Grafikprozessor (GPU)

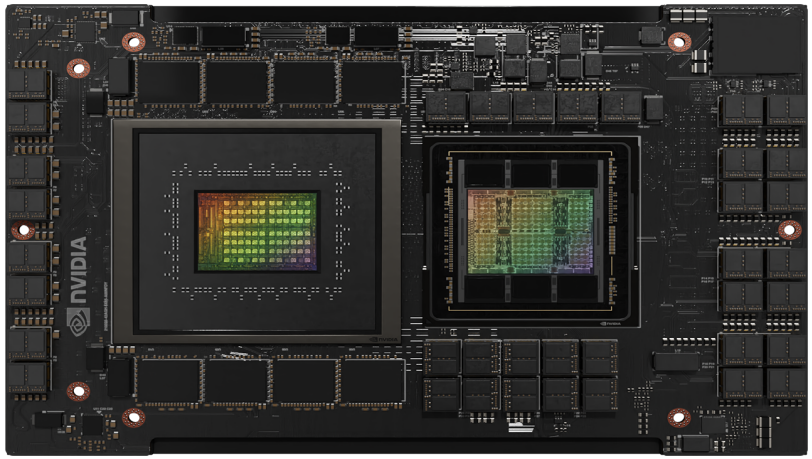
Ein Grafikprozessor (Graphics Processing Unit, GPU) ist eine spezialisierte elektronische Schaltung, die entwickelt wurde, um die Berechnung und Darstellung von Bildern und Grafiken auf einem Computerbildschirm zu beschleunigen. GPUs sind besonders gut geeignet für parallele Verarbeitung großer Datenmengen, was sie ideal für Anwendungen wie Videospiele, 3D-Modellierung und maschinelles Lernen macht.

Leistungsfähigkeit von GPUs

Die Leistungsfähigkeit von GPUs wird oft in Teraflops (TFLOPS) gemessen, was die Anzahl der Billionen Gleitkommaoperationen pro Sekunde angibt, die sie ausführen können. Moderne GPUs können je nach Modell und Anwendungsbereich unterschiedliche Leistungsstufen erreichen. Typische Rechengenauigkeiten sind 32-Bit (FP32) und 16-Bit (FP16) Gleitkommazahlen. Bei 64-Bit (FP64) Gleitkommazahlen ist die Leistung in der Regel deutlich geringer.

GPU-Klasse	Leistung
Einsteiger- / Mittelklasse	2 - 10 TFLOPS
High-End-Gaming	10 - 100 TFLOPS
Workstation- / AI-GPUs	20 - 1000 TFLOPS

Quelle: GPU Monkey Benchmark NVIDIA RTX 6000 ADA FP32
Quelle: NVIDIA H100 Datasheet


NVIDIA

- **Architektur:** Viele einfache Rechenkerne in *Streaming Multiprocessors (SM)*, optimiert für parallele Berechnung.
- **Speicherhierarchie:**
  - *Global Memory (VRAM)*: Großer, langsamer Speicher für die gesamte GPU.
  - *Shared Memory*: Schneller Zwischenspeicher pro SM.
  - *Register*: Klein, extrem schnell, lokal pro Kern.
- **SIMD-Prinzip:** Gleiche Operation auf mehrere Daten gleichzeitig (Single Instruction, Multiple Data).
- **Thread-Modell:** Threads in *Thread-Blocks* organisiert, diese in einem *Grid*.
- **Spezialisierte Hardware:** Rasterisierung, Texturierung und Shading für Grafikoperationen.

Shader Unit

- Eine **Shader Unit** ist ein **ALU-Rechenwerk** (z. B. FP32/INT) **innerhalb eines Streaming Multiprocessors (SM)**.
- Ein **SM** enthält **viele Shader Units** plus **Warp/Thread-Scheduler**, **Register-Datei** und **Shared Memory**.
- Viele SMs zusammen bilden den **GPU-Rechenteil** (neben festen Grafik-Blöcken wie Rasterizer/Textur-Einheiten).

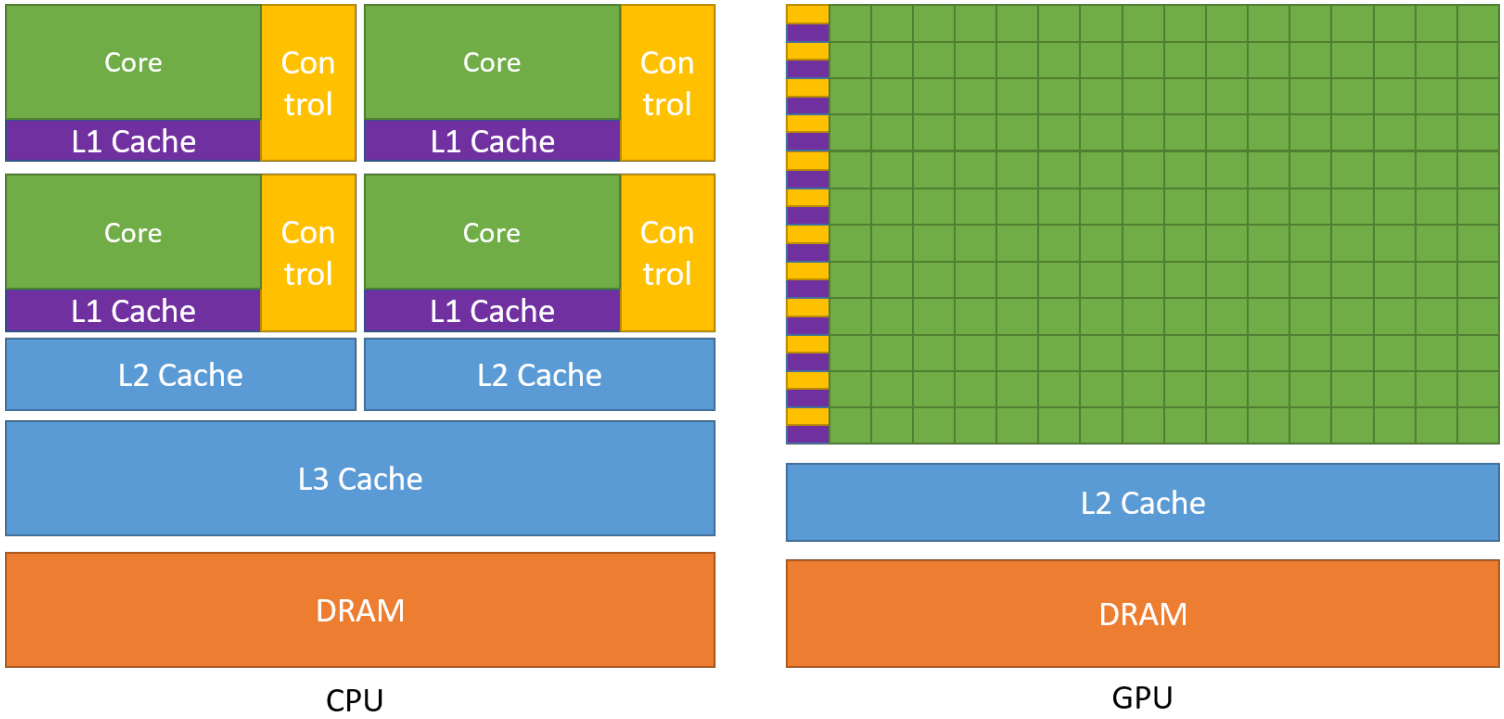


Abbildung: Quelle:
https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

Was macht die GPU schnell?

Die GPU ist schnell, wenn sie **sehr viele ähnliche Rechnungen gleichzeitig** machen kann.

- Bild: viele Pixel werden parallel berechnet
- Geometrie: viele Vertices werden parallel transformiert
- KI/Mathe: viele Zahlen in großen Matrizen werden parallel verarbeitet

Was ist wichtig beim Programmieren?

- **Große Aufgaben auf einmal:** lieber 1 große Rechnung als 1000 kleine
- **Daten nicht ständig hin- und herschieben:** CPU $\leftrightarrow$ GPU kostet Zeit
- **Wenig Verzweigungen:** wenn viele Threads unterschiedliche Wege gehen, wird es langsamer

Zwei typische Wege

- **Grafik (Shader):** Programme für Vertex/Pixel in OpenGL/Vulkan/Direct3D/Metal/WebGPU
- **Rechnen (Compute):** GPU als Parallelrechner (Compute Shader oder CUDA/HIP/OpenCL)

PyTorch als "GPU-Programmierung ohne viel API-Aufwand"

In PyTorch sind viele Tensor-Operationen intern GPU-Kernels. Man schreibt Code in Python, aber die GPU rechnet die großen Operationen.

Grobe Größenordnung

Die Kopie läuft typischerweise über **PCI Express (PCIe)**:

- **Große, zusammenhängende Datenblöcke:** typischerweise **Millisekunden pro 100 MB**
- **Sehr große Transfers (GB-Bereich):** typischerweise **Zehner-Millisekunden pro GB**
- **Viele kleine Transfers:** oft **deutlich langsamer**, weil pro Transfer ein fester Overhead anfällt

Warum kleine Kopien besonders schlecht sind

PCIe ist paketbasiert und jede Kopie hat Overhead. Deshalb: **lieber wenige große Kopien statt vieler kleiner.**

Quelle: NVIDIA Blog (Transfers bündeln, pinned memory)
Quelle: NVIDIA Forum (realistische PCIe4 x16 Durchsatzwerte, Overhead bei kleinen Transfers)

Das Nadelöhr: PCIe vs. GPU-internes Speichertempo

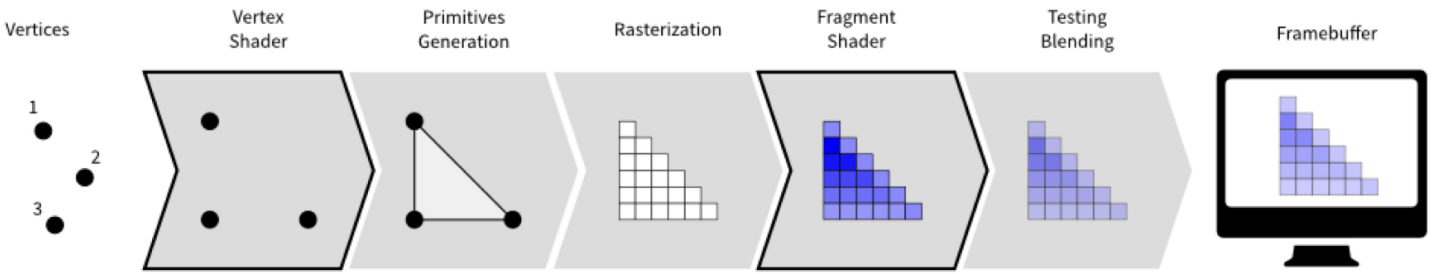
- PCIe 4.0 x16 liegt grob bei **einigen Dutzend GB/s**, PCIe 5.0 x16 bei **bis zu 64 GB/s** (theoretisch).
- Der Speicher **auf der GPU** (VRAM/HBM) ist dagegen in einer ganz anderen Liga: Datacenter-GPUs erreichen **bis über 2 TB/s** Speicherbandbreite.

Praktische Konsequenz (Merksatz)

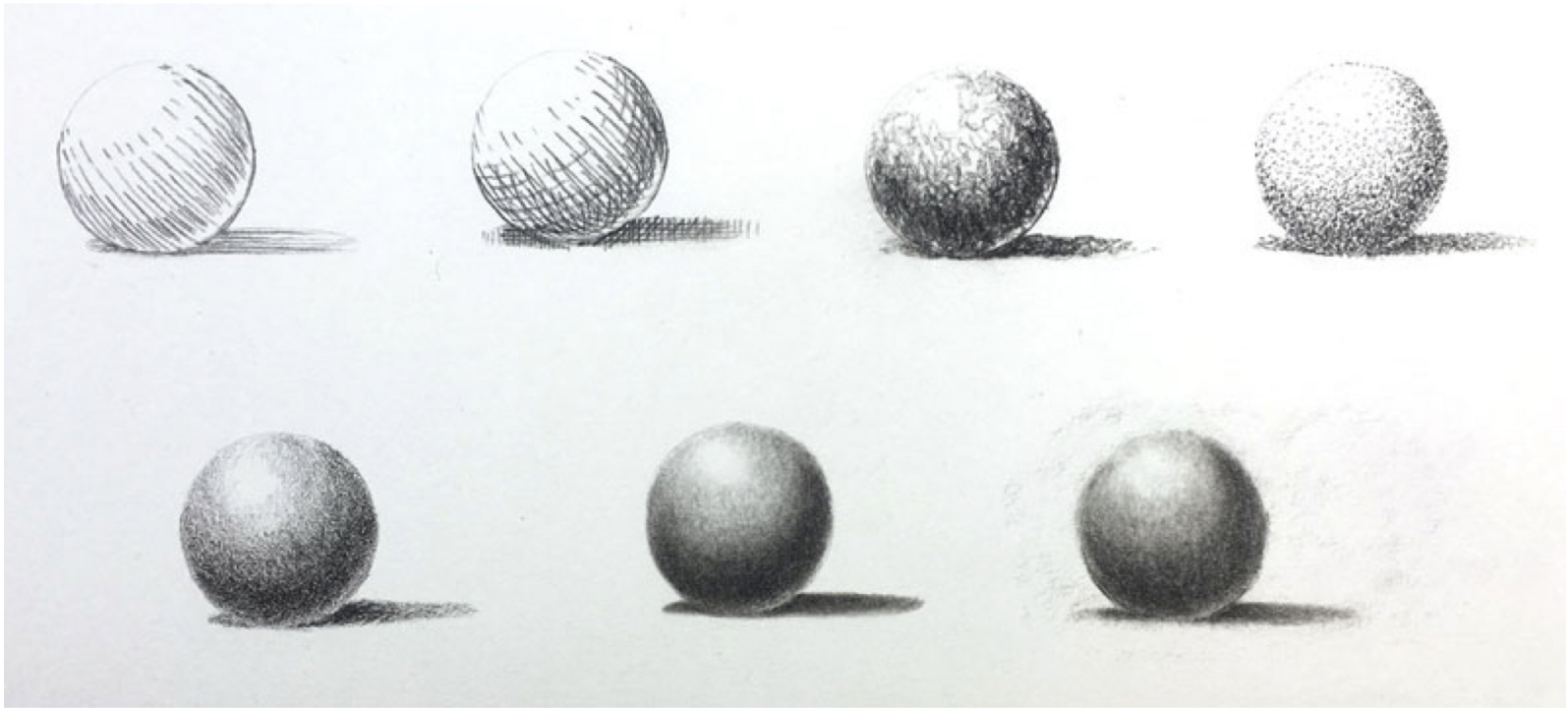
Wenn Daten jeden Frame / jeden Iterationsschritt über PCIe nachgeladen werden müssen, dann bestimmt oft **der Transfer** die Gesamtgeschwindigkeit. Deshalb: **Daten einmal auf die GPU laden und dort möglichst viel damit machen.**

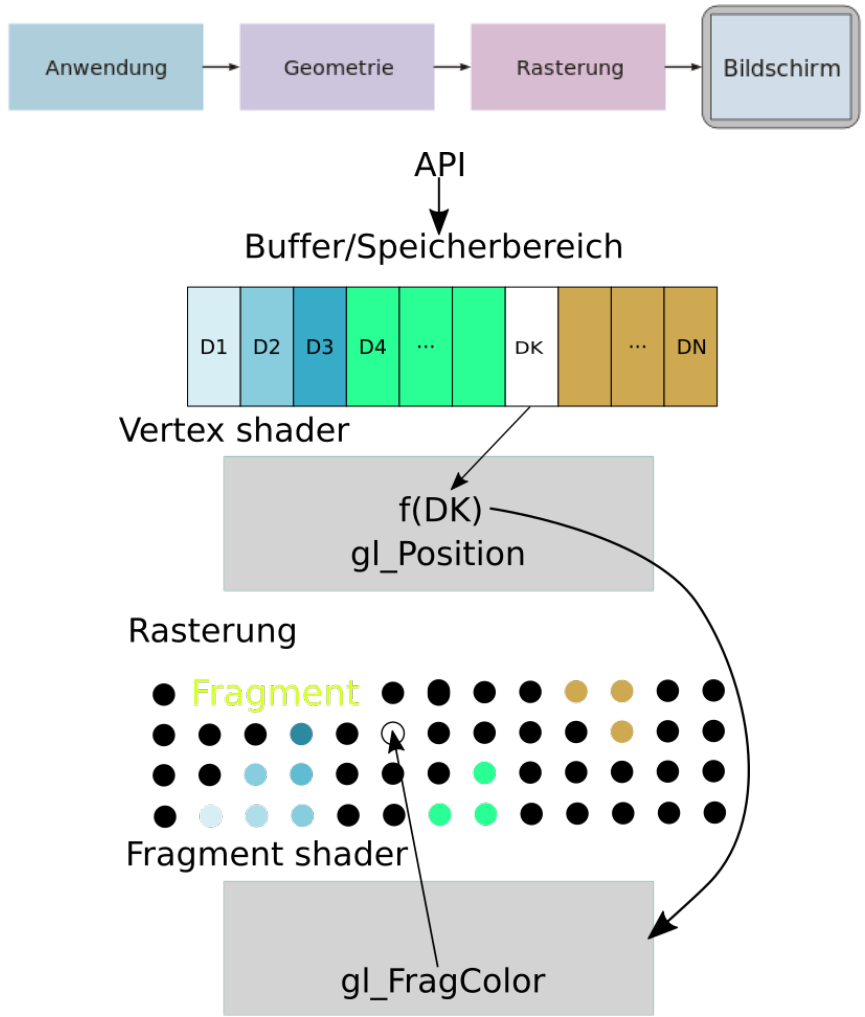
Quelle: NVIDIA A100 Datasheet (GPU Memory Bandwidth > 2 TB/s)
Quelle: ComputerBase (PCIe 5.0 x16 $\sim$ 64 GB/s, dual-simplex erklärt)

Vertex und Fragmentshader



Shader=Schattierer




Anwendung
Geometrie
Rasterung
Bildschirm
API
Buffer/Speicherbereich
D1
D2
D3
D4
...
DK
...
DN
Vertex shader
f(DK)
gl_Position
Rasterung
Fragment
Fragment shader
gl_FragColor

https://www.shadertoy.com/