ARM-Plattform reif für den Alltag?

Desktop-Erfahrungen mit der ArchivistaBox auf ARM

Contents

1 Einleitung

Warum soll eine Applikation, sprich die ArchivistaBox, welche seit Jahren unter 64-Bit Intel/AMD läuft, auf ARM unter 32-Bit portiert werden? Welche Leistung lässt sich mit Kleinstrechnern à la Raspberry 2 und ODROID XU4 erzielen? Welche Vorteile, Hürden und/oder Nachteile gibt es dabei zu meistern?

Der Vortrag (bzw. vorliegend auch das Skript dazu) zeigt anhand der Portierung der ArchivistaBox auf die ARM-Plattform auf, welche Möglichkeiten und Grenzen der Einsatz von ARM im Jahre 2015 ergibt. Dabei geht es nur insofern um das Produkt ArchivistaBox, als dass dabei beispielhaft aufgezeigt werden kann, welche Chancen, Risiken und Limiten die ARM-Plattform derzeit beinhaltet.

Anlässlich des Vortrages müssen die Kleinstrechner gegen die Intel/AMD-Boliden antreten. Von Web-Services über Office-Dokumente, Texterkennung, Bildbearbeitung, Datenbanken und Virtualisierung werden typische Szenarien vorgestellt, die im 'klassichen' Alltag anfallen. Der Vortrag richtet sich an all jene, die derzeit mit Intel/AMD-Rechnern arbeiten und die einen Blick über den Tellerrand hinaus werfen möchten.

1.1 Ausser Konkurrenz

Nun ist es so eine Sache mit den Ankündigungen. Kaum sind sie gemacht, sind sie auch schon wieder veraltet. Der Vortragende hat sich zwei Wochen vor dem Vortrages dazu entschieden, dem Testfeld zusätzlich eine NVidia TV Box zu spendieren, weil dieser Rechner eine ARM 64 Bit CPU enthält.

2 Die Vorgeschichte

2.1 Intel/AMD

Intel und AMD sind zwei Firmen, welche Prozessoren für die gängigen Rechner herstellen. Während AMD zeitweise Intel kurz das Fürchten lernen konnte (mit AMD64), besteht seit Jahren die Konstellation, für Intel die Marktmacht, für AMD im besten Falle den Flohmarkt. Um dies mit Zahlen zu belegen. Intel erreichte in Q2 2015 marktbeherrschende 81.8 Prozent, AMD noch 'schlappe' 18.2 Prozent. Quelle:

www.pcgameshardware.de/CPU-Hardware-154106/News/

Alleine aus diesem Grunde tut ein/e Jede/r gut daran, sich gelegentlich nach Alternativen zu Intel/AMD zu machen. Wenn der eine (um es vielleicht etwas pointiert zu sagen) quasi nur noch am Leben bleibt, damit beim andern niemand von Monopol reden kann, dann muss sich niemand wundern, wenn die Innovation im besten Falle tröpfchenweise vorwärtsschreitet.

Der Vortragende machte diese Erfahrung gut und gerne bei der Auswahl geeigneter Hardware für die ArchivistaBox. Eine DualCore Intel CPU (T7100) mit einigermassen moderater Leistungsaufnahme kostete über Jahre ca. 400 USD im Einkauf. Um es vorweg zu nehmen, der nachfolgend vorgestellte ODROID XU4 ist diesbezüglich deutlich leistngsfähiger und kostet samt Platine und Netzteil einen Bruchteil davon.

Darum geht es bei einer Portierung der ArchivistaBox auf die ARM-Plattform keineswegs nur um einen Versuch, mit Rechnern zu spielen, sondern darum, bei den verwendeten Prozessoren Alternativen zu haben. Denn diese gibt es bei Intel/AMD im Prinzip seit längerer Zeit nicht mehr wirklich, bei 81.8 Prozent Marktanteil für Intel ist dies auch nicht weiter verwunderlich.

2.2 ARM

Der Begriff ARM ist weiter zu fassen. Einmal ist ARM eine Prozessorenarchitektur, weiter jedoch gibt es die gleichnahmige Firma, die selber keine Prozessoren produziert und verkauft, stattdessen jedoch die technologischen Grundlagen dazu. Firmen, welche einen ARM-basierten Prozessor auf den Markt bringen möchten, zahlen der Firma ARM meist eine Lizenz (derzeit in etwa 1 bis 1.5 USD pro Einheit), um dann daraus ein fertiges Produkt (sprich Prozessor/CPU) zu generieren. Kernstück der Intel/AMD-Prozessoren ist, dass sie leistungsfähige Befehlssätze verarbeiten können, bei ARM dagegen gibt es nur wenige Befehle, diese können aber einfach(er) in die Hardware 'gegossen' werden. ARM-CPUs sind daher in aller Regel betr. der Leistungsaufnahme genügsamer als die Intel/AMD-Pendants.

Dies ist auch der Grund, weshalb im Zusammenhang mit der ArchivistaBox seit 2009 Erfahrungen mit ARM bestehen. Irgendwo in einem staubigen Büro, ob in der Produktionshalle oder einem Kellergeschoss werden Dokumente gesannt. Die Erfahrungen mit lüfterbasierten Intel/AMD-Rechnern waren nicht schlecht, doch gab es jährlich im höheren Dutzender-Bereich Ausfälle zu beklagen, weil z.B. im Sommer die Lüfter nicht gerade Amok, aber doch am Anschlag liefen und dann so schön im Herbst, wenn niemand Zeit hat, den Geist aufgaben.

2.3 Am Anfang war das Scannen

Die Anfänge von ARM bei der ArchivistaBox gehen zurück auf Geräte wie die erste Version BeagleBaord, SheevaPlug und die letztlich zum Einsatz kommende Cisco NSLU2. Allen gemeinsam war, dass sie plus/minus nicht geeignet waren, die ArchivistaBox im gesamten Umfang zu beherbergen.

Als die erste Version des Raspberry PI im Jahre 2012 auf den Markt kam, war die ArchivistaBox Albis (basierend auf dem Cisco NSLU2 bereits auf dem Markt). Im Jahre 2014 wurde die ArchivistaBox auf das Modell Raspberry PI Modell B portiert. Obwohl sämtliche Programme migriert wurden, eignete sich der Raspberry PI Modell B als vollwertige ArchivistaBox kaum. Das Arbeiten mit einer Desktop-Umgebung erwies sich als gemächlich. Für wenige anspurchsvolle Sachen (wie z.B. das Scannen von Belegen oder das Überprüfen der Datensicherung) eignet sich das Modell B aber bis heute durchaus.

2.4 Cubieboard, Radxa, Odroid C1

Die zweite Charge bestand aus Geräten wie Cubieboard III, Radxa Rock Pro und dem Odroid C1. Mit Preisen zwischen 35 und 100 Einheiten (ob Franken, EURO, USD, so weit auseinander

liegen die alle nicht mehr) waren die Geräte sehr interessant, nur hatten alle Modelle eine eher bescheidenere Verbreitung. Mal schnell Linux installiert kein Problem, doch bald schon zeigten sich Probleme, die meistens daher ruhten, dass die mitgelieferten Distributionen nicht an die Hardware angepasst waren.

2.5 Raspberry PI 2

Mit der Veröffentlichung des Raspberry PI 2 änderte sich dies. Raspian läuft seit geraumer Zeit stabil und das Modell 2 erreicht plus/minus etwa die Geschwindigkeit der obengenannten Geräte. Auch wenn die propagierten 1:6 im Vergleich zum Modell 1 schon etwas hoch gegriffen sein dürften, klassische Desktop-Anwendungen laufen auf dem Raspberry PI 2 schon recht flott, wenn auch nicht famos. Kurz und gut, aus dem Raspberry PI 2 wurde die ArchivistaBox Bachtel.

2.6 8 Kerne zum Preis von 74 USD

Waren ARM-Rechner vor einigen Jahren nicht wirklich günstiger als preiswerte Intel/AMD-Systeme, so ist dies seit einiger Zeit nicht mehr der Fall. Bei Preisen von 35 USD für den Rasberry PI 2 oder 74 USD für den Odroid XU4, zu diesen Preisen lässt sich kein Intel/AMD-Rechner zusammenstellen, geschweige denn mit vier oder gar acht Kernen, wie dies z.B. beim ODROID XU4 seit Mitte 2015 der Fall ist. Kurz und gut, anfangs August wurden einige ODROID XU4 bestellt, und die ersten Messungen liessen durchaus aufhorchen.

Zur grossen Überraschung waren selbst Applikationen, welche mit einer CPU arbeiten (und dies dürfte der grösste Teil sein) nicht langsamer als dies bei einer Einstiegsmaschine mit Intel/AMD der Fall ist. Intel/AMD-Prozessoren im unteren Segment werden allerdings meist mit 2 CPU-Kernen ausgeliefert, bei acht CPUs ist der Geschwindigkeitsvorteil des ODROID XU4 doch recht massiv. Nicht dass damit gleich eine Intel i7 geknackt würde, doch kosten solche CPUs ja gut und gerne um den Faktor X mehr als ein Gesamtsystem auf der Basis ODROID XU4. In diesem Sinn ist der ODROID XU4 ein echtes 'Schnäppchen'.

2.7 Was bleibt ausser dem Preis?

Zunächst einmal sind beide Geräte, sowohl der Raspberry PI 2 als auch der ODROID XU4 extrem klein und leicht, beide Rechner passen problemlos in eine jede Hosentasche und mit einem Gewicht um die 100 Gramm sind sie auch deutlich leichter als die Intel/AMD-Rechner. Wie bereits eingangs erwähnt, auch bei der Leistungsaufnahme können die Winzlinge punkten. Ca. 4 bis 8 Watt für eine Quad-Core-Maschine (Gesamtsystem, nicht nur CPU) beim Raspberry

PI sind extrem tief, und auch die ca. 12 bis 18 Watt beim ODROID XU4 (6 Watt im Ruhezustand) sind an sich bescheiden. Natürlich verbraucht eine stromsparende Intel NUC summa summarum nicht wahnsinnig viel mehr Strom, dies allerdings bei 2 CPU-Kernen. Hochgerechnet auf die verfügbaren Kerne verbrauchen die ARM-Maschinen noch immer deutlich weniger Strom.

Was bleibt sonst? Selbst mit Intel/AMD-Maschinen im unteren Bereich lässt sich doch gänzlich einfach leben. Wenn es denn so einfach wäre? UEFI und Secure Boot gefällig? Intel-Monopol? AMD in tiefen Verlusten, schon mal an eine Alternative gedacht? Und warum gibt es erst einigermassen stromsparende CPUs von Intel/AMD, seit sich ARM bei den Handys und Tablets durchgesetzt hat? Natürlich können die paar Dutzend bis Hundert Watt Energie eines Rechners bezahlt werden, doch warum viele Watt 'verbraten', wenn es mit weniger auch geht? Ganz abgesehen vom Gewicht der Rechner. Wer will rackweise Hardware schleppen, wenn es auch mit dem Gewicht einer Tafel Schokolade geht?

Wir hätten hier mal 20 CPUs mit ca. 550 Gramm (2xODROID XU4, 1xRaspberry PI 2) gegen 2 Cores (Intel NUC) mit ca. 700 Gramm. Spasseshalber sei die Rechnung gemacht, dass in einem Rucksack die Kleinigkeit von gut und gerne 256 Rechenkernen (32 ODROIDs) Platz finden würden. Wir hätten dann 2.9 Kilogramm und (das ist jetzt gerade etwas unschön) für die Netzteile dazu ca. 4.6 Kilogramm. Macht zusammen ca. 7.5 Kilogramm für 256 CPUs.

Es gab da um 2010 herum Vorträge zur Virtualisierung mit KVM seitens des Vortragenden, da waren Server mit 4 Cores dabei, welche 10 Kilogramm auf die Wage brachten. Damit

ein Cluster gezeigt werden konnte, mussten 2 Rechner und somit 20 Kilogramm geschleppt werden. Macht bei 8 CPUs 2.5 KG pro CPU. Dieses Jahr hätten wir beim ODROID XU4 pro CPU (inkl. Anteil Netzteil) keine 30 Gramm mehr, macht im Vergleich zu 2011 um ca. den Faktor 85 weniger Gewicht pro CPU. Nicht schlecht, wenn es so weitergeht, werden es in fünf Jahren noch ca. 1 Gramm pro CPU sein; der Vortragende geht davon aus, dass diese Annahme duraus eintreffen könnte.

Und genau darum macht ARM heute und jetzt schon extrem viel Sinn, gerade auch auf dem Desktop. Weniger ist heute und wird in Zukunft ganz einfach mehr sein. Oder will jemand ernsthaft behaupten, Microsoft würde freiwillig Windows10 auf den Raspberry PI 2 portieren wollen? Viele (auch Microsoft) sprechen dabei von Internet of Things (IoT), sagen wir dazu doch einfach Alltag.

2.8 Ausser Konkurrenz

Wie bereits bei der Einleitung erwähnt, schaffte es die NVidia TV Box in letzter Minute auch noch in den Rucksack. Wir hätten damit nochmals zusätzlich ca. 500 Gramm, wobei die NVidia TV Box zusätzliche 4 bzw. 8 CPUs und 256 Grafikkerne (GPU) mitbringt, d.h. wir wären dann schon im Jahre 2015 bei ca. 2 Gramm pro CPU/GPU.

3 Testkandidaten

Nun zum praktischen Teil. Anhand einiger typischer Aufgaben sollen die drei Geräte Raspberry PI 2, ODROID XU4 und Intel NUC (kleinste Variante) miteinander verglichen werden. Weiter wird die NVidia TV Box ausser Konkurrenz antreten, da letztere 'nur' mit Ubuntu läuft.

Eines vorweg, die Kandidaten entsprechen nicht dem Maximum dessen, was gezeigt werden könnte. Ein Intel NUC mit Celeron ist schwachbürstig, das gleiche gilt aber auch für den Raspberry PI 2 und den ODROID XU4. Letztere beide Kandidaten arbeiten ja noch nicht mal mit 64 Bit. Und selbst bei der NVidia TV Box muss gesagt werden, dass diese nicht den vollen Power zeigen konnte, weil die 256 GPU-Kerne nicht angesteuert werden konnten.

3.1 Raspberry PI 2

Hardware: 1 GB RAM, 4 CPU Kerne (ARM Cortex-A7), Netzwerk: 100 MBit, Festplatte: Micro SD-Card mit 32 GByte (SanDisk), 4xUSB2. Kostenpunkt für Gerät, Netzteil, Gehäuse, 32 GB SD-Karte ca. 100 EURO.

3.2 ODROID XU4

Hardware: 2 GB RAM, 8 CPU Kerne (ARM Cortex-A7, SAMSUNG Exynos 5422), Netzwerk: 1000 MBit, Festplatte: eMMC (Version 5), 2xUSB3, 1xUSB2. Kostenpunkt für Gerät (Netzteil enthalten), Gehäuse, 32 GB eMMC-Karte: 150 EURO.

eMMC: Dieser Speichertyp wird in sämtlichen moderneren Handys verwendet und normalerweise fest auf die Platine verlötet. Beim ODROID XU4 ist dies glücklicherweise nicht der Fall, die eMMC-Datenträger werden austauschbar auf einer kleinen steckbaren Platine geliefert. Alternativ kann beim ODROID XU4 auch eine Micro-SD-Karte zum Einsatz kommen. Die SD-Karte kann von aussen zugesteckt werden. Um das Boot-Medium (eMMC oder SD-Karte) zu bestimmen, besteht ein Schalter, der von aussen zugänglich ist.

Neben der Grundversion gibt es eine erweiterte Version, mit welcher (via USB3) eine SATA-Platte direkt in einem Gehäuse (ca. 350 Gramm) angeschlossen werden kann. Der Durchsatz über USB3 liegt im Zusammenhang mit einer SSD-Platte bei ca. 140 bis 150 MByte, ohne SSD wird die Geschwindigkeit der Platte plus/minus erreicht.

Hinweis: Beim Einsatz von konventionellen Platten sollte die Stromaufnahme gering bleiben, da ansonsten der ODROID XU4 unter Volllast vom mitgelieferten Netzteil zu wenig 'Saft' erhält.

3.3 Intel NUC mit Celeron

Hardware: 8 GB RAM, 2 CPU Kerne (Intel Celeron N3050), Netzwerk: 1000 MBit, Festplatte: 500 GB SSD (Samsung EVO), 3xUSB3, 1xUSB2. Kostenpunkt für Gerät, RAM-Riegel, Festplatte (Annahme 120 GB): 250 EURO.

Installation ohne UEFI: Die ArchivistaBox wird derzeit nicht mit UEFI oder SecureBoot ausgeliefert. Im Grundzustand war es daher nicht möglich, die ArchivistaBox zu installieren. Für eine/n normale/n Anwender/in ist dies nicht ganz einfach hinzukriegen. Zwar sind die modernen BIOS schön und bunt geworden, nur übersichtlicher wurden die Menus damit nicht. Weiter weigerte sich der Intel NUC, den Stick ab FAT16 zu booten, erst das Bereitstellen eines Sticks über UNetBootin führte zum Erfolg, siehe dazu:

archivista.ch/cms/language/de/support/community/usb-stick-ab-iso-datei-windows

3.4 NVidia TV Box (mit Ubuntu, ausser Konkurrenz)

Die NVidia TV Box wird später mit einem eigenen Kapitel bedacht, weil bereits das Aufsetzen von Linux an sich ein Vortragsthema darstellen würde. Und weil die ArchivistaBox nicht auf die NVidia TV Box portiert wurde/wird, kann die NVidia TV Box auch nur ausser Konkurrenz antreten.

4 Tests und Ergebnisse

Nachfolgend müssen die Kandidaten diverse Tests über sich ergehen lassen. Natürlich liegt in der Anordnung der Tests immer eine gewisse Willkür. Vorliegend dürfte diese dadurch gegeben sein, dass die ArchivistaBox verwendet wird und dass die Tests Standardjobs umfassen, die bei der ArchivistaBox anfallen.

Dem sei hier entgegengehalten, dass für die ArchivistaBox die empfohlenen Debian-Derivate sowohl beim Rasperry PI II als auch beim ODROID XU4 verwendet wurden, beim Intel NUC handelt es sich um die üblicherweise ausgelieferte ArchivistaBox (ebenfalls Debian basierend). Diese Anordnung macht insofern Sinn, als damit nicht einfach irgendwelche Test-Images verwendet werden, welches sich nie in einen praktischen Einsatz bewähren müssen, sondern vielmehr jene Linux-Distribution, die sich nach wie vor guter Beliebtheit erfreut, und die der Vortragende mittlerweile doch sehr gut kennt.

Weiter mussten/konnten sich die Test-Kandidaten über mehrere Monate bewähren. So konnten z.B. die Optimierungen für die Parallelität erst vor einigen Wochen abgeschlossen werden. Ohne diese Optimierungen jedoch würde ein Test nicht das hergeben, was die Hardware hergibt. Und dann wiederum wären die Tests auch nicht praxisbezogen.

4.1 Neustart des Systems

Gemessen wird die Zeit zwischen dem Zeitpunkt, bei dem das System runtergefahren wird und bis zu jenem Zeitpunkt, mit dem mit SSH wieder auf das System zugegriffen werden kann.

Folgende Ergebnisse liegen vor:

- Intel NUC: 60 Sekunden
- Raspberry PI 2: 30 Sekunden
- ODROID XU4: 50 Sekunden

Erstaunlich, dass der kleinste Zwerg (Raspberry PI 2) nach 30 Sekunden sehr schnell wieder zur Verfügung steht. Allerdings ist zu beachten, dass bei der ArchivistaBox auf Basis Intel/AMD immer die gesamte ISO entpackt wird, um das Betriebssystem im RAM laufen zu lassen. Von daher sind die 60 Sekunden nicht ganz vergleichbar. Alles in allem sind aber 30 bis 60 Sekunden für den kompletten Zyklus (Runter- wie Hochfahren) ohnehin akzeptabel.

Kernaussage: Das Runter- wie das Hochfahren eines Systems sagt nur bedingt etwas über die Leistungsfähigkeit eines System aus, zu sehr hängt der Prozess von der verwendeten Software

bzw. dem Umfang der gestarteten Applikationen ab. Trotzdem stören lange Start-Zeiten merklich. Diese aber sind mit ARM-Rechnern der aktuellen Generation keinesfalls mehr zu befürchen, die Systeme starten in deutlich unter einer Minute. Erstaunlich ferner, dass der Raspberry PI 2 hier derat flott arbeitet. Etwas ketzerisch gesagt profitiert der Raspberry PI 2 wohl davon, dass dise Optimierungen bei der ersten Modellserie notwendig waren, um anständige Startzeiten zu erreichen.

4.2 Starten LibreOffice

- Intel NUC: 3 Sekunden (ab Hauptspeicher)
- Raspberry PI 2: 10 Sekunden (ab SD-Card)
- ODROID XU4: 5 Sekunden (ab eMMC)

Die Werte liegen im Rahmen dessen, was erwartet werden kann. 5 Sekunden für LibreOffice auf dem ODROID XU4 sind ganz flott. eMMC ist ja auch bekannt dafür, dass das Starten von Applikationen flink (über 100 MByte/Sekunde Durchsatz) von sich geht.

Kernaussage: Auch das Starten von LibreOffice eignet sich nur bedingt, um eine Aussage über die Geschwindigkeit zu machen. Es konnte jedoch festgestellt werden, dass LibreOffice einmal gestartet keine Probleme bereitet. Wer flüssig mit Office-Applikationen arbeiten möchte/muss, der ist mit dem ODROID XU4 sehr gut bedient, selbst der Raspberry PI 2 schlägt sich tapfer. Die schnelle Startzeit beim Intel NUC beruht daher, dass LibreOffice komplett im Hauptspeicher installiert ist (siehe Hinweise zum Starten). Die gleichen Aussagen zum Arbeitsverhalten können plus/minus für das Surfen im Web gemacht werden, wobei hier der Raspberry PI 2 am ehesten ans Limit kommt.

4.3 Arbeiten mit Bildern (Rotieren)

Nachfolgend geht es darum, die Seite 111 des deutschen ArchivistaBox-Handbuches um 90 Grad zu rotieren. Dazu wird zunächst einmal der Befehl 'convert' verwendet:

```
convert seite111.png -rotate 90 seite111.tif
```

Die folgenden Ergebnisse erhalten wir:

- Intel NUC: 0.483 Sekunden
- Raspberry PI 2: 2.999 Sekunden
- ODROID XU4: 0.867 Sekunden

- NVidia TV: 0.367 Sekunden

Auf den ersten Blick erscheinen die Resultate plausibel, und dennoch 'wurmt' es etwas, dass der ODROID XU4 soviel langsamer ist, als der Intel NUC. Die ausser Konkurrenz antretende NVidia arbeitet bereits schneller als der INTEL-Rechner. Um die Resultate zu verifizieren, wurde das von der ArchivistaBox verwendete econvert verwendet. Das Programm kann mit Debian wie folgt installiert werden:

```
apt-get install exactimage
```

Der Aufruf ist wie folgt abzusetzen:

```
econvert -i seite111.png --rotate 90 -o seite111.tif
```

Folgende Ergebnisse werden erreicht:

- Intel NUC: 0.163 Sekunden
- Raspberry PI 2: 0.575 Sekunden
- ODROID XU4: 0.106 Sekunden
- NVidia TV: 0.093 Sekunden

So, und nun sieht die Sache schon ganz anders aus. Zunächst einmal erreicht der Raspberry PI 2 fast den Speed des Intel NUC mit dem Program 'convert'. Damit ist auch gesagt, dass 'convert' kein guter Kandidat ist, um Bilder zu manipulieren. Erstaunlich ist nun aber, dass mit econvert nun der Intel NUC plötzlich deutlich langsamer ist als der ODROID XU4. Im Grundsatz sagt auch dies nur insofern etwas aus, als dass econvert (wie fast alle Programme zur Manipulation von Bildern) die Aufgaben an die Standard-Bilbiotheken weiterreichen (hier z.B. libpng bzw. libtiff). Es ist anzunehmen, dass diese Bibliotheken bei den ARM-Rechnern in effizienter Form ausgeliefert werden. Somit lässt sich feststellen, auch eine einzelne CPU des ODROID XU4 braucht sich nicht hinter einem Intel-Prozessor zu verstecken.

Kernaussage: Wenn wir Software testen bzw. miteinander vergleichen, müssen wir uns immer im Klaren sein, dass die Umstände doch sehr entscheidend sind. Die Erfahrung des Vortragenden ist die, dass mit der Optimierung bei der Software in der Regel um Faktoren mehr herausgeholt werden kann, als dies der Fall ist, wenn ohne Optimierung einfach zur nächst schnelleren Hardware gewechselt wird. Natürlich ist es für Anwender/innen eher nicht machbar, die Software selber zu kompilieren bzw. derartige Verbesserungen vorzunehmen, doch müssen sich Anwender/innen immer bewusst sein, dass mit Software meistens mehr herausgeholt werden könnte, als dies der Fall ist, wenn zu einem neuen Rechner gegriffen wird.

4.4 Verarbieten PDF-Dateien

Verarbeitet (importiert) werden sollen jeweils die Handbücher der ArchivistaBox (205 Seiten in Deutsch und 204 Seiten in Englisch) über den Office-Importer, d.h. die beiden PDF-Dateien werden in den Ordner 'office/archivista' des Netzlaufwerkes gelegt. Folgende Ergebnisse konnten gemessen werden:

- Intel NUC: 4 Minuten 32 Sekunden
- Raspberry PI 2: 13 Minuten 16 Sekunden
- ODROID XU4: 3 Minuten 46 Sekunden

Obwohl der Intel NUC viermal mehr RAM und eine SSD-Festplatte (Durchsatz 450 MByte) besitzt, liegt der ODROID XU4 schon mal deutlich vorne. Abgeschlagen der Raspberry PI 2, der im Schnitt in etwa viermal so lange für den entsprechenden Job benötigt.

Nun wird bei der Standardeinstellung der ArchivistaBox nur jeweils die Hälfte der CPUs für die Jobverarbeitung freigegeben. Schalten wir daher um auf eine agressivere Einstellung, bei der alle CPUs freigegeben werden, dann ergeben sich die folgenden Messungen:

- Intel NUC: 3 Minuten 18 Sekunden
- Raspberry PI 2: 9 Minuten 41 Sekunden
- ODROID XU4: 2 Minuten 51 Sekunden

Die Machtverhältnisse bleiben die gleichen, ODROID XU4 liegt vorne. Es kann die Frage gestellt werden, warum nicht mehr Geschwindikeitszuwachs resultiert, wo doch der einzelne Job alle CPUs erhält. Dies hat den einfachen Grund, dass die Jobs sich z.T. überlagern, d.h. schon bei der ersten Messung war es nicht so, dass nur immer die Hälfe der CPUs arbeitete und zweitens werden auch beim Zuweisen aller CPUs für einen Job die CPUs nicht 'agressiv' bewirtschaftet, denn auch hier gilt, die Benutzer/innen sollen mit dem System ja noch arbeiten können.

Kernaussage: Wenn in 2 Minuten 51 Sekunden (sprich 171 Sekunden) 409 Seiten verarbeitet werden könne, dann ergibt dies hochgerechnet pro Tag 86400/171*409 Seiten = 206545 Seiten, die ohne Probleme und mit mässiger Last verarbeitet werden können. Natürlich können aktuell mit den schnellsten Intel/AMD-Rechnern noch mehr Seiten verarbeitet werden, genau so wie die Optimierung auf der ArchivistaBox nicht abgeschlossen ist. Kurz und gut, solche Zahlen waren noch vor ein zwei Jahren undenkbar für eine ARM-basierte Lösung. Der erreichte Wert liegt ferner weit über dem, was mit der ersten ArchivistaBox im Jahre 2005 erreicht wurde. In diesem Sinne hat der ODROID XU4 ein hohes Potential.

4.5 Rastern von Bildern mit Ghostscript (Parallelität)

Dieses Beispiel ist als Spezialfall des vorangegangen Beispiels zu betrachten. Diesmal messen wir nur noch die Geschwindigkeit, welche benötigt wird, um die Seiten zu rastern. Dazu verwenden wir Ghostscript. Weil nun Ghostscirpt selber nicht in der Lage ist, die Seiten parallel abzuarbeiten, gibt es unter Linux das Konsolenprogramm 'parallel'. Der Aufruf sieht etwas kryptisch aus:

```
seq 1 204 | parallel -P 8 \
gs -dNOPAUSE -dNOPROMPT -dBATCH -q -sDEVICE=tiffg4 -r300x300 \
-dFirstPage={} -dLastPage={} \
-sOutputFile=/in/documentation_en_{}.tif \
/in/documentation_en.pdf
```

Damit beauftragen wir **gs** die Seiten einzeln zu rastern, sagen aber gleichzeitig, dass das Programm **parallel** immer dann die nächste Seite automatisch verarbeiten soll, wenn eine Seite beendet werden konnte. Dies im übrigen solange, bis die Seiten 1 bis 204 abgearbeitet sind. Mit dem Flag **-P 8** bestimmen wir die Anzahl der Prozessoren (ODROID XU=8, Raspberry PI 2=4, Intel NUC=2). Wie gesagt, für Anwender/innen mag dies etwas kryptsich erscheinen, die Profis sollten das Beispiel genauer betrchten, lassen sich damit doch viele konventionelle Jobs mit einem Einzeiler parallel abarbeiten. Doch nun zu den Ergebnissen:

- Intel NUC: 1 Minute 8 Sekunden
- Raspberry PI 2: 2 Minuten 30 Sekunden
- ODROID XU4: 28 Sekunden

Zum Vergleich sei die gleiche Datei als Single-Core-Anwendung auf unserer üblicherweise verbauten CPU für die ArchivistaBox Matterhorn gestartet. Dies deshalb, weil bis vor wenigen Wochen sämtliche Jobs in Single-Core-Ausführung gestartet wurden. Wir erhalten dabei eine Ausführungszeit von 52 Sekunden. Exact diese Zeit erhalten wir im übrigen auch mit der NVidia TV Box, womit klarerweise festgestellt werden kann, wie gut die NVidia TV Box mithalten kann.

Kernaussage: Der schnellste Multi-Core-Prozessor bringt nur soviel, wie die Software damit umzugehen weiss. Der ODROID XU4 schlägt dabei selbst unser bisheriges Flaggschiff ganz erheblich, sofern auf dem bisherigen Flagschiff nicht parallel gearbeitet wird. Auch dazu noch ein kleines Rechenbeispiel: 86400 Sekunden (Tag) / 28 * 204 = 629340 gerasterte Seiten ab PDF-Dateien pro Tag. Ist doch gar nicht mal so schlecht für einen Rechner, der knapp 100 Gramm wiegt?

Nachbemerkung: Werden sämtliche vier CPU-Kerne auf dem bisherigen Flaggschiff angeworfen, so resultiert auf dem 'Flagschiff' eine Verarbeitungszeit von 15 Sekunden, womit wir analog dazu gerechnet eine Tagesleistung von 1.175 Millionen gerasterte PDF-Seiten hätten. Wenn denn das nächste Mal der Entwickler eine schnelleren Maschine verlangt, dann soll die gute Fachkraft sich mal in Parallelität einarbeiten. Das kostet möglicherweise zwar auch ein paar Tausender, die für die schnellste Intel-CPU derzeit hingelegt werden müssten. Nur kann dabei eine Software entstehen, die um höhere Faktoren bessere Ergebnisse erzielt.

Fairerweise muss hier angefügt werden, dass diese Technologie im Hause Archivista schon seit länger fallweise zum Einsatz gelangte, jedoch nicht für die Standardprozesse. Der ODROID XU4 hat hier massebend dazu beigetragen mehr herauszuholen. Wenn von 1 CPUs gerade mal eine läuft, dann fällt dies bei 2 CPUs (bisherige Entwicklungsmaschine) kaum auf, bei 8 Kernen ist es derart offensichtlich, dass es einem 'zwingt', etwas dagegen zu unternehmen.

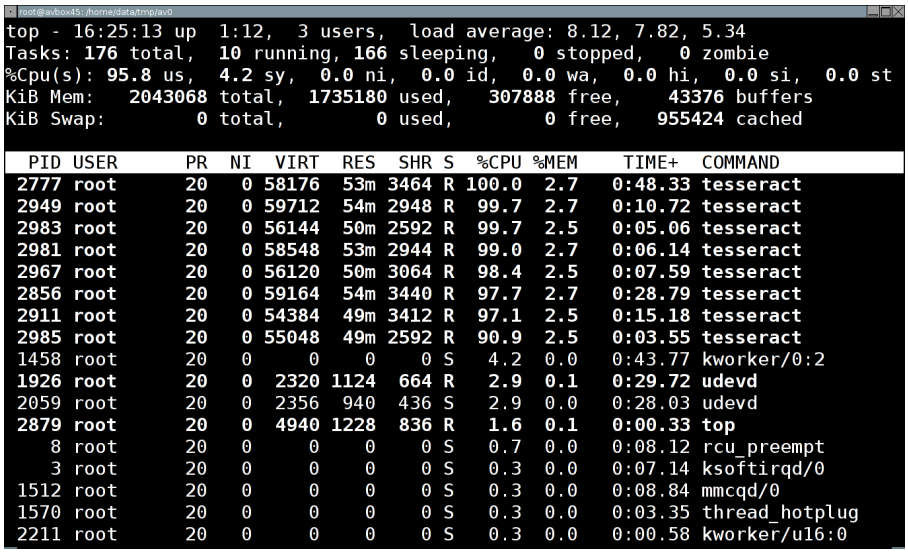
4.6 Texterkennung mit Tesseract

Natürlich wird nicht jede/r Benuter/in Texterkennung verwenden, anstelle von Textverarbeitung passend wäre sicher auch Videoschnitt oder Spracherkennung. Aber, die Texterkennung bildet nun mal das Herzstück der ArchivistaBox, und Tesseract gilt als sehr ressourcenhungrig. Die besten Voraussetzungen, um damit Tests zu fahren. Auch hier werden die beiden Handbücher komplett in die ArchivistaBox importiert (Ordner pdf/archivista/deu bzw. pdf/archivista/eng). Die Unterordner in PDF-Ordner sind notwendig, damit Tesseract weiss, dass wir einmal das deutsche Handbuch und einmal das englische Handbuch verarbeiten möchten, dazu später mehr.

Hinweis: Auf den ersten Blick mag eine solche Testanordnung wenig Sinn ergeben. Warum soll eine am Computer erstellte PDF-Datei mit Texterkennung abgearbeitet werden? Ganz einfach, weil viele am Computer erstellte PDF-Dateien den Suchtext nicht optimal zur Verfügung stellen, dies kommt in der Praxis doch recht häufig vor (z.B. auch beim deutschen Handbuch, LaTeX weigert sich bis heute die Umlaute anders als mit **a**¨ darzustellen). Nun zu den Zahlen:

- Intel NUC: 36 Minuten 2 Sekunden
- Raspberry PI 2: 1 Stunde 36 Minuten
- ODROID XU4: 15 Minuten 57 Sekunden

Klarer Sieger ist der ODROID XU4, dies dürfte aufgrund der 8 Kerne auch nicht überraschen. Bei Tesseract werden die 8 Kerne des ODROID XU4 endlich mal ausgelastet:



Aus den Zahlen ersichtlich ist, dass der ODROID XU4 plus minus ca. 2.3 mal so schnell arbeitet wie der Intel NUC. Wird die reine Zeit für die OCR-Erkennung berechnet, so resultiert auf dem ODROID XU4 eine Erkennungszeit pro Seite von knapp über 1.9 Sekunden, beim Intel NUC sind es ca. 4.2 Sekunden. Damit wird der ODROD XU4 damit nicht mit der schnellsten Intel i7 mithalten können, immerhin kann der ODROID XU4 aber etwas über 44'000 Seiten pro Tag verarbeiten.

Hinweis: Äpfel sollten nicht mit Birnen verglichen werden. Die schnellste Intel i7 oder XEON kostet im Minimum ein paar Hundert, wenn nicht Tausende EURO (ohne Board, Speicher, Gehäuse etc). Dagegen kostet der ODROID XU4 samt allem Zubehör ca. 100 EURO. Und noch etwas sei hier erwähnt, würde Tesseract auf der Intel i7 mit einer CPU, mit mehreren Sprachen, Seitenausrichtung, Tabellenerkennung und Säubern des Textes gestartet, so wäre der ODROID XU4 auch dem schnellsten Prozessor haushoch überlegen, kurz und gut, die Software machts!

Abgeschlagen, wer wundert sich, der Raspberry PI 2, hier ist ganz einfach die CPU zu 'leistungsschwach', um mitreden zu können. Erstaunlich sind die Resultate insofern, dass der ODROID XU4 um ca. den Faktor 6 schneller arbeitet als dies beim Raspberry PI 2 der Fall ist. Diesbezüglich ist der ODROID (kostet als Gesamtgerät ja nur ca. 40 EURO mehr) die weit bessere Wahl.

Kernaussage: An diesem Beispiel gut ersichtlich ist, dass der Einsatz von mehreren CPUs sehr viel Potential bietet. Auf die ArchivistaBox bezogen bedeutete dies, dass selbbst die schnellste ArchivistaBox mit vier Kernen bisher aufgrund der fehlenden parallelen OCR-Erkennung bei Tesseract weit hinter den Ergebnissen des ODROID XU4 blieb. Selbstverständlich stehen diese Optimierungen nun für sämtliche ArchivistaBoxen zur Verfügung. Es bleibt dem hinzuzufügen, dass noch heute, im Jahre 2015, viele Anwender/innen mit Desktop-Geräten nur minim von

den vorhandenen CPUs (eine QuadCore hat ja heutzutage fast jede/r) profitieren, weil die Software nicht darauf vorbereitet ist, parallel zu arbeiten.

Tipp: Mit den richtigen Optionen kann fast immer nochmals (neben der Parallelität) ein hohes Mass an Geschwindigkeitsgewinn erzielt werden. Dazu ein Beispiel: Ein englischer Text mit zugewiesener deutscher Sprache bei Tesseract führt z.B. dazu, dass die Erkennung ca. 50 Prozent mehr Zeit in Anspruch nimmt (einmal abgesehen davon, dass die Ergebnisse schlechter ausfallen). Konkret ist dies dem Vortragenden selber passiert, weil mal schnell beide Handbücher (Deutsch wie Englisch) in den gleichen Verarbeitungsordner kopiert wurden (darum die unterschiedlichen Verarbeitungsordner zu Beginn diess Abschnittes).

4.6.1 Ausser Konkurrenz

Die NVidia TV Box schlägt sich wacker, für das Erkennen einer Seite (es wurde die Seite 111 aus dem deutschen Handbuch verwendet) mit einer CPU wird eine Zeit von 15.5 Sekunden gemessen. Zum Vergleich, der ODROID XU4 benötigt dafür 20.8 Sekunden und die mit 3.3 GHz getaktete AMD-CPU, welche üblicherweise in der ArchivistaBox Matterhorn verbaut wird, erreicht 9.6 Sekunden. Die gemessene Zeit der NVidia TV Box ist umso eindrücklicher, wenn bedenkt wird, dass die NVidia TV Box ja an sich auf 256 GUP-Kerne zurückgreifen könnte. Schade, dass NVidia hier z.B. OpenCL nicht unterstützt.

4.7 Kompilieren (Tesseract)

Bislang wurde unter ARM oft zur Cross-Kompilierung gegriffen, d.h. der Quellcode grösserer Projekte wird auf einem leistungsfähigeren Rechner für eine bestimmte Plattform übersetzt. Derartige Laufzeitumgebungen sind nicht ganz einfach aufzubauen, das direkte Übersetzen auf dem Laufsystem ist die einfachere Wahl, sofern dies nicht um den Preis erkauft werden muss, dass es unendlich lange dauert.

Mit dem Raspberry PI 2 sind derartige Jobs recht gemächlich, ein zwei Stunden für grössere Projekte sind keine Seltenheit, beim ODROID XU4 läuft die ganze Sache recht flott ab. Tesseract besteht bekanntlich aus der Bibliothek 'leptonica' und dem eigentlichen Texterkennungsprogramm. Das Übersetzen von 'leptonica' benötigt 2 Min 25 Sekunden, für Tesseract fallen nochmals 4 Min 3 Sekunden an. Auf die ArchivistaBox bezogen: All jene Pakete, welche nicht von Debian direkt übernommen werden konnten, wurden direkt auf dem ODROID XU4 übersetzt.

Tipp: Weil sowohl der Raspberry PI 2 als auch der ODROID XU4 die gleichen CPU-Befehlssätze haben (ARMv7), können auf dem ODROID XU4 erstellte Programme auf dem Raspberry PI 2 ausgeführt werden. Doch aufgepasst, derartige Programme laufen nicht auf dem Raspberry

PI 1, da dieser nur über ARMv6 verfügt. Dagegen kann ein auf dem Raspberry PI 1 erstelltes Programm auf allen drei Rechnern (Raspberry PI 1 und 2 sowie ODROID XU4) laufen.

Kernaussage: Auch wenn das Übersetzen von Quellcode nicht umbedingt das ist, wonach sich Anwender/innen sehnen, so dürften Linux-Anwender/innen dann und wann halt doch beim Kompilieren landen, dies gilt für ARM- wie Intel/AMD-basierte Distributionen. Gerade im Falle von Tesseract, sind die Fortschritte enorm, sodass niemand warten sollte, bis jemand das passende Paket gebaut hat. Das Kompilieren von Quellcode ist an sich nicht schwierig. Folgende Befehle werden plus/minus auf einem Terminal (xterm) mit Root (su auf xterm eingeben) benötigt:

```
apt-get install build-essential
tar xvf xy.sourcen.tar.gz
cd xy.sourcen
./configure --prefix=/usr
make -j 8
make install
```

Dem sei hier hinzugefügt, dass mit **`make -j 8`** die Anzahl der vorhandenen CPUs angesprochen wird. In Anleitungen wird meist nur 'make' empfohlen, und dies ist bei acht CPU-Kernen recht jammerschade, denn ohne **`-j 8`** benötigt das Übersetzen von Tesseract beinahe 50 Minuten. Anhand von Tesseract sollen die obigen Ausfhührungen in die Tat umgesetz werden. Beziehen wir über Github die aktuelle Tesseract-Version (es können dort auch ältere Versionen bezogen werden):

github.com/tesseract-ocr/tesseract/archive/master.zip

Weiter benötigen wir die Bibliothek Leptonica, diese finden wir hier:

www.leptonica.com/source/leptonica-1.72.tar.gz

Nun entpacken wir diese beiden Datei:

```
unzip tesseract-master.zip
cd tesseract-master
mv ../leptonica-1.72.tar.gz
tar xvfz leptonica-1.72.tar.gz
cd leptonica-1.72
./configure --prefix=/usr
make -j 8
make install
cd ..
./configure --prefix=/usr
make -j 8
```

```
make install
```

Es kann sein, dass dabei einige Bibliotheken nachgeladen werden müssen, dabei einfach die Namen der fehlenden Pakete in einer Suchmaschine eingeben und danach mit **`apt-get install xyz`** installieren.

4.7.1 1'000 Bestellungen anlegen

ArchivistaERP bietet sich wunderbar an, um einen Job zu starten, der die Leistungsfähigkeit im Zusammenhang mit Datenbanken und Web-Services aufzeigt. In ArchivistaERP werden daher 1'000 Bestellungen eröffnet, folgende Zeiten:

- Intel NUC: 28 Sekunden
- Raspberry PI 2: 1 Min 15 Sekunden
- ODROID XU4: 19 Sekunden

Diese Zahlen gelten für eine CPU. Werden die Jobs parallel gestartet, kann im Faktor der vorhandenen CPUs hochskaliert werden. Dies ergibt folgende Tagesleistungen:

- Intel NUC: 4.88 Mio
- Raspberry PI 2: 3.64 Mio
- ODROID XU4: 28.8 Mio

Kernaussage: Dies sind recht eindrückliche Zahlen, es fragt sich, weshalb ERP-Lösungen gut und gerne Dutzende von GByte an RAM erfordern, einfacher ginge wohl. Gerne sei dabei auf den Vortrag vom letzten Jahr verwiesen: www.archivista.ch/de/media/erp.pdf

Auf Seite 28 wird dort ausgeführt, dass bereits mit dem Raspberry PI 1 pro Tag immerhin fast 500'000 Bestellvorgänge abgewickelt werden können. Um es etwas pointiert zu formulieren, wenn es nur darum ginge, übliche Business-Prozesse einer Firma abzubilden, so würden heute Rechner wie der Raspberry PI längst ausreichen. Dass dem dennoch nicht so ist, dürfte wohl damit zusammenhängen, dass Software ein virtuelles (nicht sichtbares) Gut ist. Dies ganz nach dem Motto: Ein Rack-Server muss doch leistungsfähig(er) sein als eine kleine Box. Die andere Erklärung wäre, bei der Software würde allenthalben 'geschlampt', wir wollen aber nicht 'lästern', sondern nachfolgend lieber Betriebssysteme 'rastern'.

5 Virtualisierung und mehr

5.1 QEMU

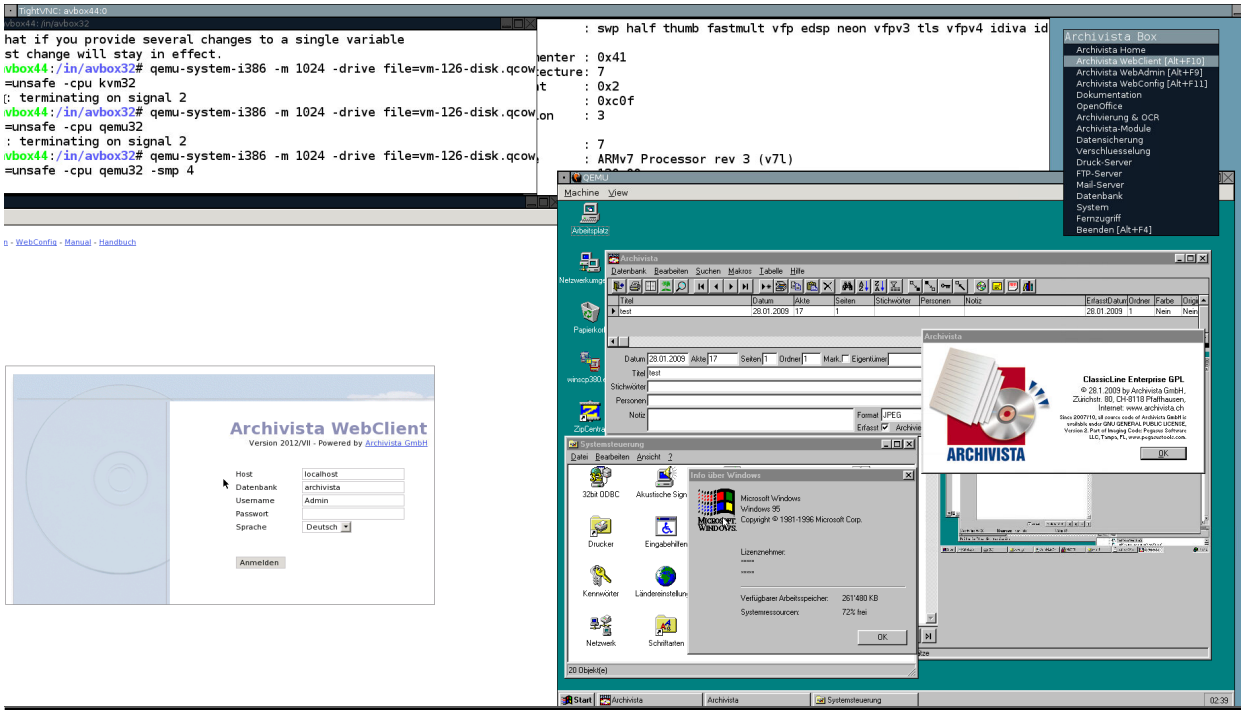
QEMU lässt sich einfach installieren. Wenn es darum geht, i386-Systeme aufzusetzen, dann kann qemu wie folgt installiert werden:

```
./configure --target-list=i386-softmmu --prefix=/usr
make -j 8
make install
```

Eine Windows95-Instanz z.B. kann wie folgt gestartet werden:

```
qemu-system-i386 -m 256 -net user -net nic,model=pcnet win95.qcow2 \
  -usb -usbdevice tablet -redir tcp:5905:10.0.2.15:5900 -vga cirrus
```

Die Geschwindigkeit mit Windows 9.x lässt kaum Wünsche offen, dies dürfte nicht für spätere Windows-Version der Fall sein.



Kernaussage: Qemu (unter ARM wie anderswo) zeigt sehr schön, wohin die Reise gehen wird. Zwar können derzeit einigermassen aktuelle Intel/AMD-Installation nicht vernünftig ausgeführt werden. Immerhin gelang es aber, eine 32-Bit-ArchivistaBox aus dem Jahre 2010 komplett zu starten. Letztlich wird es der einzige Weg über Plattformen hinweg sein, 'alte' Software bzw. Betriebssysteme zu starten.

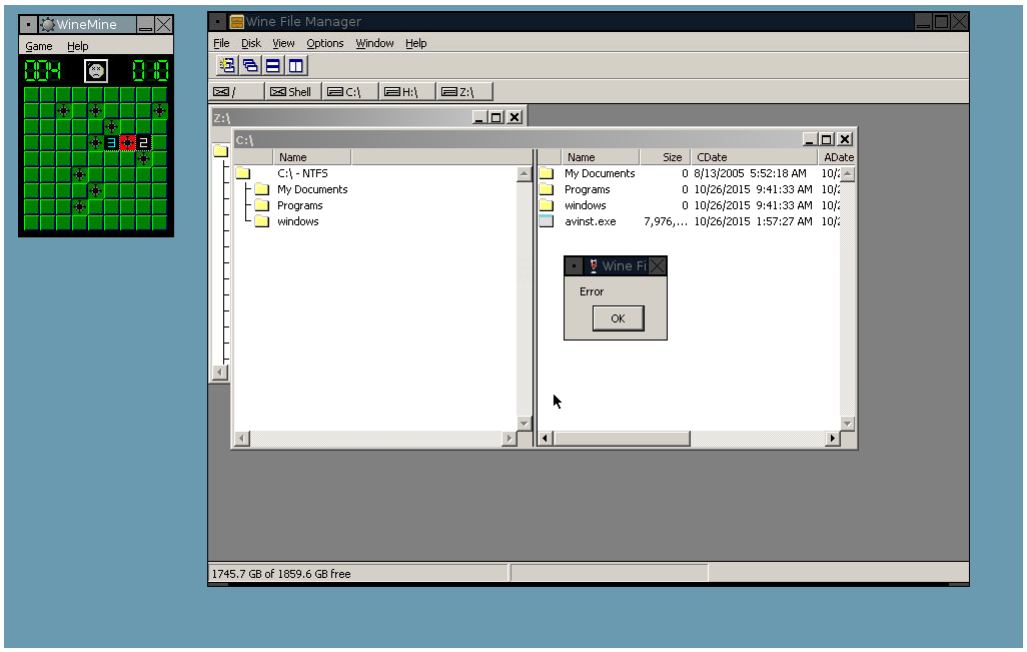
5.2 KVM

Es gibt vom Hersteller Hardkernel eine Anleitung zum Aufsetzen von KVM, und es konnte auch ein Ubuntu in der ArchivistaBox (aka Debian) gestartet werden. Es ist aber derzeit nicht so, dass einfach ein physikalisches Image genommen werden und mal schnell mit KVM gestartet werden könnte. Vielmehr müssen die entsprechenden Kernel zwischen Host und Gast aufeinander abgestimmt sein.

Kernaussage: Wenn Virtualisierung bedingt, dass es nur geht, wenn bstimmte Kernel mit bestimmten modifizierten Images laufen, dann bringt Virtualisierung auf Stufe Hardware kaum etwas. Unter den ARM-Rechnern erschwerend hinu kommt, dass RAM nicht einfach aufgerüstet werden kann. Der Vortragende wünschte sich hier z.B. Festplatten-Images, die mit **`kvm image.raw`** gestartet werden könnten.

5.3 Wine

Viele dürften die Situation noch kennen, wir hätten da eine alte Windows-Applikation. Virtualisierung anwerfen, dauert halt eine Weile. Mal schnell eine Windows-Applikation unter Linux zu starten, das wärs. Genau dies mach Wine (mit gewissen Einschränkungen) möglich. Allerdings ist Wine eine für Intel/AMD-Prozessoren ausgelegte Software, welche die Systemaufrufe von Windows unter Linux zur Verfügung stellt. Nun lässt sich Wine dennoch problemlos auf dem ODROID XU4 kompilieren, siehe dazu die untenstehende Abbildung:



Allerdings muss hier angeführt werden, dass damit keine Windows-Applikationen gestartet werden können, sodass derzeit ausser Minekeeper wenig übrigbleibt. Um echte Windows-Applikationen starten zu können, müsste (nach ca. 2 Stunden wurde der Versuch aufgegeben)

qemu zusammen passend mit einer unter 32-Bit-kompilierten und gepachten Wine-Version installiert werden. Als Alternative wird für ARM-Rechner ExaGear Desktop empfohlen, allerdings handelt es sich dabei um eine Closed-Source-Applikation, siehe dazu:

eltechs.com/product/exagear-desktop/

Kernaussage: Wer zu einem ARM-Rechner greift, darf/sollte sich bewusst sein, dass ARM keine Intel/AMD-Plattform darstellt. Wer darauf Wert legt, dürfte mit einem Intel NUC (die Preise sind mittlerweile auch da ja nicht mehr hoch) besser fahren.

5.4 OpenCL

OpenCL bedeutet, dass gewisse (rechenintensive) Jobs an die Prozessoren der Grafikkarte ausgelagert werden. Nach viel Mühe und Arbeit konnte OpenCL in Verbindung von Tesseract eingerichtet werden, um danach jedoch 'frustriert' feststellen zu müssen, dass die GPUs (Grafikprozessoren) des ODROID XU4 eben gerade keinen Vorteil in Sachen Geschwindikeit bringen, siehe dazu:

`http://forum.odroid.com/viewtopic.php?f=98&t=16475`

Kernaussage Bei Rechnern, welche nu wenige Kerne in der Grafikeinheit mitbringen, dürfte der Einsatz von OpenCL wenig bringen. Anders könnte es bei Prozessoren aussehen, die sehr viele GPU-Graifkkerne beinhalten, allerdings ermutigen entsprechende Anleitungen für die entsprechenden Modelle (z.B. Tegra K1 oder X1) nicht unbedingt zum Nachahmen. Entweder ist die Unterstützung inexistent oder dann abhängig zu gewissen Applikationen. Schmerzhaft ist dies bei der NVidia TV Box, da diese ja 256 GPU-kerne enthält und dafür ausserordentlich prädestiniert wäre.

5.5 Container

Bei einem Preis von 74 USD fragt sich, wieviel Zeit für den Aufbau von Containern verwendet werden kann/sollte. Nach ca. 4 Stunden wurde die Übung 'Linux-Instanzen' sowohl mit docker als auch lxc abgebrochen. Es gibt zwar für den ODROID XU4 speziell erstelle Linux-Distributionen, welche getestet werden können, doch war dies nicht das Ziel. Es ging dem Vortragenden vielmehr darum, wie einfach es wäre, dies mit der bestehenden Linux-Umgebung (Debian) zu ereichen, und hier klappte es nicht.

Kernaussage Wer heute auf einen ARM-Rechner setzt, sollte sich bewusst sein, dass Container-basierte Lösungen nicht out-of-the-box arbeiten. Dies gilt umso mehr dann, wenn er/sie nicht bereit ist, die gesamte Installation bzw. Distribution mal schnell auszutauschen.

6 Stolpersteine bei ARM

6.1 Qualität SD-Karten

Auch wenn die Hersteller von SD-Karten es noch so beteuern, SD-Karten (insbesondere Micro-SD-Karten) sind schnell(er) defekt als einem lieb ist. Aus diesem Grunde ist der Einsatz von eMMC beim ODROID XU4 deutlich angenehmer. In den ca. 4 Monaten gab es bisher keine Probleme, dies kann so beim Raspberry PI 2 bzw. den verwendeten SD-Karten nicht gesagt werden.

Empfehlung: Der Griff zu einer sehr günstigen Micro-SD-Karte kann sich fatal auswirken, die Fehler sind nicht in jedem Falle offensichtlich feststellbar, am ehesten noch auf einem Terminal mit **`dmesg`**. Daher der Tipp, lieber ein paar EURO für eine gute SD-Karte mehr ausgeben. Nach Erfahrung des Vortragneden kann über den Daumen gepeilt schon gesagt werden, dass die Probleme mit den günstigsten Karten existierten, mit den teureren Karten dagegen nicht.

6.2 Hitze bzw. Lüftung

Nur weil kein Lüfter verbaut ist, heisst dies noch lange nicht, dass nicht doch die durch die CPUs abgegebene Wärme ein Problem darstellen kann. Gerade beim Raspberry PI 2 kann dies im Zusammenhang mit Tesseract beobachtet werden. Ein Dauerbetrieb über alle vier CPUs führt dazu, dass die Geschwindigkeit bei der Erkennung nachlässt. Dies deshalb, weil die Wärmeentwicklung dazu führte, dass der CPU-Taktfrequenz gedrosselt wurde.

6.3 Netzteil beim ODROID XU4

Das Netzteil beim ODROID XU4 wird mitgeliefert. Leider lässt sich das Teil (zumindest an Schweizer Steckdosen) mitunter nur schwerlich anstöpseln. Alternative Netzteile können empfohlen werden, sollte der ODROID oft an wechselnden Standorten zum Einsatz kommen.

6.4 Durchstarten statt Ausschalten

Mit der verwendeten Debian Wheezy für den ODROID XU4 kam es bei einem Ausschalten immer zu einem Reboot. Solche bzw. ähnliche 'Problemlchen' gab/gibt es leider ab und wann. Abhilfe auf die harte Tour: **`/etc/initd./mysql stop;poweroff`**. Abhilfe auf die noch hartere Tour, siehe dazu:

http://forum.odroid.com/viewtopic.php?f=111&t=11325

Empfohlen wird darin der Austausch der U-Boot-Datei im Boot-Ordner (erfordert etwas Suchen, wo diese Datei liegt und hängt von der Distribution ab). Wer beide 'Touren' nicht mag, möge dann halt nach **shutdown now -h** nach einigen Sekunden den Stecker ziehen. Für nicht Linux-Anwender/innen mögen solche Geschichten 'abstrus' klingen, doch wie war das gleich nochmals, wenn der USB-Port am PC nicht mehr arbeitet: Ausschalten Maschine, Stecker ziehen, Einige Sekunden warten (bei neueren Boards bis zu einer Minute), dann Maschine wieder einschalten und hoffen, der USB-Port komme wieder... Auch das ist an sich abstrus und lässt sich mit Software bis heute nicht beheben.

6.5 Signierte uBoot-Dateien

odroid.com/dokuwiki/doku.php?id=en:exynos4412bootsequence

*** You need to sign the bl2.bin with Hardkernel's private key to make it bootable.**

Vorliegend geht es darum, dass die erstellte Datei bl2.bin (wird beim Kompilieren eigener Kernel erstellt) durch den Hersteller signiert werden müssen. Wie war dies doch gerade mit UEFI und SecureBoot? Immerhin, der Hersteller Hardkernel signiert die erstellten Dateien zwar kostenfrei, doch dauert dies bis zu 48 Stunden, nicht unbedingt das, was gewünscht ist, um einen neuen Kernel zu erstellen.

6.6 'Verknüppelte' Hardware

Der grösste Nachteil der ARM-Boxen liegt derzeit wohl darin, dass fast alle ARM-Hersteller in irgendeiner Art und Weise die Lauffähigkeit des Systems mit Binär-Dateien 'verknüppeln'. Beim ODROID XU4 stellt dies kein wirkliches Hindernis dar, bei der NVidia TV Box werden fundierte Kenntnisse benötigt, um überhaupt ein Linux aufspielen zu können. Bei der NVidia TV Box (dies gilt ebenfalls für das angekündigte Jetson TX1-Entwicklerboard) kommt überdies ein stark modifizierter 3.10-Linux-Kernel zum Einsatz. Zwar sind die Sourcen bei NVidia publiziert, allerdings konnte der Vortragende damit auch nach ca. 10 Stunden (sämtliche Foren von hinten wie vorne durchgestöbert) noch immer keinen eigenen Kernel backen. Trotzdem soll die NVidia Shield TV Station begutachtet werden, weil damit gezeigt werden kann, was mit 64 Bit möglich ist.

7 Nvidia Shield TV Station

ARM-Systeme mit 64 Bit gibt es bislang fast keine. Obwohl solche Boards immer wieder angekündigt werden (z.B. AMD oder Cavium), folgen diesen Ankündigungen bislang keine Taten, von den nicht sehr leistungsfähigen Boards bei www.96boards.org einmal abgesehen. Einzig die NVidia TV Shield Box ist (seit Oktober 2015) für 'Normalsterbliche' erhältlich. In Deutschland können die entsprechenden TV-Konsolen für 199.– EURO erworben werden. Daher soll diese Box hier 'kurz' (das Experiment kostete einige Tage) besprochen werden.

7.1 Eckdaten der Box

Die Eckdaten der NVidia TV Shield sind: 64 Bit Octa-Core (4xA53+4xA57, dazu später mehr) mit 2 GHz Taktfrequenz. Ausgestattet ist das Gerät mit 3 GB RAM sowie 16 GB eMMC-Karte. Es gibt ein erweitertes Modell zu 299.– EURO, das mit einer 500 GB SSD-Platte bestückt ist. Bei einigen der erweiterten Modellen soll es Probleme mit den verbauten Platten geben; daher fiel die Entscheidung klarerweise auf das kleinere Modell.

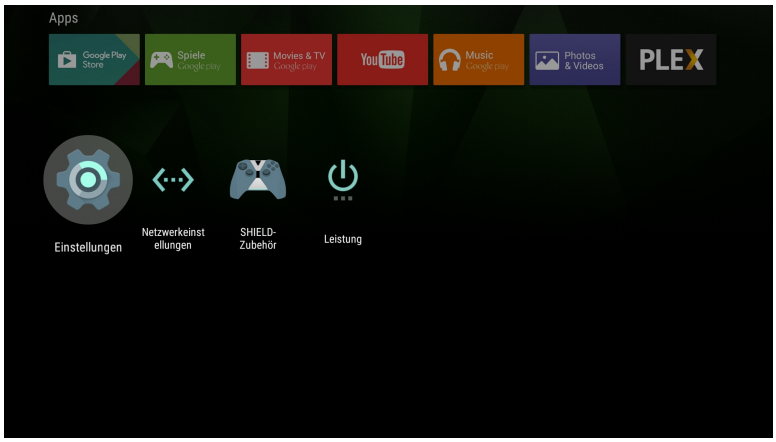
Hinweis: Ein nachträglicher Einbau einer Platte ist nicht möglich, da dazu ein Board fehlt, um überhaupt die Platte anhängen zu können. Dagegen kann eine Micro-SD-Karte verwendet werden. Ebenfalls stehen 2 USB3-Anschlüsse, ein Micro-USB-Anschluss sowie ein 1 GBit-Netzwerkanschluss zur Verfügung. Weiter an Board sind WiFI und Bluethooth.

Zum Lieferumfang gehören einige Kabel sowie ein Gamepad. Die Box wird ja nicht nur als IP-basierter 'Fernseher' angepriesen, sondern primär als Spielkonsole. Installiert ist Android 5.1, womit das Gerät derzeit sehr à jour ist.

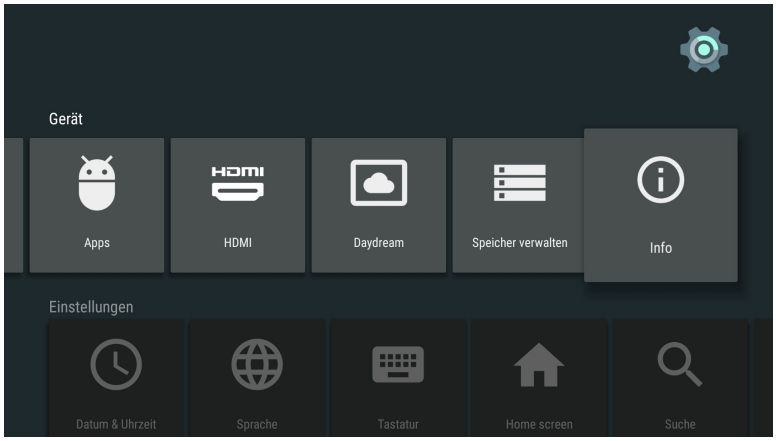
Die Begeisterung für Android basierte Geräte seitens des Vortragneden hält sich in Grenzen. Dies ändert sich auch bei der NVidia TV Box nicht. Um die Box in Betrieb nehmen zu können, ist eine Anmeldung bei Google notwendig. Zwar kann der Account selber später entfernt werden, doch folgen während dem Betrieb dauernd Meldungen, die Box werde mit dem Google-Konto verbunden und es müsste dies und das bestätigt werden. Zu den Angeboten selber gilt es zu sagen, dass diese fast ausnahmslos kostenpflichtig sind. Deutschsprachige Angebote sind spärlich zu finden (die Angebote von Youtube lassen sich genauso gut übers Web finden). Noch weniger Inhalte verfügbar sind, wenn diese familientauglich sein sollen. Am ehesten können (kostenpflichtige) Filme und einige (meist kostenpflichtige) Spiele bezogen werden. Naturgemäss würden dies die Kinder des Vortragenden vielleicht etwas anders sehen, vorausgesetzt Mama oder Papa würde die Kreditkarte immer schön griffbereit halten... Zeit also, Linux auf die Box zu kriegen.

7.2 Developer-Modus aktivieren

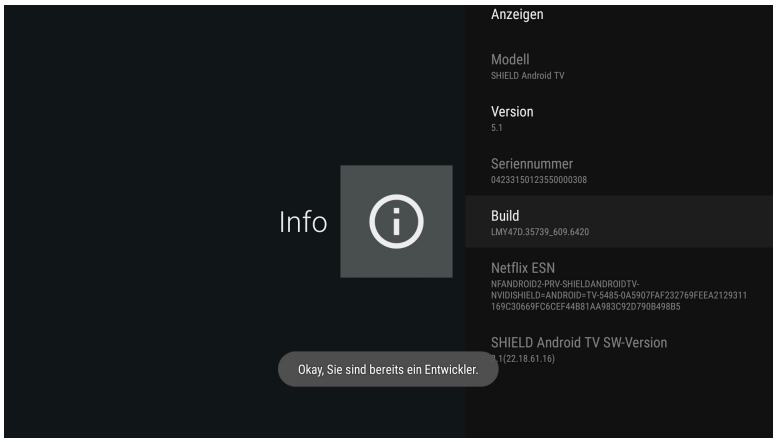
Bevor daran gedacht werden kann, ein Linux zu installieren, muss die Android-Box in den sogenannten Developer-Modus versetzt werden.



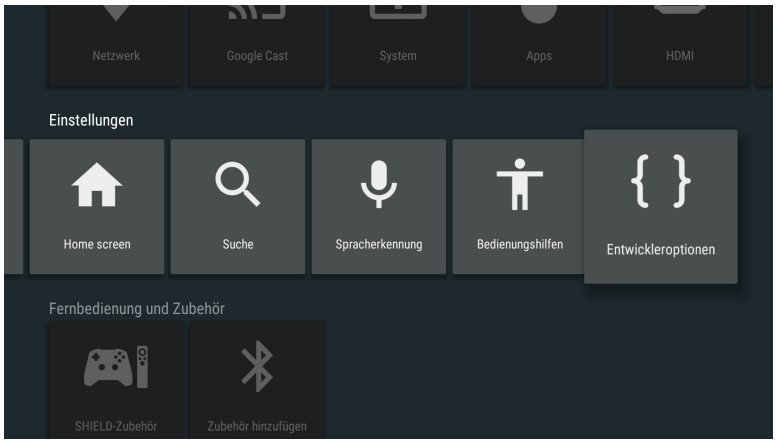
Dazu ist bei den Einstellungen unter Gerät die Kachel 'Info' aufzurufen.



Dort muss der Punkt 'Build' gesucht werden und nun ist die Enter-Taste (auf dem Gamepad die 'A'-Taste) einige Male zu drücken. Dann erscheint die Meldung 'Okay, Sie sind bereits ein Entwickler'.



Mit dem Aktivieren des Developer-Modus erscheint bei den Einstellungen die Kachel 'Entwickleroptionen'.



Innerhalb dieser Option ist 'Debugging' und dann 'USB-Debugging' zu aktivieren.

7.3 Ubuntu 14.04-Image

Vorliegend wird ein Ubuntu-Image verwendet. Dieses kann über den folgenden Link goo.gl/sl6rMu bezogen werden.

Dort können die folgenden beiden Dateien bezogen werden:

nvidia-shield-tv-utopic.arm64.img.2G.gz
nvidia_boot.img

Die erste Datei müssen wir entpacken und auf eine SD-Karte schreiben:

```
zcat nvidia-shield-tv-utopic.arm64.img.2G.gz >nvidia.img
dd if=nvidia.img of=/dev/sdh bs=4194304 oflag=sync
```

Vorliegend wäre **/dev/sdh** die Micro-SD-Karte. Den letzen Buchstaben durch den ermittelten Wert aus **dmesg** tauschen, sobald die Micro-SD-Karte eingelegt ist. Die Micro-SD-Karte danach auf der NVidia TV Box einlegen.

7.4 adb/fastboot mit Linux

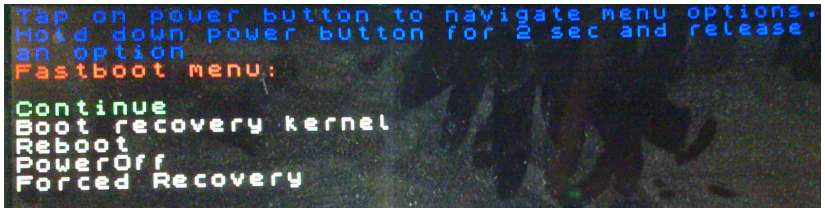
Mit dem aktivierten USB-Debugging kann die Box nun über einen anderen Rechner überwacht bzw. auch mit neuen Versionen bestückt werden. Nachfolgend wird der Weg mit Debian Wheezy aufgezeigt, es könnte aber auch ein Ubuntu oder anderes Linux sein. Ausführliche Aneleitungen gibt es ebenfalls für Windows und Mac. Unter Debian Wheezy benötigen wir die folgenden beiden Pakete:

```
apt-get install android-tools-adb
apt-get install android-tools-fastboot
```

Sobald diese Pakete installiert sind, und auf der NVidia TV Box USB-Debugging aktiviert ist (grundsätzlich sollte dies mit jedem Android-Device gehen), können wir die NVidia TV Box über den entfernt zugreifenden Rechner bzw. das Micro-USB-Kabel neu starten:

```
adb reboot-bootloader
```

Wir erhalten dabei den folgenden Bildschirm:

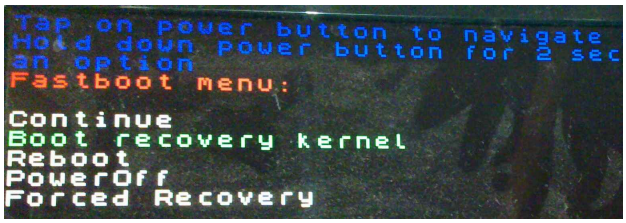


Nun benötigen wir die zuvor bezogende Datei **nvidia_boot.img**. Diese Datei können wir nun auf die NVidia TV Box wie folgt übertragen:

```
fastboot flash recovery nvidia_boot.img
```

Dabei erscheint kurz die Meldung auf dem Bildschirm der NVidia TV Box:

Der Ubuntu-Kernel ist nun im zweiten Boot-Sektor der NVidia TV Box gespeichert, sodass wir diesen aktivieren können, dazu müssen wir auf der NVidia TV Box zum Menüpunkt 'Boot recovery kernel' gelangen. Dies erreichen wir, indem wir den Start-Button kurz tippen:



Hinweis: Wer zulange tippt, startet den aktuellen Kernel, oder landet nicht in der gewünschten Option. Der Vortragende selber übte dabei ein paar Dutzend Mal. Bevor er jedoch übte, suchte er 'verdammt' lange nach dem Start-Button:

Der Power-Knopf findet sich unten rechts im Dreieck (siehe Finger auf Abbildung). Wenn der Eintrag 'Boot recovery kernel' grün eingefärbt ist, passt es. Nun den Start-Button einige Sekunden gedrückt halten, bis der Kernel gestartet wird.

Nun sollte an sich Ubuntu starten, wobei anzumerken gilt, dass einige Dinge angepasst werden müssen, ehe richtig gearbeitet werden kann. Beispielsweise liegt die Netzwerkkarte unter **`eth1`** und nicht unter **`eth0`** wie voreingestellt.

Hinweis: Mit dem obenstehenden Verfahren ist der Kernel immer beim Hochfahren der Maschine auszuwählen. Dies ist nur bedingt 'praktisch'. Wer dies nicht möchte, kann den Linux-Kernel auch in die erste Partition schreiben, wobei hier anzuführen gilt, dass sich dann die NVidia TV Box nicht mehr starten lässt.

7.5 Android (fast) 'verbannen'

Wenn eine automatisch startende Maschine gewünscht wird, kann dies wie folgt erreicht werden:

```
fastboot flash boot nvidia_boot.img
```

Hinweis: Obenstehenden Befehl nur ausführen, falls zuvor der ursprüngliche Kernel für die NVidia TV Box über developer.nvidia.com/shield-open-source bezogen wurde, ohne die initalie Boot-Datei kann Android danach nicht mehr gestartet werden.

7.6 Android (hoffentlich) wieder aktivieren

Mit dem folgenden Befehl sollte sich Android auf der NVidia TV Box wieder aktivieren lassen:

```
fastboot flash boot boot.img
```

Hinweis: Dazu muss die NVidia TV Box im Fastboot-Modus gestartet werden, d.h. der Start-Knopf ist während des Startens einige Sekunden zu halten bzw. im richtigen Moment auch wieder loszulassen, ehe der Bildschirm im Startmodus 'hängen' bleibt. Der Vortragende selber übte dies doch etliche Male, ehe es gelang.

Irgendwann war die Geduld des Tippens am Ende, und so gelangte der Vortragende zur folgenden Lösung. Innerhalb der gestarteten Ubuntu-Version wurde nach Boot-Partitionen gesucht, welche von der Grösse her passen könnten (die Boot-Dateien sind um die 25 MByte gross). Es stellte sich bald heraus, dass 'nur' die Partitionen **/dev/mmcblk0p16** und **/dev/mmcblk0p18** in Frage kommen dürften. Um es vorweg zu nehmen, leider war/ist es die Nummer 18 und nicht die Nummer 16. Folglich muss unter Ubuntu die Partition 18 auf dem internen Datenträger überschrieben werden:

```
root@ds-arm64:/# dd if=boot.img of=/dev/mmcblk0p18 oflag=direct
40000+0 records in
40000+0 records out
20480000 bytes (20 MB) copied, 29.4665 s, 695 kB/s
```

Danach kann die Maschine mit **shutdown now -r** neu gestartet werden.

Wichtig: Es muss beachtet werden, dass die Partitionen bei anderen Maschinen bzw. dem erweiterten Modell an anderer Position liegen könn(t)en und obendrein besteht die Gefahr, dass danach die ganze Maschine im 'Eimer' ist. Demgegenüber steht, dass sich damit die NVidia TV Box jeweils von und nach Ubuntu migrieren lässt, ohne den lästigen Startknopf auf der Box genau drei Sekunden drücken zu müssen.

7.6.1 Kleiner Exkurs Boot-Dateien

Nun interessierte den Vortragenden, was genau die Boot-Dateien sind. Dazu wurde zunächst das Programm **file** verwendet:

```
root@ds-arm64:/in# file nvidia_boot.img
nvidia_boot.img: Android bootimg, kernel (0x10008000),
  ramdisk (0x11000000), page size: 2048
```

Wohlan, da werden Kernel und initram-Datei in eine Datei gepackt. Um die boot-Dateien betrachten zu können, können entsprechende Tools (lauffähig unter Intel/AMD) bezogen werden:

android-serialport-api.googlecode.com/files/android_bootimg_tools.tar.gz

Ausgepackt entstehen daraus die Dateien:

unpackbootimg
mkbootimg

Nun können die boot-Dateien entpackt werden:

./unpackbootimg -i <filename.img> -o <output_path>

Konkret bei der **nvidia_boot.img**-Datei:

./unpackbootimg -i nvidia_boot.img -o linux

Dabei entstehen die folgenden Dateien:

```
     4 nvidia_boot.img-base
     4 nvidia_boot.img-cmdline
     4 nvidia_boot.img-pagesize
  7620 nvidia_boot.img-ramdisk.gz
 16572 nvidia_boot.img-zImage
```

Die Datei **nvidia_boot.img-zImage** ist der Kernel, **nvidia_boot.img-ramdisk.gz** die initiale (komprimierte) Ramdisk. Letzere kann abermals entpackt werden:

gunzip -c boot.img-ramdisk.gz | cpio -i

Nun kann in diesem Mini-Verzeichnisbaum nachgesehen werden, wie es geht, dass der Kernel weiss, dass von der SD-Karte zu booten ist. Dies geht vorliegend einfach, weil die Sourcen in **main.c** enthalten sind (ansonsten es nicht so einfach wäre). Betrachten wir diese mit **cat init.c**, so erhalten wir das folgende Resultat:

```
#include <sys/mount.h>
#include <unistd.h>
#include <linux/reboot.h>
#include <fcntl.h>
extern char **environ;

int main(int argc, char **unused) {
  mount("/dev", "/dev", "devtmpfs", 0, NULL);
  mount("/dev/mmcblk1p1", "/dest", "ext4", 0, NULL);
  mount("/dev", "/dest/dev", NULL, MS_BIND, NULL);
```

```
  chroot("/dest");
  chdir("/");
  char * const argv[] = { "/sbin/init", NULL };
  execve(argv[0], argv, environ)
}
```

Wir sehen, dass dort explizit **/dev/mmcblk1p1** gemountet wird bzw. dass danach die Datei **/sbin/init** gestartet wird. Der Vortragende gibt gerne zu, dass damit der Rahmen des Vortrages gesprengt ist, aber darum erscheint das Kapitel ja auch als Exkurs.

7.6.2 Jetson TX1 Board

Einige Tage vor dem Vortrag wurde das Jetson TX1 Board angekündigt. Dieses enthält plus/minus die gleichen Eckdaten wie die NVidia TV Station, kostet aber mit USD 599.– fast dreimal so viel wie die NVidia TV Station mit Gehäuse und Gamepad. NVidia nennt für Bestellungen ab 1000 Stück noch immer satte USD 299.– als Preis. Die Zukunft wird zeigen, ob sich damit der Tegra X1 Prozessor durchsetzen wird können.

7.7 Einige Eindrücke mit ARM64

Einmal eingerichtet, läuft die Maschine recht schön. So arbeitet z.B. Firefox äusserst flink. Dennoch gibt es derzeit gewisse Einschränkungen:

Erstens passierte es zuweilen, dass nach dem Hochfahren Maus und Tastatur nicht arbeiteten. In einem solchen Falle musste die Box komplett ausgeschaltet werden (vom Strom nehmen half sehr, errinnert irgendwie an die gleichen Probleme unter AMD/Intel). Zweitens konnte LibreOffice auf Anhieb nicht installiert werden (alle anderen gängigen Applikationen wie Gimp oder Scribus liefen). Und drittens, und dies ist weit schmerzhafter, können von den acht Kernen bei der NVidia TV Box nur vier Kerne angesprochen werden. Es handelt sich dabei um einen sogenannten Standard big.LITTLE-Prozessor. Dieser enthält zwar jeweils vier A53- und A57-Kerne, doch können diese (z.B. im Unterschied zum Odroid XU4) nicht gleichzeitig angesprochen werden, weil die Erweiterung HMP (Heterogeneous Multi Processing) fehlt. D.h. aus dem Acht-Kerne-Monster wird ein zwar sehr flügger Vier-Kerner, aber eben nur ein vier Kerner:

```
Processor       : Cortex A57 Processor rev 1 (aarch64)
processor       : 0
processor       : 1
processor       : 2
processor       : 3
Features        : fp asimd aes pmull sha1 sha2 crc32
```

```
CPU implementer : 0x41
CPU architecture: AArch64
CPU variant     : 0x1
CPU part        : 0xd07
CPU revision    : 1
Hardware        : foster_e
Revision        : 0000
Serial          : 09e203a2e2000000
```

Der Durchsatz auf der SD-Karte bewegt sich im üblichen Rahmen, im Vergleich dazu liefert der interne eMMC-Speicher bessere Werte. Kurioserweise meldet **cfdisk /dev/mmcblk0** 29 Partitionen, die letzte enthält die Datenpartition:

```
root@ds-arm64:/# mkdir /android
root@ds-arm64:/# mount /dev/mmcblk0p29 /android/
```

Der Durchsatz ist hier (zumindest beim Lesen) nicht zu beklagen:

```
root@ds-arm64:/# hdparm -tT /dev/mmcblk0
/dev/mmcblk0:
 Timing cached reads:   4606 MB in  2.00 seconds = 2304.35 MB/sec
 Timing buffered disk reads: 602 MB in  3.00 seconds = 200.61 MB/sec
```

Beim Schreiben allerdings konnten nicht wahnsinnige Resultate verbucht werden:

```
dd if=/dev/zero of=eins2.img bs=16M count=10 oflag=direct
10+0 records in
10+0 records out
167772160 bytes (168 MB) copied, 8.08665 s, 20.7 MB/s
```

Diese Werte beim Schreiben werden plus/minus auch von Micro-SD-Karten erreicht (eine SanDisk Extreme Pro erreichte mit ca. 60 MByte sogar deutlich mehr); allerdings ist bei SD-Karten das Lesen deutlich langsamer als dies bei eMMC-Datentägern der Fall ist.

Zum Abschluss noch dies. Die publizierten 10 Watt Leistungsaufnahme lassen sich nicht bestätigen. Im Ruhezustand unter Android konnten 13 Watt gemessen werden, plus/minus ditto unter Ubuntu. Unter 'Vollast' (es konnten wie ausgeführt nur vier CPUs) angesprochen werden, stiegt der Verbrauch auf ca. 26 Watt. Dieser Wert ist nun nicht mehr wahnsinnig weit enternt von der Intel NUC-Maschine, doch auch hier gilt, von der Geschwindigkeit her betrachtet ist die NVidia TV Box der Intel NUC (selbst mit vier Kernen) aber bereits deutlich überlegen. Und auch das darf gesagt werden, die Android TV Box kommt ohne Lüfter daher.

8 ARM-Desktop-Cluster

Nachdem die 256 GPU-Kerne der NVidia TV-Box leider nicht für Tesseract genutzt werden konnten (OpenCL ist derzeit nicht verfügbar), stellte sich für den Vortragenden der Frage, ob nicht doch mehr aus der Texterkennung herausgeholt werden kann als die gut 30 Seiten pro Minute. Nun können parallele Jobs, wie dies für das Abarbeiten der einzelnen Seiten bei der Texterkennung der Fall ist, ja an sich auch auf mehrere Rechner verteilt werden.

Nebenbmerkung: Natürlich gehört ein Cluster nicht primär zum Anwendungsszenario einer Desktop-Umgebung. Texterkennung (OCR) jedoch schon. In diesem Sinne passt der Desktop-Cluster ganz bestimmt zum Thema, zumal der ARM-Desktop-Cluster, wie noch zu zeigen sein wird, auch bequem auf einem Desktop Platz findet. Weiter gibt der Vortragende gerne zu, dass der ARM-Cluster zu Beginn (Stichwort Rucksack-Test) nicht vorgestellt wurde, aber ein wenig Überraschung darf ja schon sein.

8.1 Cluster mit Rasperry PI 2

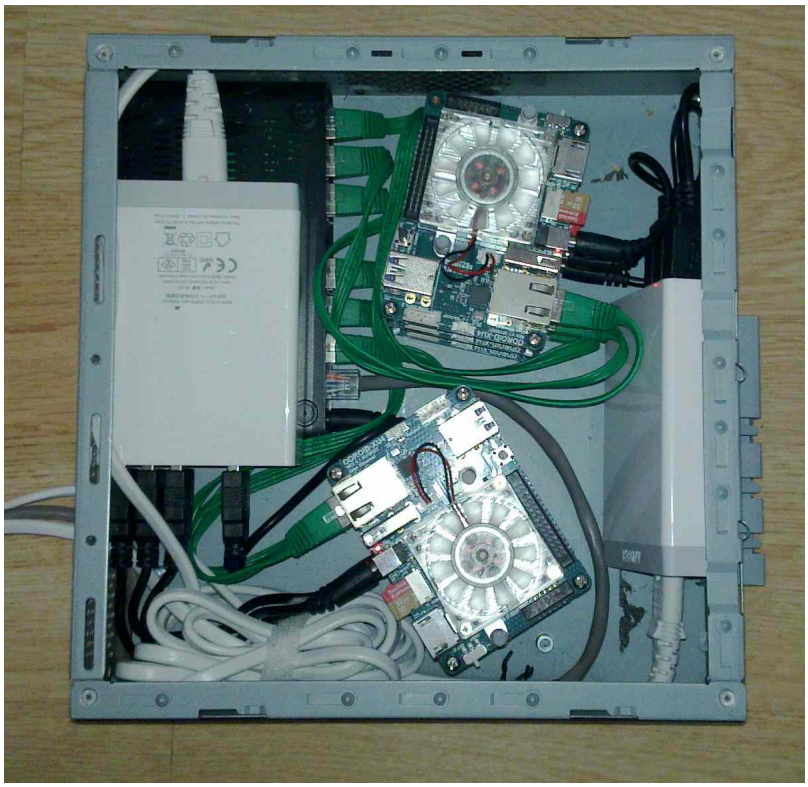
Die ersten Versuche fanden bereits im Frühjahr mit dem Rasperry PI 2 statt. Ganze 12 Rasperry PI 2 wurden in ein MicroATX-Desktop-Gehäuse verfrachtet. Anstelle der PI 2 Netzteile wurden 2 Anker 6-Port-USB-Ladegeräte mit je 60 Watt verwendet, für das interne Netzwerk kamen zwei 8-Port USB-Netzwerkswitches Edimax ES-5800G V3 zum Einsatz. Die Kosten für diesen 12-er-Cluster konnten wie folgt ermittelt werden:

- 12 Rasperry PI 2: 420 Euro
- 12 Gehäuse zu PI 2: 120 Euro
- 2 Anker 6-Port USB-Ladegeräte 60W: 120 Euro
- 2 Edimax ES-5800G V3: 80 Euro
- 12 Netzwerkkabel intern: 60 Euro
- 12 Micro-USB-Kabel (90 Grad gedreht): 100 Euro
- 1 MicroATX Gehäuse Chembro: 60 Euro
- 12 Mirco SD Karten (8 GByte): 60 Euro
- Externes Netzwerkkabel: 5 Euro

Damit entsteht ein 12-Cluster mit 48 Cores für 1025 Euro, pro Core ergeben sich gute 21 EUR0 an Kosten. Problematisch erwies sich dabei die Hitzeentwicklung des Raspberry PI 2 im geschlossenen Gehäuse. Dies führte dazu, dass die CPU gedrosselt wurde, sodass die Geschwindigkeit der Texterkennung merklich nachliess, mehr als 30 bis 40 Seiten pro Minute konnten nicht erreicht werden. Zum Vergleich, eine schnelle 6-Core erreicht pro Minute ca. 60 Seiten; die Kosten für ein Gesamtgerät dürften bei der 6-Core-Maschine um die 500 Euro liegen. Aus diesem Grunde wurde das Projekt Rasperry PI 2 Cluster nicht mehr weiterverfolgt.

8.2 48-Core-Cluster im mITX-Format mit ODROID

Wären die 256 GPU-Kerne der Android TV Box mit OpenCL verfügbar gewesen, so wäre der ODROID XU4 Cluster wohl nicht entstanden. Und wäre mit der Ankündigung des Jetson TX1-Boards nicht klar gewesen, dass NVidia nicht gewillt ist, OpenCL zu unterstützen, so wäre es ebenfalls nicht zum Desktop-Cluster gekommen. Zusätzlich bestärkt wurde der Entscheid dadruch, dass das Jetson-TX1-Package 599 USD kostet und dass bei Abnahme von 1000 Stück TX1-Modulen (ohne Board, Gehäuse etc) 300 USD pro Einheit zu zahlen sind. Die Jetson-TX1-Plattform erscheint dem Vortragenden (so stark die ARM64-Bit CPU ist) derzeit für preisgünstiges Cluster-Computing nicht sinnvoll. Aus diesem Grunde sollte ein ODROID XU4-Cluster her. Immerhin erreicht ein ODROID XU4 ca. 30 Seiten Texterkennung pro Minute. Und weil es diesmal ein 'echter' Desktop werden sollte, fiel die Entscheidung, den Cluster in ein mITX-Gehäuse von Chembro zu verbauen:



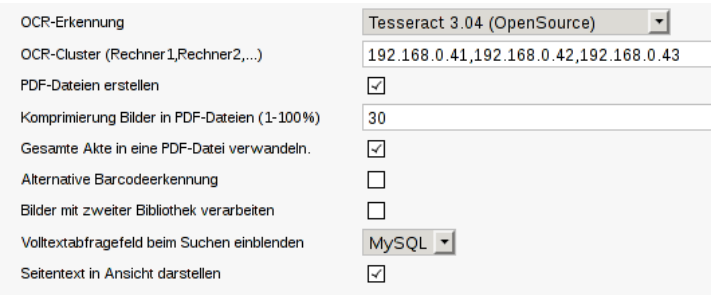
Die ODROIDS XU4 werden an die Halterungen im Gehäuseboden verankert (je zweimal drei Stück übereinander), bei den Netzwerken fiel die Wahl auf die Anker-USB-Netzteile, welche mit 60 Watt immerhin drei ODROID XU4 'bedienen' können. Der Netzwerk-Switch wird ebenfalls über das Anker-Netzteil versorgt, sodass letztlich nur drei Kabel nach aussen führen (zweimal Strom sowie ein Netzwerkkabel). Folgende Kosten sind zu verzeichnen:

- 6 x ODROID XU4 (inkl. Stromadapter): 450 EURO
- 2 Anker 6-Port USB-Ladegeräte 60W: 110 Euro
- 2 Edimax ES-5800G V3: 75 Euro
- Gehäuse Chembro ITX: 25 EURO
- 6 x Micro SD-Karte: 25 EURO
- 6 x Netzwerkkabel: 15 EURO
- 6 Power-to-USB-Kabel: 10 EURO
- Übriges (Befestigung Platinen): 5 EURO

Damit ergeben sich Hardwarekosten von 700 EURO, dies ergibt bei 48 Cores Kosten von ca. 14.5 EURO pro Core. Nun fragt sich, wofür ein Cluster verwendet werden kann. Grundsätzlich eignet sich ein Cluster für alle Projekte, bei denen parallele Jobs abzuarbeiten sind. Im Falle der ArchivistaBox ist dies bei der Texterkennung der Fall, womit wir beim OCR-Cluster wären.

8.3 ArchivistaBox OCR-Cluster

Dank den neuen ARM-Boxen konnte die OCR-Erkennung (just einige Tage vor dem Vortrag) so erweitert werden, dass die Texterkennung auch über mehrere Rechner (Knoten) parallel arbeiten kann. Dieser OCR-Cluster soll an dieser Stelle demonstriert werden. Vorliegend hätten wir dsechst Knoten unter den Adressen 192.168.0.37, 38, 39, 41, 42 und 43. Die Datenbank selber liegt auf 192.168.0.41. Die Verwaltung des OCR-Clusters erfolgt webbasiert. Das Einrichten ist denkbar einfach. In WebAdmin anmelden und dort bei 'Archiv verwalten' die entsprechenden Rechner bei 'OCR-Cluster' eintragen:



Danach ist der Cluster aktiviert. Alle Jobs betr. der Textverarbeitung werden ab sofort über die Cluster-Konoten verteilt. Nun zum Testergebnis: Für das englische Handbuch konnte für die Texterkennung über die 204 Seiten eine Zeit von ca. 1 Minuten 10 Sekunden gemessen werden. Dies ergibt pro Minute plus minus ca. 180 erkannte Seiten bzw. pro Tag immerhin ca. 250'000 Seiten (bei 6 Knoten). Nicht schlecht, wenn berücksichtigt wird, dass dafür sechs ARM-'Böxli' zuständig sind. Der vorliegende OCR-Cluster erreicht mehr Leistung, als ausgewachsene Dokumenten-Scanner an Seiten pro Minute anzuliefern vermögen. In diesem Sinne bietet der OCR-Cluster Texterkennung in Echtzeit.

Der vorliegende OCR-Cluster ist nicht an die ARM-Hardware 'gebunden', er arbeitet genauso gut mit Intel/AMD-Rechnern (z.B. in Verbindung von Master/Slave). Voraussetzung ist einzig, dass mindestens Quad-Core-CPUs vorhanden sind und dass Systeme ab Ausbaustufe ArchivistaBox Dolder (ARM-Variante) oder ArchivistaBox Rothorn/Eiger (Intel/AMD) zur Verfügung stehen. Damit ist auch gesagt, dass bei der ARM-Variante der ArchivistaBox die Cluster-Knoten deutlich günstiger sind; tiefere Einkaufskonditionen geben wir gerne weiter.

Nebenbemerkung: Eine kommerzielle OCR-Erkennung (Kommando-Zeilen-Programm), welche pro Jahr (nicht pro Tag!) 500'000 Seiten verarbeiten mag, kostet (nur schon die Software) fast doppelt so viel, siehe dazu www.ocr4linux.com/de:pricing (Seitenlimiten inklusive). Die kommerzielln Kommando-Zeilen-Lösung ist im Umfang eingeschränkt, so können z.B. trainierte Zeichensätze nicht verwendet werden, beim ArchivistaBox OCR-Cluster dagegen schon. Überdies arbeitet der ArchivistaBox OCR-Cluster webbasiert und lässt sich, im Unterschied zum Mitbewerber, beliebig skalieren. Beispiele: Ein 24-er Cluster (192 Cores) erreicht ca. 1 Million Seiten Tagesleistung. Bei 240 Knoten bzw. 1920 Cores könnten ca. 10 Millionen Seiten pro Tag verarbeitet werden; dies wären ca. 50'000 Bücher pro Tag.

Kernaussage: Der vorliegende OCR-Cluster ist bei 6 Knoten mit ARM der schnellsten Intel i7 bzw. XEON deutlich überlegen. Damit erriecht der mITX-OCR-Cluster sowohl von der Performanz als auch von der Leistungsaufnahme Werte, welche mit einem Intel/AMD-Rechner nicht erreicht werden können, für den 48-Core-Cluster werden unter Last ca. 75 Watt benötigt, im Leerlauf sind es ca. 35 Watt.

Werbung: Der hier beschriebenen Cluster sind bei der Firma des Vortragenden zum Preis der ArchivistaBox Dolder erhältlich, siehe dazu die Preise unter shop.archivista.ch (dort Archivista-DMS sowie 6xModell Dolder für 6-er-Cluster wählen). Selbstverständlich können mit dem Cluster auch durchsuchbare PDF-Dateien erstellt werden.

9 Aussichten für ARM

9.1 Einsatz als Desktop

Sowohl der Raspberry PI 2 als auch der ODROID XU4, noch pointierter die NVidia TV Box, sind Vertreter einer neuen ARM-Generation, welche ein hohes Mass an Leistung in einen Kleinstrechner packen. Der Raspberry PI 2 eignet sich bedingt als Desktop-Ersatz, beim ODROID XU4 ist dies ohne Einschränkungen der Fall. Die NVidia TV Box ist zunächst einmal gerade nicht als Linux-Rechner erhältlich. Wer eine ARM64-Bit-Hardware haben möchte und sich mit **`fastboot`** anfreunden kann, erhält mit der NVidia TV Box aber zweifelsohne ein sehr leistungsfähiges Gerät (auch ohne GPU-Kerne).

Gewisse Abstriche gemacht werden müssen am ehesten, wenn es darum geht, virtualisierte Instanzen zu betreiben. Dabei spielt sicher eine Rolle, dass 1 bis 3 GByte an Speicher für virtualisierte Instanzen ohnehin knapp bemessen sind. Doch seien wir ehrlich, wer als Anwender/in arbeitet schon mit virtuellen Instanzen. Für einen Desktop-Rechner ist dies bei einem Preis bei 100 bis 200 EURO kaum eine Einschränkung. Letztlich können für den Preis einer ausgewachsenen Desktop-Maschine 2 oder 3 ODROIDs aufgebaut werden, dürfte vom Budget und vom Gewicht her noch immer passen.

9.2 ARM und ArchvistaBox

Die NVidia TV Box wird für die ArchivistaBox produktiv wohl kaum zum Einsatz gelangen. Dies alleine schon deshalb, weil die Modellpfelge im Consumer-Markt sich fast jeden Tag wieder ändern kann. Trotzdem bietet die NVidia TV Box eine ideale (kostenpassende) Grundlage, um sich frühzeitig mit ARM64 vertraut machen zu können.

Für die ArchivistaBox bedeuten sowohl der Raspberry PI 2 als auch der ODROID XU4 Meilensteine. Beim Raspberry PI 2 ist dies der Fall, weil für kleinere Archive (Tagesvolumen unter 500 Seiten) eine extrem preiswerte Box zur Verfügung steht, beim ODROID XU4 können Archive bis zu 4 TByte auf einer ca. 350 g leichten Box mit acht Kernen realisiert werden. Offen gestanden, dies hätte der Vortragende vor einem Jahr nicht zu träumen gewagt. Damit die Kerne allerdings ausgereizt werden konnten, musste der Code nicht in weiten Teilen, aber doch im Grundsatz geändert werden (Stichwort Parallelität). Diese Optimierungen sind mittlerweile bei allen Modellen aktiviert, bei 2 CPUs ist der Vorteil nicht erheblich, doch bereits bei vier, noch viel mehr bei acht Kernen, ergeben sich gewaltige Geschwindigkeitsvorteile, für sämtliche ArchivistaBox-Systeme, mit oder ohne ARM-Prozessor.

Dies gilt umso mehr für den vorgestellten OCR-Cluster auf ARM-Basis, mit dem Werte erreicht werden können, die selbst die besten Intel/AMD-Boliden alt aussehen lassen. Dies bei einem weit besseren Preis-/Leistungsverhältnis und überdies bei deutlich geringeren Leistungsaufnahme.

9.3 Abschliessendes Fazit

Das Arbeiten mit ARM-Rechnern bereitet Ende 2015 ein hohes Mass an Freude. Die anfallenden Arbeiten können locker und speditiv erledigt werden, zu einem Preis notabene, zu dem keine Intel/AMD-Rechner zur Verfügung stehen. Natürlich kommt eine ARM-CPU derzeit nicht an die schnellsten Intel/AMD-Rechner heran, doch zeigt der Einsatz des hier vorgestellten ARM-Desktop-Clusters, dass mit paralleler Verarbeitung selbst Intel/AMD-Boliden bei weitem in den Schatten gestellt werden können.

Etwas getrübt wird die Freude einzig dadurch, dass bei ARM ähnliche 'Stolpersteine' wie bei UEFI und SecureBoot unter Intel/AMD bestehen können. Allerdings muss hier angefügt werden, dass derartige Probleme erst dann zu Tage treten, wenn Kernel gebaut werden. Betrachtet man die Anzahl der neu publizierten Images z.B. beim ODROID XU4, kann nicht per se behauptet werden, die Kernel-Hacker würden sich von diesen Hürden abhalten lassen.

All denjenigen, die keinen Anlass sehen, Intel/AMD den Rücken zuzuwenden, sei gesagt, dass moderate Preise für CPUs heute nicht zuletzt dadurch wieder en vogue sind, weil es mit ARM eine Alternative gibt, die es früher nicht gab. In diesem Sinne profitieren letztlich alle.

Zu hoffen bleibt, dass in einem Jahr sich ARM mit 64 Bit und mehr RAM auf vielen weiteren Boxen durchsetzen wird bzw. dass diese Systeme auch für Linux (und nicht primär nur für Android) zur Verfügung stehen werden. Doch schon heute gilt, Intel/AMD müssen sich 'warm' aniehen, denn warum sollte jemand zu einem Intel/AMD-Rechner mit leistungsfähiger GPU-Grafikkarte greifen wollen, wenn das gleiche in kleinen ARM-Boxen verfügbar ist? Herzlichen Dank für die Aufmerksamkeit.

Kontakt: Urs Pfister, Archivista GmbH, Stegstr. 14, CH-8132 Egg, www.archivista.ch