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, bjlad! The abstract goes here, bjlad! The abstract goes here, bjlad! The abstract goes here, bjlad! The abstract goes here, bjlad!**

I. Einleitung

Verteilte Systeme helfen bei der Berechnung von komplexen mathematischen Problemen. Durch eine verteilte Berechnung kann das Ergebnis schneller berechnet werden, somit ergeben sich zum Teil große Performance Verbesserungen. Ein allgemein bekanntes Beispiel für parallele Berechnungen ist das Multi-Threading in gängigen Prozessoren von PCs. Hier werden große Aufgaben, wie bspw. Berechnungen, auf mehrere CPU Kerne aufgeteilt, typischerweise besteht eine CPU aus 2 bis 6 Kernen. Bei der Berechnung von grafikintensiven Aufgaben, bspw. die Berechnung einer Disparitätenmatrix aus Stereobildern, bieten GPUs auf Grund der großen Anzahl an Kernen und der Möglichkeit diese parallel zu nutzen enorme Performance Vorteile gegenüber CPUs. Diesen hohen Grad an Parallelität für Probleme abseits des Renderings zu nutzen, nennt sich "General Purpose" Berechnungen auf GPUs (GPGPU). Da aber auch die Anzahl der Kerne einer GPU begrenzt ist, ist es möglich für sehr große Probleme die Rechenleistung mehrerer GPUs, verteilt über das Netzwerk, zu verwenden. Für die Verteilung der Aufgaben im Netzwerk gibt es bereits eine Anwendung, die Performance Optimierung dieser Anwendung mit einem Ansatz der auf der Serialisierung mit nativen C-Client-Server-Sockets basiert ist Hauptbestandteil dieser Arbeit. Der Abschluss der Arbeit besteht aus einem Performance Vergleich der beiden Anwendungen bei der verteilten Berechnung von Disparitätenmatrizen aus Stereobildern.

II. Ansatz

Compute Unified Device Architecture (CUDA) ist eine Bibliothek die es erlaubt, auf NVIDIA GPUs eigenen, C ähnlichen Code auszuführen. Um Rechenaufgaben auf mehrere GPUs im Netzwerk zu verteilen und für den Zeitpunkt der Berechnung ein lokales Rechencluster zu bilden gibt es bereits eine Anwendungen 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. Im Vergleich zu einer lokalen GPU Berechnung ist die verteilte Berechnung durch den Overhead der Serialisierung langsamer. Der Ansatz dieser Arbeit ist es die Verteilung basierend auf C-Client-Server Stream-Sockets zu realisieren

III. Implementierung & Validierung

Die verteilte GPGPU Berechnung der Disparitätenmatrixauf wird auf 6 identischen Alienware Aurora PCs ausgeführt, diese besitzen folgende Hardware:

- Intel i7 6400 Prozessor mit 4 GHz
- 8 GB Arbeitsspeicher
- NVidia GTX1080 Grafikkarte mit 2560 CUDA Kernen und 8 GB VRAM
- Alle Rechner sind mit Gigabit Ethernet verbunden

Bei der verteilten Anwendung werden die Bilder von der Anwendung Stereo-Lab zeilenweise aufgeteilt, auch der CUDA Algorithmus wird bereits von der Stereo-Lab Anwendung bereitgestellt. Die Verteilung der Bilddaten für die Berechnung der Disparitätenmatrix basiert auf Client-Server Sockets. Hierbei werden TCP-Stream Sockets verwendet und die Datenpakete mit folgenden Daten gefüllt:

- **Bild Höhe** in absoluten Pixeln.
- **Bild Breite** in absoluten Pixeln.
- **Tau** Suchbereich des Berechnungsfensters.
- **Fenster Höhe** des Berechnungsfenster in absoluten Pixeln.
- **Fenster Breite** des Berechnungsfenster in absoluten Pixeln.

Der genaue Paketaufbau ist in Abbildung **??** zu sehen. Nach dem Initial Paket folgen noch zwei weitere Pakete. Die Größe der Pakete ist von der Pixelgröße des Bildes abhängig. Zur Berechnung der Disparität müssen sowohl das Linke als auch das Rechte Bild übertragen werden. Das berechnete Stereo Bild wird nach der Berechnung zum Client zurückgesendet. Die Bilder können auf beliebig viele Rechner verteilt werden. Im weiteren Verlauf dieser Arbeit erfolgt ein Vergleich zwischen den beiden Implementierungen: CUDA WS mit gsoap und XML CUDA Socket

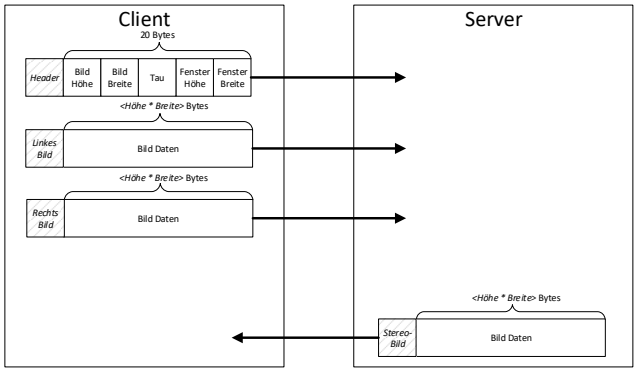

Fig. 1. Paketaufbau Client-Server-Sockets.

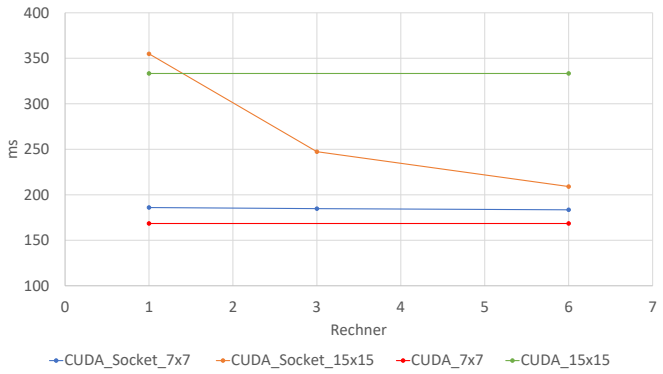

Fig. 2. Vergleich der Socket Implementation gegenüber dem CUDA-Algorithmus auf einer Grafikkarte.

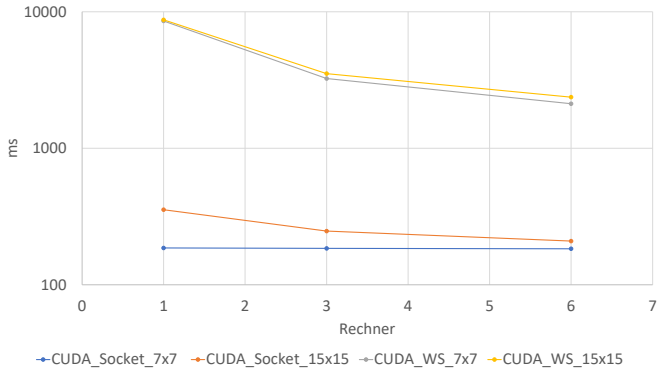

Fig. 3. Vergleich der Socket Implementation gegenüber der Webservice Variante.

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.