

Bildverarbeitung mit OpenCL

Johannes Hackel und Falco Prescher

23. Mai 2013

Bildverarbeitung mit OpenCL

1 OpenCL

- Allgemeines zu OpenCL
- Kernelverteilung unter Devices und Command Queue
- Workgroups und Compute Units und Device Model
- Events in OpenCL

2 Vergleich von OpenCL mit CUDA

- Unterst[Pleaseinsertintopreamble]tze Plattformen
- Performance
- API/Modell
- Entwicklungsaufwand
- Fazit

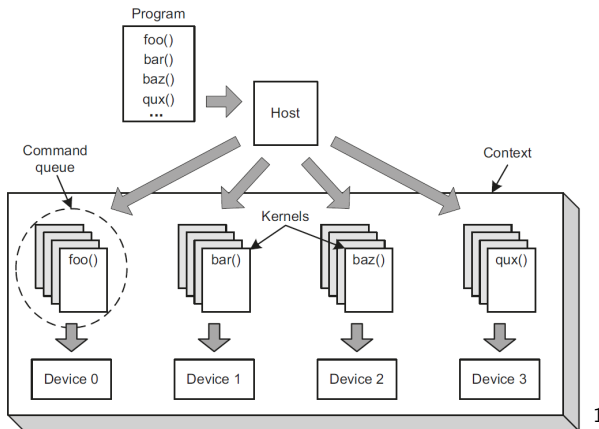
3 Bildverarbeitung

- Allgemeines zu Bildverarbeitung mit OpenCL
- Kantenerkennung in Bildern
- Kantenerkennung in Bildern mit OpenCL

Allgemeines zu OpenCL

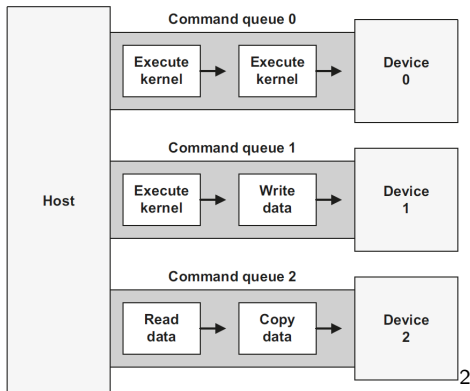
- Test

Kernelverteilung unter Devices



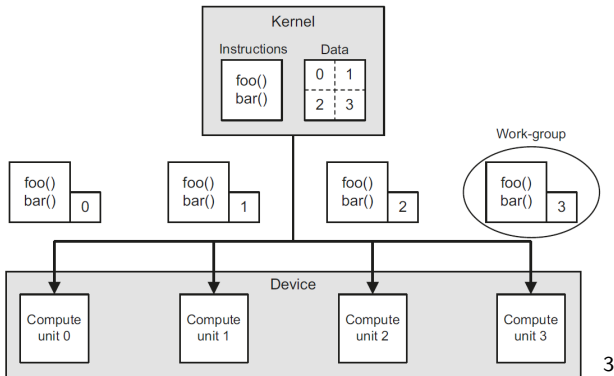
¹ Scarpino Matthew: OpenCL In Action,
Manning Publications Co., 2012, S. 8

Command Queue



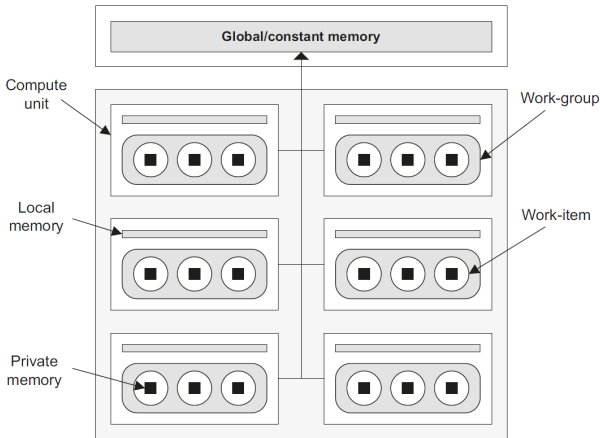
²Scarpino Matthew: OpenCL In Action,
Manning Publications Co., 2012, S. 39

Workgroups und Compute Units



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

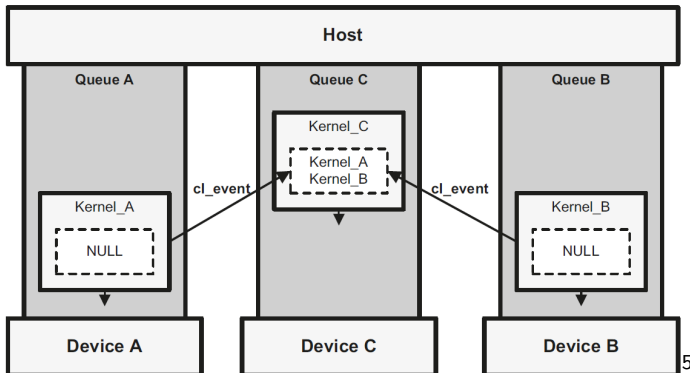
Device Model



4

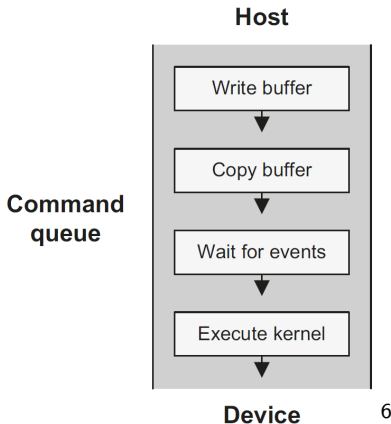
⁴ Scarpino Matthew: OpenCL In Action,
Manning Publications Co., 2012, S. 87

Wait Lists



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

Wait Command



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

Unterstützte 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

API/Modell

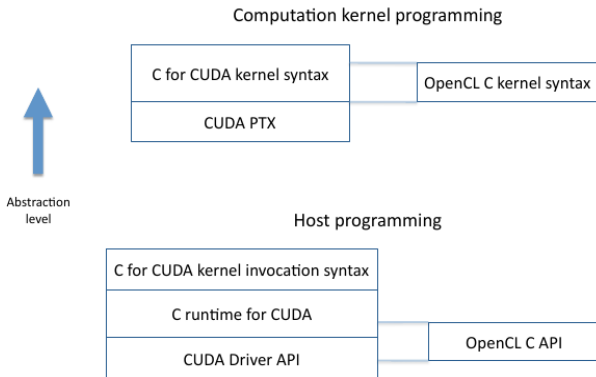
- Modelle ähneln sich
- Begriffsunterschiede
- weitere Unterschiede in der Syntax

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

7

⁷ Quelle: [2]

API abstraction levels compared



8

⁸ Quelle: [3]

Entwicklungsaufwand

OpenCL:

- API mit geringer Abstraktion
- verschiedene Debugging-Möglichkeiten je Plattform
- guter Debugger(cross-plattform): gDEBbugger
- 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`