Prof. Dr.-Ing. Thomas Pucklitzsch
ist am 03.07.1976 in Halle/Saale geboren. Er ist verheiratet und hat zwei Kinder. Von 1996-2003 studierte Thomas Pucklitzsch an der Technischen Universität Dresden das Fach Informatik. Anschließend arbeitete er als CIO in einem Krankenhaus und promovierte an der TU Dresden am Institut für Rechnernetze. Seit dem 01.04.2016 ist Thomas Pucklitzsch Dozent an der Staatlichen Studienakademie Glauchau im Studiengang Wirtschaftsinformatik.

KONTAKT: Prof. Dr. Thomas Pucklitzsch | Berufsakademie Sachsen
Staatliche Studienakademie Glauchau
pucklitzsch@ba-glauchau.de

Berechnung neuronaler Netze mit CUDA

Prof. Dr. Thomas Pucklitzsch

AUSGANGSPUNKT:
Der Simulation neuronaler Netze sowie für viele andere Problemstellungen mit hoher kombinatorischer Komplexität setzt die Leistungsfähigkeit konventioneller PC-Architekturen enge Grenzen. Somit ist die parallele Verarbeitung großer Datenmengen von entscheidender Bedeutung für die Performance solcher Anwendungen. Um Tasks mit Hilfe von Rechentechnik zu parallelisieren gibt es verschiedene Ansätze.

1. Grid Computing

Die Idee hinter dem Grid-Computing besteht darin, die Ressourcen die zum Lösen einer Aufgabe nötig sind (Speicher, Rechenkapazität, Festplattenkapazität) über das Internet verfügbar zu machen. Diese Technologie ist interessant, falls Tasks verteilt werden, deren zur Lösung benötigte Rechenzeit in einem guten Verhältnis zur relativ langsamen Datenübertragung über das Internet stehen. D.h. falls beispielsweise Daten übertragen werden, an denen ein Knoten mehrere Minuten oder Stunden rechnet und anschließend seine Ergebnisse zurücksendet, ist Grid-Computing ein vielversprechender Ansatz. Ein Beispiel hierfür ist das SEDI@Home Projekt, bei welchem große Mengen an Radiosignalen aus dem All durchsucht werden.

2. Height Performance Computing

Beim Height Performance Computing werden universelle CPUs mit Hilfe besonders schneller Netzwerktechnologie wie z.B. Infiniband miteinander vernetzt. So können die Zwischenergebnisse der einzelnen Rechenknoten sehr schnell übertragen werden und als Eingabe für weitere Berechnungen dienen. Diese Technologien sind sehr kostenintensiv.

3. Benutzung von Grafikkarten

Eine andere Möglichkeit Tasks zu parallelisieren ist die Verwendung von Grafikkarten, die dafür gebaut sind, dreidimensionale Objekte möglichst schnell zu rendern. Dazu werden diese Grafikkarten mit vielen kleinen speziellen Stream-Prozessoren (auch Shader units genannt) ausgestattet. Für bestimmte Problemstellungen kann man diese Shader units benutzen, um auch andere Aufgaben zu lösen wie z.B. das Bitcoin-Minen, bei dem ein kompliziertes Rätsel gelöst werden muss, um einen neuen Bitcoin zu erstellen.

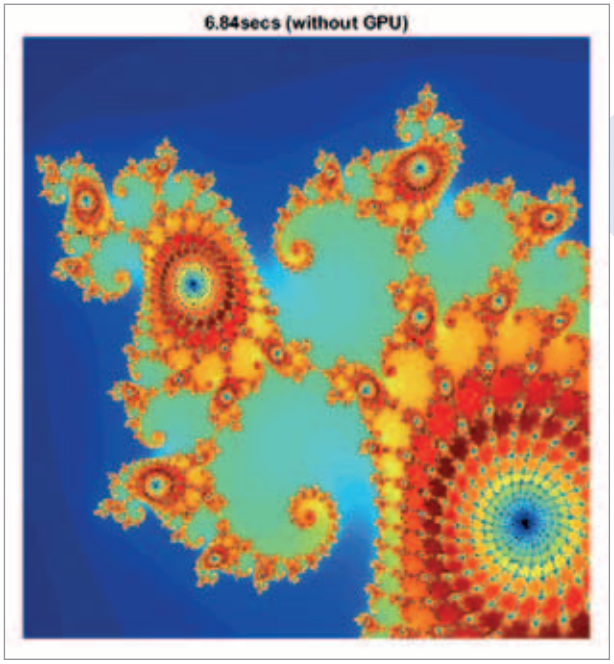


Abbildung entnommen www.mathworks.com
Abbildung 1

ZIEL:

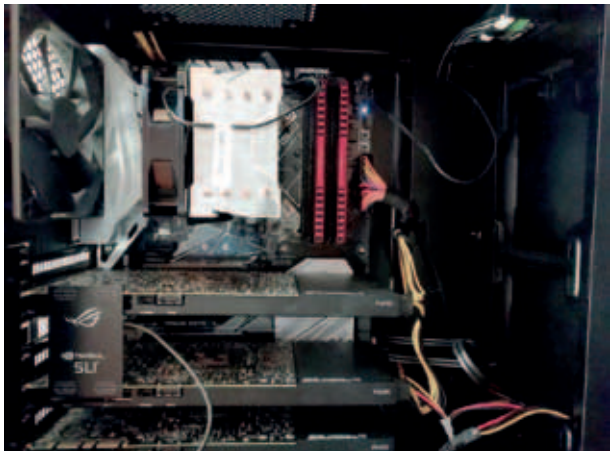
Mit dem Ziel, leistungsfähige Neuronale Netze zu untersuchen wurde nach einer kostengünstigen Lösung gesucht, die Berechnung der einzelnen Neuronen zu parallelisieren. Hierfür ist die Benutzung von Grafikkarten eine geeignete und bewährte Lösung.

ERGEBNIS:

Die Umsetzung erfolgte mit einem PC, der mit drei NVIDIA P400 Grafikarten ausgerüstet wurde. Jede dieser Grafikkarten verfügt über 1792 Shader units. Zwei der Grafikkarten sind zudem über eine Brücke (SLI) miteinander verbunden (Abbildung 2). Dies ermöglicht den schnellen Datenaustausch zwischen den verbundenen Grafikkarten ohne den PCIe-Bus passieren zu müssen, der leicht zum Flaschenhals werden kann. Der Grafiktreiber stellt eine Schnittstelle zur NVIDIA CUDA-Bibliothek zur Verfügung, welche es ermöglicht, die Shader units der Grafikkarte für Berechnungen auf Anwendungsebene zu benutzen. Die durch die Parallelisierung zu erreichende Beschleunigung wird beispielsweise bei der Berechnung eines Mandelbrotbäumchens mit Hilfe von Mathlab [1] einmal ohne und einmal mit Verwendung der Stream-Prozessoren der Grafikkarte deutlich. Berechnet man diese Struktur ohne Unterstützung der Grafikkarte, so benötigt ein handelsüblicher Rechner je nach CPU eine Zeit im Bereich von einer bis zehn Sekunden. Mit Hilfe einer Grafikkarte können diese Berechnungen dank der CUDA-Schnittstelle von MATHLAB um den Faktor 760 [2] beschleunigt werden.

Als Betriebssystem kommt Scientific Linux 7 zum Einsatz. Diese Linux-Distribution wurde speziell für wissenschaftliche Anforderungen zusammengestellt. Darüber hinaus existiert für den Linuxkernel ein Treiber für die verwendeten Grafikkarten, der die CUDA-Schnittstelle unterstützt, was beispielsweise für FreeBSD nicht der Fall ist, wie sich bei der Umsetzung zeigte. Die Rechenleistung der Grafikkarten kann mit Hilfe unterschiedlicher Programmiersprachen genutzt werden. Es gibt Anbindungen an C und Java, aber auch für Scriptsprachen wie Python. Das Projekt cl-cuda implementiert eine Anbindung der CUDA-Schnittstelle an Common Lisp und stellt eine weitere interessante Option zur Implementation einer parallelisierten Anwendung dar.

Die Hardware soll in Zukunft in der Lehre zum Einsatz kommen. Hier können Studierende mit Hilfe von Tensorflow mit Python Bilderkennungssoftware implementieren und diese dann mit Hilfe der Grafikkarten beschleunigen. Weitere Anwendungsfelder der Hardware sind die Umsetzung unterschiedlicher Forschungsprojekte, bei denen rechenintensive Aufgaben anfallen.



Literaturverzeichnis

[1] Wolf Dieter Pietruszka; MATHLAB und Simulink in der Ingenieurpraxis; Springer Verlag 4. Auflage 2014, ISBN 978-3-658-058-06419-8

[2] https://www.mathworks.com/help/parallel-computing/examples/illustrating-three-approaches-to-gpu-computing-the-mandelbrot-set.html Abruf: 23.07.2019