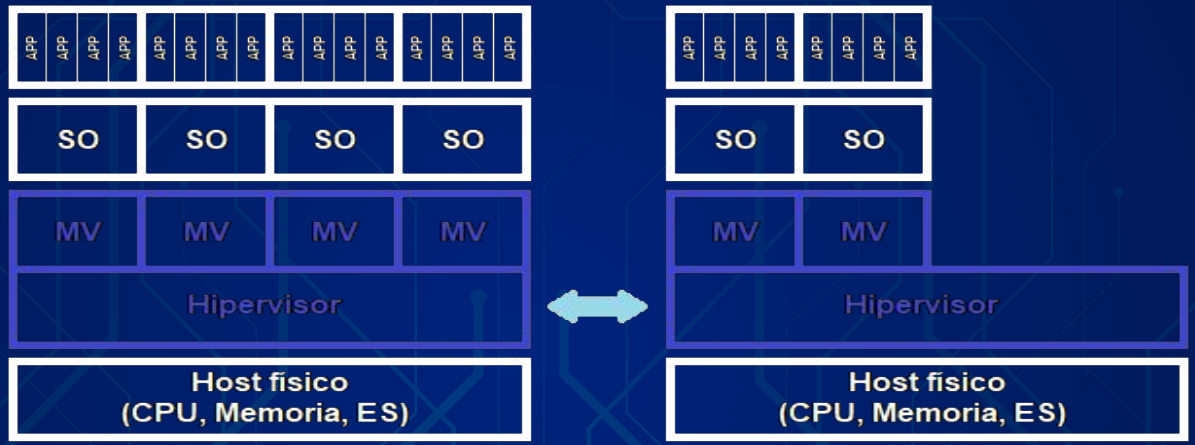
Arquitectura del Computador
y
Sistemas Operativos

Vigesimoséptima Clase

Granjas de Máquinas Virtuales (VM Farms) (1/3)



Una ventaja adicional es que algunos hipervisores pueden trabajar en conjunto con otras instancias del mismo hipervisor corriendo en otro(s) host(s) físico(s).

Estos conjuntos de hipervisores trabajando en conjunto se llaman "*granjas*".

Granjas de Máquinas Virtuales (VM Farms) (2/3)

De esta forma, si un hipervisor detecta que su CPU se está saturando porque las aplicaciones que corren en sus VMs están muy activas, puede “mudar” una máquina completa a otro servidor físico.

Lo hace migrando una “foto” de la memoria de la VM y los datos que el hipervisor tiene de esa VM (sólo algunos Gb).

Obviamente, para que la migración se factible, toda la información en los discos tiene que estar almacenada en un “Storage”, dado que mover todo el contenido de todos los discos sería inviable.

La funcionalidad que permite mover estas máquinas “en caliente” de un host físico a otro se denomina “Machine Migration”.

Granjas de Máquinas Virtuales (VM Farms) (3/3)

Una gran ventaja del machine migration, es la inmunidad ante fallas. Si por algún motivo un host físico tiene problemas, las máquinas virtuales pueden ser pasadas (inclusive mientras ejecutan) a otros miembros de la granja.

El servidor con problemas puede repararse sin que los usuarios noten el problema.

Esta funcionalidad por este motivo también es conocida como HA (High Availability).

GPUs (1/9)

Diferencias y Similitudes con la CPU

Diferencias	Similitudes
Las CPUs son de propósito general, mientras que las GPUs son altamente especializadas en los cálculos de punto flotante necesarios para graficar.	Ambas tienen la arquitectura de los ordenadores vistos hasta ahora. Tienen µP, RAM y E/S.
LAs GPUs tienen muchos más cores más simples.	
Las GPUs están optimizadas para procesamiento paralelo, como las instrucciones SIMD vistas en la Clase 06, las CPUs no tanto (si bien tienen instrucciones algunas).	
Las GPUs suelen tener memorias más rápidas.	
Las CPUs suelen estar en plataformas altamente documentadas para el programador, las GPUs no.	
Las CPUs suelen tener teclas para interactuar directamente con el usuario.	

GPUs (2/9)

Interfaz con la CPU

Como dijimos antes las GPUs no tienen interfaz con el usuario (teclado, mouse, etc). La única forma de indicarles qué hacer es a través de comandos. Estos comandos están agrupados en APIs (Application Programmer Interfaces).

Las APIs más conocidos para comandar una placa de video son:

- OpenGL (Arq. abierta, Silicon Graphics Inc, 1992-2017)
- Vulkan (Arq. abierta, Khronos Group, 2016 a la fecha)
- Direct3D (Microsoft, 1995 a la fecha)
- CUDA (NVIDIA, 2007 a la fecha)

Las primeras tres están específicamente orientadas a mostrar gráficos en pantalla, pero CUDA permite mandar código C/C++/Fortran a la CPU y ejecutarlo.

CUDA es estrictamente para placas NVIDIA, pero existe un API compatible llamado ZLUDA para placas AMD. El grueso del mercado está entre las dos marcas.

GPUs (3/9)

Forma de Graficar

La forma más rápida de graficar objetos es dividir las superficies en polígonos. Por ejemplo si consideramos triángulos, tenemos varias formas de especificárselos a la GPU:

GL_TRIANGLES GL_TRIANGLE_STRIP GL_TRIANGLE_FAN

Como norma general, puede usarse cualquier polígono especificándole a la GPU los vértices, aunque para triángulos y cuadriláteros tiene funciones específicas y es más rápido.

Se puede colocar un color en cada vértice, y la propia GPU calcula el gradiente de colores para toda la superficie automáticamente.

GPUs (4/9)

Capacidad de multiprocesamiento

La especialización es tal, que las GPUs tienen distintos tipos de cores:

- Los núcleos CUDA están destinados a generar las texturas de los poliedros.
- Los núcleos de Tensores son usados para procesar las roto-translaciones y el punto de vista del espectador.
- Los núcleos RT (Ray Tracing) se usan para calcular puntos de intersección entre líneas (rayos) y los otros objetos.

Para tener una idea de dimensionamiento, una placa GeForce RTX 4090 (precio a la fecha aproximadamente USD 1500) tiene:

- 16384 CUDA cores (Referencia: Intel Core i9 14900K de USD 800 tiene 24)
- 512 Tensor cores
- 128 RT cores

Normalmente todos trabajan en paralelo haciendo pipelining: Las CPU determina la posición de los objetos y se los reporta a la GPU. La GPU calcula la textura, luego los rota y traslada para determinar qué se vería desde el punto de vista del observador.
Los procesadores RT se encargan de determinar la efectividad de los “disparos”.

GPUs (5/9)

Ray Tracing (1/2)

Con el gran auge de los juegos, hay dos problemas que son comunes a casi todos ellos:

- Determinar el punto de impacto de un proyectil. Matemáticamente es encontrar la intersección de una recta con los objetos que se muestran.
- Determinar la sombra de un objeto. Matemáticamente es lo mismo, tomando como rayo la recta que parte de la fuente de luz, pasa por el borde de un objeto (el que proyecta la sombra) y se debe determinar dónde impacta (donde se debe mostrar la sombra).

Conceptualmente hay dos formas de acelerar este proceso, reducir su complejidad o hacerlo más rápido:

- Aumento de la velocidad de cálculo
- Reducción de la complejidad del problema

GPUs (6/9)

Ray Tracing (2/2)

Aumento de la velocidad de cálculo

La forma más eficiente es encontrar algoritmos que permitan encontrar en pasos previos los objetos candidatos entre los cuales está la solución.

Reducción de complejidad dado un conjunto de candidatos

- Volúmenes contenedores: La GPU determina primero un poliedro que contiene un grupo de elementos. Si no hay intersección con éste, no puede haber intersección con ninguno de los contenidos.
- División espacial: Se divide el espacio en secciones (pueden ser iguales o no). Luego se calcula por qué sección(es) pasa el rayo y se analizan sólo los poliedros que ocupan las secciones afectadas.
- Técnicas direccionales: Se parte de un cubo centrado en el orígen del rayo. Primero se calcula la posición (cara y coordenadas) sobre las que pasa el rayo. Luego se proyectan todos los objetos sobre las caras del cubo. Una vez hecho ésto se busca la intersección del rayo sólo con los objetos que están en la cara de impacto.

GPUs (7/9)

Breve introducción a CUDA (1/3)

CUDA es un lenguaje específicamente orientado a instrucciones SIMD. Se basa en funciones que se llaman "*kernels*" (nada que ver con el homónimo del SO).

Estos kernels se declaran así:

__global__ void Funcion(void)

Cuando hay que ejecutarla, no se lanzan de a una, sino que se las llama así:

Funcion<<<GridSize, BlockSize>>>();

GridSize: Es una matriz de tres dimensiones que determina la cantidad de instancias a ejecutar.

BlockSize: Determina la cantidad de instancias a ejecutar en paralelo. Permite a la CPU "*elegir*" cuántos cores destinará a este cálculo.

GPUs (8/9)

Breve introducción a CUDA (2/3)

Las funciones ven una serie de variables globales que le permiten determinar qué cálculo deben hacer:

blockDim: Cantidad de threads en cada bloque
gridDim: Dimensiones de la matriz GridSize con que se lanzó el kernel
blockIdx: Número de bloque
threadIdx: Número de thread dentro del bloque

Hay que tener en cuenta que los cores de la GPU no ven la memoria principal de la CPU, por lo tanto todos los datos que requieren para calcular se deben colocar en una buffer de intercambio entre procesadores.

Por lo general ese buffer tiene un conjunto de matrices. La función kernel utiliza las variables globales descritas para determinar qué elemento le toca procesar. Levanta los datos de las matrices y coloca los resultados en otra.

GPUs (9/9)

Breve introducción a CUDA (3/3)

Para los interesados en comenzar a usar esta tecnología, pueden comenzar con:

https://github.com/berkeley-scf/gpu-workshop-2014

Fin

¿Preguntas?