Verteilte Disparitätsberechnung mit WebServices und Base64 Encoding

Peter Kathmann,[1*] Anton Antonenko,[1] Martin Hepke[2]

[1]Systemarchitekturen für Verteilte Systeme, Technische Universität Braunschweig,

In der hier vorliegenden Ausarbeitung wird die Erzeugung von Disparitätsbildern von zwei Stereoskopischen Bildern über ein verteiltes Netzwerk betrachtet. Mittels CUDA Programmiertechnik wird die Berechnung auf mehrere Grafikprozessoren aufgeteilt. Dieses Paper betrachtet die Effizienz des Systems, wenn die Datenübertragung zwischen den Rechnern mittels einer Base64-Verschlüsselung stattfindet.

1 Einführung

Das hier präsentierte Paper behandelt die im Seminar *Distributed Filesystem* aufgetragene Anpassung des bestehenden Systems zur Ermittlung von Disparitäts-Karten aus zwei vorliegenden Stereobildern. Das vorgegebene System operiert mittels CUDA und gsoap über ein Netzwerk von Rechnern um die rechenaufwändige Aufgabe in kleinere Teilaufgaben zu unterteilen und damit die Rechenzeit insgesamt zu verkürzen. Im Seminar wurde eine Anpassung bzw. Verbesserung des Systems von Seiten der Teilnehmer gefordert, was zentraler Bestandteil dieser Ausarbeitung ist. Unsere Gruppe *"Ein Cooler Name"* entschied sich dafür, die Daten, die vom Server an die angemeldeten Clienten weitergeleitet werden, nicht im XML-Format zu versenden sondern stattdessen die Bildwerte im das Base64 Format zu verschicken und vor ort wieder in Bildinformationen zurück zu konvertieren. Als erhofftes Ergebnis war eine Laufzeitverringerung erhofft, die durch das Aussetzen einer aufwendigen XML-Konvertierung auftreten sollte.

Das Nachfolgenden Paper beschreibt dies in den folgenden Kapiteln. Im Kapitel "Implementierung" erfolgt eine Beschreibung vom programmiertechnisch Aufwand und Entscheidungen. Daraufhin folgt eine Beschreibung des Testverlaufs und der Ergebnisse im Kapitel "Auswertung". Das Paper schließst mit dem einem Fazit und beschreibt auf Basis der Testerwerte das Ergebnis dieser Ausarbeitung und ob die ursprüngliche Idee zu einem Erfolg führte unter dem Punkt "Messergebnisse".

2 Implementierung

Im folgenden Kapitel wird die Implementierung der Schnittstelle näher betrachtet. Zunächst wird die bestehende Schnittstelle analysiert und es wird definiert, welche Änderungen durchzuführen sind. Anschließend wird die Implementierung der Änderungen näher untersucht, bevor im folgenden Kapitel die Ergebnisse betrachtet und ausgewertet werden.

2.1 Ausgangslage

Wie in der Einleitung beschrieben, sollte eine bestehende Softwarelösung abgeändert werden. In der bestehenden Lösung wurde die verteilte Ausführung der CUDA-Berechnungen über SOAP-Requests eingeleitet, welche die Bilddaten als Array von Integern beinhalteten. Die

primären Nachteile der Übertragung der Bilddaten als Array von Integern liegen darin, dass zum Einen ein erhöhter Netwerk-Traffic anfällt, zum Anderen auch ein nicht unerheblicher Aufwand durch das Auswerten der XML-Daten entsteht.
Der erhöhte Netzwerk-Traffic lässt sich auf die Struktur von XML-Dokumenten zurückführen, welche pro Datum ein öffnendes und ein schließendes Tag benötigen. Da es sich bei den Daten um ein Graustufen handelt, werden Integer übertragen, die im Allgemeinen nicht mehr als drei Stellen beinhalten. In diesem Fall bestehen die Tags jeweils aus mehr Zeichen als der zu übertragende Wert. Hierdurch entsteht ein nicht zu vernächlässigender Overhead, welcher bei einer Übertragung als Base64-Codiertes Datum minimiert wird, da die Tags nur ein mal pro Bild übertragen werden müssen.
Der Aufwand, welcher durch die Auswertung der XML-Daten entsteht, ist daher nicht unerheblich, da zum einen die Daten als Strings vorliegen, welche in Integer konvertiert werden müssen. Zum anderen jedoch bei SOAP-Schnittstellen im Allgemeinen auch eine Validierung der Daten vorgenommen wird. Aufgrunddessen sollte klar sein, warum die Übertragung der Bilder als Arrays einzelner Integer einen h´oheren Aufwand erzeugt, als die Übertragung zweier Base64-Codierter Strings.

2.2 Durchführung

In diesem Abschnitt wird die Durchführung der im vorherigen Kapitel beschriebenen Änderungen erklärt. Die vorzunehmenden Änderungen lassen sich in zwei Teilschritte unterteilen. Zum einen die Anpassung der SOAP-Schnittstelle und zum anderen die Konvertierung der Bilddaten in Base64.
Um die SOAP-Schnittstelle anzupassen wurde die WSDL dahingehend geändert, dass die Bilder nun nicht mehr als Arrays einzelner Pixel übertragen werden, sondern als einzelner String. Auf Basis der geänderten WSDL wurden unter Verwendung des Tools gsoap Client- und Serverseitig Code erzeugt, welcher die Abhandlung des SOAP-Prozesses ermöglicht.
Um die Bilder als String zu übertragen, müssen diese noch in das Base64-Format konvertiert werden. Hierzu wurde eine einfache Konvertierungsfunktion implementiert. Diese Funktion liefert die binären Eingabedaten als Base64-String zurück. Bei der Übertragung wird dieser String zusammen mit der Größe des Originalbildes versandt. Die Serverseite muss nun auf Basis der Originalgröße die Größe der Daten als Base64-String errechnen und kann anschließend unter Zuhilfenahme der neu errechneten Base64-Größe wieder ein Bild erzeugen. Diese Bilder werden dann miteinander verglichen und die Ergebnisse werden weiterhin als Array von Inte-

gern an den Client zurück gesandt.

3 Auswertung

Im folgenden Kapitel werden die Messergebnisse der vorhandenen WebService Implementierung und der modifizierten Version mit Base64-Encoding verglichen. Dazu wird zuerst die Testumgebung sowie die Einstellungen die wir vorgenommen haben vorgestellt auf der wir die Implementierungen ausgeführt haben.

3.1 Testumgebung

Dieser Abschnitt soll die Umgebung und Einstellungen mit denen wir die Implementierungen gegeneinander getestet haben beschreiben.
Die Testsysteme waren 3 Computer mit folgenden Spezifikationen:

CPU Intel Core i7-6700K CPU @ 4.00GHz 8

GPU GeForce GTX 1080

RAM 31.2 GiB

OS Ubuntu 16.04 TLS

Für alle Tests wurden die gleichen Bilder genutzt: (horse_left_886x818.raw, horse_right_886x818.raw) in 3 unterschiedlichen Auflösungen (100%, 70%, 30%).Bei 100% und 30% Auflösung wurde die Fenstergröe 15x15 und TauMax von 40 gewählt. Um zu prüfen, wie der Einfluss der Fenstergröße und des TauMax Wertes ist, wurden 3 verschiedene Kobinationen für die Auflösung 70% gewählt:

Settings A: Tau 20 - Fenster 7x7

Settings B: Tau 40 - Fenster 7x7

Settings C: Tau 20 - Fenster 15x15

3.2 Messergebnisse

In diesem Abschnitt werden die Messergebnisse vorgestellt.

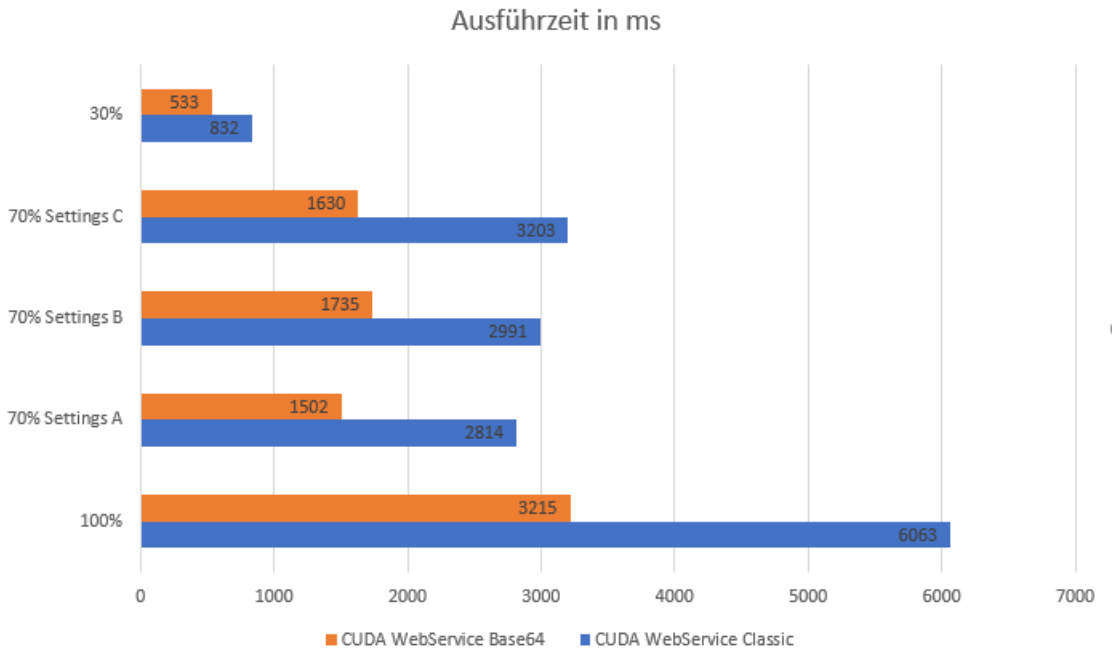

Figure 1: Messergebnisse

Anhand der Ergebnisse kann man sehen, dass selbst bei einer geringen Auflösung des Ausgangsbildes von 30%, eine Reduktion der Ausführzeit von etwa 35% erreicht wird, bei den höhereh Auflösungen sogar 42-49%. Die Base64-Implementierung kodiert und dekodiert die Rohdaten und erhöht deren Größe um 33-36 %, dennoch ist sie fast doppelt so schnell. Das deutet darauf hin, dass der Overhead bei der Array Lösung größer ist und das Parsing einzelner Bytes aufwendig.
Möchte man also nicht auf WebServices verzichten, aber trozdem eine Performance Steigerung erreichen, ist das Übertragen eines Base64 Kodierten Strings anstatt einzelner Bytes eine Alternative die man in Betracht ziehen sollte.