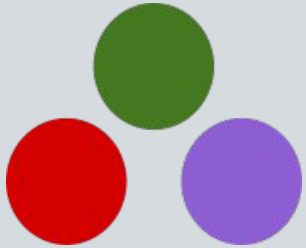
CURSO-TALLER
Introducción a Julia



CUDA
Compute Unified Device Architecture

La GPU

¿Qué es?

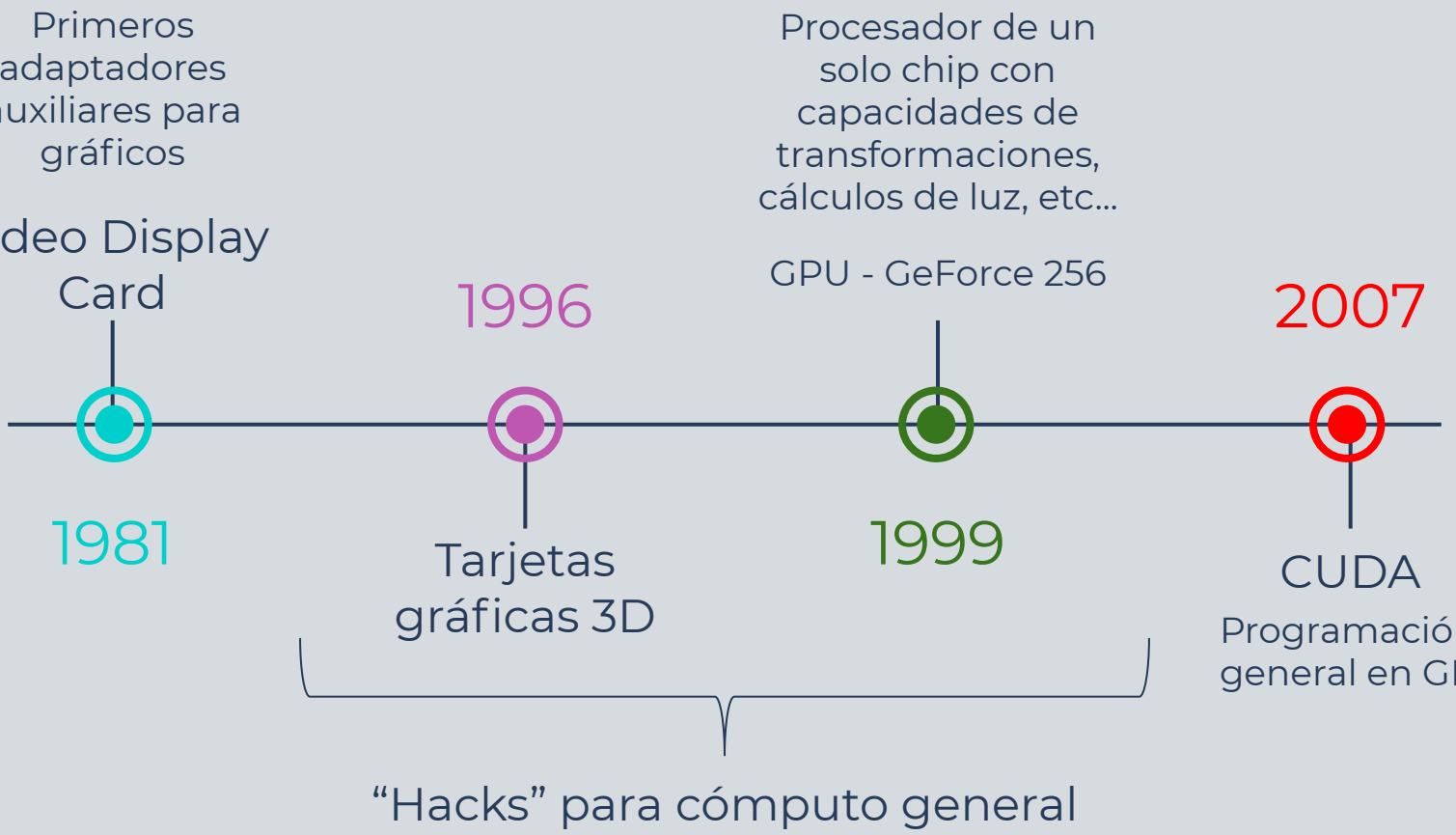
GPU

- Unidad de procesamiento **gráfico**.
- Está diseñada para el procesamiento **paralelo**.
- Propósito original: Acelerar el renderizado de gráficos 3D.
- Obtuvieron mayor flexibilidad y mayores facilidades de programación con el paso del tiempo.

Historia de la GPU

Una breve visita

Historia del GPU



Programación general en GPUs (en el pasado)

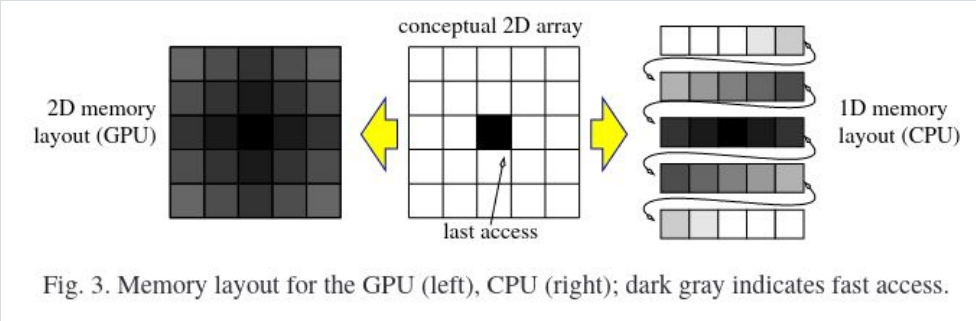


Fig. 3. Memory layout for the GPU (left), CPU (right); dark gray indicates fast access.

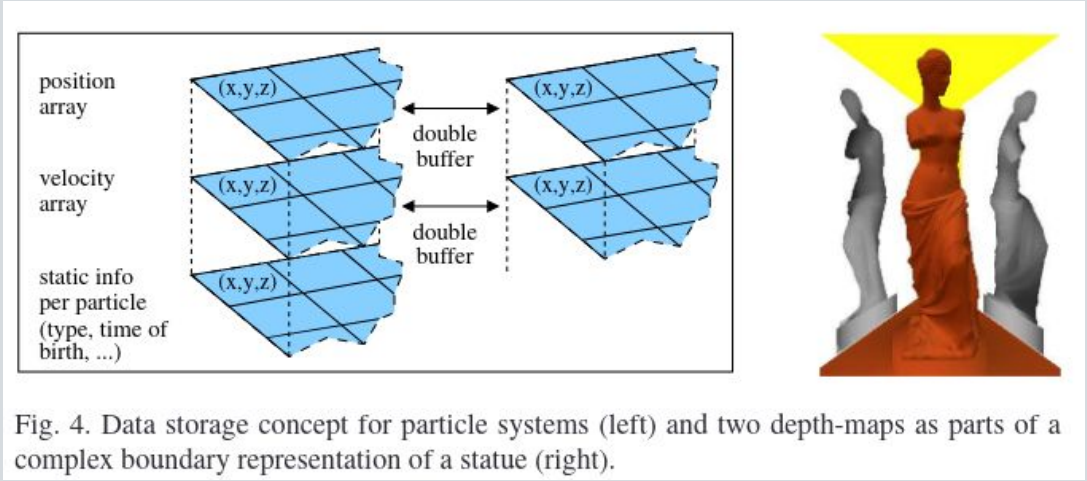


Fig. 4. Data storage concept for particle systems (left) and two depth-maps as parts of a complex boundary representation of a statue (right).

CUDA

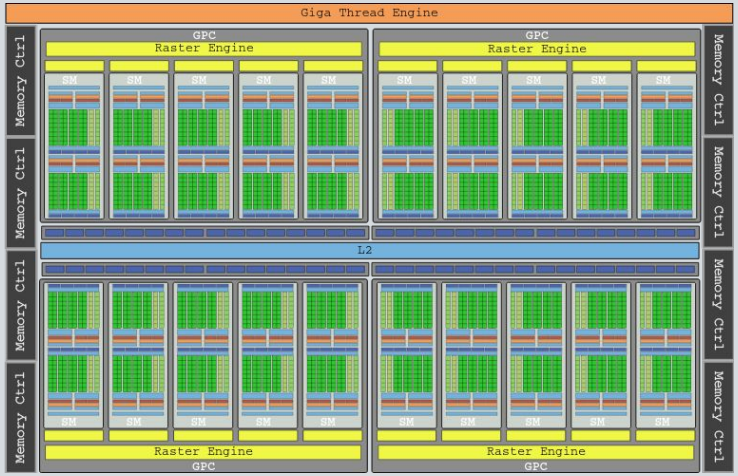
Arquitectura de cómputo paralelo general

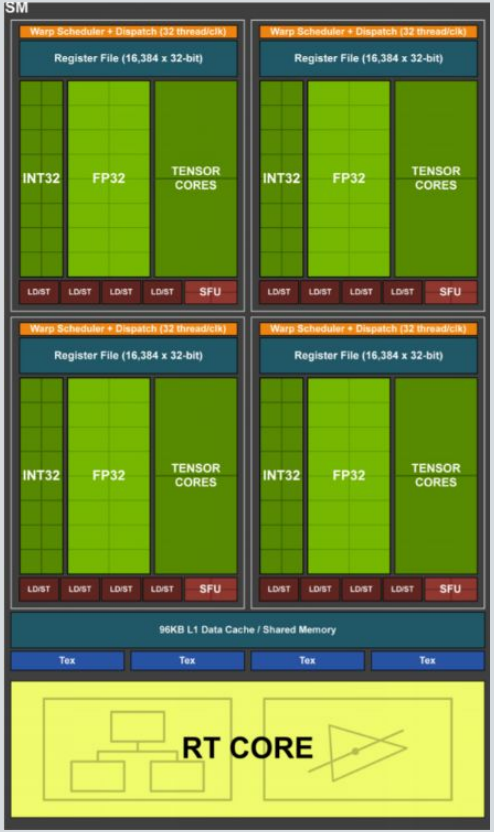
Idea de CUDA

- Las GPU tienen una gran capacidad de paralelismo que solo se está utilizando para gráficos.
- Se puede generar una nueva interfaz para comunicarse con las GPU, que permita el **cómputo general** accesible para cualquiera.
- Brindar facilidades al usuario para desarrollo, así como una documentación completa.
- Gradualmente dar soporte más enfocado a las aplicaciones de interés para el cómputo paralelo.

Paradigma de CUDA (I)

- Se busca tener múltiples procesadores, en los cuales operan muchos hilos. (Miles de hilos al mismo tiempo)

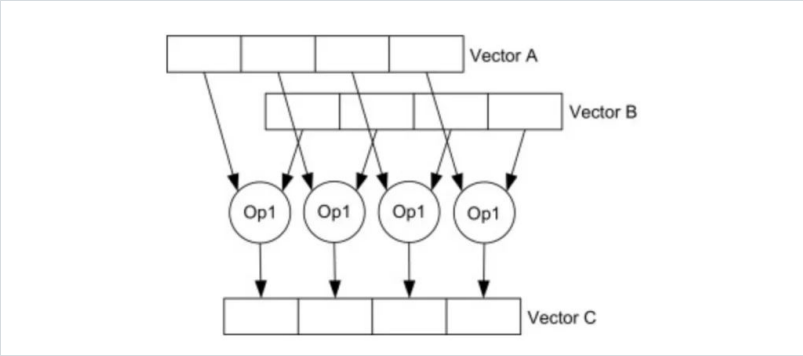


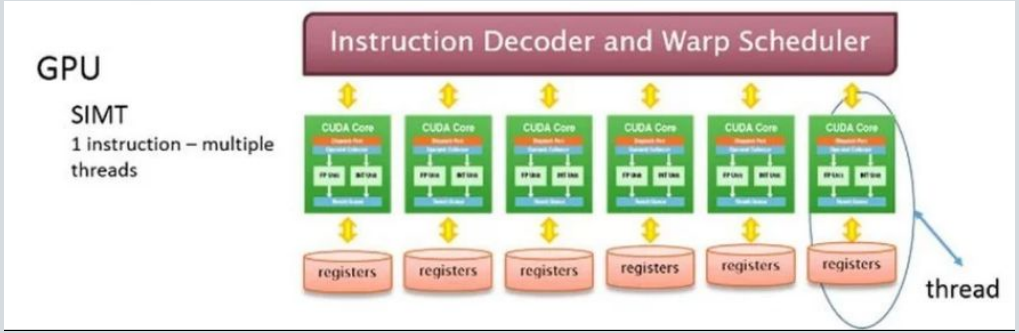


Paradigma de CUDA (II)

- Un programa que se ejecuta en GPU se conoce como **kernel**, y este se ejecuta con los mismos datos de entrada en múltiples hilos.

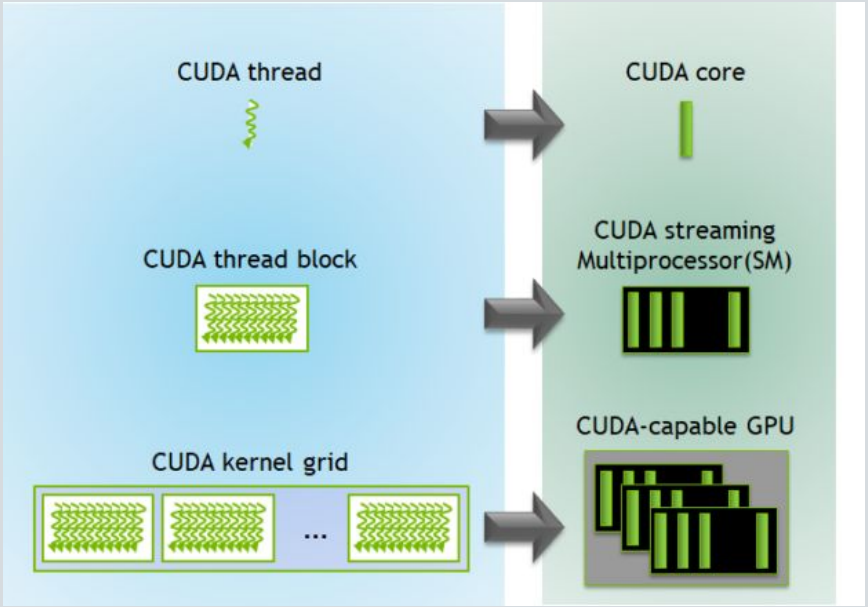
- Se hace una división del trabajo entre los hilos, ya que cada uno actúa sobre un subconjunto distinto de los datos de entrada.

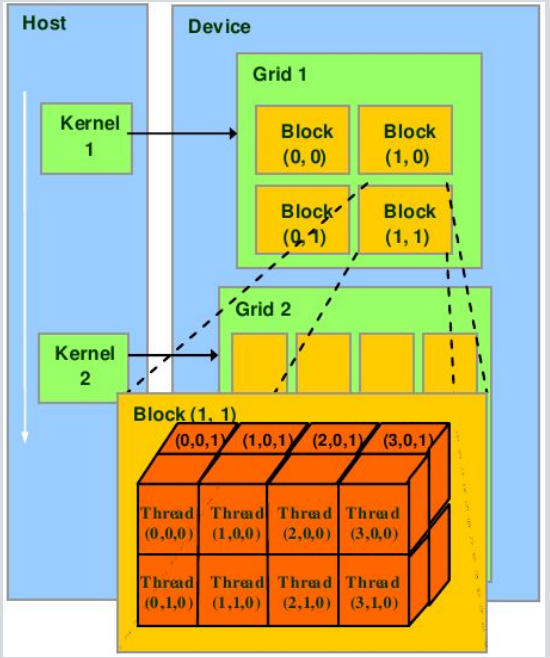




Paradigma de CUDA (III)

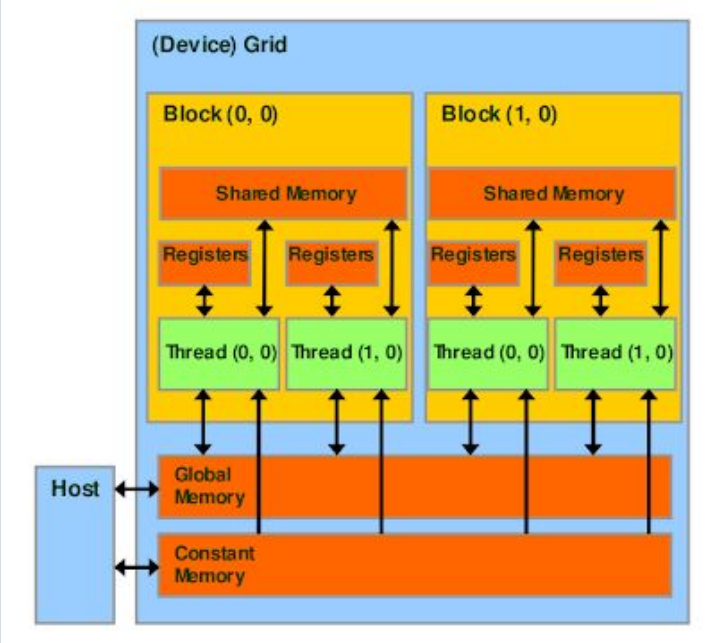
- Los hilos que ejecutan el mismo kernel se agrupan en bloques y retículas. (Transparente con el hardware)





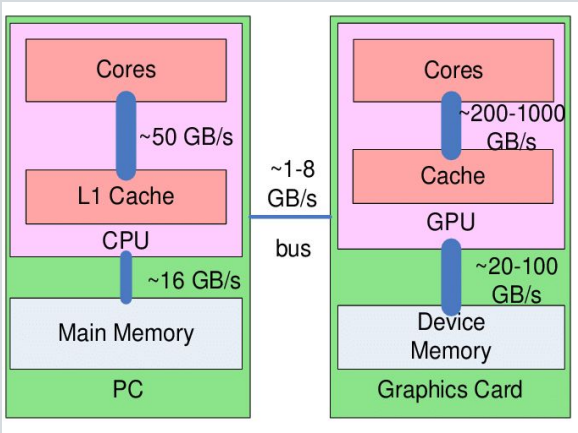
Paradigma de CUDA (IV)

- Los procesadores tienen una memoria propia pequeña, y existe una memoria global muy grande.



Paradigma de CUDA (V)

- La comunicación suele verse de la siguiente manera:
  - La computadora (host) llama un kernel con ciertas entradas.
  - Las entradas se transfieren al GPU.
  - Se realiza el procesamiento, se obtiene una salida en el GPU.
  - La salida se transfiere de regreso al host.



Kernels

¿Cómo se ven? ¿Cómo se programan?

CUDA C

- Extensión para el lenguaje C.
- Funciones para transferir/reservar memoria entre dispositivos.
- Diferenciar las funciones que se ejecutan en CPU y GPU.
- Las funciones en GPU (kernel) modifican una sección en memoria global del GPU, no devuelven valores.
- Facilidades para dividir el trabajo (asignación de índices uni/bi/tridimensionales automáticamente).

Ejemplo - CUDA C

Kernel - El programa que se ejecuta en el GPU

```
1  #include <iostream>
2  __global__ void add(int a, int b, int *c) {
3   *c = a + b;
4  }
5  int main(void) {
6   int c;
7   int *dev_c;
8   cudaMalloc((void**)&dev_c, sizeof(int));
9   add<<<1,1>>>(3, 4, dev_c );
10  cudaMemcpy(&c, dev_c, sizeof(int),
        cudaMemcpyDeviceToHost );
11  printf("3 + 4 = %d\n", c);
12  cudaFree(dev_c);
13  return 0;
14 }
```

Reservar memoria en el GPU

Ejecutar el kernel

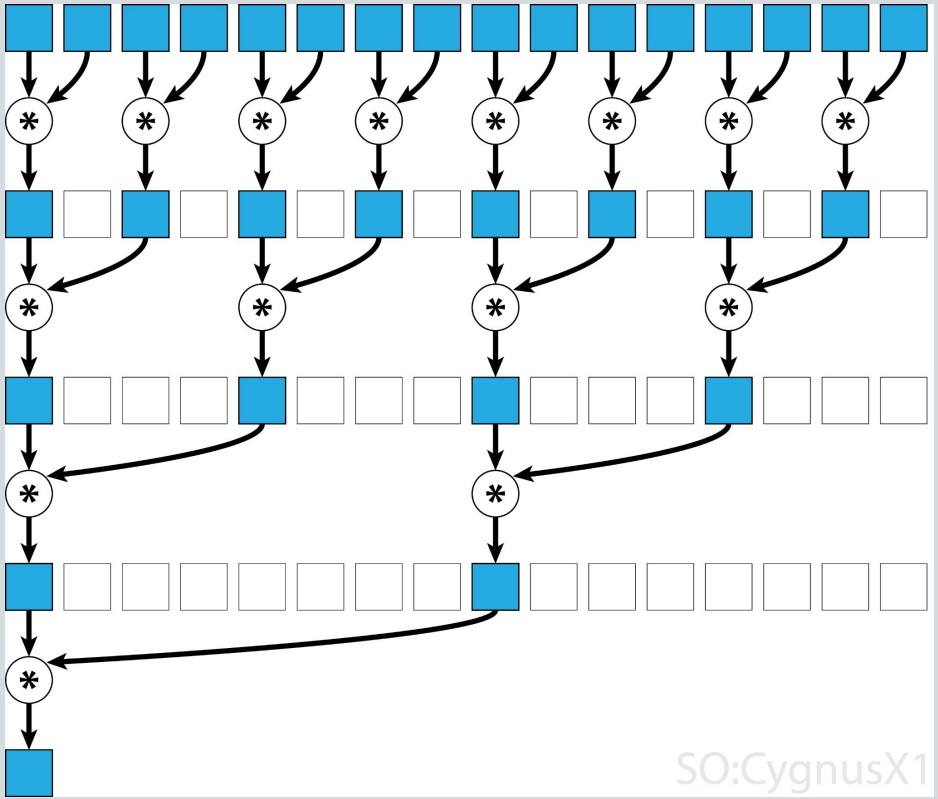
Transferir resultado a CPU

Liberar memoria del GPU

Ejemplo 2 - CUDA C

```
__global__ void vector_add(float *out, float *a,
    float *b, int n) {
    int indice = threadIdx.x;//Indice del thread.
    int paso = blockDim.x;//El numero de threads por
     bloque
    for(int i = indice; i < n; i+=paso){
        out[i] = a[i] + b[i];
    }
}
```

Patrones de diseño más complejos



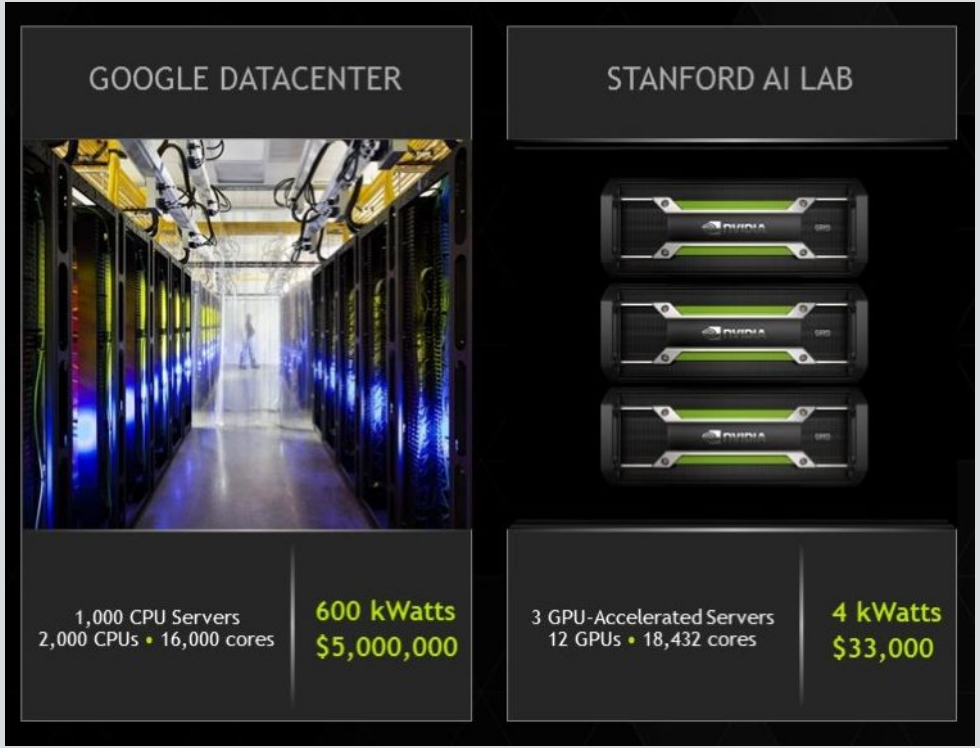
Más allá de CUDA C

Su desarrollo y alternativas

Bibliotecas para cómputo científico

- cuBLAS (Algebra lineal)
- cuFFT (Transformadas rápidas de Fourier)
- cuRAND (Generación de números aleatorios)
- cuSOLVER (Resolver sistemas de ecuaciones)
- cuDNN (Primitivas para redes neuronales profundas)

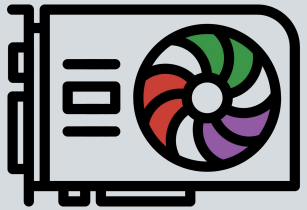
Desempeño y costo



(2015)

- Mundo real: Inputs de 100GB
- Hadoop/Spark: Diseñados para procesamiento en petabytes.
- Severo desperdicio de recursos con un cluster.

Integración



CuPy