



**Hochschule für Technik
und Wirtschaft Berlin**

University of Applied Sciences

**Innovativ und vielfältig: die Hochschule
für Technik und Wirtschaft Berlin**

Verwendung der Grafikkarte für allgemeine Berechnungen

Nvidia CUDA

Gliederung

- Geschichte der Grafikkarten
- Leistung: CPU vs. GPU
- GPGPU
- CUDA
- Anwendung

EINLEITUNG

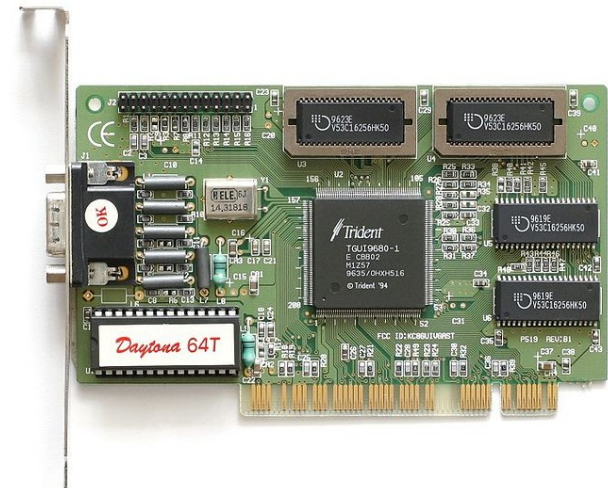
Geschichte der Grafikkarten (1)

- 1977: Erste Grafikkarte im Apple II



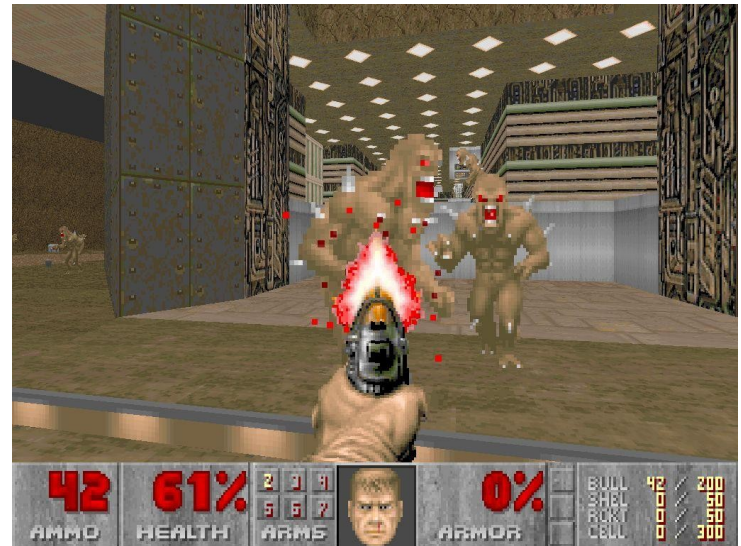
Geschichte der Grafikkarten (2)

- 1981: IBM-PC mit Grafikkarte
- 1982: Hercules Grafikkarten
- 1989: Farbgrafikkarten
- 1991: Einführung der Grafikprozessoren



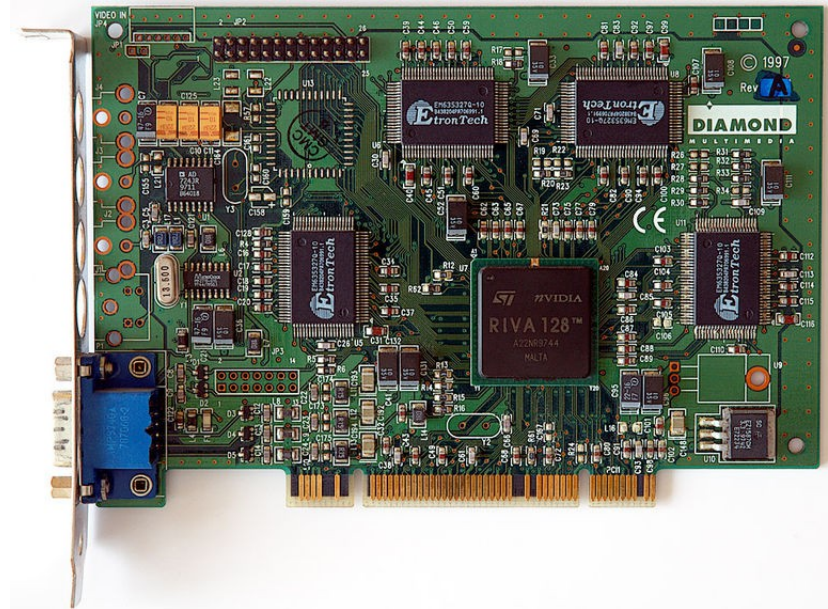
Geschichte der Grafikkarten (3)

- 1993: „Doom“ -> 3D-Beschleuniger-Karten
- 1995: Beschleunigung von Videowiedergabe und Dekodierung (z.B. MPEG)
- 1998: Multi-GPUs (SLI)



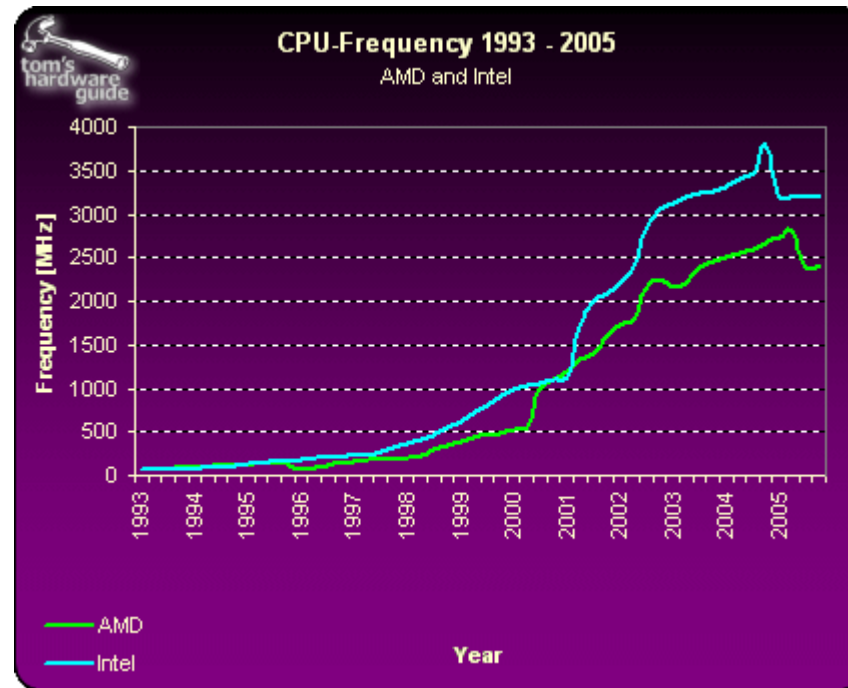
Geschichte der Grafikkarten (4)

- 1999: Integration des 3D-Beschleunigers und Grafikchip



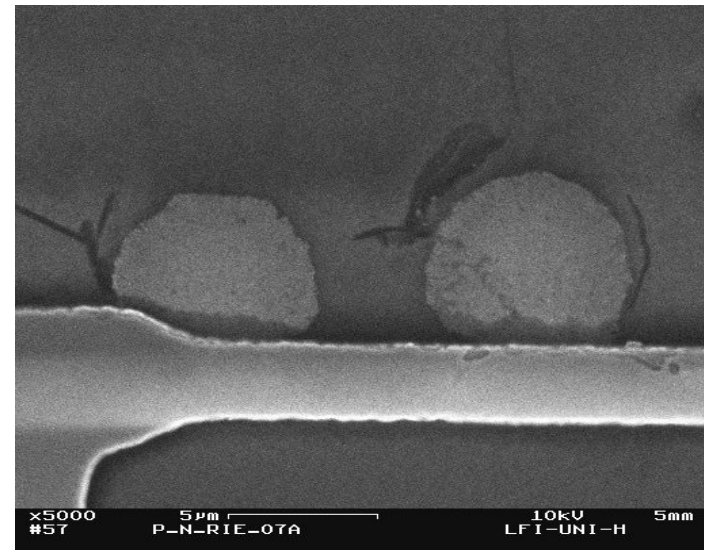
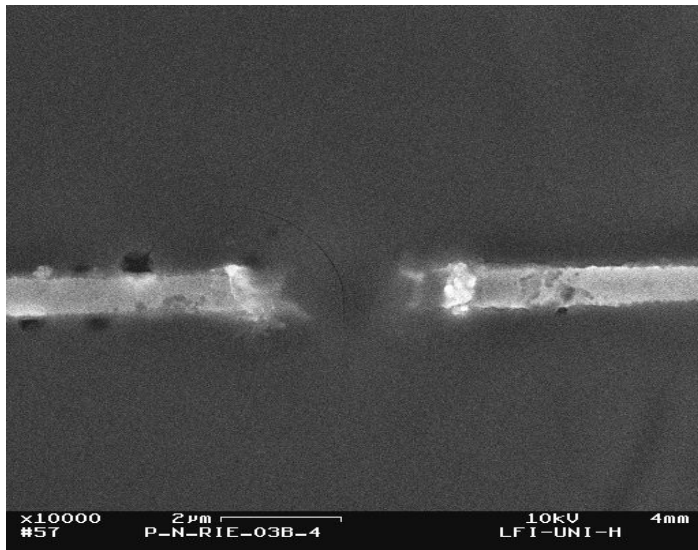
LEISTUNG DER GPU_s

Entwicklung der CPU

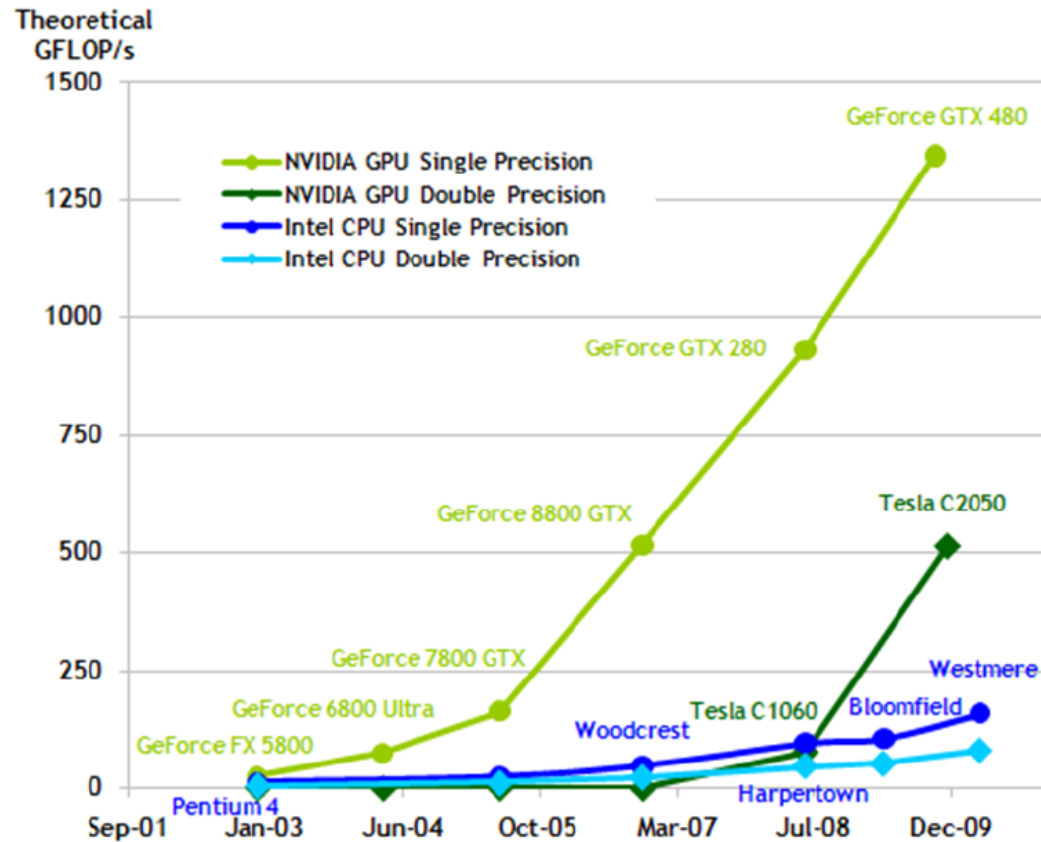


Gründe für den Einbruch bei der CPU-Entwicklung

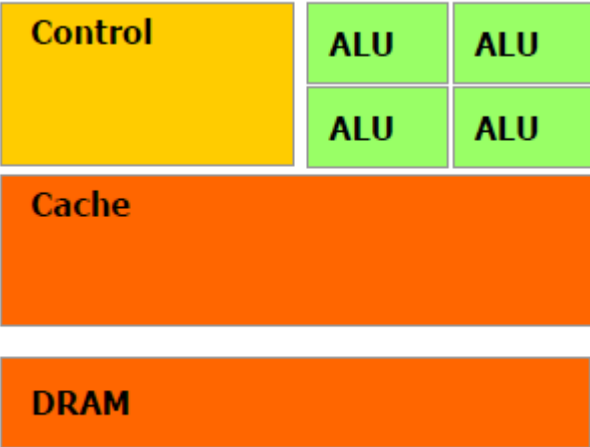
- Abwärme
- Elektromigration



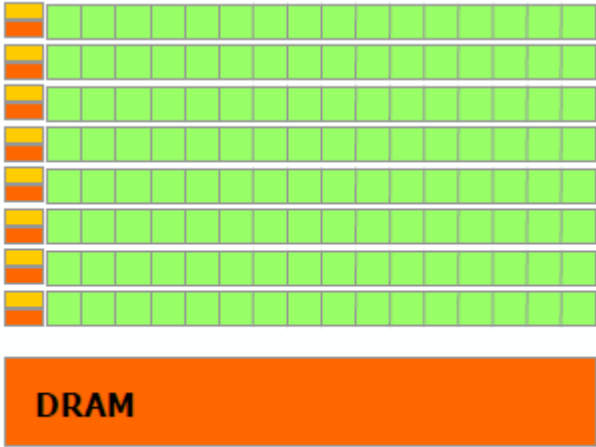
Potential der GPU



Architektur Vergleich



CPU



GPU

GPGPU

Was ist GPGPU?

- General Purpose Computation on Graphics Processing Unit
- Nutzung der GPU für allgemeine rechenintensive Aufgaben
- Viele parallel ausführbare Berechnungen notwendig (SIMD), sonst keine Vorteile gegenüber CPU

Nachteile durch GPGPU

- Unterschiede zwischen GPUs verschiedener Hersteller
- Spezielle Konzepte für parallele Programmierung
- Meist wenig Speicher auf Grafikkarten
- Speicher ist von minderer Qualität -> Fehleranfälligkeit
- Auf Unterstützung von CPU angewiesen

2000: Die Anfänge

- erste Grafikkarten mit programmierbarer GPU (NV20 und R200)
- Shader-Programmierung
- Nutzung: Erzeugung neuartiger Grafikeffekte
- Jedoch Einschränkungen:
 - Max 10 Instruktionen pro Programm
 - Exotische Datentypen (9 oder 12 Bit Festkomma)

2003: Die große Erweiterung

- Nvidia GeForce FX (NV30)
- Unterstützung von 32 Bit Gleitkommaberechnungen auf GPU
- Über 1000 Befehle
- Einsatz: mathematische Berechnungen (Matrizen)
- Zugriff auf GPU nur über DirectX oder OpenGL möglich

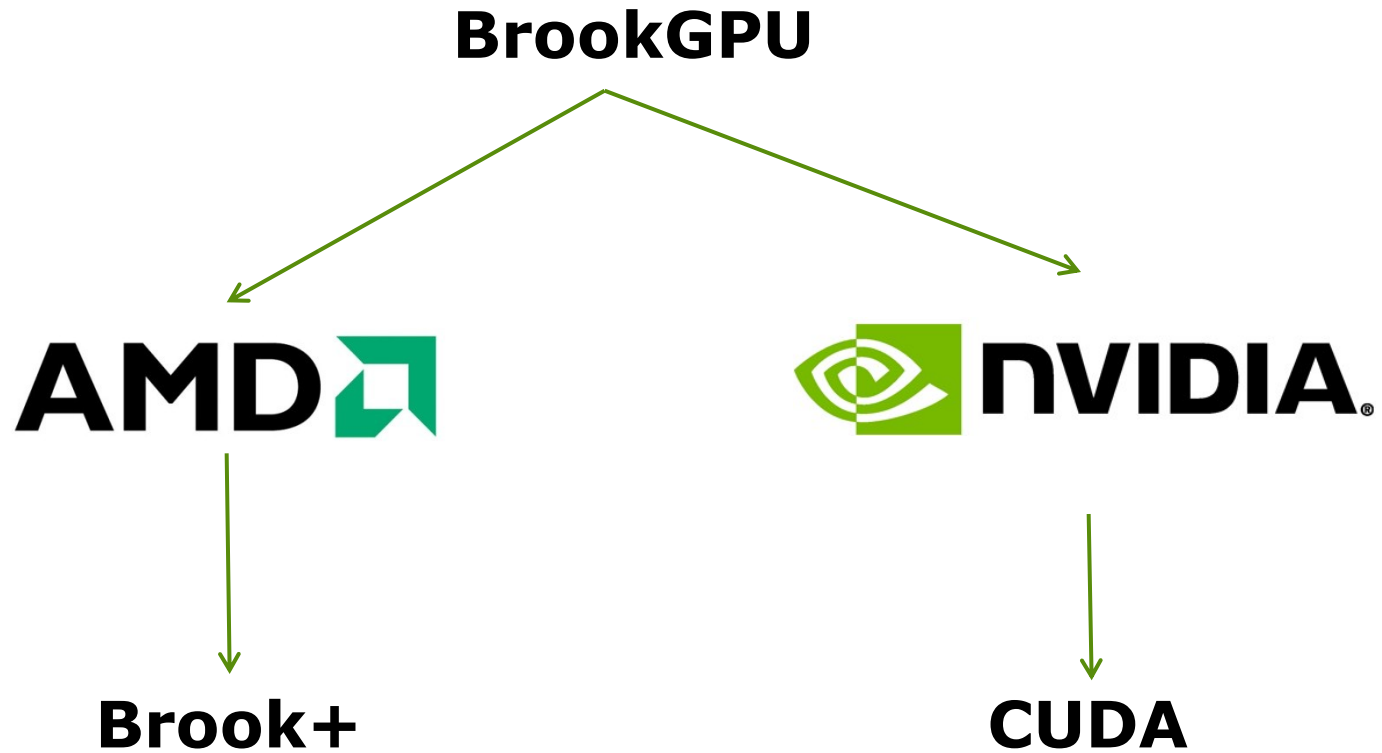
2004: „BrookGPU“ (1)

- Studienprojekt „BrookGPU“
- GPGPU-API
- Macht GPU zum mathematischen Co-Prozessor
- Programmiersprache: C
- Verteilt Algorithmus auf hunderte Shader-Prozessoren
- Schleust Datenstrom durch Algorithmus (Stream Computing)

2004: „BrookGPU“ (2)

- Neue Grafikkartentreiber erfordern fortlaufende API-Anpassungen
- -> Industrie verzichtet auf Einsatz
- Trotzdem großer Erfolg bei Wissenschaftlern und Hobby-Programmierern

2004: „BrookGPU“ (3)

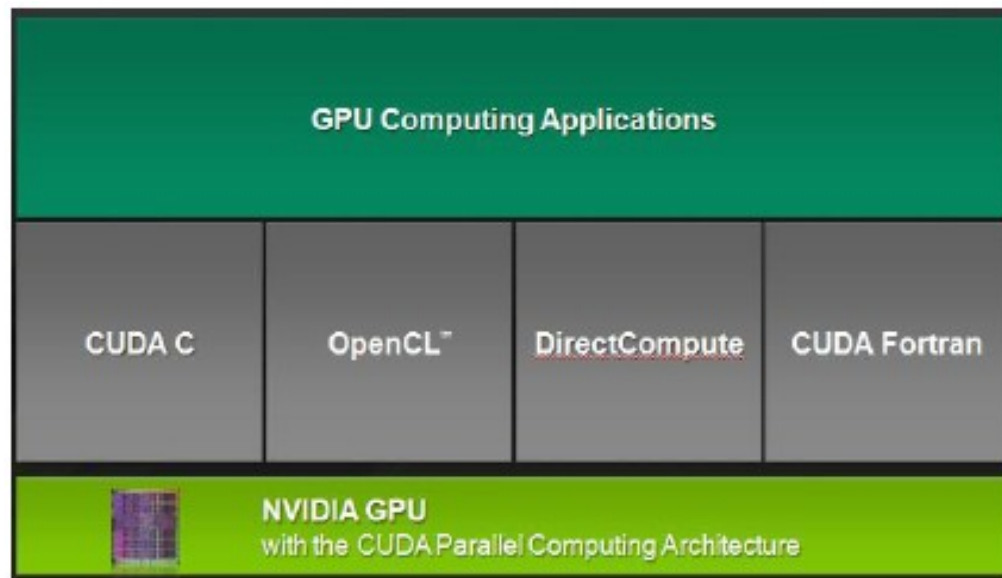


CUDA

Was ist CUDA?

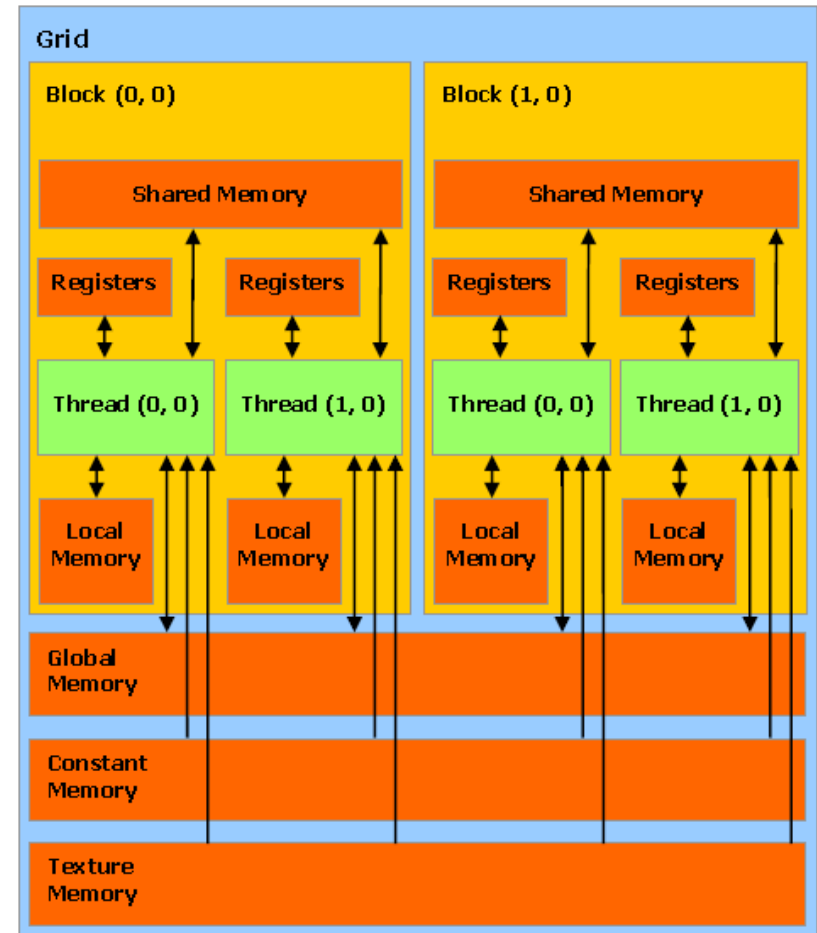
- Compute Unified Device Architecture
- API zur Nutzung der GPU als Co-Prozessor
- 2006 von Nvidia eingeführt
- Unterstützung: Ab GeForce 8 (G80)
- Hard- und Softwareseitig implementiert
- Programmiersprache: C, ab „Fermin“ C++
- Wrapper für Python, Java, .Net und andere

Architektur von CUDA



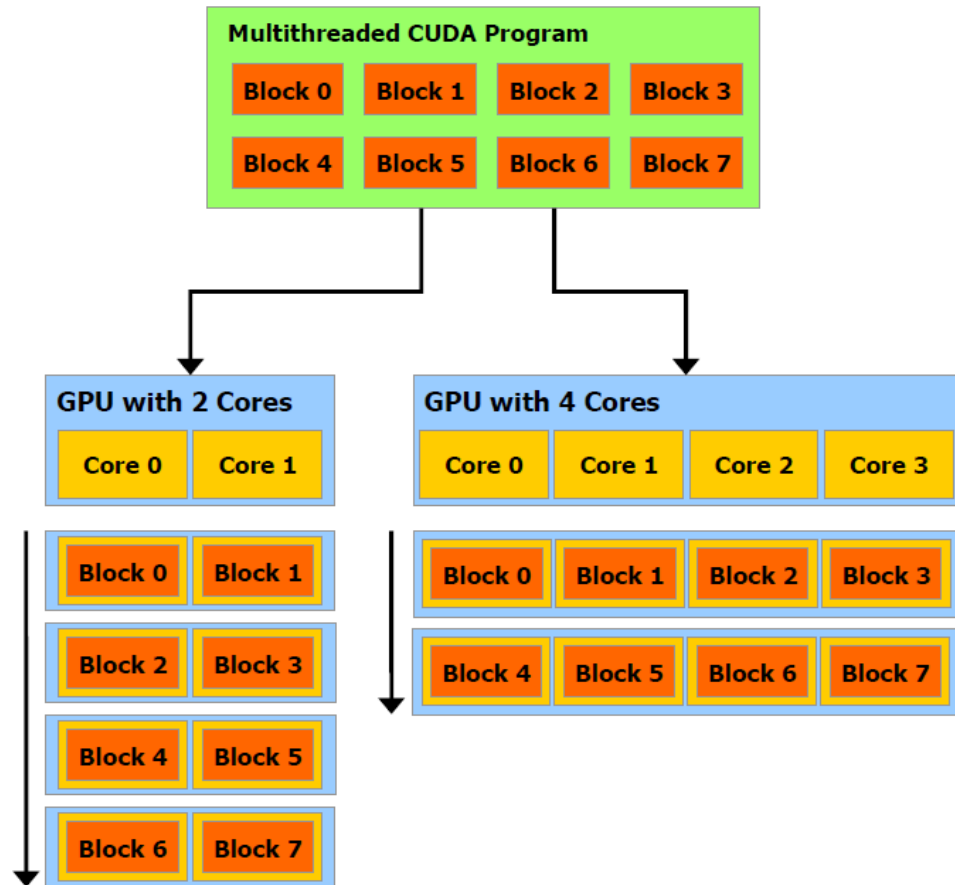
CUDA - Hardwareseitig

- Threads
- Blöcke
- Grids



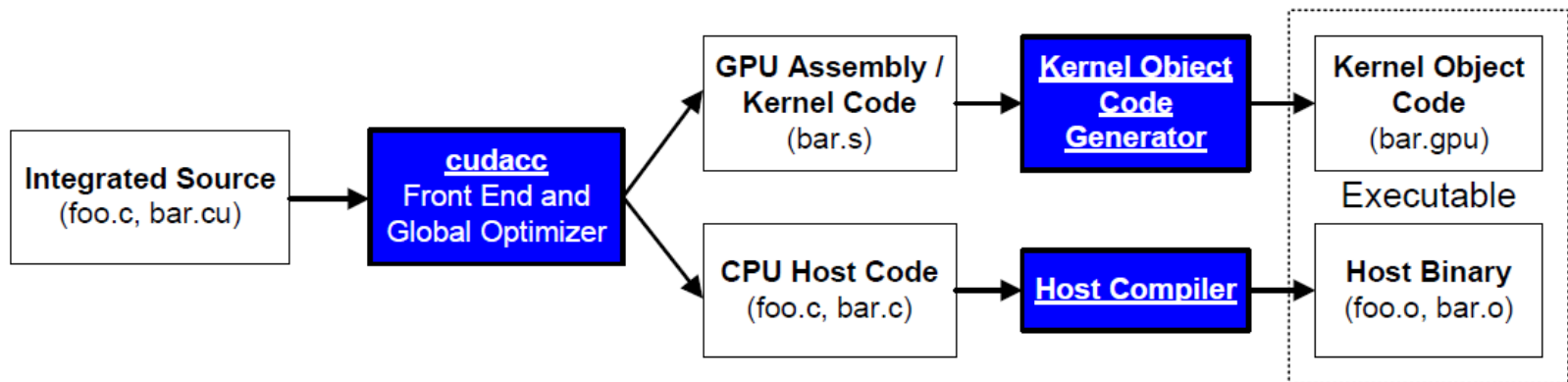
Autoskalierung

- Daten werden in Blöcke zerlegt
- CUDA übernimmt Skalierung



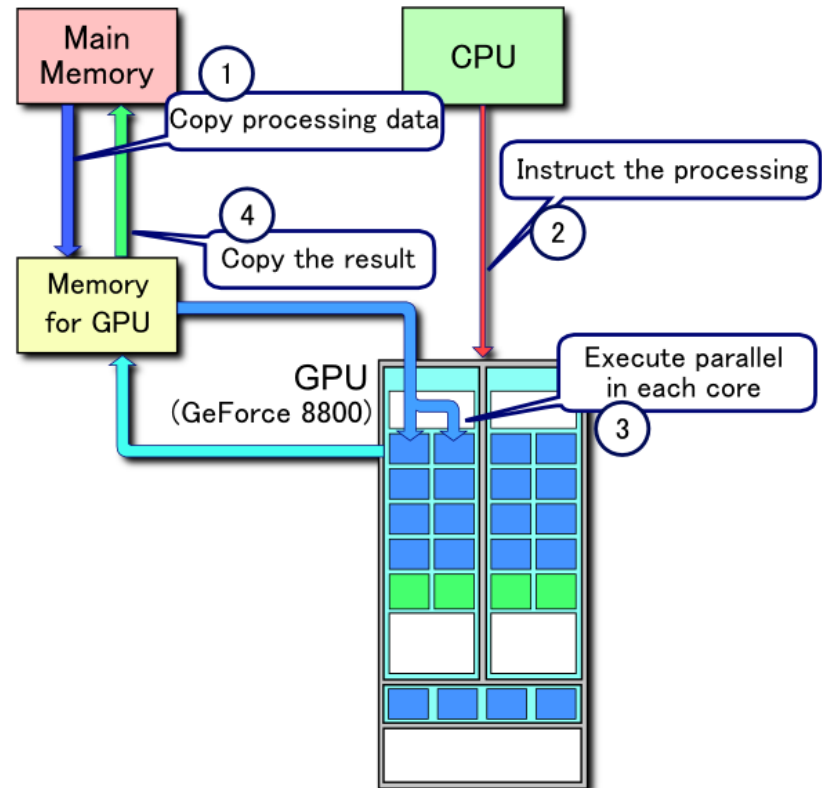
Aufbau CUDA-Programm

- Host (CPU)
 - Bereitstellung von Daten und Entgegennahme von Ergebnissen
- Kernel (GPU)
 - Rechenanweisungen



Ausführung CUDA-Programm

1. Copy data from main mem to GPU mem
2. CPU instructs the process to GPU
3. GPU execute parallel in each core
4. Copy the result from GPU mem to main mem

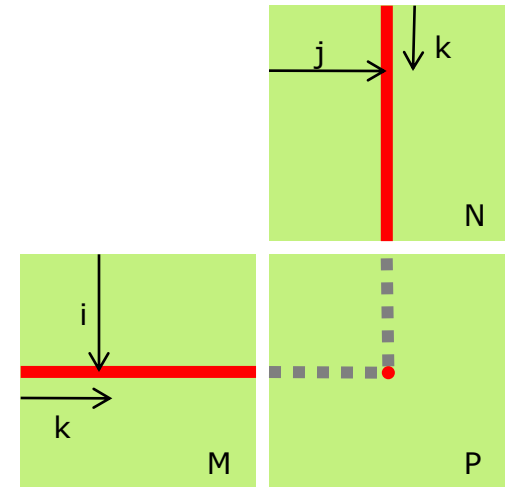


Matrizen-Multiplikation

BEISPIEL

CPU Version

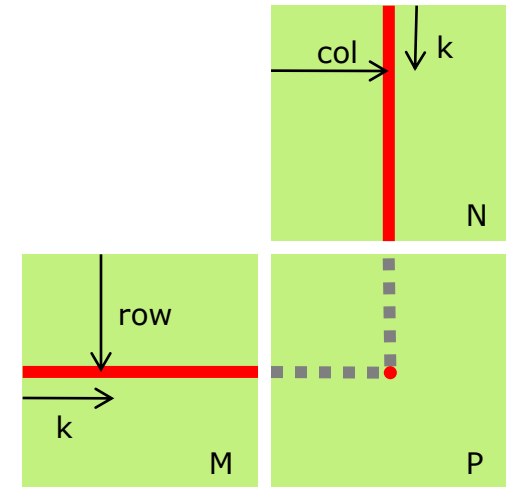
```
1 void MatrixMulOnHost( float* M, float* N, float* P, int Width) {  
2     for (int i = 0; i < Width; ++i)  
3     for (int j = 0; j < Width; ++j) {  
4         float sum = 0;  
5         for (int k = 0; k < Width; ++k) {  
6             float a = M[i * Width + k];  
7             float b = N[k * Width + j];  
8             sum += a * b;  
9         }  
10        P[i * Width + j] = sum;  
11    }  
12 }
```



Quelle: <http://www.cse.shirazu.ac.ir/~azimi/gpu89/lectures/07-MatMult-Basic.pdf>

GPU Version (1)

```
1  __global__
2  void MatrixMulKernel(float* M, float* N, float* P, int Width) {
3      int row = threadIdx.y;
4      int col = threadIdx.x;
5      float P_val = 0;
6      for (int k = 0; k < Width; ++k) {
7          float M_elem = M[row * Width + k];
8          float N_elem = N[k * Width + col];
9          P_val += M_elem * N_elem;
10     }
11     P[row * Width + col] = P_val;
12 }
```



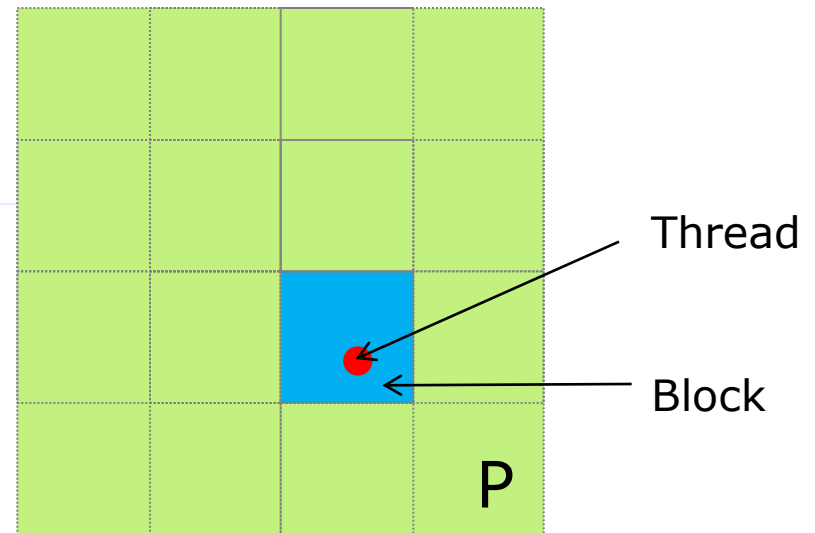
Problem

- Wird in nur einem Block ausgeführt
- Matrix-Größe ist auf max. Anzahl der Threads eines Blocks beschränkt
- Matrizen liegen im Global Memory -> langsam

Quelle: <http://www.cse.shirazu.ac.ir/~azimi/gpu89/lectures/07-MatMult-Basic.pdf>

GPU Version (2)

```
1  __global__
2  void MatrixMulKernel(float* d_M, float* d_N, float* d_P, int Width) {
3      int row = blockIdx.y * blockDim.y + threadIdx.y;
4      int col = blockIdx.x * blockDim.x + threadIdx.x;
5      float P_val = 0;
6      for (int k = 0; k < Width; ++k) {
7          float M_elem = d_M[row * Width + k];
8          float N_elem = d_N[k * Width + col];
9          P_val += M_elem * N_elem;
10     }
11     d_p[row*Width+col] = P_val;
12 }
```



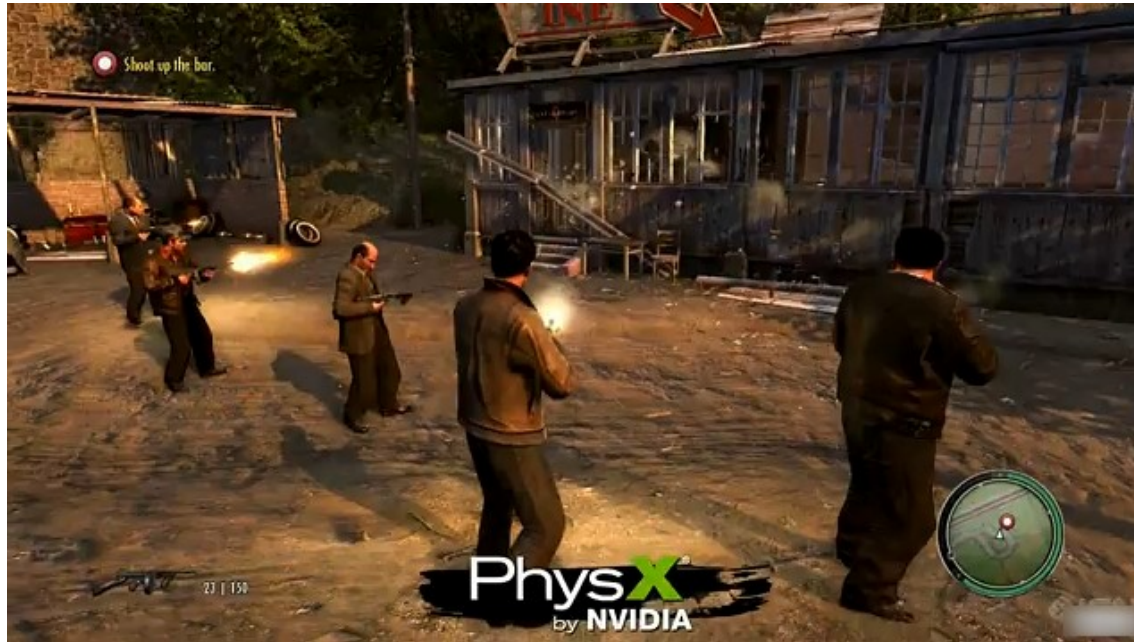
Quelle: <http://www.cse.shirazu.ac.ir/~azimi/gpu89/lectures/07-MatMult-Basic.pdf>

ANWENDUNG

Anwendung

- physikalische Simulationen (Strömung, Gravitation, Temperatur, Crash-Tests)
- Wettervorhersage
- Datenanalyse und Finanzmathematik
- Verarbeitung von akustischen und elektrischen Signalen
- CT- und Ultraschall-Bildrekonstruktion
- Kryptografie
- ...

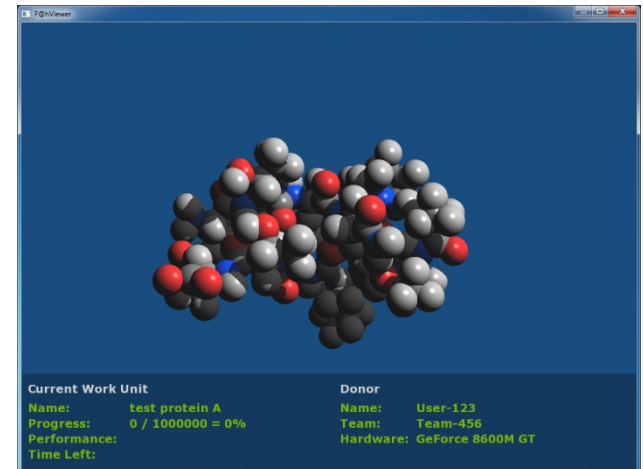
Nvidia PhysX



<http://www.youtube.com/watch?v=0v1NgzyQ9tM>

Folding@home

- Projekt der Stanford University
- Simulation der Proteinfaltung
- Zur Erforschung von Alzheimer, BSE und Krebs
- Läuft als im Hintergrund
- Distributed Computing
- <http://folding.stanford.edu/>



Adobe Creative Suite

- Seit CS4 Unterstützung durch CUDA
- Rendern von Videos
- Drehen und Skalieren von Bildern
- Berechnung von Filtern und Effekten



Quellen

- http://www.nvidia.de/object/cuda_home_new_de.html
- http://www.nvidia.com/object/what_is_cuda_new.html
- http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/CUDA_C_Programming_Guide.pdf
- http://www.nvidia.com/content/CUDA-ptx_isa_1.4.pdf
- <http://ambermd.org/gpus/benchmarks.htm>
- http://www.tomshardware.de/CUDA-Nvidia-CPU-GPU_testberichte-240065.html
- <http://en.wikipedia.org/wiki/CUDA>
- <http://de.wikipedia.org/wiki/Grafikkarte>
- http://de.wikipedia.org/wiki/Apple_II
- <http://www.informatik.uni-hamburg.de/WSV/teaching/sonstiges/EwA-Folien/Mueller-Paper.pdf>
- <http://www.cis.udel.edu/~cavazos/cisc879/papers/ppopp-08-ryoo.pdf>
- <http://www.cse.shirazu.ac.ir/~azimi/gpu89/lectures/07-MatMult-Basic.pdf>
- [c't 11/2009, „Parallel-Werkzeuge“, Seite 142ff, Manfred Bertuch](#)



Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

**Vielen Dank für Ihre
Aufmerksamkeit!**