



Tema 1.

Programación Paralela y sus aplicaciones

U.A.Q. Fac. de Informática

Dra. Sandra Luz Canchola Magdaleno

Correo: sandra.canchola@uaq.mx

Dra. Reyna Moreno Beltrán

Correo: reyna.moreno@uaq.mx

Introducción

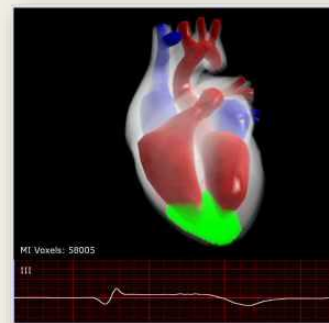
- Existen problemas que requieren alto poder de procesamiento.



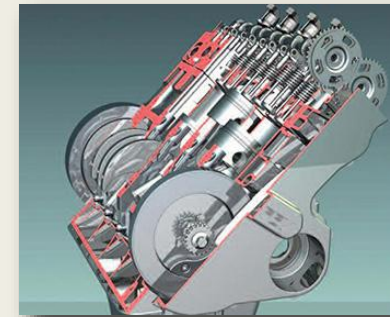
Recuperación de
Información



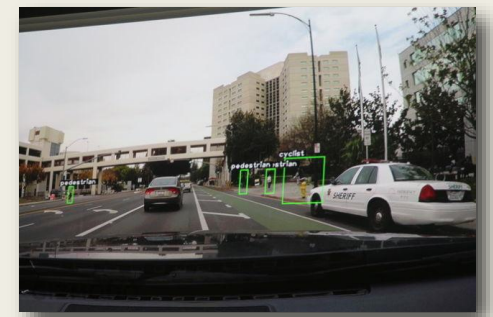
Aplicaciones
financieras



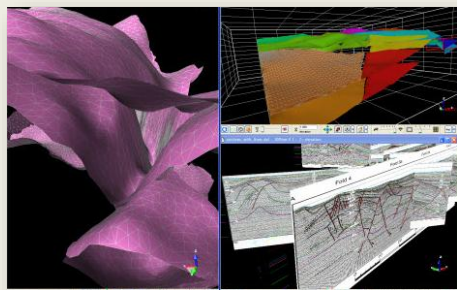
Imágenes médicas
y biofísicas



CAD/CAM/CAE



Tecnologías de
manejo



Energía y
exploración
petrolera



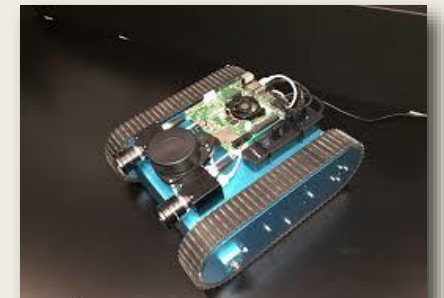
Creación de
Contenidos
Digitales



Modelado y
predicción
de clima



Video juegos



Vehículos no
tripulados

Introducción

- Se pueden realizar diversas plataformas para programación



Clusters



Servidores



Computadoras de escritorio



Computadoras portátiles



Tabletas & Teléfonos inteligentes



Sistemas embebidos

Computación paralela

- Es el uso de computadoras paralelas para reducir el tiempo de ejecución necesario para resolver un problema computacional.
- Ahora es una forma estandar para resolver problemas en áreas como modelación de clima, dinámica molecular, evolución de galaxias,.....



Galaxy Formation



Planetary Movments



Climate Change



Rush Hour Traffic



Plate Tectonics



Weather



Auto Assembly

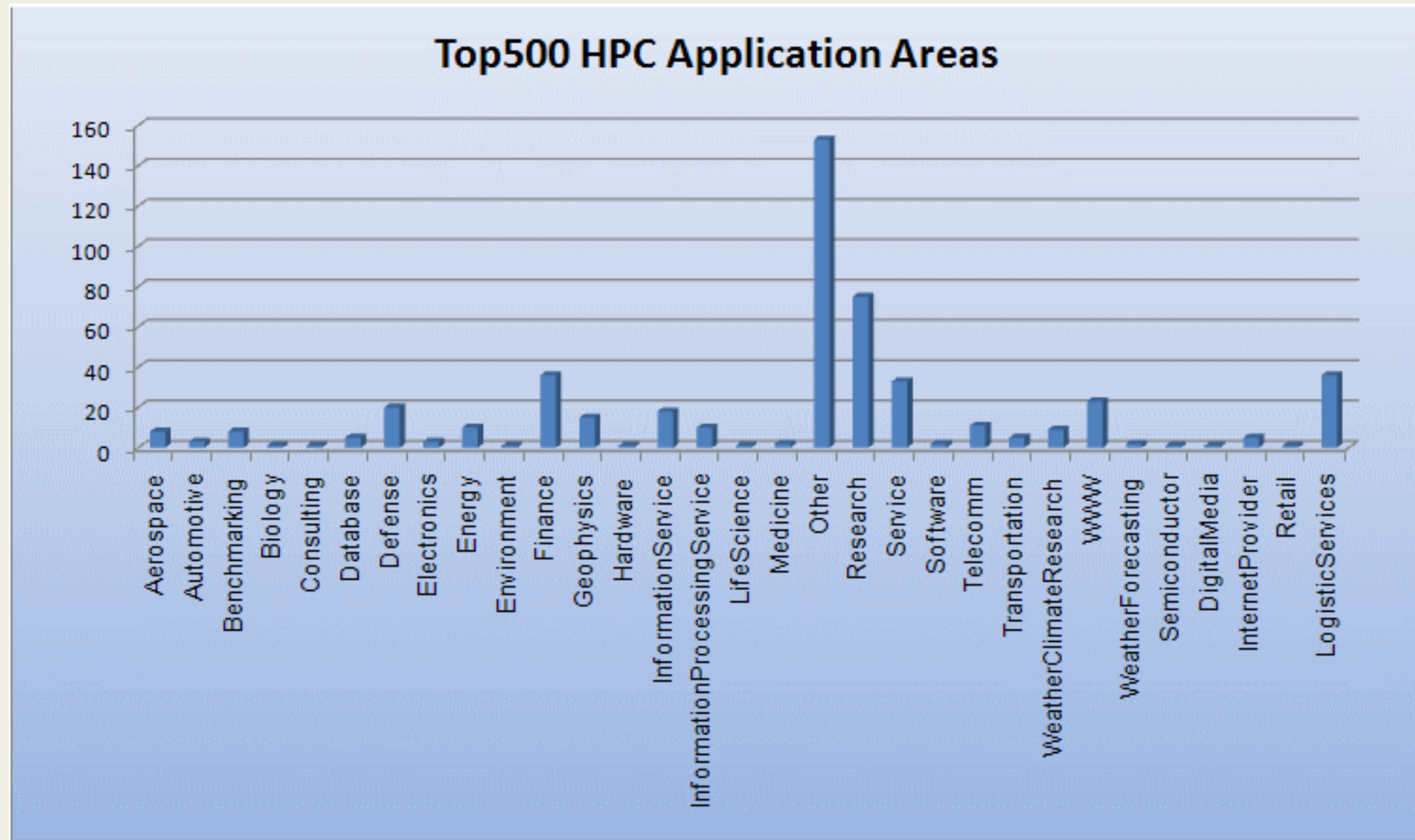


Jet Construction



Drive-thru Lunch

Computacion paralela

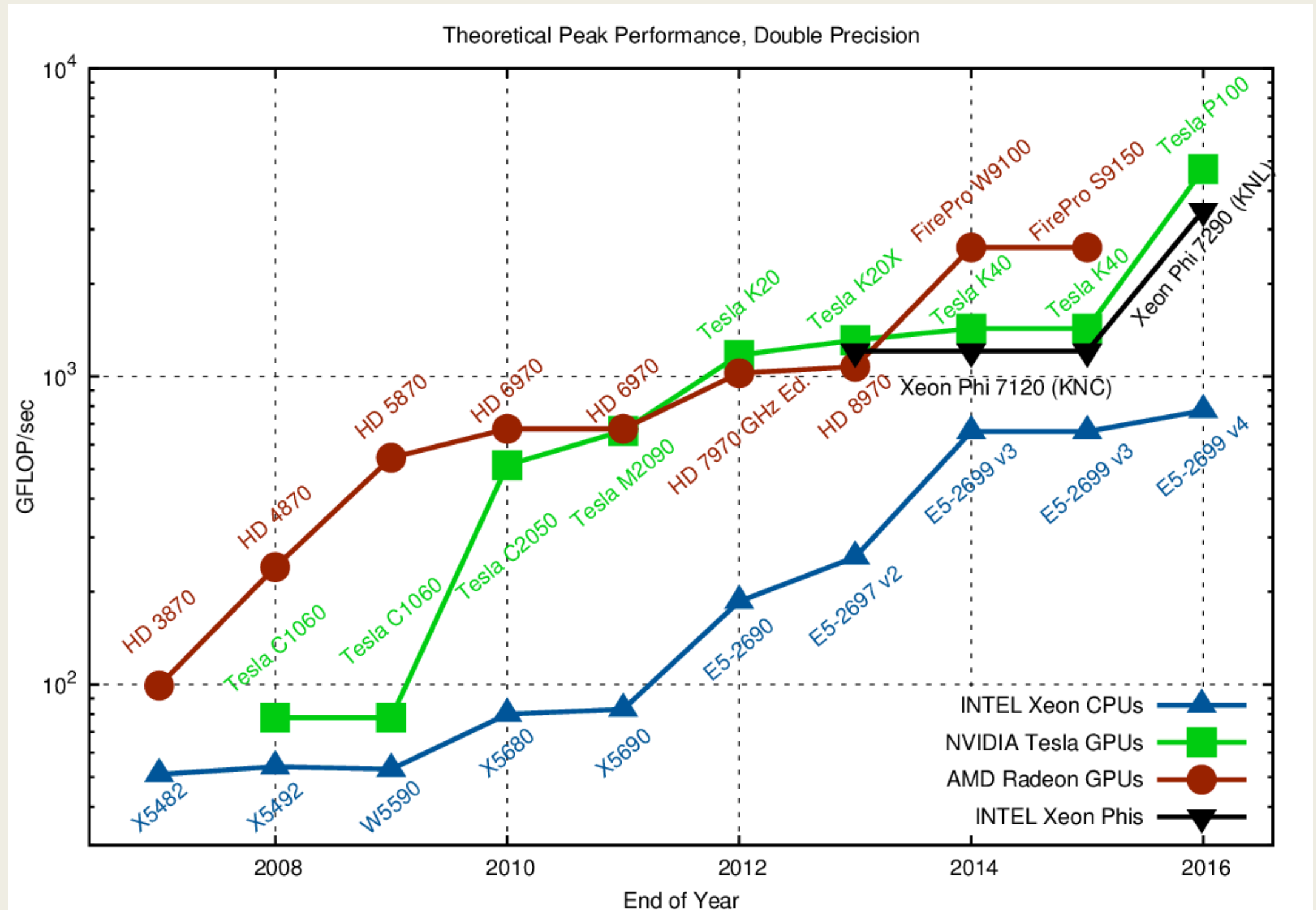


Tecnologías de GPGPU

(General-purpose computing on graphics processing units)

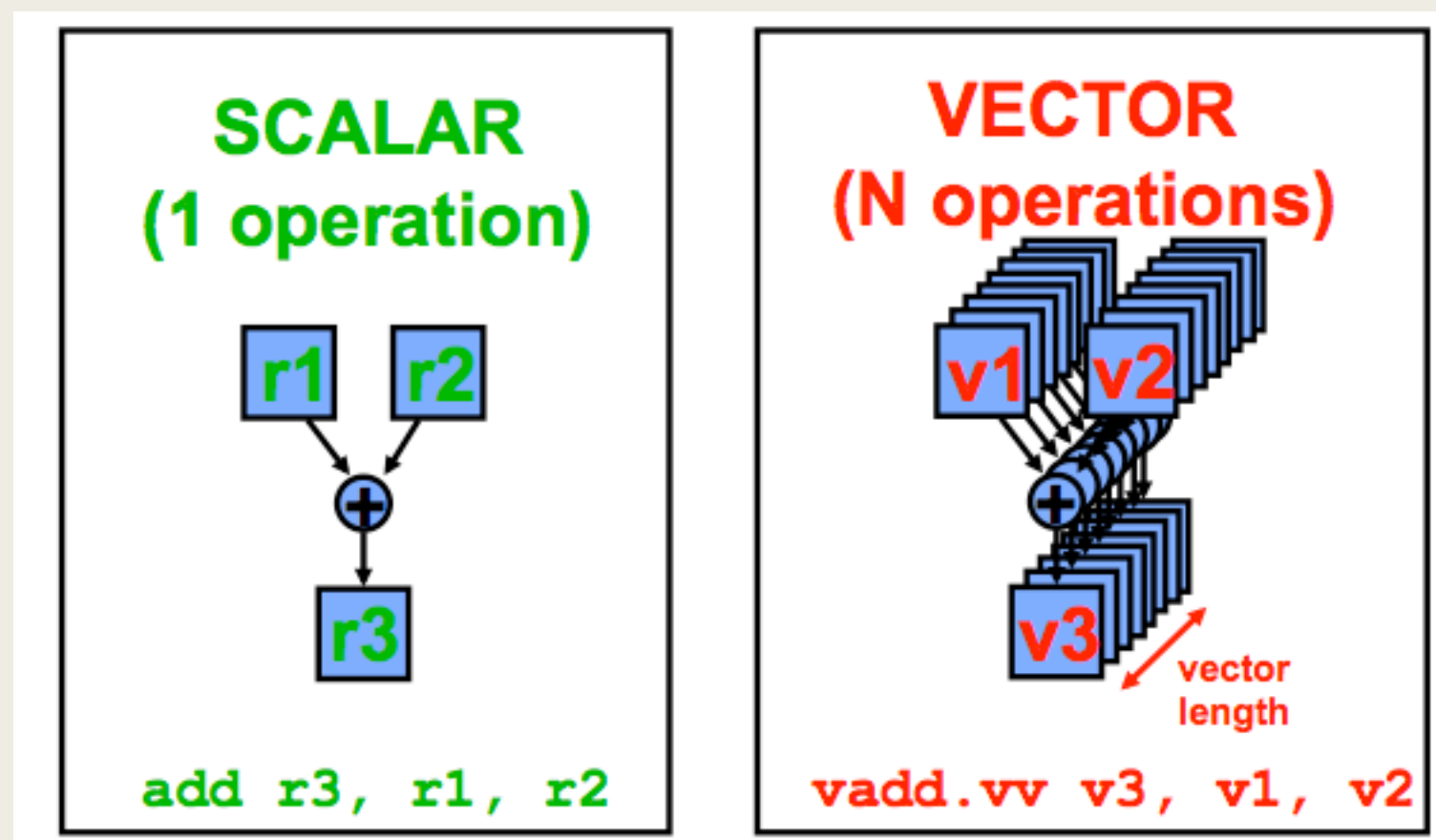
- Unidades de procesamiento gráfico de propósito general.
- Procesadores vectoriales.
- Fabricantes: nVidia, ATI, Intel...
- La idea es aprovechar las unidades aritméticas y lógicas de los GPU's para hacer computaciones de alto rendimiento.

Tecnologías de GPGPU



GPGPU - Procesadores vectoriales

- Procesador escalar opera sobre números sencillos (escalares).
- Procesadores vectoriales operan en vectores de números.



GPGPU - Procesadores vectoriales

Beneficios de los procesadores vectoriales

- Compacto: una simple instrucción define N operaciones.
 - *Ademas reduce la frecuencia de saltos.*
- Paralelo: N operaciones son paralelas en datos.
 - *No hay dependencias.*
 - *No se requiere de HW para detectar paralelismo.*
 - *Puede ejecutar en paralelo asumiento N flujos de datos paralelos.*
- Usa patrones de acceso a memoria continua.

CPU vs GPU

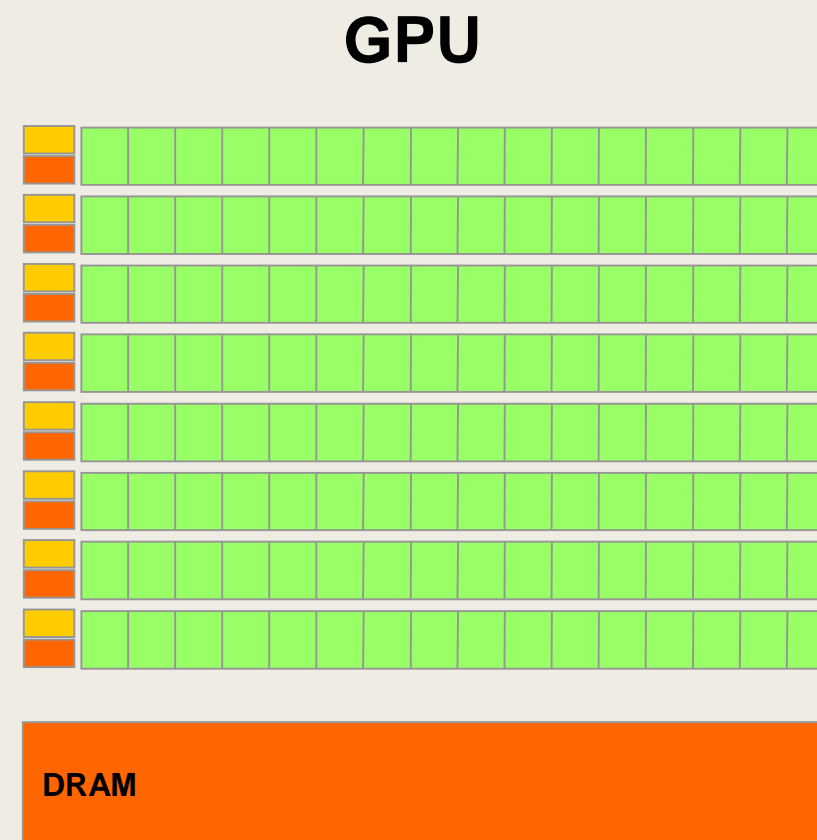
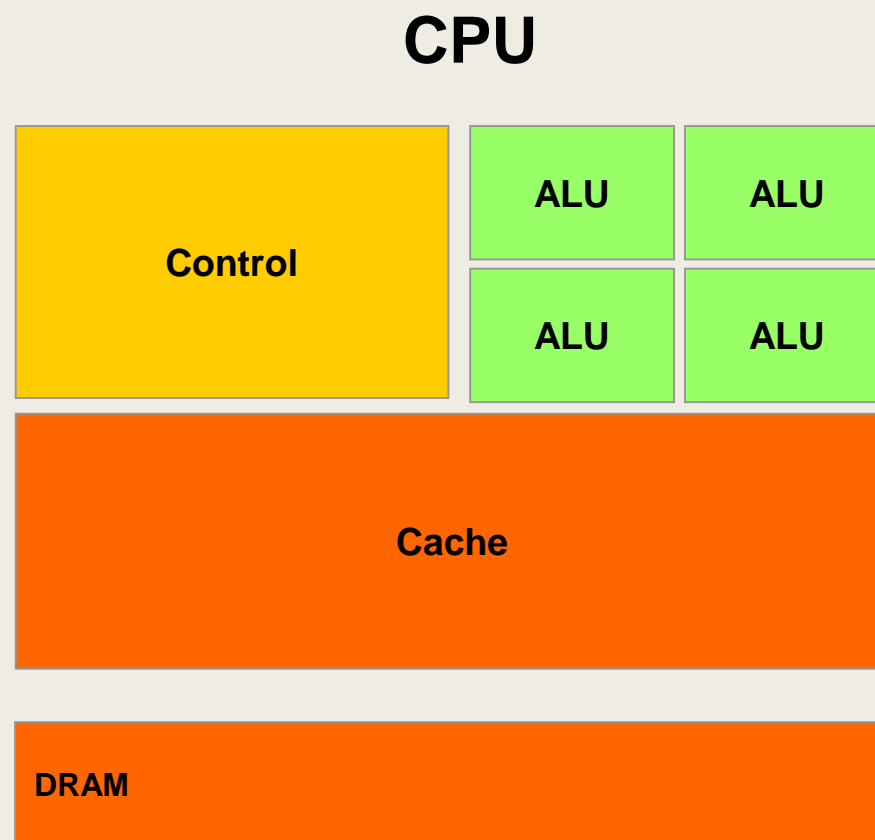
CPU

- Baja latencia de memoria.
- Acceso aleatorio.
- 20GB/s ancho de banda.
- 0.1 Tflop.
- 1 Gflop/watt
- Procesador escalar
- Modelo de programación altamente conocido.

GPU

- Gran ancho de banda.
- Acceso secuencial.
- 100GB/s ancho de banda.
- 1 Tflop.
- 10 Gflop/watt
- Procesador vectorial
- Modelo de programación **muy** poco conocido.

CPU vs GPU



Lista del top500 de super computadoras

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	El Capitan - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,039,616	1,742.00	2,746.38	29,581
2	Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	Aurora - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	Eagle - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
5	HPC6 - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, RHEL 8.9, HPE Eni S.p.A. Italy	3,143,520	477.90	606.97	8,461
6	Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
7	Alps - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Cray OS, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	2,121,600	434.90	574.84	7,124
8	LUMI - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107
9	Leonardo - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA Italy	1,824,768	241.20	306.31	7,494

November 2024

The 64th TOP500 List was published Nov. 19, 2024 in Atlanta, GA.

Computación paralela en GPU (ATI-AMD)

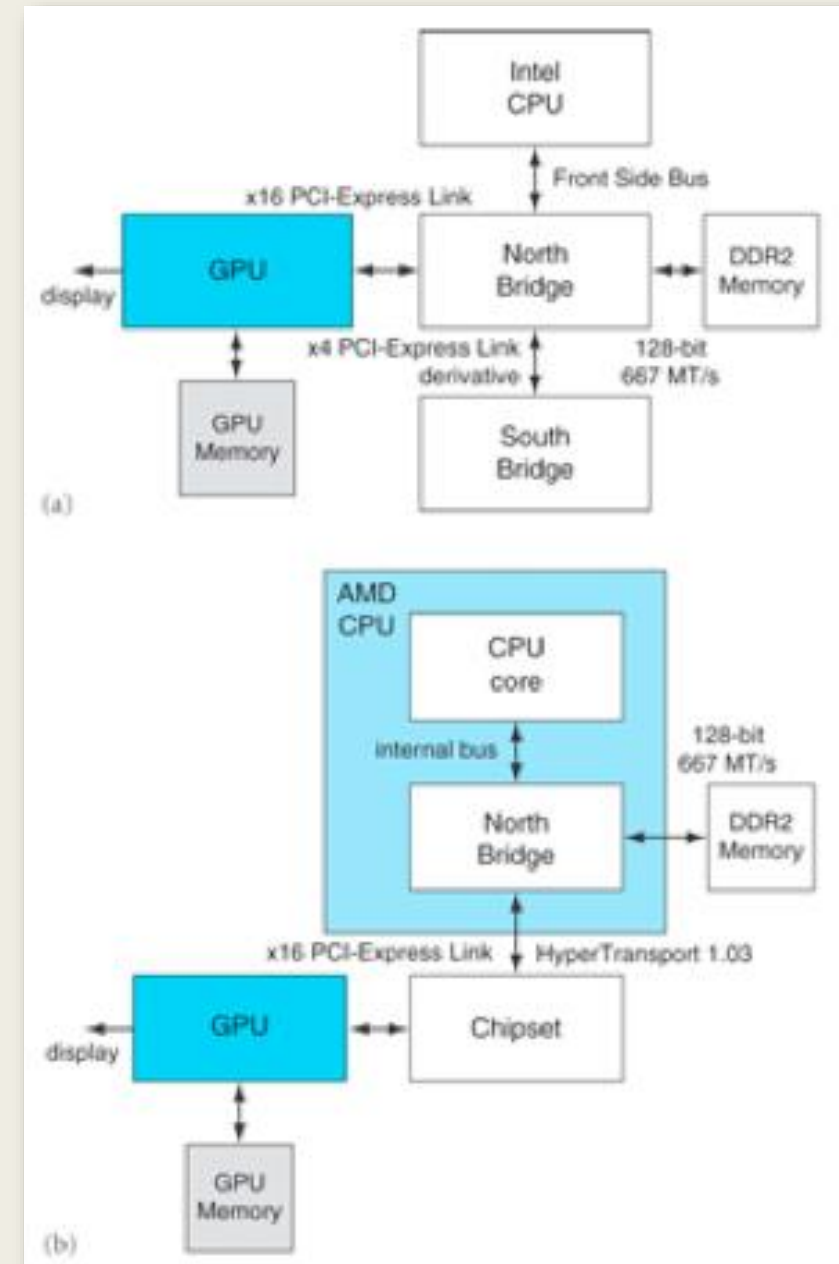
- Tarjetas: ATI Radeon Disponible en laptops, desktops, servidores y nodos para clusters

ATI Radeon



GPU en los servidores actuales

- Coprocesadores del CPU.
- Conexión PCIe (8GB/s) por direccion.
- Memoria independiente del GPU (gran ancho de banda local, hasta 100GB/s).



Configuraciones de HW

a)



b)



c)



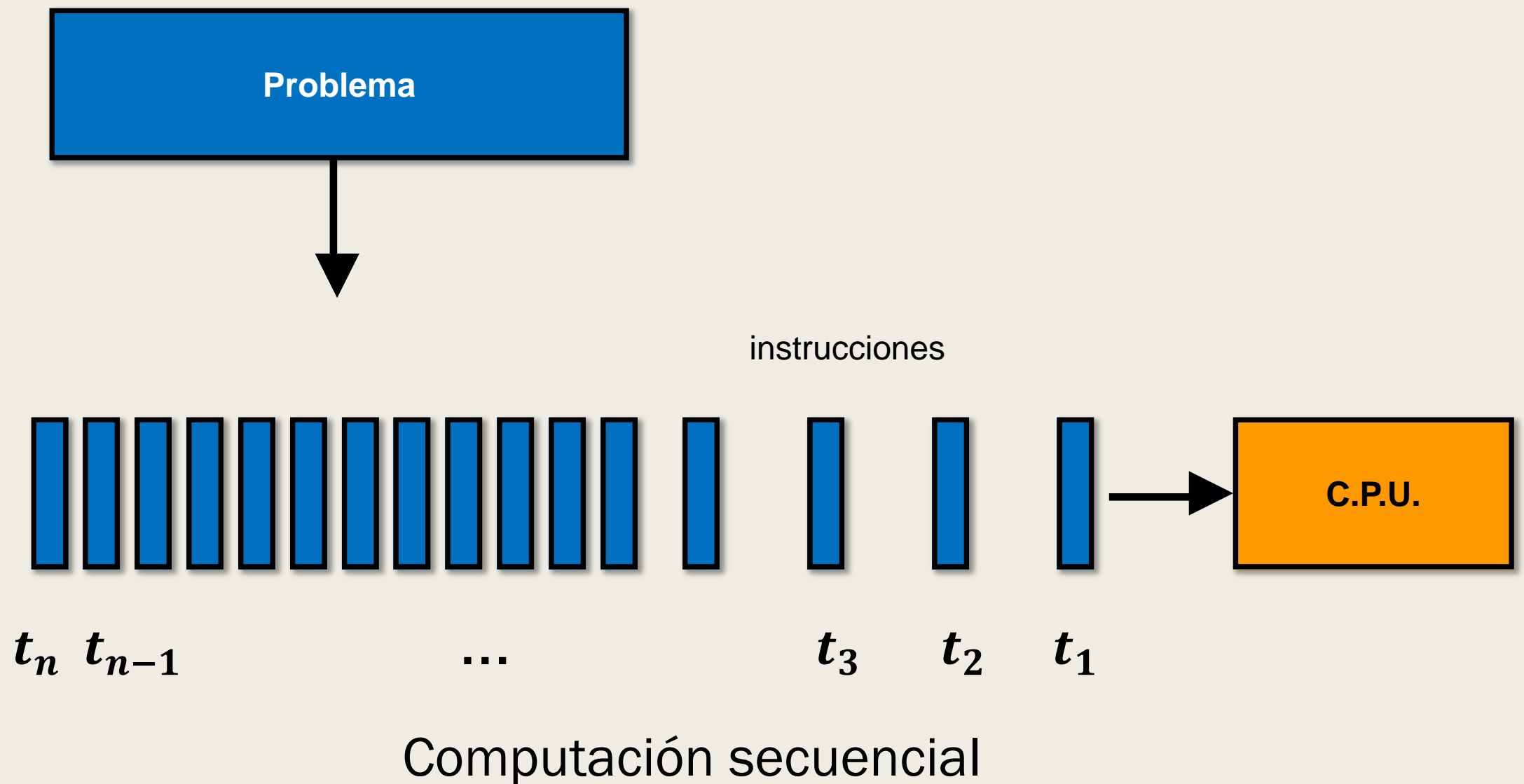
d)



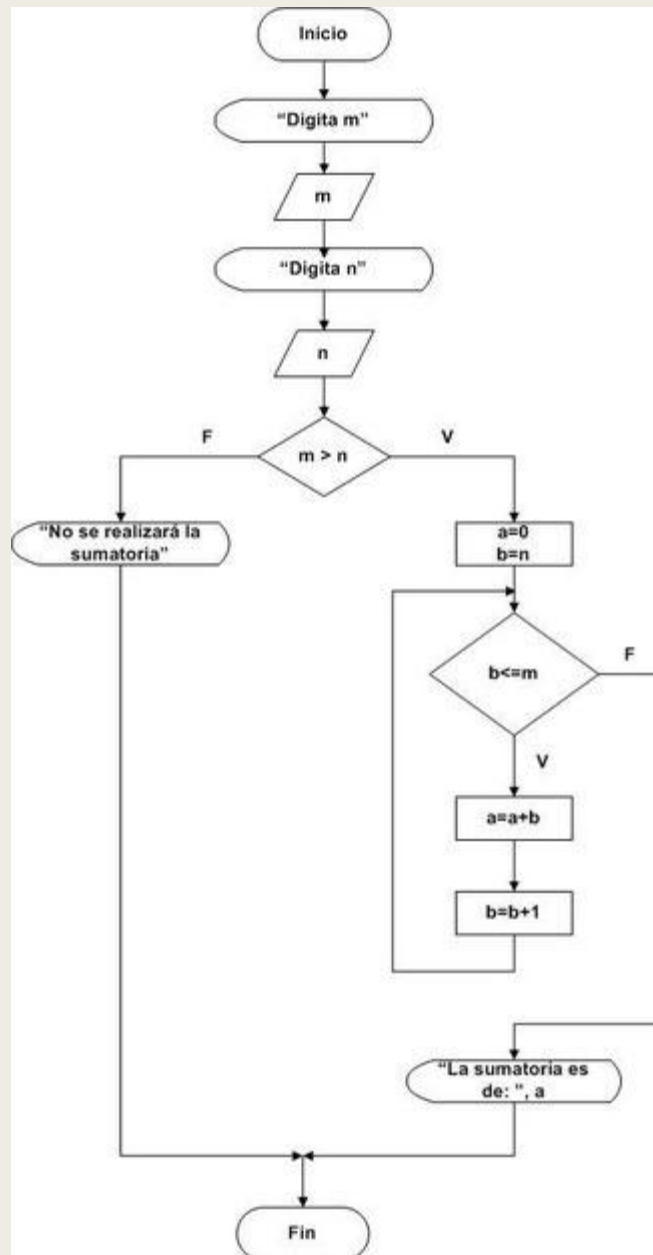
Computación paralela en GPU

- Lenguajes de programación:
 - *CUDA: C, C++ (nVidia) y CUDA Fortran (PGI)*
 - *OpenCL: Consorcio Kronos.*
 - *DirectCompute: C (Microsoft).*
 - *Brook+: C,C++ (Stanford University).*
 - *OpenACC (PGI, CRAY, CAPS y nVidia)*

Conceptos generales

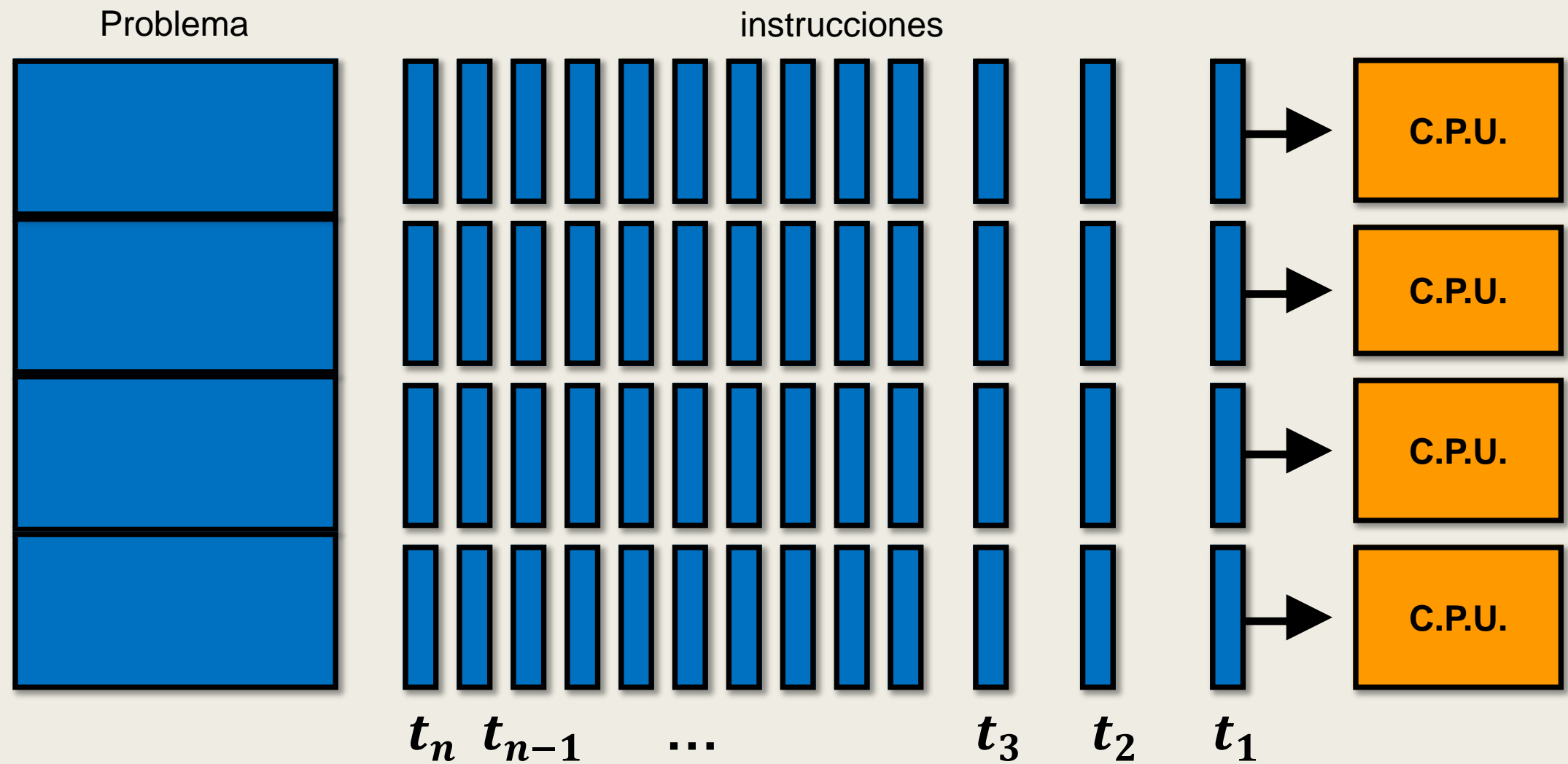


Conceptos generales



Computación secuencial

Conceptos generales



Computadora paralela

- Computadora con múltiples procesadores que soporta programación paralela. Hay dos categorías importantes de computadoras paralelas: multicomputadoras y multiprocesadores centralizados.
- Multicomputadora: es una computadora paralela construida por múltiples computadoras y una red de interconexión.
- Multiprocesador centralizado: (o SMP) es un sistema más integrado donde todos los CPU comparten el acceso a una memoria global.

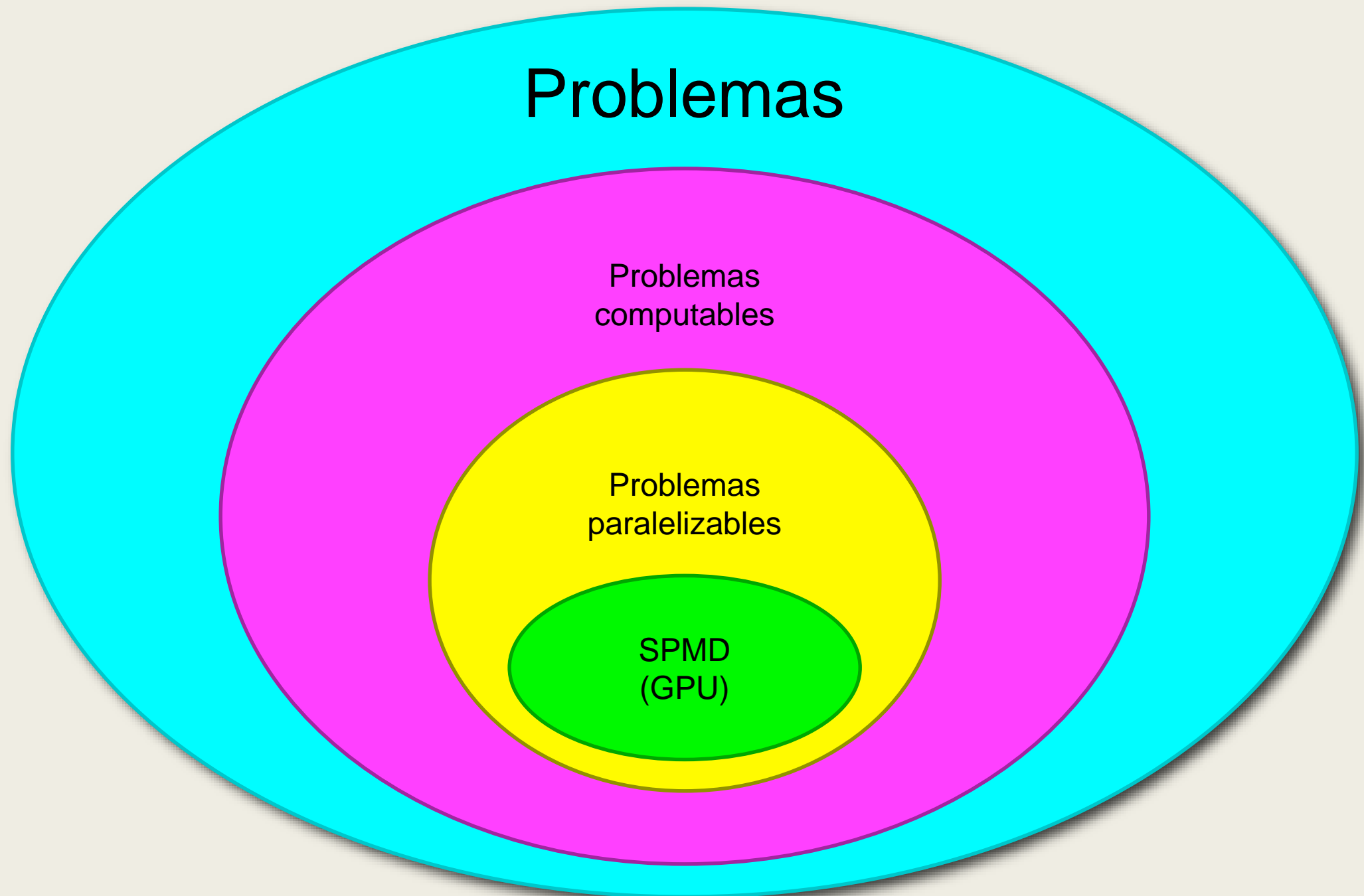
Programación paralela

- Es la programación en un lenguaje que permita indicar explícitamente como distintas partes de una computación pueden ejecutarse concurrentemente por diferentes procesadores.
 - *El modelo de programación paralela es una abstracción de la arquitectura de la computadora paralela.*
 - Memoria compartida: Multithreading (pthreads, OpenMP, CUDA, OpenCL).
 - Memoria distribuida: Paso de mensajes (MPI).

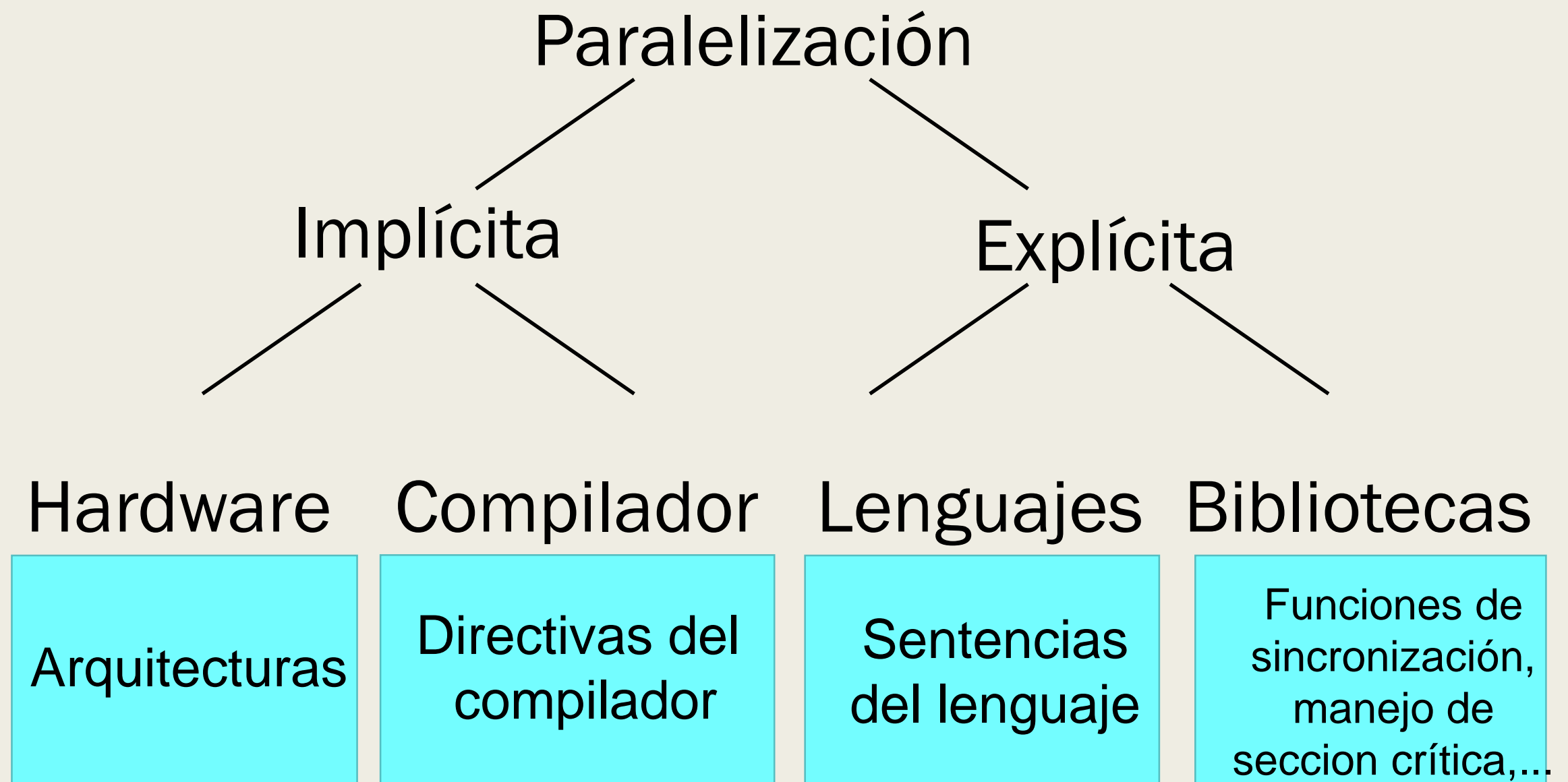
Programación paralela

- Detalles a considerar:
 - *Control*
 - *Synchronization*
 - *Communication*
- Los lenguajes de programación ofrecen diferentes formas de trabajar con éstas.

Modelo de programación



Paralelización



Acceleration of algorithms

- Only three options:

1. *Increase clock speed*

Faster technologies, already at limit

2. *Reduce number of instructions*

Better compilers, better algorithms, simplified algorithms

3. *Increase CPI- Clock cycles Per Instruction*

Better architectures, parallel execution of instructions

Extensión para variaciones en frecuencia

- Fabricantes han incorporado el control de cambio de frecuencia en sus procesadores



Turbo Core

Precision Boost



Turbo Boost

Procesadores con cambio de frecuencia

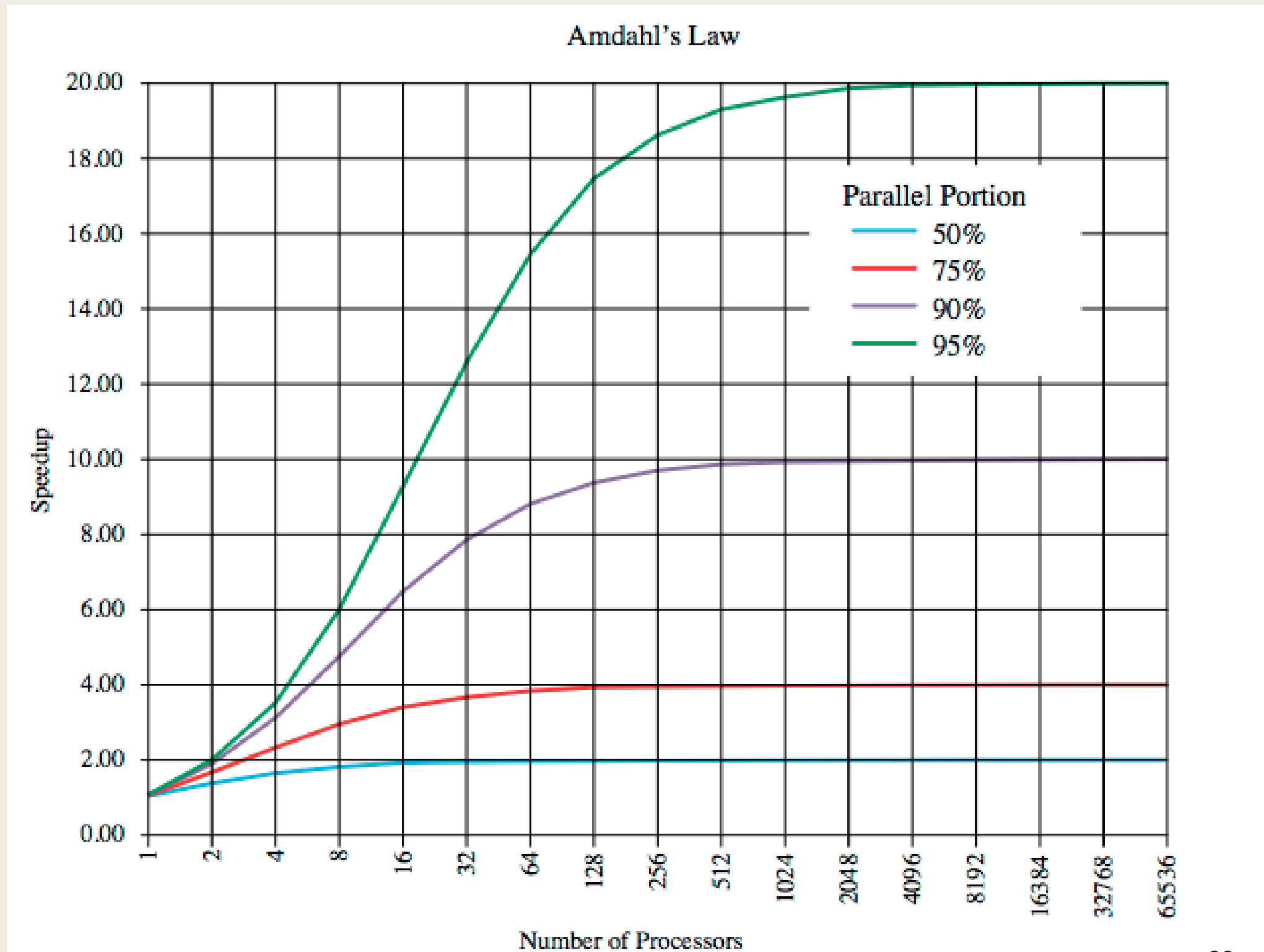
Ley de Amdahl

Gene Amdahl publica en 1967 su artículo:

Validity of the single processor approach to achieving large scale computing capabilities. In Proceedings of the April 18-20, 1967, spring joint computer conference. ACM, 1967. p. 483-485.

Se puede predecir la aceleración de un programa paralelo conociendo su parte secuencial.

Ley de Amdahl



Ley de Amdahl

- Tiempo de ejecución de un programa secuencial

$$T(N, 1) = \sigma(N) + \rho(N)$$

$\sigma(N)$ Tiempo de ejecución de la parte inherentemente secuencial

$\rho(N)$ Tiempo de ejecución de la parte potencialmente paralela

- Tiempo de ejecución de un programa paralelo

$$T(N, p) = \sigma(N) + \frac{\rho(N)}{p} + \kappa(N, p)$$

$\kappa(N, p) \geq 0$ Tiempo de ejecución del *overhead* producido por la paralelización

p es el número de procesadores

Que es GPGPU ?

- Computación de propósito general usando GPU y API de gráficas en aplicación es distintas a gráficos en 3D.
 - *GPU acelera la trayectoria crítica de una aplicación.*
- Algoritmos paralelos sobre datos aprovechan los atributos del GPU.
 - *Grandes arreglos de datos, rendimiendo de “streaming”.*
 - *Paralelismo de grano fino SIMD.*
 - *Computaciones de punto flotante de baja latencia.*
- Aplicaciones – ver [//GPGPU.org](http://GPGPU.org)
 - *Efectos físicos de video juegos (FX), procesamiento de imágenes.*
 - *Modelado físico, ingeniería computacional, algebra matricial, convolución, correlación, ordenamientos.*



CUDA

- “Compute Unified Device Architecture”
- Modelo de programación de propósito general
 - *El usuario inicializa conjuntos de threads en el GPU*
 - *GPU = super-threaded dedicado para procesamiento masivo de datos (co-processor)*
- Conjunto de software dedicado
 - *Manejadores de dispositivos, lenguaje y herramientas.*
- Manejador para carga de programas al GPU
 - *Manejador Independiente - Optimizado para computaciones*
 - *Interfaz diseñada para computaciones - API no gráfica*
 - *Comparte datos con objetos OpenGL en buffer*
 - *Aceleración garantizada en los accesos a memoria*
 - *Manejo explicito de la memoria del GPU*



Computación paralela en GPU

- Diversas Familias de Tarjetas:
Fermi, Kepler, Maxwell, Pascal, Volta, ...
- Productos que manejan CUDA (<http://developer.nvidia.com/cuda-gpus>)
 - *Disponible en laptops, desktops, servidores y nodos para clusters*



CUDA-Enabled Tesla Products



CUDA-Enabled Quadro Products



CUDA-Enabled NVS Products

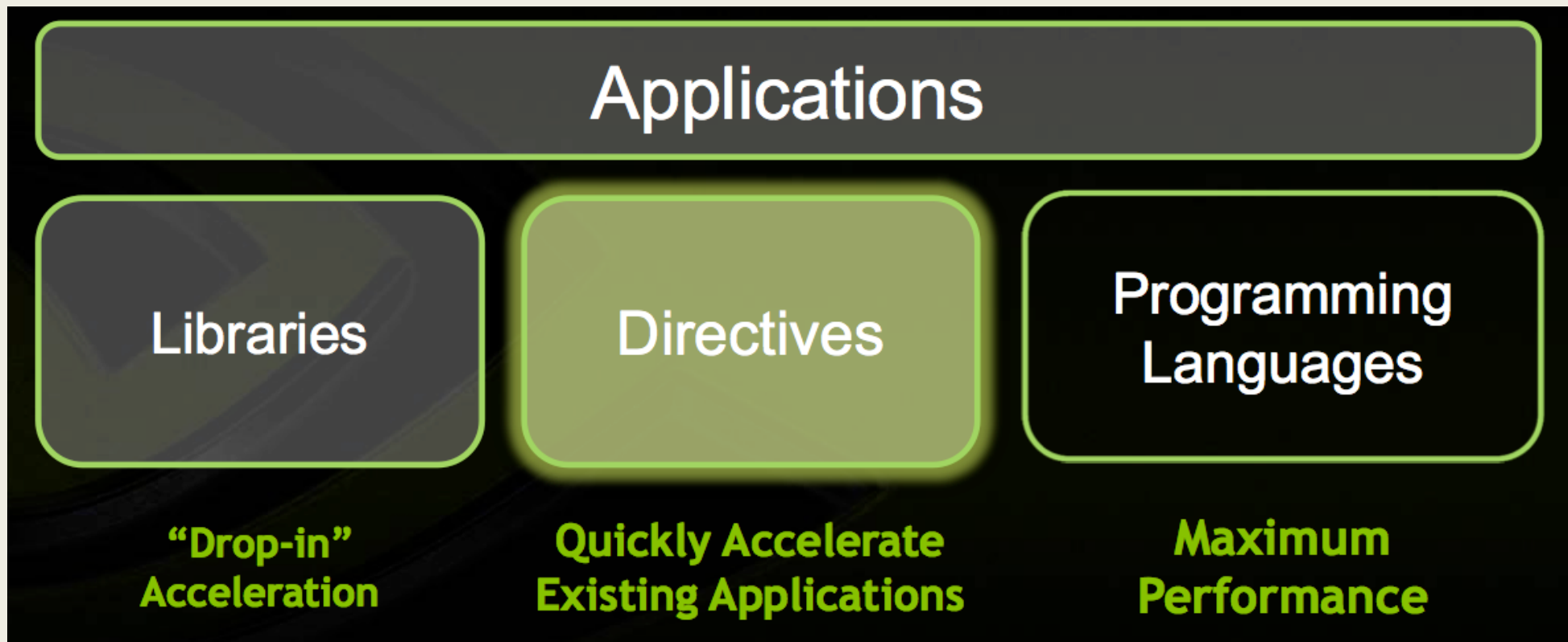


CUDA-Enabled GeForce Products







CUDA-Enabled TEGRA /Jetson Products

Modelo de programación

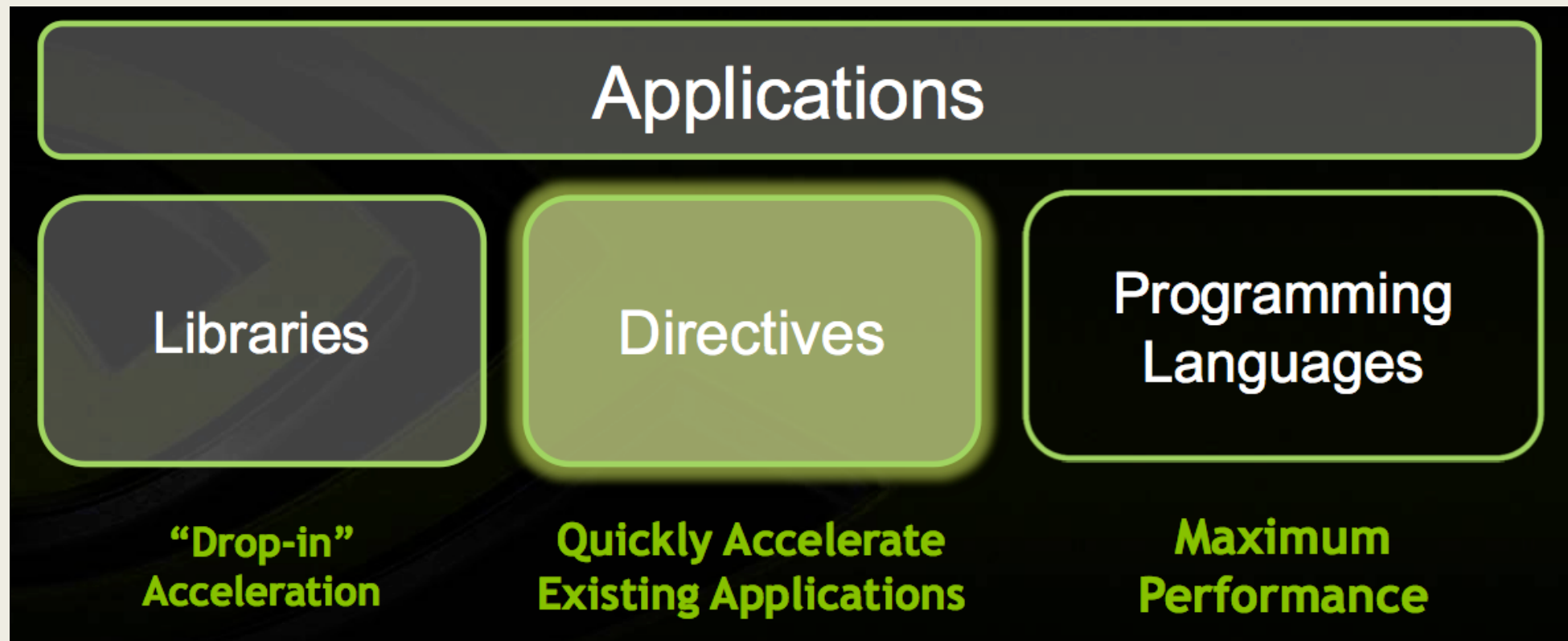


1 GPU-Accelerated Libraries
Drop in a GPU-accelerated library to replace MKL, IPP, FFTW and other widely-used libraries


 Thrust C++ Template Library	 cuBLAS Linear Algebra	 cuSPARSE	 NPP Signal & Image Processing	 cuFFT
--	---	--	--	---

More GPU-Accelerated Libraries

Modelo de programación



2 GPU Directives
Automatically parallelize loops in your Fortran or C code using OpenACC directives

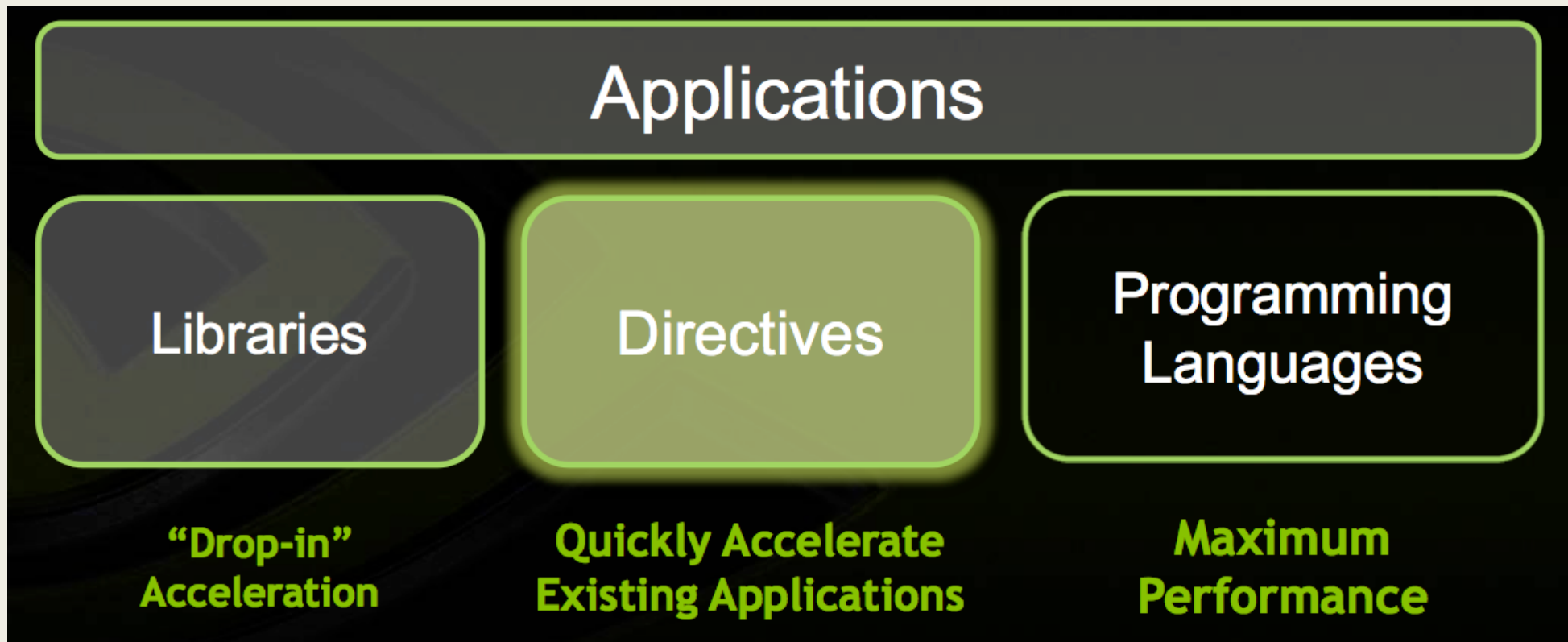


- Easy : simply insert hints in your code
- Open : run on either CPU or GPU
- Powerful: tap into the power of GPUs within minutes

[Learn More...](#)

[Learn More About Directives](#)

Modelo de programación



3

Programming Languages

Develop your own parallel applications and libraries using a programming language you already know



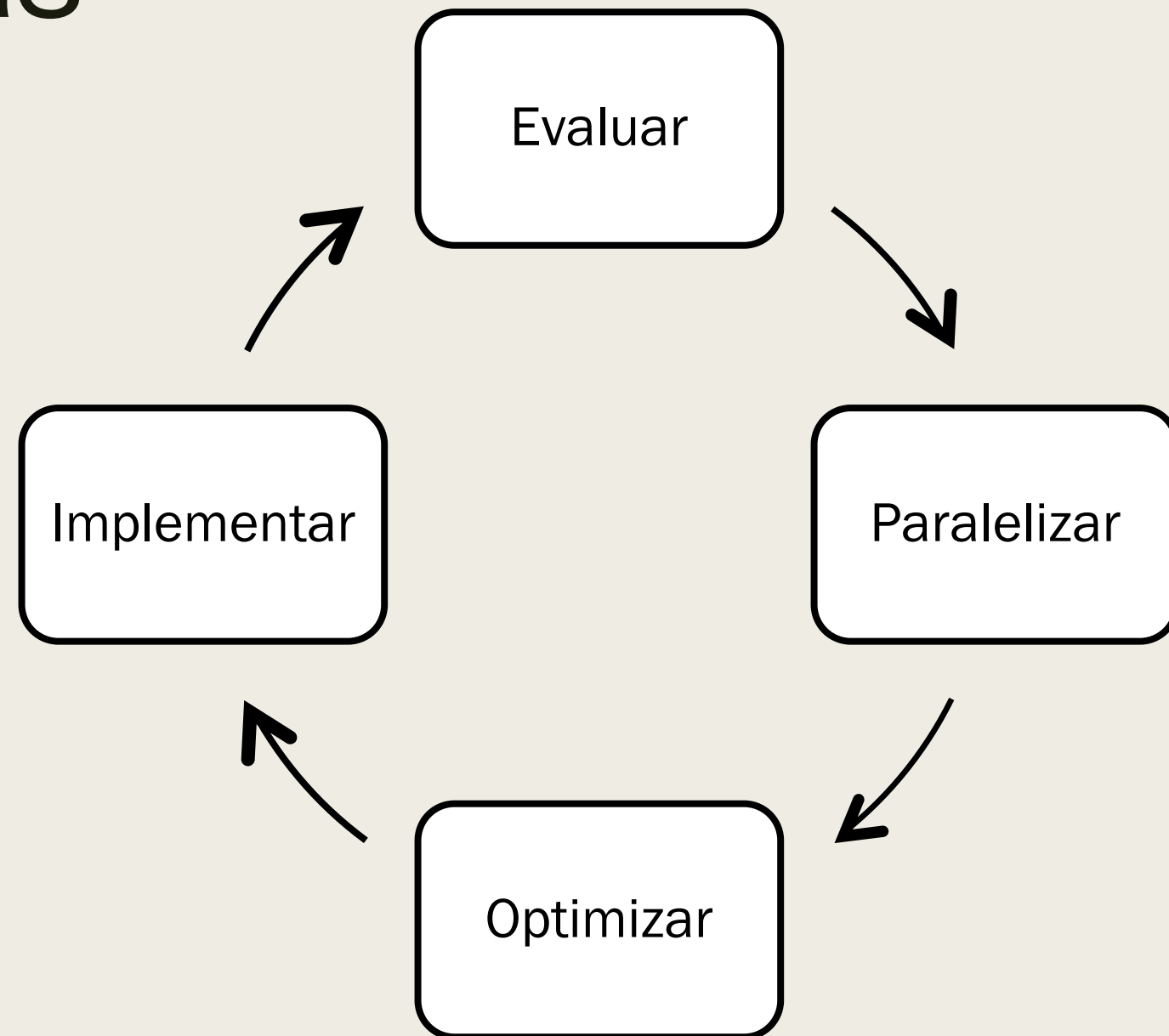
CUDA C/C++
GPU Acceleration for
C and C++ Apps.
[Learn more...](#)



CUDA Fortran
GPU Acceleration for
Fortran Applications
[Learn more...](#)

[More Programming Language Solutions](#)

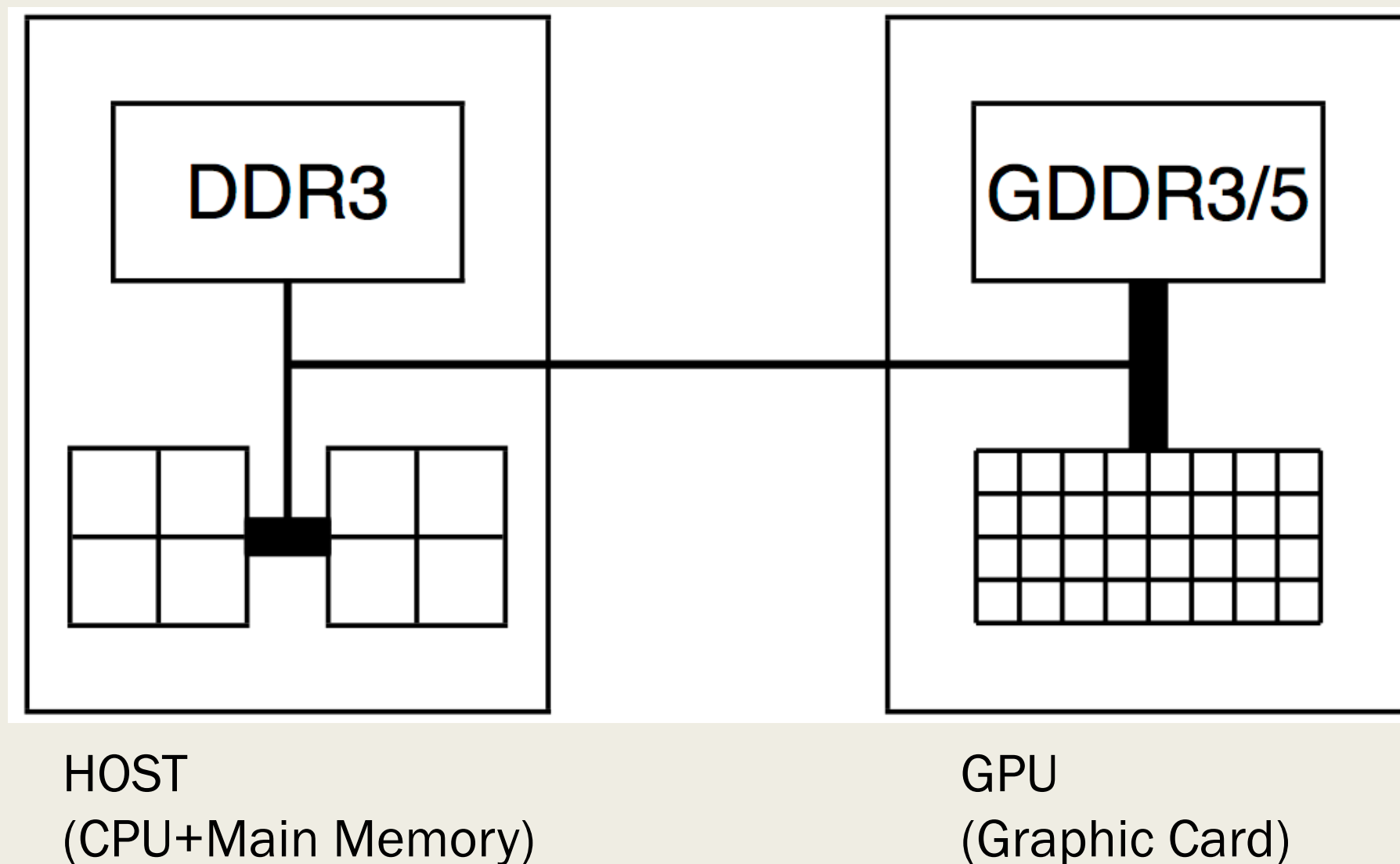
Ciclo de diseño de aplicaciones paralelas



Modelo de programación

- Requerimientos para correr CUDA
 - *Tarjeta nVidia (Tecnología, Fermi, Kepler, Maxwell, Pascal)*
<http://developer.nvidia.com/cuda-gpus>
 - Quadro
 - GeForce
 - Tesla
 - Embedded
 - Link
 - *Manejador de tarjeta (Driver)*
 - *Kit de desarrollo de nVidia (Cuda Developer)*
<http://developer.nvidia.com/cuda-toolkit>

Modelo de programación



Modelo de programación

Código secuencial
(host)

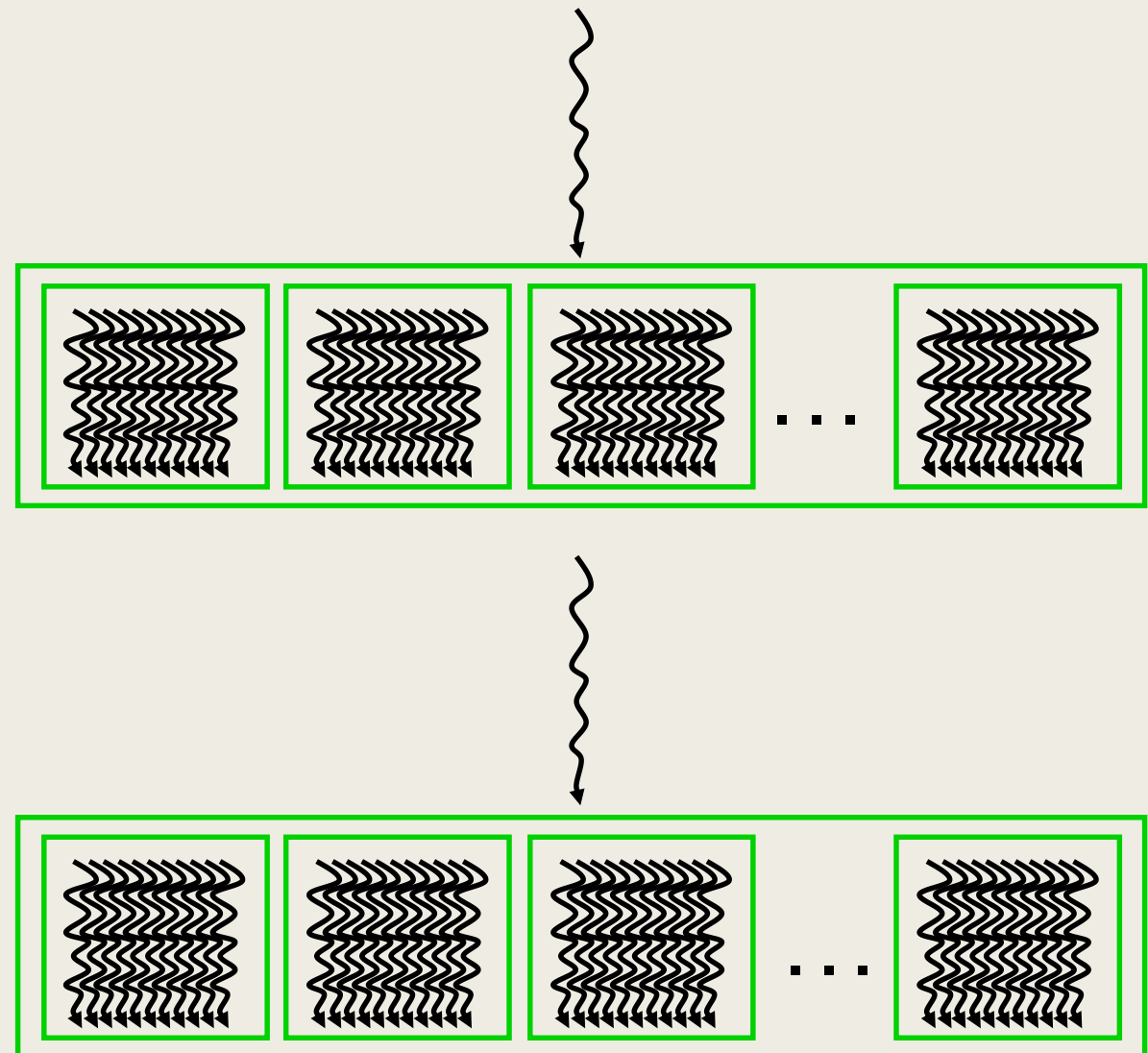
Kernel paralelo (device)

KernelA<<< nBlk, nTid >>>(args);

Código Secuencial
(host)

Kernel paralelo (device)

KernelB<<< nBlk, nTid >>>(args);

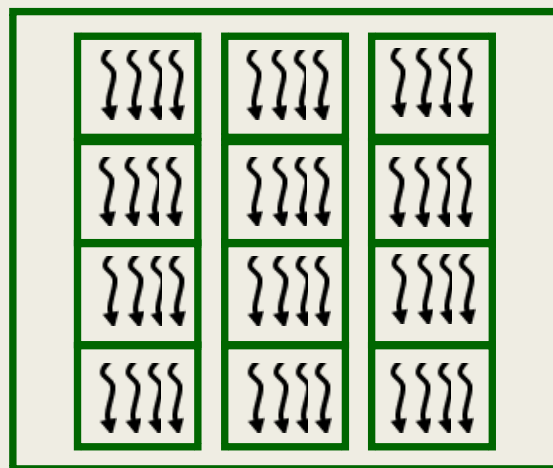


Conceptos de programación

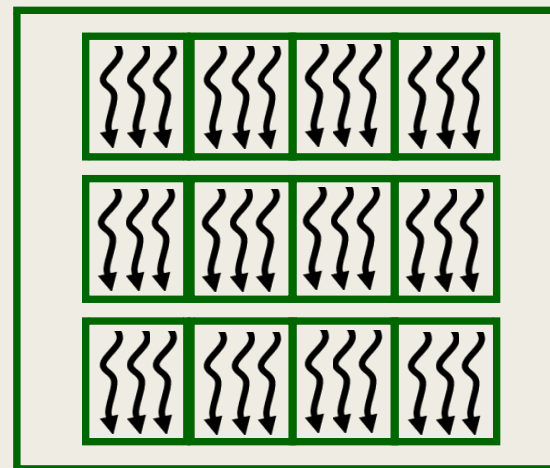
Thread (hilo). Es la unidad básica de control y ejecución del programa.

Block (bloques). Es un conjunto de hilos.

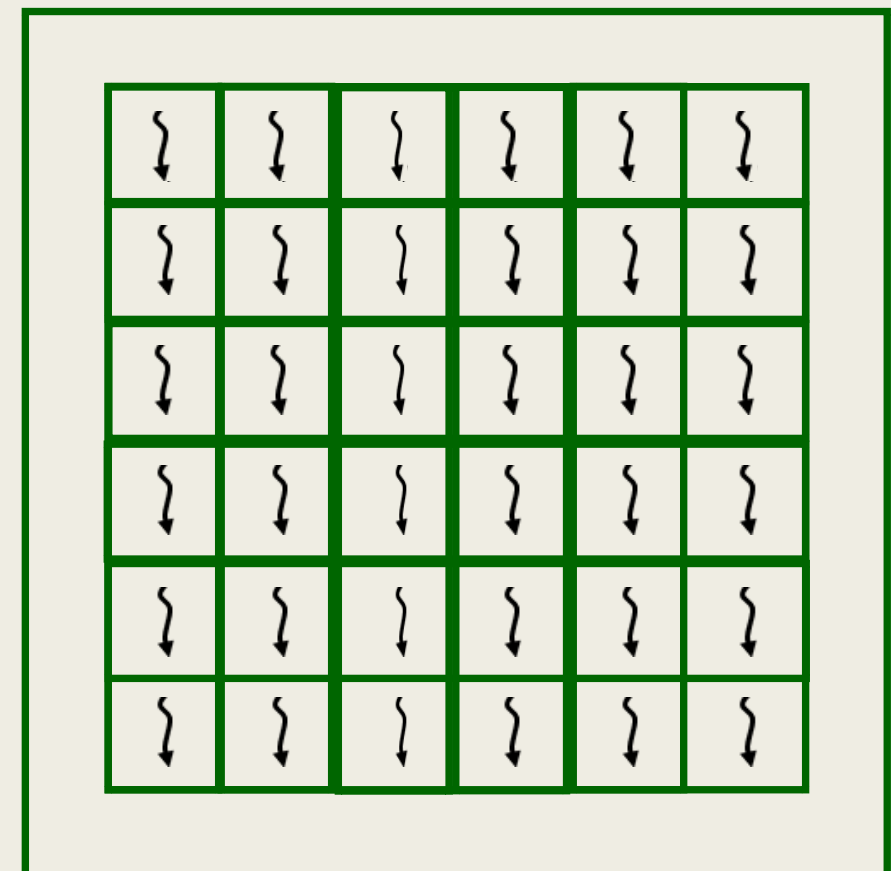
Grid. Es el arreglo de control para un código que contiene bloques de hilos.



Configuración 1

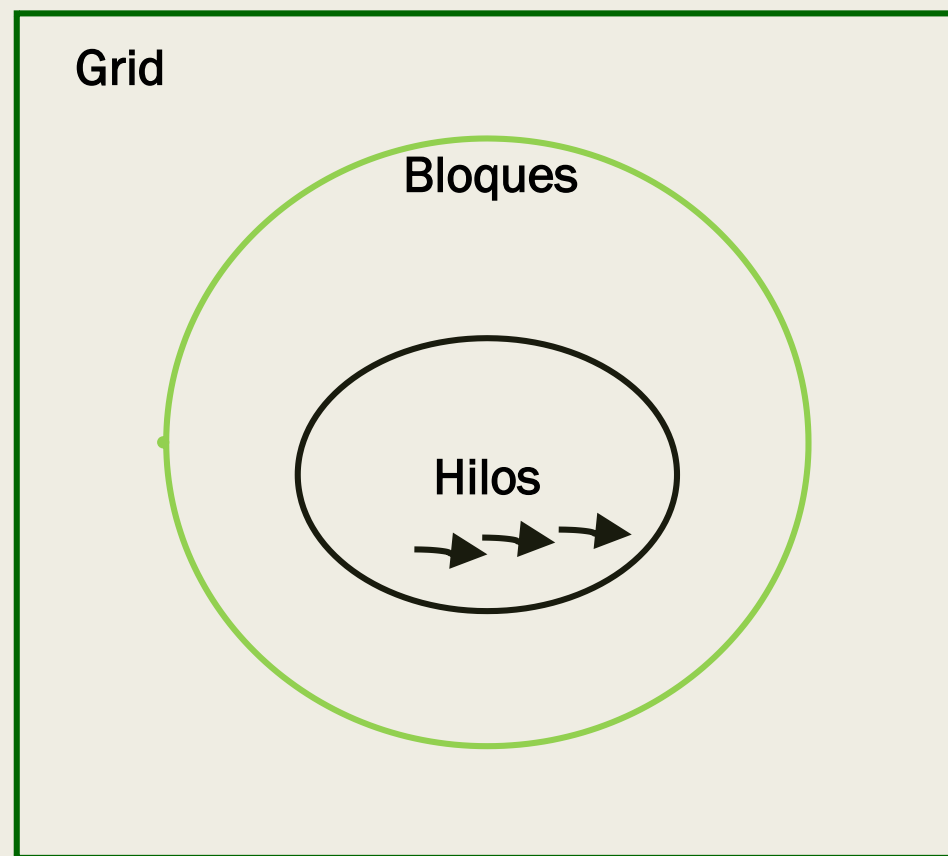


Configuración 2



Configuración 3

Conceptos de programación



Indices:

- a) De hilos: `threadIdx.x`, `threadIdx.y`, `threadIdx.z`
- b) De bloques: `blockIdx.x`, `blockIdx.y`, `blockIdx.z`

De la configuración de ejecución:

- a) Del grid (Número de bloques que contiene en cada dimensión):
`gridDim.x`, `gridDim.y`, `gridDim.z`
- b) Del bloque (Número de hilos que contiene en cada dimensión):
`blockDim.x`, `blockDim.y`, `blockDim.z`

Extensiones a C

■ Declspecs

- *global, device, shared, local, constant*

■ Palabras clave

- *threadIdx, blockIdx*

■ Escencial

- *__syncthreads*

■ Runtime API

- *Memory, symbol, execution management*

■ Funcion de lanzamiento

```
__device__ float filter[N];

__global__ void convolve (float *image) {

    __shared__ float region[M];
    ...

    region[threadIdx] = image[i];

    __syncthreads()
    ...
    image[j] = result;
}

// Allocate GPU memory
void *myimage = cudaMalloc(bytes)

// 100 blocks, 10 threads per block
convolve<<<100, 10>>> (myimage);
```

Extensiones a C

Integrated source
(foo.cu)

cudacc
EDG C/C++ frontend
Open64 Global Optimizer

GPU Assembly
foo.s

CPU Host Code
foo.cpp

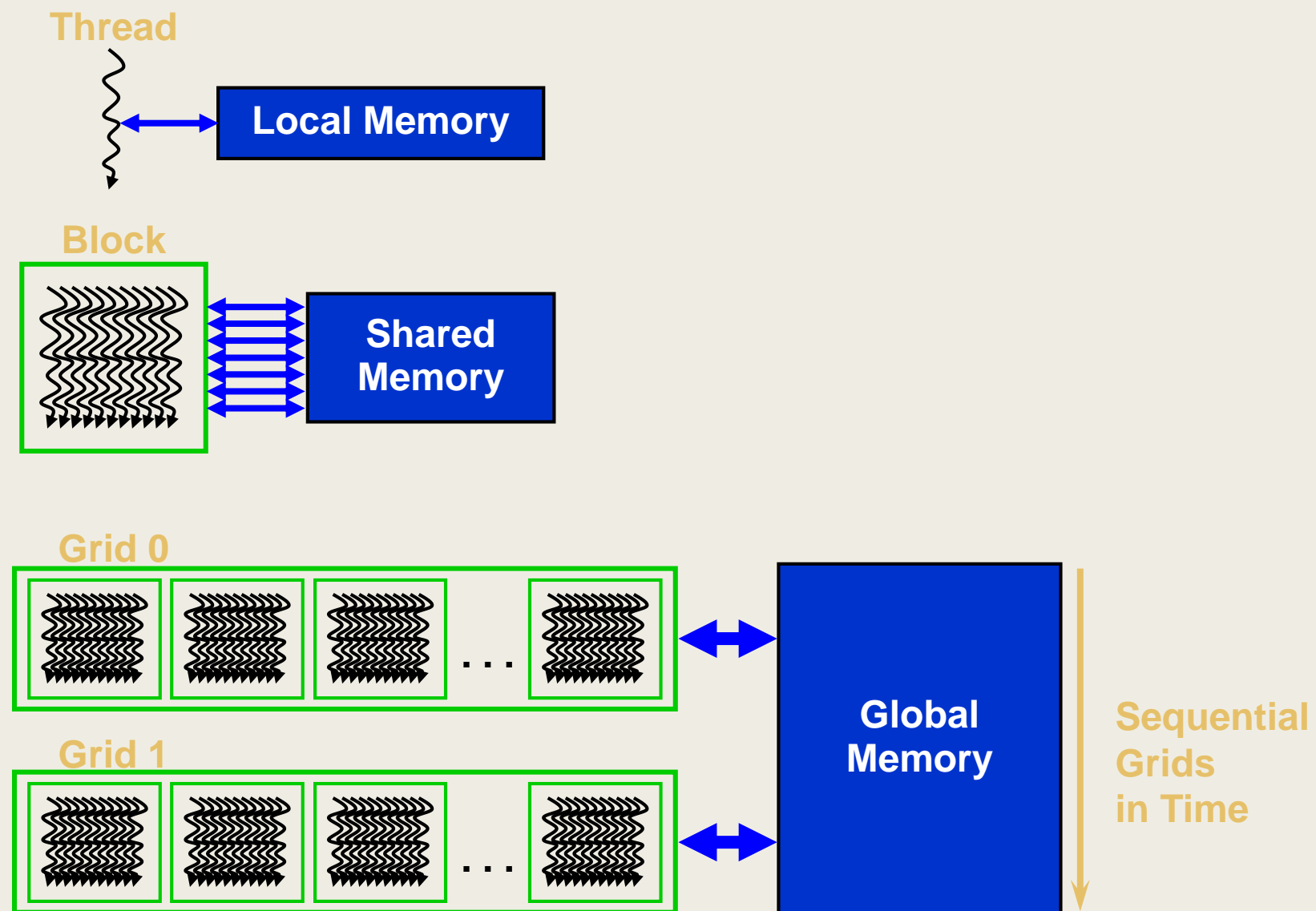
OCG

gcc / cl

G80 SASS
foo.sass

Mark Murphy, "NVIDIA's Experience with Open64,"
www.capsl.udel.edu/conferences/open64/2008/Papers/101.doc

Memoria compartida en paralelo



BIBLIOGRAFÍA

- “Taller Computación Paralela” Dr. Amilcar Meneses Viveros (UNAM) Dic. 2018
<http://computacion.cs.cinvestav.mx/%7Eameneses/index.html>
- Sitio: “Introduction to Parallel Computing Tutorial”
Lawrence Livermore National Laboratory (EUA)
<https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial>
- Sitio “CUDA Toolkit Documentation”
NVIDIA
<https://docs.nvidia.com/cuda/index.html>