Bildverarbeitung mit OpenCL

Johannes Hackel und Falco Prescher

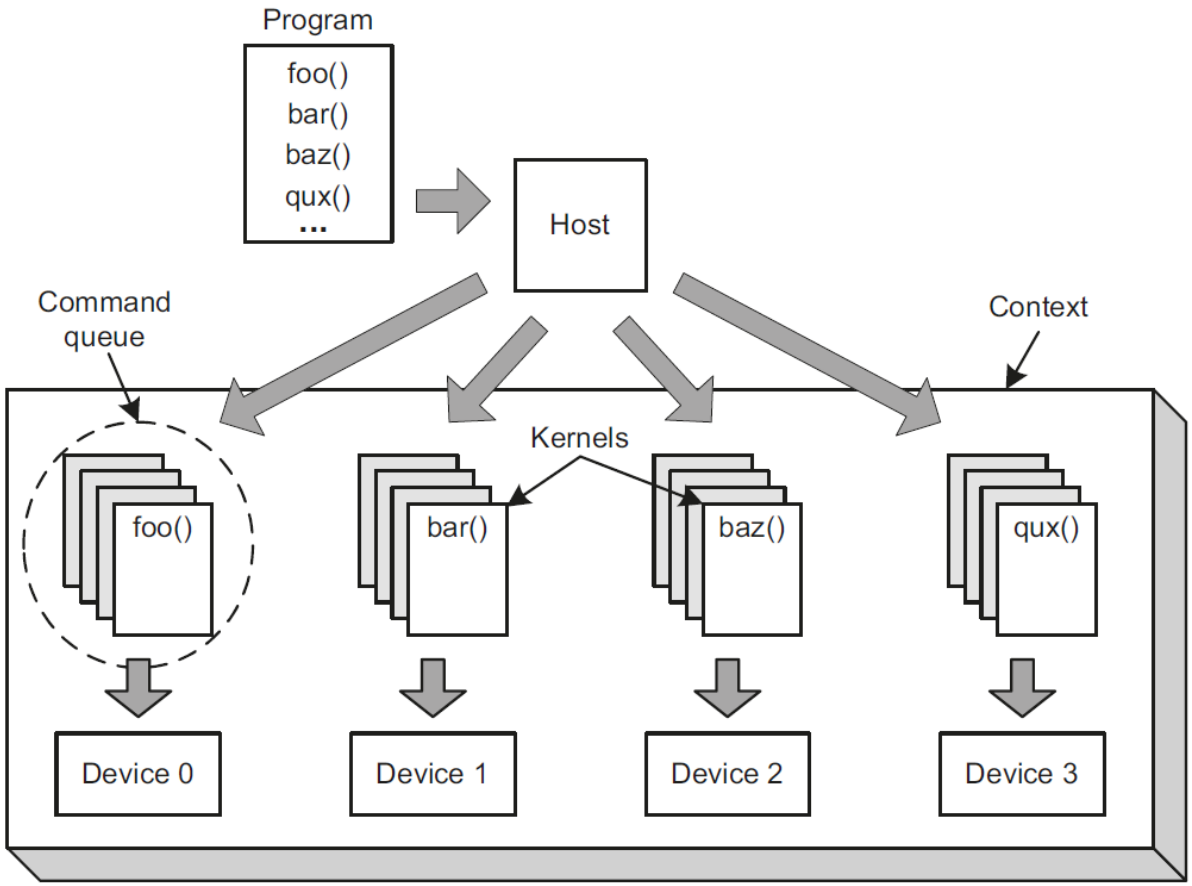
23. Mai 2013

Bildverarbeitung mit OpenCL

Allgemeines zu OpenCL

- Test

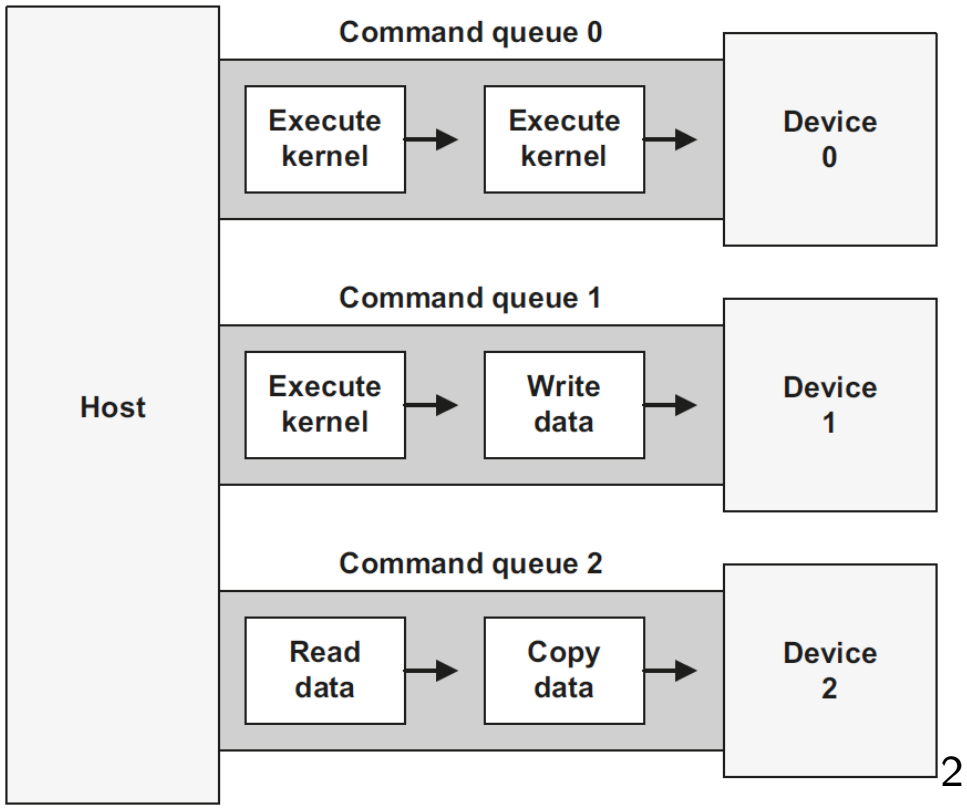
Kernelverteilung unter Devices



[1]

[1] Scarpino Matthew: OpenCL In Action, Manning Publications Co., 2012, S. 8

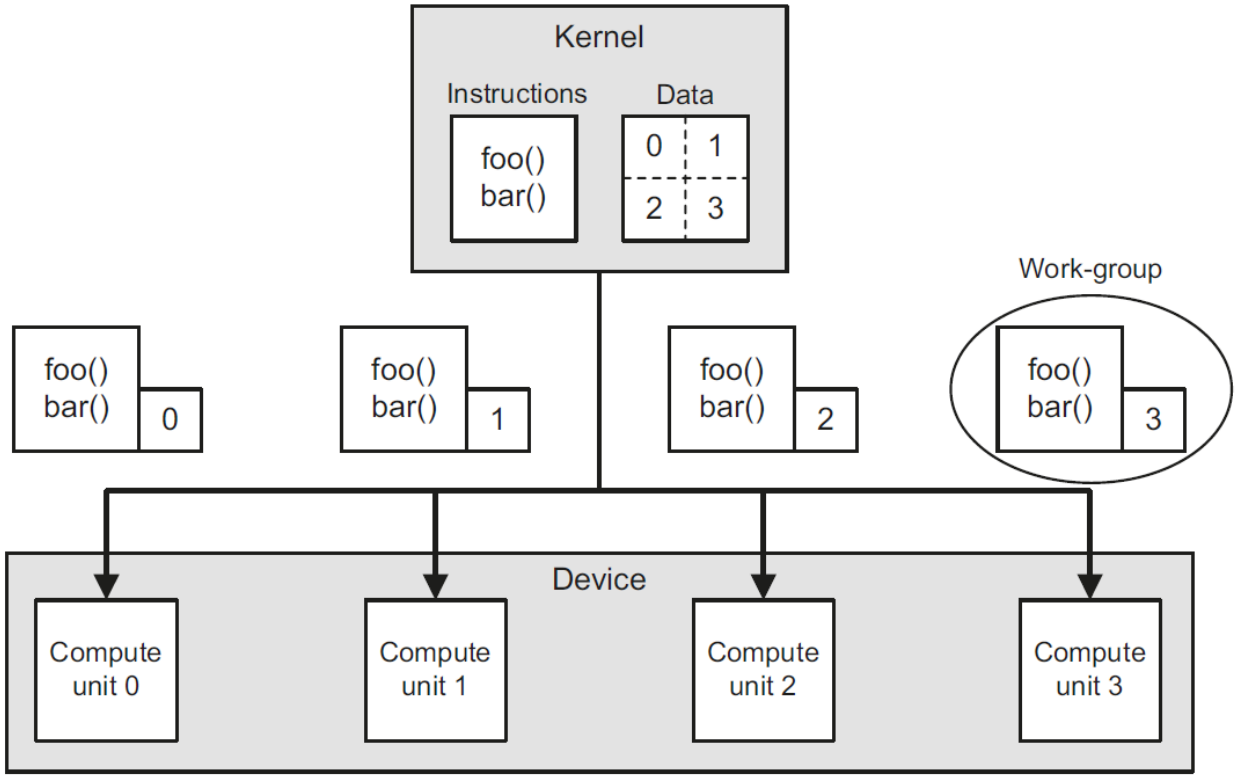
Command Queue



[2]

[2] Scarpino Matthew: OpenCL In Action, Manning Publications Co., 2012, S. 39

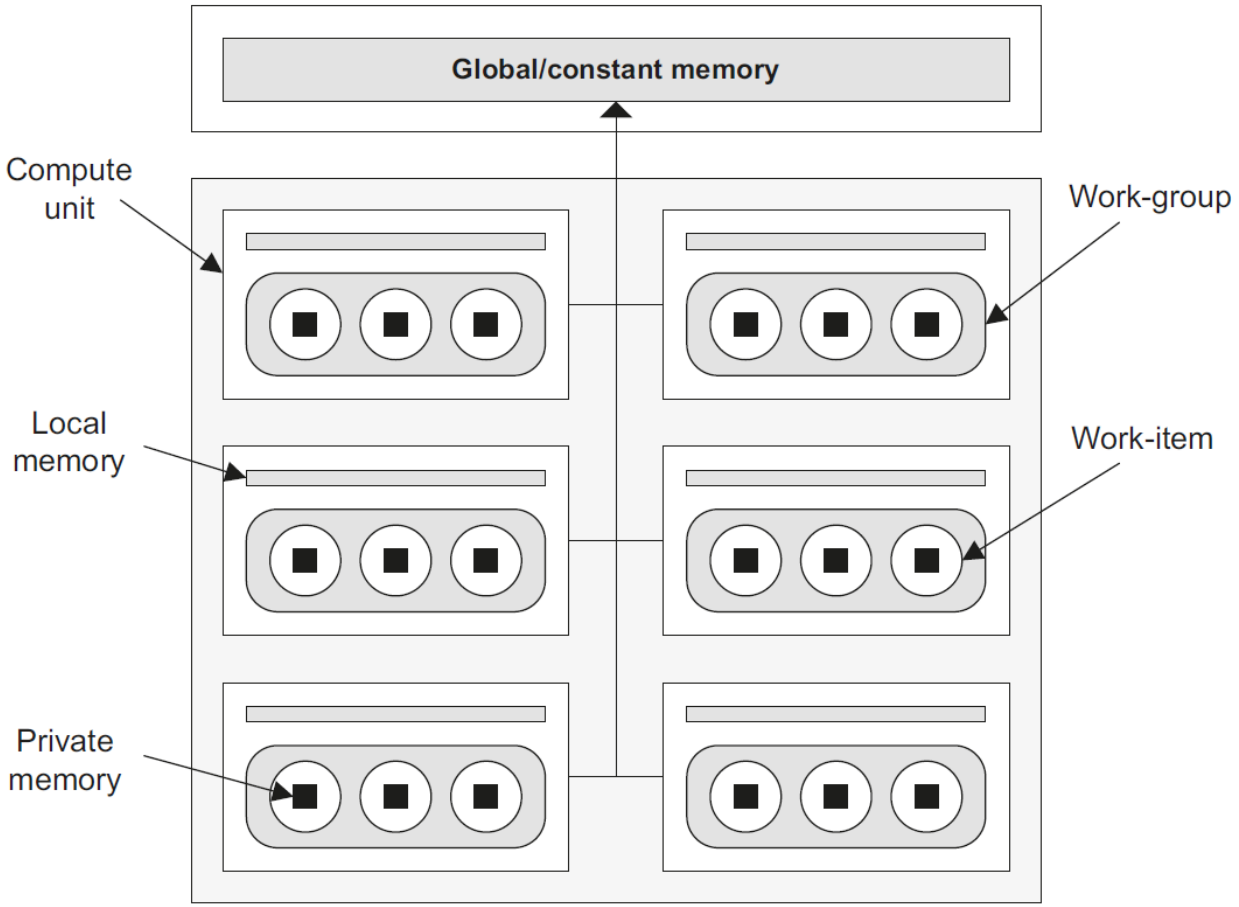
Workgroups und Compute Units



3

[3] Scarpino Matthew: OpenCL In Action, Manning Publications Co., 2012, S. 66

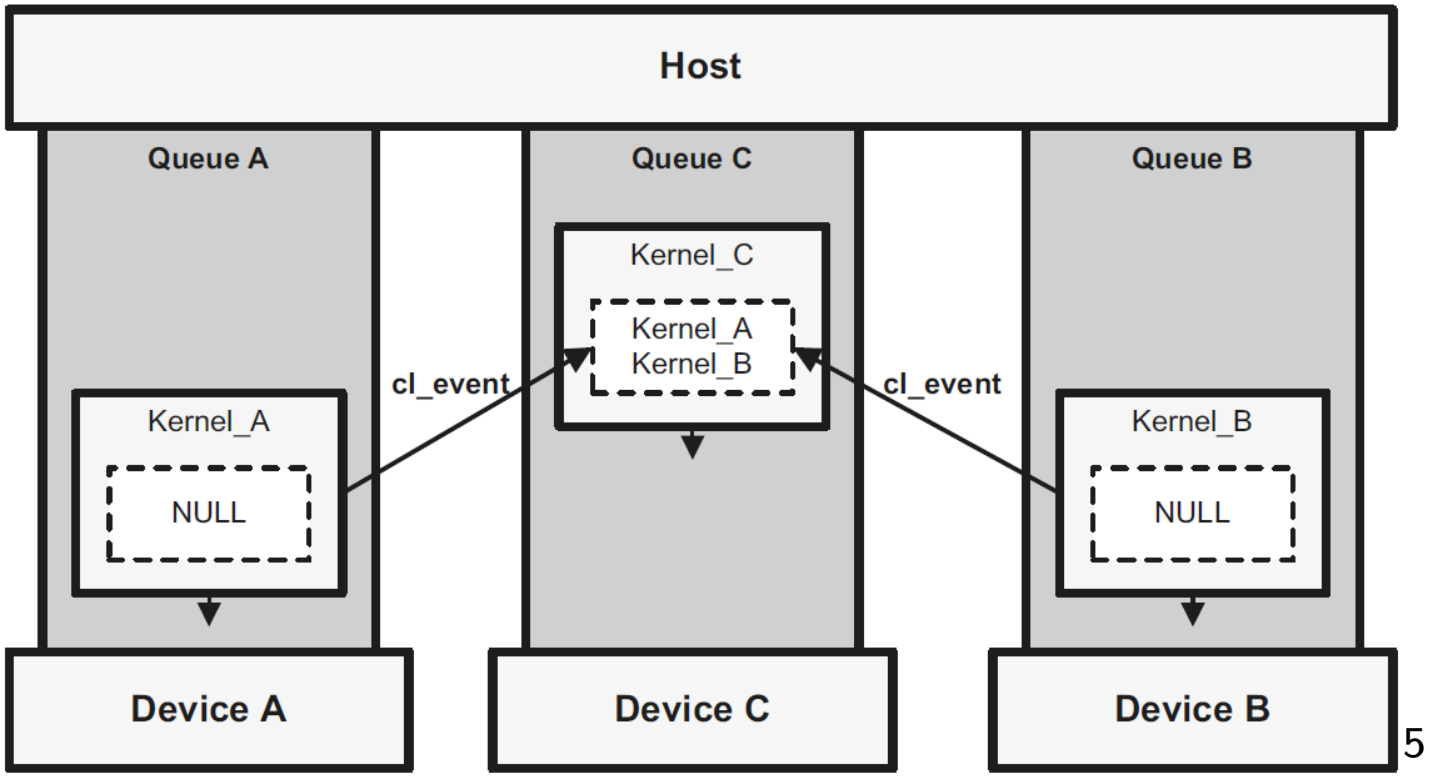
Device Model



[4]

[4] Scarpino Matthew: OpenCL In Action, Manning Publications Co., 2012, S. 87

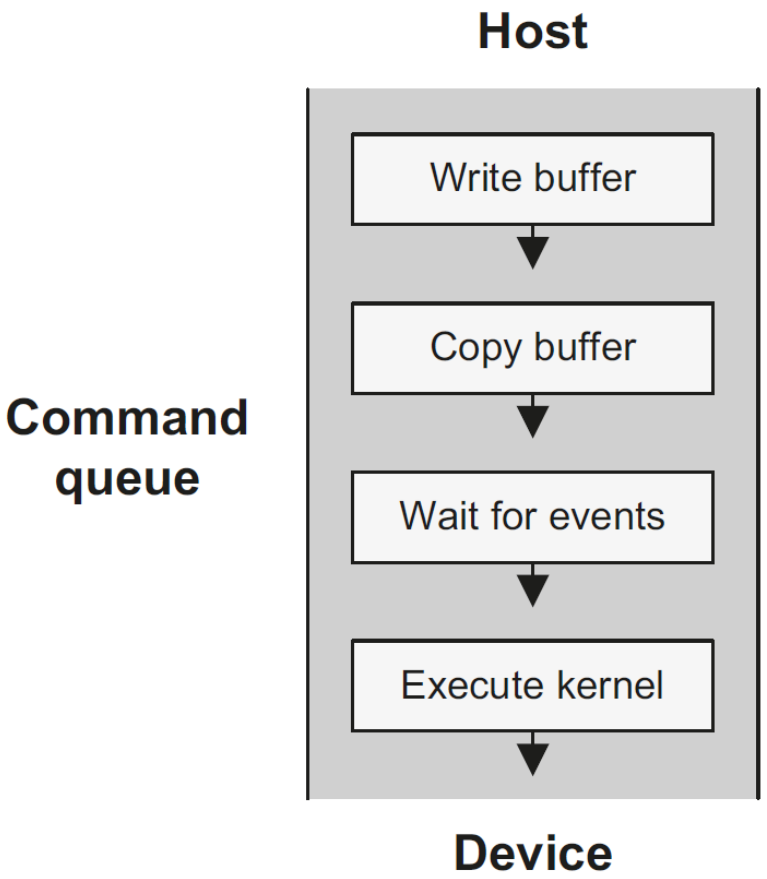
Wait Lists



[5]

[5] Scarpino Matthew: OpenCL In Action, Manning Publications Co., 2012, S. 146

Wait Command



[6]

[6] Scarpino Matthew: OpenCL In Action, Manning Publications Co., 2012, S. 150

Unterstütze Plattformen

CUDA:

- NVIDIA-GPUs

OpenCL:

- alle Recheneinheiten die OpenCL unterstützt
- CPUs, GPUs

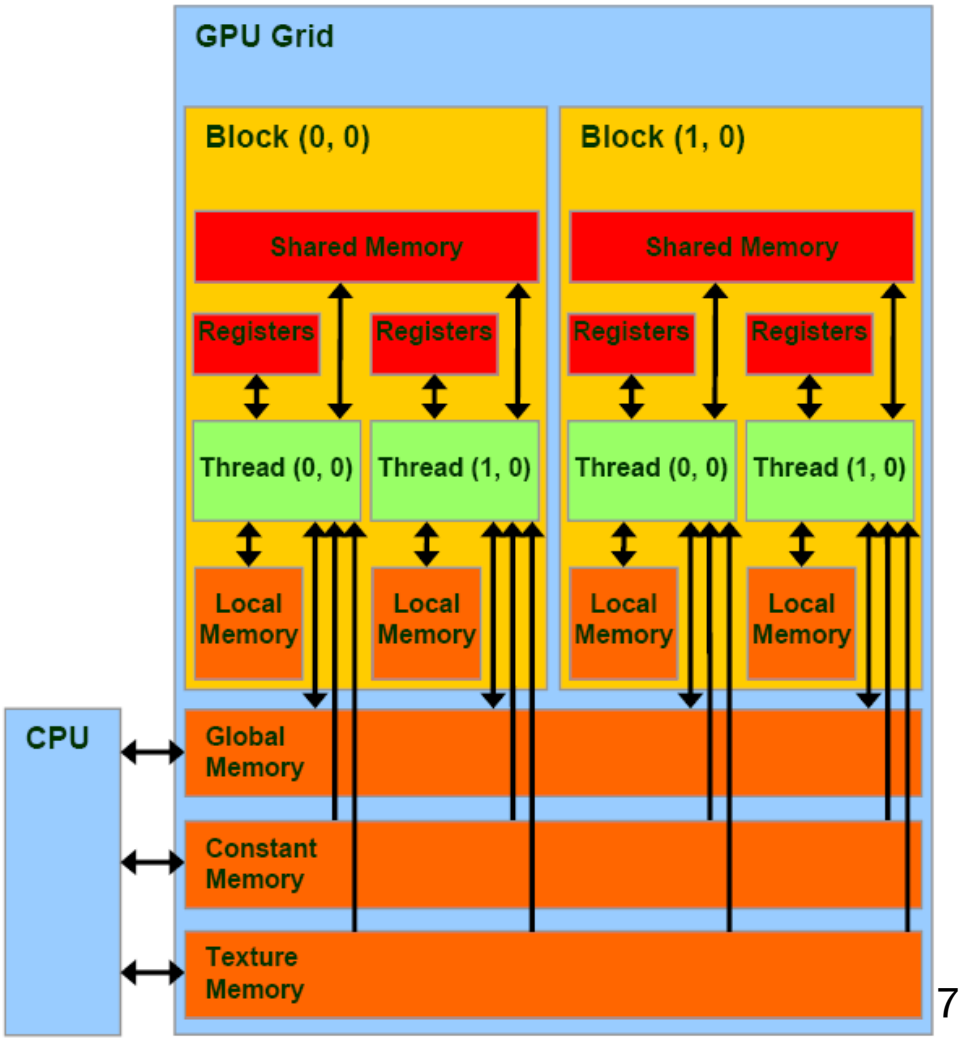
Performance

CUDA:

- Hardware und Technologie vom gleichen Hersteller
- gute Implementation
- gute Leistung

OpenCL:

- von Plattform abhängig
- Faktoren: Leistung, Implementation



[7]

[7] CUDA Modell, Quelle: [2]

API/Modell

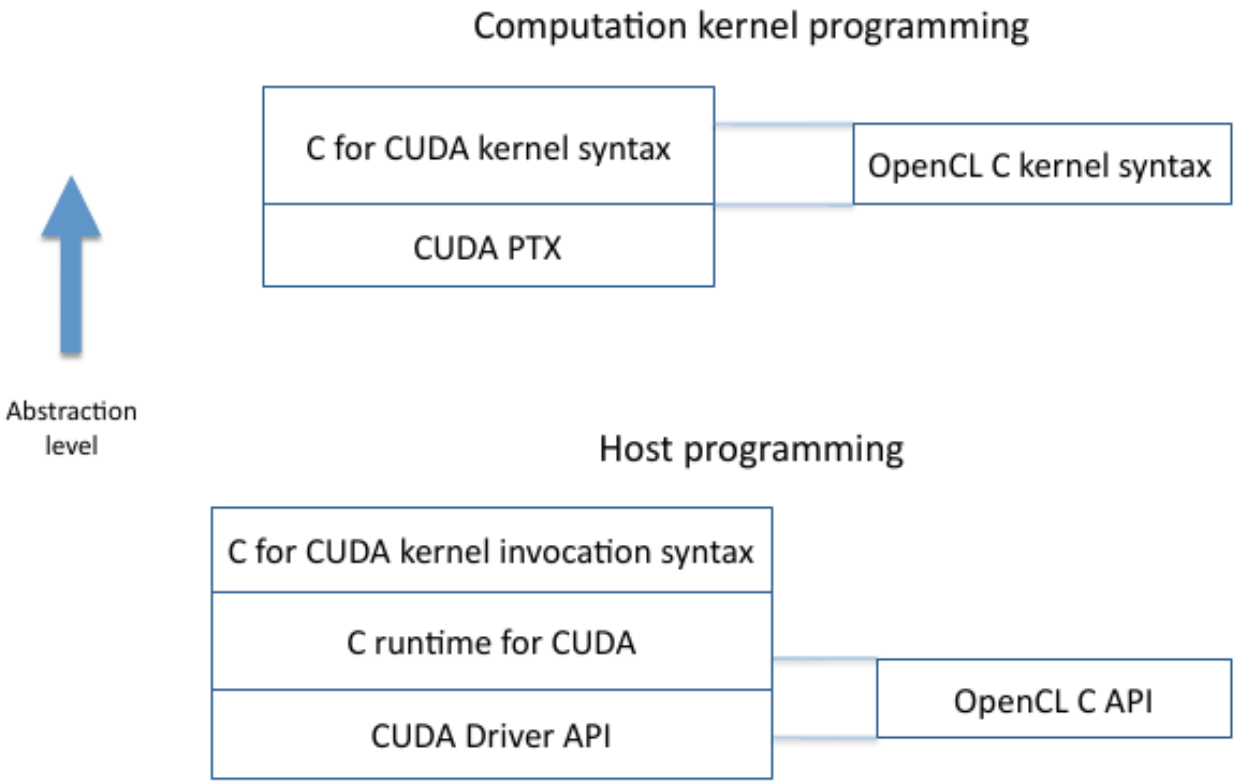
- Modelle ähneln sich
- Begriffsunterschiede
- weitere Unterschiede in der Syntax

Erklärung	CUDA	OpenCL	Eigenschaft
Gerät, Grafikkarte	CUDA GPU	Device	
Funtkionstypen	__device__	__kernel	keine Differenzierung bei OpenCL
	__global__	__kernel	keine Differenzierung bei OpenCL
	__host__	__kernel	keine Differenzierung bei OpenCL
Variablentypen	__device__	__global	im globalen Speicher
	__constant__	__local	im konstanten Speicher
	__shared__	__constant	im gemeinsamen Speicher
Ausführung	Thread	Work-Item	kleinste Zerlegegung
	Block		
	Grid	Work-Group	Arbeitsgruppe

8

[8] Quelle: [2]

API abstraction levels compared



9

[9] Quelle: [3]

Entwicklungsaufwand

OpenCL:

- API mit geringer Abstraktion
- verschiedene Debugging-Möglichkeiten je Plattform
- guter Debugger(cross-plattform): gDEBugger
- verschieden Geräte: unterschiedliche Implementation

CUDA:

- geringe und hohe Abstraktion
- viele Bibliotheken
- guter Debugger durch CUDA-SDK

Fazit

→ für High-Performance-Cluster mit gleicher Hardware und speziell angefertigter Software
⇒ CUDA zu bevorzugen

→ im Consumerbereich bei Verwendung von verschiedener Hardware
⇒ OpenCL bevorzugt

Allgemeines zu Bildverarbeitung mit OpenCL

- spezielle Datentypen (image2d_t, image3d_t) und Funktionen im OpenCL C
- Image2D und Image3D Klassen in der C++ API
- Verschiedene Datentypen für Channels
- Verschiedene Reihenfolge der Channels
- maximale Größe eines Images je Plattform festgelegt

Kantenerkennung in Bildern

Ablauf:

1. Graubild erstellen und entrauschen
2. Anwendung des Sobeloperators je Pixel in X-Richtung und Y-Richtung mit den Nachbarwerten
3. Kombination beider Ergebnisse ergibt Kantenwert des Pixels

Kantenerkennung in Bildern mit OpenCL

Verwendeter einfacher Algorithmus:

1. Differenzen der gegenüberliegenden Pixel
2. der höchste Wert wird der Kantenwert des Pixels

Quellen

1. Scarpino Matthew: OpenCL In Action,
   Manning Publications Co., 2012
2. ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart_fi/DIP-3178/DIP-3178.pdf
3. https://wiki.aalto.fi/download/attachments/40025977/Cuda+and+OpenCL+API+comparison_presented.pdf