

Sistemas Distribuidos y Paralelos

Ingeniería en Computación



Introducción a la programación sobre GPUs

Universidad Nacional de La Plata



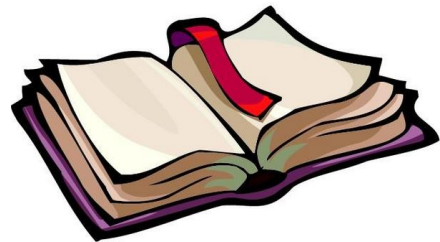
Facultad de Informática



Agenda

2

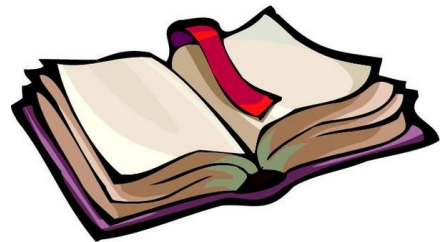
- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos



Agenda

3

- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos



Era Manycore - GPGPU

4

Manycore: arquitectura que posee una gran cantidad de cores simples (x10, x100, x1000 ...)

- La **era manycore** surge a partir de las placas gráficas o **GPUs** (siglas en inglés de Graphics Processing Units)
 - Las GPUs aparecen a fines de la década de 1980 y evolucionan a partir de la industria de los videojuegos.
 - Inicialmente, implementaban el pipeline gráfico en hardware usando procesadores NO programables destinados a tareas específicas (GPUs de Arquitectura Fija)
 - A partir de 2006, incluyeron procesadores programables (GPUs de Arquitectura Unificada) que permitieron resolver problemas de propósito general (NO gráfico).
 - Surge el concepto GPGPU:

General-Purpose Computing on Graphics Processing Units

- Las GPUs poseen una gran cantidad de procesadores simples y han demostrado alcanzar un alto rendimiento.

Era Manycore - GPGPU

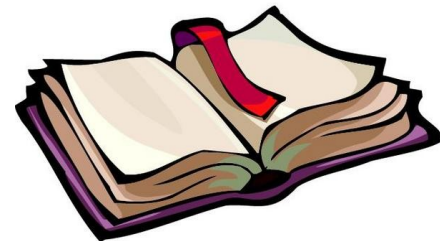
5

- Existen dos empresas que lideran el mercado de las GPUs:
 - Nvidia
 - AMD – ATI
- Nvidia y AMD fabrican placas para el mercado de la computación gráfica, que además pueden ser programadas para aplicaciones de propósito general utilizando la GPU como co-procesador conectado al bus PCI express (PCIe).
- Otras empresas han fabricado procesadores manycores para propósito general similares. Por ejemplo Intel:
 - Xeon PHI (actualmente discontinuado)
 - Intel ARC (Placa gráfica PCIe con capacidad para cómputo general)

Agenda

6

- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos



Características paralelas de las GPUs

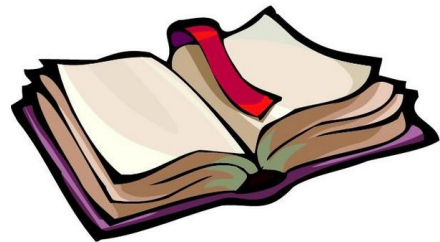
7

- Las GPUs como arquitecturas paralelas tienen las siguientes características:
 - Arquitectura de memoria compartida
 - Modelo SIMD - STMD
 - Se adaptan mejor a paralelismo de datos sobre datos regulares
 - Adecuadas para aplicaciones intensivas en CPU

Agenda

8

- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos



Arquitectura Nvidia

9

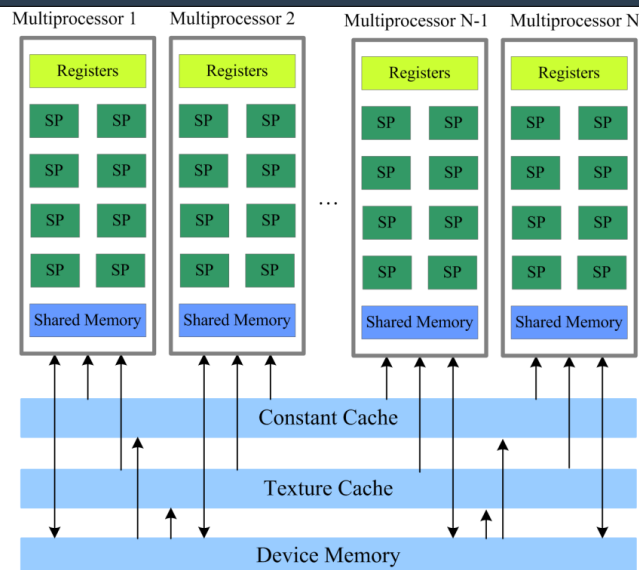
- Una **GPU Nvidia** está compuesta por:
 - Grupos de procesadores simples que forman el **sistema de procesamiento**
 - Una jerarquía de memoria compleja que forma el **sistema de memoria**



Grupo de cores



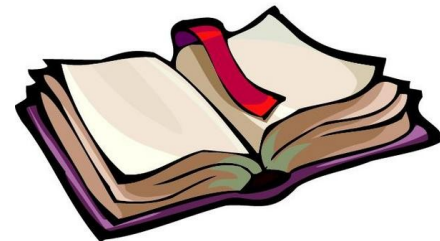
Jerarquia de Memoria



Agenda

10

- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos



Arquitectura Nvidia

Sistema de procesamiento

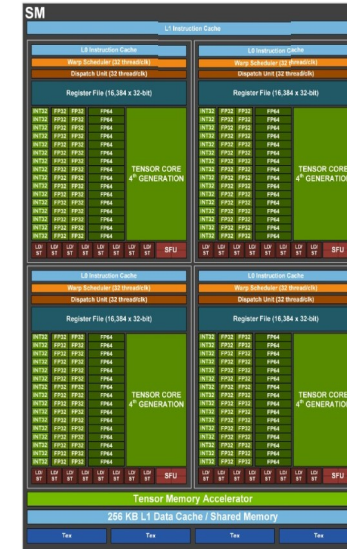
11

- Aunque la estructura de las GPUs Nvidia puede variar de un modelo a otro, las diferentes arquitecturas tienen algunos componentes comunes.
- Cada placa está compuesta por un conjunto de multiprocesadores llamados Stream multiprocessors o SM.

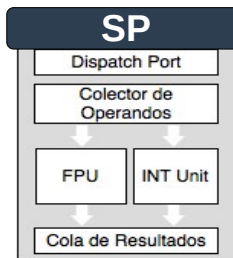
Modelos 2010 - 2016



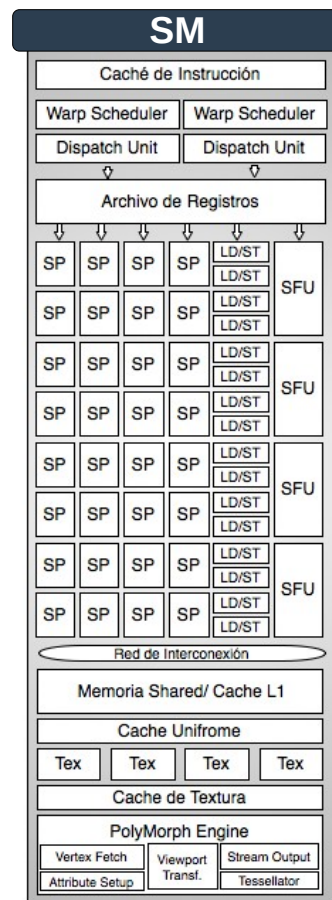
Modelos 2017 - Actualidad



- En modelos 2010-2016
- Cada SM:
 - 32 procesadores simples llamados Scalar processors o SPs
 - 16 Unidades de Load/Store:
 - Escriben o leen direcciones de 16 threads por ciclo
 - 4 Unidades de Funciones Especiales o SFUs
 - Planificadores de Hilos o Warp Schedulers
- Cada SP es un procesador muy simple, contiene:
 - Una unidad de punto flotante (FPU) simple y doble precisión
 - Una unidad aritmético lógica (ALU)



Modelos 2010 - 2016



- En modelos 2017 y actuales.
- Cada SM 4 bloques de procesamiento:
 - Cada bloque:
 - 32 cores FP32
 - 16 cores FP64
 - 16 cores INT32
 - 1 Tensor core de 4ta generación

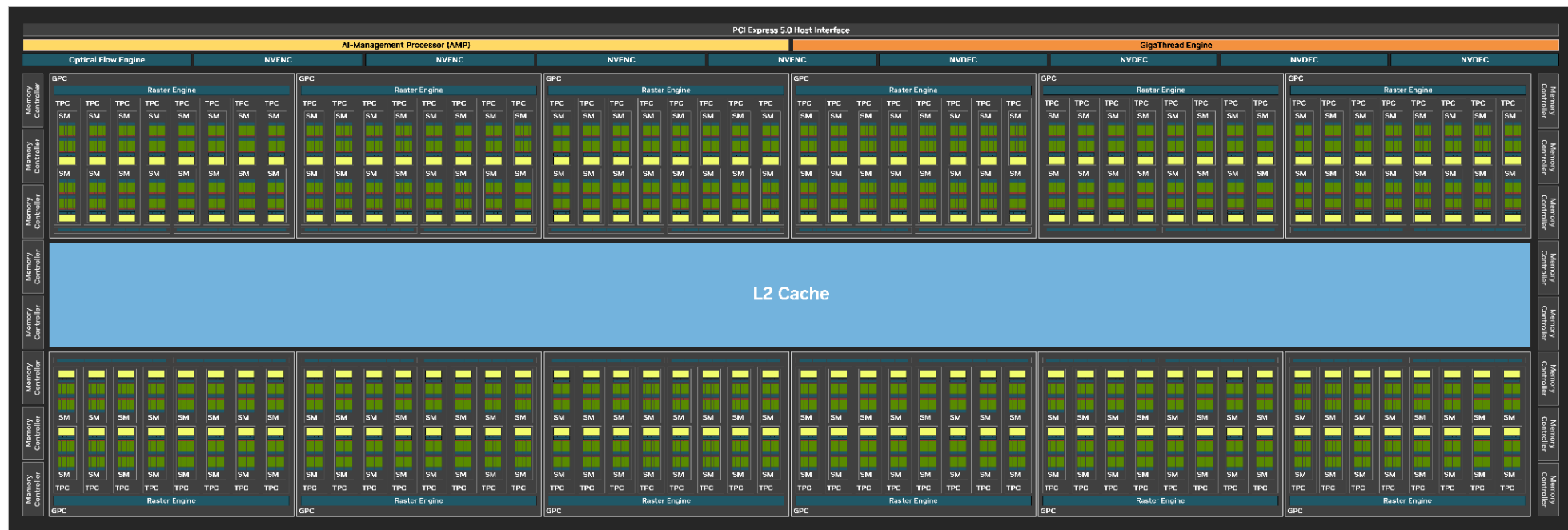
Modelos 2017 - Actualidad



Arquitectura Nvidia

Sistema de procesamiento – Modelo 2017-Actualidad

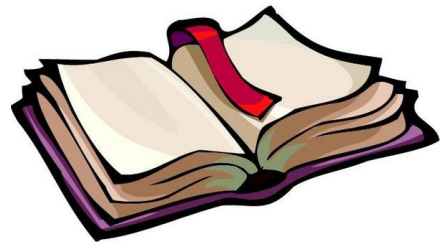
- Nvidia microarquitectura Blackwell (RTX 5090):
 - 24576 Cores
 - 768 Tensor Cores



Agenda

15

- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos



Arquitectura Nvidia

Sistema de memoria

16

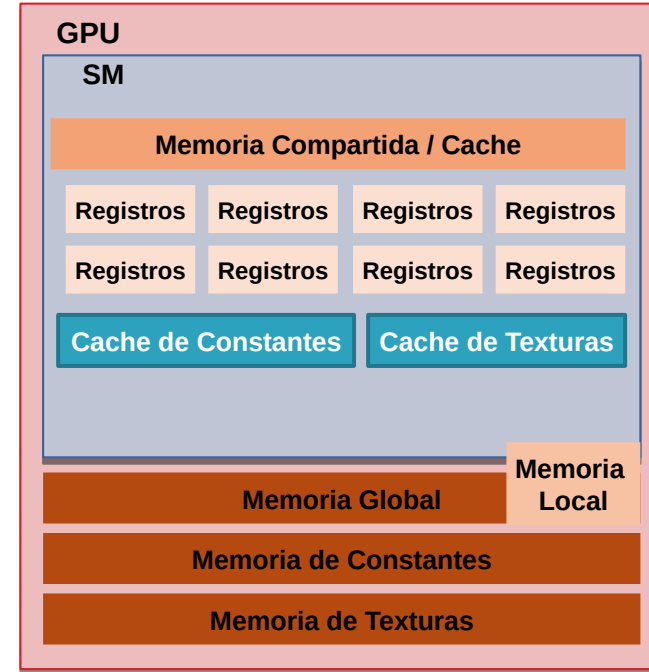
- El sistema de memoria se organiza en una jerarquía compleja que puede clasificarse:

- **On-Chip:** Dentro de cada SM

- Registros
- Memoria compartida (shared)
- Distintas caché

- **Off-Chip:** Fuera de los SMs

- Memoria global
- Memoria de constantes
- Memoria de texturas
- Memoria local

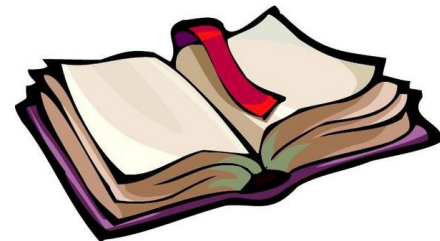



- El costo de acceso a las memorias On-Chip (más rápidas) es menor al costo de acceso a las memorias Off-Chip (más lentas).

Agenda

17

- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos

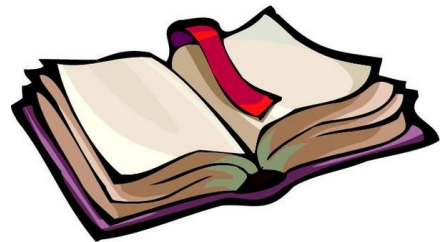


- A partir del concepto de GPGPU, Nvidia provee una plataforma, basada en lenguaje C, llamada **CUDA** que facilita la programación sobre sus arquitecturas de GPU.
- La programación utilizando CUDA tiene algunas características específicas, a tener en cuenta:A yellow thinking face emoji with a hand on its chin, indicating a list of specific characteristics.
 - Copia explícita de memoria: Los datos a ser procesados dentro de una GPU y los resultados obtenidos deben copiarse explícitamente (lo hace el programador) desde la memoria RAM de la CPU a la memoria de la GPU, y viceversa.
 - Organización de los hilos: los hilos se organizan en grupos (grids, bloques, warps)
 - Todos los hilos ejecutan el mismo código.
 - La unidad de planificación no es un hilo individual sino un grupo de hilos que se asignan a los SMs.

Agenda

19

- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos



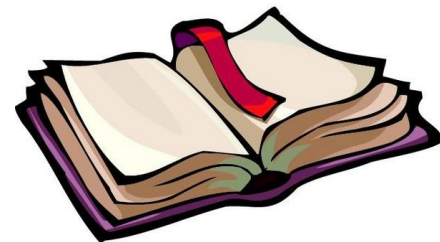




- No existe tal cosa como ejecutar un programa secuencial en una GPU.
- Si se utiliza como métrica el Speedup, será respecto al mejor algoritmo secuencial ejecutado sobre alguna CPU. Por lo tanto, será un Speedup relativo.
- Es posible calcular el Speedup respecto a otras arquitecturas paralelas (multicores, clusters, etc.)
- Por las características específicas de la programación sobre GPU, la idea de Speedup ideal no queda bien definida.
- Aunque la ley de Amdahl, la ley de Gustafson y el concepto de escalabilidad siguen valiendo, el grado de transparencia al ejecutar en una única GPU hacen difícil evaluarlo.
- Es mas útil utilizar otro tipo de métricas como el **Rendimiento efectivo (Throughput)** medido en GFLOPs.

Agenda

21

- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos

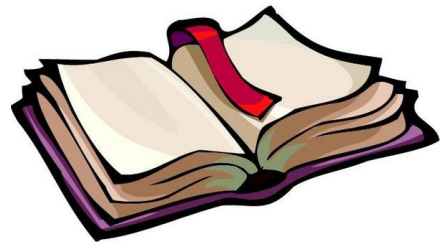


- Para alcanzar un buen rendimiento sobre GPUs es necesario aprovechar las ventajas de la arquitectura subyacente.
- Necesitamos conocer las características y factores limitantes en el rendimiento de las GPU.
- Existen distintas técnicas de optimización de una aplicación que abordan distintos aspectos:
 - Relacionadas a la memoria:
 - Organización de los accesos (**Coalescence**)
 - Técnicas de carga anticipada de datos (**Prefetching**).
 - Relacionadas a la ejecución de los threads (**Divergence**).
 - Relacionadas al rendimiento de las instrucciones: mezcla y granularidad.
 - Relacionadas a la asignación de recursos en un SM (**Occupancy**).

Agenda

23

- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos



Ocultamiento de latencia – CUDA Streams

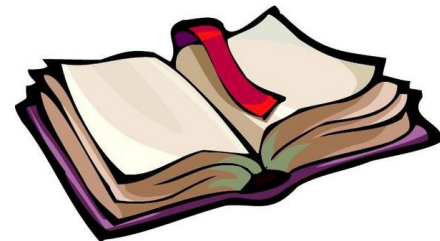
- Flujo de procesamiento típico sobre GPU:
 - 1) Copiar los datos de entrada de CPU a GPU
 - 2) Realizar la ejecución sobre GPU
 - 3) Copiar los resultado de GPU a CPU
- Las transferencias de datos desde y hacia la GPU es costosa.
- CUDA permite aplicar una técnica, conocida como **CUDA Streams**, para ocultar la latencia de estas transferencias, solapando el cómputo con la copia de datos en ambos sentidos.
- Por otro lado, CUDA Stream permite ejecutar varias aplicaciones simultáneamente sobre una única GPU cuando cada una de ellas no utiliza la GPU al 100%.



Agenda

25

- I. Era Manycore - Introducción al concepto de GPGPU
- II. Características paralelas de las GPUs
- III. Arquitectura Nvidia
 - i. Sistema de procesamiento
 - ii. Sistema de memoria
- IV. Modelo de programación Nvidia CUDA
- V. Métricas en GPU
- VI. Optimizaciones Nvidia CUDA
- VII. Ocultamiento de latencia – CUDA Streams
- VIII. MultiGPU y modelos híbridos



Nvidia CUDA

MultiGPU y modelos híbridos



26

- Actualmente, podemos contar con máquinas que poseen mas de una GPU y utilizarlas para resolver problemas específicos.
- Además de contar con varias GPUs, esas máquinas suelen ser multicores muy potentes y podrían estar conectadas con otras máquinas similares conformando un cluster.
- Esto resulta en una **arquitectura paralela híbrida heterogénea** muy potente y podemos utilizar un modelo de programación híbrido que incluya las GPUs:
 - MPI + Pthreads + CUDA
 - MPI + OpenMP + CUDA
- Además de la dificultad en la programación y compilación, existe una dificultad en la distribución de carga ya que las CPUs y las GPUs tienen potencias de cómputo diferentes.



