OpenCL-Einführung

David Losch

Rahmeninformationen

- Open computing language
- Ursprünglich von Apple entwickelt, dann von *Khronos Group* als Standard übernommen
- Aktuelle Version: 2.0 (November 2013)

Motivation

- Moderne Systeme sind parallele Architekturen
- Multi-core CPUs und GPUs als heterogene Plattform
- CPUs und GPUs haben unterschiedliche Eigenschaften
- OpenCL als generische, offene Plattform(Framework) für heterogene Programmierung

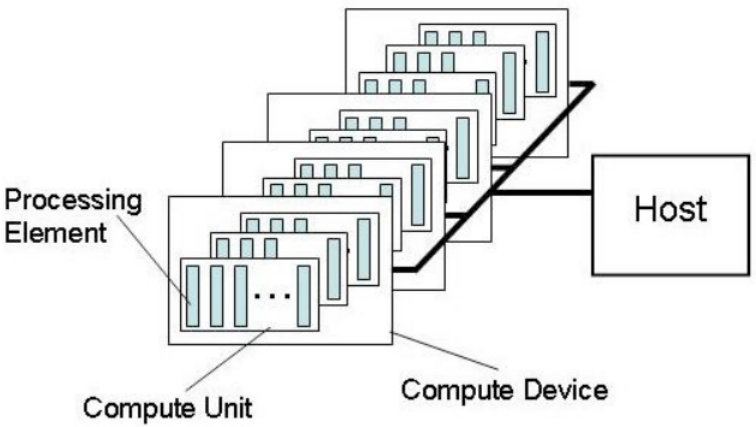
OpenCL

- Unterstützt daten- und taskparallele Programmiermodelle
- Baut auf ISO C99 auf
- Kann mit OpenGL direkt interagieren

Grundstruktur

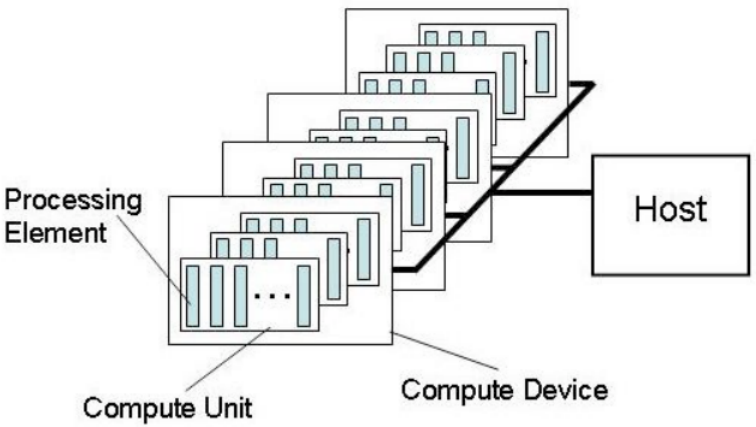
- Platform Model
- Programming Model
- Execution Model
- Memory Model

Platform Model



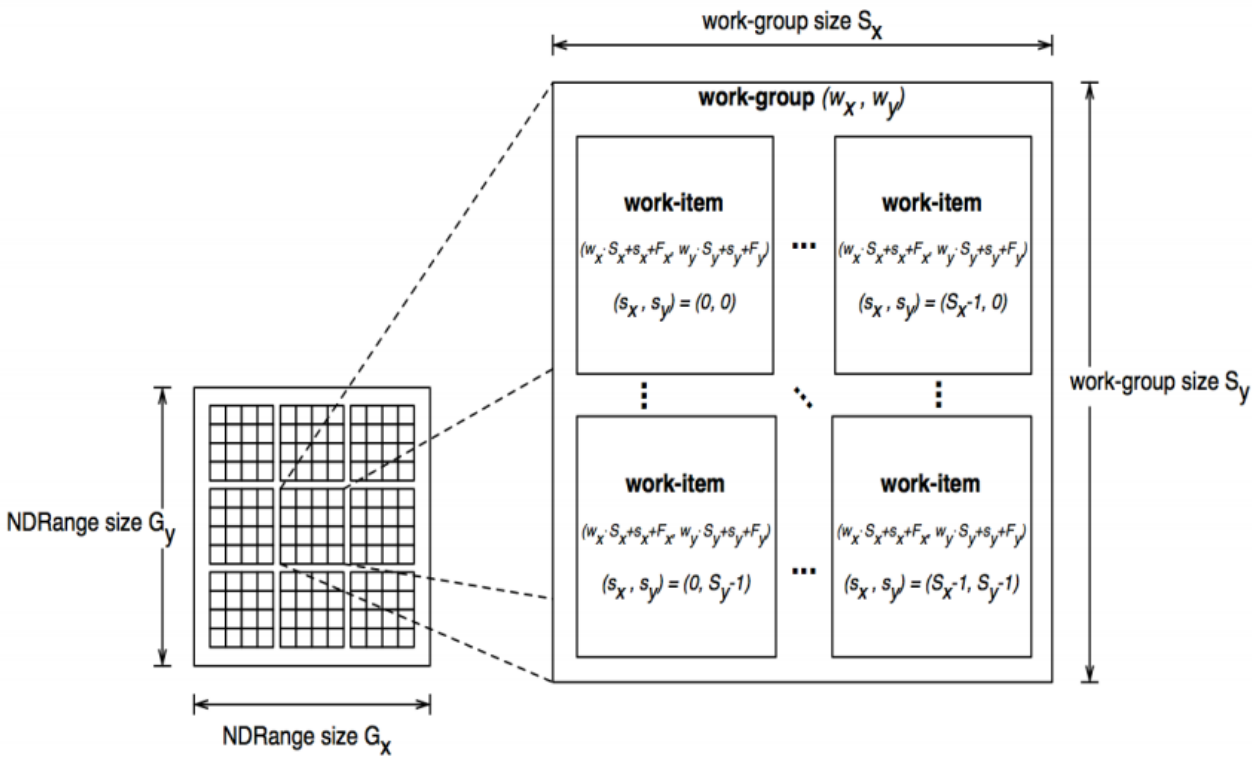
- Host hat Zugang zu einem oder mehreren OpenCL Devices
- Jedes OpenCL Device besteht aus einer oder mehreren Compute Units
- Compute Unit kann in weitere Processing Elements eingeteilt werden

Platform Model



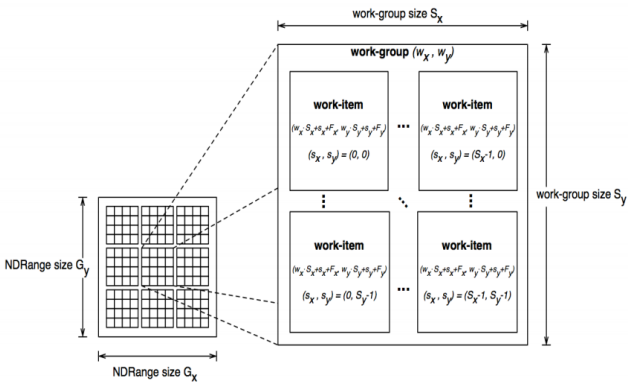
- Das Programm wird auf dem Host ausgeführt und sendet Commands an Device

Execution Model



- Host program ruft Kernels auf, die auf einem Device laufen

Execution Model

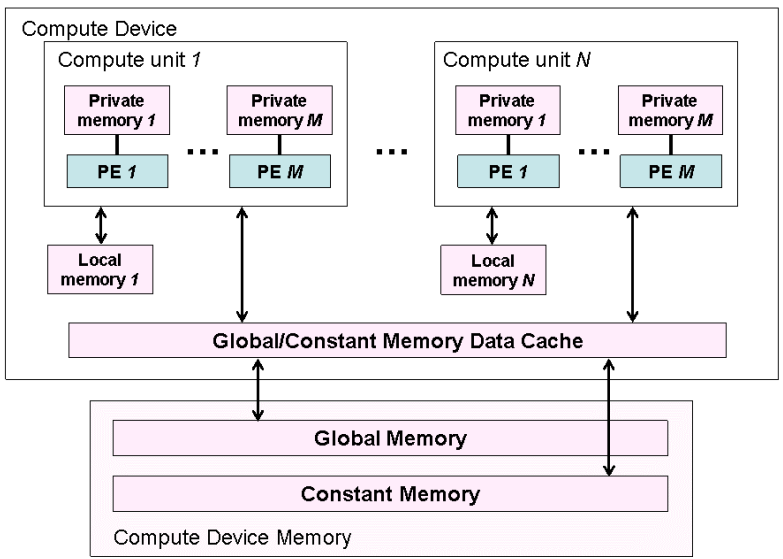


- Instanz eines Kernels = Work Item
- Bei jeder Kernelausführung wird ein Index Space definiert
- Work Item bekommt eine Global ID (pro Dimension)
- Code in einem Work Item immer der gleiche, wobei natürlich unterschiedliche Ausführungspfade entstehen können

Execution Model

- OpenCL implementiert mit diesem Muster sowohl Data Parallelism (SIMD) als auch Task Parallelism

Memory Model



- Es existieren vier verschiedene Speichergebiete

Memory Model

- **Global Memory** kann von jedem Work-Item gelesen und geschrieben werden und wird je nach Device eventuell auch gecachet
- **Constant Memory** kann nur vom Host vor dem Ausführen eines Kernels beeinflusst werden
- **Local Memory** kann von allen Work-Items einer Work-Group gelesen und geschrieben werden und muss daher auch synchronisiert werden
- **Private Memory** kann nur von einem Work-Item selbst gelesen und geschrieben werden

OpenCL-Kontext

- **Device:** Geräte, die benutzt werden sollen
- **Program Object:** Quelltext und erzeugte Binary des OpenCL-Codes, der die Kernel implementiert
- **Kernel:** OpenCL-Funktionen, die aufgerufen werden sollen
- **Memory Objects:** Eine Menge von Speicher, der von Host und Device sichtbar ist und von einem Kernel verändert werden kann

OpenCL-Kontext

- Host kann mit Hilfe einer Command-Queue Befehle an das Device senden
- Diese werden auf dem Device geschedulet

Befehle können sein:

- **Kernel Execution Commands**
- **Memory Commands**
- **Synchronization Commands**

OpenCL-Kontext

Befehle in der Command-Queue können unterschiedlich ausgeführt werden:

- **In-order Execution** Serielle Ausführung der Befehle
- **Out-of-order Execution** Befehle werden in gegebener Reihenfolge ausgeführt, aber es können auch Befehle beginnen bevor Vorgänger noch nicht fertig sind

Beispiel Gaussian Blur

- Dauer CPU: 115 ms
- Dauer GPU: 3 ms

Code-Demonstration

Code-Demonstration

Quelle

- Inhalt: Offizielle OpenCL-Spezifikation
- Code: modifiziert von https://bitbucket.org/Anteru/opencltutorial/overview