

TALLER DE PROGRAMACIÓN SOBRE GPUS

Facultad de Informática – Universidad Nacional de La Plata



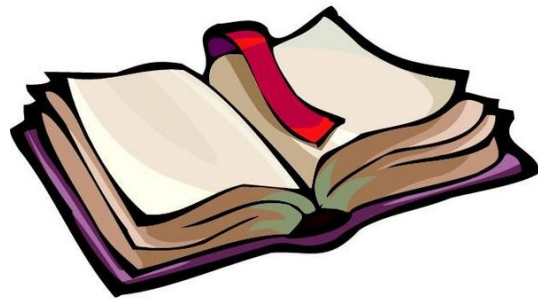
Dr. Adrián Pousa

Introducción al HPC y GPGPU

Agenda

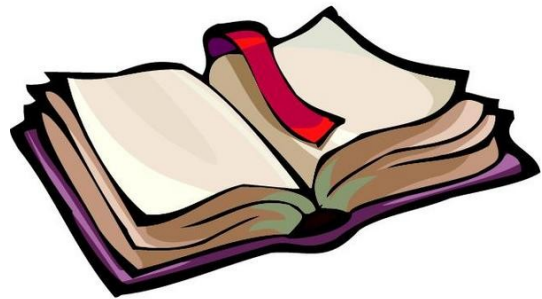
2

- I. Introducción al concepto de HPC**
- II. La GPU como arquitectura paralela**
 - I. Mecanismos de control (Taxonomía de Flynn)**
 - II. Modelos de comunicación**
- III. La GPU y software paralelo**
 - I. Descomposición de problemas**
 - II. Características de las aplicaciones**
 - III. Herramientas de programación sobre GPU y modelos de programación Híbridos**
- IV. Resumen de características**



Agenda

- I.** *Introducción al concepto de HPC*
- II.** *La GPU como arquitectura paralela*
 - I.** *Mecanismos de control (Taxonomía de Flynn)*
 - II.** *Modelos de comunicación*
- III.** *La GPU y software paralelo*
 - I.** *Descomposición de problemas*
 - II.** *Características de las aplicaciones*
 - III.** *Herramientas de programación sobre GPU y modelos de programación Híbridos*
- IV.** *Resumen de características*



HPC (High Performance Computing)

4

- Se apoya en la programación paralela, las arquitecturas y técnicas que permiten alcanzar mayor rendimiento en la resolución de problemas
- Abarca varios temas:
 - ▣ Arquitecturas:
 - ILP
 - Clusters
 - Multi-procesadores/Multi-cores
 - Grid
 - Cloud
 - **GPU**
 - ▣ Programación paralela:
 - En arquitecturas de memoria compartida
 - En arquitecturas de memoria distribuida
 - ▣ Tolerancia a Fallos
 - ▣ Monitorización
 - ▣ Simulación
 - ▣ Administración de recursos
 - ▣ Optimización de rendimiento
 - ▣ Instrumentación

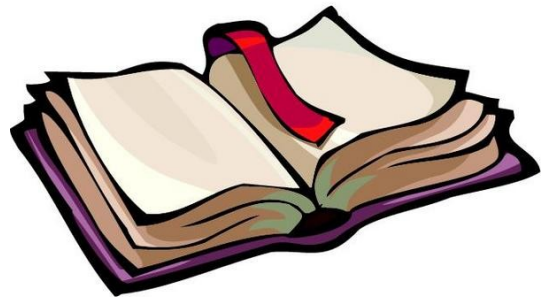
Sistemas paralelos

5

- Sistema paralelo compuesto por:
 - ▣ **Arquitectura paralela:** arquitectura con varias unidades de procesamiento que permiten el procesamiento paralelo
 - ▣ **Software paralelo:** varios procesos o hilos que cooperan para la resolución de un problema, con el objetivo de lograr mayor rendimiento que un software secuencial

Agenda

- I.** *Introducción al concepto de HPC*
- II.** **La GPU como arquitectura paralela**
 - I.** **Mecanismos de control (Taxonomía de Flynn)**
 - II.** **Modelos de comunicación**
- III.** **La GPU y software paralelo**
 - I.** *Descomposición de problemas*
 - II.** *Características de las aplicaciones*
 - III.** *Herramientas de programación sobre GPU y modelos de programación Híbridos*
- IV.** *Resumen de características*



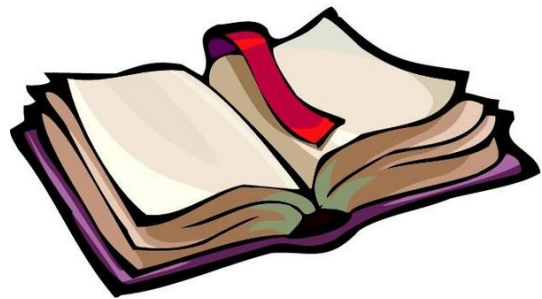
Arquitecturas Paralelas - Clasificación

7

- Existen varias clasificaciones de arquitecturas paralelas.
- Para ubicar las GPUs como arquitectura paralela nos vamos a enfocar sólo en dos de estas clasificaciones:
 - ▣ Mecanismos de control (Taxonomía de Flynn)
 - ▣ Modelo de comunicación (relacionado con la memoria)

Agenda

- I.** *Introducción al concepto de HPC*
- II.** *La GPU como arquitectura paralela*
 - I.** *Mecanismos de control (Taxonomía de Flynn)*
 - II.** *Modelos de comunicación*
- III.** *La GPU y software paralelo*
 - I.** *Descomposición de problemas*
 - II.** *Características de las aplicaciones*
 - III.** *Herramientas de programación sobre GPU y modelos de programación Híbridos*
- IV.** *Resumen de características*



Taxonomía de Flynn

9

- **Taxonomía de Flynn:** clasificación de arquitecturas paralelas propuesta por Michael Flynn en 1972.
- Se basa en dos aspectos:
 - ▣ El flujo o número de instrucciones concurrentes (control)
 - ▣ El flujo de datos

	Una Instrucción	Múltiples instrucciones
Un dato	SISD	MISD
Múltiples datos	SIMD	MIMD

Taxonomía de Flynn: SISD

10

- Single Instruction Single Data (SISD): Un único flujo de instrucciones y un único flujo de datos

Arquitecturas monoprocesadores, procesadores secuenciales sin paralelismo.



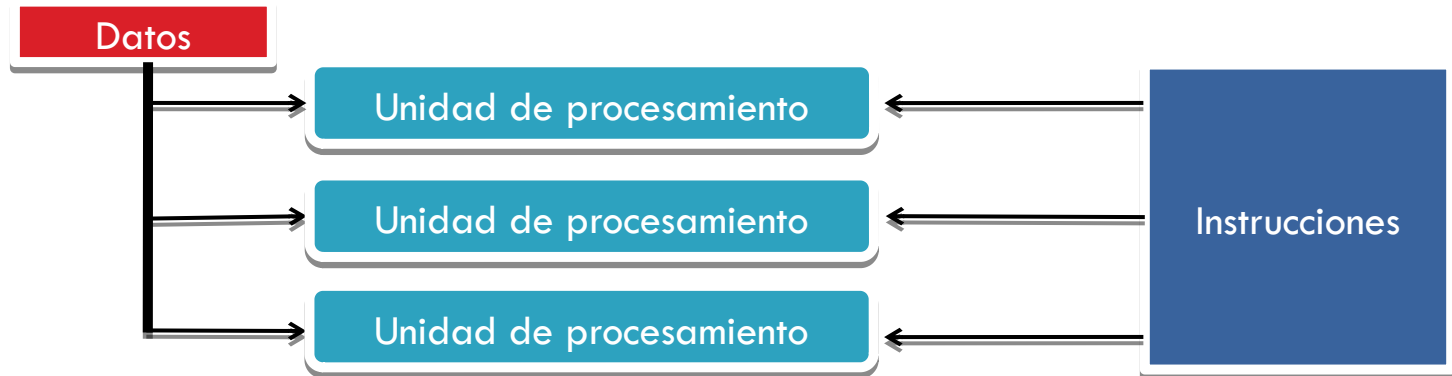
Taxonomía de Flynn: MISD

11

- Multiple Instruction Single Data (MISD): Múltiples flujos de instrucciones y un único flujo de datos.

Arquitecturas con varias **Unidades de Procesamiento** donde todas ejecutan flujos de instrucciones distintos sobre un único flujo de datos.

(**Poco común**: sistemas redundantes o tolerantes a fallos).

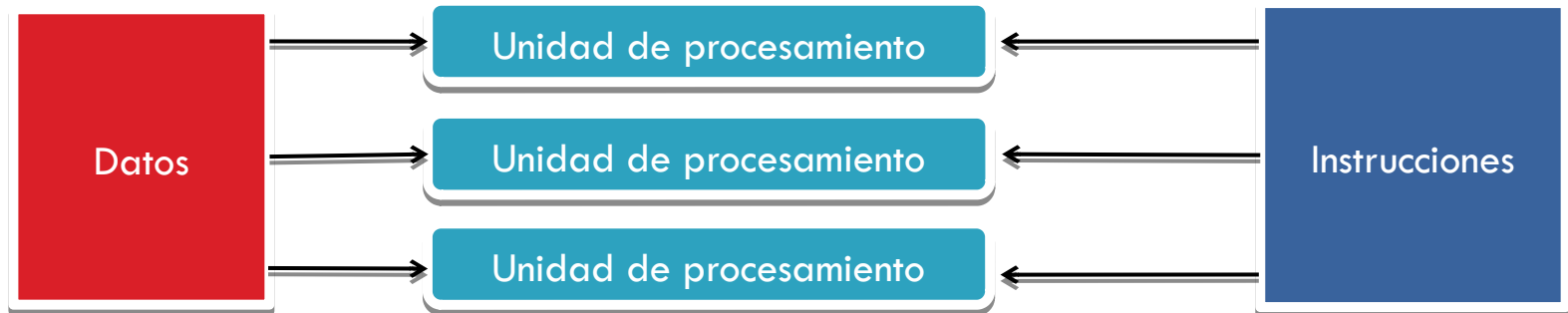


Taxonomía de Flynn: MIMD

12

- Multiple Instruction Multiple Data (MIMD): Múltiples flujos de instrucciones y múltiples flujos de datos.

Arquitecturas con varias **Unidades de Procesamiento** independientes que ejecutan sus propios flujos de instrucciones sobre sus propios flujo de datos. En este grupo se encuentran arquitecturas de memoria compartida (ej: multicores) y arquitecturas de memoria distribuida (ej: clusters)



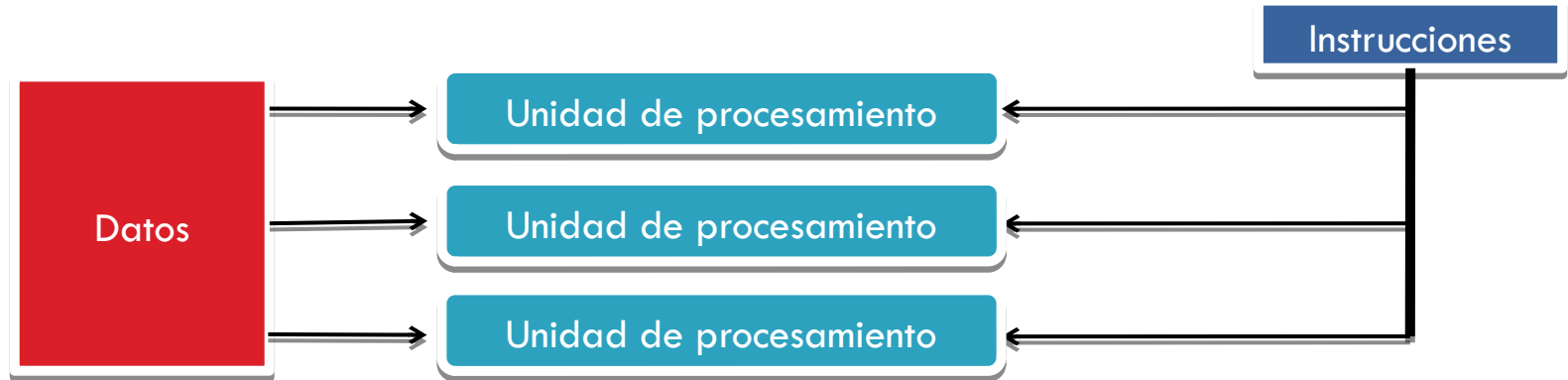
Taxonomía de Flynn: SIMD

13

- Single Instruction Multiple Data (SIMD): Un único flujo de instrucciones y múltiples flujos de datos.

Arquitecturas con varias **Unidades de Procesamiento** donde todas ejecutan sincronizadamente el mismo flujo de instrucciones sobre distintos datos.

En este grupo se encuentran los procesadores vectoriales y **GPUs**.



Taxonomía de Flynn: extensiones SIMD

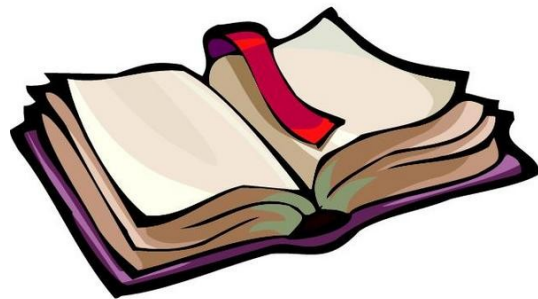
14

- Existen algunas extensiones al modelo SIMD:
 - ▣ **Single Program Multiple Data (SPMD)**: múltiples procesadores autónomos que trabajan simultáneamente sobre el mismo conjunto de instrucciones (aunque en puntos independientes) sobre datos diferentes.
 - ▣ **Single Thread Multiple Data (STMD)**: Varios Hilos donde todos ejecutan el mismo código.

STMD es como se conoce al modelo de programación en GPU.

Agenda

- I.** *Introducción al concepto de HPC*
- II.** **La GPU como arquitectura paralela**
 - I.** *Mecanismos de control (Taxonomía de Flynn)*
 - II.** **Modelos de comunicación**
- III.** *La GPU y software paralelo*
 - I.** *Descomposición de problemas*
 - II.** *Características de las aplicaciones*
 - III.** *Herramientas de programación sobre GPU y modelos de programación Híbridos*
- IV.** *Resumen de características*



Clasificación según la memoria

16

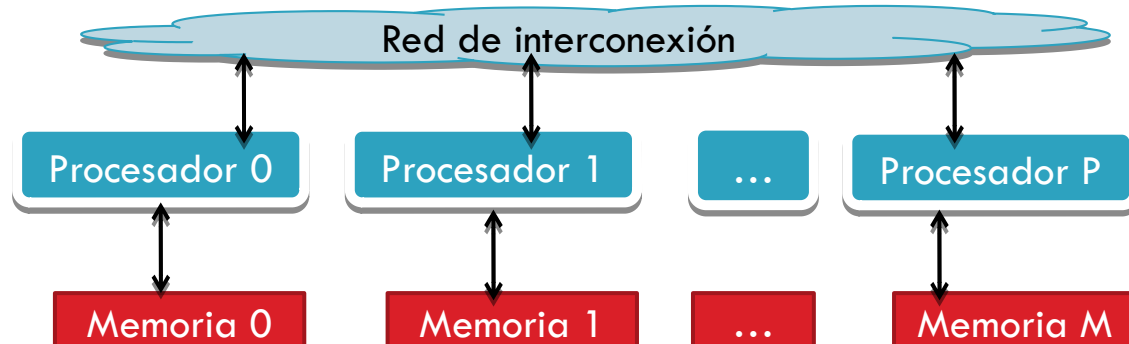
- Otra clasificación es la que considera la memoria y la comunicación entre procesadores.

- Dos grupos:
 - ▣ Arquitecturas paralelas de Memoria Distribuida
 - ▣ Arquitecturas paralelas de Memoria Compartida

Arquitecturas de memoria distribuida

17

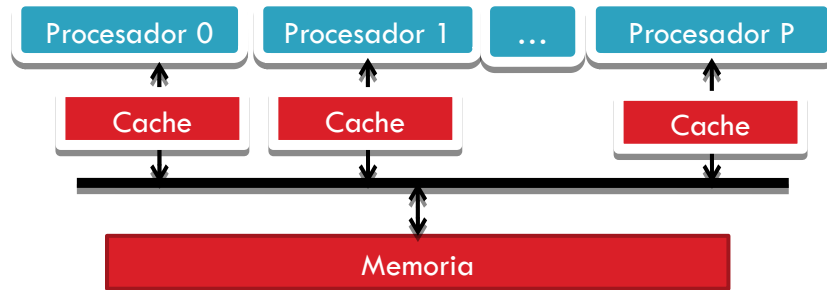
- ❑ Cada procesador tiene su propia memoria.
- ❑ Los procesadores se comunican por una red de interconexión mediante mensajes (no pueden acceder directamente a la memoria de otro procesador).
- ❑ Un ejemplo de estas arquitecturas son los clusters.



Arquitecturas de memoria compartida

18

- Se tiene un espacio único de direcciones.
- Todos los procesadores pueden acceder a cualquier dirección de memoria.

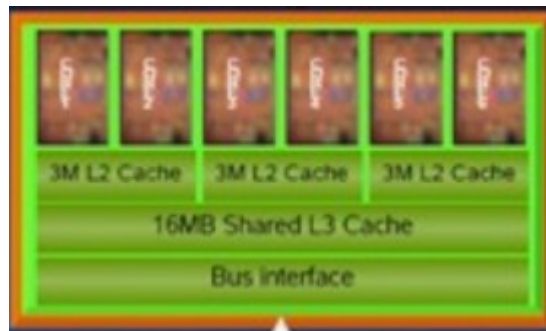
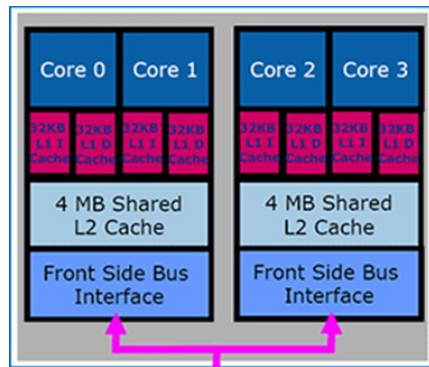


- Incluye a la mayoría de los multicores modernos y las **GPUs**.

Jerarquías de memoria

19

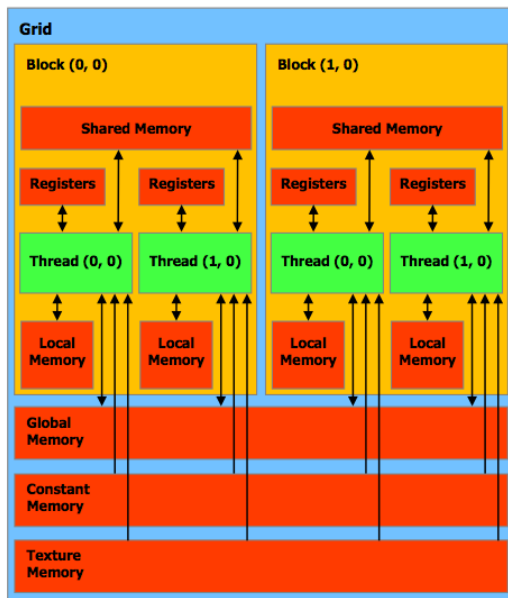
- En arquitecturas multicore es común tener jerarquías de memoria.
- Generalmente 3 niveles de cache (L1, L2, L3, L4)
- Según la arquitectura, los niveles 2, 3 y 4 suelen estar compartidos entre cores.



Jerarquías de memoria

20

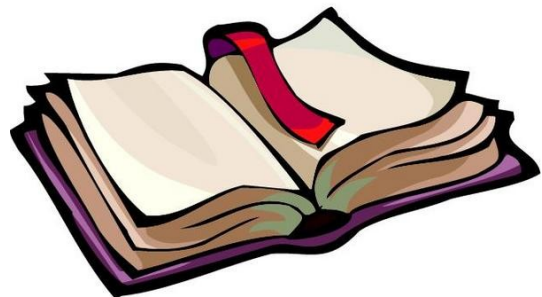
- Las **GPUs** poseen una jerarquía de memoria más compleja.



Agenda

21

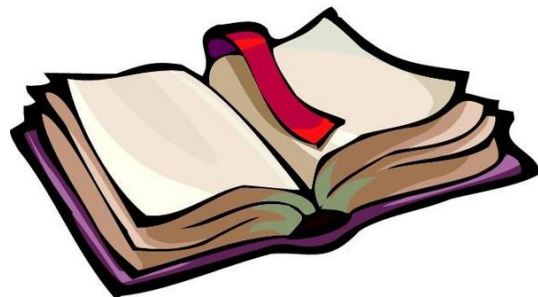
- I.** *Introducción al concepto de HPC*
- II.** *La GPU como arquitectura paralela*
 - I.** *Mecanismos de control (Taxonomía de Flynn)*
 - II.** *Modelos de comunicación*
- III.** *La GPU y software paralelo*
 - I.** *Descomposición de problemas*
 - II.** *Características de las aplicaciones*
 - III.** *Herramientas de programación sobre GPU y modelos de programación Híbridos*
- IV.** *Resumen de características*



Agenda

22

- I.** *Introducción al concepto de HPC*
- II.** *La GPU como arquitectura paralela*
 - I.** *Mecanismos de control (Taxonomía de Flynn)*
 - II.** *Modelos de comunicación*
- III.** *La GPU y software paralelo*
 - I.** *Descomposición de problemas*
 - II.** *Características de las aplicaciones*
 - III.** *Herramientas de programación sobre GPU y modelos de programación Híbridos*
- IV.** *Resumen de características*



Software paralelo

23

- Un problema se divide en subproblemas.
- Cada subproblema se resuelve concurrentemente.
- Aspectos a tener en cuenta:
 - ▣ División del trabajo en tareas
 - ▣ Asignación de tareas a procesadores (Mapping)
 - ▣ Organización de la arquitectura (relación de los procesadores, caches)
 - ▣ Sincronización

Software paralelo

24

- Para desarrollar un algoritmo paralelo el primer paso es descomponer el problema en sus partes concurrentes.
- La descomposición puede ser:
 - ▣ Funcional (Paralelismo de Control o de Tareas)
 - ▣ De datos o de dominio (Paralelismo de Datos)


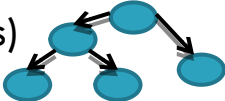
Paralelismo de control

25

- El paralelismo de control o de tareas (descomposición funcional) se logra mediante la aplicación simultánea de diferentes operaciones sobre diferentes elementos de datos.
- Es más adecuado para arquitecturas de tipo MIMD.
- Un ejemplo son los pipelines de las arquitecturas:
 - ▣ Cómputo dividido en etapas
 - ▣ Cada etapa trabaja en paralelos realizando distintas operaciones tomando como entrada los datos producidos por la etapa anterior

Paralelismo de datos

26

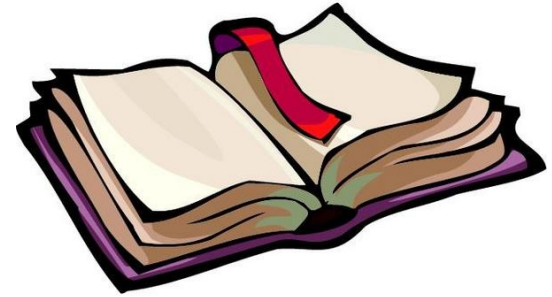
- El paralelismo de datos (descomposición de datos o de dominio) se caracteriza por la ejecución simultánea de la misma operación sobre diferentes elementos de datos.
- Es más adecuado para arquitecturas de tipo SIMD.
- Los datos pueden tener estructuras:
 - Regulares (arreglos, matrices) 
 - Irregulares (grafos) 

Las **GPUs** suelen adaptarse mejor al paralelismo de datos sobre estructuras regulares.

Agenda

27

- I.** *Introducción al concepto de HPC*
- II.** *La GPU como arquitectura paralela*
 - I.** *Mecanismos de control (Taxonomía de Flynn)*
 - II.** *Modelos de comunicación*
- III.** *La GPU y software paralelo*
 - I.** *Descomposición de problemas*
 - II.** *Características de las aplicaciones*
 - III.** *Herramientas de programación sobre GPU y modelos de programación Híbridos*
- IV.** *Resumen de características*



Características de aplicaciones

28

- Un aspecto importante a considerar son las características de las aplicaciones:
 - ▣ Intensivas en cómputo (CPU Bound)
 - ▣ Intensivas en memoria (Memory Bound)
 - ▣ Intensivas en Entrada-Salida (E/S Bound)
 - ▣ Híbridos (Fases)

Características de aplicaciones

29

- Las aplicaciones **intensivas en cómputo** llevan al límite a la CPU.
- La mayoría de las aplicaciones de cómputo científico.
- Algoritmos con orden polinómico alto o exponenciales:
 - ▣ Álgebra lineal
 - ▣ Audio, imágenes y/o video

Características de aplicaciones

30

- Las aplicaciones **intensivas en memoria** llevan al límite al sistema de memoria.
- El límite puede estar puesto por:
 - ▣ Ancho de banda
 - ▣ Capacidad
- Algoritmos que no utilizan unidades funcionales de punto flotante o que generan muchos datos intermedios:
 - ▣ Ordenación
 - ▣ Búsquedas
 - ▣ Algunos algoritmos de orden exponencial (grafos)

Características de aplicaciones

31

- Las aplicaciones **intensivas en Entrada-Salida** llevan al límite al sistema de entrada salida.
- Los sistemas de entrada-salida suelen ser las partes más lentas del sistema.
- Suelen pasar más tiempo esperando por la lectura/escritura de datos que procesándolos.
- Algoritmos que leen entradas de datos muy grandes o generan gran cantidad de información:
 - ▣ Simulaciones.
 - ▣ Data maining.
 - ▣ Big data (Map-Reduce).

Características de aplicaciones

32

Event	Real time	Scaled time
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access	120 ns	6 min
Solid-state disk I/O	50-150 μ s	2-6 days
Rotational disk I/O	1-10 ms	1-12 months

Características de aplicaciones

33

Event	Real time	Scaled time
Internet: SF to NYC	40 ms	4 years
Internet: SF to UK	81 ms	8 years
Internet: SF to Australia	183 ms	19 years
OS virtualization reboot	4 s	423 years
SCSI command time-out	30 s	3000 years
Hardware virtualization reboot	40 s	4000 years
Physical system reboot	5 m	32 millenia

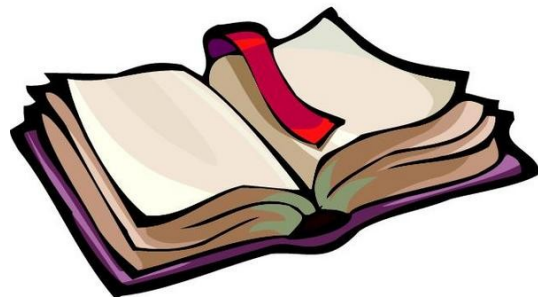
Características de aplicaciones

34

- ❑ Las aplicaciones intensivas en CPU se benefician más de las **GPUs**.
- ❑ Esto por las características de las GPU como arquitecturas y su modelo de programación.
- ❑ Las aplicaciones intensivas en memoria se benefician pero no de la misma forma.
- ❑ Las GPU no tienen posibilidad de hacer entrada-salida por esto las aplicaciones intensivas en este aspecto requieren relacionarse fuertemente con la CPU.

Agenda

- I.** *Introducción al concepto de HPC*
- II.** *La GPU como arquitectura paralela*
 - I.** *Mecanismos de control (Taxonomía de Flynn)*
 - II.** *Modelos de comunicación*
- III.** *La GPU y software paralelo*
 - I.** *Descomposición de problemas*
 - II.** *Características de las aplicaciones*
 - III.** *Herramientas de programación sobre GPU y modelos de programación Híbridos*
- IV.** *Resumen de características*



Programación Paralela: Herramientas

36

- Programación en memoria distribuida:

- ▣ MPI.

- Programación en memoria compartida:

- ▣ Multiprocesadores/Multicores:

- Pthreads.

- OpenMP.

- Cilk.

- ▣ **GPUs**

- Cuda.

- OpenCL.

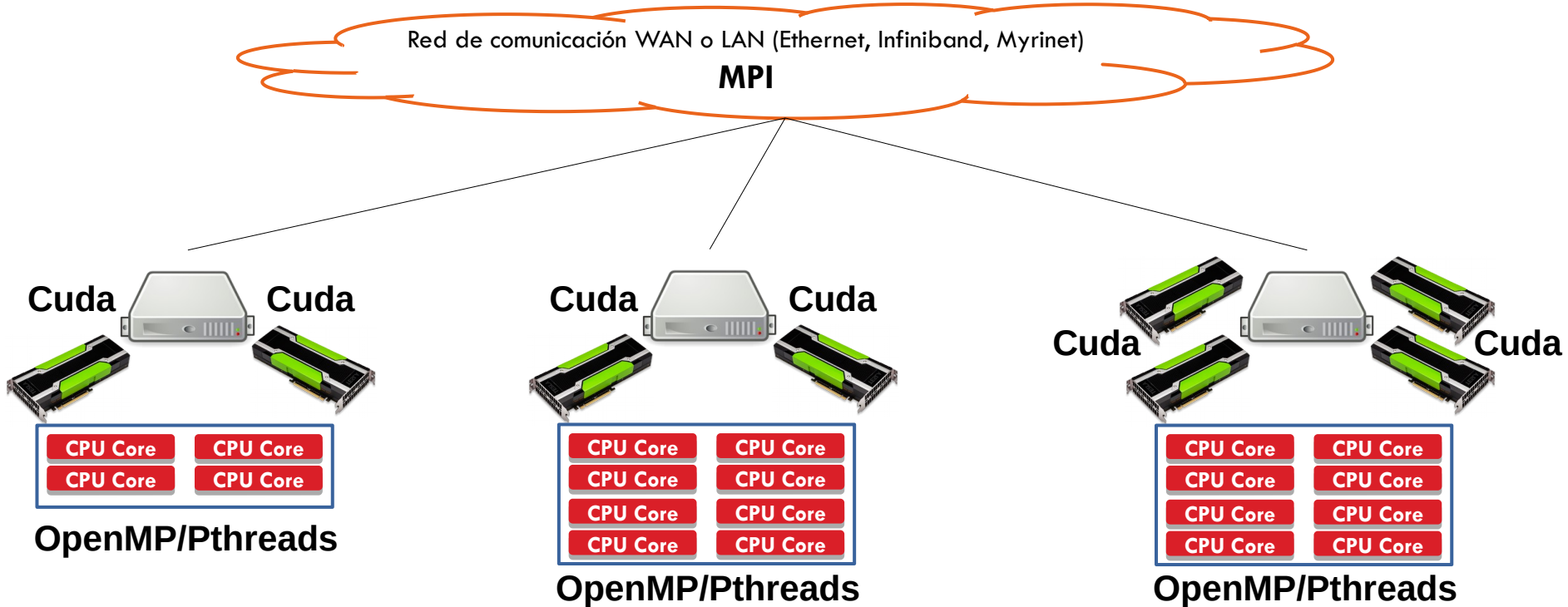
Evolución de sistemas paralelos

37

- Multiprocesadores: en su momento muy costosos.
- Crecimiento de las redes permiten conectar máquinas en red formando un cluster:
 - ▣ Programación Distribuida (PVM, MPI, Sockets, RMI)
- Limitación en la velocidad de los procesadores uncore impulsó los multicores:
 - ▣ Varios cores con diversas jerarquías de memoria
 - ▣ Programación Memoria Compartida (Pthreads, OpenMP)
- Surgimiento de las GPUs para programación de propósito general (GPGPU)
 - ▣ Nvidia (CUDA, OpenCL)
 - ▣ ATI-AMD (OpenCL)
- Modelos híbridos

Modelo híbrido

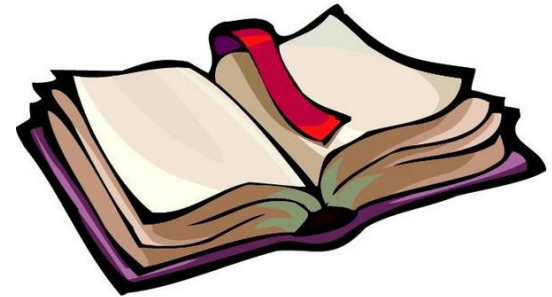
38



Agenda

39

- I.** *Introducción al concepto de HPC*
- II.** *La GPU como arquitectura paralela*
 - I.** *Mecanismos de control (Taxonomía de Flynn)*
 - II.** *Modelos de comunicación*
- III.** *La GPU y software paralelo*
 - I.** *Descomposición de problemas*
 - II.** *Características de las aplicaciones*
 - III.** *Herramientas de programación sobre GPU y modelos de programación Híbridos*
- IV.** *Resumen de características*



Características paralelas de las GPUs

40

- ❑ Arquitectura de memoria compartida.
- ❑ Modelo SIMD - STMD.
- ❑ Se adaptan mejor a paralelismo de datos (regulares)
- ❑ Adecuada para aplicaciones intensivas en CPU.
- ❑ Modelos híbridos - Gran potencia de cómputo:
 - ▣ Máquinas Multi-GPU
 - ▣ Maquinas Multicore/MultiGPU (OpenMP – Cuda, OpenMP – OpenCL, etc)
 - ▣ Cluster de Multicore/MultiGPU (OpenMP – MPI – Cuda, OpenMP – MPI – OpenCL, etc)