GPU-Programmierung: OpenCL

Markus Hauschild

Seminar: Multicore Programmierung
Sommerstemester 2009

04.06.2009

Inhaltsverzeichnis

GPU-Programmierung

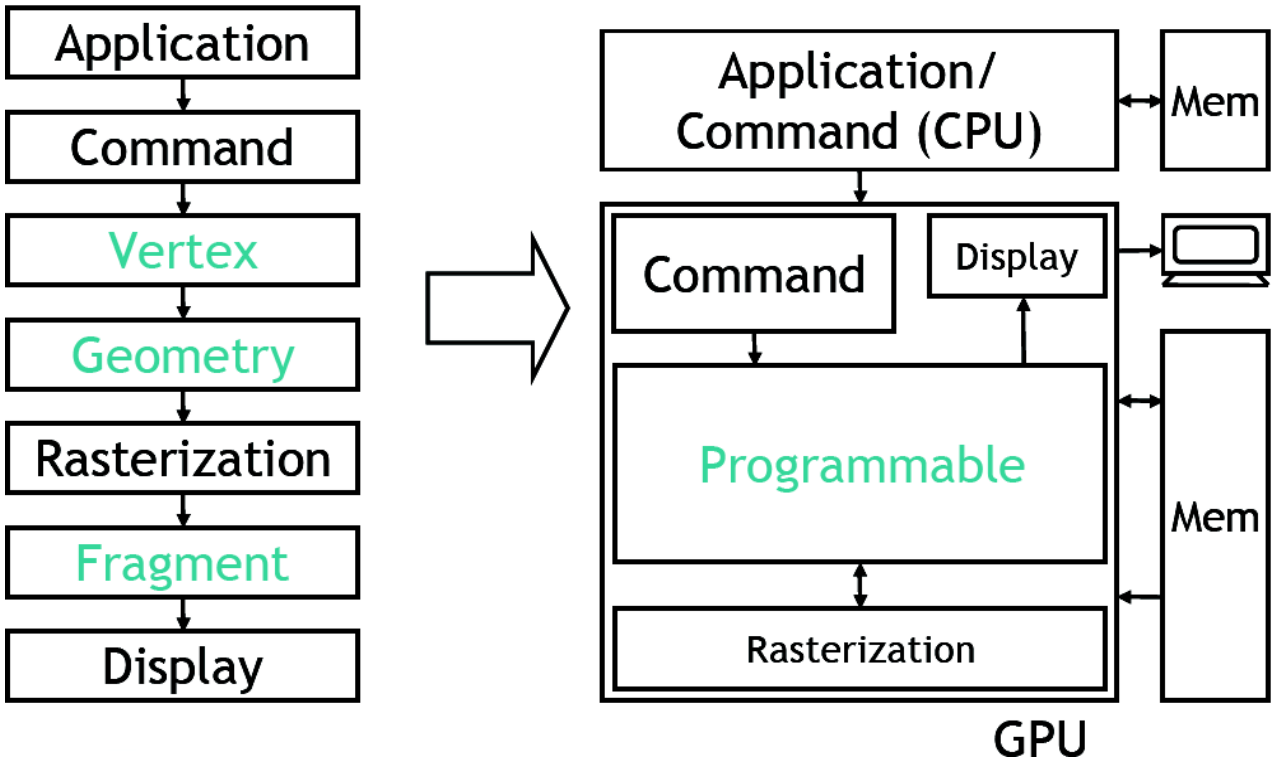
Was ist GPU-Computing

Unter GPU-Computing versteht man die Nutzung der Grafikkarte zur Berechnung allgemeiner Probleme.

Wieso

- Günstige Hardware durch die Verbereitung von PC-Spielen
- Speedup um Faktor 10 bis 100 im Vergleich zu CPUs (bei geeigneten Anwendungen)

Entwicklung von Grafikkarten



[Owe07]

Einsatzgebiete von GPU-Computing

Allgemein

- Streamcomputing
- Anwendungen mit hoher Datenparallelität

Beispiele

- Signalverarbeitung
- Audio/Bild/Video-Verarbeitung
- Raytracing
- Kryptographie
- Physik-Simulation/Effekte

Übersicht

Sprachen und APIs

- Shader in OpenGL/DirectX
- BrookGPU
- CUDA
- OpenCL

Shader in OpenGL/DirectX (1)

Entwicklung

Erste wissenschaftliche Nutzung mit Einführung von Gleitkommazahlen in Shadern mit DirectX 9 kompatiblen Grafikkarten.

Vorteile/Neuerungen

- Erstmalige Nutzung der Rechenleistung von Grafikkarten

Nachteile

- Algorithmen als Pixel-Shader
- Manuelles Ver- und Entpacken der Daten in Texturen
- Texturdimension $2^n \times 2^m$, max 4096×4096

Shader in OpenGL/DirectX (2)

Als Beispiel für die vorgestellten Sprachen soll die folgende Berechnung in C dienen:

Berechnung in C

```
for (int i=0; i<N; i++)
{
  z[i] = alpha * x[i] + y[i];
}
```

Shader in OpenGL/DirectX (3)

Berechnung als Shader (in Cg)

```
float4 saxpy (
  float2 coords : TEXCOORD0,
  uniform samplerRECT textureY,
  uniform samplerRECT textureX,
  uniform float alpha ) : COLOR
{
  float4 y = texRECT(textureY,coords);
  float4 x = texRECT(textureX,coords);
  return alpha * x + y;
}
```

BrookGPU (1)

Entwicklung

Entwickelt an der Stanford University seit 2003.

Vorteile/Neuerungen

- Einführung des kernel
- API vereinfacht kopieren von Daten in den/aus dem Grafikspeicher

BrookGPU (2)

Nachteile

- BrookGPU erzeugt Shader, daher z.B. keine Nutzung des Scratchpad möglich
- Nur Streamcomputing möglich, d.h. keine Arrays als kernel-Parameter

BrookGPU (3)

Berechnung als Brook kernel

```
kernel void k(float y<>, float x<>,
              float alpha, out float z<>)
{
  z = alpha * x + y;
}
```

CUDA (1)

Vorteile/Neuerungen

- Direkte Ausführung auf der Hardware durch den Grafikkartentreiber
- Gruppierung und Indizierung von Threads
- Arrays statt Streams als Parameter
- Nutzung weiterer Hardware Features wie z.B. dem Scratchpad

Nachteile

- Läuft nur auf Nvidia Grafikkarten
- Unzureichende Fehlerbehandlung in kerneln

CUDA (2)

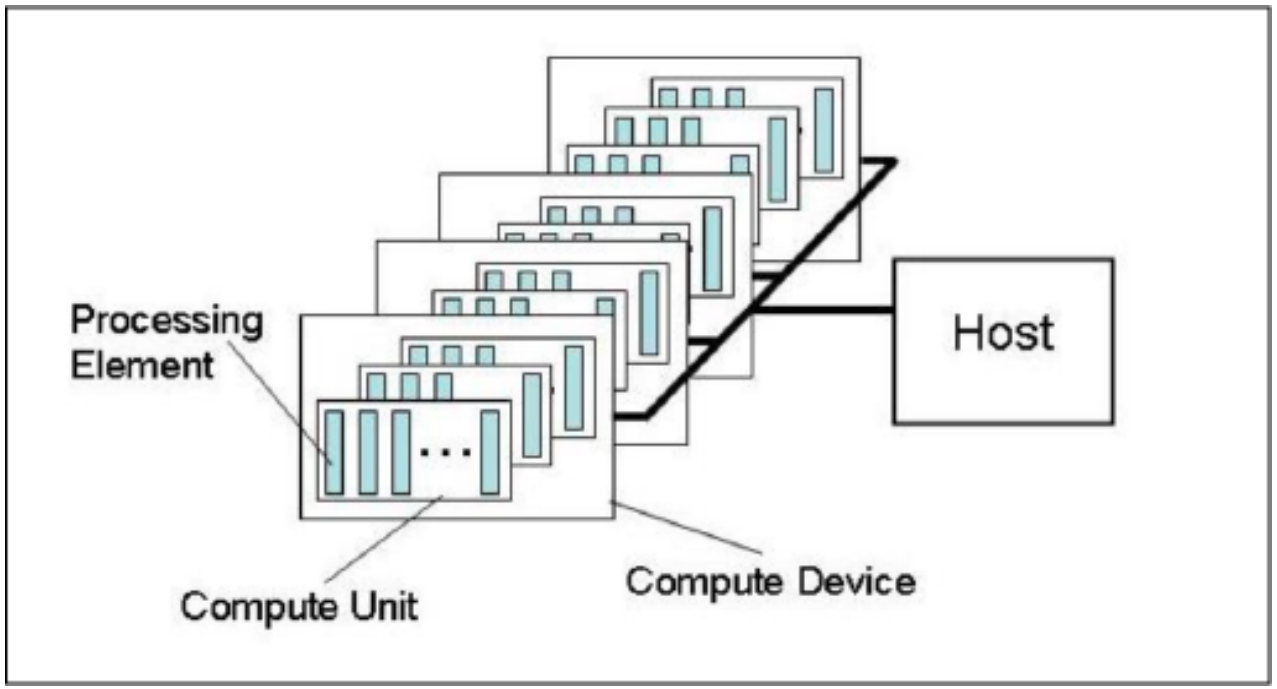
Berechnung als CUDA kernel

```
__global__ void k(float* x, float* y,
                  float alpha, float* z)
{
  int i = threadIdx.x +
          blockDim.x * blockIdx.x;
  z[i] = alpha * x[i] + y[i];
}
```

Entwicklung

- offener Stanard von der Khronos Group betreut
- initiiert von Apple, AMD, Intel und Nvidia
- Juni 2008: OpenCL working group bei Khronos eingerichtet
- Dezember 2008: Spezifikation fertiggestellt

Architektur



[Khr09]

Execution Model (1)

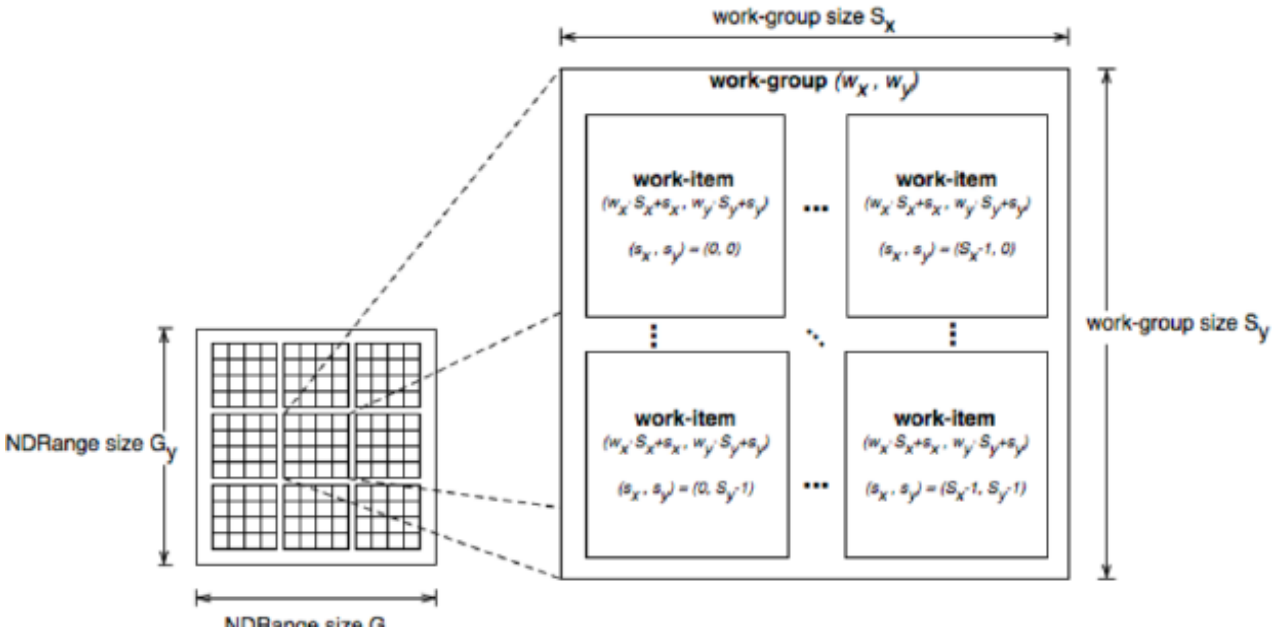
Bestandteile eines OpenCL Programms

- Host-Anwendung (Initialisierung, Speicherverwaltung, etc)
- Kernel (enthält den auf dem OpenCL-Device auszuführenden Code)

Kernel Ausführung

- Hostprogramm ruft kernel auf einem Indexraum mit 1-3 Dimensionen auf
- Eine Instanz eines kernels ist ein work-item
- Work-items sind in work-groups aufgeteilt

Execution Model (2)

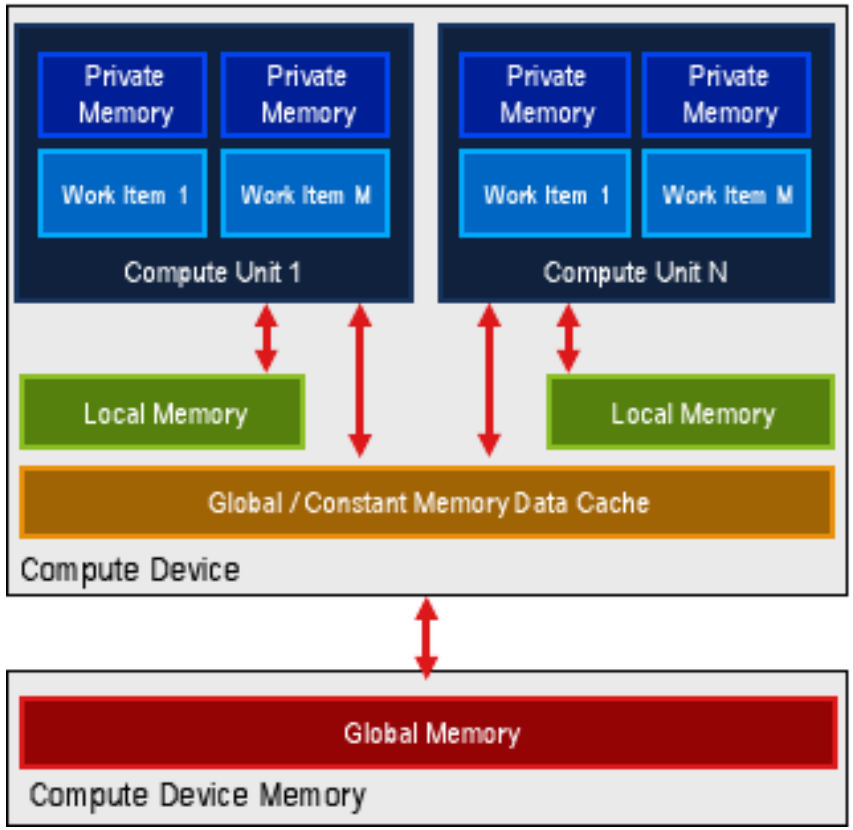


Command-queue

Command-queue

- Kernelausführung
- Speichertransfer
- Synchronisation

Speichermodell (1)



Speichermodell (2)

Adressräume

- private - zu einem work-item gehörend
- local - zu einer work-group gehörend
- global - von allen work-items aus allen work-groups benutzbar
- constant - nur lesender zugriff möglich

Spracheigenschaften

Kernel erlauben im wesentlichen C99, ohne

- Funktionspointer
- Rekursion
- Arrays variabler Länge
- structs

optionale Features

- Gleitkommezahlen mit doppelter Genauigkeit
- Atomare Funktionen (z.B. increment)

Vergleich mit CUDA

Unterschiede

- API-Befehle ähneln sich, haben jedoch z.B. unterschiedliche Parameter
- Pointer müssen in OpenCL mit einem `address space qualifier` versehen werden
- Command-Queues
- OpenCL kompiliert kernel erst zur Laufzeit für die entsprechende Plattform

Beispiel

Berechnung als OpenCL kernel

```
__kernel void k(__global const float* y,
                __global const float* x,
                __global const float alpha,
                __global float* z)
{
  int index = get_global_id(0);
  z[index] = alpha * x[index] + y[index];
}
```

Besonderheiten

Unterstützte Hardware

- (neuere) Grafikkarten von AMD/ATI und Nvidia
- Intel Larrabee
- Cell
- (multicore) CPUs
- embedded/mobile Devices

Verfügbarkeit

AMD/ATI

Treiber angekündigt für das zweite Halbjahr 2009.

Nvidia

Beta-Treiber verfügbar seit Ende April und zur Zertifizierung eingereicht.

Referenzen


Khronos Group.
OpenCL - The Open Standard for Heterogeneous Parallel Programming, February 2009.
Available online at http://www.khronos.org/developers/library/overview/opencl_overview.pdf.


John Owens.
GPU Architecture Overview.
SIGGRAPH, 2007.
Available online at http://gpgpu.org/static/s2007/slides/02-gpu-architecture-overview-s07.pdf.