

OpenCL



Pensare in parallelo: come programmare una GPU con OpenCL

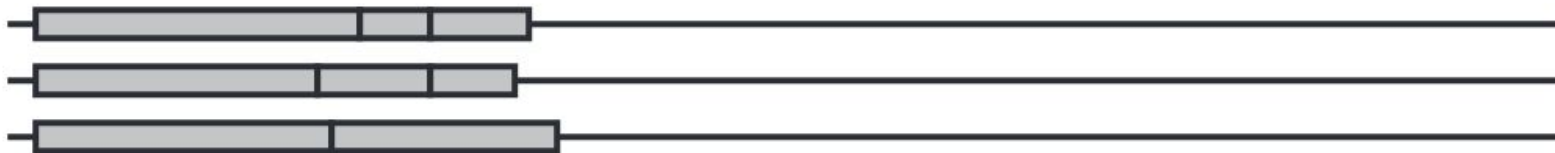
Emanuele Petriglia
Associazione Culturale PCOfficina
30 Settembre 2025



Contenuti distribuiti con licenza Creative Common: Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale



Concurrent, non-parallel execution

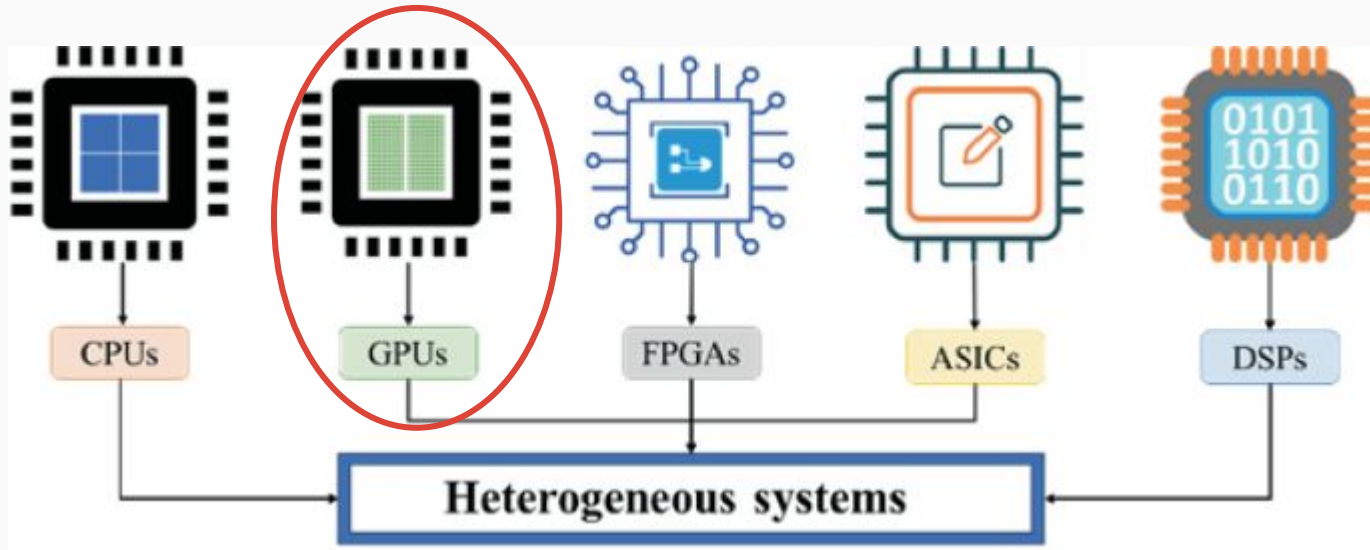


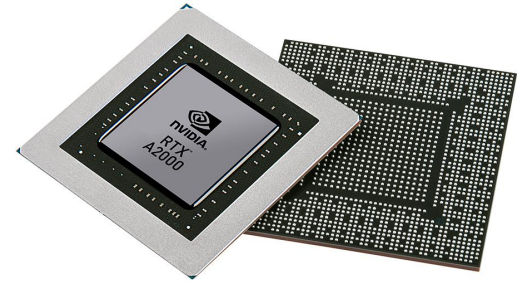
Concurrent, parallel execution

Cos'è OpenCL?



- Framework **eterogeneo** per scrivere ed eseguire programmi di **calcolo parallelo**
- Standard aperto
- Portatile: CPU, GPU, FPGA...
- OpenCL → API per il calcolo
⚠ OpenGL → API grafica







Intel Xeon Silver 4114 (x2) (2017)

- 10 core / 20 thread (x2)
- 384 GB DDR4
- TDP 85 W (x2)
- OpenCL 2.0
- Costo (all'uscita): ~700€ (x2)

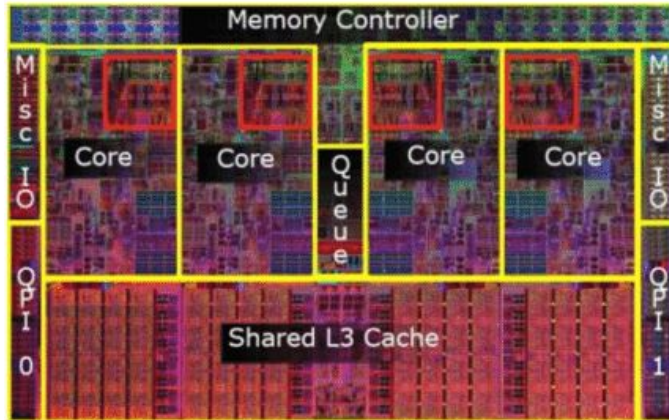
NVIDIA Tesla T4 (2018)

- 2560 CUDA core
- 16 GB GDDR6
- TDP 70W
- OpenCL 3.0
- Nota: no uscita video
- Costo (all'uscita): 2000/2500 € (non pubblico)

Intel Xeon Silver 4114 (x2) (2017) NVIDIA Tesla T4 (2018)

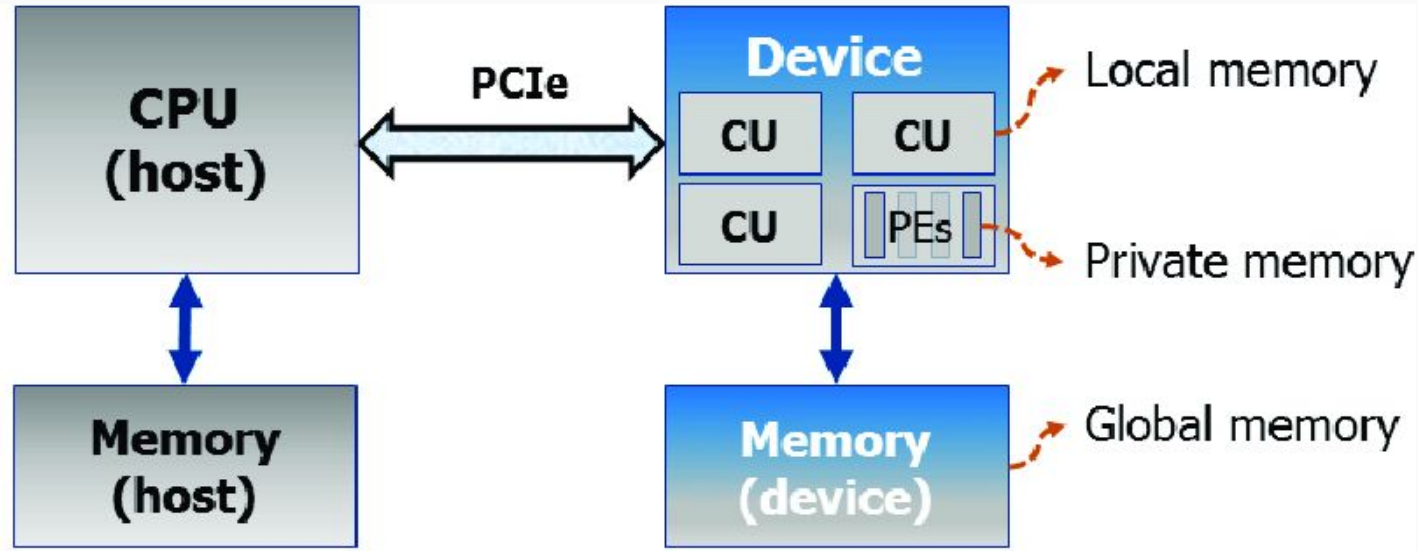
Obiettivo: sfruttare al massimo la GPU con OpenCL

Problema: ripensare algoritmi per sfruttare i migliaia di core



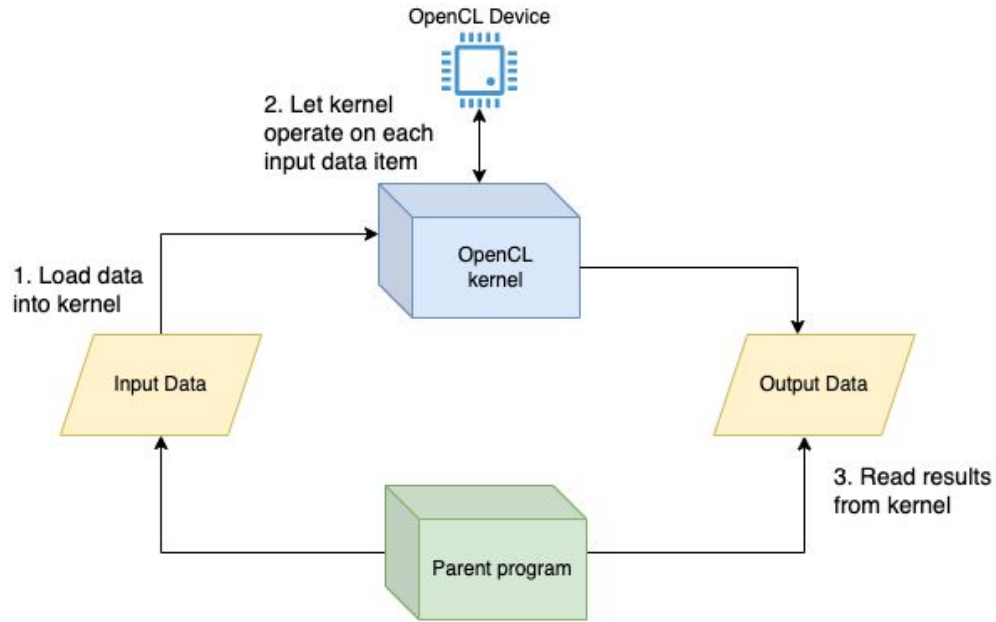
500

- **Host:** CPU e programma che coordina l'esecuzione sul device
- **Device:** l'acceleratore su cui vengono eseguiti i **kernel** OpenCL





- **Kernel: funzione eseguita sul device** da molti work-item in parallelo
- Linguaggio: OpenCL C
 - Compilato dall'host durante l'esecuzione
- Dati di input/output spostati manualmente
- Problema: progettare un kernel non è semplice





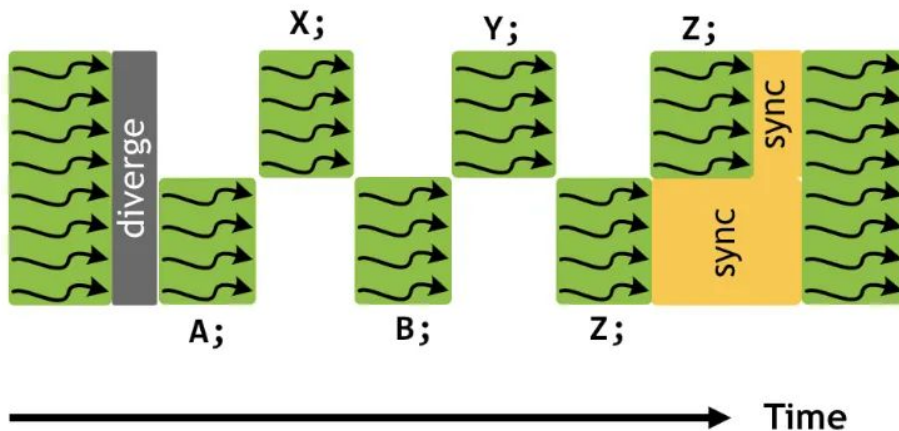
```
void vector_add(int n,  
                float *A,  
                float *B,  
                float *C) {  
    for (int i = 0; i < n; i++) {  
        C[i] = A[i] + B[i];  
    }  
}
```

```
__kernel void vector_add(__global const float *A,  
                        __global const float *B,  
                        __global float *C) {  
    int i = get_global_id(0);  
    C[i] = A[i] + B[i];  
}
```

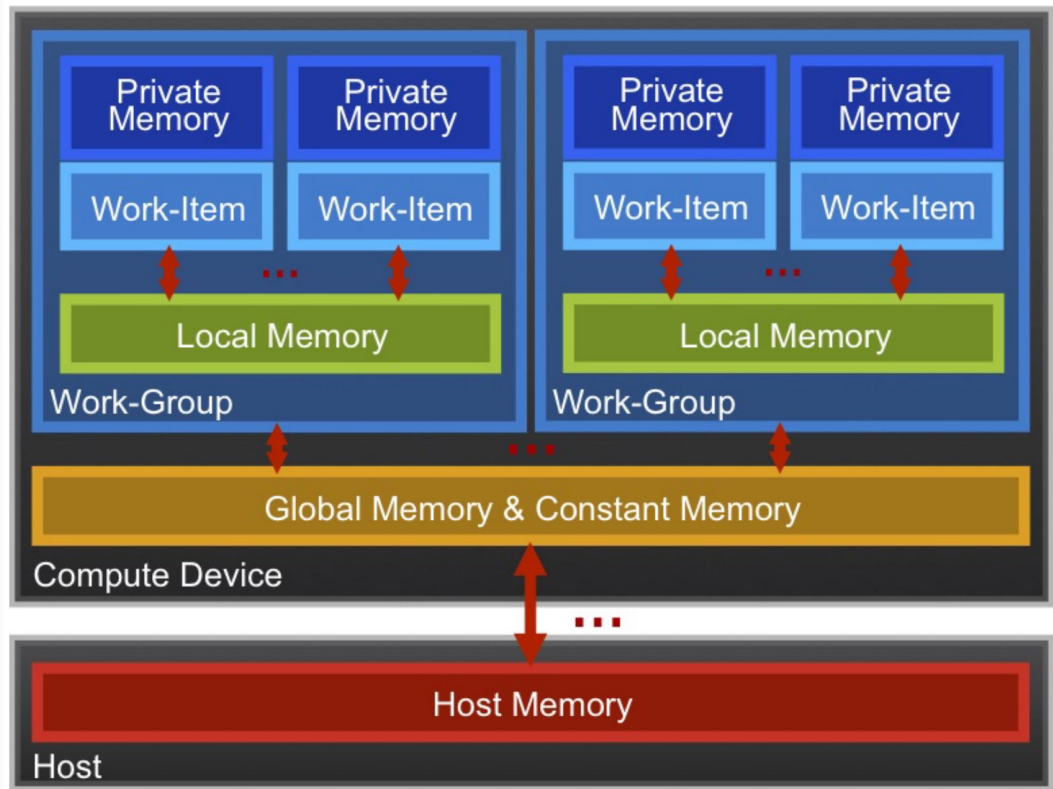


- **Lockstep:** gruppo di work-item che eseguono la stessa istruzione
- Problema: la presenza di branch causano divergenze
 - L'esecuzione delle divergenze è serializzata automaticamente

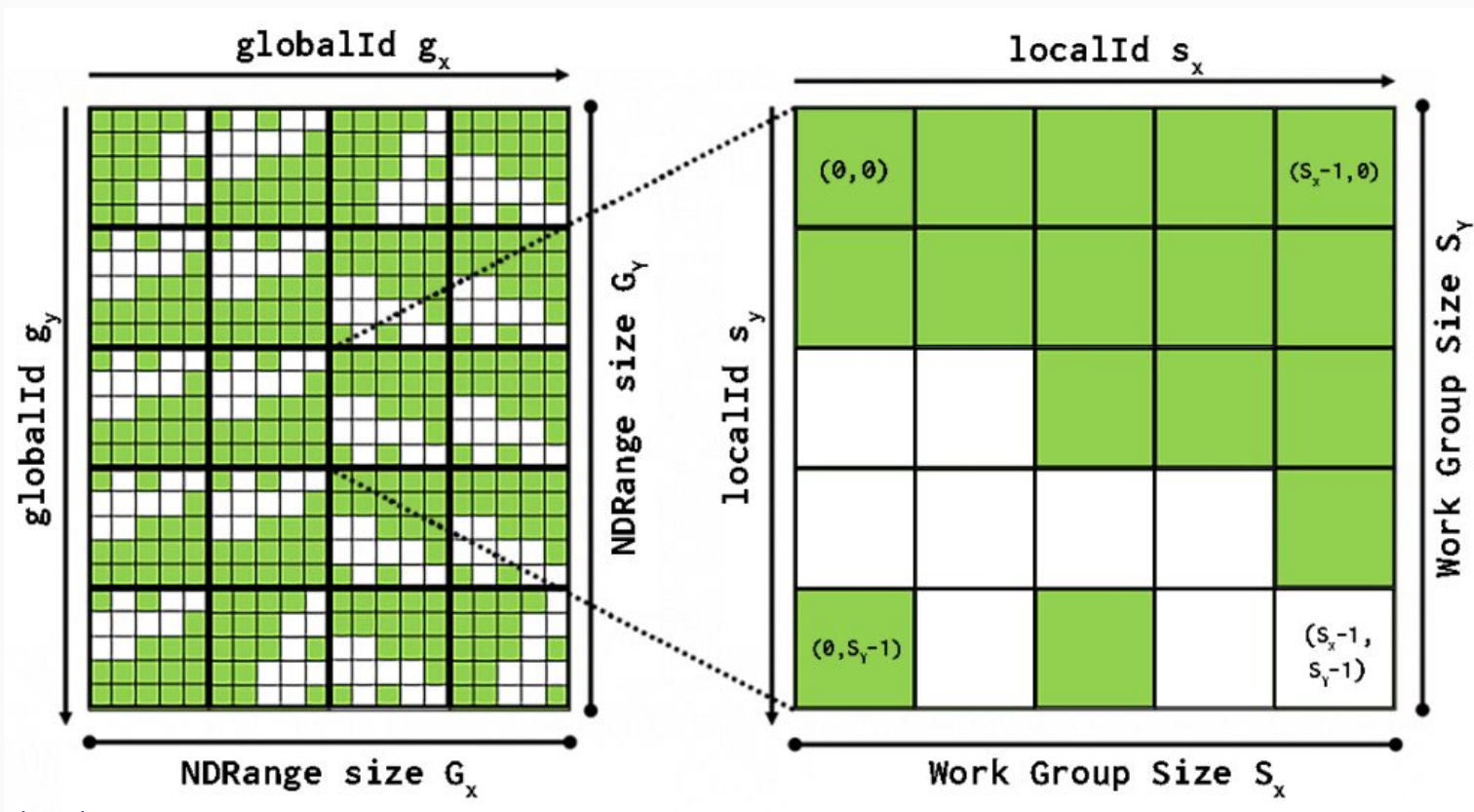
```
if (threadIdx.x < 4) {  
    A;  
    B;  
} else {  
    X;  
    Y;  
}  
Z;  
__syncwarp()
```

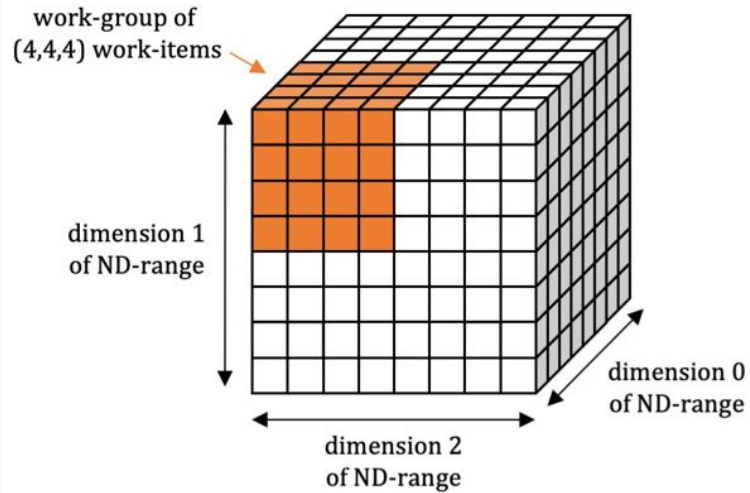


Gerarchia della memoria

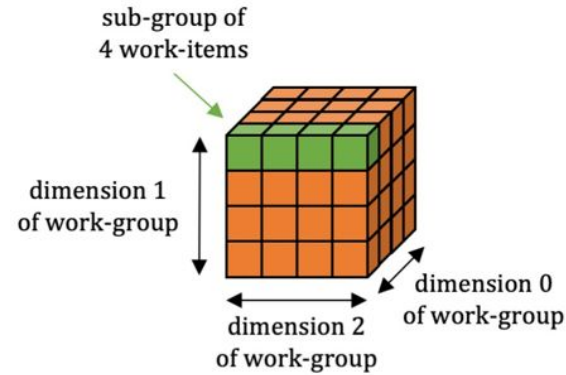


- Dati più vicini al core → accesso più veloce
- La gestione della memoria è esplicita!

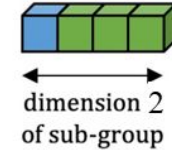




ND-Range



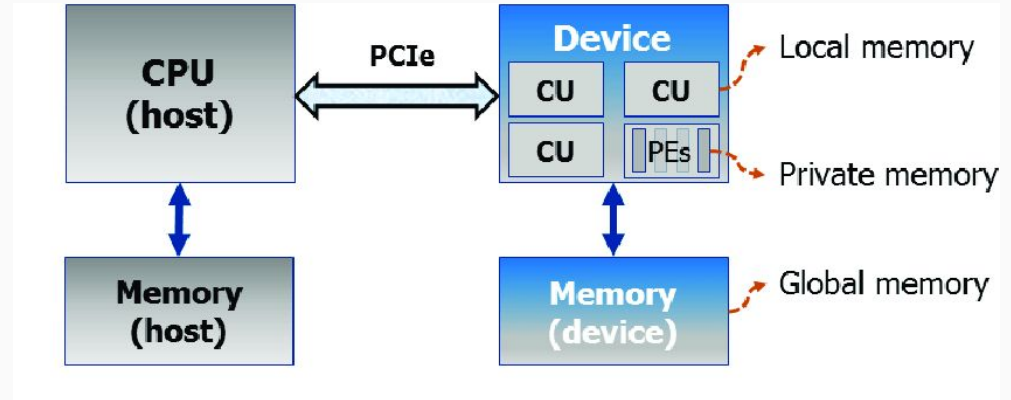
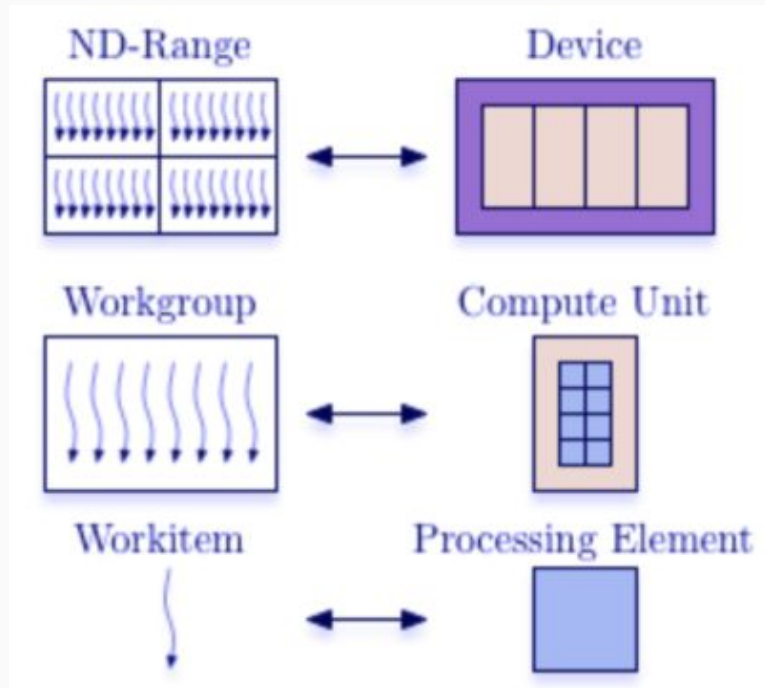
Work-group



Sub-group



Work-item



Spazio di esecuzione dei kernel e device

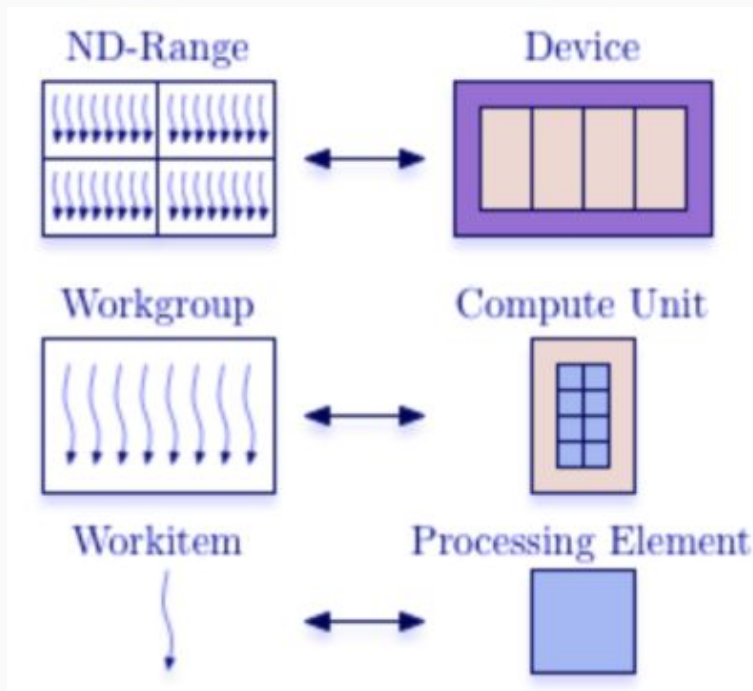
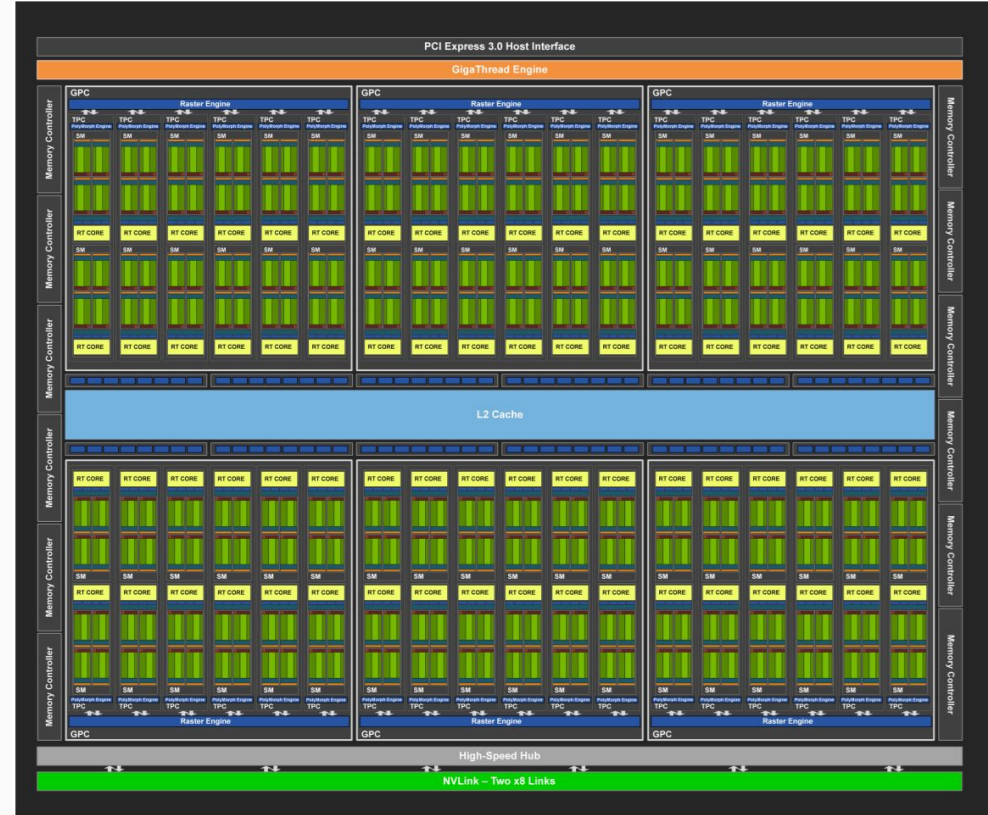
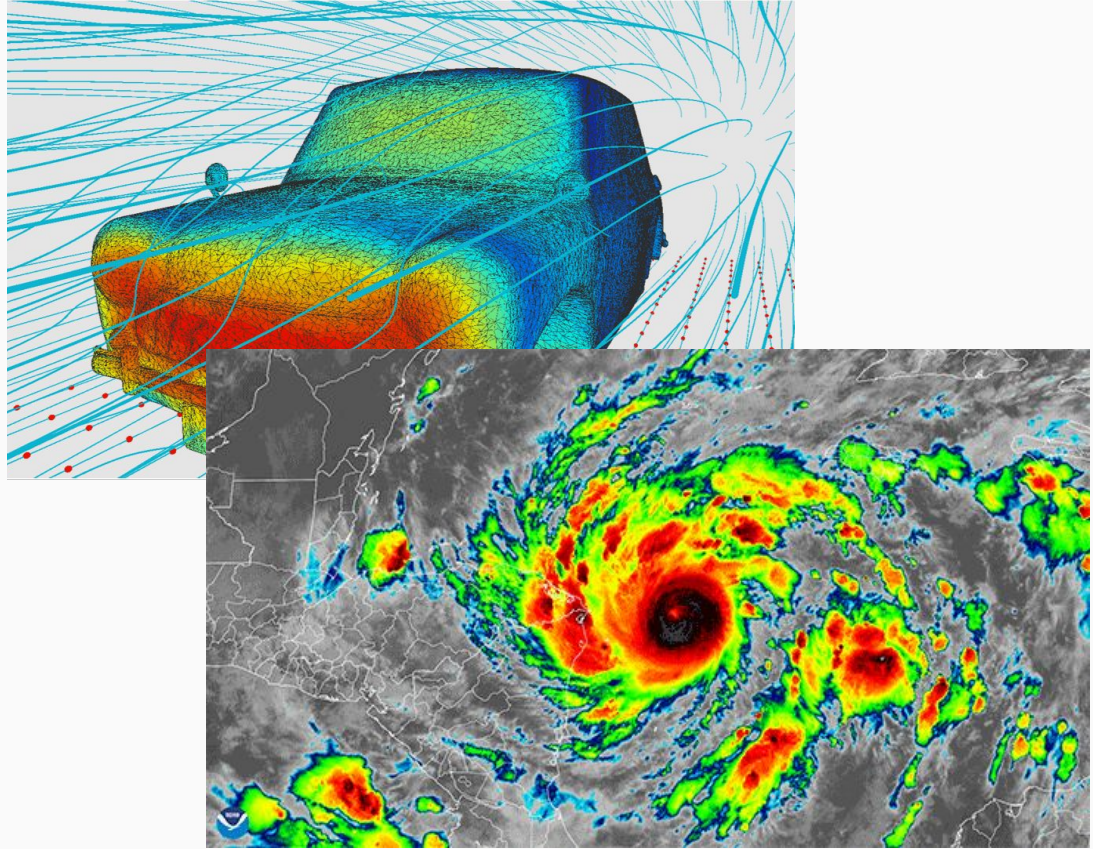


Immagine: thebeardsage.com, nvidia.com



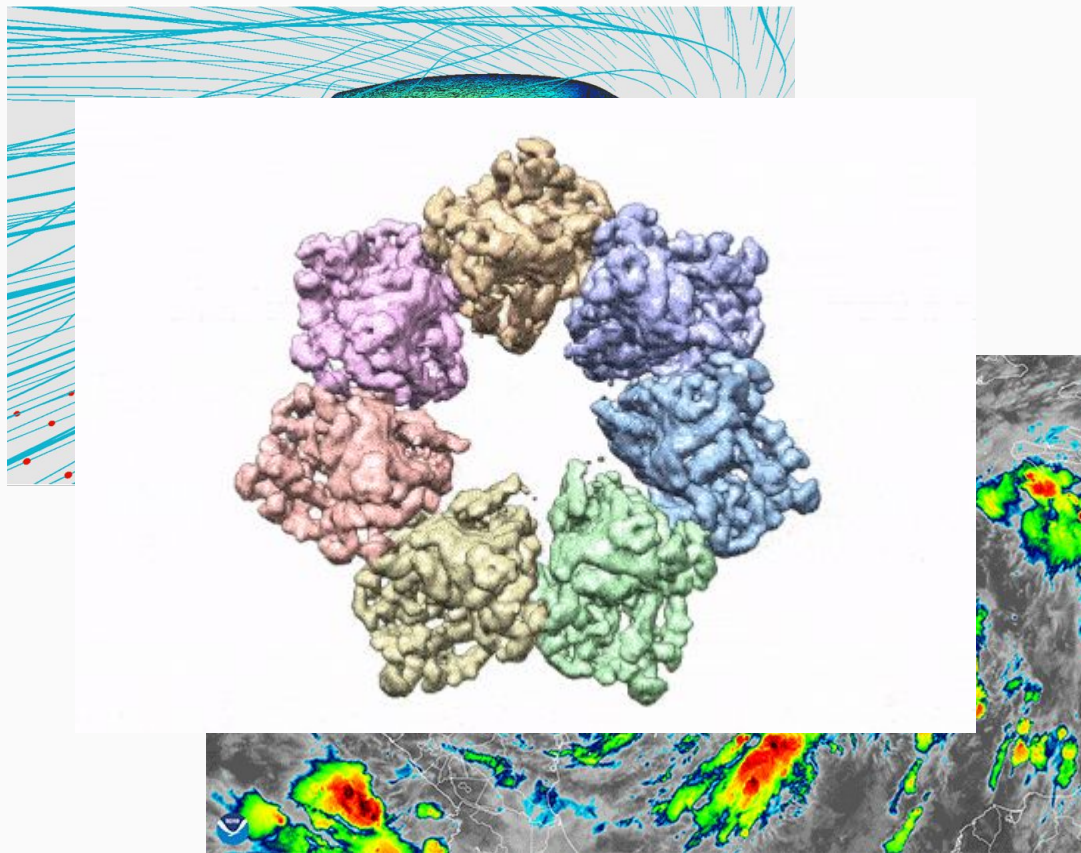
Applicazioni di OpenCL nella realtà

- Simulazioni scientifiche
- Bioinformatica
- Simulazioni industriali
- Grafica e audio
- Machine learning
- Crittografia



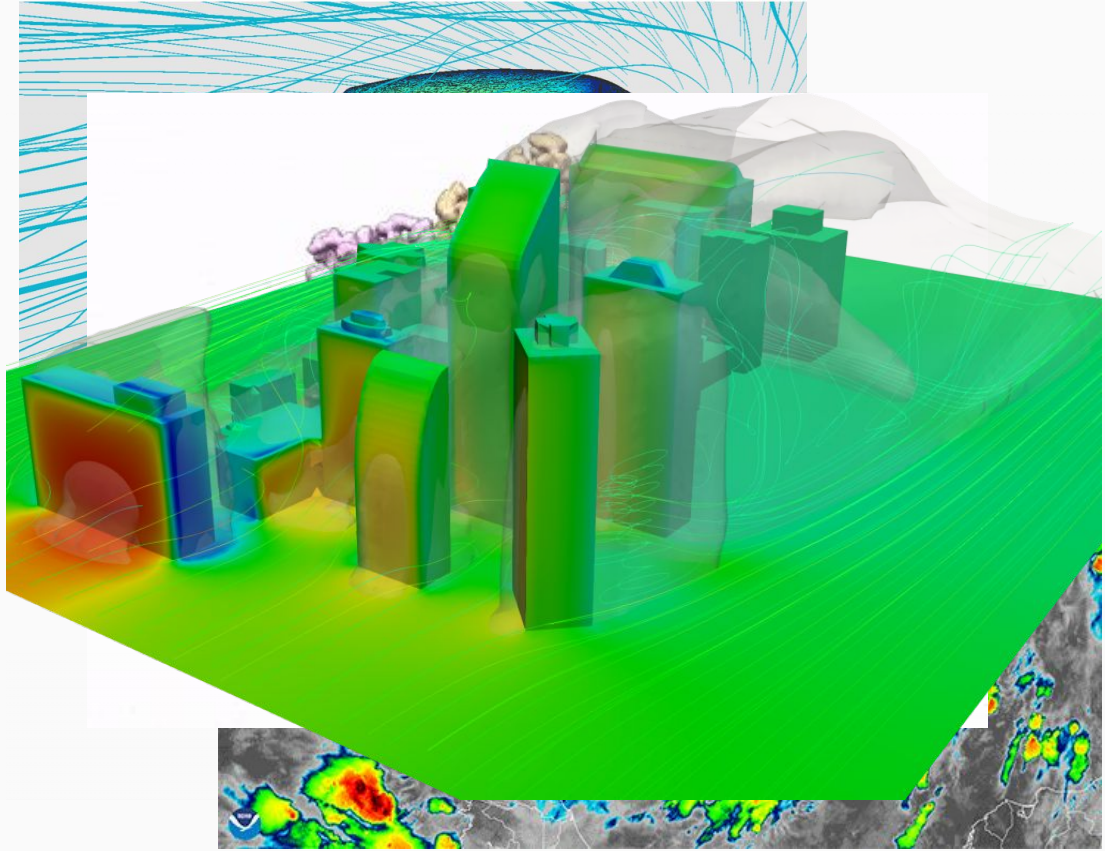
Applicazioni di OpenCL nella realtà

- Simulazioni scientifiche
- Bioinformatica
- Simulazioni industriali
- Grafica e audio
- Machine learning
- Crittografia



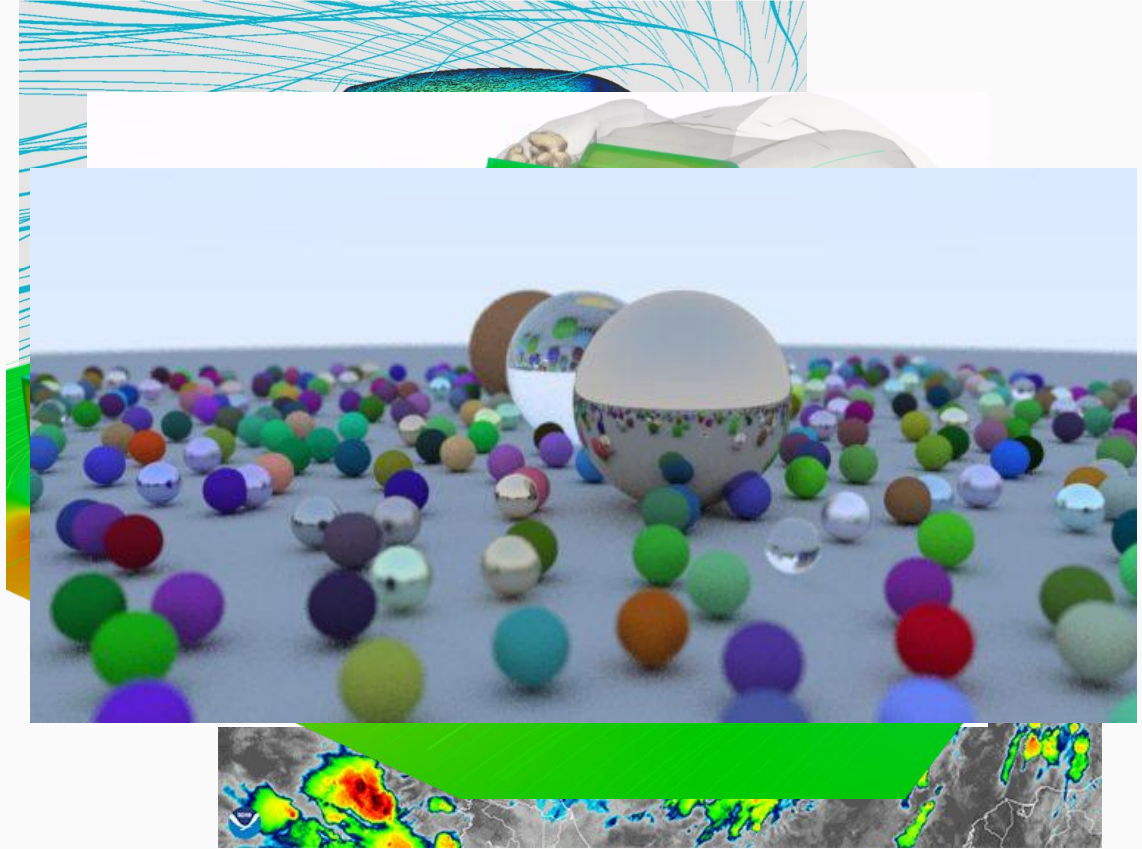
Applicazioni di OpenCL nella realtà

- Simulazioni scientifiche
- Bioinformatica
- Simulazioni industriali
- Grafica e audio
- Machine learning
- Crittografia



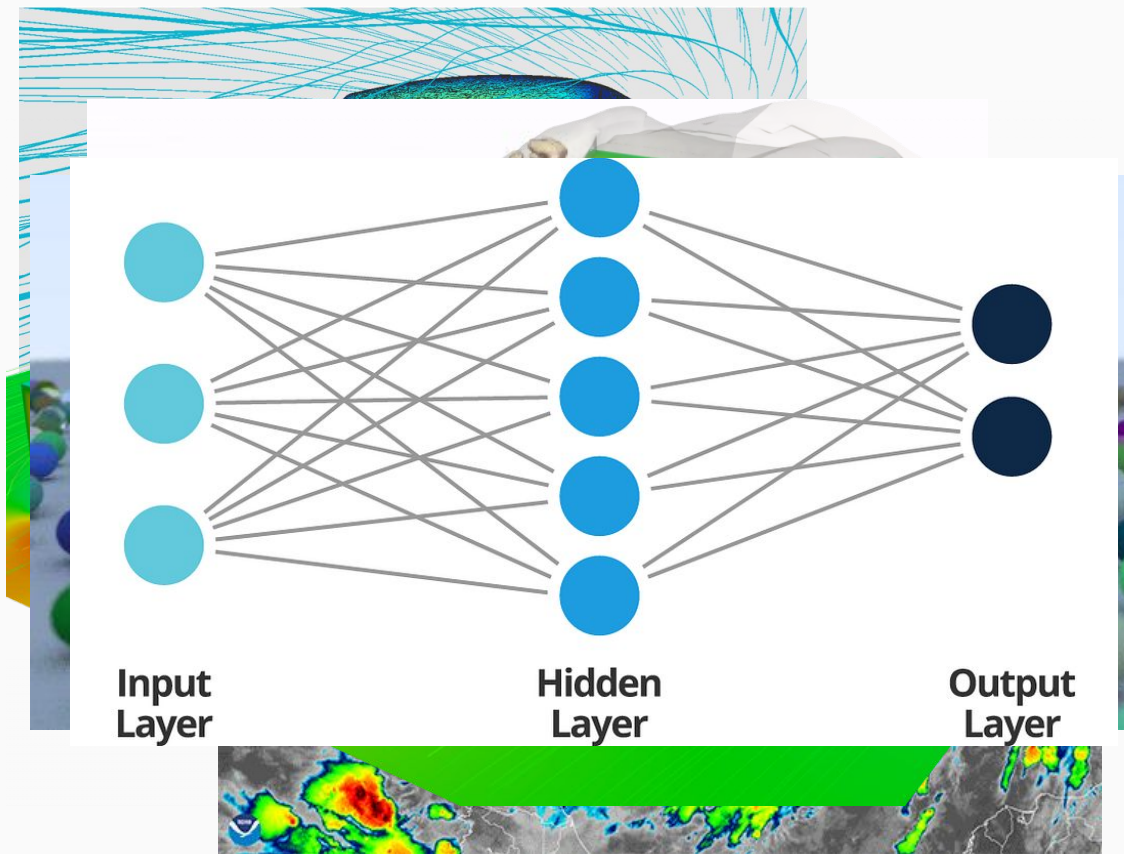
Applicazioni di OpenCL nella realtà

- Simulazioni scientifiche
- Bioinformatica
- Simulazioni industriali
- Grafica e audio
- Machine learning
- Crittografia



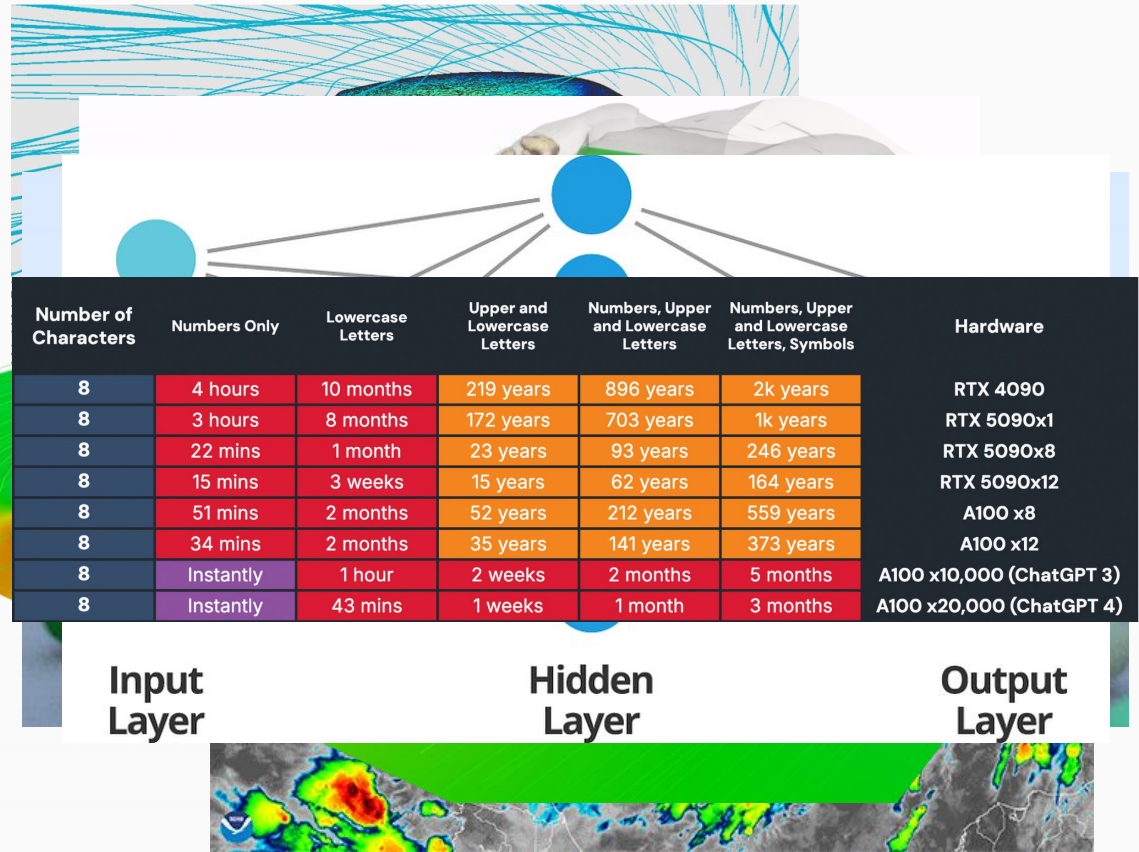
Applicazioni di OpenCL nella realtà

- Simulazioni scientifiche
- Bioinformatica
- Simulazioni industriali
- Grafica e audio
- Machine learning
- Crittografia



Applicazioni di OpenCL nella realtà

- Simulazioni scientifiche
- Bioinformatica
- Simulazioni industriali
- Grafica e audio
- Machine learning
- Crittografia





OpenCL

- Standard aperto
- Multipiattaforma
- Meno performante di CUDA
- Più complesso da usare[†]

CUDA

- Tecnologia proprietaria
- Solo GPU NVIDIA
- Ottimizzato per GPU NVIDIA
- Più semplice da usare

†

SYCL: kernel e codice host nello stesso file C++





- OpenCL: un **framework** aperto per il **calcolo parallelo** su piattaforme **eterogenee**
- **Kernel**: funzione che viene eseguita in parallelo
- Dove partire?
 - PyOpenCL in Python
 - OpenCL API in C/C++ (OpenCL-Wrapper può aiutare)

DOMANDE?

Come sostenere PCOfficina



Se volete entrare a far parte dell'associazione, potete iscrivervi oggi stesso!

Potete sostenererci anche senza associarvi:

- donando materiale informatico in buono stato
- tramite donazione
- facendo conoscere PCOfficina ai vostri conoscenti

GRAZIE!

Sito Web: www.pcofficina.org

E-mail: info@pcofficina.org

Newsletter: Iscrivetevi dal sito

Canale Telegram: @pcofficina

Youtube: @pcofficina

Gruppo Facebook: @pcofficina

X: @pcofficina