Frankfurt University of Applied Sciences

Fachbereich 2 - Studiengang Informatik

Bachelorthesis

zur Erlangung des akademischen Grades

Bachelor of Science

Installation und Analyse des Dateisystems ZFS auf Einplatinencomputern

Autor: Aleksej Antonov

Matrikel-Nummer: 1019139

Referent: Herr Prof. Dr. Baun

Korreferent: Herr Prof. Dr. Gabel

Eidesstattliche Erklärung

Hiermit versichere ich,

Antonov, Aleksej, geboren am 21.10.1981 in Karaganda,

dass ich die vorliegende Bachelorarbeit selbstständig verfasst und auch keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Ausführungen, die anderen veröffentlichten oder nicht veröffentlichten Schriften wörtlich oder sinngemäß entnommen wurden, wurden kenntlich gemacht.

Die Bachelorarbeit hat in gleicher oder ähnlicher Fassung noch keiner anderen Prüfungsbehörde vorgelegen.

_____ _____

Ort, Datum Unterschrift

Danksagung

Auf diesem Wege möchte ich mich bei allen bedanken, die mir diese Bachelorarbeit ermöglicht haben. Vor allem möchte ich mich bei meiner Familie bedanken, die in der Erstellungszeit dieser Arbeit auf viel gemeinsame Zeit verzichten musste.

Für die Betreuung und die Idee zu dieser Abschlussarbeit danke ich Prof. Dr. Christian Baun sowie Prof. Dr. Thomas Gabel für seine Bereitschaft als Korreferent aufzutreten.

Zusammenfassung

Die vorliegende Bachelorarbeit beschäftigt sich mit der Installation, Konfiguration und der Analyse des Dateisystems ZFS auf einem Raspberry Pi 3 Model B mit dem Betriebssystem Raspbian. Das Dateisystem ZFS wurde für das Verwalten großer Datenmengen entwickelt. Die Installation auf dem Raspberry Pi soll die Möglichkeiten, zum einen von ZFS, zum anderen von dem Raspberry Pi 3 Model B, aufzeigen. Die Installation und Konfiguration des Systems, werden ausführlich beschrieben, ebenso die ersten Schritte mit dem Umgang mit ZFS. Ein erstellter RAID-Z, dient zum Analysieren der Geschwindigkeiten sowie zum Testen der Ausfallsicherheit.

Abstract

The present bachelor thesis deals with the installation, configuration and analysis of the file system ZFS on a Raspberry Pi 3 Model B with the Raspbian operating system. The file system ZFS was developed to manage large bulks of data. The installation on the Raspberry Pi should demonstrate the possibilities of ZFS on the one hand as well as Raspberry Pi 3 Model B on the other hand. The installation and configration of the system is described in detail as well as the first steps of the approach to ZFS. A RAID-Z is created in order to analyse the speed as well to test the failure safety.

Inhaltsverzeichnis

Abkürzungsverzeichnis

APT	Advanced Packaging Tool
DDR	Double Data Rate
EiB	Exbibyte
GPIO	General-Purpose Input/Output
HDMI	High Definition Multimedia Interface
ID	Identifikation
IP	Internetprotokoll
IT	Informationstechnik
KB	Kilobyte
KB/s	Kilobyte pro Sekunde
MB	Megabyte
MB/s	Megabyte pro Sekunde
Micro-SD	Micro Secure Digital Memory Card
PiB	Pebibyte
RAM	Random Access Memory
SD	Secure Digital Memory Card
SO-DIMM	Small Outline Dual Inline Memory Module
SSH	Secure Shell
USB	Universal Serial Bus
WLAN	Wireless LAN

Abbildungsverzeichnis

Tabellenverzeichnis

Diagrammverzeichnis

1 Überblick der Technik

Dieses Kapitel zeigt einen Überblick über die eingesetzte Technik in dieser Bachelorarbeit.

1.1 Raspberry Pi

Der Raspberry Pi ist ein Einplatinencomputer der britischen Raspberry Pi Foundation. Die Foundation entwickelte diesen Computer, um Schüler und Studenten für Computerwissenschaften zu begeistern. [1] Die Entwickler und Bastler setzen die Geräte, unter anderem als Experimentierplattform, als Medienzentrale zur Wiedergabe verschiedener Abspielformate, in der Robotik oder in der Hausautomation ein.

Der Preis für den aktuellen Raspberry Pi, den Raspberry Pi 3 Model B beträgt ca. 35 €. Für den Betrieb sind weitere Komponenten nötig, die nicht im Lieferumfang enthalten sind. Je nach dem, was mit dem Raspberry Pi realisiert werden soll, können die Kosten stark variieren. Für die Konfiguration und die ersten Schritte mit dem Raspberry Pi müssen eine Micro-SD-Speicherkarte, ein Netzteil, eine USB-Tastatur, eine USB-Maus, ein Bildschirm, ein HDMI-Kabel und ein Netzwerkkabel vorhanden sein.

Das Betriebssystem befindet sich auf einer Micro-SD-Speicherkarte. Als Betriebssystem für den Raspberry Pi empfiehlt die Foundation das System Raspbian, basierend auf der Linux-Distribution Debian. Die Foundation bietet das System auf der offiziellen Seite zum kostenlosen Herunterladen an. Ebenso ist es dort möglich das Betriebssystem auf einer Micro-SD-Speicherkarte käuflich zu erwerben. Aber auch andere, für den Raspberry Pi angepasste Linux-Distributionen oder auch Nicht-Linux-Distributionen wie zum Beispiel das Betriebssystem FreeBSD, können auf dem Einplatinencomputer installiert werden. Es besteht die Möglichkeit Microsoft Windows 10 IoT (Internet of things = Internet der Dinge) als Betriebssystem auf dem Raspberry Pi zu installieren. Dies hat aber keine klassische, von Windows gewohnte grafische Benutzeroberfläche, sondern eine speziell für die Programmierung gestaltete Oberfläche. Das

Ausführen von Windows-Programmen oder Diensten, die auf den herkömmlichen Rechnern mit Microsoft Windows 10 laufen, ist auf dem Raspberry Pi mit Windows 10 IoT nicht möglich.

1.1.1 Raspberry Pi Modelle

Eine Auflistung der Raspberry Pi Einplatinencomputer, die seit der Einführung im Jahre 2012 bis zu diesem Zeitpunkt erschienen sind (Stand April 2017) [2]:

Die erste Generation

- Raspberry Pi Model B (Vorstellung Februar 2012)
- Raspberry Pi Model A (Vorstellung Februar 2013)
- Raspberry Pi Model B+ (Vorstellung Juli 2014)
- Raspberry Pi Model A+ (Vorstellung November 2014)

Die zweite Generation

- Raspberry Pi 2 Model B (Vorstellung Februar 2015)

Die dritte Generation

- Raspberry Pi 3 Model B (Vorstellung Februar 2016)

Im November 2015 vorgestellte Raspberry Pi Zero, ist der $5-Raspberry. „Der Raspberry Pi Zero ist fast 50% kleiner als der Raspberry A+, verfügt aber gleichzeitig über mehr Leistung“. [3] Sein, im Februar 2017 vorgestellter, Nachfolger Raspberry Pi Zero W, ist identisch mit dem Pi Zero, hat aber zusätzlich integrierte WLAN- und Bluetooth-Module. „Die zusätzliche Hardware hat allerdings auch Auswirkungen auf den Preis: Der Pi Zero W soll um 10 Euro verkauft werden, also doppelt so "teuer" wie sein Vorgänger sein“. [4] Die Vorteile der beiden Zero-Modelle sind, der geringe Stromverbrauch und die noch kompaktere, platzsparende Bauweise. Die Nachteile sind die nicht vorhandenen USB-2.0-Anschlüsse und die nicht vom Werk aus angelötete GPIO-Leiste. Diese kann der Benutzer bei Bedarf, an der vorgegebener Stelle der Platine, anlöten.

Seit 2014 stellt die Raspberry Pi Foundation sogenannte Compute-Module her, die speziell für die industriellen Zwecke konzipiert sind. Diese Module haben alle Anschlüsse wie der Raspberry Pi, aber in Form von Kontakten. Die Größe des Compute-Module 3 beträgt 68 x 31 mm und hat die Form eines DDR 2 SO-DIMM-Speicherriegels. [5] Diese Bauweise ermöglicht den weiterverarbeitenden Betrieben platzsparend zu bauen.

Insgesamt verkaufte die Raspberry Pi Foundation 12,5 Millionen Raspberry Pis (Stand April 2017). Der Raspberry Pi 3 Model B ist das meistverkaufte Modell der Foundation. [6]

1.1.2 Raspberry Pi 3 Model B

Für diese Bachelorthesis wurde der Raspberry Pi 3 Model B gewählt. Wie aus dem Namen hervorgeht, ist es die dritte Generation des Einplatinencomputers. Der Verkauf dieser Generation startete im Februar 2016. Die Abmessungen des Raspberry Pi 3 Model B betragen 93 mm in der Länge, 63,5 mm in der Breite und 20 mm in der Höhe. [7] Diese Gesamtgröße ist seit dem ersten Raspberry Pi fast gleichgeblieben.

Anders als der Raspberry Pi 2 Model B und die Vorgänger, besitzt der Raspberry Pi 3 Model B ein integriertes WLAN-Modul und ebenso ein integriertes Bluetooth-Modul. Bei den Vorgängermodellen besteht die Option externe WLAN- oder Bluetooth-Module an die USB-Anschlüsse des Raspberry Pi anzuschließen. Der Vorteil der integrierten Module, neben den freien USB-Anschlüssen ist, in erster Linie, die Umgehung des Problems der Treiberunterstützung der externen Netzwerkadapter. [8] Nicht alle Adapter können, wegen der nicht vorhandenen Treibern, am Raspberry Pi funktionieren.

Die wohl wichtigste Neuerung am Raspberry Pi 3 Model B ist, dass es der erste Raspberry Pi ist, der einen 64-Bit-Prozessor (Broadcom BCM2837) erhielt. Dieser Prozessor kann im 64-Bit-Modus und im 32-Bit-Modus arbeiten, im Auslieferungszustand ist der Prozessor weiterhin auf 32-Bit eingestellt und startet auch in diesem. Der Grund für diese Einstellung ist, dass der Raspberry Pi 3 Model B mit seinen 1 GB RAM zu wenig von der 64-Bit-Einstellung profitiere und ein Leistungszuwachs nicht spürbar wäre. [9] Aber auch im 32-Bit-Modus ist der Prozessor dank höherer Taktfrequenzen, schneller als sein Vorgänger. [9]

Es gibt mehrere inoffizielle Anleitungen, die die Freischaltung der 64-Bit-Archtektur auf dem Raspberry Pi 3 Model B ermöglichen. Im November 2016 stellte OpenSUSE ein offizielles 64-Bit-Abbild für den Raspberry Pi vor.

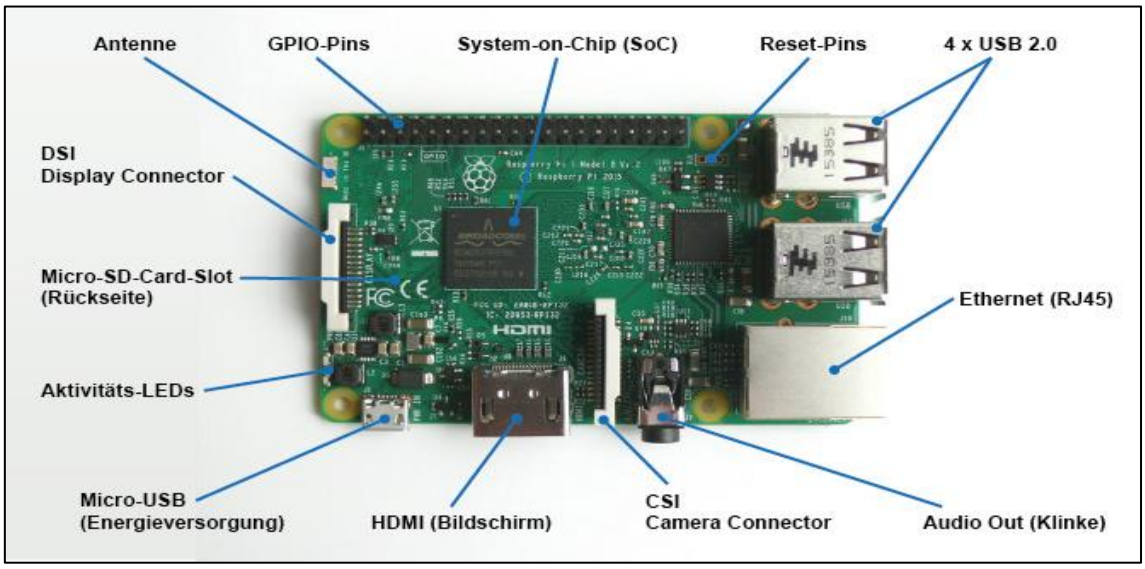

Abbildung 1: Raspberry Pi 3 Model B[1]

Ein Überblick der verbauten Komponenten des Raspberry Pi 3 Model B: [10]

- Broadcom BCM2837 ARM Cortex A53 64-Bit Prozessor 4x 1.2 GHz
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 4 USB 2.0 Anschlüsse
- Display-Schnittstelle (DSI)
- Grafikkarte: Dual Core VideoCore IV 3D
- Audio Ausgänge: 3.5 Klinke und HDMI
- Ethernet-Anschluss
- Kamera-Schnittstelle (CSI)
- HDMI-Anschluss
- 40 GPIO-Pins
- Micro SD Kartenschlitz

[1] Bildquelle: https://www.elektronik-kompendium.de/sites/raspberry-pi/1905251.htm Aufgerufen: 08.04.17

1.2 ZFS

Das im Jahre 2001 von IT-Unternehmen Sun Microsystems entwickelte Dateisystem ZFS, war ursprünglich für das Betriebssystem Solaris bestimmt. Die Abkürzung stand früher für Zettabyte File System (Zettabyte Dateisystem), heute wird diese Bezeichnung aber nicht mehr verwendet. [11]

„ZFS ist ein 128-bit-Dateisystem, das die Adressierung von 1.000.000.000.000.000.000.000 (10^{21}) Bytes erlaubt. Es ist wahrscheinlich das fortschrittlichste Dateisystem, das es derzeit gibt." [11]

Das Produkt ZFS ist eine Kombination aus einem Dateisystem und einem Plattenspeicherverwaltungsmanager. Das ZFS-Dateisystem ist für die Verwaltung großer Datenmengen, zum Beispiel für die Rechenzentren, konzipiert worden. Das Dateisystem ermöglicht, mit wenigen Befehlen, die Erstellung und Verwaltung von Speicherpools. Ein Speicherpool/Pool ist eine Zusammenfassung mehrerer Speichermedien, die zu einem logischen Laufwerk arrangiert sind. Es gibt die Möglichkeit des Kreierens weiterer Partitionen im selben Pool, dabei vererben diese die Parameter des Ausgangspools. [12]

Außerdem besitzt das System die Funktionsweise von einem Software-RAID mit einem Copy-on-Write-Dateisystem (Kopieren-beim-Schreiben), abgekürzt COW. Bei dieser Methode überschreibt das System die geänderten Dateien nicht, sondern legt sie an einen freien Speicherplatz auf dem System ab. [13] Diese Vorgehensweise ermöglicht die Erstellung der Schnappschüsse (Snapshots) eines Systems. Dabei werden alte Dateien oder Blöcke dem jeweiligen Schnappschuss zugeordnet und nicht gelöscht. [13] Ein Schnappschuss in der Informationstechnik bedeutet eine Speicherung des Systemabbildes zu einem vom Benutzer festgelegten Zeitpunkt. Die Schnappschussfunktion erlaubt dem Benutzer das komplette System oder Dateien auf den Zeitpunkt eines bestimmten Schnappschusses wiederherzustellen.

Bei der Entwicklung von ZFS wurde der Wert auf die einfache Bedienung des Systems gelegt. „ZFS kommt mit nur zwei Befehlen aus: zpool für Festplatten-orientiertes Management und zfs für den Rest (also was man früher womöglich Partitionieren genannt haben würde)." [11]

Das Dateisystem ZFS kann beliebig viele Daten verwalten, die Voraussetzung dafür ist die geeignete Hardware. [11]

> „ZFS überwacht die Integrität der verwalteten Daten, erkennt also auftretende Fehler selbst. Liegen die Daten in redundanter Form vor, wird es beim Erkennen von Fehlern selbständig deren Reparatur auslösen. Deswegen hat es den Ruf, Datenschäden selbstständig zu erkennen und zu reparieren, die in den darunterliegenden Schichten entstehen können. Es wurde sogar eine Technologie entwickelt, die das Write-hole-Problem bei RAID-5 umgeht und die RAID-Z genannt wird.“ [11]

Das Write-hole-Problem (Schreiblücke) tritt zum Beispiel bei einem Stromausfall auf. Das System kann nicht feststellen, welche Datenblöcke auf das Speichermedium schon geschrieben sind und welche nicht. [14]

Das ZFS Dateisystem besitzt die Funktion der Selbstheilung. Mittels der Checksumme erkennt das System, ob ein fehlerhafter Dateiblock vorhanden ist und ersetzt diesen auf der betroffenen Platte. [15] Die Abbildung 2 zeigt den Verlauf der Selbstheilung der fehlerhafte Block ist rot dargestellt.

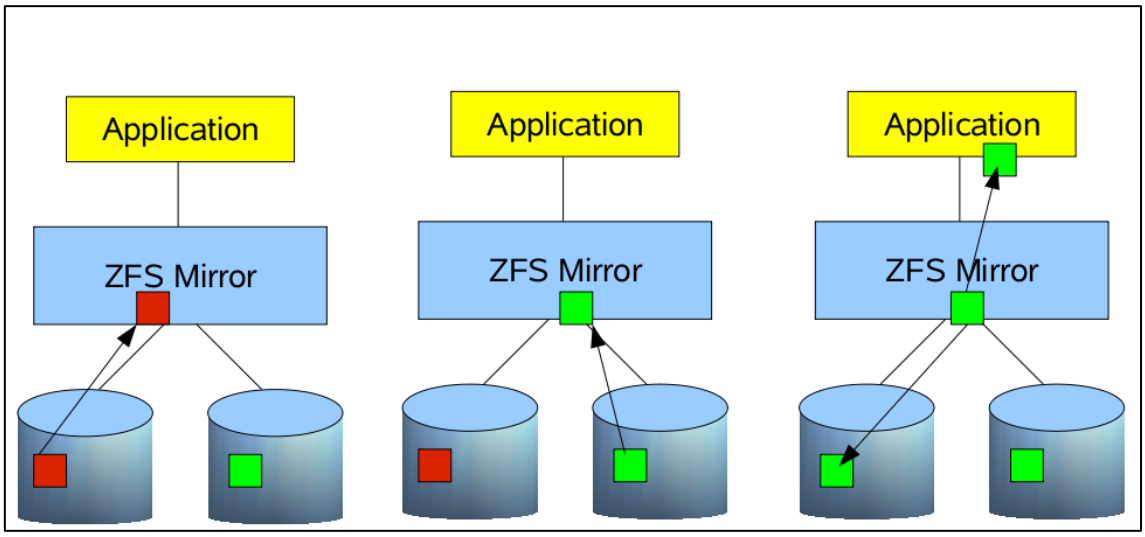

Abbildung 2: Selbstheilung in ZFS[2]

[2] Bildquelle https://www.root.cz/clanky/suborovy-system-zfs-konzistentnost-dat/ Aufgerufen: 20.04.17

Die Tabelle zeigt eine Übersicht der möglichen Größen in ZFS.

Wortlänge	128 Bit
Volumemanager	Integriert
Ausfallsicherheit	RAID 1, RAID-Z1 (1 Parity-Bit, ~RAID 5), RAID-Z2 (2 Parity-Bits, ~RAID 6) und RAID-Z3 (3 Parity-Bits) integriert
maximale Größe des Dateisystems	16 EiB (= 2^{64} Byte)
maximale Anzahl an Dateien in einem Dateisystem	2^{48}
maximale Größe einer Datei	16 EiB (= 2^{64} Byte)
maximale Größe jedes Pools	3×10^{23} PiB (ca. 2×10^{38} Byte)
maximale Anzahl an Dateien in einem Verzeichnis	2^{48}
maximale Anzahl an Geräten im Pool	2^{64}
maximale Anzahl an Dateisystemen im Pool	2^{64}
maximale Anzahl an Pools im System	2^{64}

Tabelle 1: ZFS Technische Daten[3]

Aus lizenzrechtlichen Gründen ist eine Integration in das Linux-Kernel nicht erlaubt, die Benutzung eines selbst kompilierten Kernel-Moduls dagegen schon. [11] Dies erfolgt unter Ubuntu mit dem so genanntem PPA (Personal Package Archive = eigenes Paketarchiv). Es ist ein Dienst, mit dem Nutzer eigene Debian-Pakete bauen, diese selber nutzen oder anderen Nutzern die erstellten Pakete als Links zur Installation freigeben. [16]

Seit Ubuntu 16.04 haben Nutzer die Möglichkeit ZFS-Pakete, aus den offiziellen Paketquellen zu installieren. [11] Juristen prüfen, ob Ubuntu damit gegen die Nutzungslizenzen verstößt. [17]

[3] Quelle: https://de.wikipedia.org/wiki/ZFS_(Dateisystem)#Datentr.C3.A4ger-Pools

1.3 RAID

Ein Redundant Array of Independent Disks (Redundante Anordnung unabhängiger Festplatten), abgekürzt RAID, ist eine Gruppierung mehrerer Speichergeräte, die zu einem logischen Laufwerk zusammengefasst sind. [18] Es gibt mehrere RAID-Systeme und RAID-Kombinationen aus diesen, die zum Einsatz kommen und so benutzerdefinierte Anforderungen abdecken. Das Erstellen eines RAIDs soll unter anderem eine höhere Ausfallsicherheit der Speichermedien gewährleisten, da die Datenspeicherung in einem RAID-Verbund redundant auf mehrere Festplatten erfolgt. Dies gilt nicht für das RAID-0-System, denn dieser Modus verteilt eine ankommende Datei auf unterschiedliche Datenträger im Verbund. Dieses Vorgehen ermöglicht höhere Geschwindigkeiten bei dem Zugriff auf die Dateien, bei einem Ausfall eines Speichermediums sind die Daten allerdings verloren.

Die Schreibgeschwindigkeit bei einem RAID 5, gegenüber zum Beispiel einem RAID 0, ist niedriger, da wegen der Anlegung der Paritätsdaten, die für die Wiederherstellung erforderlich sind, noch zusätzliche Eingriffe auf den Speicherplatz stattfinden. [19] Der Lesezugriff dagegen ist davon nicht betroffen. Der RAID-5-Verbund übersteht einen Ausfall einer der Festplatten, ohne die Daten zu verlieren. Die Ersatzfestplatte wird im Verbund durch die Paritätsdaten, welche bei der Speicherung der Daten im RAID auf alle vorhandenen Datenträgern verteilt werden, auf den Stand der anderen gebracht. Der Ausfall von zwei Datenträgern, bedeutet einen Datenverlust und eine Wiederherstellung des Verbundes ist in so einem Fall nicht möglich. [19]

Der verfügbare Platz für die Nutzdaten in einem RAID 5 mit drei Festplatten, wird durch die folgende Formel errechnet:

$$(3 - 1) * (Festplatte\ mit\ dem\ kleinsten\ Speichervolumen) = Nutzdaten$$

Im Idealfall sollte ein RAID aus Speichermedien der gleichen Größe bestehen. So würde bei drei Festplatten zu jeweils 100 GB, der Anteil an Nutzdaten 200 GB betragen. Die restlichen 100 GB sind dann für die Parität reserviert, die für die Wiederherstellung benötigt wird. [20]

2 Installation des Betriebssystems

Dieses Kapitel befasst sich mit der Installation und Konfiguration des Betriebssystems Raspbian auf dem Raspberry Pi 3 Model B.

2.1 Raspbian

Das für diese Bachelorarbeit gewählte Betriebssystem Raspbian von der Raspberry Pi Foundation, basiert auf der Linux-Distribution Debian. Das Raspbian ist für den Raspberry Pi optimiertes Betriebssystem und hat zusätzlich eine grafische Oberfläche. Da es auf Debian basiert, steht auch das Paketverwaltungswerkzeug Advanced Packaging Tool (APT) zur Verfügung. Mit diesem Werkzeug hat der Benutzer die Möglichkeit Debian-Pakete unter anderem zu installieren oder zu deinstallieren.

2.2 Installation von Raspbian

Das erste Unterkapitel zeigt die Vorgehensweise der Abbildspeicherung auf eine Micro-SD-Speicherkarte, wenn als Betriebssystem das Microsoft Windows 10 zur Verfügung steht. Der zweite Abschnitt beschreibt die Installationsschritte auf dem Raspberry Pi 3 Model B.

2.2.1 Vorbereitung mit Windows 10

Die Abbilddatei mit Raspbian wird von der offiziellen Internetseite der Raspberry Pi Foundation heruntergeladen. Es ist eine Zip-Datei mit einer Größe von rund 1,4 GB. Die Abbilddatei soll in einem beliebigen Speicherort auf dem Rechner aus der heruntergeladenen Zip-Datei entpackt werden.

Für die Installation des Betriebssystems, müssen eine leere Micro-SD-Speicherkarte, ein Kartenlesegerät und, wenn erforderlich, ein Speicherkartenadapter von Micro-SD auf SD, vorhanden sein. Um das Raspbian auf die Speicherkarte zu bringen, ist ein Programm für das Brennen der Abbilddateien notwendig. Das Programm „Win32 Disk Imager“ kann diese Aufgabe auf dem

Windows-10-System übernehmen. Dieses steht kostenlos zur Verfügung und kann auf der Herstellerseite heruntergeladen werden.

Es folgt eine Anleitung zur Vorbereitung des Raspbian-Abbildes.

1. Die Micro-SD-Speicherkarte kommt in das Kartenlesegerät. Die Auflistung aller Speichermedien in einem Rechner, ist im Windowsexplorer unter Dieser PC → Geräte und Laufwerke zu finden (Abbildung 3). Die Speicherkarte ist als Laufwerk G gekennzeichnet.

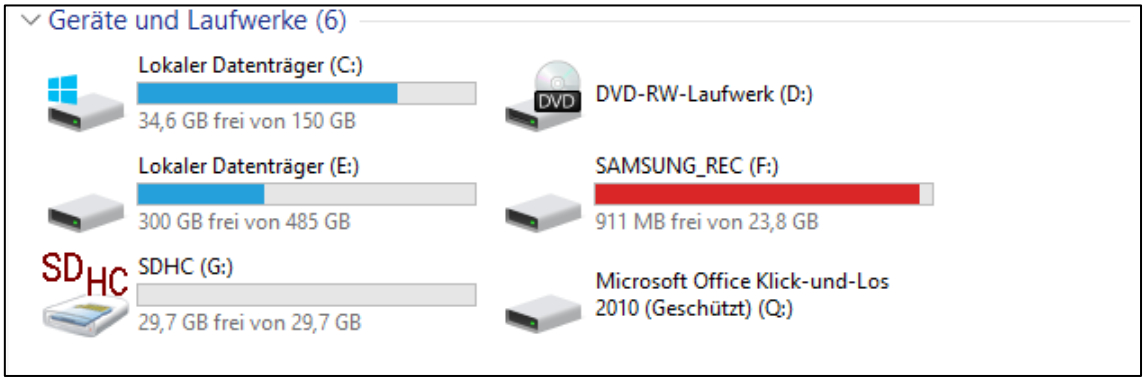

Abbildung 3: Bildschirmfoto: Windows Geräte und Laufwerke

2. Das Programm Win32 Disk Imager als Administrator starten (Abbildung 4). Es erkennt selbstständig unter welchem Laufwerksbuchstaben sich die SD-Speicherkarte befindet. In diesem Fall ist es der Buchstabe G, im Feld „Device“ zu sehen.

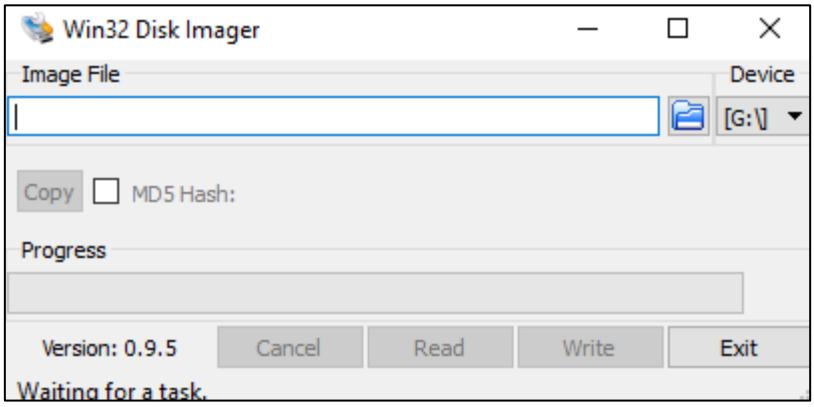

Abbildung 4: Bildschirmfoto: Win32 Disk Imager Oberfläche

3. Die Schaltfläche mit dem Ordnersymbol, erlaubt die Navigation zum Speicherort des Raspbian-Abbildes (Abbildung 5). Das Betätigen der Schaltfläche „Write“ startet das Schreiben des Abbildes auf die Speicherkarte.

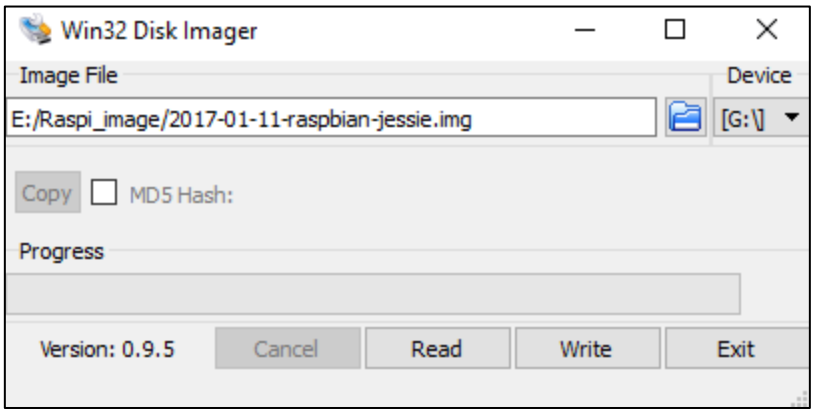

Abbildung 5: Bildschirmfoto: Win32 Disk Imager Abbild auswählen

4. Nach der Bestätigung der Meldung, über die Gefahren von Defekten an der Speicherkarte, die beim Schreiben des Abbildes auftreten können, startet der Schreibprozess des Abbildes auf die Speicherkarte (Abbildung 6)

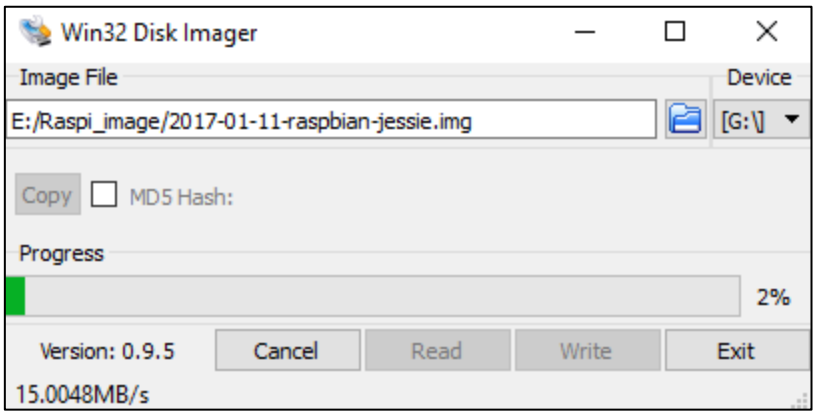

Abbildung 6: Bildschirmfoto: Win32 Disk Imager Abbild wird geschrieben

5. Wenn der Schreibvorgang erfolgreich war, erfolgt darüber eine Informationsmeldung. Die Entnahme der Micro-SD-Speicherkarte aus dem Kartenlesegerät, ist der letzte Schritt in der Vorbereitung des Abbildes für die Installation auf dem Raspberry Pi 3 Model B.

2.2.2 Installation auf dem Raspberry Pi 3 Model B

Das Einlegen der Micro-SD-Speicherkarte in den Kartenschlitz des Raspberry Pi, soll am einen stromlosen Gerät stattfinden. Neben der Speicherkarte und dem Raspberry Pi 3, ist ein Netzteil, das 5,1 Volt Spannung und 2,5 Ampere schafft, die Voraussetzung für den Betrieb des Gerätes. Für die erste Inbetriebnahme und die Erstkonfiguration des Gerätes, müssen zusätzlich eine USB-Tastatur, eine USB-Maus, ein HDMI-Kabel, ein Monitor und ein Netzwerkkabel vorhanden sein.

Wenn alles angeschlossen ist, das Netzteil als letztes, startet der Bootvorgang des Systems. Danach erscheint die grafische Oberfläche des Raspbian Betriebssystems (Abbildung 7).

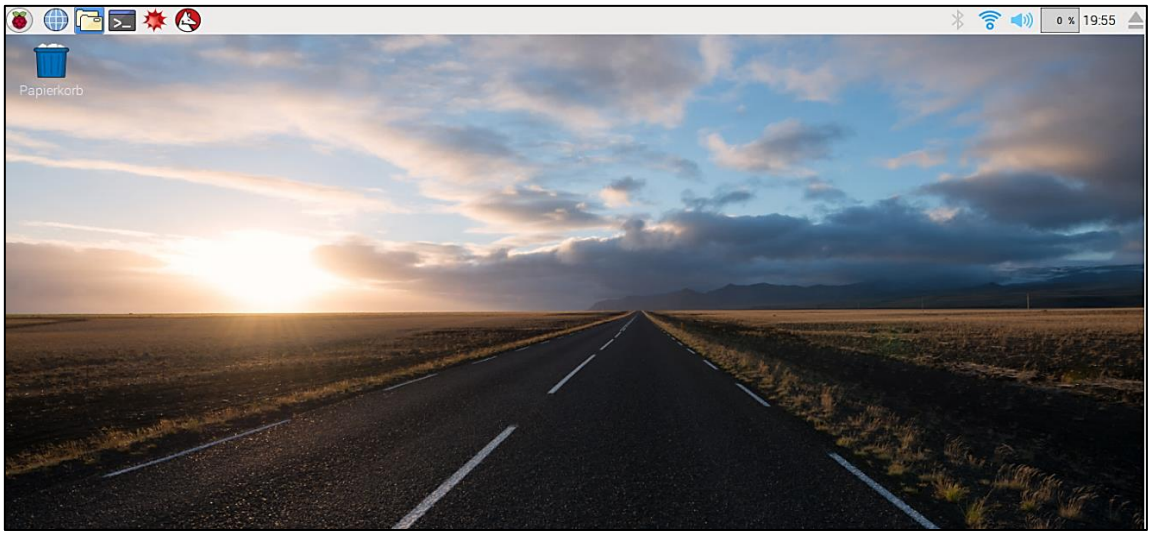

Abbildung 7: Bildschirmfoto: Raspbian, grafische Oberfläche

Als erstes kann der Nutzer das Tastaturlayout ändern, um Tippfehler zu vermeiden, da standardmäßig das englische Tastaturlayout eingestellt ist. Durch das Klicken des Himbeersymbols oben links auf dem Desktop, öffnen sich die Einstellmöglichkeiten für den Raspberry Pi. Unter Preferences → Mouse and Keyboard Settings → Keyboard → Keyboardlayout findet die Auswahl des Tastaturlayouts statt. Die Änderung der Lokalisation ist unter Preferences → Raspberry Pi Configuration → Localisation zu finden. Die Auswahl der Zeitzone und der WiFi Zone erfolgt ebenso unter diesem Pfad. Zusätzlich hat der Benutzer in den Raspberry-Einstellungen die Möglichkeit, den Computerna-

men zu ändern, der im Auslieferungsabbild „raspberry“ heißt. Um die Einstellungen zu übernehmen, muss der Raspberry Pi noch einmal neustarten.

Diese Änderungen kann der Benutzer auch mit dem Terminal vornehmen. Nach dem Ausführen des Befehls `sudo raspi-config`, öffnet sich das Raspberry Pi Kofigurationswerkzeug (Abbildung 8).

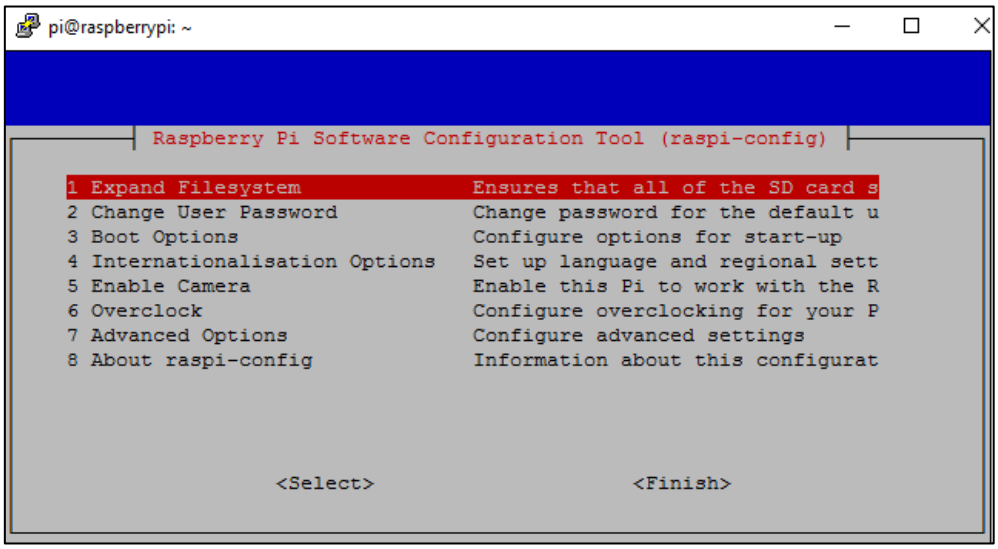

Abbildung 8: Bildschirmfoto: raspi-config

Das Einschalten der WLAN-Funktion erfolgt durch die Betätigung des Netzwerksymbols oben rechts auf der grafischen Oberfläche. Mit den richtigen Zugangsdaten des ausgewählten Anschlusses, stellt der Raspberry Pi die Internetverbindung her.

Ebenso ist es möglich das standardmäßig eingeschaltete Bluetooth durch das Auswählen des Bluetoothsymbols auszuschalten.

Der Befehl `sudo apt-get update`, aktualisiert die Liste, der zu Verfügung stehenden Installationspakete.

Auch ist es empfehlenswert die Softwarepakete des Raspberry Pi zu aktualisieren. Der Befehl `sudo apt-get update`, aktualisiert die Liste, der zu Verfügung stehenden Installationspakete. Die Aktualisierung der bereits installierten Debian-Pakete, geschieht mit dem Befehl `sudo apt-get upgrade`.

2.3 Fernzugriff

Dieser Abschnitt erläutert zwei Möglichkeiten des Fernzugriffes unter Microsoft Windows 10 auf den Raspberry Pi 3 Model B. Die Beschreibung bezieht sich auf den Fernzugriff in einem Heimnetzwerk, das heißt alle Geräte sind in einem WLAN-Netz eingebunden.

2.3.1 SSH

Eine weitere Einstellung, die auf dem Raspberry Pi 3 Model B aktiviert werden kann, ist das SSH. Diese Funktion erlaubt dem Nutzer Raspberry Pi von anderen Rechnern aus fernzusteuern.

SSH ist ein Netzwerkprotokoll, das eine verschlüsselte Verbindung zwischen zwei Rechnern herstellt. Das Protokoll verschlüsselt die Datenübertragung und authentifiziert die Gegenstelle, somit findet keine Manipulation der übertragenen Daten statt. [21]

Seit November 2016 ist diese Option zur Sicherheit der Benutzer auf den Abbilddateien ausgeschaltet. Das beschloss die Foundation, da viele bei den Standardanmeldedaten des Raspberry Pi blieben und so einen unbefugten Zugriff auf den Minirechner ermöglicht haben. [22] Wenn der Nutzer die SSH-Funktion aktiviert, die Zugangsdaten aber nicht ändert, meldet sich das System beim Anmelden in der Kommandozeile, mit dem Hinweis zur Passwortänderung. Die Änderung erfolgt mit dem Befehl `passwd` im Terminalfenster des Raspberry Pi. Zuerst verlangt das System die Eingabe des aktuellen Passwortes, danach erfolgt die erste Eingabe des neuen Passwortes. Die erneute Eingabe des neuen Passwortes im letztem Schritt, dient der Vermeidung von Tippfehlern bei der Erstellung. Stimmt die zweite Eingabe mit der ersten überein, erhält der Benutzer die Meldung über die erfolgreiche Änderung des Passwortes.

2.3.2 Verbindung mit PuTTY

Benutzer, die nicht auf die grafische Oberfläche des Raspbians angewiesen sind und mit dem Terminal arbeiten, können unter Windows mit dem Programm PuTTY auf den Raspberry Pi zugreifen.

PuTTY ist ein kostenloses Programm, das eine SSH-Verbindung zu dem Raspberry Pi herstellen kann. Für die Verbindung durch PuTTY benötigt der Benutzer die IP-Adresse, den Anmeldenamen und das Passwort des Raspberry Pi. Um die IP-Adresse herauszufinden, ist eine Ausführung des Befehls *ifconfig* im Terminalfenster des Raspberry Pi nötig. Wie in der Abbildung 9 zu sehen, listet die Ausgabe die IP-Adressen des Raspberry Pi auf. Das Gerät ist mit Ethernet (kabelgebundenes Netzwerk) eth0 über die Adresse 192.168.0.87 und mit WLAN wlan0 über 192.168.0.80 verbunden.

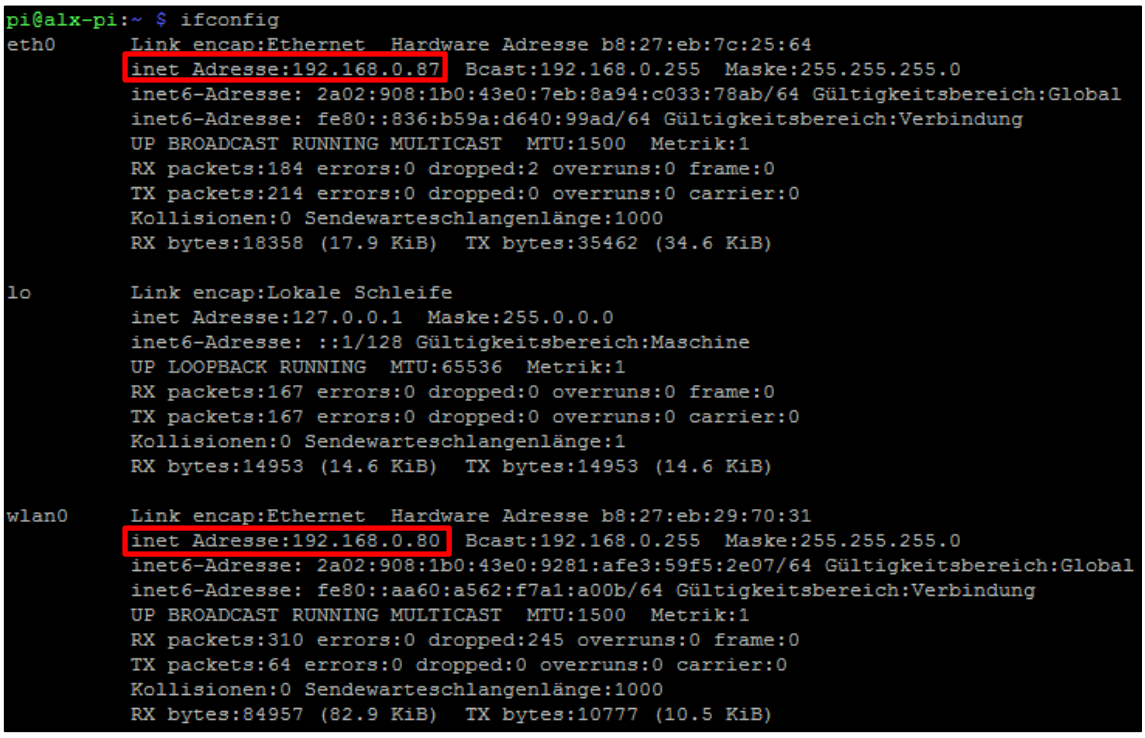

Abbildung 9: Bildschirmfoto: Befehl ifconfig

Die IP-Adresse des Raspberry Pi kommt in das Feld „Host Name (or IP address)“ rein. Der Port ist standardmäßig auf 22 eingestellt und bedarf keiner Änderung (Abbildung 10).

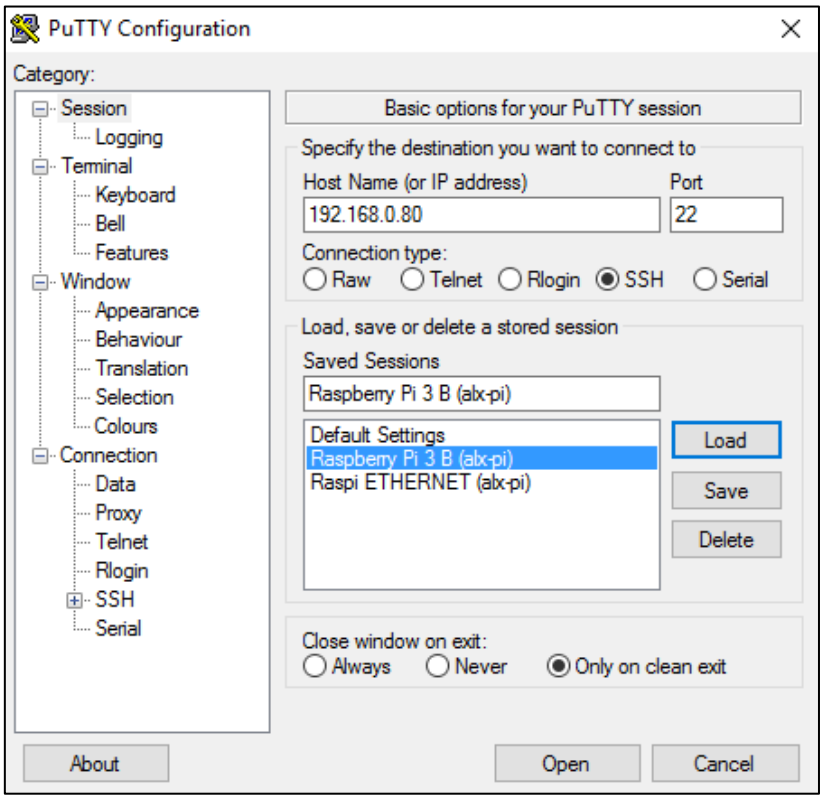

Abbildung 10: Bildschirmfoto: PuTTY

Nach dem Eintragen der IP-Adresse und dem Bestätigen der „Open“-Schaltfläche, ist die Verbindung hergestellt. Im nächsten Schritt erfolgt die Eingabe der Benutzerdaten. Stimmen die Daten überein, so öffnet sich das Terminal von Raspberry Pi (Abbildung 11) und der Nutzer hat den vollen Zugriff auf das System, allerdings ohne die grafische Oberfläche.

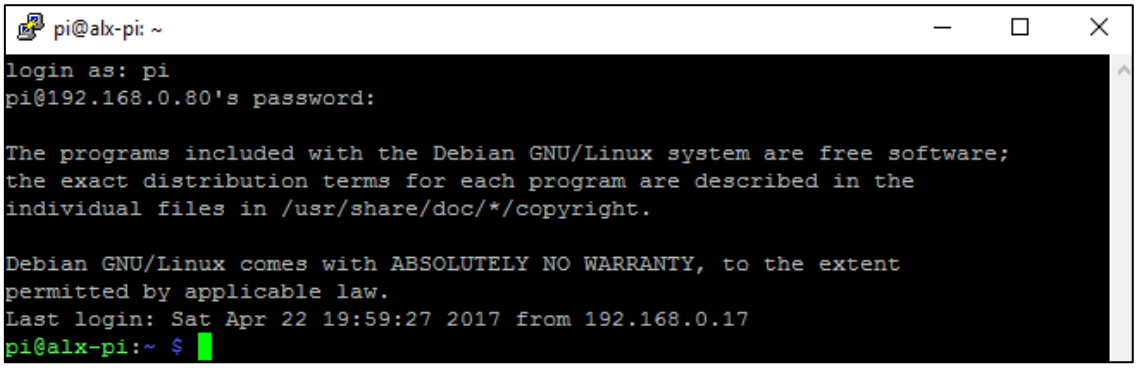

Abbildung 11: Bildschirmfoto: PuTTY, erfolgreiche Anmeldung

2.3.3 Verbindung mit Remotedesktop

Falls der Benutzer auf die grafische Oberfläche nicht verzichten kann oder möchte, dann besteht die Möglichkeit unter Windows 10 das von Haus aus installierte Werkzeug Remotedesktopverbindung zu benutzen. Dies setzt die Installation des Programms XRDP auf dem Raspberry Pi 3 Model B voraus. Da das Programm XRDP mit dem auf dem Raspberry Pi vorinstallierten Programm RealVNC Server in Konflikt stehen kann, muss der Nutzer dieses vor der Installation von XRDP deinstallieren. [23] Die Deinstallation erfolgt mit dem Befehl `sudo apt-get purge realvnc-vnc-server`. Der Befehl `sudo apt-get install xrdp`, im Terminalfenster des Raspberry Pi, installiert das Programm XRDP. Nach dem erfolgreichen Installieren und einem Neustart des Betriebssystems, steht das Programm zur Verfügung. Jetzt kann der Benutzer das Windowswerkzeug Remotedesktopverbindung unter Windows 10 aufrufen und sich damit mit dem Raspberry Pi verbinden.

Folgende Schritte beschreiben den Sitzungsaufbau zwischen einem Windowsrechner und einem Raspberry Pi 3 Model B.

1. Das Programm ist auf dem Windowsrechner unter Start → Windows Zubehör zu finden. Zuvor muss der Benutzer, wie im Punkt 2.3.2 beschrieben, die IP-Adresse des Raspberry Pi herausfinden. In die Zeile „Computer:“ kommt die IP-Adresse rein und mit dem Betätigen der „Verbinden“-Schaltfläche startet der Aufbau der Sitzung zwischen den zwei gewünschten Rechnern (Abbildung 12).

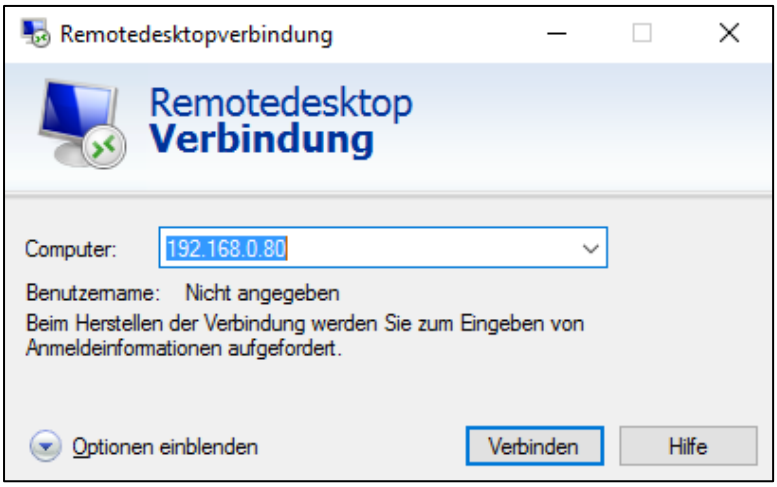

Abbildung 12: Bildschirmfoto: Remotedesktopverbindung

2. Im zweiten Schritt findet die Abfrage der Anmeldedaten des Raspberry Pi (Abbildung 13) statt. Sind die Angaben richtig, erscheint die grafische Oberfläche des Raspberry Pi (Abbildung 14) und der Nutzer hat vollen Zugriff auf das Betriebssystem. Der Eintrag in der Zeile „Module“ ist standardmäßig auf „sesman-Xvnc“ eingestellt und benötigt keine Änderung.

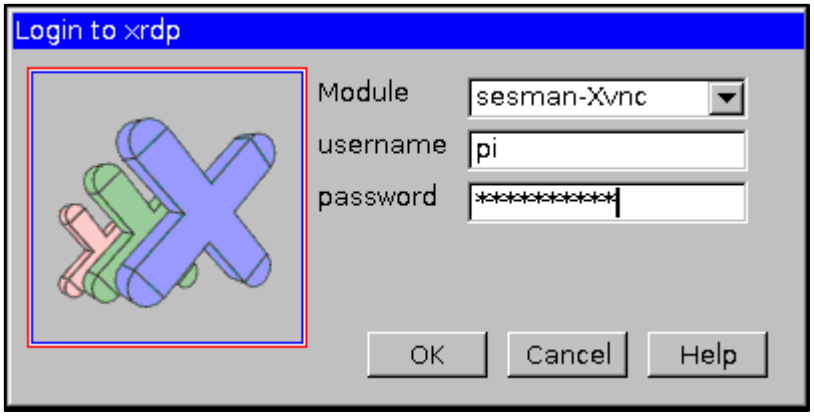

Abbildung 13: Bildschirmfoto: XRDP

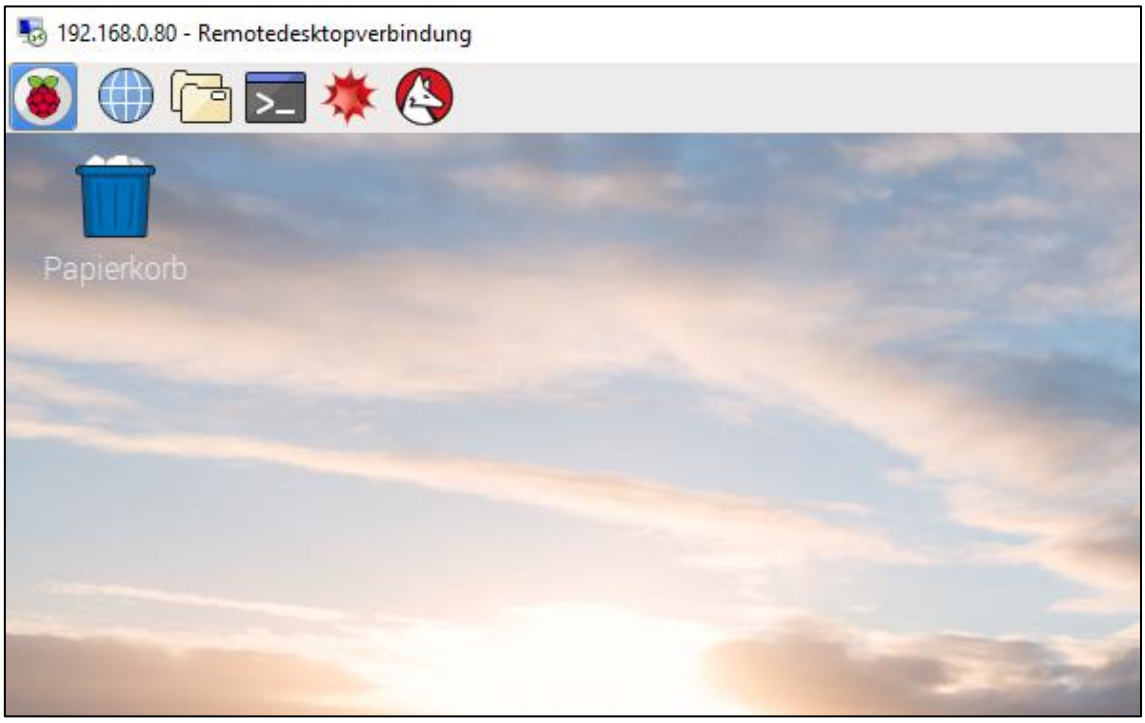

Abbildung 14: Bildschirmfoto: Raspbian auf Remotedesktopverbindung

2.4 Abbild speichern

Es ist empfehlenswert nach allen geänderten Einstellungen das Abbild des Betriebssystems extern zu speichern, um im Falle eines Defektes der Micro-SD-Speicherkarte oder durch eine falsche Konfiguration, nicht alle vorher beschriebene Schritte noch einmal ausführen zu müssen.

Die Speicherung des Abbildes übernimmt das Programm Win32 Disk Imager (siehe Punkt 2.2.1). Durch das Einsetzen der Micro-SD-Speicherkarte in das Kartenlesegerät eines Windowsrechners, erkennt das Programm Win32 Disk Imager automatisch das Laufwerk der Speicherkarte. Im Feld „Image File“ legt der Benutzer den Speicherort und den Namen des Abbildes fest. Der Name der zu speichernden Datei muss eine `.img`-Endung, für Image = Abbild, haben. Nach dem Betätigen der „Read“-Schaltfläche, speichert das Programm die Abbilddatei auf dem Windowsrechner. In diesem Beispiel erfolgt die Speicherung des Abbildes auf dem `Desktop`, unter dem Namen `raspi_geaendert.img` (Abbildung 15).

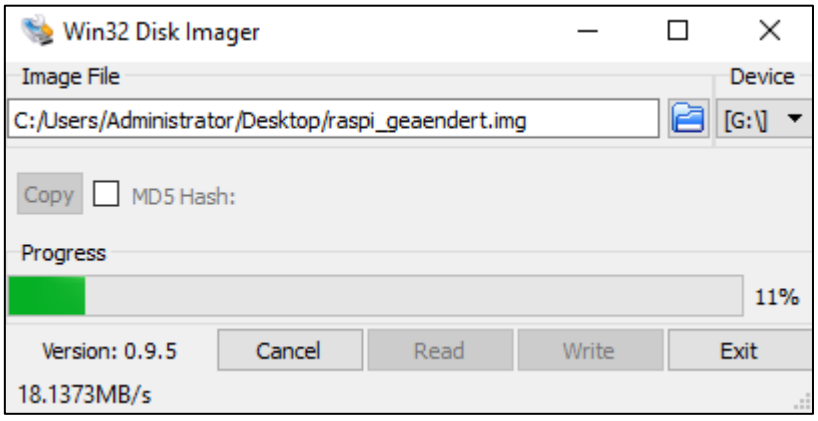

Abbildung 15: Bildschirmfoto: Win32 Disk Imager Abbildspeicherung

3 ZFS auf Raspberry Pi 3 Model B

Dieses Kapitel dient als Anleitung zur Installation und Konfiguration von ZFS auf einem Raspberry Pi 3 Model B mit dem Betriebssystem Raspbian und einem Linux-Kernel in der Version 4.4.50-v7+. Die Installationsschritte beziehen sich zum Teil auf die Anleitung, die auf der Internetseite http://qiita.com vom Benutzer mt08 [24] veröffentlicht wurde.

3.1 Vorbereitung zur Installation von ZFS

Die Befehle *sudo apt-get update* und *sudo apt-get upgrade* aktualisieren die Paketlisten und die bereits installierten Pakete des Raspberry Pi (vergleiche Punkt 2.2.2). Die Kommandos *uname -a* und *hostnamectl* zeigen die Systeminformationen des Raspberry Pi an.

```
pi@alx-pi:~ $ uname -a

Linux raspberrypi 4.4.50-v7+ #970 SMP Mon Feb 20 19:18:29 GMT
2017 armv7l GNU/Linux

pi@alx-pi:~ $ hostnamectl

Static hostname: alx-pi

Icon name: computer

Chassis: n/a

Machine ID: 2e7311d7868244978f314762db251c8e

Boot ID: 0610df70b7c8411fb855617da5417d31

Operating System: Raspbian GNU/Linux 8 (jessie)

Kernel: Linux 4.4.50-v7+

Architecture: arm
```

Wie die Ausgabe der Befehle zeigt, handelt es sich um einen Raspberry Pi mit einem Linux-Kernel in der Version 4.4.50-v7+ und einem Prozessor der arm-Architektur – armv7l. Als Betriebssystem ist Raspbian, in der Version Jessie, installiert.

3.1.1 Benötigte Pakete

Folgende Pakete muss der Benutzer vor der eigentlichen Installation des ZFS, mit Hilfe von *sudo apt-get install*, auf dem Raspberry Pi 3 Model B installieren:

```
pi@alx-pi:~ $ sudo apt-get install libtool \
                          autoconf build-essential
pi@alx-pi:~ $ sudo apt-get install gawk alien fakeroot \
                          gdebi
pi@alx-pi:~ $ sudo apt-get install uuid-dev zlib1g-dev \
                          libblkid-dev libattr1-dev
pi@alx-pi:~ $ sudo apt-get install libselinux-dev ksh \
                         wget libudev-dev parted lsscsi
pi@alx-pi:~ $ sudo apt-get install \
                          raspberrypi-kernel-headers
pi@alx-pi:~ $ sudo apt-get install libdevmapper*
```

Für die Erzeugung der Debian-Pakete ist das Paket *build-essential* zuständig. Es beinhaltet unter anderem Compiler für C und C++ Programmiersprachen. [25]

Das nächste Paket ist *autoconf*. Es dient zum Schreiben oder zum Verändern eigener oder fremder Programme. [26]

Das generische Skript *libtool* unterstützt das Erstellen von statischen Bibliotheken, zum Beispiel C- oder C++-Bibliotheken. [27]

Als nächstes wird das Paket *gawk* installiert. „Ein Programm, das Sie zur Auswahl bestimmter Sätze in einer Datei und zur Durchführung von Veränderungen an diesen nutzen können“. [28]

Das Paket *alien* erlaubt zum Beispiel Red-Hat oder Slackware-Pakete in Debian-Pakete zu konvertieren und diese zu installieren. [29]

fakeroot ist ein Werkzeug zur Simulation von Root-Privilegien. [30]

Mit dem Paket *gdebi* werden lokale Debian-Pakete installiert, anders als APT, das nicht lokale Pakete installiert. [31]

zlib1g-dev ist eine Bibliothek, die das Komprimieren der Dateien ermöglicht. Das Programm *gzip* benutzt diese Bibliothek. [32]

uuid-dev ist eine Entwicklungsumgebung für die *uuid* Bibliothek. UUID ist eine eindeutige Kennung, die 128-Bit lang ist und für verschiedene Verwendungszwecke eingesetzt werden kann. [33]

libattr1-dev enthält die Bibliotheken und Header-Dateien, die benötigt werden, um Programme zu entwickeln, die erweiterte Attribute nutzen. [34]

libblkid-dev ist eine Bibliothek für Programme, die Blockgeräte anhand der UUID erkennen. [35]

Die Bibliothek *libselinux-dev* stellt eine API für SELinux-Anwendungen zur Verfügung. SELinux (Security-Enhaced Linux = sicherheitsverbessertes Linux) wird von der amerikanischen Geheimdienstagentur NSA und dem Linux-Distributor Red Hat entwickelt und soll den Nutzern und Administratoren mehr Kontrolle über die Zugriffskontrolle ermöglichen. [36]

ksh ist ein UNIX-Kommandointerpreter für Shell-Skripte und interaktive Skripte. [37]

Mit *wget* werden Dateien aus dem Internet herunterladen. Die Syntax des Befehls ist *wget* http://www.<Internetadresse der herunterzuladenden Datei>

Das Paket *libudev-dev* ist eine Bibliothek, die für das Paket *libudev* die erforderlichen Dateien für die Anwendungsentwicklung parat hält. [38]

Für die Partitionierung der Festplatten wird das Paket *parted* benötigt. Außerdem erkennt es das Dateisystem ZFS, kann die Partitionen dort auch erstellen oder ändern. [39]

Das Paket *lsscsi* kann alle SCSI-Geräte auflisten, die mit dem System verbunden sind. [40]

Wenn ein Kernel-Modul kompiliert werden soll, wird das Paket *raspberry-pi-kernel-headers* benötigt. Die Header bieten die erforderlichen Funktionsdefinitionen für das Kompilieren des Codes. [41]

libdevmapper ist ein Kernel-Treiber, der sich um die Zuordnung von Datenträgern kümmert. Dies ist für die Raid-Software oder für Treiber wichtig, die die virtuellen Festplatten erstellen. [42]

3.2 Installation von ZFS

Nach der Installation der benötigten Pakete, erfolgt die Installation von ZFS.

Als erstes wird ein neuer Ordner mit dem Namen *zfs* in dem Verzeichnis *home/pi* auf dem Raspberry Pi, erstellt. Mit dem Befehl *cd zfs*, gelangt der Benutzer in den soeben erstellten Ordner.

```
pi@alx-pi:~ $ mkdir -p zfs
pi@alx-pi:~ $ cd zfs
```

Da die ZFS-Installationspakete für Raspberry Pi nicht in den APT-Paketen vorhanden sind, erfolgt die Installation direkt von der Seite www.github.com. Das Herunterladen in das *zfs*-Verzeichnis löst der Benutzer mit dem Kommando *git clone* aus.

```
pi@alx-pi:~/zfs $ git clone https://github.com/zfsonlinux/zfs
```

Das zweite Paket ist SPL (Solaris Porting Layer). Es ist ein Linux-Kernel-Modul, das Solaris APIs für die ZFS Kompatibilität implementiert. [43] Eine API ist eine Anwendungsprogrammierschnittstelle, die eine Anbindung an die Software anderer Programme erlaubt. [44]

```
pi@alx-pi:~/zfs $ git clone https://github.com/zfsonlinux/spl
```

Im Verzeichnis *home/pi/zfs* sind nach dem Herunterladen der Dateien, zwei neue Ordner entstanden: *zfs* und *spl.* Mit *cd spl* gelangt man in das *spl*-Verzeichnis und führt die Skript-Datei *autogen.sh* aus:

```
pi@alx-pi:~/zfs/spl $ ./autogen.sh
```

Ist das Script erfolgreich durchgelaufen, so kann man das *spl*-Verzeichnis verlassen und in das *zfs*-Verzeichnis, im Ordner *zfs*, wechseln. Hier führt man ebenso die Skript-Datei *autogen.sh*, die im *zfs*-Verzeichnis liegt, aus. Nachdem auch dieses Skript durchgelaufen ist, ist es nötig sich zurück in das *spl*-Verzeichnis zu begeben, um weitere Installationsschritte auszuführen.

Der Benutzer muss drei Befehle im Ordner *spl* nacheinander, in der weiter unten gezeigten Reihenfolge, ausführen. Zuerst muss das Skript *configure* ausgeführt werden, das zuvor von *autogen.sh* erstellt wurde. Dieses prüft, ob das zu installierende Programm mit dem System kompatibel ist. [45] Das Ausführen des Befehls *sudo make* bewirkt die Kompilation des Programms und kann einige Zeit in Anspruch nehmen. Den Prozess kann der Nutzer beschleunigen, indem er zusätzlich zum Befehl *make*, die zur Verfügung stehenden Prozessor-Kerne hinzufügt (vier bei Raspberry Pi 3 Model B): *sudo make -j4.*Das Kommando *sudo make install* installiert das Programm.

```
pi@alx-pi:~/zfs/spl $ ./configure
pi@alx-pi:~/zfs/spl $ sudo make
pi@alx-pi:~/zfs/spl $ sudo make install
```

Sind alle Schritte erfolgreich abgelaufen, müssen auch im *zfs*-Verzeichnis diese drei Installationsanweisungen in derselben Reihenfolge folgen.

```
pi@alx-pi:~/zfs/spl $ cd ..
pi@alx-pi:~/zfs $ cd zfs
pi@alx-pi:~/zfs/zfs $ ./configure
pi@alx-pi:~/zfs/zfs $ sudo make
pi@alx-pi:~/zfs/zfs $ sudo make install
```

Die Mitteilung über neue Bibliotheken an die interne Verwaltung der Bibliotheken, erfolgt mit dem Befehl *sudo ldconfig.*

```
pi@alx-pi:~ $ sudo ldconfig
```

Dieser Aufruf bewirkt die Erstellung der symbolischen Links auf Bibliotheken in dem Bibliotheken-Zwischenspeicher. [46] Die ZFS-Bibliotheken liegen im Verzeichnis */usr/local/lib.*

Die ZFS-Module sind installiert und müssen in der Datei */lib/modules/4.4.50-v7+/modules.dep* hinterlegt werden. Die folgende Anweisung erledigt dies:

```
pi@alx-pi:~ $ sudo depmod -a
```

Zuletzt erfolgt die Ladung der ZFS-Module:

```
pi@alx-pi:~ $ sudo modprobe zfs
```

Die Module sind geladen und das Dateisystem ZFS ist erfolgreich auf dem Raspberry Pi 3 Model B installiert. Das zeigt die folgende Ausgabe des Kommandos *dmesg -T | grep ZFS.* Das Programm *dmesg* (display message = zeig Meldungen) zeigt Meldungen des Kernelpuffers an, *grep* durchsucht die Meldungen nach den, durch den Benutzer, vorgegebenen Kriterien der Datei. Die Ausgabe zeigt dank *grep,* nur die gesuchte Meldung zu *ZFS*. Der Zusatz -T gibt die Startzeit des Ereignisses aus.

```
pi@alx-pi:~ $ dmesg -T | grep ZFS

[Mo April 10 03:56:15 2017] ZFS: Loaded module v0.7.0-
rc3_228_g692e55b, ZFS pool version 5000, ZFS filesystem
version 5
```

3.3 Konfiguration von ZFS

Nach der Installation von ZFS ist es nötig, weitere Konfigurationsschritte auf dem Raspberry Pi 3 Model B vorzunehmen.

3.3.1 Automatisches Hochladen der Module

In der Anfangskonfiguration von ZFS, wenn man nach der im Punkt 3.2 aufgezeigter Installationsmethoden vorgeht, erfolgt keine Ladung der ZFS-Module bei einem Neustart des Betriebssystems. Dies zeigt auch die folgende Befehlsausgabe nach einem Neustart. *zpool list* listet bereits erstellte ZFS-Pools auf.

```
pi@alx-pi:~ $ zpool list
The ZFS modules are not loaded.
Try running '/sbin/modprobe zfs' as root to load them.
```

Die Module, die mit dem Kommando *sudo modprobe zfs* geladen werden, wie es die Ausgabe vorschlägt, sind nur bis zum nächsten Herunterfahren des Systems geladen. Damit die Module und bereits erstellte Pools bei dem Hochfahren des Raspberry Pi automatisch laden, muss der Benutzer im System- und Sitzungsmanager *systemd* weitere Einstellungen, mit der Hilfe vom Kommandozeilenwerkzeug *systemctl*, vornehmen. Der Befehl *systemctl list-unit-files | grep zfs* listet die ZFS-Dienste auf. Diese sind in dieser Standardkonfiguration ausgeschaltet (disabled).

```
pi@alx-pi:~ $ systemctl list-unit-files | grep zfs
zfs-import-cache.service                disabled
zfs-import-scan.service                 disabled
zfs-mount.service                       disabled
zfs-share.service                       disabled
zfs-zed.service                         disabled
zfs.target                              disabled
```

Der Dienst *zfs-import-cache.service* wird benötigt, um nach den erstellten Pools im Zwischenspeicher, dem *zpool.cache*, zu suchen und diese zur Verfügung zu stellen. *zfs-import-scan.service* durchsucht die Geräte, zum Beispiel USB-Speichermedien, nach vorhandenen Pools. Um die Dateisysteme automatisch, bei ihrer Erstellung einzuhängen, muss der Dienst *zfs-mount.service* aktiv sein. Der nächste ZFS-Dienst, *zfs-share.service*, soll eingeschaltet werden, wenn die Dateifreigabe erwünscht ist. *zfs-zed.service*: ZED (ZFS Event Daemon) überwacht die vom ZFS-Kernelmodul erzeugten Ereignisse. Wird ein ZFS-Ereignis erzeugt, so führt ZED alle Verknüpfungen aus, die für diese Klasse aktiviert wurden.[4] Damit das ZFS-System beim Systemstart mitgeladen wird, ist es notwendig, den Dienst *zfs.target* zu aktivieren.

Das Aktivieren des jeweiligen Dienstes erfolgt mit dem Befehl *sudo systemctl enable <Dienst>*.

```
pi@alx-pi:~ $ sudo systemctl enable zfs-zed.service
```

Nach der Aktivierung aller benötigten Dienste, stehen die ZFS-Module und die bereits erstellten Pools dem Benutzer, auch nach dem Neustart des Betriebssystems, zur Verfügung. Die Ausgabe von *dmesg | grep ZFS* zeigt, dass die Ladung der Module bei rund sieben Sekunden, in der Phase des Hochfahrens des Betriebssystems, stattfand.

```
pi@alx-pi:~ $ dmesg | grep ZFS

[ 7.359307] ZFS: Loaded module v0.7.0-rc3_228_g692e55b,
ZFS pool version 5000, ZFS filesystem version 5
```

Das Kommando *zpool list* zeigt nun das vorhandene Pool *Test_1* an.

```
pi@alx-pi:~ $ zpool list

NAME   SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP
HEALTH  ALTROOT

Test_1  80M    98K  79,9M            -      5%       0%
1.00x  ONLINE  -
```

[4] Kommando *man zed* im Terminal, das Handbuch für zed

3.3.2 Speicherort der Pools

Bei der Erstellung und Benutzung der Pools nach dem Stand der Konfiguration aus Punkt 3.3.1, kann noch ein Fehler auftreten - es können die bereits erstellten Pools nach dem Neustart nicht auffindbar sein.

```
pi@alx-pi:~ $ zpool list
no pools available
```

Beim Erstellen eines neuen Pools, muss ein Eintrag in die Datei *zpool.cache* von ZFS automatisch erfolgen. Dies geschieht hier aber nicht, so dass die Pools nur bis zum Ausschalten des Raspberry Pi vorhanden sind. Der Benutzer kann den Pfad nach der Erstellung des neuen Pools manuell setzen und erstellt somit den benötigten Eintrag.

```
pi@alx-pi:~ $ zpool set cachefile=/usr/local/etc/zfs
/zpool.cache <Name des Pools>
```

Das Problem bei dieser Methode ist, dass der Benutzer nicht vergessen darf, dies auch nach jeder Änderung im Pool auszuführen, sonst ist dieser nach einem Neustart des Raspberry Pi verschwunden.

Damit eine automatische Speicherung der Einträge stattfindet, müssen weitere Einstellungsschritte erfolgen. Nach mehreren Tests und Konfigurationsversuchen, ergab sich eine Lösung, die weiter im Abschnitt erläutert wird.

Das ZFS-System unter Ubuntu 16.10, speichert automatisch die Pools in die Datei *zpool.cache*, unter dem Pfad */etc/zfs*. Der Speicherort auf dem Raspberry Pi, mit der ZFS-Version *v0.7.0-rc3_228_g692e55b*, soll laut Anleitungen aus den Raspberryforen manuell auf */usr/local/etc/zfs/zpool.cache* gesetzt werden. [24] Dieser Pfad ist standardmäßig in allen dafür nötigen Konfigurationsdateien des Raspberry Pi 3 Model B bereits gesetzt, trotzdem fand die automatische Ablegung der Pools dort nicht statt. Die Lösung ist, ein neues Speicherortverzeichnis zu definieren. Der neue Speicherort auf dem Raspberry Pi, ist an den Speicherort des Betriebssystems Ubuntu 16.10 angelehnt. Dafür muss der Benutzer auf dem Raspberry Pi 3 Model B manuell einen neuen, leeren Ordner *zfs* unter */etc* erstellen. Die Erstellung eines neuen Pools erzeugt automatisch die Datei *zpool.cache*.

Damit dies auch geschieht, muss in der Datei */etc/default/zfs* in der Zeile *ZPOOL_CACHE* der neue Speicherpfad eingetragen werden. Der Eintrag *ZPOOL_IMPORT_OPTS* ist für das Importieren der vorhandenen Pools zuständig, die auch im *zpool.cache* gespeichert sind.

```
pi@alx-pi:~ $ sudo nano /etc/default/zfs
(...)
ZPOOL_IMPORT_OPTS="-c /etc/zfs/zpool.cache"
ZPOOL_CACHE="/etc/zfs/zpool.cache"
(...)
```

Eine weitere Änderung muss in der Datei */usr/local/etc/zfs/zfs-functions* erfolgen. In dieser Datei ist ebenfalls der neue Speicherpfad in den Eintrag *ZPOOL_CACHE* zu setzen. Der neue Pfad wird zusätzlich in dem Verzeichnis */usr/lib/systemd/system/zfs-import-cache.service* eingetragen:

```
pi@alx-pi:~ $ sudo nano /usr/lib/systemd/system/zfs-
import-cache.service
(...)
ConditionPathExists=/etc/zfs/zpool.cache
(...)
ExecStart=/usr/local/sbin/zpool import -c
/etc/zfs/zpool.cache -aN
```

Ebenso bedarf die Datei */usr/lib/systemd/system/zfs-import-scan.service* einer Änderung im Eintrag *ConditionPathExists*, die nach dem Setzen neuer Parameter folgendermaßen aussehen soll:

```
pi@alx-pi:~ $ sudo nano /usr/lib/systemd/system/zfs-
import-scan.service
(...)
ConditionPathExists=!/etc/zfs/zpool.cache
(...)
```

Nach diesen Änderungen erfolgen die Einträge der neuen ZFS-Pools automatisch in der Datei /etc/zfs/*zpool.cache*, ebenso die weiteren Modifikationen an diesen. Bei dem Systemneustart findet der Import der Pools automatisch aus der *zpool.cache*-Datei statt.

4 Arbeiten mit ZFS auf Raspbian

Dieses Kapitel zeigt die Erstellung und Bedienung der Pools mit dem Dateisystem ZFS auf dem Raspberry Pi 3 Model B mit dem Betriebssystem Raspbian. Die in weiteren Unterkapiteln vorgestellten Anleitungen und Befehle, lehnen sich an das Oracle Solaris ZFS-Administrationshandbuch [47] und an das Forum wiki.ubuntuusers.de [11] an.

4.1 Erstellung und Verwaltung eines Pools

Dieses Beispiel zeigt die Erstellung eines Versuchspools mit simulierten Festplatten. Diese Vorgehensweise dient dazu, sich mit dem ZFS-Dateisystem vertraut zu machen, ohne zusätzliche Ausgaben für die Speichermedien zu tätigen.

Die Mindestgröße des Speichermediums in ZFS muss 64 MB betragen. In das neuerstellte Verzeichnis */mnt/platten* kommen vier virtuelle Festplatten, zu je 70 MB, rein. Dies geschieht mit dem Befehl *dd*, der die Dateien *platte1*, *platte2*, *platte3*, *platte4* im Ordner *platten* erzeugt. Das Kommando *ls* listet die Inhalte des Verzeichnisses *platten*, in dem sich die soeben erstellten Dateien befinden, auf.

```
pi@alx-pi:~ $ sudo mkdir -p /mnt/platten
pi@alx-pi:~ $ cd /mnt/platten
pi@alx-pi:/mnt/platten $ sudo dd if=/dev/zero
of=/mnt/platten/platte1 bs=1M count=70; sudo dd
if=/dev/zero of=/mnt/platten/platte2 bs=1M count=70; su-
do dd if=/dev/zero of=/mnt/platten/platte3 bs=1M
count=70; sudo dd if=/dev/zero of=/mnt/platten/platte4
bs=1M count=70

pi@alx-pi:/mnt/platten $ ls
platte1  platte2  platte3  platte4
```

Das Erstellen eines Pools geschieht mit dem Befehl *zpool create* und der Angabe des Pfades der eingehängten Speichermedien. Der Name des Pools darf nur mit Buchstaben beginnen.

```
pi@alx-pi:~ $ zpool create <Name> <Einhängepunkt>
```

In diesem Beispiel wird mit den virtuellen Festplatten ein Pool mit dem Namen *testpool* erstellt. Die Festplatten 1 und 3 werden mittels dem Befehl *mirror*, gespiegelt, ebenso die Platten 2 und 4. Das heißt, die gespiegelten Speichermedien werden gleichzeitig mit denselben Daten beschrieben, um so bei einem Ausfall einer der Festplatten keinen Datenverlust zu erleiden.

```
pi@alx-pi:~ $ sudo zpool create testpool mirror
/mnt/platten/platte{1,3} mirror /mnt/platten/platte{2,4}
```

Mit dem Befehl *zpool status* kontrolliert der Benutzer die richtige Erstellung des Pools. Laut der folgender Ausgabe, ist der Pool *testpool* online und jeweils zwei Festplatten spiegeln sich. Außerdem erhält der Benutzer die Information über eventuelle Fehlermeldungen am Ende der Ausgabe.

```
pi@alx-pi:/mnt/platten $ zpool status testpool
pool: testpool
 state: ONLINE
  scan: none requested
config:
        NAME                      STATE     READ WRITE CKSUM
        testpool                  ONLINE       0     0     0
          mirror-0                ONLINE       0     0     0
            /mnt/platten/platte1  ONLINE       0     0     0
            /mnt/platten/platte3  ONLINE       0     0     0
          mirror-1                ONLINE       0     0     0
            /mnt/platten/platte2  ONLINE       0     0     0
            /mnt/platten/platte4  ONLINE       0     0     0
errors: No known data errors
```

Das Einhängen des Pools *testpool* im System erfolgt automatisch, in diesem Fall unter dem Anhängepunkt */testpool*.

```
pi@alx-pi:~ $ mount
(...)
testpool on /testpool type zfs (rw,xattr,noacl)
```

Der Befehl *zpool list* zeigt die Eigenschaften des Pools, wie zum Beispiel den verfügbaren Platz in diesem, an. Dadurch, dass jeweils zwei Festplatten im *testpool* sich spiegeln, zeigt die Ausgabe auch die Größe (SIZE) und den freien Speicherplatz (FREE) von zwei Festplatten an.

```
pi@alx-pi:~ $ zpool list testpool
NAME       SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP
DEDUP  HEALTH  ALTROOT

testpool   128M  96,5K   128M         -     1%     0%
1.00x  ONLINE  -
```

Der Nutzer kann jederzeit zusätzliche Festplatten zu einem Pool hinzufügen, aber, nicht ohne Schwierigkeiten, nachträglich entfernen. Mit dem Kommando *zpool detach* kann die Entfernung, der nicht mehr benötigten Festplatten, erfolgen. Dies ist allerdings nur dann möglich, wenn zusätzlich noch Kopien der betroffenen Festplatte, zum Beispiel in einem gespiegelten Einsatz (Mirror), vorhanden sind. Sind diese redundanten Festplatten nicht vorhanden, so ist eine Entfernung, der nicht mehr benötigten Festplatten aus dem Pool, nicht möglich. [48]

Das folgende Beispiel beschreibt das Entfernen der Festplatten aus einem Pool. Im ersten Schritt wird die *platte4* aus dem *mirror-1* entfernt, im zweiten soll die *platte2* entfernt werden. Das Löschen der Festplatte *platte2* aus dem Pool *testpool* ist nicht gestattet, da diese eine gespiegelte Festplatte der *platte4* ist, die nicht mehr vorhanden ist.

```
pi@alx-pi:~ $ sudo zpool detach testpool /mnt/platten/platte4

pi@alx-pi:~ $ zpool status testpool
(...)
testpool                    ONLINE       0     0     0
 mirror-0                   ONLINE       0     0     0
      /mnt/platten/platte1  ONLINE       0     0     0
      /mnt/platten/platte3  ONLINE       0     0     0
   /mnt/platten/platte2     ONLINE       0     0     0
(...)
pi@alx-pi:~ $ sudo zpool detach testpool /mnt/platten/platte2

cannot detach /mnt/platten/platte2: only applicable to mirror
and replacing vdevs
```

Falls *platte2* trotzdem entfernt werden soll, bleibt nur eine Lösung – die komplette Zerstörung des Pools *(zpool destroy testpool)* und falls nötig, die anschließende Erstellung des Pools mit den Festplatten *platte1* und *platte3*. Zuvor müssen die bereits vorhandenen Daten der verbleibenden Speichermedien extern gesichert werden und nach der Neuerstellung des Pools *testpool* wieder aufgespielt werden. [11] Dies kann bei großen Mengen an Daten viel Zeit in Anspruch nehmen und es besteht die Gefahr des Datenverlustes bei der externen Sicherung oder bei dem Zurückspielen der Daten in den Pool. Falls nicht ausreichend Speicherplatz zur externen Speicherung vorhanden ist, können zur Beschaffung des Platzes zusätzliche Kosten, in Form von Ausgaben für neue Speichermedien, entstehen. Eine richtige Einschätzung und die vorausschauende Planung bei der Erstellung neuer Speicherpools ist deshalb sehr wichtig.

Zur Wiederherstellung des *mirror-1* im Speicherpool *testpool* wird folgender Befehl benötigt:

```
pi@alx-pi:~ $ sudo zpool attach testpool /mnt/platten/platte2
/mnt/platten/platte4
```

Wenn ein Speichermedium im Pool durch ein anderes Speichermedium ersetzt werden soll, so muss die Speicherkapazität gleich oder höher sein. Bei Nichteinhaltung dieser Vorgabe, wird bei dem Vorgang die Meldung ausgegeben, dass das Ersatzmedium nicht getauscht werden kann, da die Speicherkapazität zu klein ist. Für den Versuch wurden im Verzeichnis */mnt/platten* eine Festplatte *platte5* mit 65 MB und eine Festplatte *platte6* mit 75 MB erstellt, die die 70 MB-Große *platte4* ersetzen sollen.

```
pi@alx-pi:/mnt/platten $ sudo dd if=/dev/zero
of=/mnt/platten/platte5 bs=1M count=65; sudo dd
if=/dev/zero of=/mnt/platten/platte5 bs=1M count=75
```

Im folgenden Befehlsfenster wird versucht die *platte4*, durch die kleinere *platte5*, zu ersetzen. Dieses Vorhaben misslingt aufgrund der zu kleinen Speichergröße der neuen Festplatte.

```
pi@alx-pi:~ $ sudo zpool replace testpool
/mnt/platten/platte4 /mnt/platten/platte5

cannot replace /mnt/platten/platte4 with
/mnt/platten/platte5: device is too small
```

Das Ersetzen der *platte4* mit der größeren *platte6* klappt dagegen, wie erwartet. In der Statusausgabe des Pools wird richtigerweise die *platte6*, statt der *platte4*, angezeigt.

```
pi@alx-pi:~ $ sudo zpool replace testpool
/mnt/platten/platte4 /mnt/platten/platte6

pi@alx-pi:~ $ zpool status testpool
pool: testpool

(...)
          mirror-1                  ONLINE       0     0     0

            /mnt/platten/platte2    ONLINE       0     0     0

            /mnt/platten/platte6    ONLINE       0     0     0
(...)
```

4.2 ZFS-Snapshots

Die Schnappschussfunktion (Snapshot) erlaubt dem Benutzer das komplette System oder Dateien auf den Zeitpunkt eines bestimmten Schnappschusses wiederherzustellen (Vergleiche Kapitel 1.2).

Die Snapshots in ZFS können vom Benutzer mit dem Befehlt *zfs snap* angelegt werden.

```
pi@alx-pi:~ $ sudo zfs snap <Pool>@<Name des Snapshots>
```

Die maximale Anzahl möglicher Snapshots im ZFS-System liegt bei 2^{64}. [49] Die erstellten Snapshots werden in dem Pool gespeichert aus dem sie erzeugt wurden und belegen am Anfang keine zusätzliche Kapazität im Pool. Allerdings werden jedes Mal die geänderten Daten im Pool den Speicherplatzbedarf des Snapshots erhöhen, da ZFS Verweise auf die alten Daten speichert. Damit soll das Löschen der Daten verhindert werden. [49]

Die Demonstration des nächsten Beispiels zeigt die Erzeugung eines Snapshots und die Anwendung dieses zur Wiederherstellung des Pools.

Mit der *ls*-Funktion wird der Inhalt des Pools *testpool* angezeigt, der zurzeit keine Dateien beinhaltet. Es wird als erstes der aktuelle Zustand des Pools mit dem Snapshot *testpool@aktuell* gespeichert. *sudo zfs list -t snap* listet alle vorhandene Snapshots auf.

```
pi@alx-pi:/testpool $ ls -l
insgesamt 0

pi@alx-pi:/testpool $ sudo zfs snap testpool@aktuell
pi@alx-pi:/testpool $ sudo zfs list -t snap
NAME                USED  AVAIL  REFER  MOUNTPOINT
testpool@aktuell       0      -    23K  -
```

Im Pool *testpool* wird jetzt eine leere Textdatei *testsnap.txt* erzeugt. Mit sudo nano *testsnap.txt* öffnet sich der Texteditor Nano. Zur Speicherung der Textdatei wird die Tastenkombination *Strg+o* verwendet, danach kann der Dateiname, falls benötigt, geändert werden und mit Enter bestätigt werden. Die Tastenkombination *Strg+x* beendet den Texteditor und die Datei *testsnap.txt* ist im *testpool* erzeugt worden.

```
pi@alx-pi:/testpool $ sudo nano testsnap.txt

pi@alx-pi:/testpool $ ls -l
insgesamt 1
-rw-r--r-- 1 root root 0 Mai 17 23:56 testsnap.txt
```

Es wird ein zusätzlicher Snapshot, mit dem Namen *testpool@neue-datei*, erzeugt und das Vorhandensein in der ZFS-Snapshot-Liste geprüft.

```
pi@alx-pi:/testpool $ sudo zfs snap testpool@neue-datei
pi@alx-pi:/testpool $ sudo zfs list -t snap

NAME                   USED  AVAIL  REFER  MOUNTPOINT

testpool@aktuell        13K      -    23K  -

testpool@neue-datei       0      -    23K  -
```

Um eine Änderung im Pool zu erzielen, wird eine Kopie der Textdatei *testsnap.txt*, mit dem Namen *testsnap2.txt*, erstellt.

```
pi@alx-pi:/testpool $ sudo cp testsnap.txt testsnap2.txt
pi@alx-pi:/testpool $ ls
testsnap2.txt  testsnap.txt
```

Um den Pool auf den Stand des Snapshots *testpool@neue-datei* zurück zu bringen, wird der Befehl *zfs rollback* benötigt. Anschließend ist in dem Pool nur die Datei *testsnap.txt* vorhanden.

```
pi@alx-pi:/testpool $ sudo zfs rollback testpool@neue-
datei

pi@alx-pi:/testpool $ ls
testsnap.txt
```

Falls die Wiederherstellung zum Snapshot *testpool@aktuell* durchgeführt werden soll, so benötigt der Befehl die *-r*-Funktion vor dem Snapshotnamen. Dies bewirkt die Löschung aller, nach dem Snapshot *testpool@aktuell*, erstellter Snapshots.

```
pi@alx-pi:/testpool $ sudo zfs rollback testpool@aktuell

cannot rollback to 'testpool@aktuell': more recent snap-
shots or bookmarks exist
use '-r' to force deletion of the following snapshots
and bookmarks:
testpool@neue-datei
```

Nach dem Hinzufügen der *-r*-Option und dem Ausführen des Befehls, erscheint in der Snapshotliste nur der Snapshot *testpool@aktuell*.

```
pi@alx-pi:/testpool $ sudo zfs rollback -r test-
pool@aktuell

pi@alx-pi:/testpool $ sudo zfs list -t snap

NAME                USED  AVAIL  REFER  MOUNTPOINT

testpool@aktuell       0      -    23K  -
```

Das Löschen eines Snapshots erfolgt mittels *zfs destroy*.

```
pi@alx-pi:/testpool $ sudo zfs destroy testpool@aktuell

pi@alx-pi:/testpool $ sudo zfs list -t snap
no datasets available
```

5 Testen von ZFS

Dieser Abschnitt befasst sich mit dem Testen der Lese- und Schreibgeschwindigkeiten eines RAID-Z1 auf dem Raspberry Pi 3 Model B. Zusätzlich wird ein Ausfall eines Datenträgers simuliert und das Verhalten beobachtet.

5.1 Geschwindigkeit der USB-Speichersticks

Für das Erstellen des Test-RAIDs, dem RAID-Z1 auf dem Raspberry Pi 3 Model B, wurden drei USB-3.0-Speichersticks mit einer Speicherkapazität von 16 GB gewählt, die durch einen externen USB-Hub an den USB-Anschluss angeschlossen werden. Trotz dem Nichtvorhandensein der USB-3.0-Anschlüsse am Raspberry Pi 3 Model B, fiel die Wahl auf diese. Die USB-3.0-Speichersticks sind abwärtskompatibel und arbeiten mit der Geschwindigkeit des USB-2.0-Anschlusses. Aufgrund der Drosselung der USB-3.0-Speichersticks durch die USB-2.0-Anschlüsse, werden diese die höchstmögliche Geschwindigkeit erreichen, die mit USB 2.0 auf dem Raspberry Pi möglich ist.

Ein Vergleich der Lese- und Schreibgeschwindigkeiten von einem USB-3.0-Speicherstick an einem USB-3.0-Anschluß und einem USB-2.0-Anschluß verdeutlicht den Unterschied (Tabelle 2). Der Testdurchlauf erfolgte mit dem Programm *CrystalDiskManager 5* auf einem Rechner mit dem Betriebssystem Microsoft Windows 10.

USB	Lesen	Schreiben
3.0	121,57 MB/s	12,55 MB/s
2.0	32,89 MB/s	10,38 MB/s

Tabelle 2: Vergleich USB 2.0 und USB 3.0

Um einen Referenzpunkt zu haben, wurden die USB-Speichersticks zusätzlich mit dem Programm `dd` auf ihre Geschwindigkeit auf dem Raspberry Pi 3 Model B getestet. Dazu muss in das Verzeichnis, wo der Speicherstick eingehängt ist, gewechselt werden. Im Verzeichnis wird eine 2000-MB-Große Datei `Testdatei` generiert, die mit Nullen gefüllt wird. Bei der Erstellung dieser Datei wird die Schreibgeschwindigkeit gemessen. Nach einer Pause von fünf Sekunden,

wird die Datei gelesen und dabei die Lesegeschwindigkeit des Speichersticks ermittelt.

```
pi@alx-pi:/media/pi/B06A-EE26 $ sudo dd if=/dev/zero
of=Testdatei bs=1M count=2000; sleep 5 ; dd if=Testdatei
of=/dev/null bs=1M
2000+0 Datensätze ein
2000+0 Datensätze aus
2097152000 Bytes (2,1 GB) kopiert, 205,74 s, 10,2 MB/s
2000+0 Datensätze ein
2000+0 Datensätze aus
2097152000 Bytes (2,1 GB) kopiert, 64,5218 s, 32,5 MB/s
```

Der Test zeigt, dass die Werte des, am USB-2.0-Anschluss getesteten, USB-3.0-Speichersticks am Raspberry Pi, nahezu identisch mit den Werten von dem Windows-Rechner sind.

5.2 Erstellen eines RAID-Z1

Für den Aufbau eines ZFS-RAIDs auf dem Raspberry Pi 3 Model B, wurde das RAID-System RAID-Z1 gewählt, das dem System RAID 5 ähnlich ist (vergleiche Kapitel 1.3). Anders als ein RAID 5, sind die RAID-Z-Systeme gegen Synchronisationsprobleme gesichert (vergleiche Kapitel 1.2 → Write-hole-Problem).

Der USB-Hub wird an den USB-Anschluss des Raspberry Pi angeschlossen. In diesen werden drei USB-Speichersticks eingesteckt und mit dem Kommando *lslbk* geprüft, ob diese vom System erkannt werden.

```
pi@alx-pi:~ $ lsblk

NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda           8:0    1 14,5G  0 disk
└─sda1        8:1    1 14,5G  0 part /media/pi/1A39-20B6
sdb           8:16   1 14,5G  0 disk
└─sdb1        8:17   1 14,5G  0 part /media/pi/B06A-EE26
sdc           8:32   1 14,5G  0 disk
└─sdc1        8:33   1 14,5G  0 part /media/pi/6A46-8366
(...)
```

Die Speichersticks wurden erkannt und automatisch in das Verzeichnis */media/pi* eingehängt. In der Spalte *SIZE* wird die tatsächliche Größe der Wechseldatenträger von 14,5 GB angezeigt. Für die Erstellung des RAID-Z1

müssen die automatisch eingehängten Sticks, manuell mit dem Befehl *umount <Einhängepunkt>*, ausgehängt werden. Dies ist notwendig, da sonst ZFS diese nicht in das RAID-Z1 einbinden kann, weil diese schon in einem anderen System arbeiten. Das wird auch bei dem Kreieren des RAID-Z1 oder eines Pools als Fehlermeldung angezeigt. Des Weiteren werden die eindeutigen Kennungen – die IDs der USB-Speichersticks benötigt. Diese sind im Verzeichnis */dev/disk/by-id* aufgelistet. Der Vorteil die Speichermedien mittels ID an das RAID-Z zu binden ist, dass diese nach einem Systemneustart des Raspberry Pi, auch richtig erkannt werden und dem jeweiligen Pool zugewiesen werden. Dies ist nötig, da bei Linux-Distributionen die Bezeichnungen der USB-Anschlüsse nach einem Neustart andere sein können, als zum Zeitpunkt des Kreierens des RAID-Z1.

Mit folgendem Befehl wird das RAID-Z1 mit dem Namen *testraidz* erstellt:

```
pi@alx-pi:~ $ sudo zpool create testraidz raidz \
/dev/disk/by-id/usb-Disk_4C530001210914116044-0:0 \
/dev/disk/by-id/usb-Disk_4C530001280826123270-0:0 \
/dev/disk/by-id/usb-Disk_4C530001300902105093-0:0
```

Im weiteren Verlauf dieser Arbeit, werden Identifikationsnummern der Laufwerke abgekürzt angegeben:

- usb-Disk_4C530001210914116044-0:0 → usb-6044
- usb-Disk_4C530001280826123270-0:0 → usb-2370
- usb-Disk_4C530001300902105093-0:0 → usb-5093

Das Kommando *zpool status testraidz* gibt Informationen über den soeben erstellten RAID-Z1 aus.

```
pi@alx-pi:~ $ zpool status testraidz
pool: testraidz
state: ONLINE
scan: none requested
config:
     NAME            STATE    READ WRITE CKSUM
     testraidz       ONLINE    0     0     0
     raidz1-0        ONLINE    0     0     0
           usb-6044  ONLINE    0     0     0
           usb-3270  ONLINE    0     0     0
           usb-5093  ONLINE    0     0     0

errors: No known data errors
```

5.3 Testwerkzeug

Die Geschwindigkeitstests des zuvor erstellen RAID-Z1, insbesondere für random read and write (das zufällige Lesen und Schreiben), erfolgten mit dem Programm *Iozone*. Die Installation des Programms erfolgt mit folgenden Schritten: [50]

```
pi@alx-pi:~ $ wget
http://www.iozone.org/src/current/iozone3_434.tar

pi@alx-pi:~ $ cat iozone3_434.tar | tar -x
pi@alx-pi:~ $ cd iozone3_434/src/current
pi@alx-pi:~ $ sudo make linux-arm
```

Um einen Geschwindigkeitstest ausführen zu können, ist ein Wechsel in das Verzeichnis *iozone3_434/src/current* nötig. Durch das Kommando *./iozone -a* startet der automatische Testdurchlauf. Die *-e*-Funktion und die *I*-Funktion bewirken, dass die Ein- und Ausgänge bei dem Test den Puffer-Zwischenspeicher umgehen, um so die Ergebnisse nicht zu verfälschen. [51] *Iozone* führt im automatischen Modus 13 verschiedene Tests durch, die nach Bedarf einzeln ablaufen können. Dies geschieht mit dem Zusatz *-i <Nummer des Tests>*:

- „0=write/rewrite
- 1=read/re-read
- 2=random-read/write
- 3=Read-backwards
- 4=Re-write-record
- 5=stride-read
- 6=fwrite/re-fwrite
- 7=fread/Re-fread,
- 8=random mix
- 9=pwrite/Re-pwrite
- 10=pread/Re-pread
- 11=pwritev/Re-pwritev
- 12=preadv/Re-preadv“ [52]

Für die folgenden Messungen im ZFS-Pool sind die Nummern 0 für die Schreibgeschwindigkeit (write), die 1 für die Lesegeschwindigkeit (read) und die Nummer 2 für das zufällige Lesen (random read) und Schreiben (random write) im Pool, zu verwenden. Für den Testdurchlauf kann die zu schreibende und die zu lesende Dateigröße mit *-s* bestimmt werden. Die Ergänzung *-r* für die Spalte *reclen*, bestimmt die Größe der zu testenden Datenblöcke an. [51] Das heißt, eine 512-MB-Große Datei, bei gesetztem -r 4, ergibt 512 MB / 4k=131.072 Datenblöcke, die das Programm *iozone* schreibt oder ausliest. Das Programm generiert die Tests für das Verzeichnis, in dem es gespeichert ist, in diesem Fall für die Micro-SD-Speicherkarte, da *iozone* im Verzeichnis */home/pi* liegt. Mit dem Zusatz *-f* entsteht ein Verweis auf ein zu testendes Dateisystem. Die Ausgabe der Testergebnisse im Terminal, erfolgt in Kilobytes pro Sekunde. Die Ergebnisse können zusätzlich mit *-b <Speicherort>* als Datei gespeichert werden.

Das folgende Befehlsfenster zeigt einen *iozone*-Aufruf zum Testen des Verzeichnisses */dev/sda1* (USB-Speicherstick). Der Test beinhaltet den Lese- und Schreibtest der 512-MB-Großen Datei mit 4-kB-Großen Datenblöcken. Die Speicherung der Werte erfolgt im Verzeichnis */home/pi/test.xls*.

```
pi@alx-pi:~/iozone3_434/src/current $ ./iozone -e -I -a
-i 0 -i 1 -s 512M -r 4 -f /dev/sda1 -b /home/pi/test.xls
```

Die Ausgabe des Testergebnisses im Terminal, nach einem erfolgreichen Testdurchlauf:

```
    kB  reclen      write  rewrite    read    reread
524288       4       4520     3861    14276    14516
iozone test complete.
```

5.4 Geschwindigkeitstests

Die Geschwindigkeitstests wurden mittels des Programms *iozone* jeweils dreimal ausgeführt, danach der Durchschnittswert ermittelt. Die Testdurchläufe liefen mit einer 2-GB-Großen Datei und den Datenblöcken von 4 Kilobyte, 512 Kilobyte und 16 MB (16.834 Kilobyte) ab. Die Messung beinhaltet die Testfälle für die Schreibgeschwindigkeit, die Lesegeschwindigkeit und die Geschwindigkeit für das zufällige Lesen und Schreiben.

Für die Referenzmessung dient, der im RAID-Z eingesetzte, USB-Speicherstick. Die Ausführung des Tests erfolgt mit folgendem Befehl:

```
pi@alx-pi:~/iozone3_434/src/current $ ./iozone -e -I -a
-s 2G -r 4 -r 512 -r 16M -i 0 -i 1 -i 2 -f /dev/sda1
```

Für das RAID-Z wurde ebenso das Kommando verwendet, allerdings mit dem Speicherpfad des *testraidz*.

```
pi@alx-pi:~/iozone3_434/src/current $ ./iozone -e -I -a
-s 2G -r 4 -r 512 -r 16M -i 0 -i 1 -i 2 -f testraidz
```

Die Durchschnittswerte der Testdurchläufe (Diagramm 1 und 2) mit *iozone* für den USB-Speicherstick und für den RAID-Z zeigen, dass die Schreibgeschwindigkeit des RAID-Z, gegenüber dem einzelnen USB-Speicherstick, bei den 4 Kilobyte und 16 MB Datenblöcken, keine nennenswerte Veränderung hervorgebracht hat. Die Steigerung der Geschwindigkeit erfolgte im 512 Kilobyte Messblock, von 7.778 Kilobyte in der Sekunde auf 12.564 Kilobyte in der Sekunde. Die Lesegeschwindigkeit ist in allen gemessenen Blöcken nach unten gefallen. Die Testfälle zufälliges Lesen und zufälliges Schreiben zeigen ebenfalls fallende Werte. Besonders im 4 Kilobyte- und im 512 Kilobytebereich ist der Unterschied zum einzelnem USB-Speicherstick, bei zufälligem Schreiben, deutlich zu sehen. Dies liegt daran, weil das Testprogramm die 2-GB-Große Datei in kleine 4-Kilobyte-Große Blöcke aufteilt und diese dann willkürlich auf die drei, im RAID-Z1 eingebundenen, Speichermedien schreibt. Zusätzlich zu jedem Block erfolgt die Anlegung der Paritätsdateien, die für die Wiederherstellung der fehlerhaften Blöcke zuständig sind. Diese Faktoren drosseln die Leistung des RAID-Z1. Bei den Messungen gab es keine Ausreißer nach oben oder nach unten, die Werte lagen dicht beieinander.

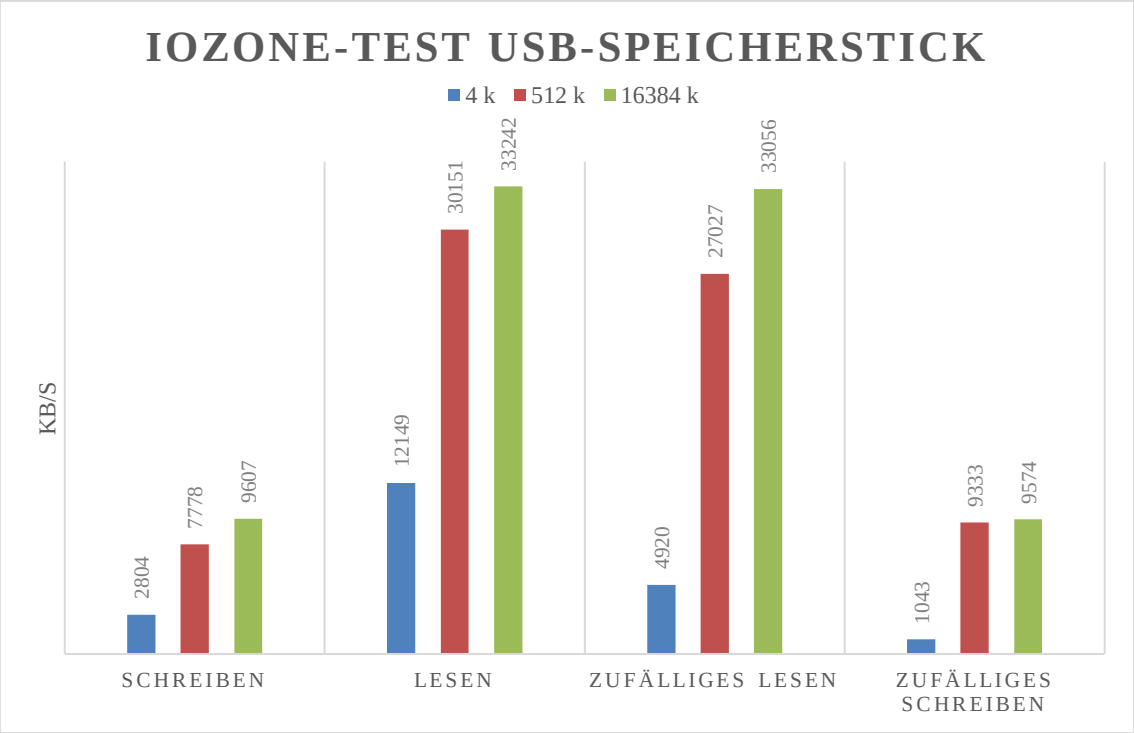

Diagramm 1: Iozone-Test USB-Speicherstick

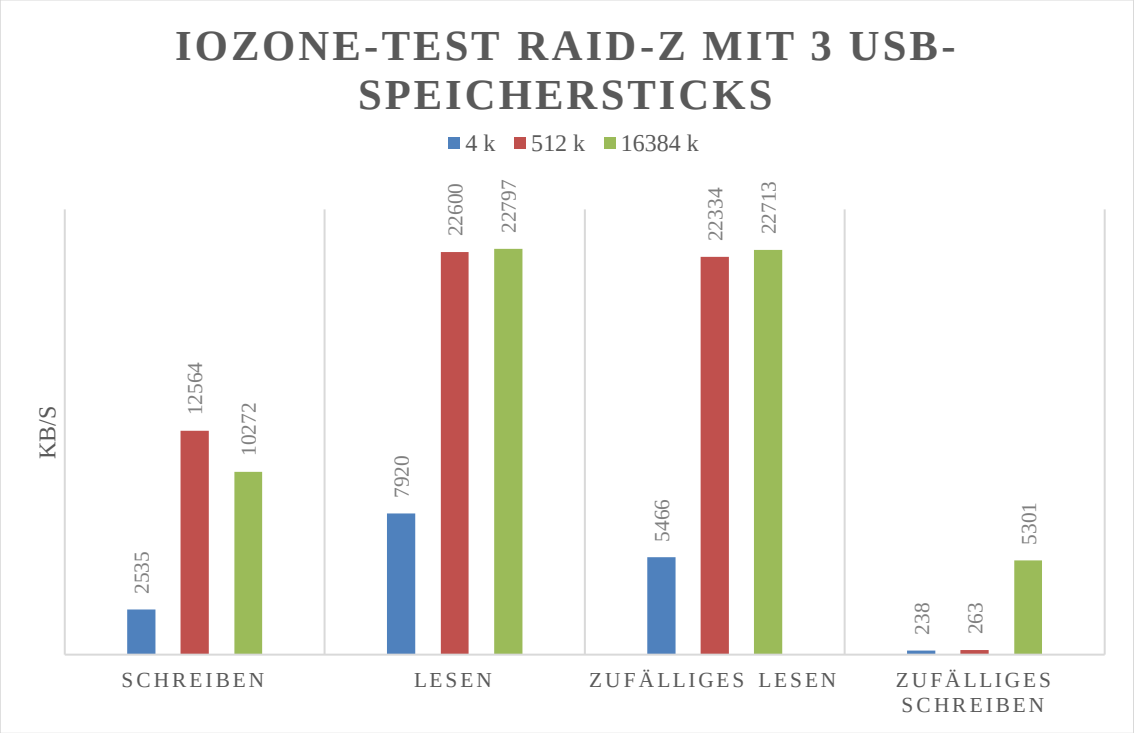

Diagramm 2: Iozone-Test RAID-Z mit 3 USB-Speichersticks

Die Tests mit einer 512-MB-Großen Datei ergaben für die Testläufe für das zufällige Schreiben eine Steigerung um 87 % (447 KB/s) im 4-Kilobyte-Messbereich und ebenso eine um 112 % (558 KB/s) gestiegene Schreibleistung im 512-Kilobyte-Messbereich. Die Testergebnisse im 16-MB-Messbereich ergaben eine Leistungssteigerung um 62 % (8592 KB/s) für das zufällige Schreiben. Die reine Schreibleistung ergab keine großen Veränderungen für die Bereiche 4 Kilobyte und 16 MB gegenüber den Ergebnissen für die 2-GB-Datei. Im 512-Kilobyte-Bereich dagegen ist die reine Schreibleistung um 21 % auf 15.292 KB/s angestiegen. Für die Testfälle Lesen und zufälliges Lesen hat die 512-MB-Große Datei keine nennenswerten Auswirkungen gehabt (vergleiche Diagramm 3).

Die Ergebnisse der Tests zeigen, dass je kleiner eine zu verarbeitende Datei ist, desto höher die Geschwindigkeit, besonders deutlich zeigt sich das im 4-Kilobyte-Messbereich.

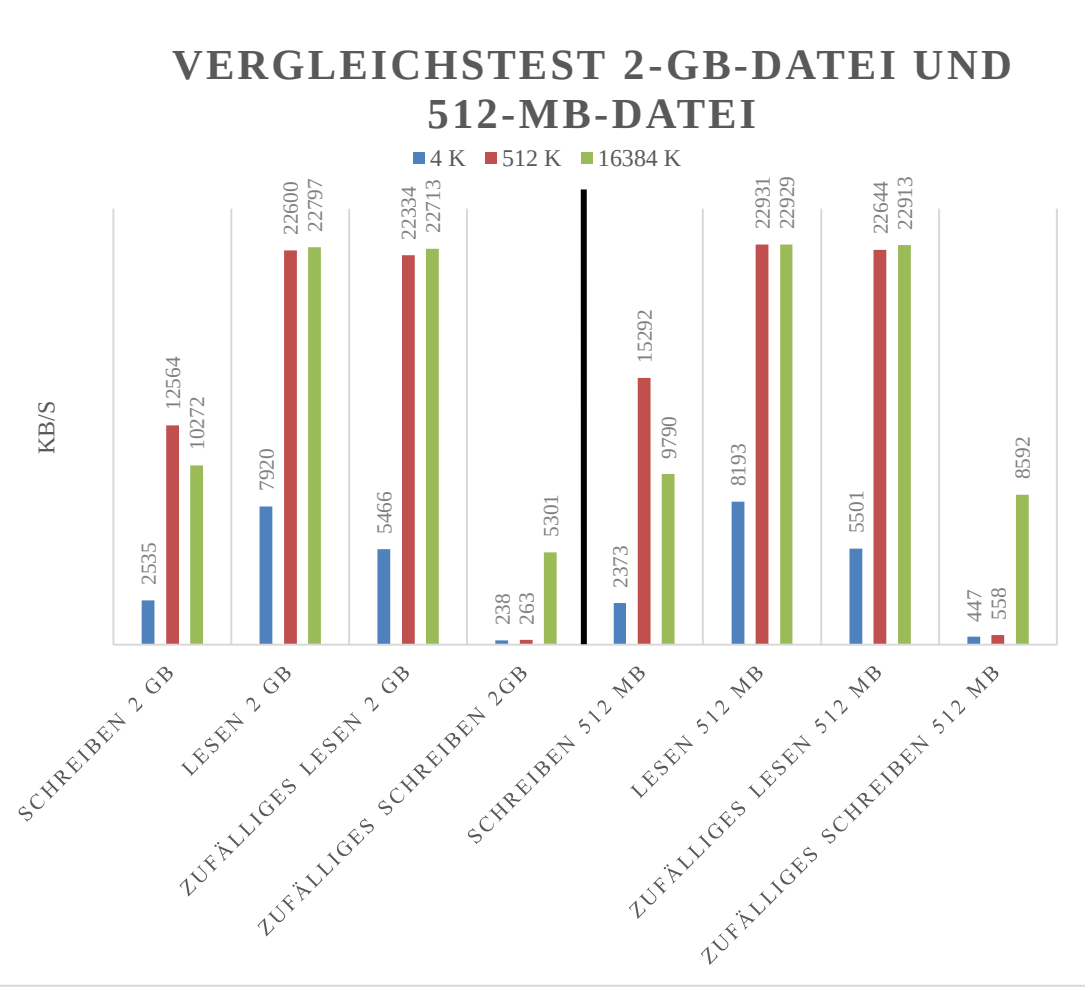

Diagramm 3: Vergleichstest 2-GB-Datei und 512-MB-Datei

Die Tests haben gezeigt, dass die Geschwindigkeit im RAID-Z1, vor allem bei den Zufallstests Lesen/Schreiben, niedriger sind, als bei einem einzelnen USB-Speicherstick. Das RAID-Z1 kann für die Datensicherung angelegt werden, aber nicht, wenn hohe Geschwindigkeiten im Vordergrund stehen.

Eine weitere Möglichkeit die Daten redundant zu Speichern ist die Spiegelung. Das heißt, zwei oder mehr Festplatten haben identische Daten vorhanden, um im Falle eines Speichermediumausfalles die Daten trotzdem vorrätig zu haben. Bei zwei Festplatten bedeutet dies allerdings 50 % Kapazitätsverlust.

In folgendem Test stehen die Geschwindigkeiten des RAID-Z1 und einem Speigel (Mirror) aus zwei USB-Speichersticks gegenüber. Der Test soll ermitteln, ob eine Spiegelung der Daten, bei einer Einsparung von einem USB-Speicherstick, Vorteile in der Leistung bringt.

Die Erstellung des Spiegelpools *testmirror* auf dem Raspberry Pi erfolgt mit folgendem Befehl:

```
pi@alx-pi:~ $ sudo zpool create testmirror mirror
/dev/disk/by-id/6044-0:0 /dev/disk/by-id/3270-0:0
```

Die Testdurchläufe für den Spiegelpool finden nach dem Muster der Tests für den USB-Speicherstick und für das RAID-Z statt.

```
pi@alx-pi:~/iozone3_434/src/current $ ./iozone -e -I -a
-s 2G -r 4 -r 512k -r 16M -i 0 -i 1 -i 2 -f testmirror
```

Der Vergleich der Ergebnisse (Diagramm 4) zeigt, dass es kaum Unterschiede in der Geschwindigkeit vorhanden sind. Auch die Tests mit einer 512-MB-Großen Datei liefern, für das RAID-Z1 und die Spiegelung, nahezu identische Werte. Die Ergebnisse beim zufälligem Lesen und bei reinem Lesen kommen bei beiden Konzepten nicht über 23 MB/s. Dies liegt an dem USB-Controller des Raspberry Pi, der hier als Flaschenhals fungiert. [53]

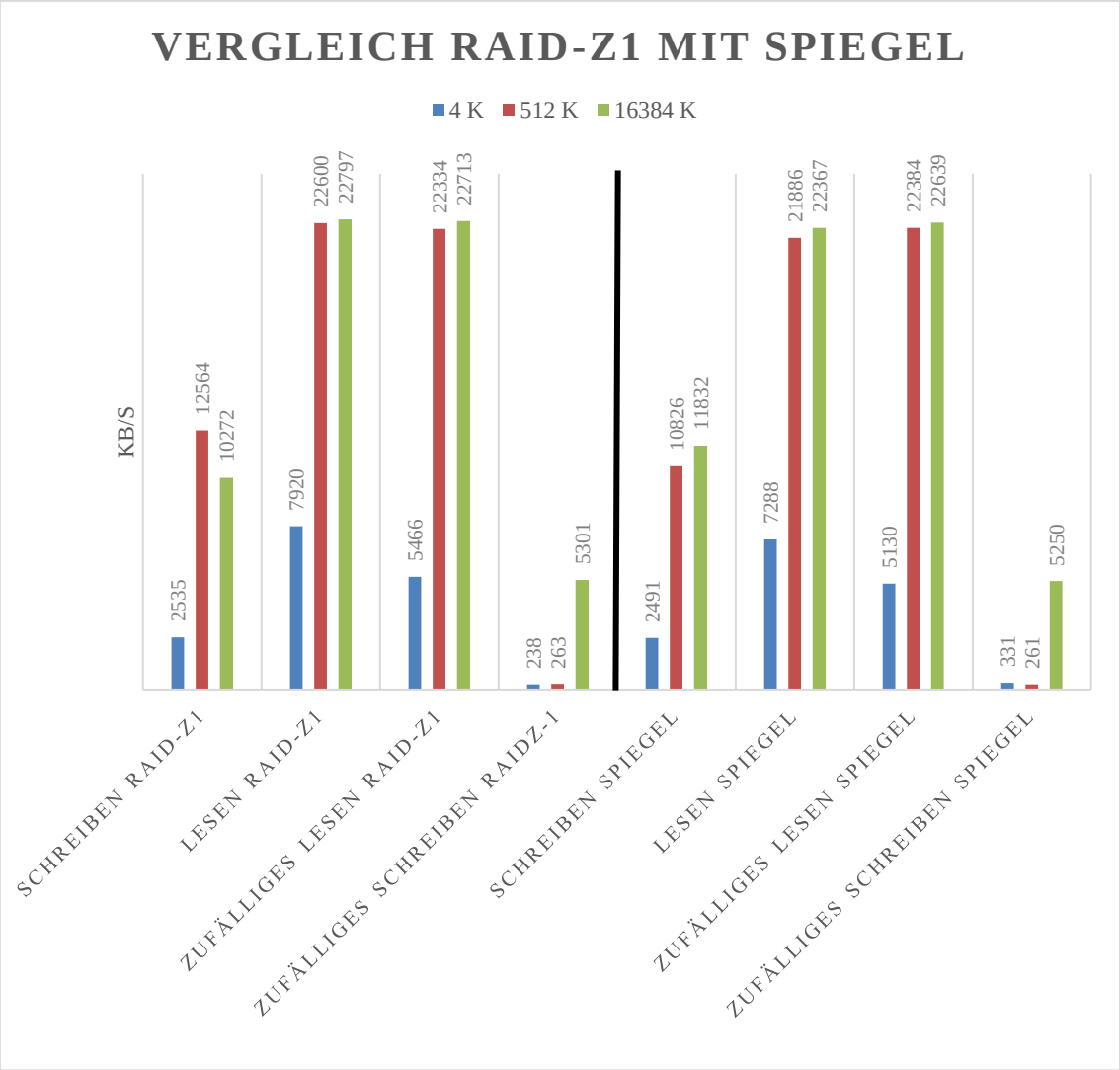

Diagramm 4: Vergleich RAID-Z1 mit Spiegel

5.5 Ausfallsicherheit

Dieser Abschnitt untersucht die Ausfallsicherheit der Spiegelung mit USB-Speichersticks. Für die Simulation des Ausfalls dient, das im Kapitel 5.4, erstellter ZFS-Spiegelpool.

Es Folgt die Beschreibung des Testfalls zur Simulation des Ausfalls eines der USB-Datenträger. Beim Kopieren einer 100-MB-Großen Datei *ausfall* nach *testmirror*, soll ein Speichermedium ausfallen. Das Simulieren des Ausfalls erfolgt durch das Herausziehen des USB-Speichersticks *usb-3270* aus dem USB-Anschluss des Raspberry Pi. Die Beobachtung des Pools während des Kopierens ist durch den Befehl *zpool status testmirror 2* möglich. Die 2 steht für das Aktualisieren des Poolstatus alle zwei Sekunden. Das Aktualisierungszeitfenster ist dabei frei wählbar. Das Wiedereinbinden des USB-Speichersticks in den Pool, geschieht nach dem Abschluss des Kopiervorganges. Die Wiederherstellung des Spiegelverbundes soll die Selbstheilung (siehe Kapitel 1.2) des ZFS bestätigen.

```
pi@alx-pi:~ $ sudo cp ausfall /testmirror

pi@alx-pi:~ $ zpool status 2
pool: testmirror
state: DEGRADED

status: One or more devices could not be used because
the label is missing or invalid. Sufficient replicas
exist for the pool to continue functioning in a degraded
state.

action: Replace the device using 'zpool replace'.

   see: http://zfsonlinux.org/msg/ZFS-8000-4J

  scan: none requested
config:
     NAME            STATE     READ WRITE CKSUM
     testmirror      DEGRADED     0     0     0
     mirror-0        DEGRADED     0     0     0
          usb-6044   ONLINE       0     0     0
          usb-3270   UNAVAIL      0     0     0
```

Die Statusausgabe von *testmirror* zeigt, dass der USB-Speicherstick *usb-3270* nicht vorhanden ist und der Pool zwar funktioniert, aber in einem eingeschränkten Modus. Die Ausgabe der Meldung erfolgt auch bei einem Ausfall eines Speichermediums in einem RAID-Z.

Beim Wiedereinbinden des *usb-3270*, durch das Einstecken in den USB-Anschluss des Raspberry Pi, erkennt ZFS zu welchem Pool das Medium gehört, bindet es dort ein und startet selbstständig mit dem Abgleich der Bits zwischen den zwei USB-Speichersticks. Der Selbstheilungsprozess bringt den USB-Speicherstick auf den Stand des anderen USB-Speichersticks. Die Zeile *scan* zeigt die erfolgreiche Wiederherstellung von 48,2 MB und der Poolstatus ist wieder *online*.

```
pi@alx-pi:~ $ zpool status testmirror
pool: testmirror
state: ONLINE
scan: resilvered 48,2M in 0h0m with 0 errors on Wed May
      24 21:43:07 2017
config:
     NAME            STATE     READ WRITE CKSUM
     testmirror      ONLINE       0     0     0
     mirror-0        ONLINE       0     0     0
          usb-6044   ONLINE       0     0     0
          usb-3270   ONLINE       0     0     0

errors: No known data errors
```

Der Test hat gezeigt, dass auch eine Spiegelung der Daten ausfallsicher und selbstheilend ist. Der Vorteil von RAID-Z1 ist der Speicherplatz. Der Platz beim Datenträger-Spiegel hat nur die Größe eines Speichermediums. Für die Spiegelung, in diesem Fall, spricht die Kostenersparnis im Wert von einem USB-Speicherstick.

6 Fazit

Das Ziel der vorliegenden Arbeit war es, das Dateisystem ZFS auf einem Raspberry Pi 3 Model B mit dem Betriebssystem Raspbian zu installieren und in Bezug auf die Geschwindigkeit und Datensicherheit zu testen.

Die Installation von ZFS und vor allem die Konfiguration, erfordert ein solides Basiswissen in der Linux-Umgebung, da bis jetzt noch wenig Dokumentation oder Konfigurationsanleitungen für den Raspberry Pi vorhanden ist. Die Anleitungen, die zurzeit (Stand Mai 2017) im Internet verfügbar sind, sind meist in Foren zu finden. Die Installationsschritte sind nicht immer vollständig oder sind an andere Betriebssysteme gebunden. Auch verschiedene Kernel-Versionen oder ZFS-Versionen müssen bei der Installation berücksichtigt werden. Die Behebung, der im Kapitel „3.3 Konfiguration von ZFS“ vorgestellten Problemen, das nicht automatische Laden der ZFS-Module oder das Nichtspeichern der neuen Pools, war sehr zeitintensiv.

Das Arbeiten mit ZFS gestaltet sich, dank der wenigen Befehle und der vielen Dokumentationen, als unkompliziert. Die Befehle sind meist selbsterklärend. Die beschriebenen Möglichkeiten von ZFS in dieser Bachelorarbeit, decken nicht alle Funktionen des Dateisystems ab. Dies war auch nicht das Ziel dieser Arbeit.

Ein Testdurchlauf mit `iozone` kann mehrere Stunden dauern. Versuche, die Tests mit dem Programm `Bonnie++` auszuführen, scheiterten. Zum einen, weil die Tests beim Lesen und zufälligem Lesen keine Werte geliefert hatten, zum anderem, setzte bei den Testdurchläufen zeitweise die WLAN-Funktion aus. Dies erschwerte die Bedienung des Raspberry Pi.

Die Tests haben gezeigt, dass die Geschwindigkeiten im RAID-Z1 und in der Spiegelung, vor allem beim zufälligem Lesen und zufälligem Schreiben, niedriger sind, als bei einem einzelnen USB-Speicherstick. Aufgrund der Problematik des USB-Flaschenhalses (Kapitel 5.4) eignen sich RAID-Z1 und die Spiegelung nur bedingt für die Bereiche, die hohe Geschwindigkeiten erfordern.

7 Ausblick

Als nächstes kann das Verhalten der Geschwindigkeiten, bei mehr als drei USB-Speichermedien im RAID-Z1 gemessen werden. Es wäre interessant einen, auf dem Raspberry Pi aufgesetzten, RAID 5 zu testen und die Ergebnisse mit den Messungen des RAID-Z1 vergleichen. Die überschaubaren Kosten der benötigten Komponenten, machen dieses Vorhaben erschwinglich. Außerdem kann ein Vergleichstest mit USB-2.0-Datenträgern durchgeführt werden.

Literaturverzeichnis

[1] www.elektronik-kompendium.de, „Raspberry Pi," [Online]. Available: http://www.elektronik-kompendium.de/sites/com/1904221.htm. [Zugriff am 08 April 2017].

[2] Manuel, „Raspberry Pi - Modellvergleich," 2014 Mai 2014. [Online]. Available: https://www.datenreise.de/raspberry-pi-unterschiede-zwischen-den-modellen. [Zugriff am 08 April 2017].

[3] M. Flasskamp, „Welcher Raspberry Pi? Alle Modelle im Vergleich," [Online]. Available: http://praxistipps.chip.de/welcher-raspberry-pi-alle-modelle-im-vergleich_41923. [Zugriff am 08 April 2017].

[4] A. Proschofsky, „Raspberry Pi Zero: Neues 10-Dollar-Modell mit WLAN und Bluetooth," [Online]. Available: http://derstandard.at/2000053308472/Raspberry-Pi-Zero-Neues-10-Dollar-Modell-mit-WLAN-. [Zugriff am 08 April 2017].

[5] A. Merz, „Compute Module 3 ist verfügbar," 16 Januar 2017. [Online]. Available: https://www.golem.de/news/raspberry-pi-compute-module-3-ist-verfuegbar-1701-125601.html. [Zugriff am 08 April 2017].

[6] www.futurezone.at, „Meistverkaufte Computer: Raspberry Pi überholt C64," [Online]. Available: Vergleiche https://futurezone.at/produkte/meistverkaufte-computer-raspberry-pi-ueberholt-c64/252.933.416 . [Zugriff am 08 April 2017].

[7] www.wikipedia.org, „Raspberry Pi," [Online]. Available: https://de.wikipedia.org/wiki/Raspberry_Pi. [Zugriff am 08 April 2017].

[8] R. Schanze, „Raspberry Pi: WLAN einrichten – So geht's," 24 Februar 2016. [Online]. Available: http://www.giga.de/zubehoer/raspberry-pi/tipps/raspberry-pi-wlan-einrichten-so-geht-s . [Zugriff am 08 April 2017].

[9] Eva-Katharina Kunst, Jürgen Quade, „Der mit dem 64-Bit-Kernel tanzt," 21 November 2016. [Online]. Available: https://www.golem.de/news/raspberry-pi-der-mit-dem-64-bit-kernel-tanzt-1611-124475.html. [Zugriff am 14 April 2017].

[10] C. Electronic, „Raspberry Pi 3 Model B 1 GB ohne Betriebssystem," [Online]. Available: https://www.conrad.de/de/raspberry-pi-3-model-b-1-gb-ohne-betriebssystem-1419716.html . [Zugriff am 08 April 2017].

[11] Lode_Runner, „ZFS on Linux,“ 16 Mai 2017. [Online]. Available: https://wiki.ubuntuusers.de/ZFS_on_Linux . [Zugriff am 26 Mai 2017].

[12] www.wikipedia.org, „ZFS (Dateisystem),“ [Online]. Available: https://de.wikipedia.org/wiki/ZFS_(Dateisystem)#Datentr.C3.A4ger-Pools. [Zugriff am 10 April 2017].

[13] www.wikipedia.org, „Copy-On-Write,“ [Online]. Available: https://de.wikipedia.org/wiki/Copy-On-Write . [Zugriff am 20 April 2017].

[14] Z. A. Recovery, „RAID Recovery Guide,“ [Online]. Available: http://www.raid-recovery-guide.com/raid5-write-hole.aspx . [Zugriff am 22 April 2017].

[15] T. Moorman, „Funktionen von ZFS,“ [Online]. Available: http://www.fh-wedel.de/~si/seminare/ws08/Ausarbeitung/02.zfs/funktionen.html. [Zugriff am 22 April 2017].

[16] Win32netsky, „PPA,“ 15 November 2016. [Online]. Available: https://wiki.ubuntuusers.de/Launchpad/PPA . [Zugriff am 22 April 2017].

[17] T. Leemhuis, „Juristen uneins bei ZFS-Lizenzproblematik in Ubuntu 16.04 LTS,“ 29 Februar 2016. [Online]. Available: https://www.heise.de/newsticker/meldung/Juristen-uneins-bei-ZFS-Lizenzproblematik-in-Ubuntu-16-04-LTS-3120072.html. [Zugriff am 25 Mai 2017].

[18] www.pcwelt.de, „Was ist ein RAID-System?,“ [Online]. Available: https://www.pcwelt.de/ratgeber/Was-ist-ein-RAID-System-NAS-Server-445517.html . [Zugriff am 19 Mai 2017].

[19] www.recoverylab.de, „RAID 5 Datenrettung,“ [Online]. Available: https://www.recoverylab.de/raid5-datenrettung-nach-ausfall-von-festplatten/ . [Zugriff am 19 Mai 2017].

[20] www.wikipedia.org, „RAID,“ [Online]. Available: https://de.wikipedia.org/wiki/RAID#RAID_5 . [Zugriff am 20 Mai 2017].

[21] Mytril, „SSH,“ 23 April 2017. [Online]. Available: https://wiki.ubuntuusers.de/SSH/ . [Zugriff am 26 April 2017].

[22] C. Langner, „SSH auf dem Raspberry Pi aktivieren (jetzt unter Raspian nötig),“ 30 November 2016. [Online]. Available: https://linuxundich.de/raspberry-pi/ssh-auf-dem-raspberry-pi-aktivieren-jetzt-unter-raspian-noetig/. [Zugriff am 17 April 2017].

[23] Manuel, „RASPBERRY PI – REMOTE DESKTOP (XRDP) INSTALLIEREN,“ 16 Oktober 2016. [Online]. Available: https://www.datenreise.de/raspberry-pi-remote-desktop-xrdp-

installieren/ . [Zugriff am 26 Mai 2017].

[24] mt08, „[メモ](RaspberryPi) Building ZFS on Raspbian,“ 02 März 2017. [Online]. Available: http://qiita.com/mt08/items/e55b285b845a8acb4b5e . [Zugriff am 25 Mai 2017].

[25] www.packages.debian.org, „Paket: build-essential (11.7 und andere),“ [Online]. Available: https://packages.debian.org/de/jessie/build-essential . [Zugriff am 25 Mai 2017].

[26] www.packages.debian.org, „Paket: autoconf (2.69-8),“ [Online]. Available: https://packages.debian.org/de/jessie/autoconf . [Zugriff am 25 Mai 2017].

[27] www.packages.debian.org, „Paket: libtool (2.4.2-1.11),“ [Online]. Available: https://packages.debian.org/de/jessie/libtool . [Zugriff am 25 Mai 2017].

[28] www.packages.debian.org, „Paket: gawk (1:4.1.1+dfsg-1),“ [Online]. Available: https://packages.debian.org/de/jessie/gawk. [Zugriff am 25 Mai 2017].

[29] www.packages.debian.org, „Paket: alien (8.92),“ [Online]. Available: https://packages.debian.org/de/jessie/alien . [Zugriff am 25 Mai 2017].

[30] www.packages.debian, „Paket: fakeroot (1.20.2-1),“ [Online]. Available: https://packages.debian.org/de/jessie/fakeroot. [Zugriff am 25 Mai 2017].

[31] www.packages.debian.org, „Paket: gdebi (0.9.5.5+nmu1),“ [Online]. Available: https://packages.debian.org/de/jessie/gdebi . [Zugriff am 25 Mai 2017].

[32] www.packages.debian.org, „Paket: zlib1g (1:1.2.8.dfsg-2 und andere),“ [Online]. Available: https://packages.debian.org/de/jessie/zlib1g . [Zugriff am 25 Mai 2017].

[33] packages.debian.org, „Paket: uuid-dev (2.25.2-6),“ [Online]. Available: https://packages.debian.org/de/jessie/uuid-dev . [Zugriff am 25 Mai 2017].

[34] www.packages.debian.org, „Paket: libattr1-dev (1:2.4.47-2),“ [Online]. Available: https://packages.debian.org/de/jessie/libattr1-dev . [Zugriff am 25 Mai 2017].

[35] www.packages.debian.org, „Paket: libblkid-dev (2.25.2-6),“ [Online]. Available: https://packages.debian.org/de/jessie/libblkid-dev . [Zugriff am 25 Mai 2017].

[36] www.wikipedia.org, „Security-Enhanced Linux,“ [Online]. Available: https://en.wikipedia.org/wiki/Security-Enhanced_Linux. [Zugriff am 25 Mai 2017].

[37] www.packages.debian, „Paket: ksh (93u+20120801-1),“ [Online]. Available: https://packages.debian.org/de/jessie/ksh . [Zugriff am 25 Mai 2017].

[38] www.packages.debian.org, „Paket: libudev-dev (215-17+deb8u7),“ [Online]. Available: https://packages.debian.org/de/jessie/libudev-dev . [Zugriff am 25 Mai 2017].

[39] www.packages.debian.org, „Paket: parted (3.2-7),“ [Online]. Available: https://packages.debian.org/de/jessie/parted . [Zugriff am 25 Mai 2017].

[40] www.packages.debian.org, „Paket: lsscsi (0.27-3),“ [Online]. Available: https://packages.debian.org/de/jessie/lsscsi . [Zugriff am 25 Mai 2017].

[41] www.raspberrypi.org, „Kernel Headers,“ [Online]. Available: https://www.raspberrypi.org/documentation/linux/kernel/headers.md . [Zugriff am 25 Mai 2017].

[42] www.packages.debian.org, „Paket: libdevmapper-dev (2:1.02.90-2.2+deb8u1),“ [Online]. Available: https://packages.debian.org/de/jessie/libdevmapper-dev . [Zugriff am 25 Mai 2017].

[43] www.wiki.archlinux.org, „ZFS,“ [Online]. Available: https://wiki.archlinux.org/index.php/ZFS . [Zugriff am 25 Mai 2017].

[44] www.ruhesmeile.com, „API,“ [Online]. Available: https://ruhmesmeile.com/glossar/definition/api/ . [Zugriff am 23 April 2017].

[45] H. Schwietering, „Programme kompilieren,“ 08 Mai 2017. [Online]. Available: https://wiki.ubuntuusers.de/Programme_kompilieren . [Zugriff am 23 Mai 2017].

[46] F. Kalhammer, „Verwaltung von Shared Libraries,“ [Online]. Available: http://www.linux-praxis.de/lpic1/lpi101/1.102.4.html . [Zugriff am 23 April 2017].

[47] www.oracle.com, „Oracle Solaris ZFS-Administrationshandbuch,“ [Online]. Available: http://docs.oracle.com/cd/E19253-01/820-2313/index.html . [Zugriff am 23 Mai 2017].

[48] www.oracle.com, „Verbinden und Trennen von Geräten in einem Speicher-Pool,“ [Online]. Available: http://docs.oracle.com/cd/E19253-01/820-2313/gcfhe/index.html . [Zugriff am 23 Mai 2017].

[49] www.oracle.com, „Kapitel 7 Arbeiten mit Oracle Solaris ZFS-Snapshots und -Klonen,“ [Online]. Available: http://docs.oracle.com/cd/E19253-01/820-2313/gavvx/index.html . [Zugriff am 23 Mai 2017].

[50] Jeff Geerling, „Test performance of a variety of microSD cards on Raspberry Pi - sequential and random read/write #7,“ 15 Februar 2015. [Online]. Available: https://github.com/geerlingguy/raspberry-pi-dramble/issues/7. [Zugriff am 23 Mai 2017].

[51] William D. Norcott, Don Capps, „Iozone Filesystem Benchmark,“ www.iozone.org, [Online]. Available: http://www.iozone.org/docs/IOzone_msword_98.pdf. [Zugriff am 23 Mai 2017].

[52] R. Natarajan, „10 iozone Examples for Disk I/O Performance Measurement on Linux,“ 02 Mai 2011. [Online]. Available: http://www.thegeekstuff.com/2011/05/iozone-examples . [Zugriff am 23 Mai 2017].

[53] G. Kainzbauer, „Raspbian Wheezy: Software RAID konfigurieren,“ 25 September 2016. [Online]. Available: http://www.gtkdb.de/index_36_2187.html. [Zugriff am 23 Mai 2017].