NVIDIA

CUDA

by Michael Schwabl, Stefan Hasslacher, Michael Hasslacher

Was ist CUDA?

Compute Unified Device Architecture

Eine Schnittstelle, die es ermöglicht, die GPU eines Computers für Berechnungen zu nutzen

Motivation für CUDA?

- Die vielen Recheneinheiten der Grafikkarte nutzen
- Ressourcen eines Computers optimal nutzen.
- Billige „Beschleunigerboards“
- Keine spezielle Hardware muss entwickelt werden
- Einen leichteren Zugang zu den Grafikkartenressourcen zu schaffen

CUDA

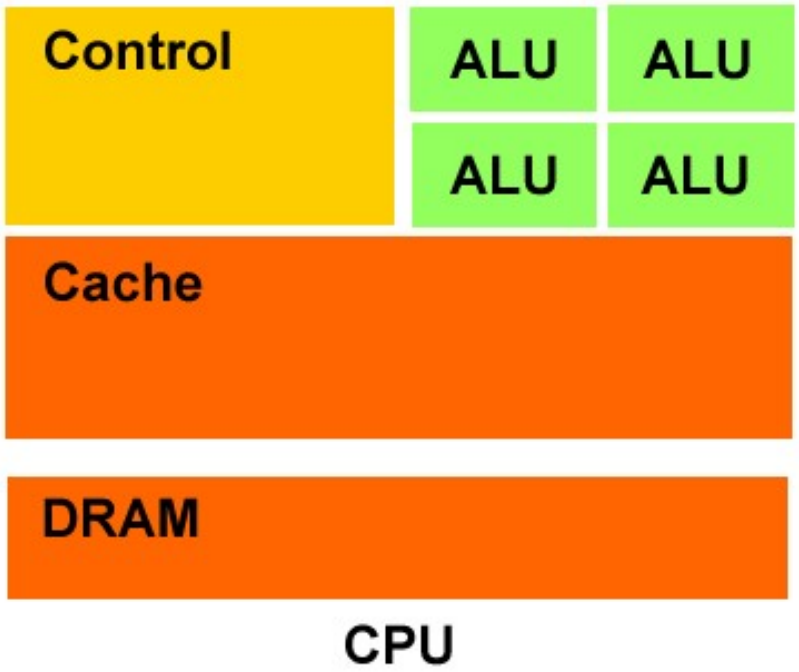
Ein Problem und die Lösung

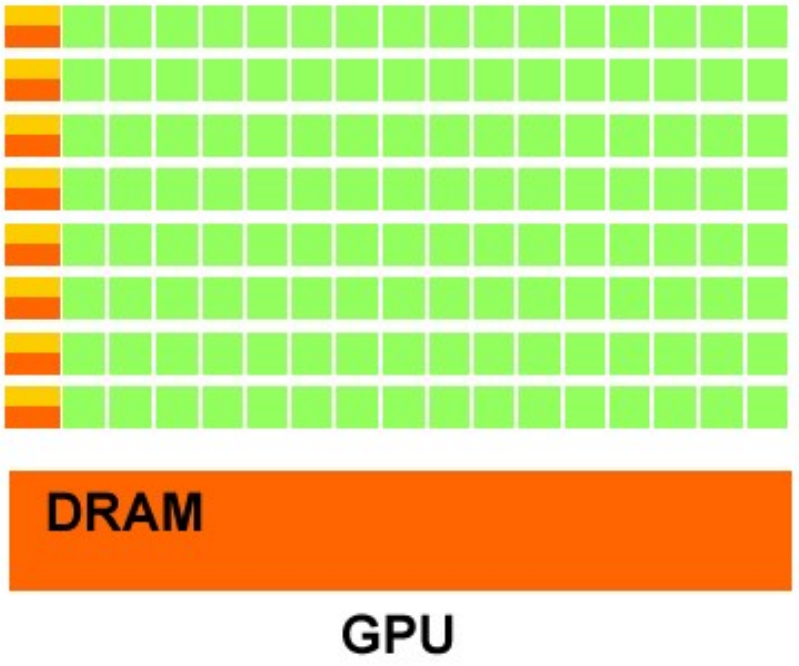
- **Zugriff nur über Direct3D und OpenGL möglich**
- **Analogien zwischen Programmierung und Grafikprogrammierung finden**

- **Stream ⇨ Textur**
- **Kernel ⇨ Pixel-Shader**
- **Speicherort bestimmen ⇨ Vertex-Shader**

CUDA

Unterschiede von CPU und GPU





CUDA

Unterschiede von CPU und GPU

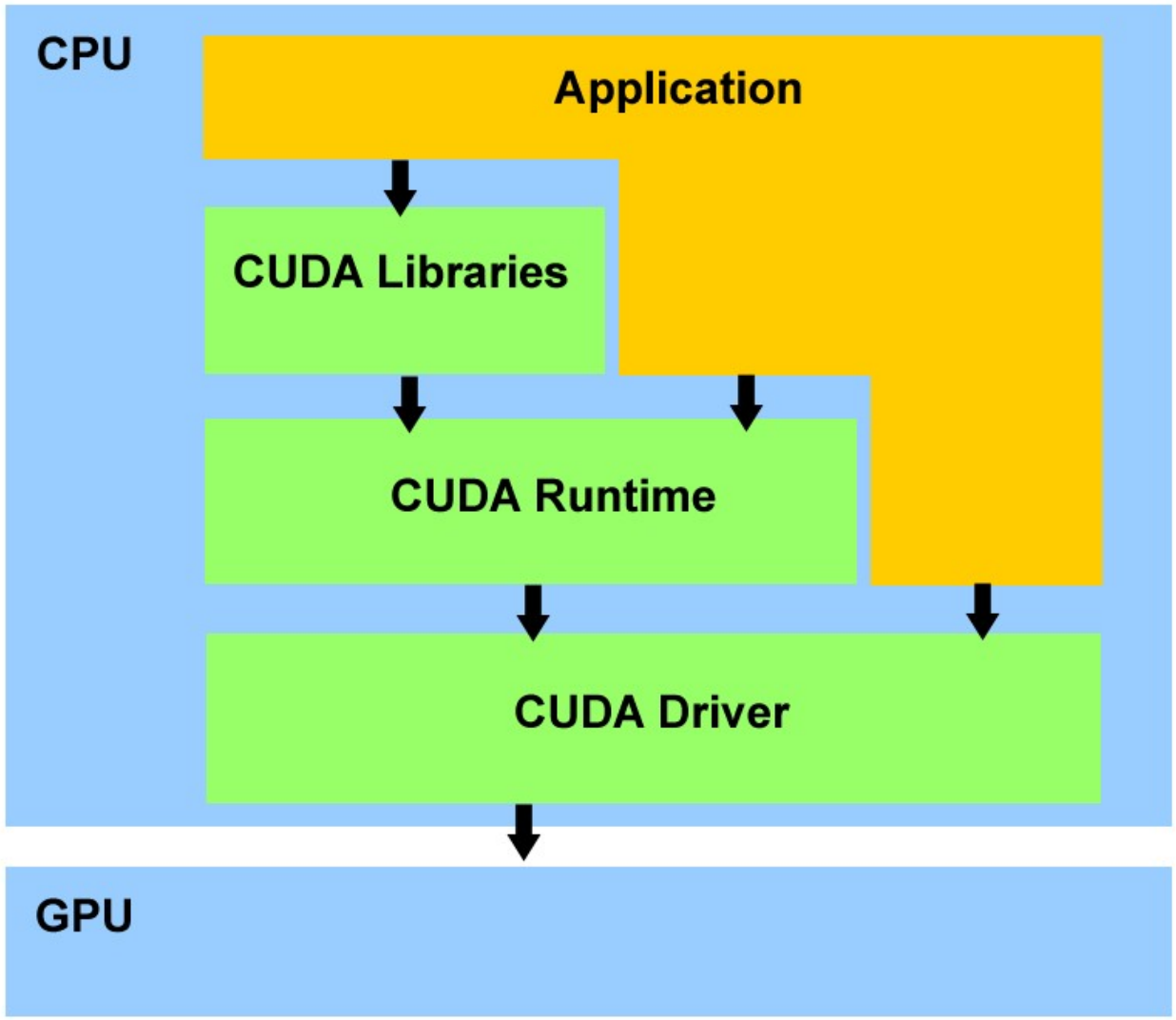
processor	**Intel Core 2 Extreme QX9650**	**Nvidia Geforce GTX 280**
transistor	820 million	1.4 billion
processor clock	3 GHz	1296 MHz
cores	4	240
cache / shared memory	6 MB x 2	16 KB x 30
threads executed per clock	4	240
peak gigaflops	96 gigaflops	933 gigaflops
memory controllers	Off-die	8 x 64-bit
memory bandwidth	12.8 GBps	141.7 GBps

Was bietet CUDA?

- Standard-C/C++-Entwicklungsumgebung
- Standardisierte Bibliotheken
- Spezieller CUDA-Treiber
- CUDA-Treiber interagiert mit OpenGL und DirectX
- Unterstützung für Linux, Windows und Mac Betriebssysteme
- breite Basis an Hardware

CUDA

CUDA API



Qualifizierer

__global__

__device__

__host__

cudaMalloc

cudaFree

cudaMemcpy

Codebeispiel

```
01  // example1.cpp : Defines the entry point for the console application.
02  //
03
04  #include "stdafx.h"
05
06  #include <stdio.h>
07  #include <cuda.h>
08
09  // Kernel that executes on the CUDA device
10  __global__ void square_array(float *a, int N)
11  {
12    int idx = blockIdx.x * blockDim.x + threadIdx.x;
13    if (idx<N) a[idx] = a[idx] * a[idx];
14  }
15
```

CUDA

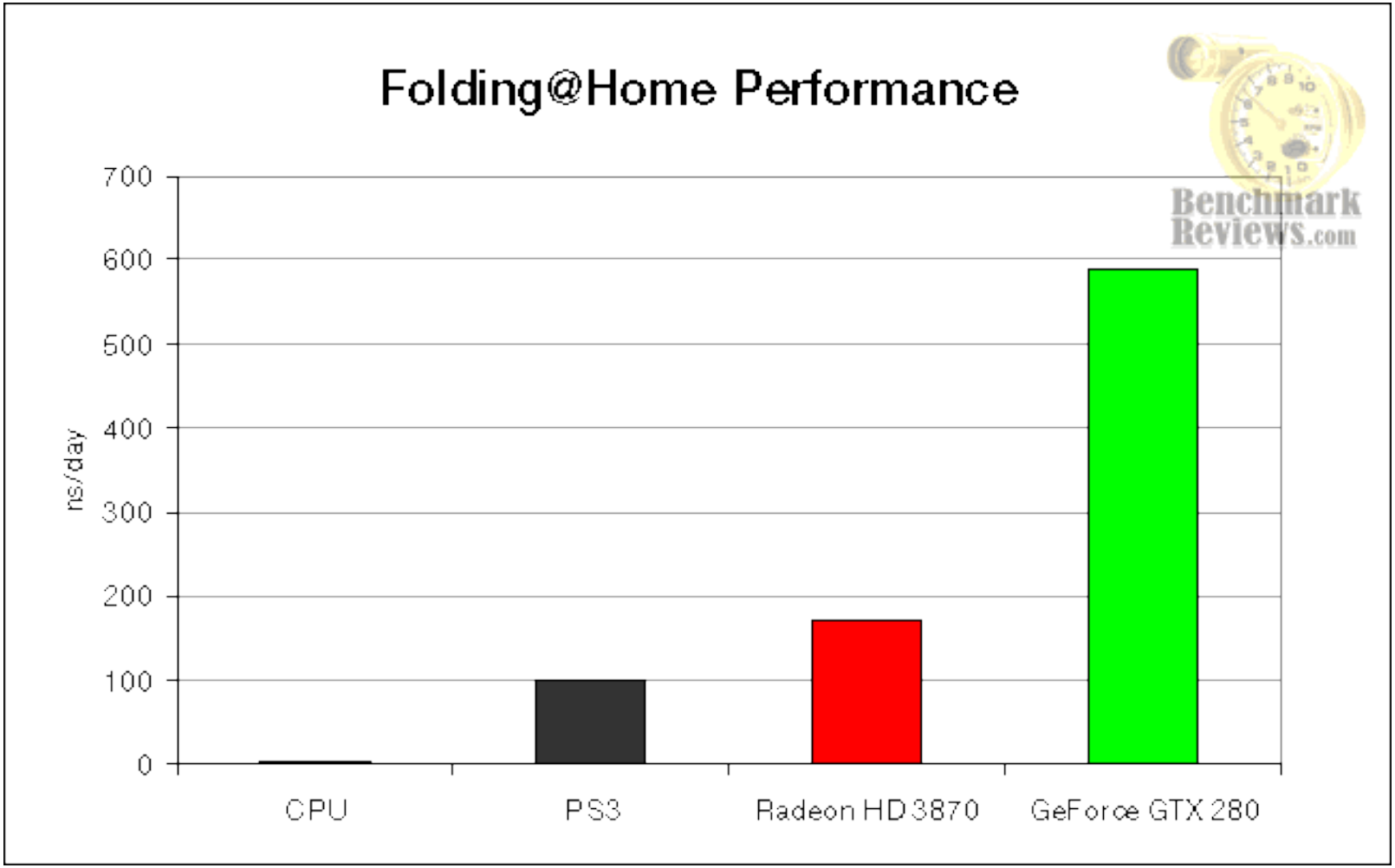
Codebeispiel

```
16  // main routine that executes on the host
17  int main(void)
18  {
19    float *a_h, *a_d;  // Pointer to host & device arrays
20    const int N = 10;  // Number of elements in arrays
21    size_t size = N * sizeof(float);
22    a_h = (float *)malloc(size);        // Allocate array on host
23    cudaMalloc((void **) &a_d, size);   // Allocate array on device
24    // Initialize host array and copy it to CUDA device
25    for (int i=0; i<N; i++) a_h[i] = (float)i;
26    cudaMemcpy(a_d, a_h, size, cudaMemcpyHostToDevice);
27    // Do calculation on device:
28    int block_size = 4;
29    int n_blocks = N/block_size + (N%block_size == 0 ? 0:1);
30    square_array <<< n_blocks, block_size >>> (a_d, N);
31    // Retrieve result from device and store it in host array
32    cudaMemcpy(a_h, a_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
33    // Print results
34    for (int i=0; i<N; i++) printf("%d %f\n", i, a_h[i]);
35    // Cleanup
36    free(a_h); cudaFree(a_d);
37  }
```

Quelle: llpanorama *http://llpanorama.wordpress.com*

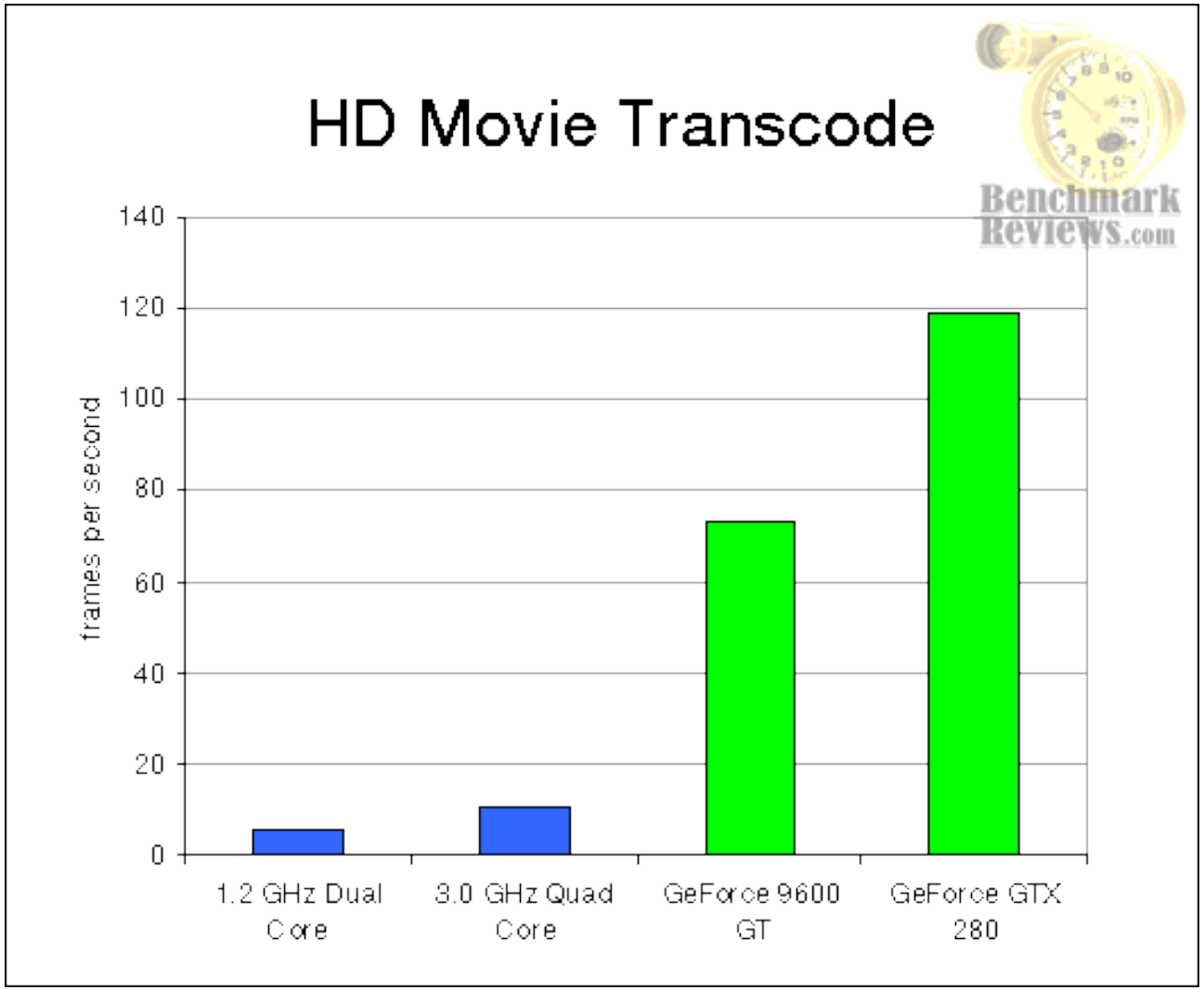
CUDA

Beispiele für beschleunigte Apps



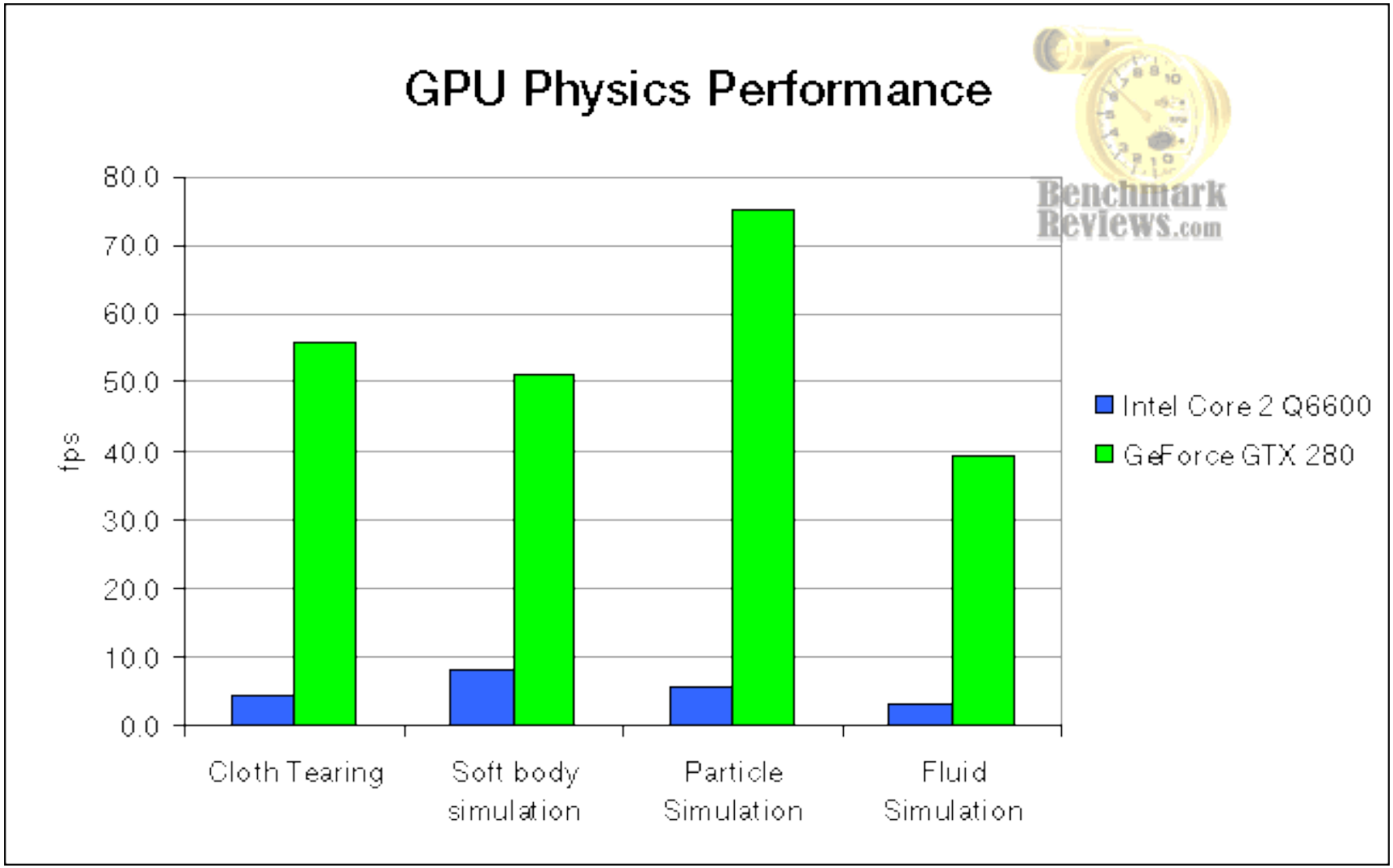
CUDA

Beispiele für beschleunigte Apps



CUDA

Beispiele für beschleunigte Apps



Zusätzliche Informationen

für SDKs, Bibliotheken, Treiber und Tutorials

http://www.nvidia.de/object/cuda_learn_de.html