Parallelisierung eines NVIDIA GPGPU Client Server Services basierend auf TCP-Sockets

Til Koke
Institut für Betriebssysteme und Rechnernetze
TU Braunschweig
Mühlenpfordstr.23, 38106 Braunschweig
Email: til.koke@tu-bs.de

Maximilian Wesche
Institut für Betriebssysteme und Rechnernetze
TU Braunschweig
Mühlenpfordstr.23, 38106 Braunschweig
Email: maximilian.wesche@tu-bs.de

David Winterland
Institut für Betriebssysteme und Rechnernetze
TU Braunschweig
Mühlenpfordstr.23, 38106 Braunschweig
Email: david.winterland@tu-bs.de

***Abstract*—The abstract goes here.**

I. Einleitung

Es gibt große Probleme, die bei der seriellen Berechnung langsam zu lösen sind. Manche dieser Probleme lassen sich sehr gut parallelisieren, da sie aus vielen kleinen Teilproblemen bestehen. Ein Beispiel hierfür ist die Berechnung einer Disparitätenmatrix aus Stereobildern. Ein CPU besitzt aber nur eine begrenzte Anzahl von Kerne, um parallel zu Arbeiten. Eine GPU hingegen besitzt sehr viele Kerne, die auf Rechenoperationen ausgelegt sind. Diesen hohen Grad an Parallelität für Probleme abseits des Renderings zu nutzen, nennt sich "General Purpose" Berechnungen auf GPUs (GPGPU). Compute Unified Device Architecture (CUDA) ist eine Bibiliothek die es erlaubt, auf NVIDIA GPUs eigenen, C ähnlichen Code auszuführen. Da aber auch die Anzahl der Kerne einer GPU begrenzt ist, ist es möglich für sehr große Probleme die Rechenleistung mehrerer GPUs zu verwenden. Dafür gibt es bereits eine Anwendung die einen GSoap Webservice verwendet, um die Berechnung der Disparitätenmatrix auf die einzelnen GPUs zu verteilen. Dadurch das jedem Bildpunkt ein xml-tag angefügt wird, ist die Übertragung sehr langsam.

Deswegen ist das Ziel dieser Arbeit, die Serialisierung der Bilder mithilfe von nativen C Sockets durchzuführen, um den Overhead gegenüber dem Webservice zu veringern und einen tatsächlichen Geschwindigkeitsvorteil gegenüber der Standard CUDA Implementierung zu erreichen.

Algorithmus mi CUDA SOAP hat Overhead Motiv für C-Sockets: weniger Overhead
(Performance)-Vergleich mit aktuellem Soap Ansatz

II. Ansatz

Implementiert mit nativen C-Client-Server Sockets nachher Rechnerverbund um Aufgabe verteilt berechenen zu können.

A. Unterkapitel xy

Subsection text here.

III. Implementierung & Validierung

HW Beschreibung: 1080GTX Titan, x Anzahl an Shader TCP Stream Sockets, Datenstruktur Beschreibung, welche Daten werden gesendet?

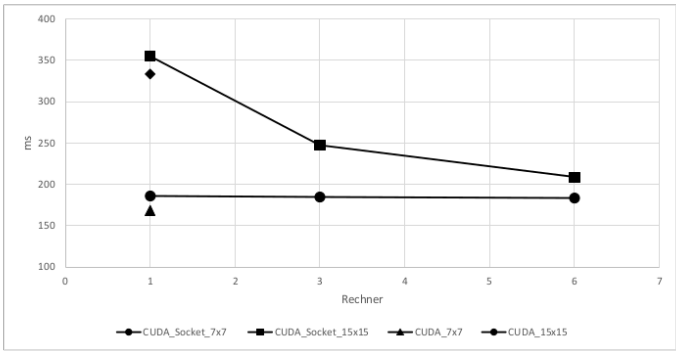

Fig. 1. Vergleich der Socket Implemtation gegenüber dem CUDA-Algorithmus auf einer Grafikkarte.

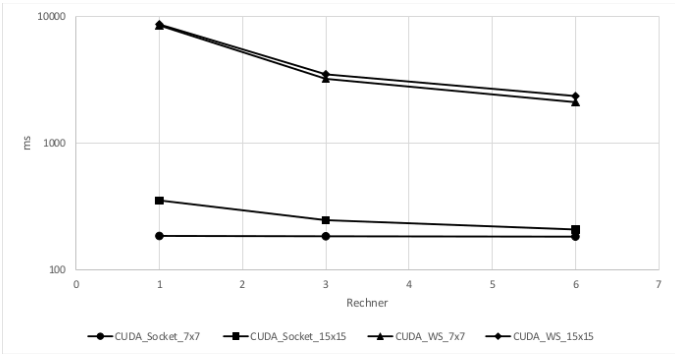

Fig. 2. Vergleich der Socket Implementation gegenüber der Webservice Variante.

IV. Zusammenfassung und weitere Arbeiten

Evaluation welches Ergebnis schneller war und warum. TCP Sockets aktuell verwendet, man könnte es auch mit UDP Sockets(Datagram Sockets) machen. Oder eine andere Midlleware bspw. JAVA RMI

References

[1] GSOAP, https://www.genivia.com/doc/soapdoc2.html#tth_sEc11.11, 6.11.2018.