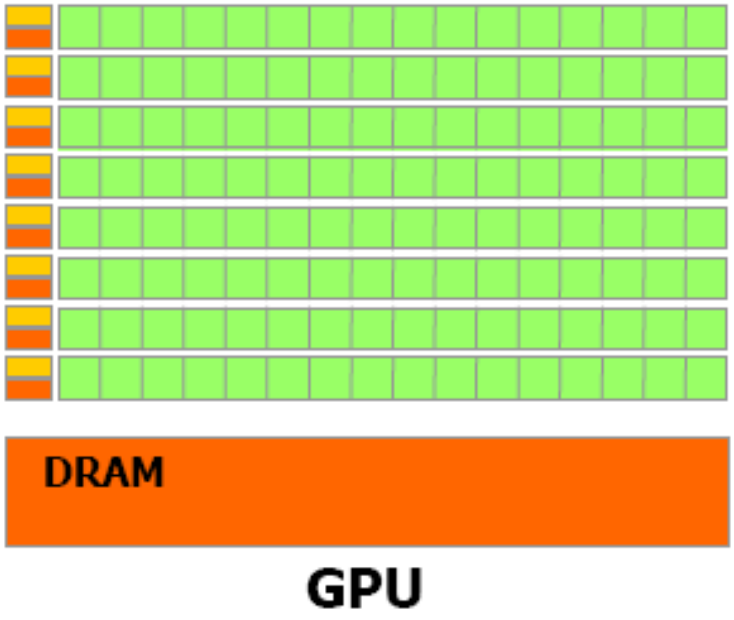
CUDA_C_Programming_Guide.pdf

GeForce GTX 780Ti

GPUs vs CPUs

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





CUDA_C_Programming_Guide.pdf

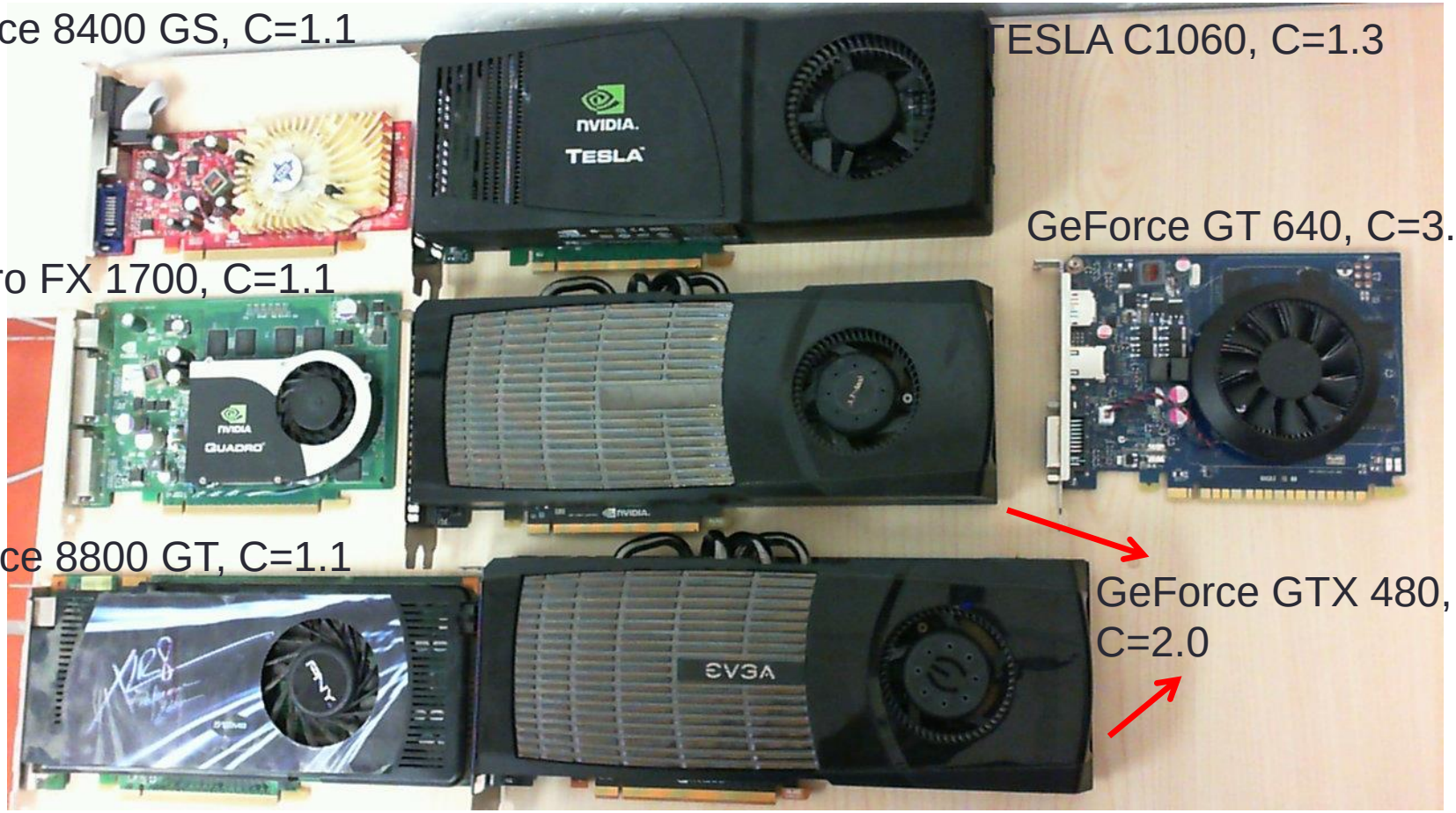
CUDA

- Es una tecnología de propósito general que nos permite ejecutar código en GPUs para hacer Cómputo Paralelo
- Desarrollado por NVIDIA en el 2006

Arquitectura	Capacidad	Nuevas Arq. (2014-2018)	Capacidad
8-200 series	1.0 - 1.3	MaxWell	5.0 - 5.2
FERMI (400 series)	2.0 - 2.1	Pascal	6.x
KEPLER (600 series)	3.0 - 3.5	Volta	7.x

Arquitecturas de las GPUs y sus capacidades

TARJETAS GRÁFICAS COMPATIBLES CON CUDA



GTX > GTS > GT > GS

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.

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)

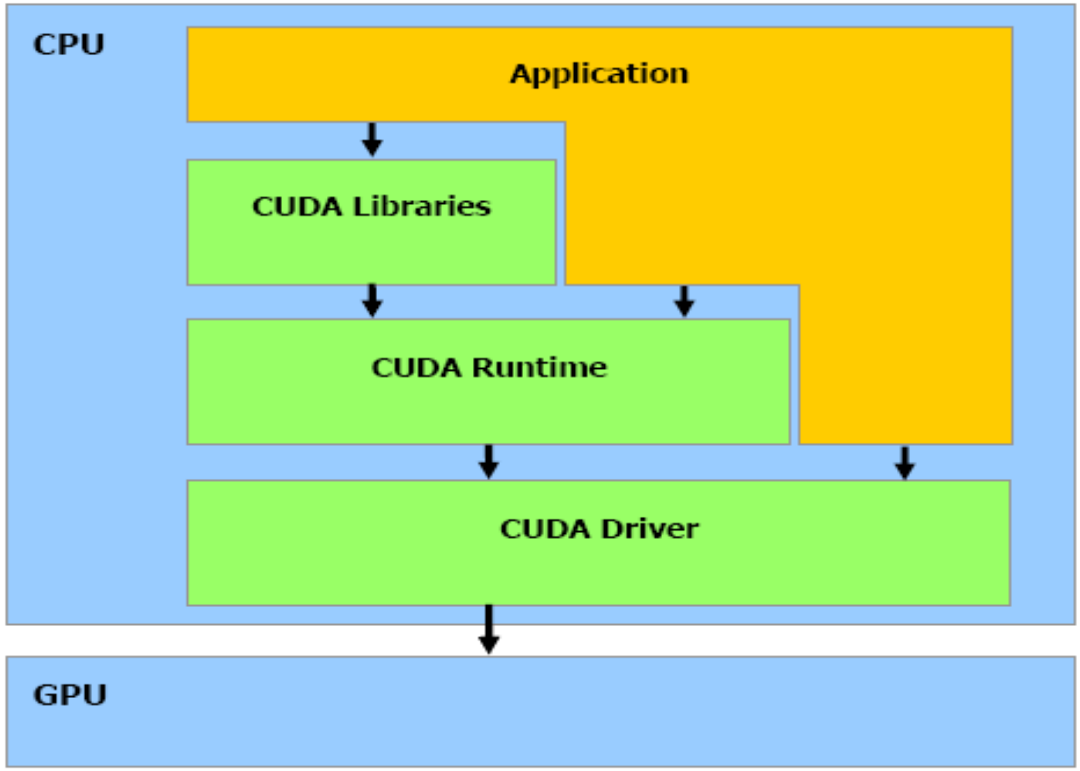
CUDA-Enabled NVIDIA GPUs

Volta Architecture (compute capabilities 7.x)				Tesla V Series
Pascal Architecture (compute capabilities 6.x)		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
				

CUDA_C_Programming_Guide.pdf

SOFTWARE CUDA

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



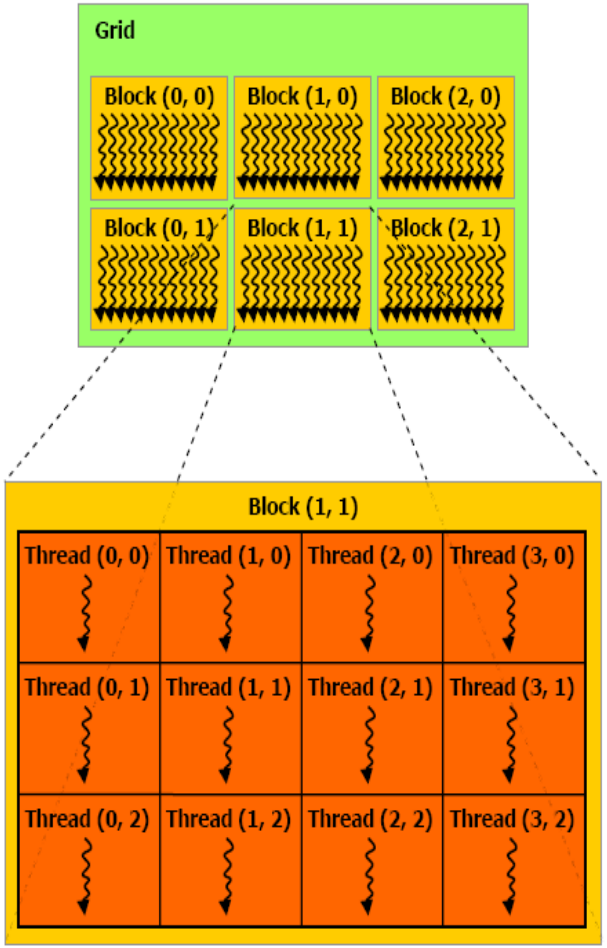
MODELO DE PROGRAMACIÓN EN CUDA

- 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 cuadricula (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.

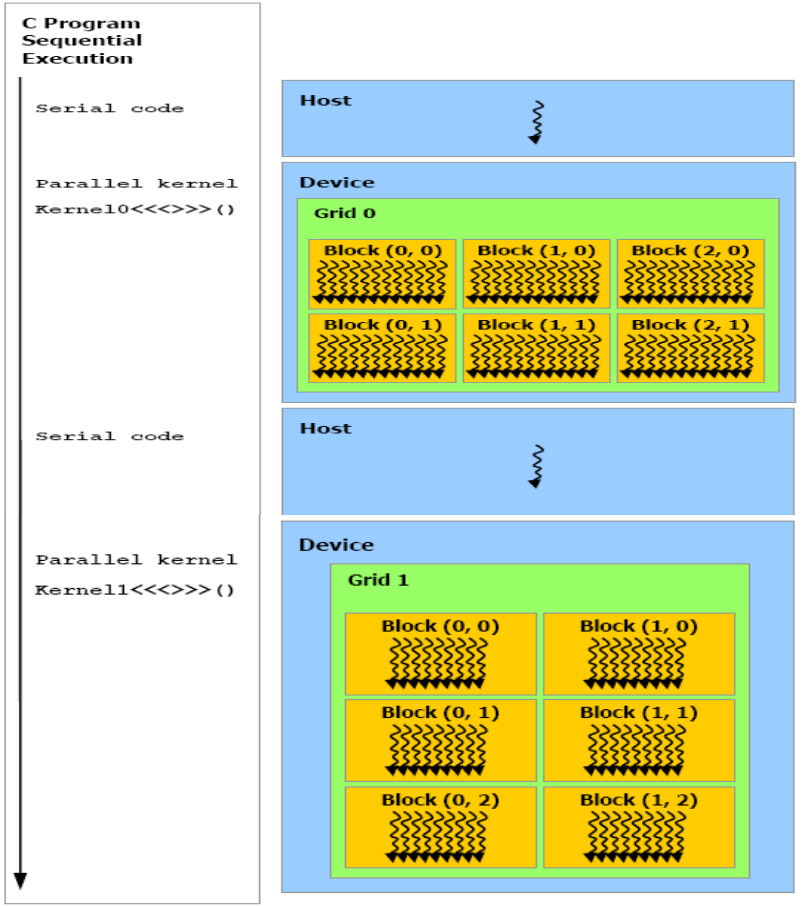
- Un Grid de bloques de hilos:
  - Tiene un número limitado de hilos en un bloque.
  - Los bloques se identifican mediante un ID.
  - Pueden ser arreglos de 1 o 2 dimensiones (actualmente ya soporta hasta 3 dimensiones).



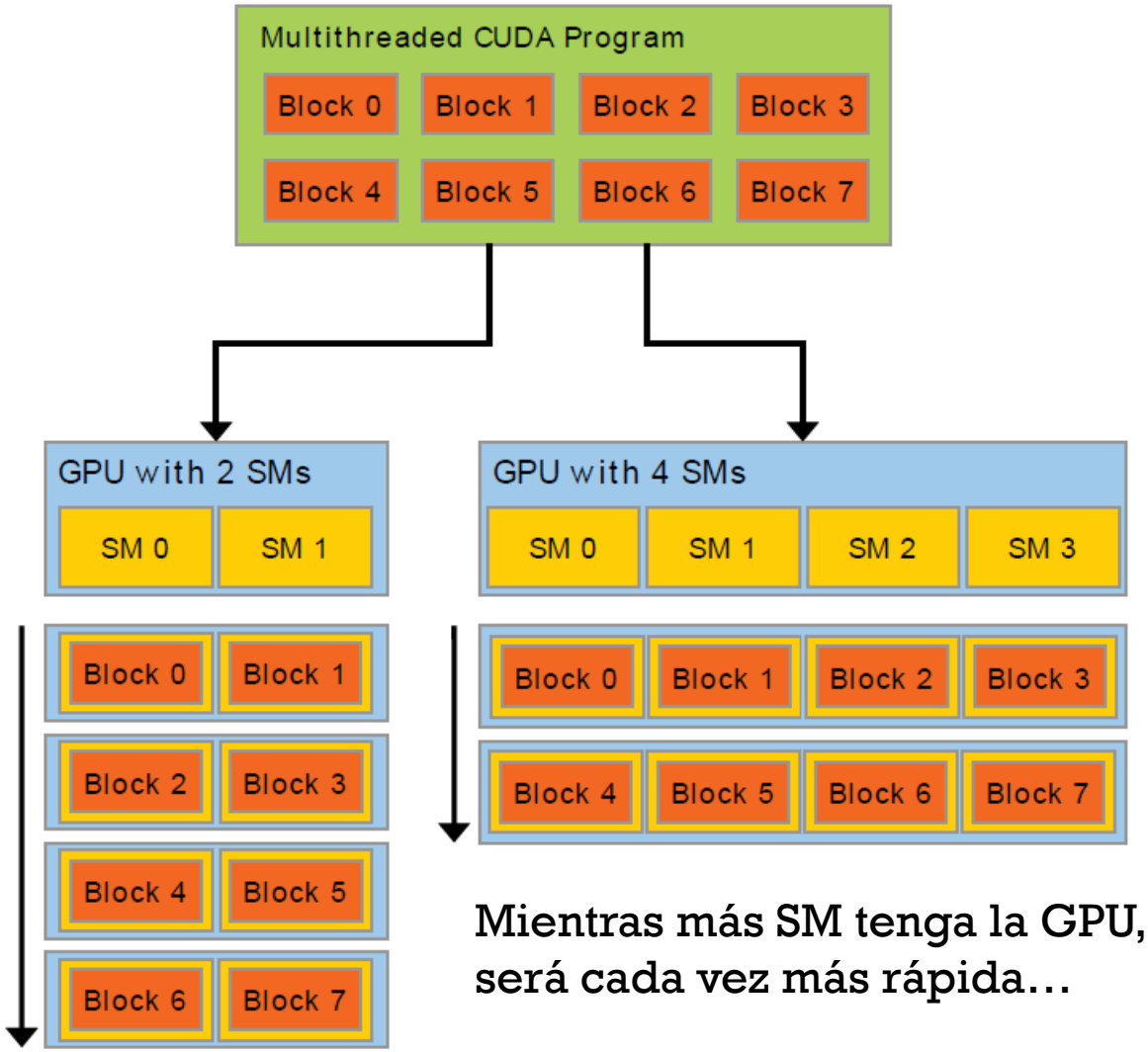
MODELO DE PROGRAMACIÓN EN CUDA

- Ejecución en el Host y Device.

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



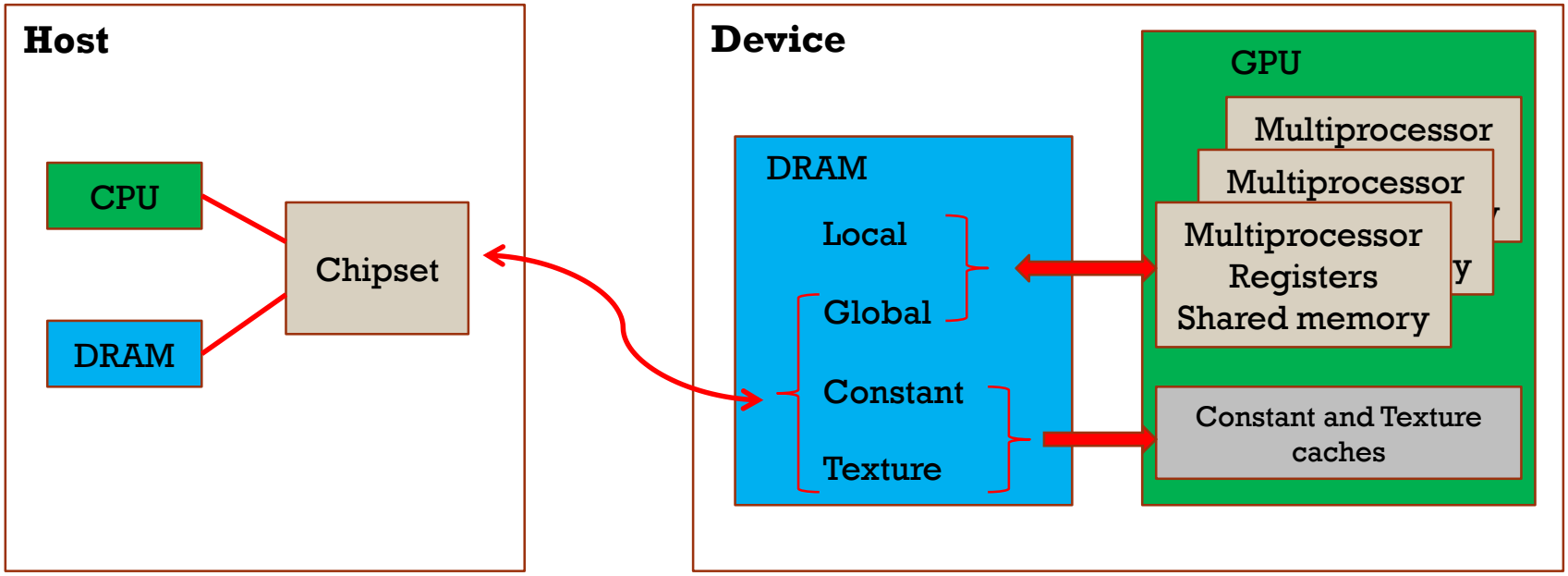
ESCALABILIDAD AUTOMÁTICA



Mientras más SM tenga la GPU, está será cada vez más rápida…

CUDA_C_Programming_Guide.pdf

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 (C1)

- 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 constant.
- 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

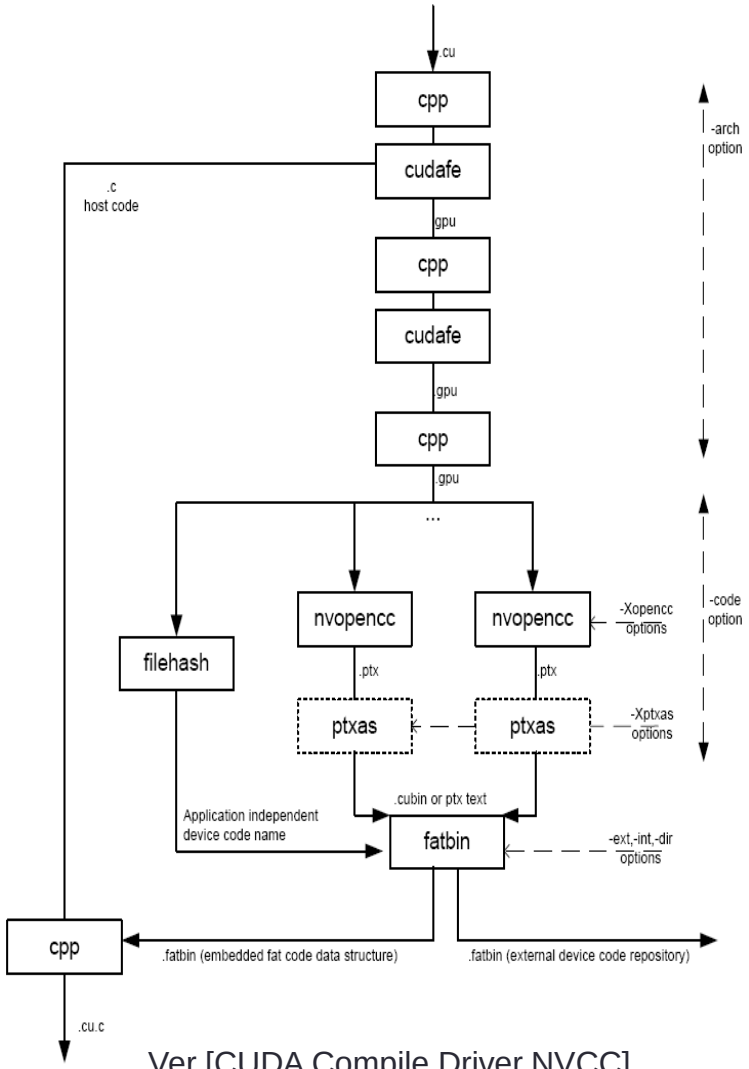
```
float3 temp[10];
……..
temp[i].x=0.0; temp[i].y=0.0; temp[i].z=0.0;
```

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

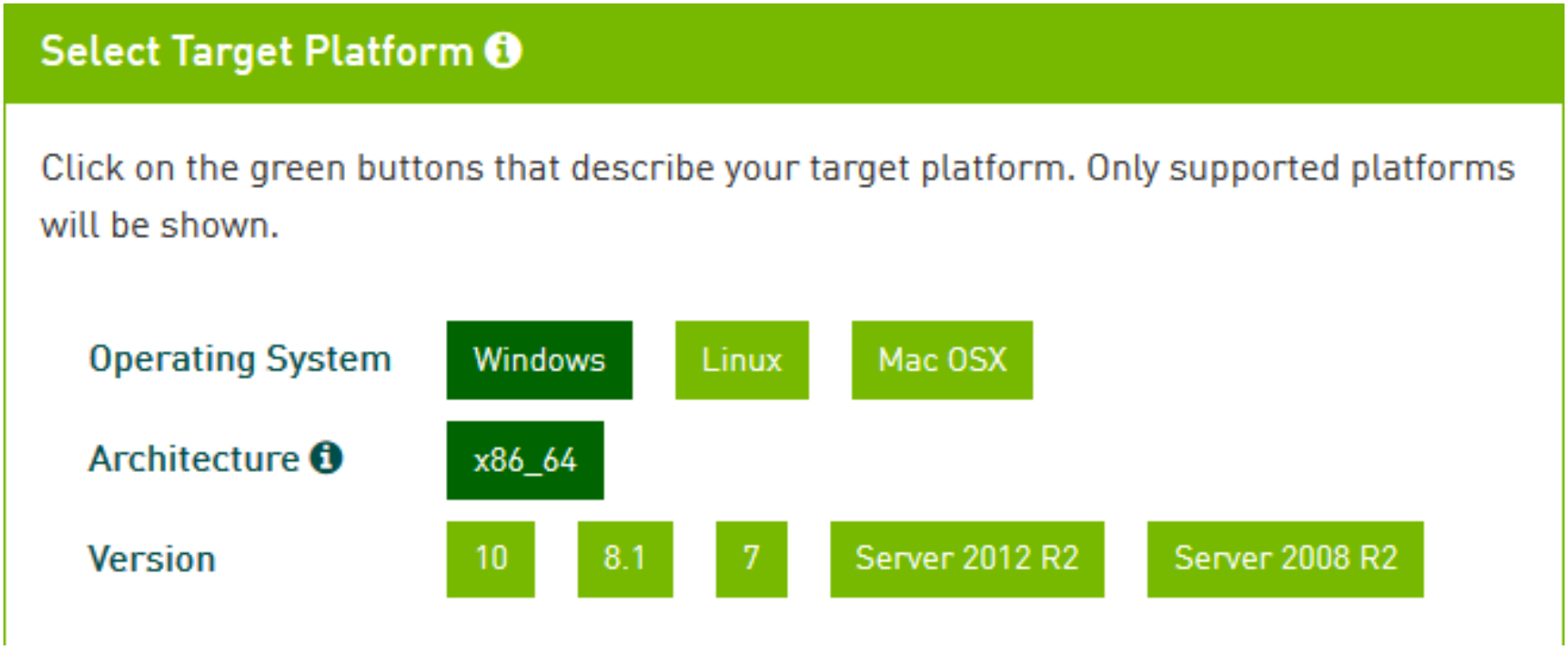
- El *nvcc*, es el encargado de compilar el código CUDA
- Soporta C/C++
- El *nvcc* utiliza los siguientes compiladores para el código *host*:
  - Linux: gcc, g++
  - Windows: Microsoft VS C/C++
  - Mac: Xcode



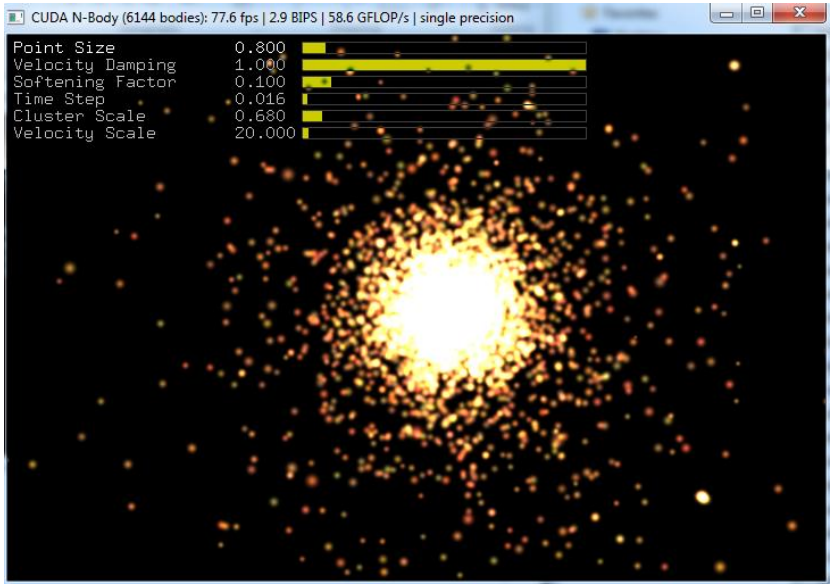
Ver [CUDA Compile Driver NVCC]

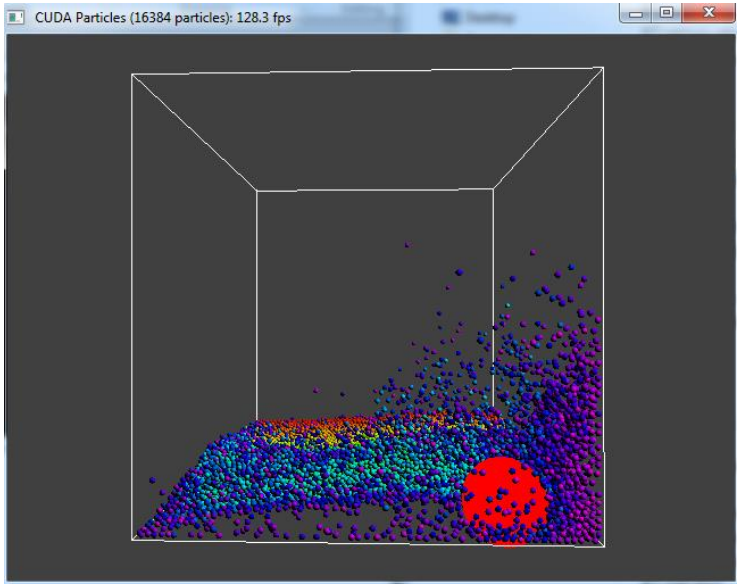
INSTALANDO CUDA

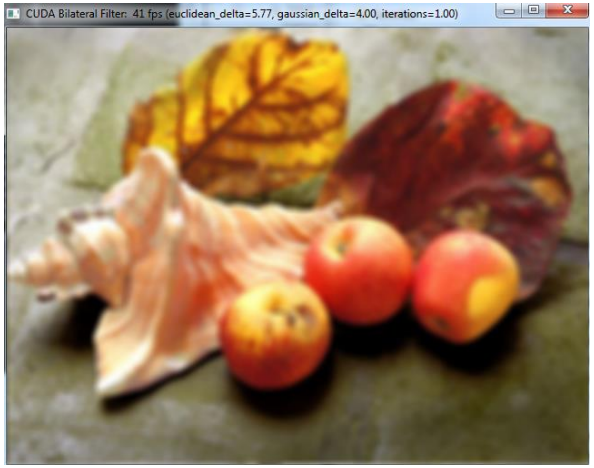
http://developer.nvidia.com/cuda/cuda-downloads

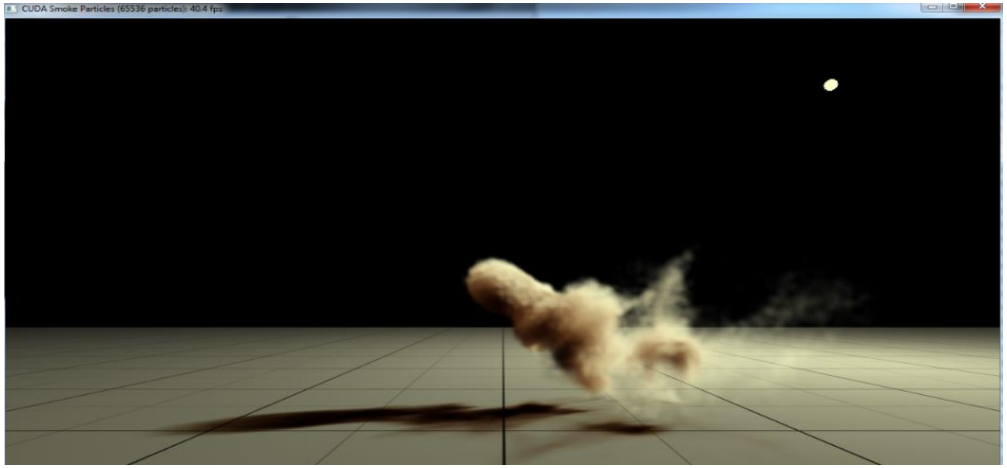


EJEMPLOS DEL SDK



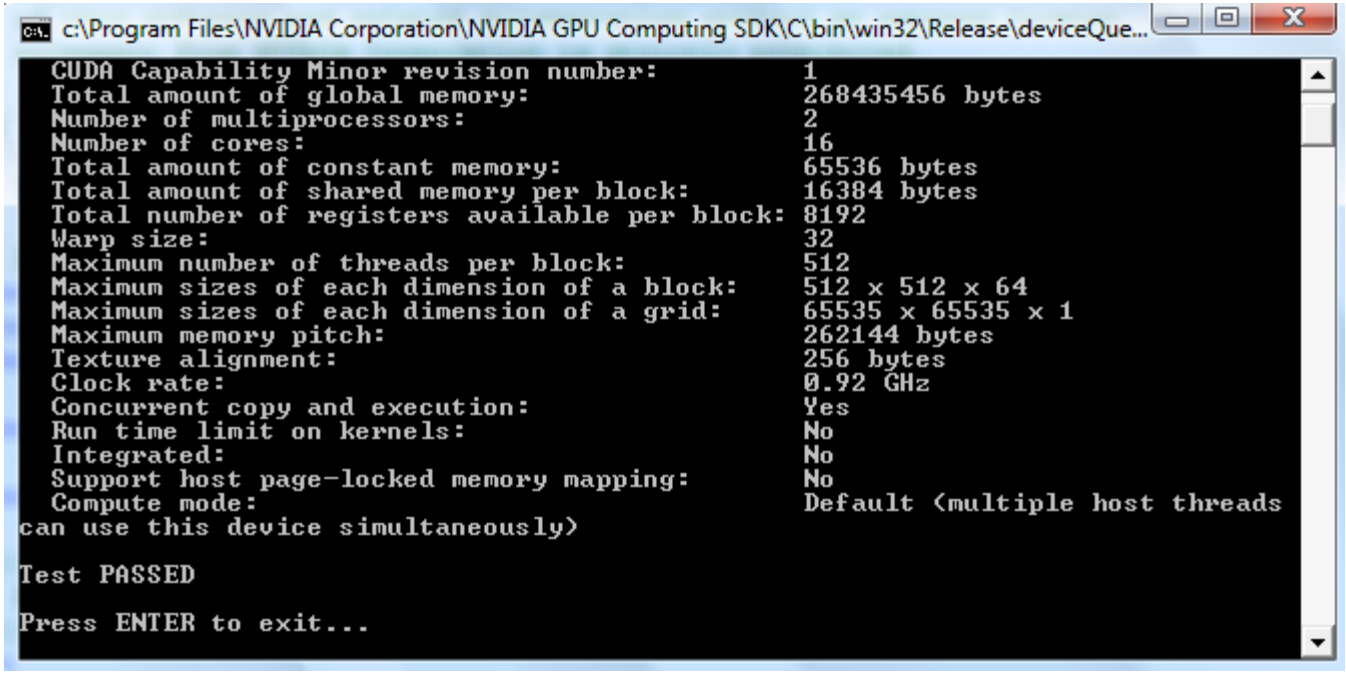






“deviceQueryDrv”

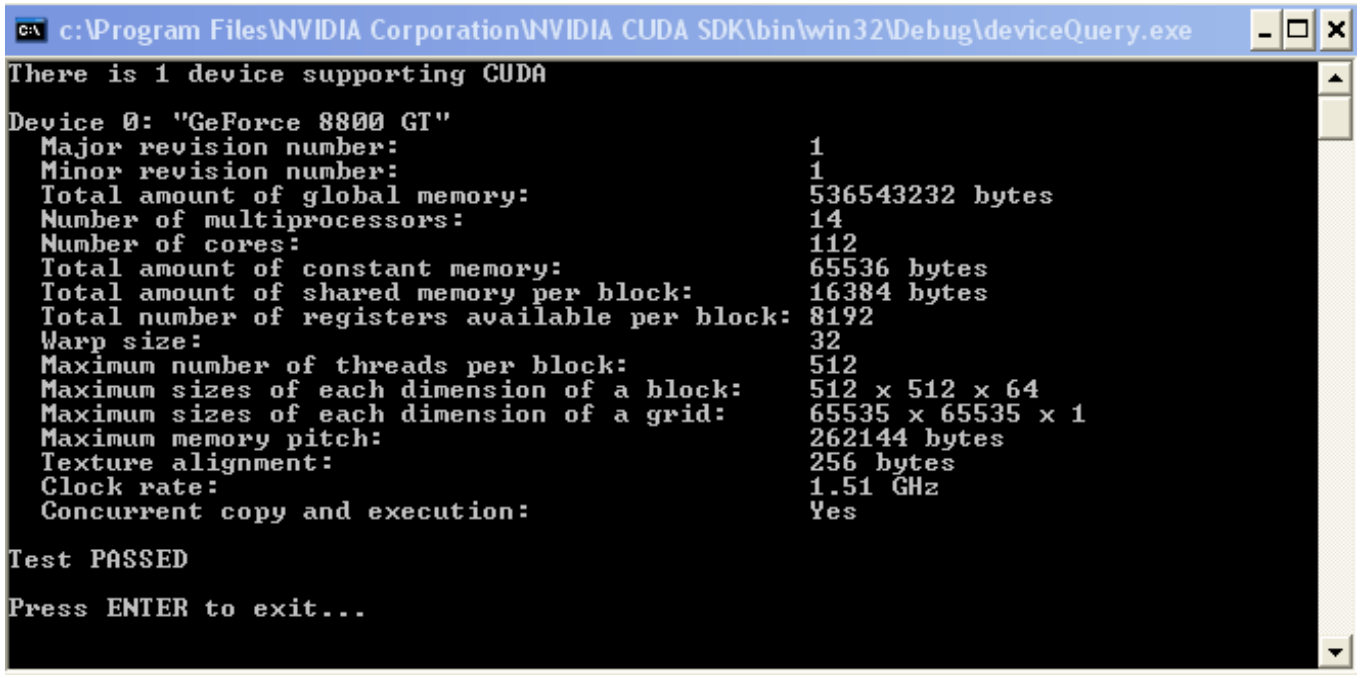
•Para saber que capacidades tiene nuestra tarjeta de video:



Resultado con una tarjeta NVIDIA GeForce 8400 GS

“deviceQueryDrv”

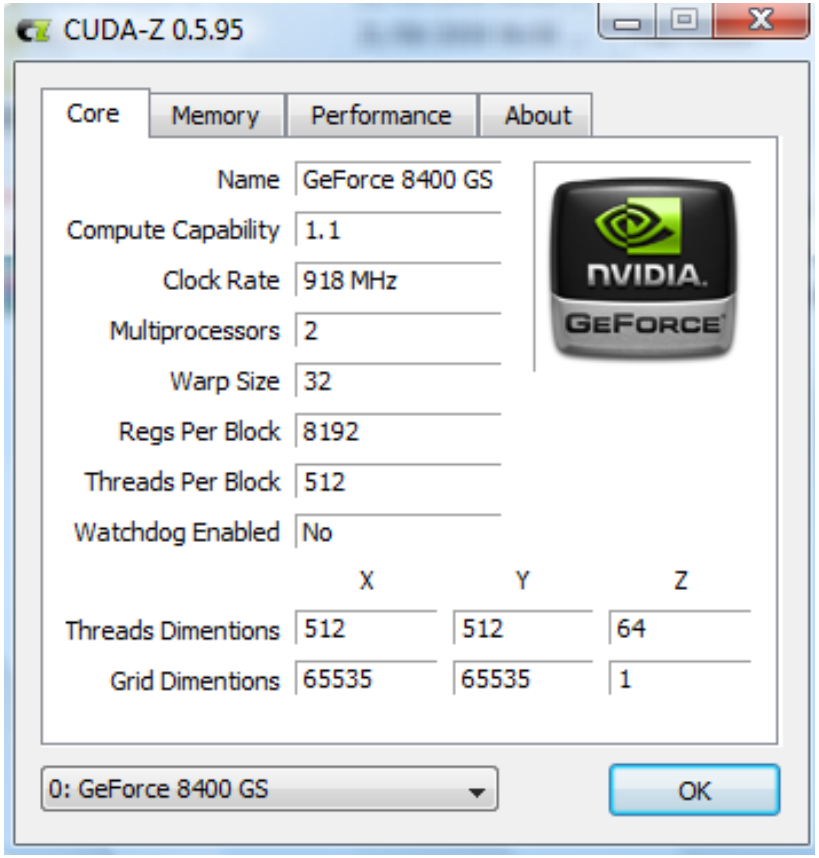
•Para saber que capacidades tiene nuestra tarjeta de video:

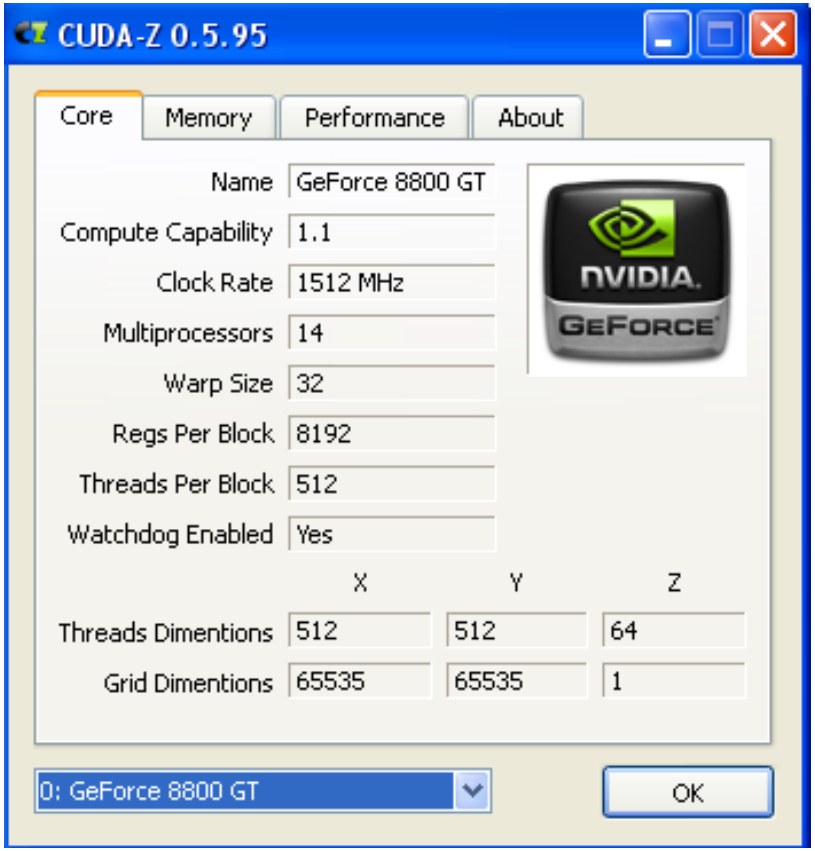


Resultado con una tarjeta NVIDIA GeForce 8800 GT

CUDA-Z

- GeForce 8400 GS & GeForce 8800 GT

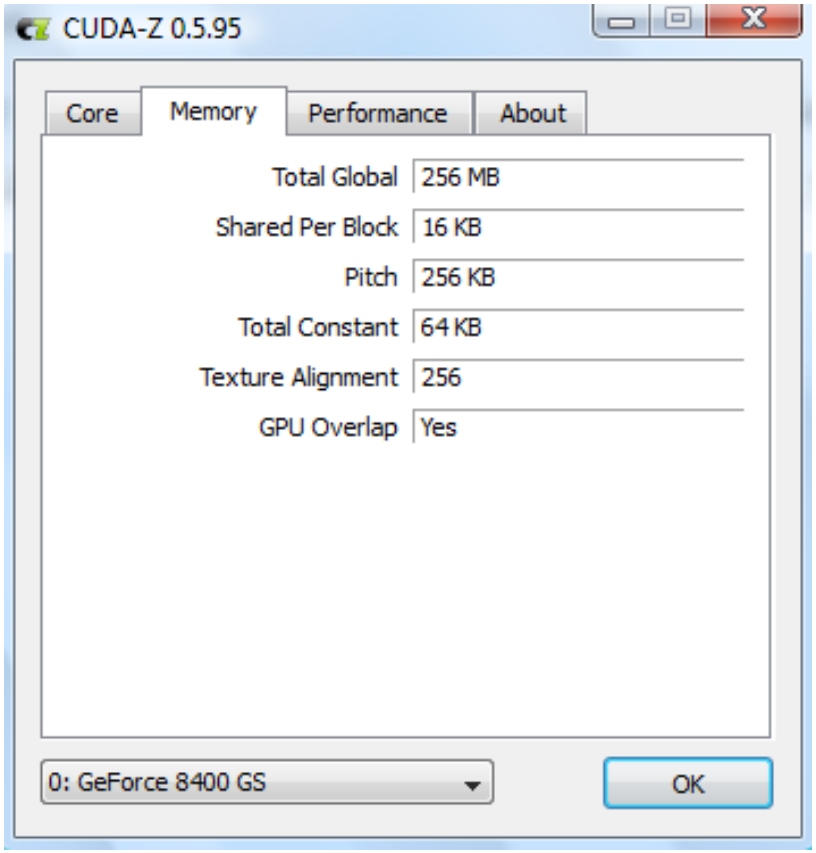


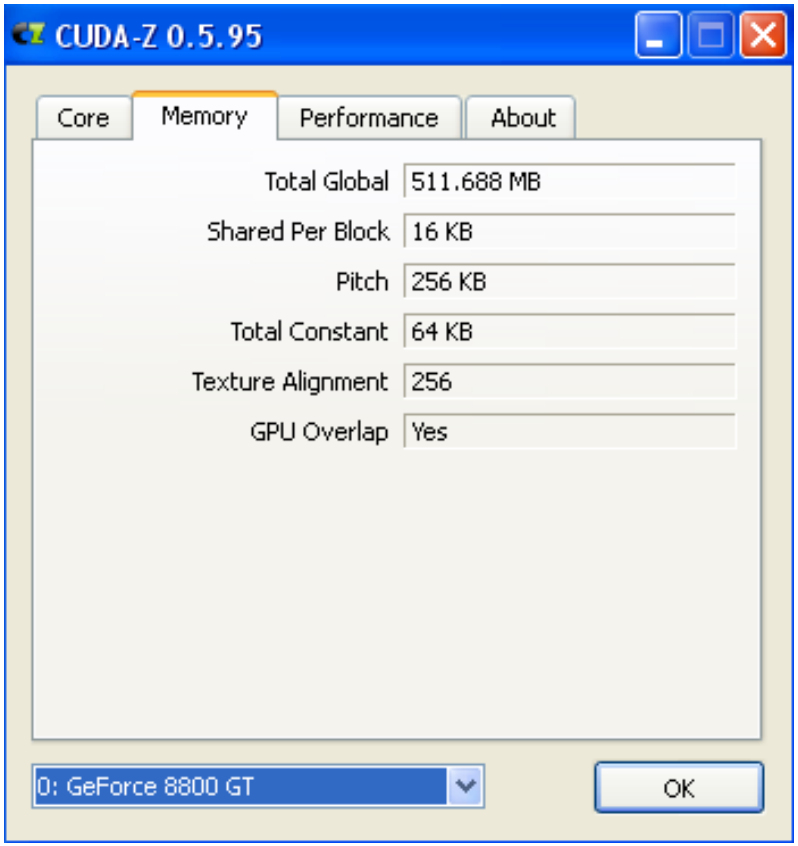


[http://cuda-z.sourceforge.net/]

CUDA-Z

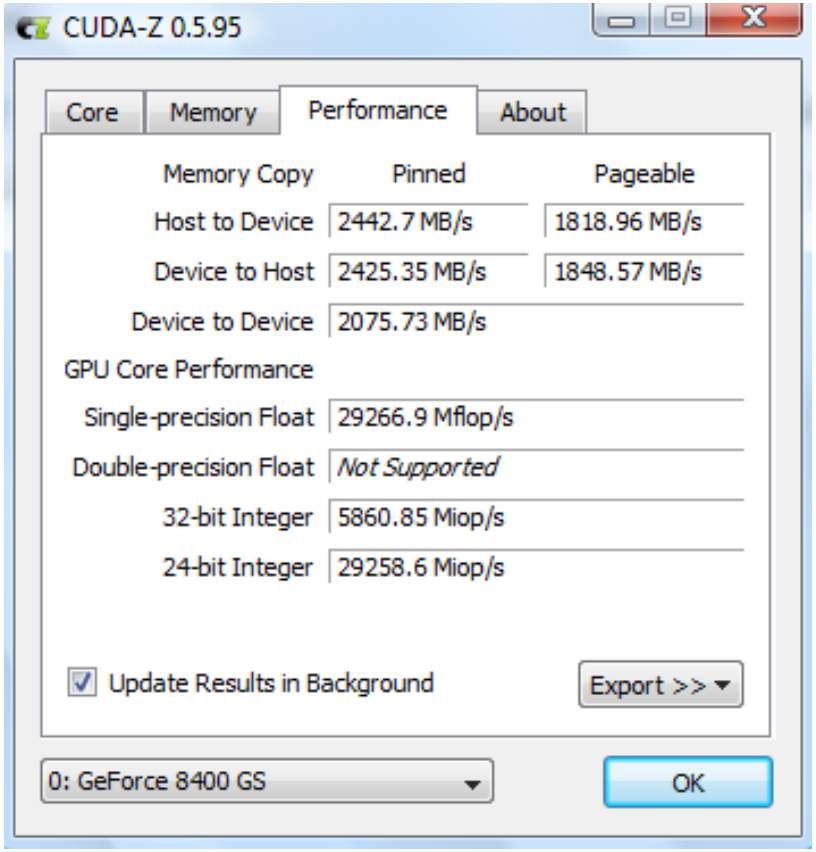
- GeForce 8400 GS & GeForce 8800 GT

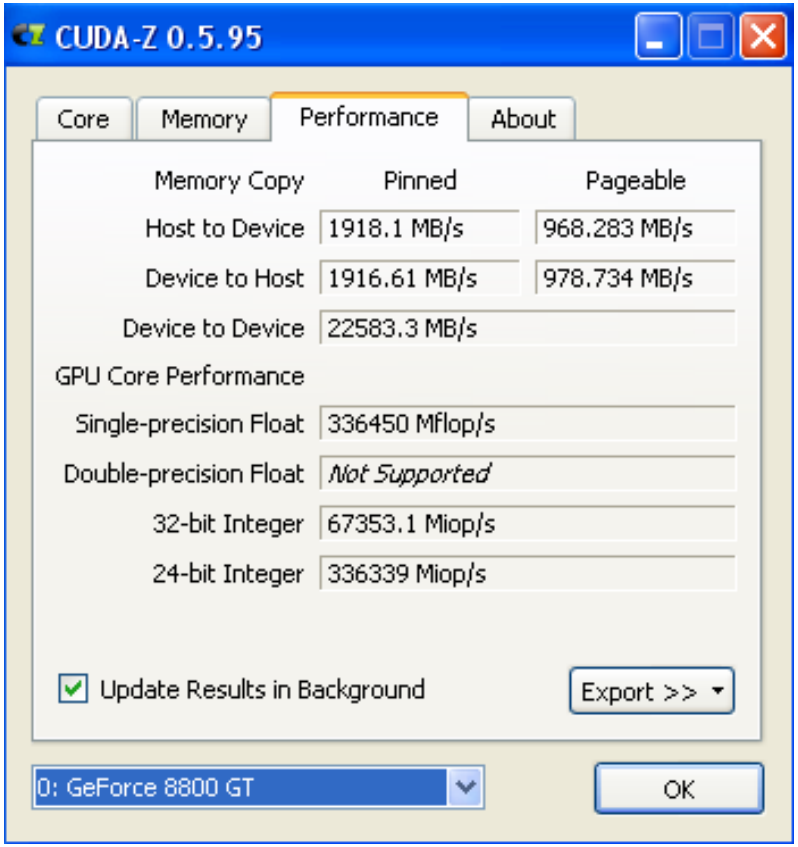




CUDA-Z

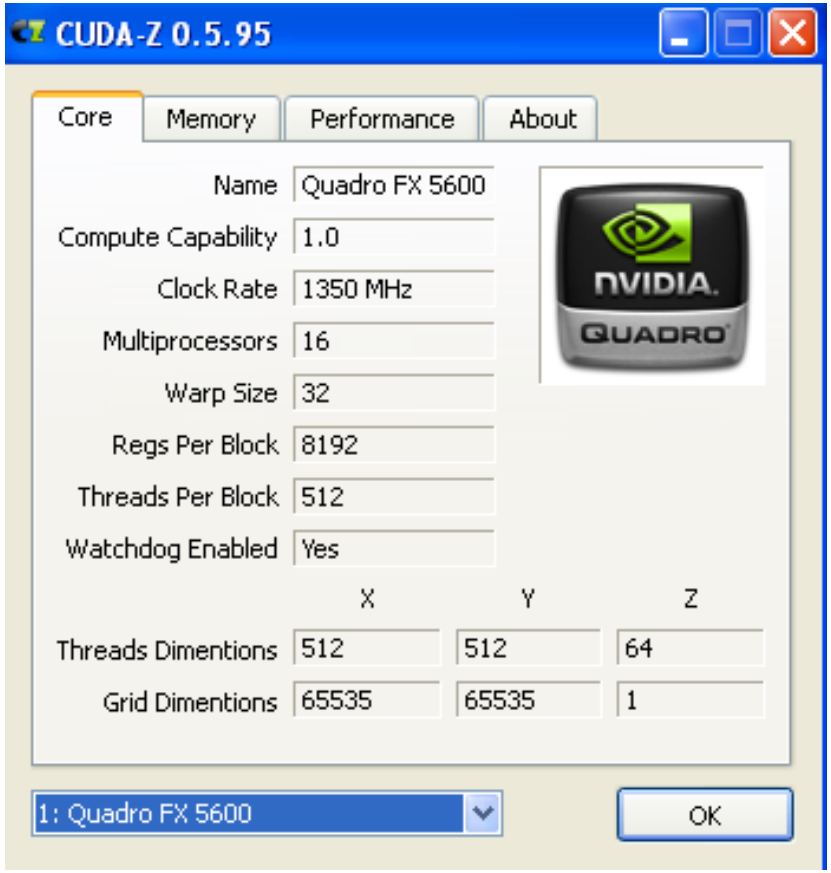
- GeForce 8400 GS & GeForce 8800 GT

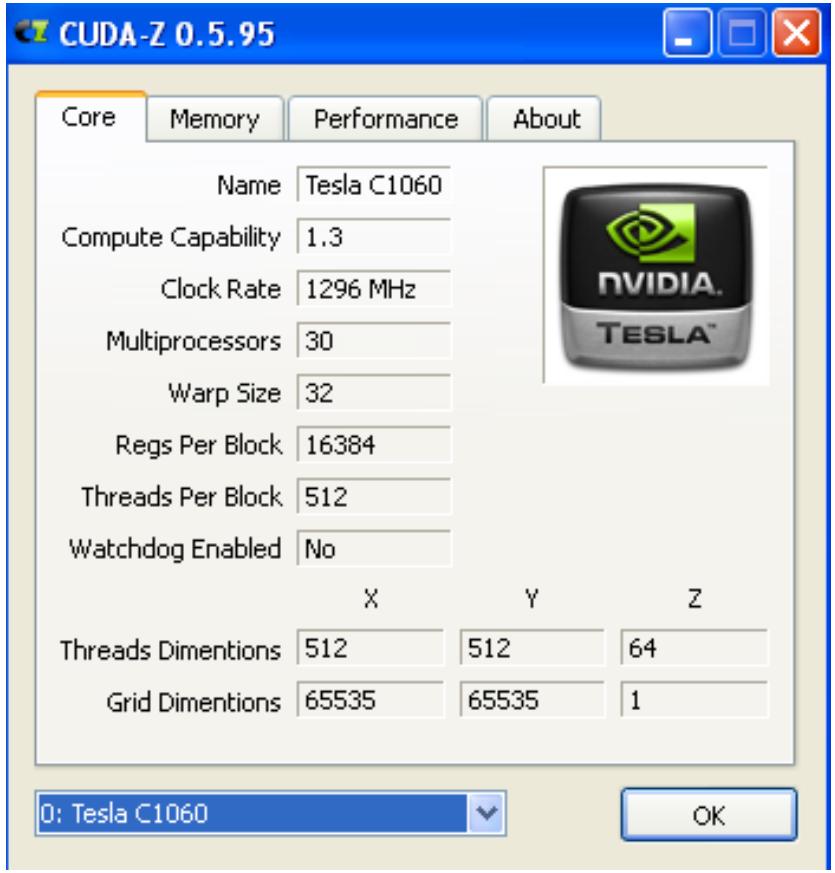




CUDA-Z

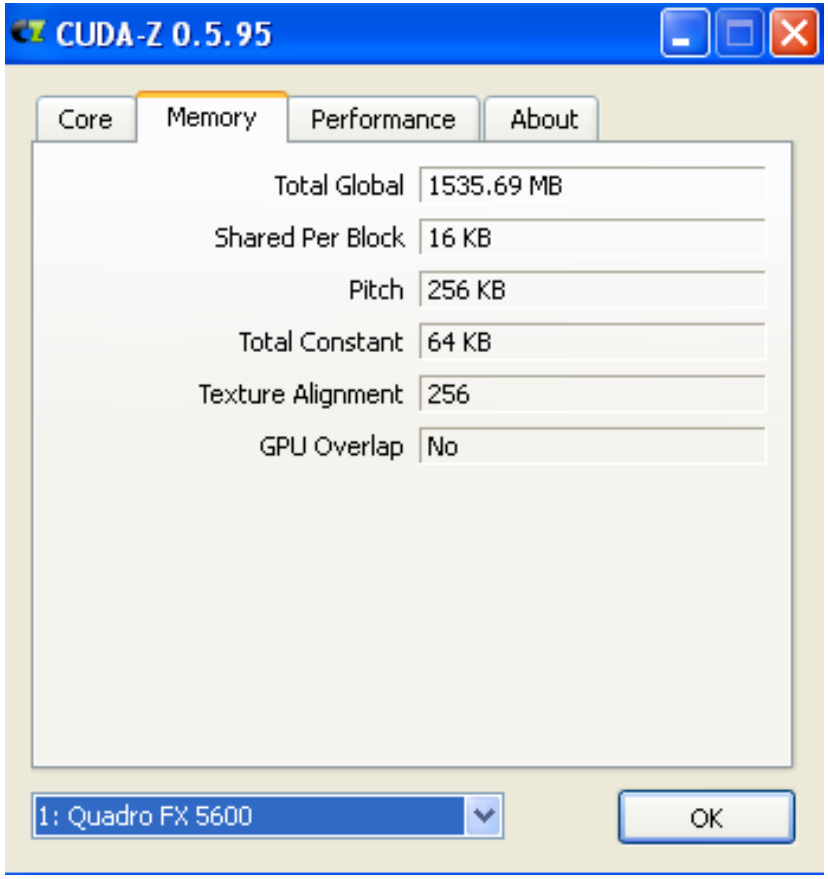
- Quadro FX 5600 & Tesla C1060

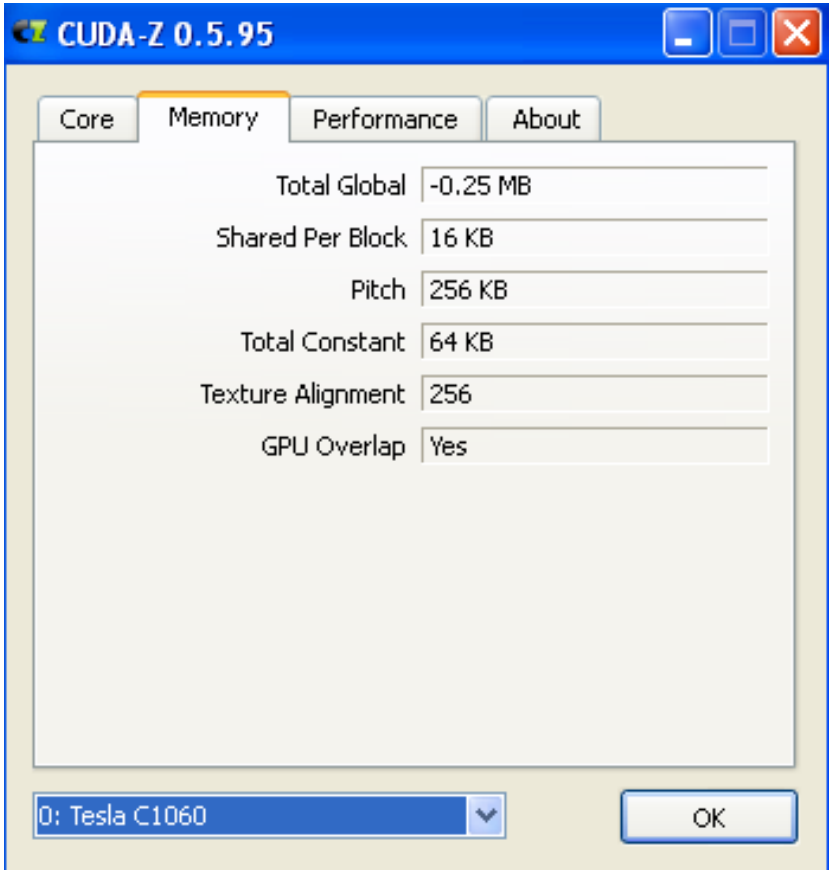




CUDA-Z

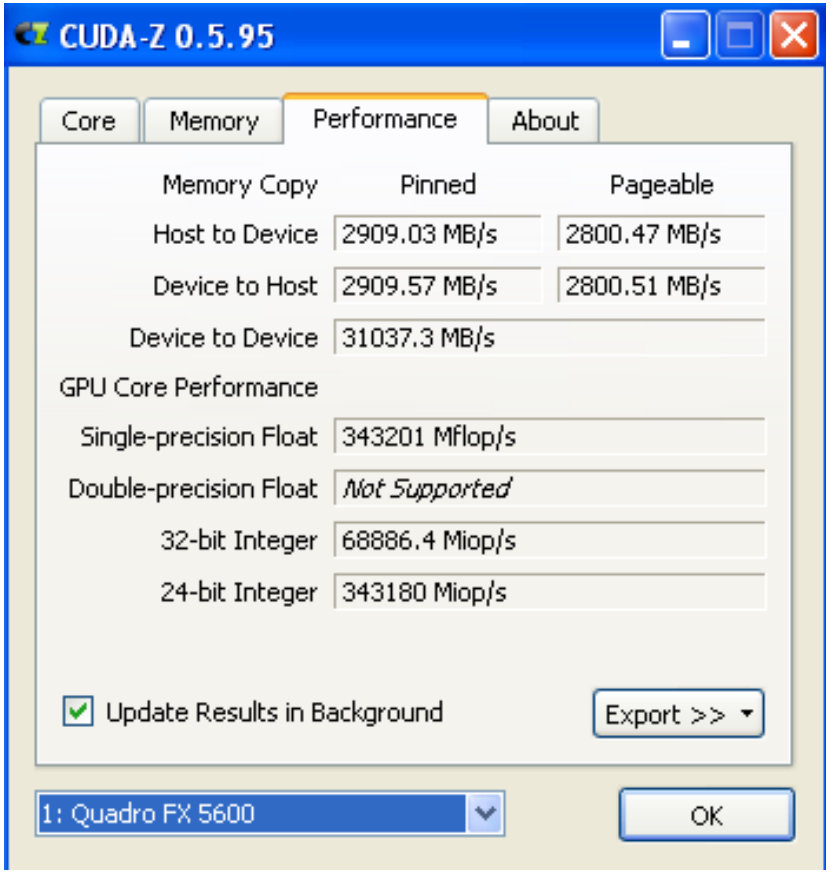
- Quadro FX 5600 & Tesla C1060

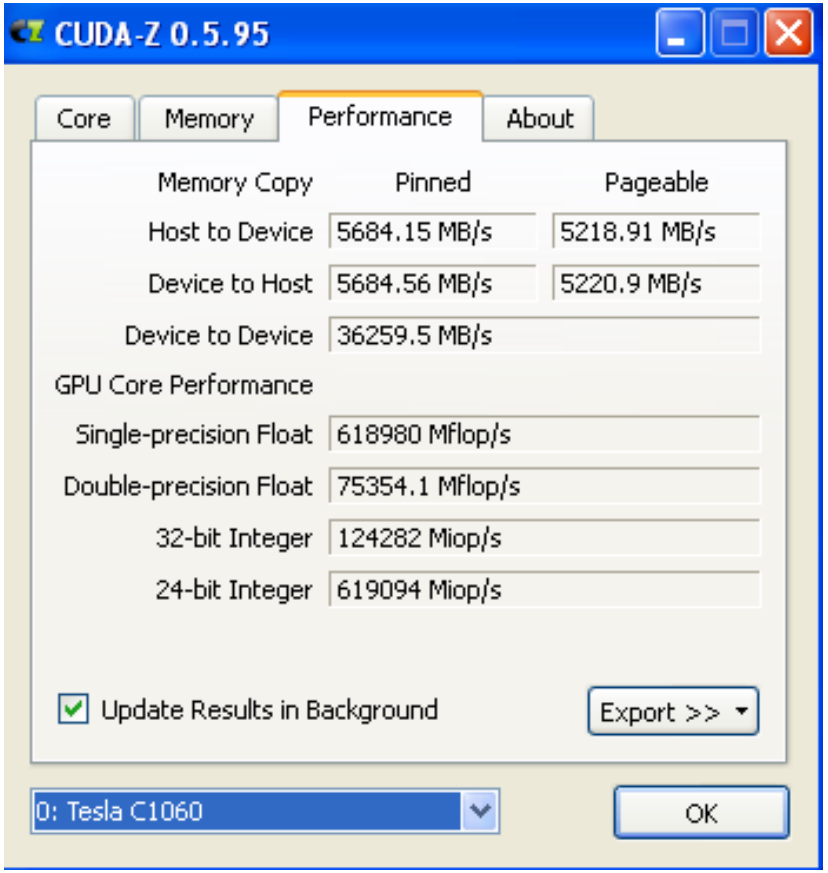




CUDA-Z

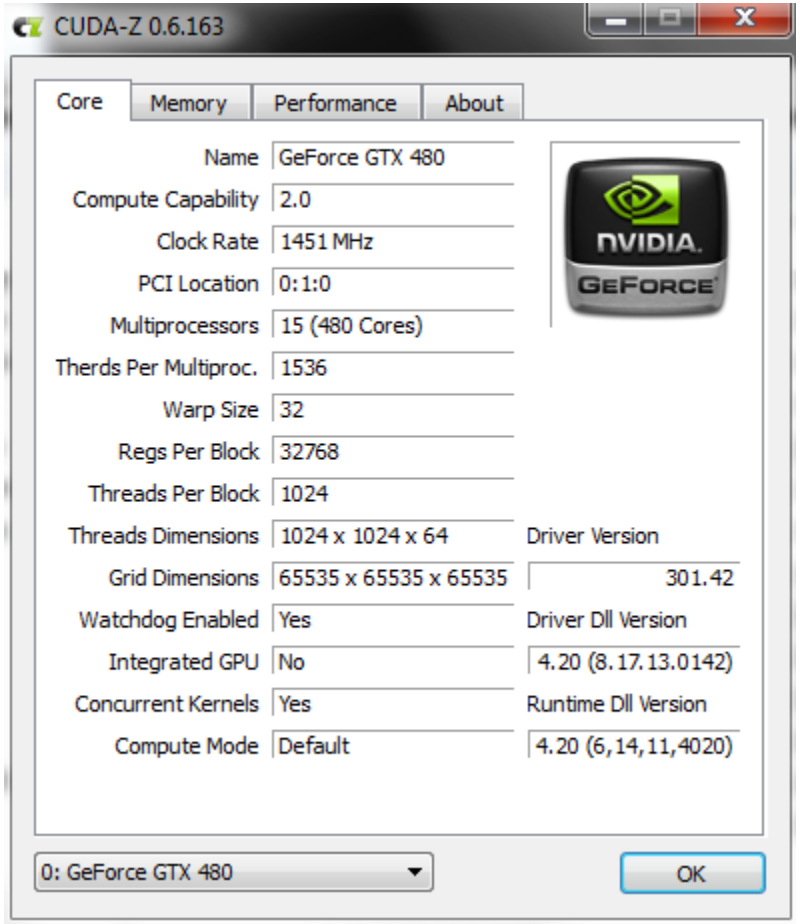
- Quadro FX 5600 & Tesla C1060

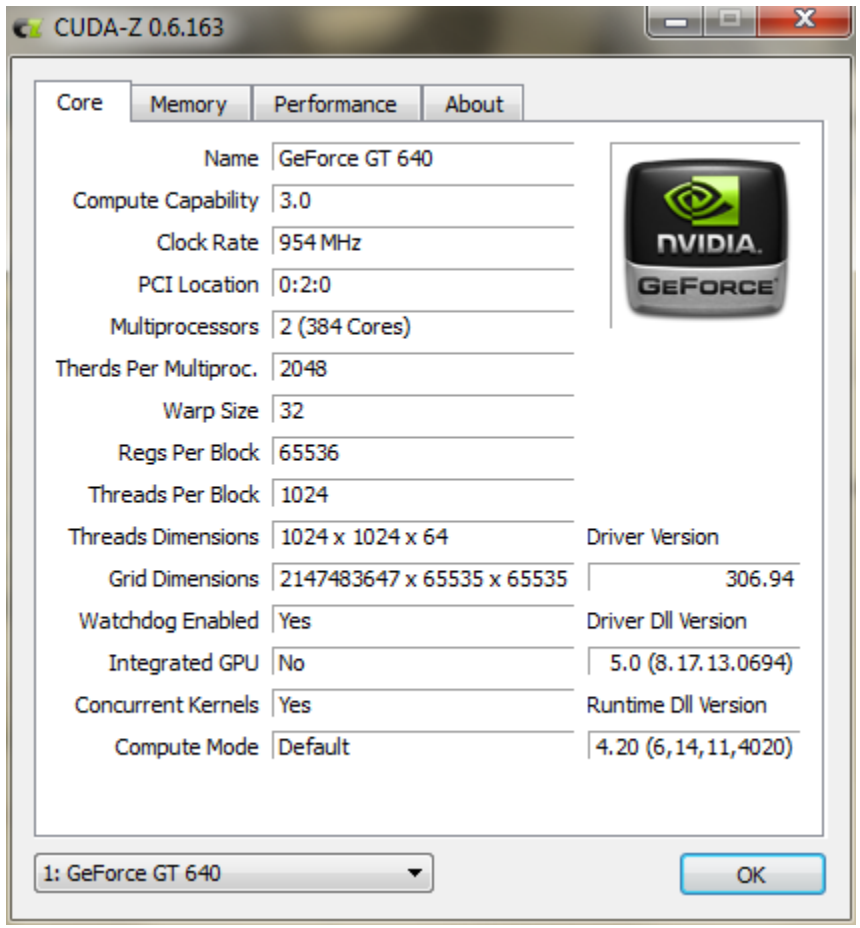




CUDA-Z

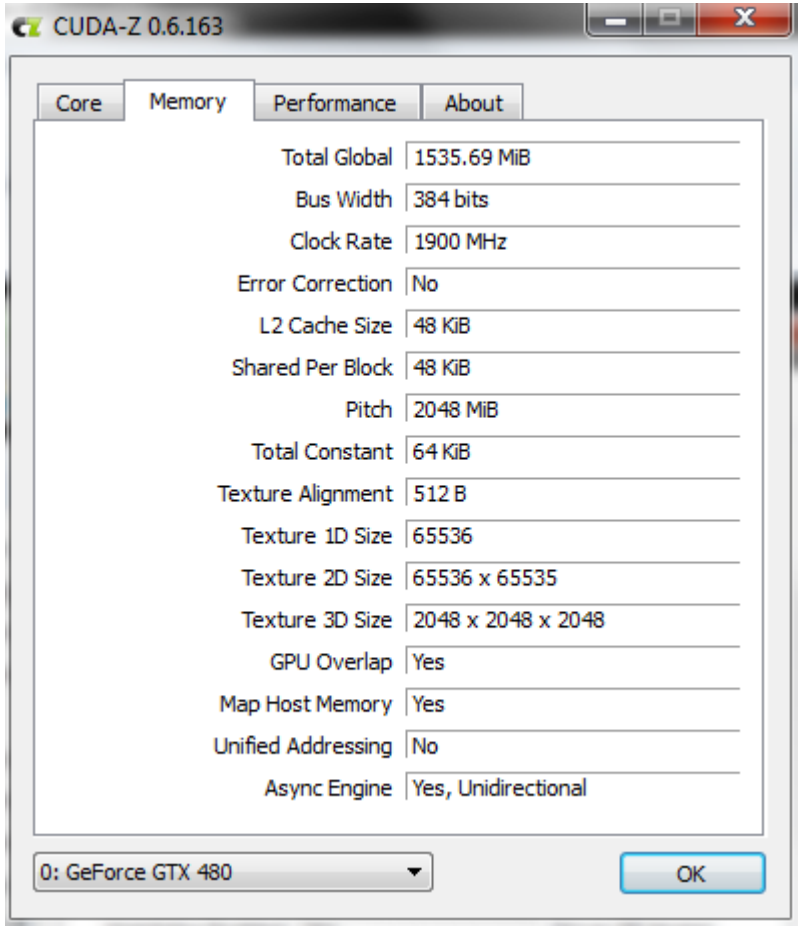
- GeForce GTX 480 & GeForce GT 640

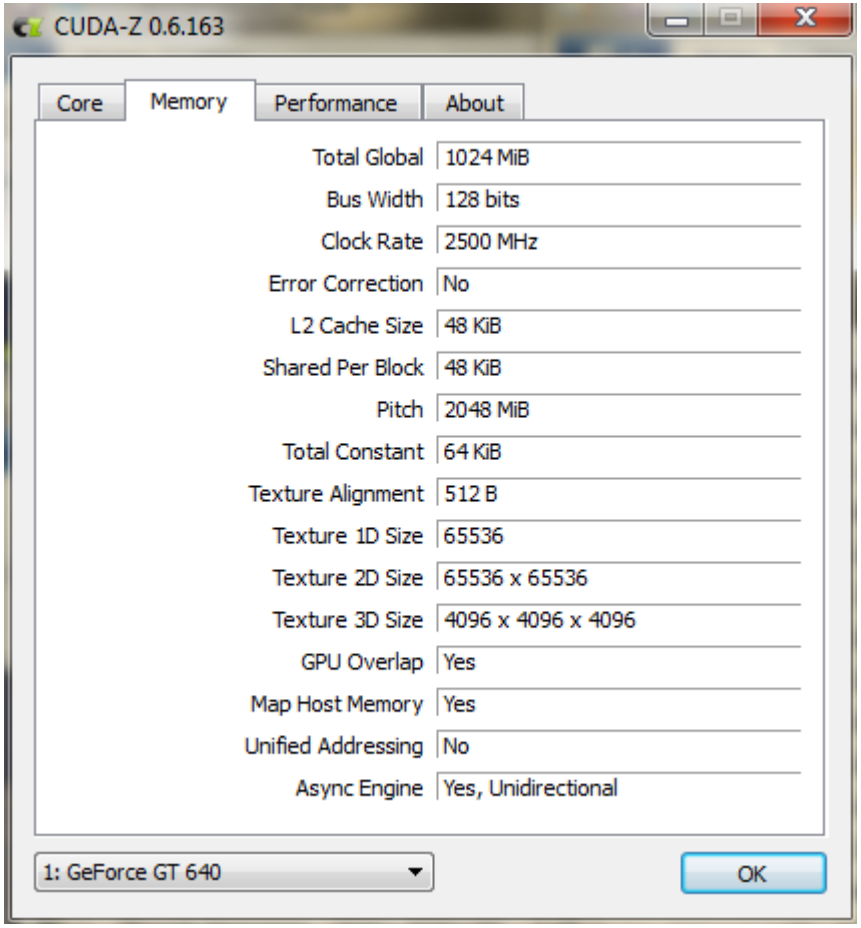




CUDA-Z

- GeForce GTX 480 & GeForce GT 640





CUDA-Z

- GeForce GTX 480 & GeForce GT 640