TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
Die Ressourcenuniversität. Seit 1765.

Fakultät für Mathematik und Informatik
Institut für Informatik
Lehrstuhl für Betriebssysteme und Kommunikationstechnologien

Seminararbeit

Aufbau eines Prototyps für verteilte CUDA Programmierung

Development of a prototype for distributed CUDA programming

Samuel Dressel

Angewandte Informatik
Vertiefung: Parallelrechner

Matrikel: 59963

28. August 2019

Betreuer/1. Korrektor:
Prof. Dr. Konrad Froitzheim

2. Korrektor:
Dr. rer. nat. Martin Reinhardt

Inhaltsverzeichnis

1. Einleitung **4**

2. Grundlagen **4**
2.1. Message-Passing-Interface (MPI) 4
2.2. CUDA-Aware MPI 4

3. Technischer Aufbau und Konfiguration **4**

4. CUDA und MPI **6**

5. Benchmarking und Test ausgewählter Algorithmen **6**

6. Fazit **6**

Anhang **7**

A. Aufbau des Clusters **7**

Literatur **8**

Eidesstattliche Erklärung

Ich versichere, dass ich diese Arbeit selbständig verfasst und keine anderen Hilfsmittel als die angegebenen benutzt habe. Die Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, habe ich in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Diese Versicherung bezieht sich auch auf die bildlichen Darstellungen.

28. August 2019 Samuel Dressel

1. Einleitung

2. Grundlagen

2.1. Message-Passing-Interface (MPI)

Das Message-Passing-Interface, kurz MPI, ist eine standartisierte Schnittstelle für die Kommunikation zwischen verteilten Prozessen.

2.2. CUDA-Aware MPI

3. Technischer Aufbau und Konfiguration

Für den Aufbau des Mini-Clusters werden zunächst folgende Dinge benötigt:

- Nvidia Jetson Nano (3x)
- SD-Karte (3x)
- Ethernet-Kabel (4x)
- 4-Port Ethernet Switch
- Optional: 40mm Lüfter (3x, in diesem Fall wurde der Noctua NF-A4x20 verwendet)

Der technische Aufbau beginnt mit der physischen Verbindung der Nvidia Jetson Nanos. Dazu werden diese alle via Ethernet-Kabel mit einem Ethernet Switch verbunden, welcher wiederum durch LAN mit einem Router und dadurch mit dem Internet verbunden ist. An sich benötigt der Cluster keinen Internetzugang, für die Einrichtung ist dieser jedoch unersetzlich. Einer der Jetson Nanos fungiert als Head-Node (Masterknoten), die anderen zwei als Worker-Nodes.
Ist das physische Setup abgeschlossen, wird als nächstes auf jedem der Jetson Nanos das Jetson Nano Developer Kit SD Card Image installiert. Dies geschieht durch das Schreiben des Images auf die jeweilige SD-Karte und einem anschließenden Setup auf den Jetson Nanos [1].
Danach wird das System durch Aktualisierung der Pakete auf den neuesten Stand gebracht:

```
sudo apt-get update
sudo apt-get upgrade
```

Es folgt die Installation des `nano`-Texteditors und des SSH-Paketes. Dabei ist SSH für den Remotezugriff auf die einzelnen Knoten notwendig; an Stelle des `nano`-Texteditors kann alternativ jeder andere Editor benutzt werden.

```
sudo apt-get install nano
sudo apt-get install openssh-server
```

Zusätzlich ist auf dem Masterknoten der NFS-Kernel-Server zu installieren:

```
sudo apt-get install nfs-kernel-server
```

Auf den Worker-Nodes wird dagegen das `nfs-common`-Paket installiert:

```
sudo apt-get install nfs-common
```

NFS und die damit verbundenen Pakete sind notwendig, um später Dateien innerhalb des Clusters auszutauschen.
Um dieses Netzwerk einzurichten, muss zunächst jedem der drei Knoten eine statische IP zugewiesen werden. Dies erfolgt entweder über die Netzwerkeinstellungen oder über das Bearbeiten der Interface-Datei:

```
cd /etc/network
sudo nano interfaces
```

Dafür werden der Datei folgende Zeilen hinzugefügt:

```
auto eth0
iface eth0 inet static
address 192.168.178.10
gateway 192.168.178.1
netmask 255.255.255.0
```

Dabei ist die Adresse `192.168.178.10` die Adresse des Masterknotens, die Worker-Nodes erhalten dann dementsprechend die IP-Adressen `192.168.178.11` und `192.168.178.12`. Damit die Änderungen wirksam werden, müssen entweder die Jetson Nanos oder der Network-Service neugestartet werden.
Als nächstes erfolgt die Einrichtung von SSH innerhalb des Clusters. Dazu wird zuerst jedem Knoten ein Name zur besseren Identifikation zugewiesen. Hierfür wird die `hostname`-Datei bearbeitet:

```
sudo nano /etc/hostname
```

Der Masterknoten erhält in diesem Fall den Namen `master`, die Worker-Nodes die Namen `slave1` und `slave2`. Als nächstes werden in der `hosts`-Datei die statischen IP-Adressen aller Knoten hinzugefügt. Dies geschieht wie der vorige Schritt auf allen Knoten:

```
sudo nano /etc/hosts
```

Die `hosts`-Datei sollte dann so aussehen:

```
192.168.178.10 master
192.168.178.11 slave1
192.168.178.12 slave2
```

Nach dem Einrichten der statischen IP-Adressen folgt nun das Setup von SSH. Dazu wird zunächst ein 2048 bit RSA Schlüsselpaar auf dem Masterknoten erstellt:

```
ssh-keygen -t rsa -b 2048
```

Danach wird die SSH ID an alle Knoten inklusive des Masterknotens weitergeben:

```
ssh-copy-id master
ssh-copy-id slave1
ssh-copy-id slave2
```

Eine Kommunikation zwischen den Knoten ohne ständige Passwort- bzw. Berechtigungsabfrage ist für die Funktionalität des Cluster von großer Bedeutung. Dies geschieht durch das Generieren der `known_hosts`-Datei im `.ssh`-Ordner. Dazu wird zunächst eine Datei mit den Namen aller Knoten im `.ssh`-Ordner angelegt:

```
cd .ssh
sudo nano name_of_hosts
```

Die Datei enthält dann folgende Einträge:

```
master
slave1
slave2
```

Damit der `ssh-keyscan` die Datei lesen kann, müssen anschließend die Zugriffsberechtigungen geändert werden:

```
sudo chmod 666 ~/.ssh/name_of_hosts
```

Mit folgendem Befehl wird letztendlich die `known_hosts`-Datei erzeugt:

```
ssh-keyscan -t rsa -f ~/.ssh/name_of_hosts >~/.ssh/known_hosts
```

Als letztes muss diese Datei und die notwendigen Schlüssel noch an die anderen Knoten kopiert werden:

```
cd .ssh
scp known_hosts id_rsa id_rsa.pub nvidia@master:.ssh
scp known_hosts id_rsa id_rsa.pub nvidia@slave1:.ssh
scp known_hosts id_rsa id_rsa.pub nvidia@slave2:.ssh
```

Der letzte Schritt der Konfiguration ist das Erstellen und Mounten eines gemeinsamen Arbeitsordners für alle Knoten. Hierfür das Network File System benutzt, dessen Pakete anfangs installiert wurden. Auf dem Masterknoten wird dabei als erstes dieser Arbeitsordner erstellt:

```
sudo mkdir /cloud
```

Danach muss die `exports`-Datei auf dem Masterknoten editiert werden:

```
sudo nano /etc/exports
```

Diese Datei enthält alle Informationen über das Exportieren des Arbeitsordners auf die einzelnen Knoten:

```
/cloud slave1(rw,sync,no_root_squash,no_subtree_check)
/cloud slave2(rw,sync,no_root_squash,no_subtree_check)
```

Die zwei Worker-Nodes müssen nun diesen gemeinsamen Arbeitsordner mounten. Dazu wird auf jedem Worker-Node der `cloud`-Ordner erstellt und mithilfe des Editierens der `fstab`-Datei gemountet:

```
sudo mkdir /cloud
sudo nano /etc/fstab

master:/cloud /cloud nfs rsize=8192,wsize=8192,timeo=14,intr
```

Der technische Aufbau und die Konfiguration des Clusters ist hiermit abgeschlossen.

4. CUDA und MPI

5. Benchmarking und Test ausgewählter Algorithmen

6. Fazit

A. Aufbau des Clusters

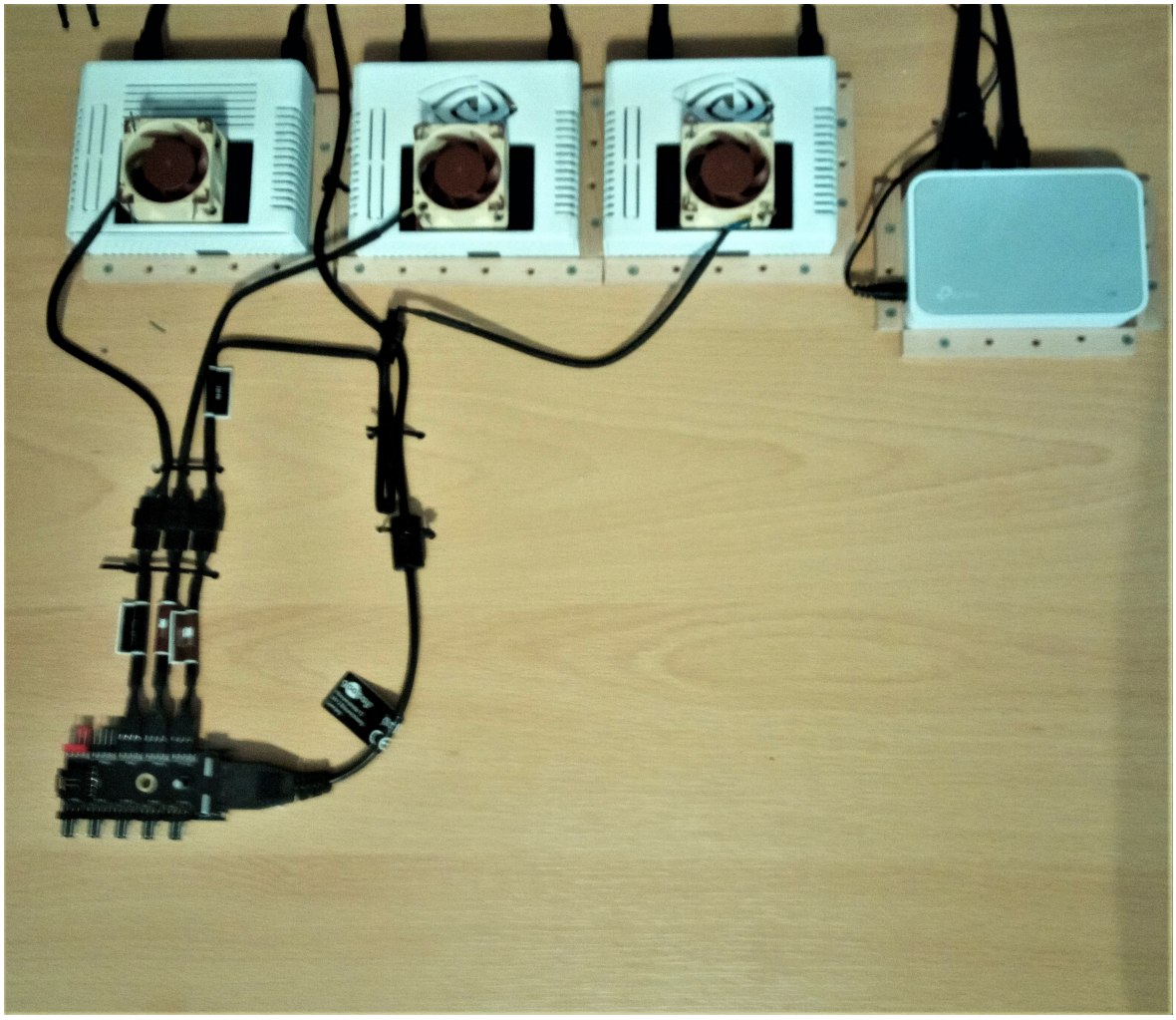

Abb. 1: Der im Rahmen dieses Projektes erstellte Mini-Cluster. Zur besseren Nutzung wurden sowohl die Nvidia Jetson Nanos als auch der Switch durch seitliche Leisten auf einem Holzbrett fixiert. Um die vollständige Leistung durch die Nutzung des Powermodus zu erzielen, wurde desweiteren auf jedem Jetson Nano ein 40mm Lüfter installiert. Für die Steuerung dieser Lüfter wurde ein externer Lüfter-Controller genutzt, weil die Lüfter eine Betriebsspannung von 12V benötigen, der Jetson Nano jedoch nur Lüfter mit 5V Betriebsspannung unterstützt.

Literatur

[1] Getting Started With Jetson Nano Developer Kit. https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit. Zuletzt besucht: 28.08.2019.