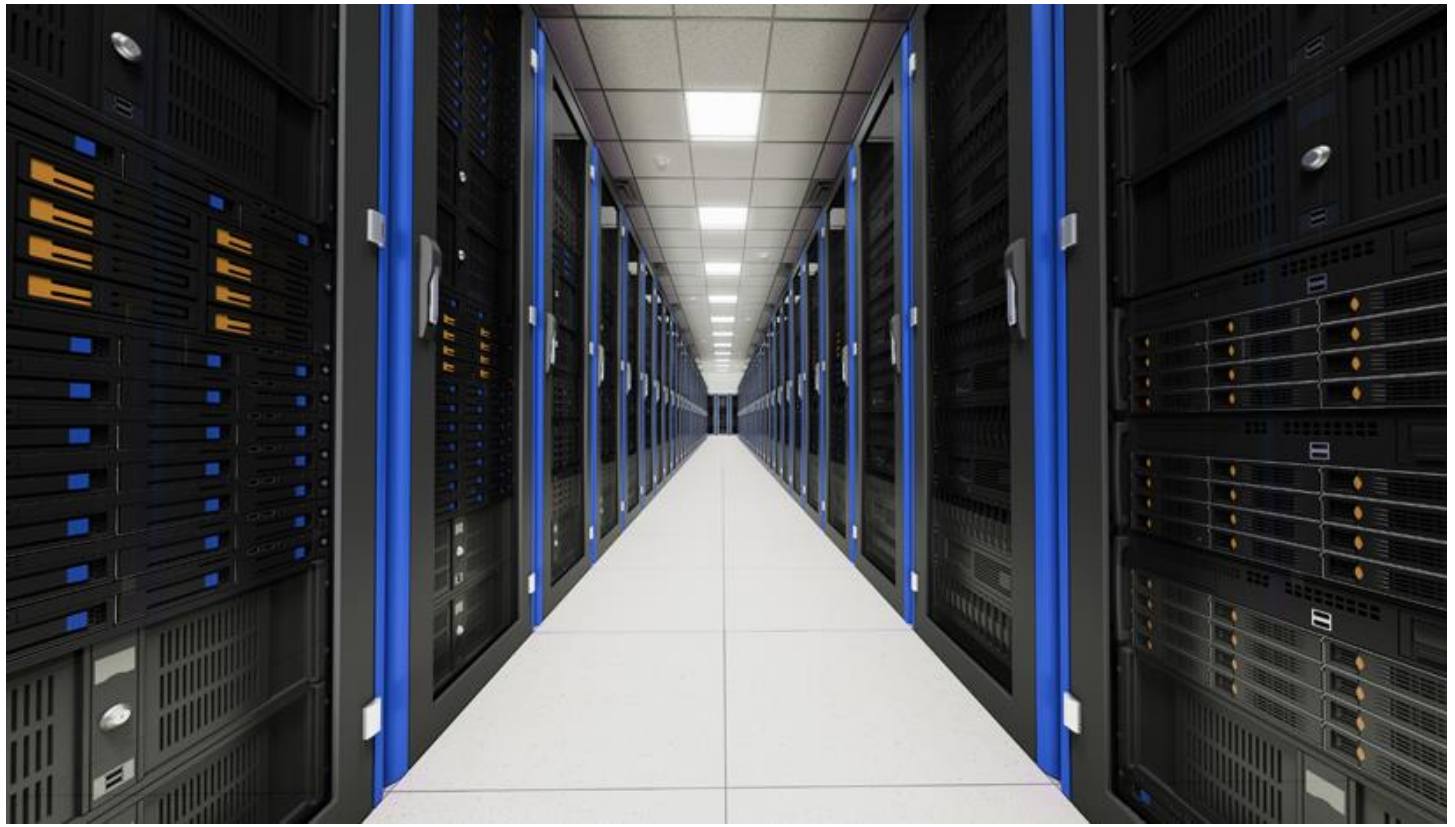
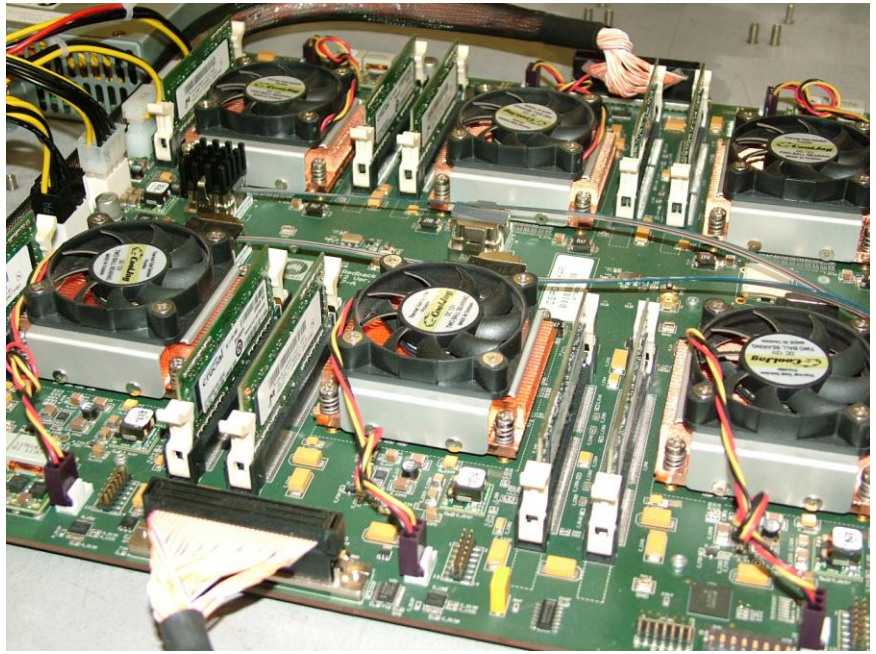
1. Introducción a las arquitecturas masivamente paralelas.
High Performance Computing

High Performance Computing (HPC)



- Clúster HPC
  - Múltiples equipos interconectados
  - Equipos de alto rendimiento
  - Red de alto rendimiento
  - Grandes espacios de memoria compartida

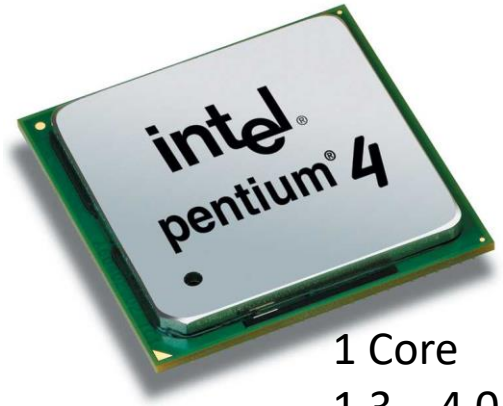
High Performance Computing (HPC)



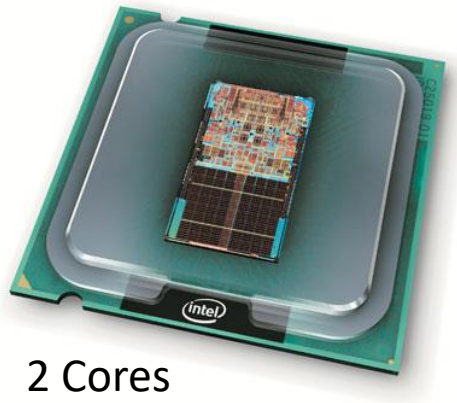
- Antes

- Ahora

GPU vs CPU

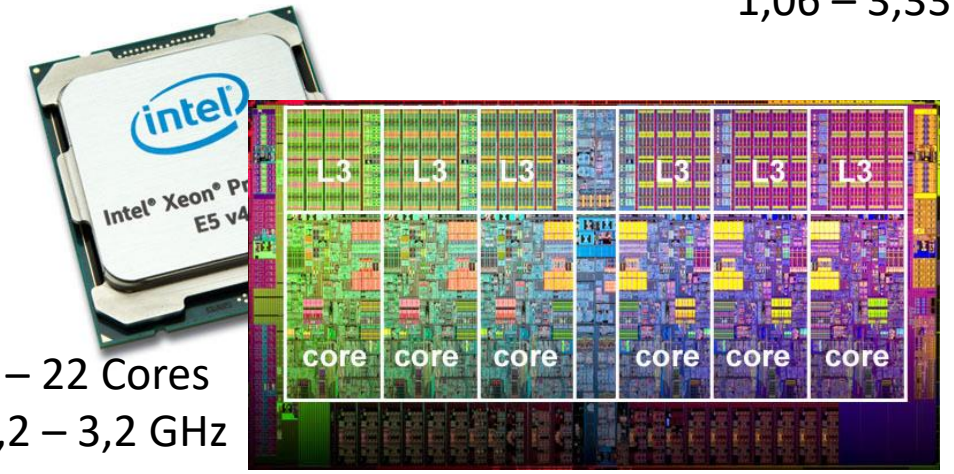


1 Core
1,3 – 4,0 GHz

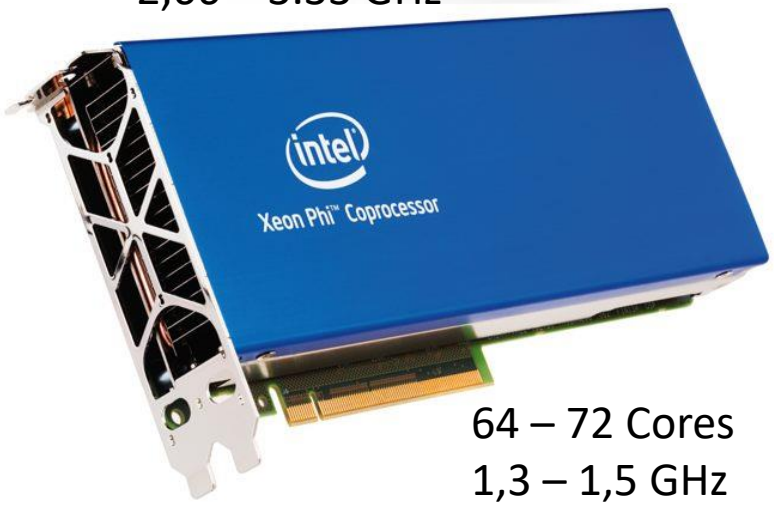


2 Cores
1,06 – 3,33 GHz

4 Cores
2,66 – 3.33 GHz

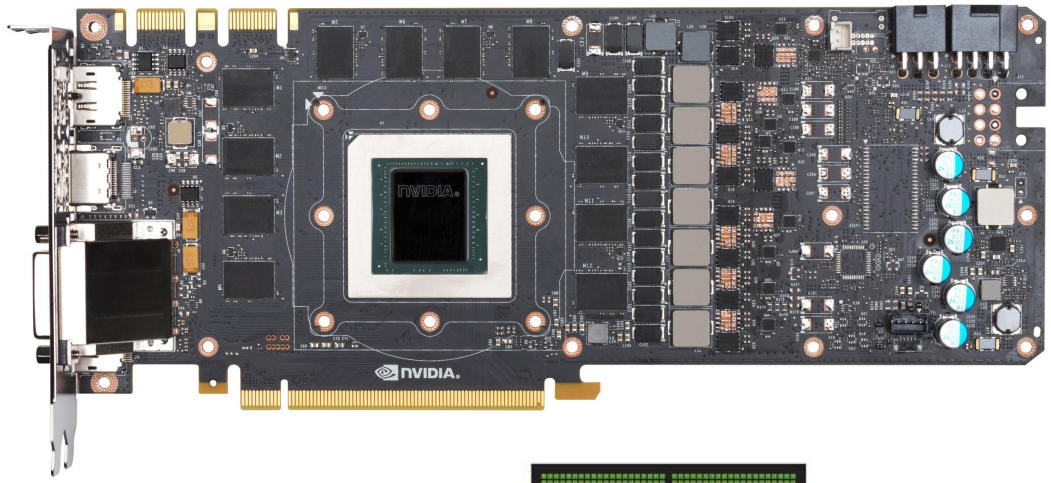


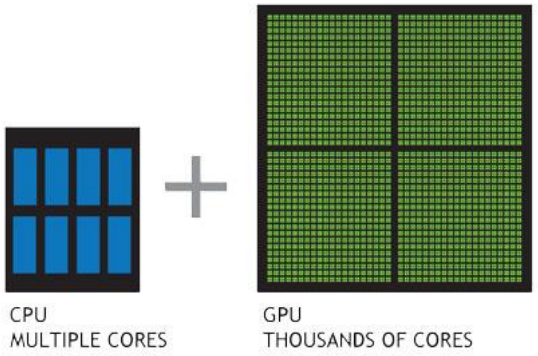
6 – 22 Cores
2,2 – 3,2 GHz



64 – 72 Cores
1,3 – 1,5 GHz

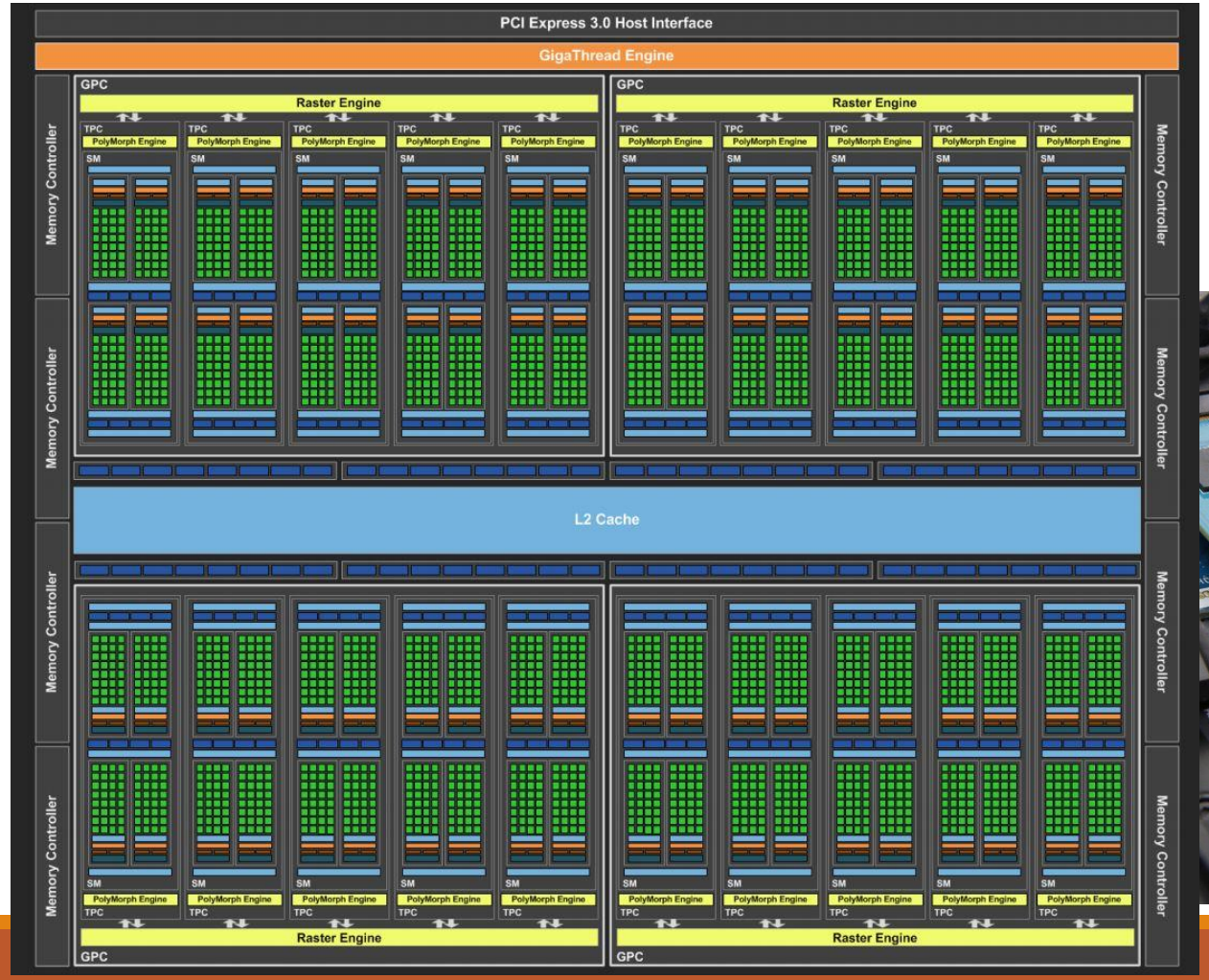
GPU





- GPU: Graphic Procesor Unit
- Diseñadas para gráficos en tiempo real
- Es una colección de procesadores organizados de acuerdo con el cauce gráfico

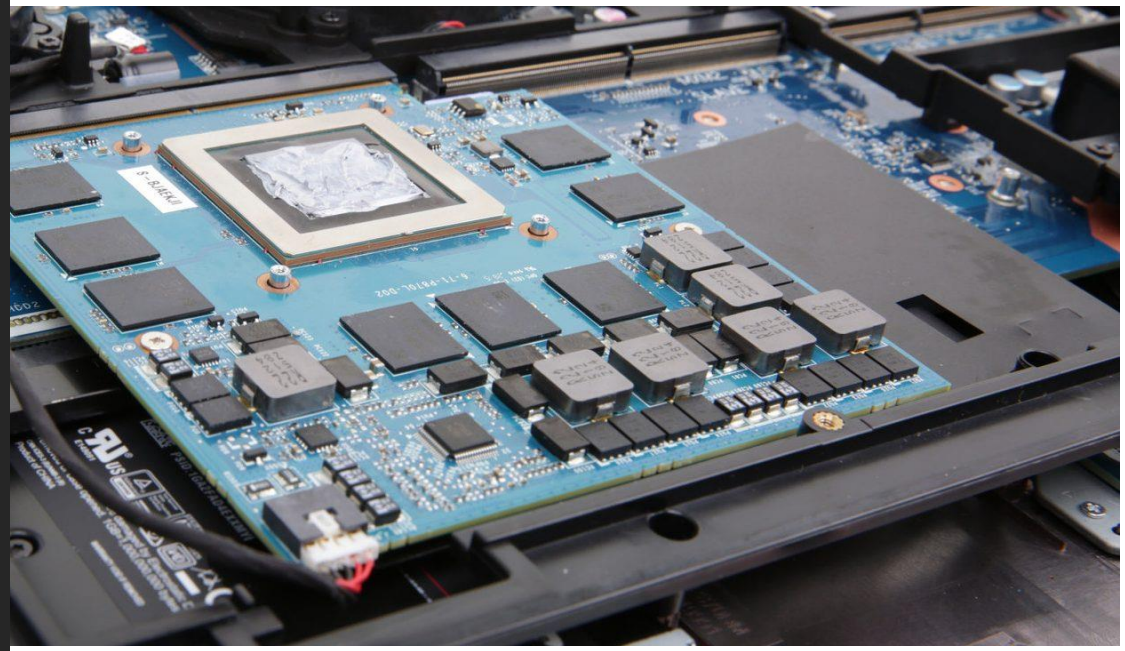
GPU vs CPU



20 SM *2.048 threads/SM =

40.960 “cores”

1,73GHz max



GPUs para calculo paralelo

- Equipos convencionales disponen de un dispositivo masivamente paralelo. La GPU
- Se puede utilizar para calculo genérico
- Precisión de al menos 32 bits

Cluster de GPUs. Aplicaciones Practicas

- Render
  - Millones de cálculos matemáticos para obtener una imagen a partir de un modelo 3D
    - Sombras, iluminación, transparencia, reflexión, desenfoques, antialiasing… Hacen crecer el numero de cálculos. Además hay que tener en cuenta la complejidad de los objetos poligonales
  - La creación de películas requiere mucho tiempo y capacidad de proceso
    - Cada segundo de película tiene al menos 24 frames
- Render Paralelo
  - Muchas tareas de render se pueden realizar en paralelo
    - Permite renderizar frames mas rápido
    - Permite visualizar mayores conjuntos de datos

Cluster de GPUs. Aplicaciones Practicas

- Granja de render
  - Conjunto de ordenadores construido para representar imágenes generadas por ordenador (CGI)
    - Distribuye el render entre varios equipos
    - Utiliza procesamiento por lotes y balanceo de carga
  - Los render se ejecutan en CPU, aunque actualmente los motores de render están empezando a migrar sus funciones a GPU

Cluster de GPUs. Aplicaciones Practicas

Caso de estudio: Pixar

- Toy Story, 1995
  - 15 horas en renderizar cada frame
  - 114.000 frames, 77 minutos de película
  - Un único equipo (de la época) habría tardado 43 años
  - Granja de render:
    - Inicialmente 53 nodos. Tardó 20 meses
    - Creció hasta los 300 nodos
- Pixar cuenta en la actualidad con 23.000 procesadores

¡Suficiente para renderizar la película original en tiempo real!

Cluster de GPUs. Aplicaciones Practicas

Aplicaciones prácticas. Ciberseguridad

- Las tarjetas gráficas ofrecen una gran capacidad computacional
- “Romper” una contraseña por fuerza bruta necesita una gran cantidad de cálculos similares
  - 1 contraseña de 8 caracteres alfanuméricos tiene 76^8 (mas de 1000 billones) de combinaciones (sin caracteres especiales)
  - 25 tarjetas AMD Radeon HD6990 pueden realizar 95^8 (mas de 8000 billones) de combinaciones en menos de 6 horas.
  - Obtuvo el 90% de las contraseñas de Linkedin a partir de los hashes almacenados en la DB, unos 6 billones de contraseñas (junio de 2012)

Cluster de GPUs. Aplicaciones Practicas

Aplicaciones prácticas. Ciberseguridad

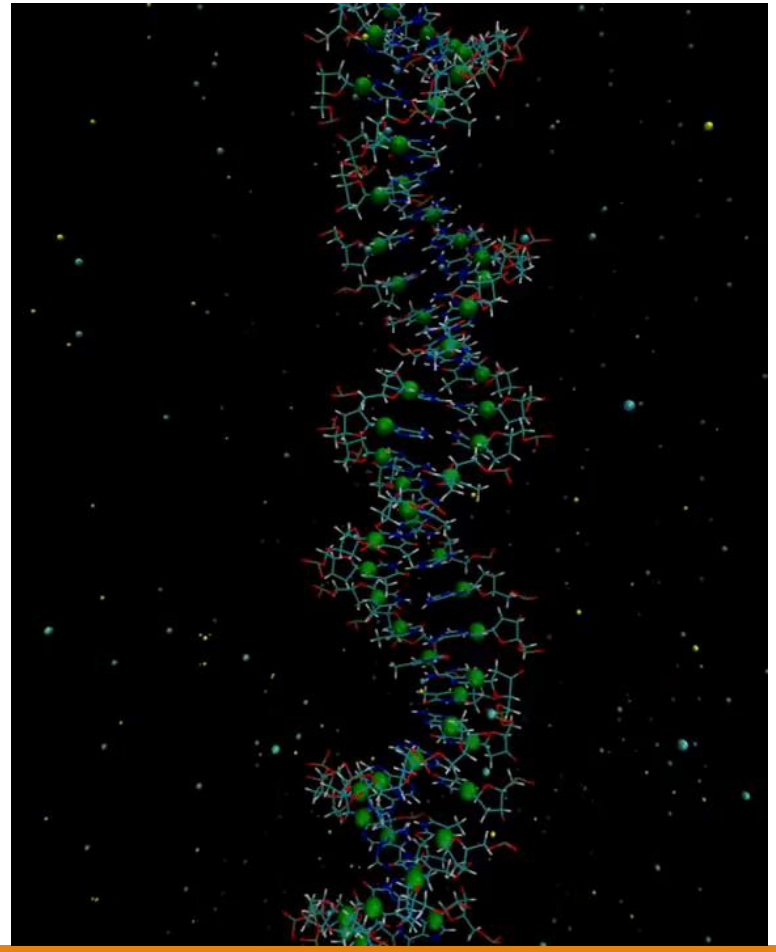
- Congreso BlackHat 2011
  - Se presenta un software que de atacar por fuerza bruta claves WPA2 en servicios de cloud computing
  - En el cloud computing de Amazon es capaz de “romper” la contraseña WPA2 en menos de 50 minutos
- Congreso BlackHat 2012
  - Se presenta otro software, capaz de “romper” la seguridad de los protocolos MS-CHAPv2(EAP/PEAP) en servicios de cloud computing

Cluster de GPUs. Aplicaciones Practicas

Simulación de fluidos

Cluster de GPUs. Aplicaciones Practicas

Simulación de dinámica molecular



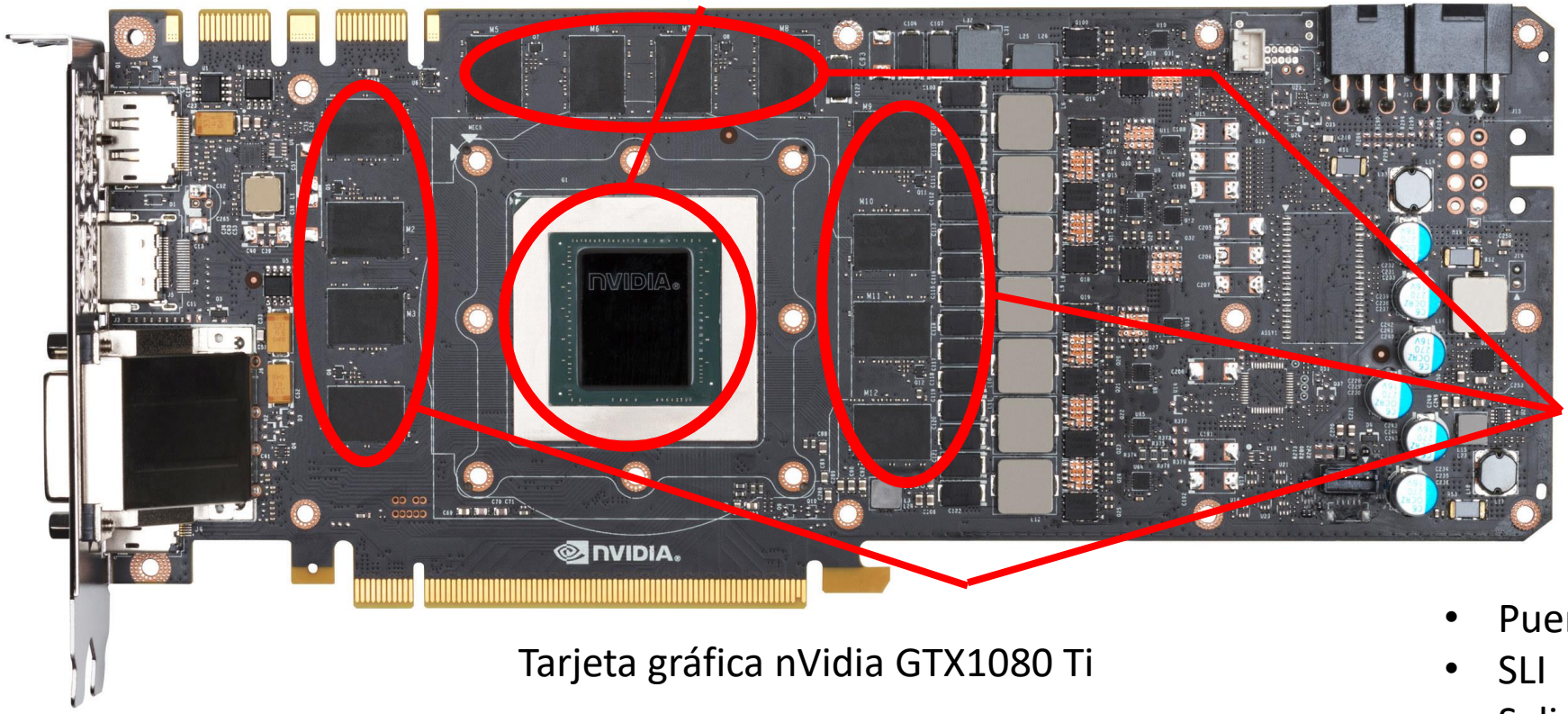
Cluster de GPUs. Aplicaciones Practicas

Simulación de dinámica molecular. Caso de estudio

- La simulación de dinámica molecular supone tratar con una gran cantidad de datos para obtener resultados muy pequeños
  - Cada paso de simulación que se realiza es del orden de 1 fs (milbillonésima parte de 1 s)
  - Un grano de arena se compone de 2,2 trillones de átomos
    - Almacenar 2,2 trillones de posiciones (**float3**) supone un uso de mas de **24 millones de Terabytes (**24.586.915.970 GigaBytes**)**

GPU

Chip gráfico (GPU)

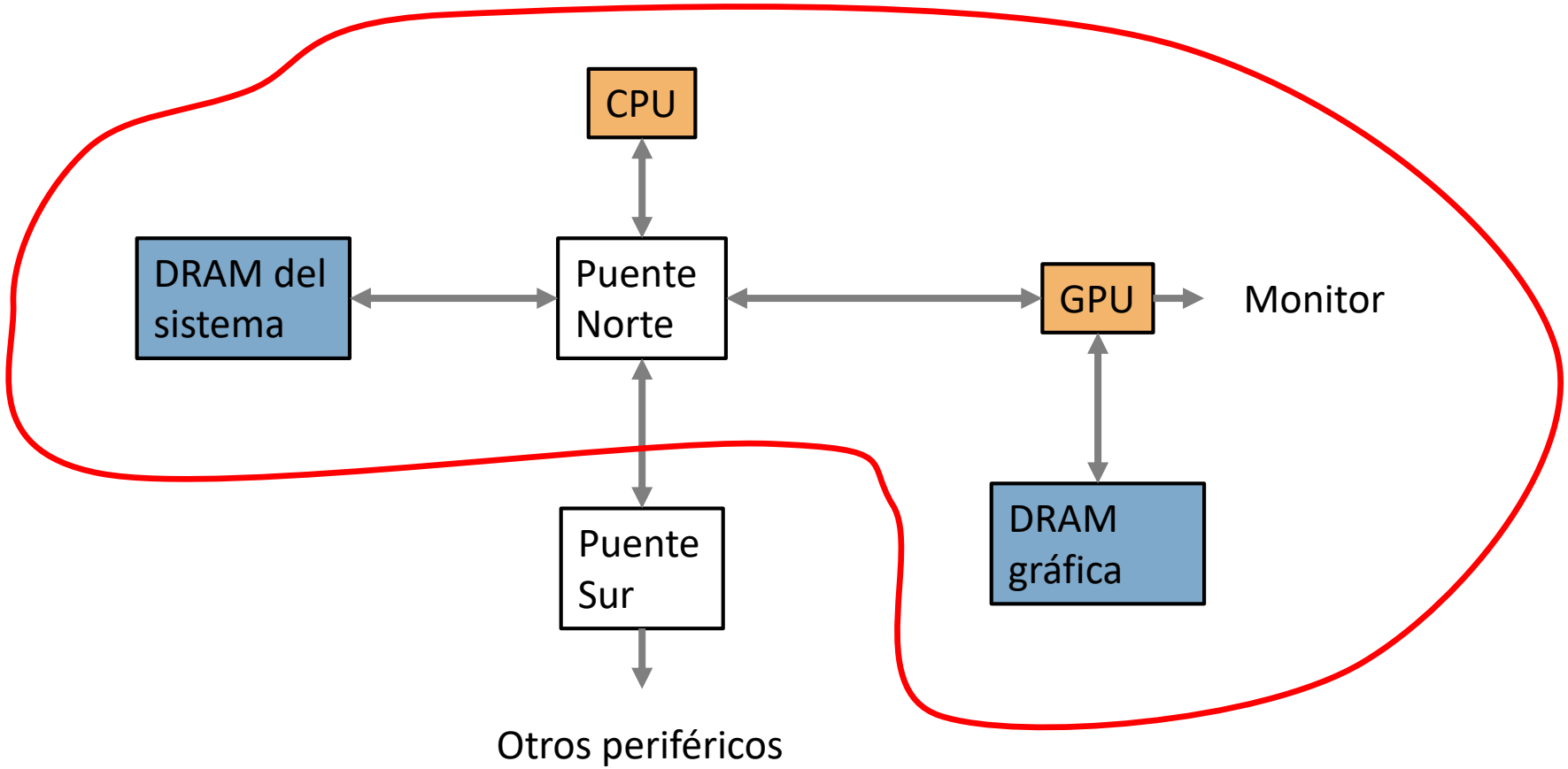


Memoria de video

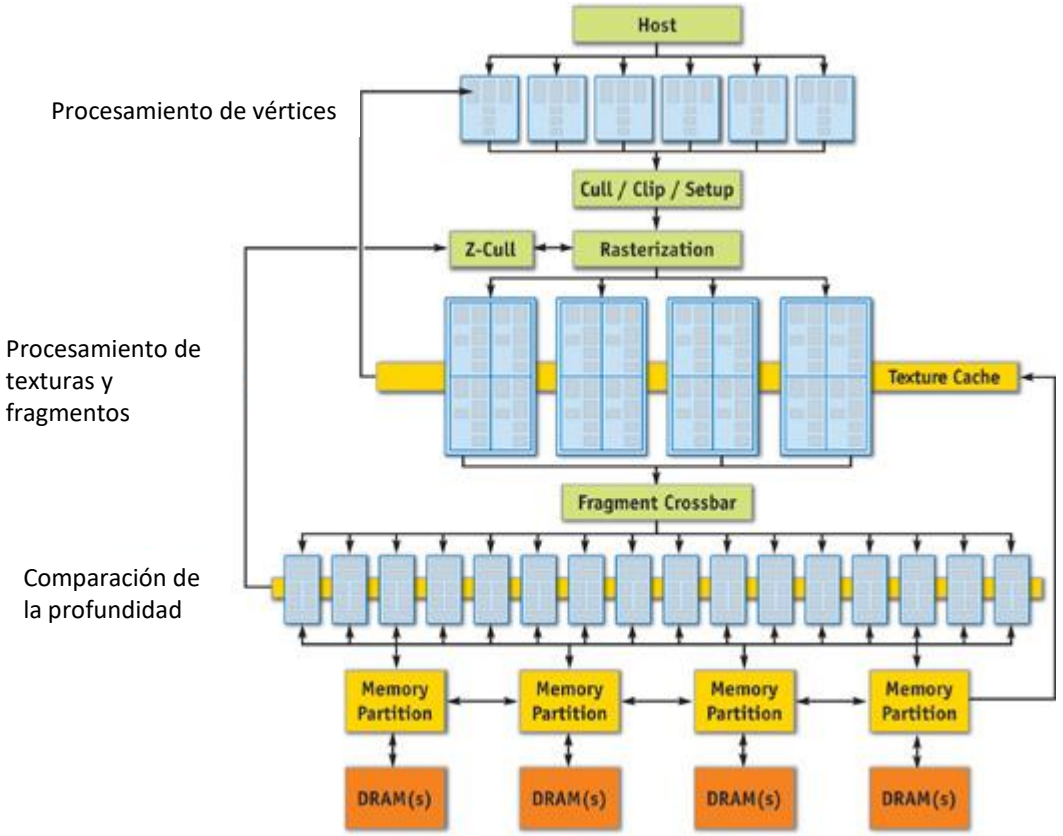
Tarjeta gráfica nVidia GTX1080 Ti

- Puerto PCIe
- SLI
- Salidas de video…

Buses y comunicación



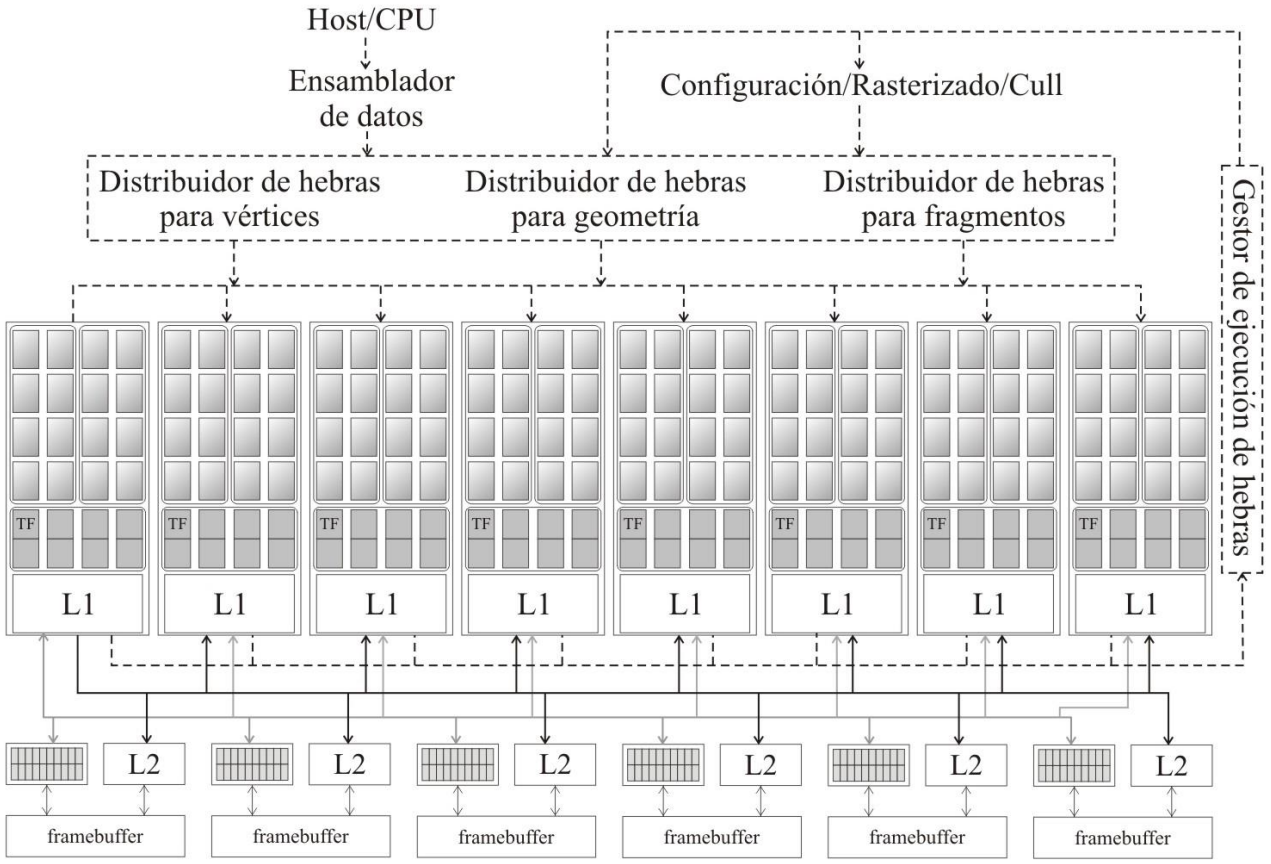
Arquitectura gráfica tradicional



Arquitectura unificada

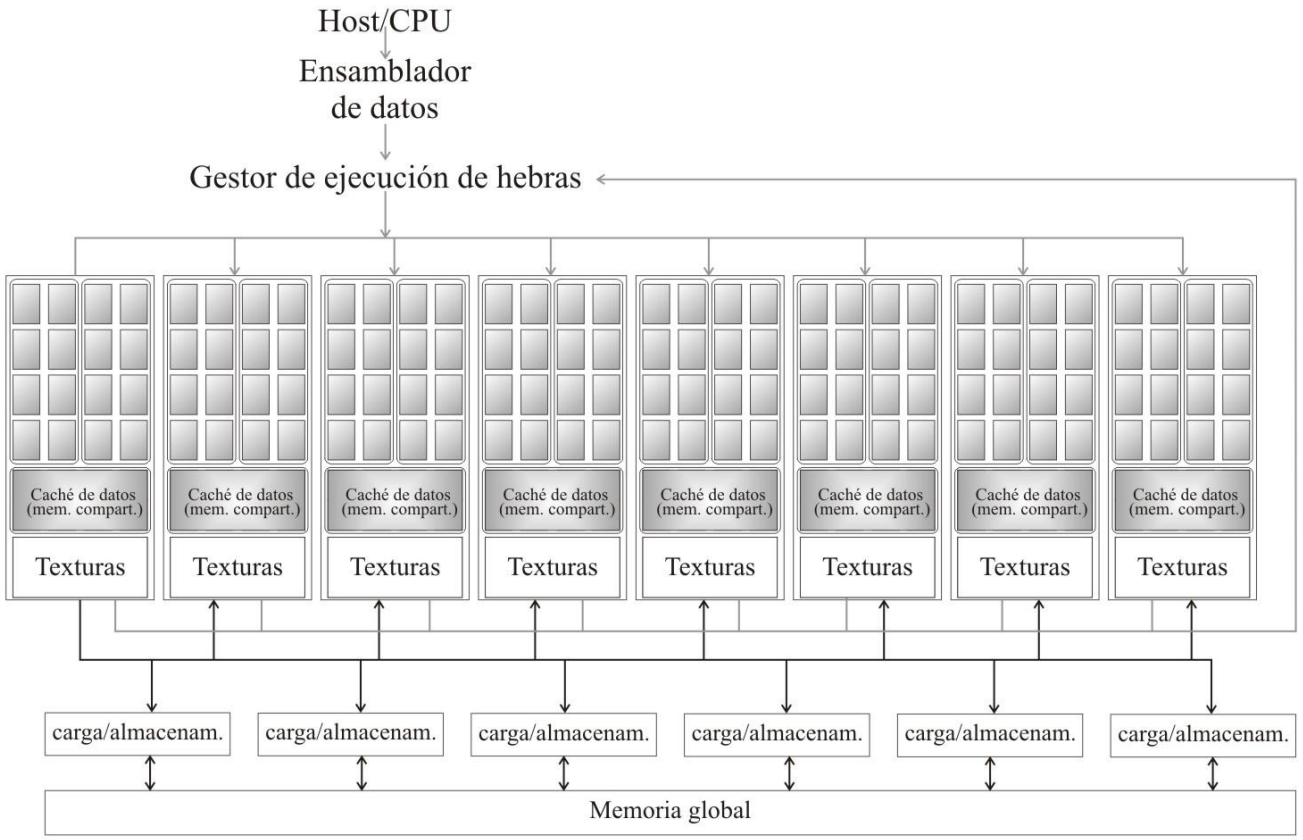
- Procesadores de propósito general
- Memoria de propósito general

Arquitectura unificada



- Arquitectura Gráfica

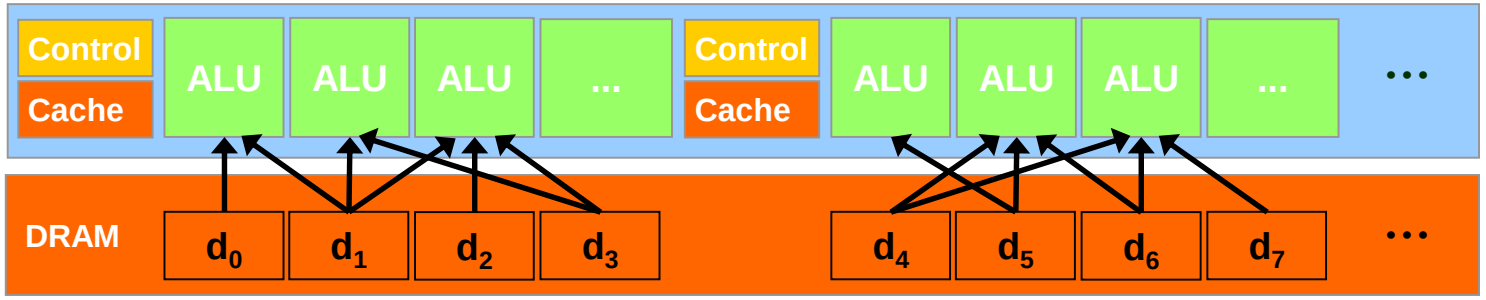
Arquitectura unificada



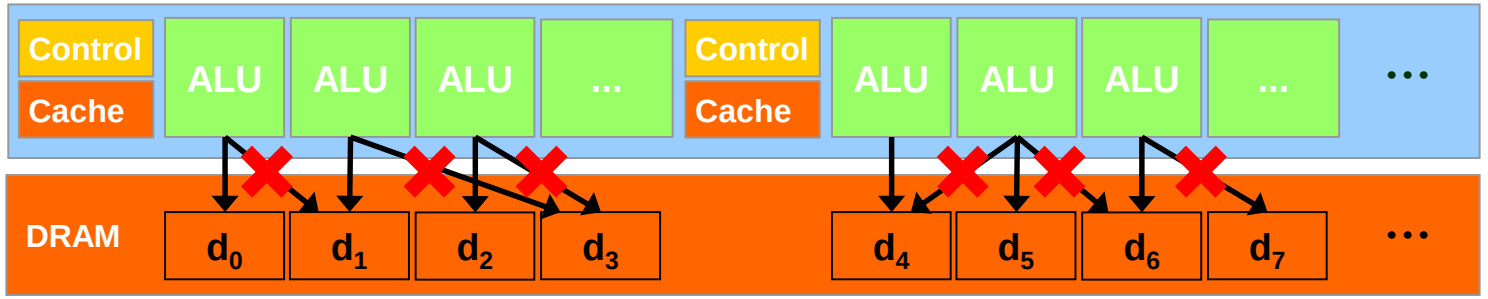
- Arquitectura computacional

Antiguas limitaciones hardware

- Problemas de accesos a memoria
  - Gathering, podemos leer datos que corresponden a otros fragmentos

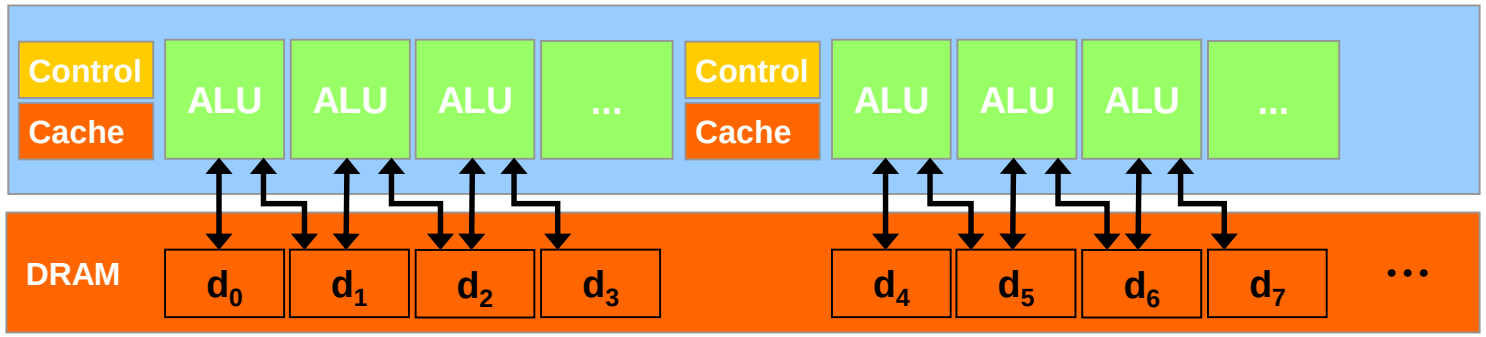


  - No hay scattering, es decir, no podemos escribir en otros fragmentos



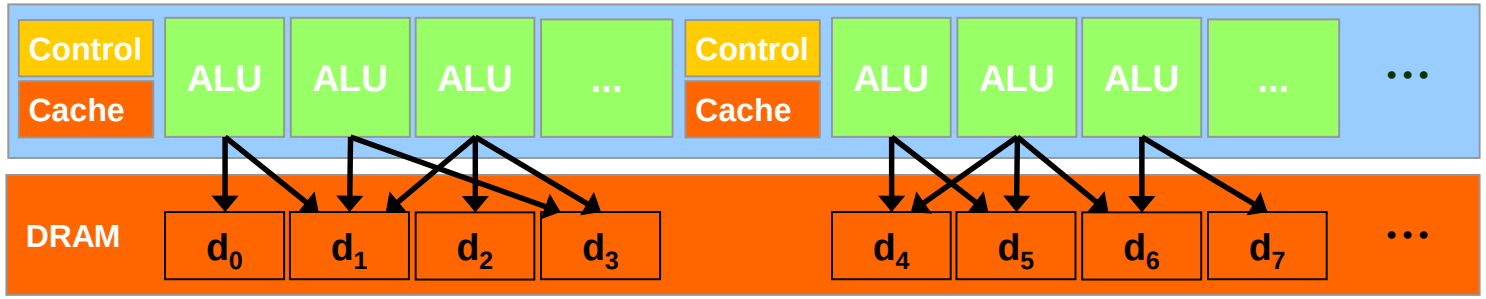
Antiguas limitaciones hardware

- Las aplicaciones están limitadas por el ancho de banda en los accesos a memoria



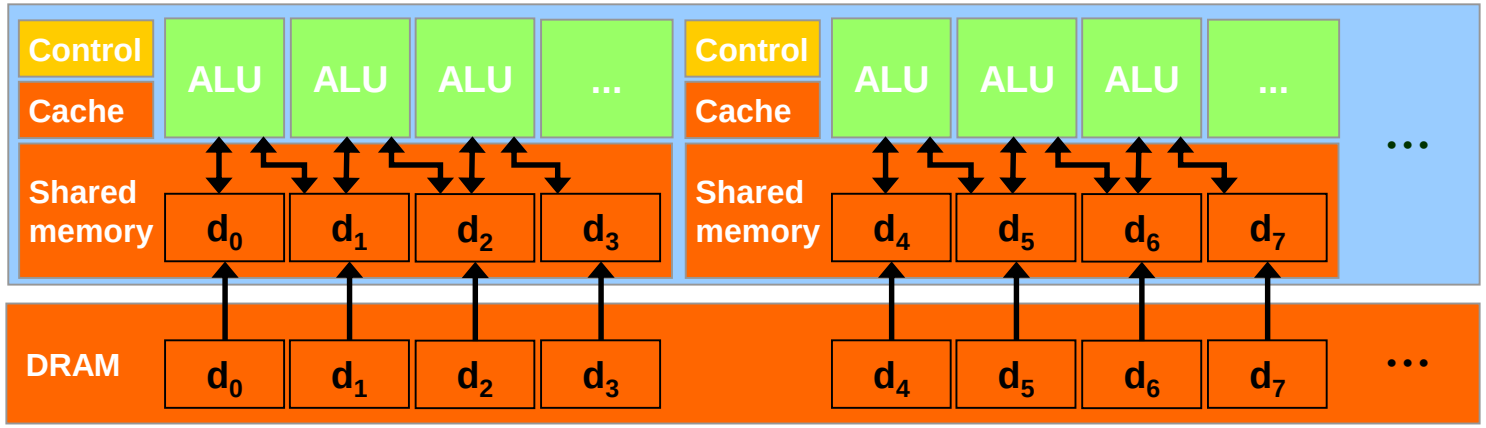
Novedades en arquitectura unificada

- Scatter: Cada procesador ya puede escribir en otras zonas de memoria



Novedades en arquitectura unificada

- Memoria compartida OnChip: Tenemos acceso a zonas de memoria compartidas para los multiprocesadores



CUDA

CUDA

- CUDA: Compute Unified Device Architecture (Arquitectura Unificada de Dispositivos de Computo)
- Según la web de nVidia: “CUDA es una arquitectura de cálculo paralelo de NVIDIA que aprovecha la gran potencia de la GPU (unidad de procesamiento gráfico) para proporcionar un incremento extraordinario del rendimiento del sistema.”
- Ofrece un modelo de programación de propósito general. Convierte la GPU en un procesador especifico masivamente paralelo, con miles de hilos trabajando simultáneamente sobre un mismo problema

CUDA abstrae el cauce gráfico para convertir la GPU en un procesador paralelo de propósito general.

CUDA

- Ofrece un driver y un lenguaje específicos de nVidia
  - Esta optimizado para problemas de computación
  - No requiere el uso de una API gráfica (OpenGL, DirectX)
  - Puede compartir datos con OpenGL y/o DirectX
  - Garantiza la mejor velocidad en carga y descarga de datos de la memoria de video
- Solución propia de nVidia. Para ATI-AMD o Intel existen alternativas como OpenCL

Modelos de programación de CUDA

- Podemos ver la GPU como un dispositivo (device)
  - Coprocesador de la CPU (host)
  - Propia DRAM (memoria de video)
  - Puede ejecutar muchos hilos en paralelo
- Los hilos en GPU y CPU son muy diferentes:
  - En GPU son muy ligeros
    - Apenas hay sobrecarga al crearlos
    - El intercambio entre hilos es instantáneo (0 ciclos de reloj)
  - La GPU necesita miles de hilos para ser eficaz
    - Las CPUs multicore solo soportan unos pocos hilos “reales” de ejecución

Modelos de computación paralela

- Memoria compartida
- Modelo de threads
- Modelo de paso de mensajes
- Modelo de datos paralelos
- Modelo híbrido

Modelos de computación paralela

- Memoria compartida
  - Modelo ideal de multiprocesador
    - Computador paralelo consistente en múltiples procesadores idénticos
    - Comparten un único espacio de memoria
    - Tiempo de acceso constante a cualquier dirección de memoria
    - Ejecución síncrona
  - Todos los procesadores ejecutan el mismo programa ejecutable
    - Particularizado de acuerdo a los índices
    - Conjunto de datos diferenciado
  - La comunicación se hace a través de variables de memoria compartida

Modelos de computación paralela

- Modelo de threads
  - Un proceso puede tener múltiples hilos de ejecución
  - Un thread o hilo es la ejecución más pequeña que puede planificar un SO
  - Dentro de un mismo proceso pueden existir múltiples hilos con recursos compartidos
  - Los hilos de un proceso comparten:
    - El código de programa
    - El contexto y la memoria global
    - Los ficheros abiertos
  - Cada hilo tiene en propiedad
    - La pila
    - Los valores de registros

Modelos de computación paralela

- Modelo de paso de mensajes
  - Modelo de computo de memoria distribuida
    - Computador consistente en múltiples procesadores
    - Cada uno tiene su propio espacio de direcciones de memoria: memoria local
    - Tiene acceso constante a la memoria local
    - La comunicación entre cualquier par de nodos se realiza a través de una red de interconexión
    - Ejecución asíncrona
  - Múltiples tareas pueden residir en la misma máquina física y/o en un numero arbitrario de máquinas
  - Las tareas se intercambian datos a través de comunicaciones explícitas mediante el envío y la recepción de mensajes
  - MPI (Message Passing Interface)

Modelos de computación paralela

- Modelo de datos paralelos
  - Un conjunto de tareas trabajan colectivamente en la misma estructura de datos
    - Cada tarea trabaja sobre una partición diferente
  - Las tareas realizan la misma operación sobre su partición
  - En arquitecturas de memoria compartida, acceden a los datos a través de la memoria global
  - En arquitecturas de memoria distribuida los datos se reparten y cada partición reside en la memoria local de la tarea

Modelos de computación paralela

- Modelo híbrido
  - Combinación de algunos modelos descritos previamente
  - Modelo de paso de mensajes (MPI) con el modelo de threads (OpenMP)
    - Los hilos implementan kernels computacionalmente intensos utilizando datos locales de los nodos
    - La comunicación entre procesos situados en diferentes nodos se realiza a través de la red utilizando MPI
  - Programación con GPUs y MPI
    - Las GPUs implementan kernels de alto coste computacional utilizando datos locales
    - La comunicación entre procesos situados entre diferentes nodos se realiza a través de la red utilizando MPI

CUDA

Resumen: Ventajas

- Acceso aleatorio a una memoria direccionable a nivel de byte
  - Un hilo puede acceder a cualquier posición de memoria
- Acceso ilimitado a posiciones de memoria
  - Se puede leer/escribir donde sea necesario
- Jerarquía de memoria para optimizar el uso del ancho de banda
  - Memoria compartida por bloque y sincronización de hilos
- No hace falta ser un “gurú” de la programación de shaders para programar GPGPU
  - No tenemos sobrecarga por utilizar la API gráfica