JUST DO IT
HÖHERE ABTEILUNG FÜR
INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

GPGPU

Systemtechnik Labor
5BHITT 2015/16, Gruppe C

Patrick Malik & Thomas Taschner

Version 1.0
Note:
Begonnen am 15. Januar 2016
Betreuer: Michael Borko
Beendet am 22. Januar 2016

Inhaltsverzeichnis

1 Einführung

1.1 Ziele

Die Aufgabe beinhaltet eine Recherche über grundsätzliche Einsatzmöglichkeiten für GPGPU. Dabei soll die Sinnhaftigkeit der Technologie unterstrichen werden. Die Fragestellungen sollen entsprechend mit Argumenten untermauert werden.

Im zweiten Teil der Arbeit soll der praktische Einsatz von OpenCL trainiert werden. Diese können anhand von bestehenden Codeexamples durchgeführt werden. Dabei wird auf eine sprechende Gegenüberstellung (Benchmark) Wert gelegt.

Die Aufgabenstellung soll in einer Zweiergruppe bearbeitet werden.

1.2 Aufgabenstellung

Informieren Sie sich über die Möglichkeiten der Nutzung von GPUs in normalen Desktop-Anwendungen. Zeigen Sie dazu im Gegensatz den Vorteil der GPUs in rechenintensiven Implementierungen auf [1Pkt]. Gibt es Entwicklungsumgebungen und in welchen Programmiersprachen kann man diese nutzen [1Pkt]?

Können bestehende Programme (C/C++ und Java) auf GPUs genutzt werden und was sind dabei die Grundvoraussetzungen dafür [1Pkt]? Gibt es transcompiler und wie kommen diese zum Einsatz [1Pkt]?

Präsentieren Sie an einem praktischen Beispiel den Nutzen dieser Technologie. Wählen Sie zwei rechenintensive Algorithmen (z.B. Faktorisierung) und zeigen Sie in einem aussagekräftigen Benchmark welche Vorteile der Einsatz der vorhandenen GPU Hardware gegenüber dem Ausführen auf einer CPU bringt (OpenCL). Punkteschlüssel:

Auswahl und Argumentation der zwei rechenintensiven Algorithmen (Speicher, Zugriff, Rechenoperationen) [0..4Pkt]

Sinnvolle Gegenüberstellung von CPU und GPU im Benchmark [0..2Pkt]

Anzahl der Durchläufe [0..2Pkt]

Informationen bei Benchmark [0..2Pkt]

Beschreibung und Bereitstellung des Beispiels (Ausführbarkeit) [0..2Pkt]

1.3 Links

https://github.com/ReneHollander/opencl-example

2 Ergebnisse

2.1 Nutzung von GPUs in normalen Desktop-Anwendungen

Im privaten Bereich gibt es praktisch keine Anwendung für GPGPU. Ausnahmen sind Programme für die Videobearbeitung. Typische Alltagssoftware lässt sich kaum parallelisieren. Im Bereich der Wissenschaft und Technik sieht es dann schon wieder anders aus. Hier machen mehrere GPGPU-taugliche Grafikkarten einen normalen PC zu einem Supercomputer. Voraussetzung dafür ist ein hochleistungsfähiger Grafikprozessor, der bis zu 800 Kerne hat und parallel arbeitet. Damit ein Grafikprozessor überhaupt sinnvoll genutzt werden kann, muss die Anwendung eine sehr hohe Zahl von parallel ausführbaren Berechnungen enthalten. Wenn man die Rechenwerke eines Grafikprozessors kontinuierlich auslasten will, dann sollte man mindestens eine Millionen gleichartiger Berechnungen liefern können, sonst hat man gegenüber einer normalen CPU keine Vorteile. Die GPU-Rechenwerke müssen mit einem kontinuierlichen Datenstrom versorgt werden. Aus diesem Grund bezeichnet man sie auch als Stream-Prozessoren. [1] Zu Ergänzen ist die erste Aussage auch noch mit Mathematikprogrammen wie Matlab oder ähnlichem. [2]

2.1.1 Anwendungen

- Physikalisch Simulationen (Strömung, Gravitation, Temperatur, Crash-Tests)
- Komplexe Klimamodelle (Wettervorhersage)
- Datenanalysen und Finanzmathematik
- Verarbeitung von akustischen und elektrischen Signalen
- CT- und Ultraschall-Bildrekonstruktion
- Modellieren von Molekühlstrukturen
- Kryptografie
- Neuronale Netze
- Videotranskodierung

2.2 Vorteile der GPU bei rechenintensiven Anwendungen

GPUs haben den Vorteil, dass sie auf spezielle Arten von Prozessen optimiert sind. CPUs sind hingegen darauf ausgelegt beliebige Aufgaben zu verarbeiten. Um rechenintensive Anwendungen auch tatsächlich auf der GPU laufen lassen zu können, müssen/sollten diese gut parallelisierbar sein. Im Vergleich zur CPU hat man wesentlich mehr Kerne auf die parallelisiert werden kann.

2.3 Entwicklungsumgebungen und Programmiersprachen

Von NVidia wird unter Anderem eine Visual Studio Abwandlung namens NSight zur Verfügung gestellt, welche es einem ermöglicht in CUDA C/C++, OpenCL, DirectCompute, Direct3D, and OpenGL zu programmieren.
OpenCL Studio eröffnet einem die Möglichkeit mit OpenCL und OpenGL zu arbeiten.
NVidia CUDA Toolkit bietet einem die Möglichkeit CUDA Applikationen in C und C++ zu Programmieren. Applikationen für OpenCL werden zum Großteil in C oder C++ programmiert. OpenCL C/C++ bieten die Option direkt für OpenCL zu programmieren.

2.4 Transkompilieren

Um bestehende Applikationen auf GPUs sinnvoll zu nutzen, müssen diese prinzipiell parallelisierbar sein. Das CUDA Toolkit stellt einem einen Transcompiler für C und C++ zur Verfügung, es lagert die rechenintensiven Teile auf die GPU aus. Rootbeer ist ein GPUCompiler, welcher Javacode auf der GPU ausführen lässt, ohne dass der Code extra angepasst werden muss.

2.5 Benchmarks

2.5.1 Specs

Laptopmodell: Lenovo Ideapad y510p

- Mainboard:
  Name: Intel HM86 Chipset
  Modell: LENOVO VIQY0Y1
- CPU:
  Modell: Intel Core i7 4700MQ @ 2.40GHz
  Kerne: 4 (virtuell 8)
  Chipset: Intel Haswell
  Southbridge: HM86
- RAM:
  16GB DDR3
  Dual Channel
  Maxfrequenz: PC3-12800 (800MHz)
- GPU (2x):
  Modell: NVidia GeForce GT 755M

Architektur: GK107 (Kepler)

Busbreite: 128b

VRAM: 2GB GDDR5(Hynix)

Core (Taktfrequenz): 980MHz (Boost bis 1020MHz)

Memory (Taktfrequenz): 1350MHz

- OS

  Name: Windows 10

-

2.5.2 Setup

2.5.3 Ergebnise

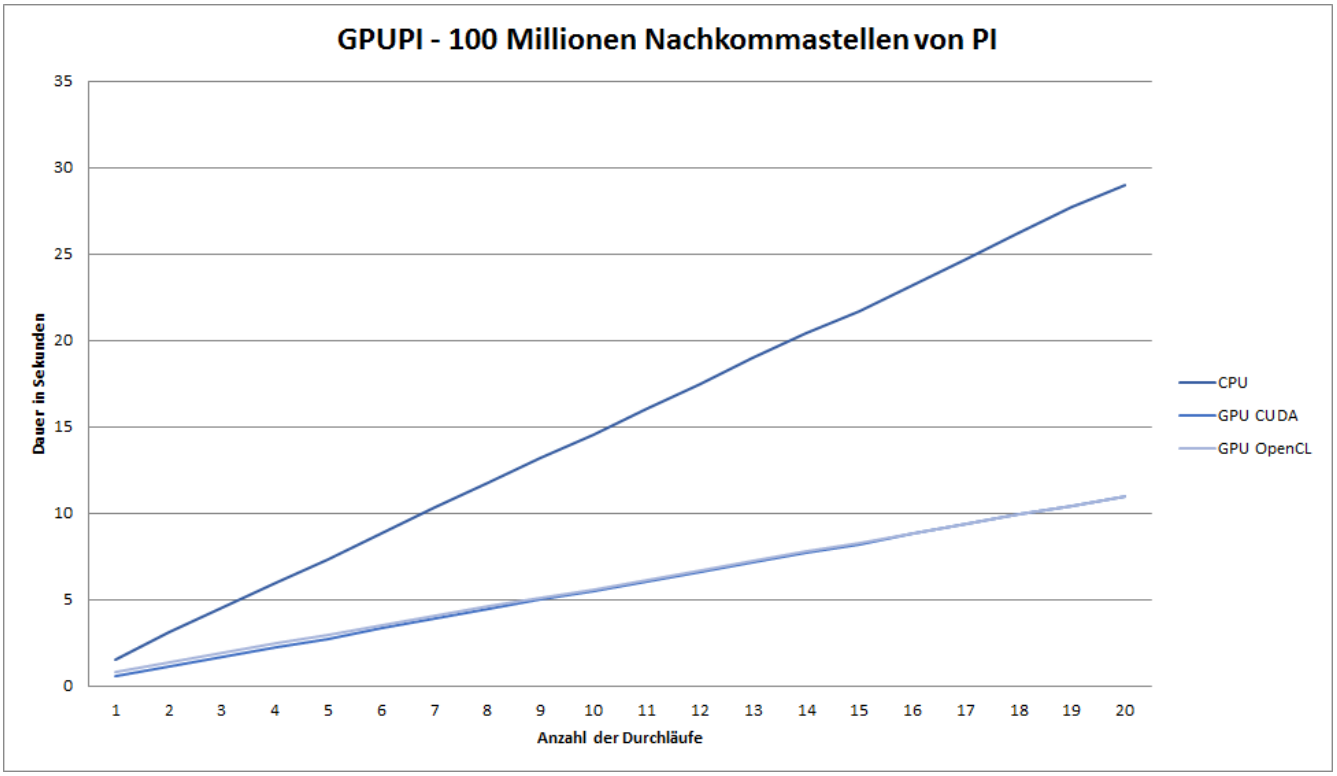

Abbildung 1: GPUPI 100 Millionen Nachkommastellen

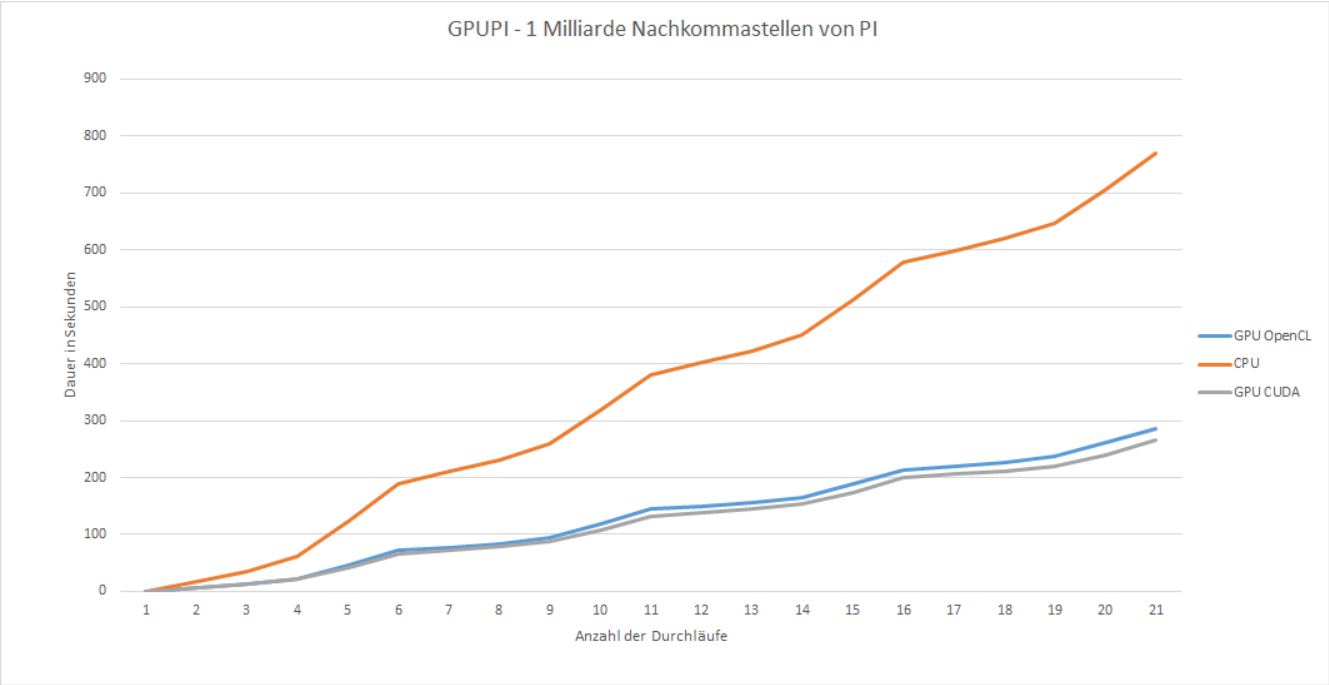

Abbildung 2: GPUPI 1 Milliarde Nachkommastellen

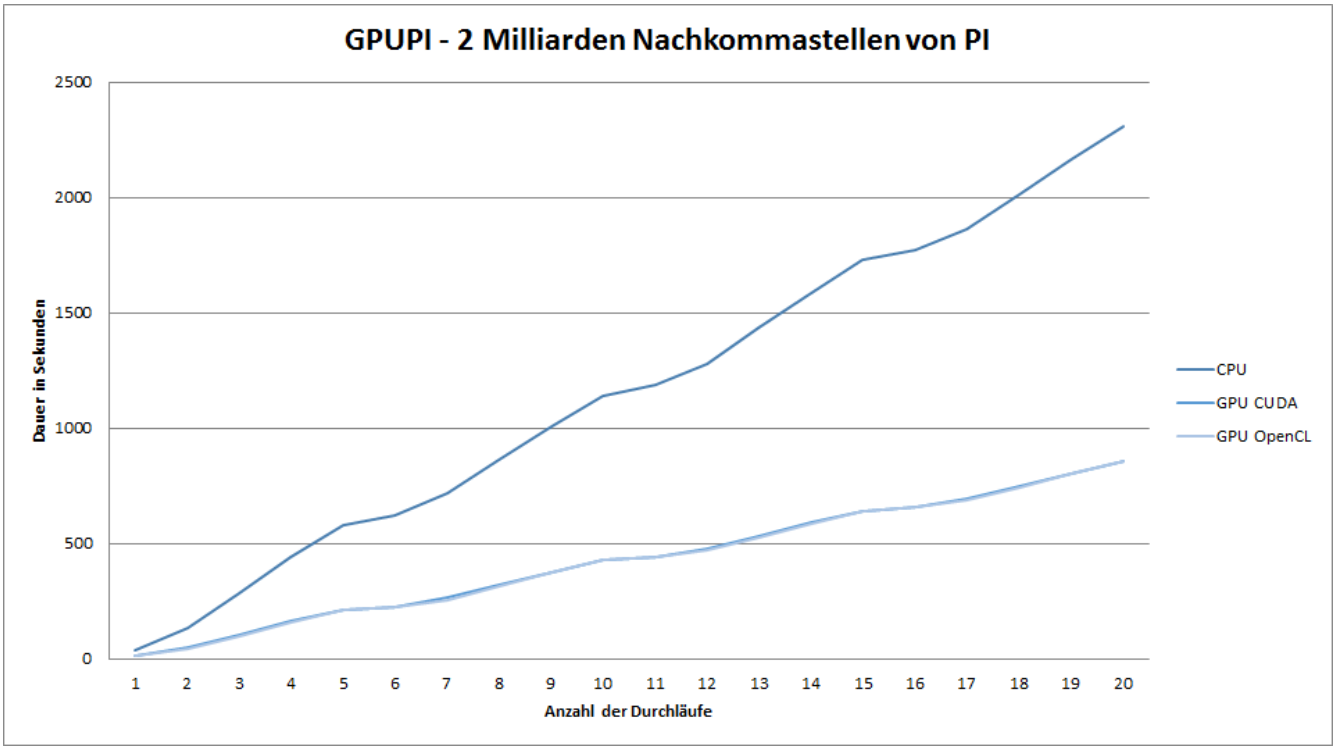

Abbildung 3: GPUPI 2 Milliarden Nachkommastellen

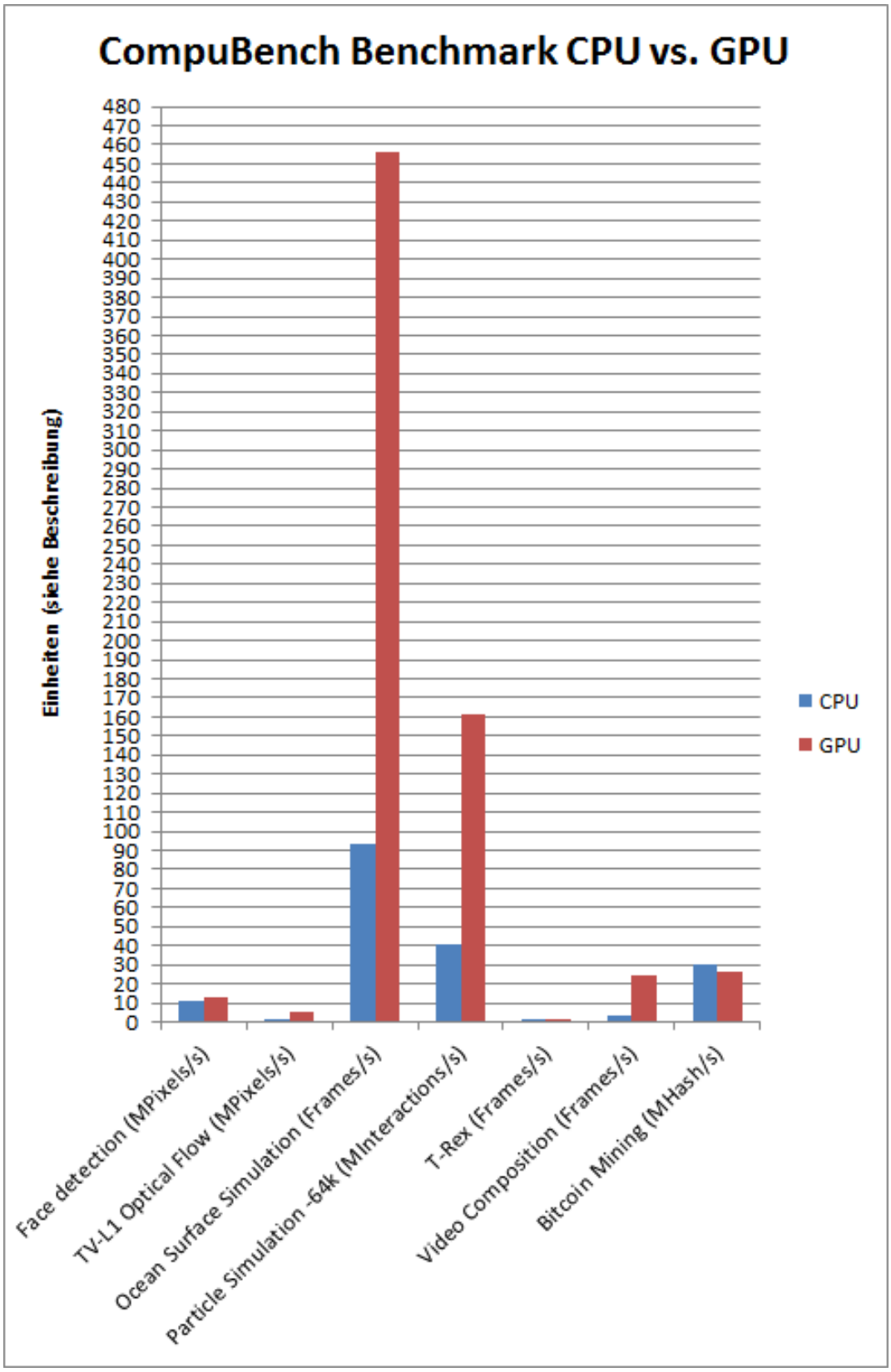

Abbildung 4: Ergebnisse des CompuBench Benchmarks

Literatur

[1]

[2]

Tabellenverzeichnis

Listings

Abbildungsverzeichnis

