OpenCL

Programmiersprachen im Multicore-Zeitalter

Tim Wiersdörfer

Inhaltsverzeichnis

Was ist OpenCL

- **Open C**omputing **L**anguage
- Offener Standard zur parallelen & plattform-unabhängigen Programmierung
- Einsatz auf unterschiedlichen Plattformen:
  - CPU
  - GPU
  - DSP
  - Cell Prozessoren
- Unterteilung in drei abstrakte Modelle:
  - Das Plattform-Modell
  - Das Ausführungs-Modell
  - Das Speicher-Modell

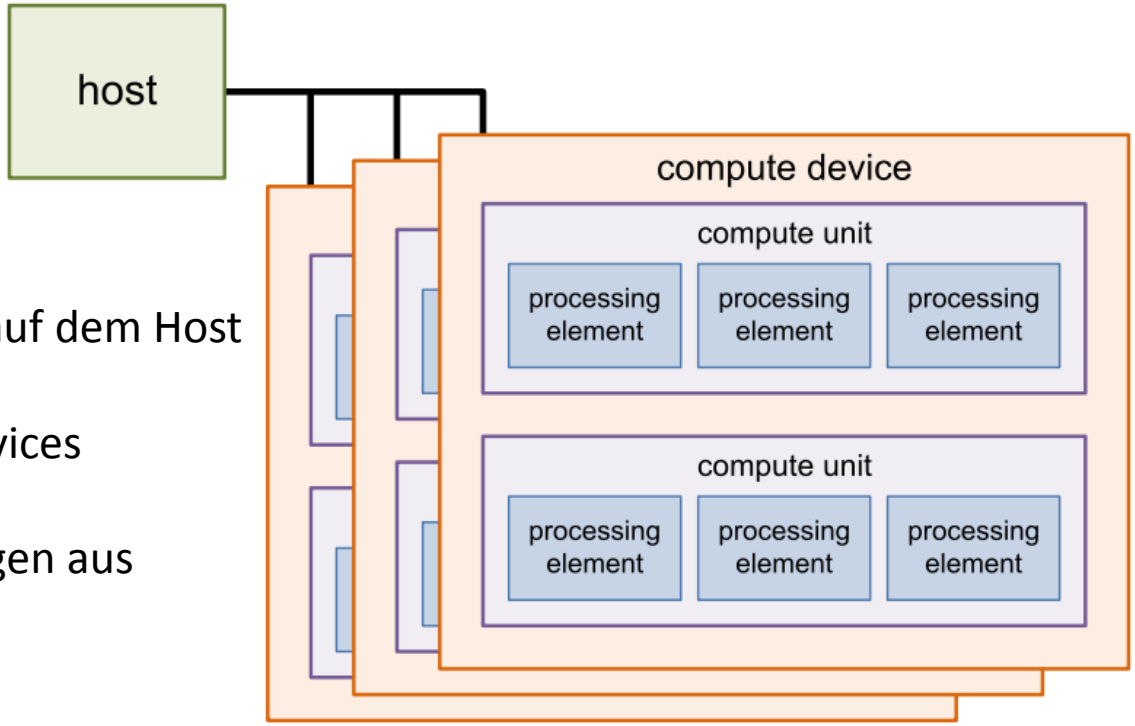
Entwicklung von OpenCL

- Ursprünglich von der Firma Apple entwickelt
  - 28. August 2009 erster im Betriebssystem Mac OS X 10.6 (Snow Leopard)

- Khronos Group
  - Industrie-Konsortium
  - Erstellt und Verwaltet offene Standards im Multimedia-Bereich
  - Dezember 2008 Veröffentlichung der OpenCL 1.0 Spezifikation
  - In Zusammenarbeit mit NVIDIA, Intel, AMD & IBM

Plattform-Modell

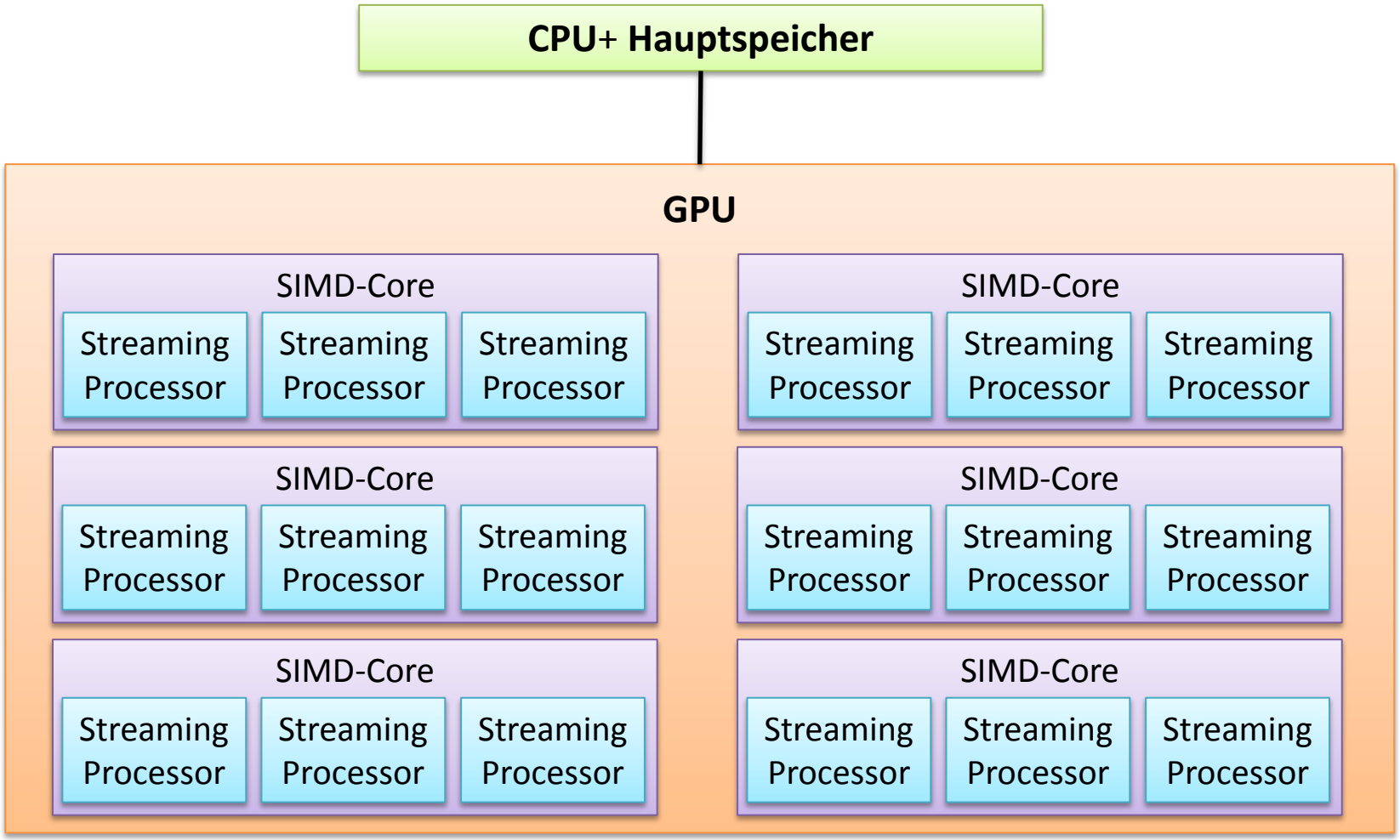
- Abstrakten Aufbau eines OpenCL-Systems
  - Wird von unterschiedlicher Hardware verschieden umgesetzt!
- Plattform besteht aus einem Host, verbunden mit mehreren OpenCL Devices



- OpenCL Anwendung läuft auf dem Host
- Host sendet Befehle an Devices
- Devices führen Berechnungen aus

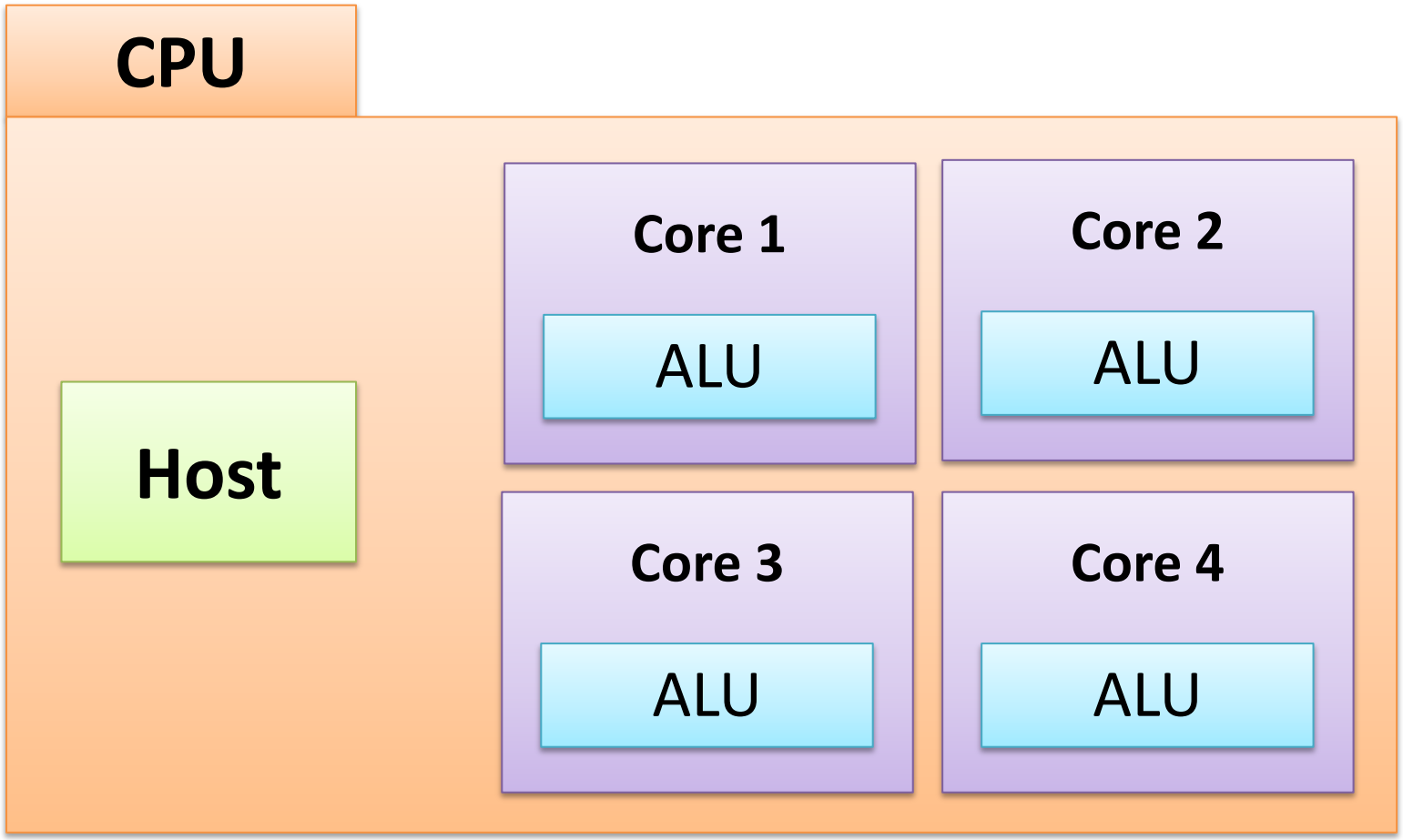
Plattform-Modell

System besteht aus einer CPU und einer GPU



Plattform-Modell

System besteht aus einer Multicore-CPU



Ausführungs-Modell

- Es wird zwischen zwei Teilen unterschieden:
  - Host-Programm
  - Kernel
- Host-Programm verwaltet und steuert die Ausführung der Kernel(-instanzen)
- Kernel ist eine Funktion, welche von einem OpenCL Device ausgeführt wird
  - Parallel durch mehrere Instanzen (Work-Items)

Ausführungs-Modell
Kernel

- Ist eine OpenCL C Funktion
- Wird zur Laufzeit für die entsprechende Hardware kompiliert
  - Plattform-unabhängig
- Wird von dem Host zur Laufzeit an ein / mehrere OpenCL Geräte verteilt
  → mehrere Instanzen
- Instanz = Work-Item
  - Jedes Work-Item wird immer sequentiell ausgeführt

Ausführungs-Modell
Host-Programm

- Läuft auf dem Host
  - Verteilt die Kernel an die angeschlossenen OpenCL Devices
  - Dabei, Auswahl des am besten passenden Device:
    - Maximale Anzahl von Recheneinheiten
    - Höchste Taktfrequenz
    - Größter Speicher
    - Usw.
- Verwaltet & steuert alle Work-Items:
  - Definiert Anzahl der Work-Items
  - Work-Items werden in Work-Groups eingeteilt

Ausführungs-Modell
Warteschlange

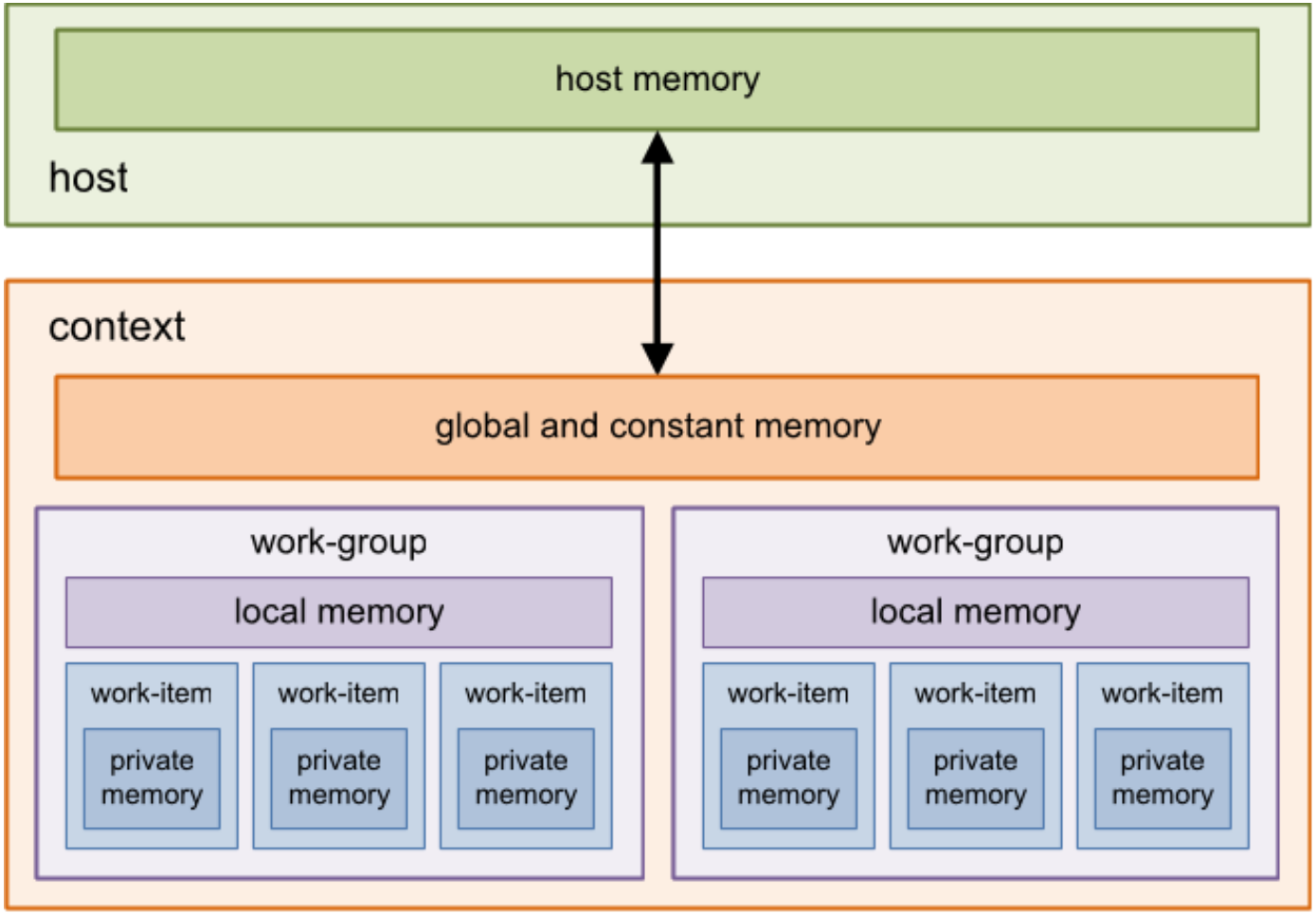
- Kernelinstanzen werden nicht direkt auf dem OpenCL Device gestartet
- Sondern alle Kommandos werden in eine Warteschlange eingereiht
  - Kernel-Ausführungen
  - Speicher-Operationen
  - Synchronisations-Operationen
- OpenCL Device arbeitet Warteschlange ab
  - Nach jedem Befehl folgt sofort der nächste
  - Keine Wartezeit auf Host
  - Effizienter Programmfluss!

Speicher-Modell

- OpenCL teil Speicher in verschiedene logische Bereiche ein:
  - OpenCL Device hat einen vom Host logisch getrennten Speicher
- Daten welche im Kernel benötigt werden, müssen daher zu dem Device kopiert werden
- OpenCL unterscheidet innerhalb eines OpenCL Devices zwischen vier Speicherbereichen:
  - Globaler Speicher
  - Konstanter Speicher
  - Lokaler Speicher
  - Privater Speicher

Speicher-Modell

Speicherbereiche im Device



OpenCL Anwendung

- Host & Device haben (logisch) getrennten Speicher
  → Daten müssen explizit ausgetauscht werden!

- Aufbau einer OpenCL Anwendung:

```
int main (void) {
    1.  Initialisierung von OpenCL

    2.  Kernel-Quellcode kompilieren

    3.  Reservieren des globalen und konstanten Speichers auf dem
        Device und Daten zum Device kopieren.

    4.  Kernelinstanzen ausführen

    5.  Daten zurück auf den Host kopieren
}
```

- OpenCL Anwendungen werden in OpenCL C verfasst

OpenCL C

- Basiert auf der Sprache C
- Da OpenCL Device eingeschränkt sind ist es auch OpenCL C
- Wesentlichsten Einschränkungen gegenüber C:
  - Standardfunktionen wie z.B. printf() oder malloc() fehlen
  - Array variabler Länge sind nicht möglich
  - Rekursionen sind nicht erlaubt
  - Es gibt keine Funktionszeiger

OpenCL C

- Neben Einschränkungen gibt es auch zusätzliche Programmkonstrukte
- Vor einer Deklaration werden Qualifier / Schlüsselworte angegeben
- Funktionen können mit dem Qualifier „__kernel“ deklariert werden:
  - Wird auf einem Device ausgeführt
  - Kann vom Host aufgerufen werden
  - Muss den Rückgabetyp void haben
- Variablen können mit Qualifier markiert werden:
  - __global
  - __local
  - __constant
  - __private (default)
  - Diese entsprechen den Speicherbereichen

OpenCL C
Code-Beispiel

Standard C

```
1) void dot_product ( const float *a,
2)                    const float *b,
3)                    const int n,
4)                    float *c)
5) {
6)     for (int i = 0; i < n; ++i)
7)     c[i] = a[i] * b[i];
8) }
```

OpenCL C

```
1) __kernel void dot_product ( __global const float4 *a,
2)                             __global const float4 *b,
3)                             __global float *c)
4) {
5)     int gid = get_global_id(0);
6)     c[gid] = a[gid] * b[gid];
7) }
```

Zusammenfassung

- OpenCL ist ein Plattform-übergreifendes Programmiermodell
- Unterscheidung zwischen Host & Device
- Kernel sind Funktionen welche auf Devices ausgeführt werden
- (logische) Trennung der Speicher
  → Daten müssen explizit ausgetauscht werden
- Host verwaltet & steuert die Kernelinstanzen
  - Eine Instanz nennt man Work-Item
- Es existieren Unterschiedliche konkrete Implementierungen von OpenCL
  - GPUs & Multicore CPUs