Erfahrungen beim Aufbau von großen Clustern aus Einplatinencomputern für Forschung und Lehre

Christian Baun · Henry-Norbert Cocos
Rosa-Maria Spanou

Einleitung

Wenn Rechenleistung, Ausfallsicherheit oder Datendurchsatz eines einzelnen Computers nicht ausreichen, sind Verbünde, sogenannte Cluster, von miteinander vernetzten eigenständigen Computern die naheliegende Lösung. Die beteiligten Computer heißen im Cluster-Kontext meist Knoten (englisch: *Nodes*) und ein Cluster verhält sich im Idealfall wie ein einheitlicher Uniprozessor.

Bestanden frühe Cluster-Systeme in den 1980er-Jahren meist aus wenigen Minicomputern (z. B. DEC VAX), kommen seit den 1990er-Jahren im professionellen Umfeld meist Server als Knoten zum Einsatz. Im Bereich Forschung und Lehre dienen häufig PCs aus Standardkomponenten (*Commodity-Hardware*) als Knoten. Diese erreichen mangels redundanter Komponenten (z. B. Netzteile und Lüfter) zwar eine geringere Verfügbarkeit und bieten auch keine leistungsfähigen Verwaltungs- und Wartungswerkzeuge wie z. B. das *Integrated Management Module* von IBM oder *Integrated Lights-Out* (iLO) von HPE, aber dafür sind die Anschaffungskosten im Vergleich zu Servern geringer.

Der Kostenaufwand, um einen Cluster mit mehreren Knoten aufzubauen, hängt in erster Linie von der Anzahl und Ausstattung der verwendeten Knoten ab. Sind die Knoten Linux-Server aus Standardhardware, verursacht jeder Knoten Anschaffungskosten von mehreren hundert bis über 1.000 Euro. Der Aufbau eines Clusters mit acht Knoten führt daher häufig zu Anschaffungskosten von über 10.000 Euro. Zudem ist der Stromverbrauch mit dem von Arbeitsplatzrechnern vergleichbar. Die Betriebskosten für einen Cluster-Knoten können in

fünf Jahren fünf- bis achtmal so hoch sein wie die Anschaffungskosten für Hard- und Software. Diese Anschaffungs- und Betriebskosten verhindern häufig die Beschaffung bzw. den Aufbau von Clustern für Forschung, Entwicklung und Lehre.

Speziell für die Lehre im Bereich parallele und verteilte Systeme, aber auch für Forschungsvorhaben, bei denen nicht eine hohe Rechenleistung und Speicherkapazität, sondern viele physische Prozessorkerne (Rechenkerne) im Vordergrund stehen, können Cluster auch aus Einplatinencomputern realisiert werden. Diese haben den Vorteil, dass ihre Anschaffungskosten und der Energieverbrauch konkurrenzlos günstig sind. Zudem benötigen solche Computer nur wenig Platz und können in den allermeisten Fällen sogar passiv gekühlt werden.

Der vorliegende Artikel benennt Anwendungen, die sich zum Betrieb auf Clustern aus Einplatinencomputern eignen und beschreibt Erfahrungen, die beim Aufbau solcher Cluster gewonnen wurden. Zudem erfolgt eine Kalkulation der entstehenden Kosten.

Sinnvolle Einsatzmöglichkeiten

Beim Aufbau eines Parallelrechners sind Einplatinencomputer wegen ihrer vergleichsweise schwachen Rechenleistung, des geringen Hauptspeichers und der wenigen Möglichkeiten,

DOI 10.1007/s00287-017-1083-9
© Die Autoren 2018. Dieser Artikel wurde mit Open Access auf Springerlink.com veröffentlicht.

Christian Baun · Henry-Norbert Cocos · Rosa-Maria Spanou
Frankfurt University of Applied Sciences,
Nibelungenplatz 1, 60318 Frankfurt am Main
E-Mail: christianbaun@fb2.fra-uas.de

Tabelle 1

Eckdaten der untersuchten Einplatinencomputer					
	Raspberry Pi 1	BananaPi 1 (ab Mitte 2014)	ODROID-U3	Raspberry Pi 2 (ab Mitte 2015)	Raspberry Pi 3 (ab Mitte 2016)
Prozessor	ARM 11	ARM A7	ARM A9	ARM A7	ARM A53
Rechenkerne	1	2	4	4	4
Taktrate	700 MHz	900 MHz	1700 MHz	900 MHz	1200 MHz
Hauptspeicher	512 MB	1024 MB	2048 MB	1024 MB	1024 MB
Ethernet-Schnittstelle	100 Mbit	1000 Mbit	100 Mbit	100 Mbit	100 Mbit
Persistenter Speicher	SD/microSD	SD, SATA	microSD	microSD	microSD
Passiv kühlbar	ja	ja	schwierig	ja	schwierig
Software	++	0	0	++	++
Community	++	0	+	++	++
Stabilität	++	++	+	++	0
Preis	++	0	--	++	+
Leistung	--	0	++	+	+

Massenspeicher anzuschließen, nicht unbedingt die erste Wahl. Tabelle 1 zeigt eine Übersicht über die relevanten Hardwarekomponenten gängiger und im Rahmen dieser Arbeit untersuchten Einplatinencomputer, die über große Benutzergemeinden (*Communities*) verfügen und die aktiv weiterentwickelt werden. Die Bewertungen in Tab. 1 sind subjektiv und ergeben sich durch Beobachtungen aus dem Alltag.

In den Jahren 2013 und 2014 wurden primär Cluster aus Raspberry Pi 1, BananaPi 1 und ODROID-U3 untersucht. Ab 2015 bzw. 2016 wurde im Rahmen dieser Arbeit primär mit Clustern aus Raspberry Pi 2/3 gearbeitet. Gründe für den Fokus auf diese Familie aus Einplatinencomputern sind die Verfügbarkeit gut gepflegter Betriebssystemabbilder für die Linux-Betriebssysteme Raspbian [12] und Ubuntu [15] und der günstige Anschaffungspreis von ca. 40 Euro pro Rechner. Für einen exemplarischen Cluster aus acht Raspberry Pi 2- oder Raspberry Pi 3-Einplatinencomputern (beide Modelle unterscheiden sich nur geringfügig in den Anschaffungskosten) ergeben sich Anschaffungskosten von unter 600 Euro (siehe Tab. 2).

Neben den Anschaffungskosten sind auch die Betriebskosten ein wichtiger Faktor für die Wirtschaftlichkeit eines Computersystems. Ein wie in Tab. 2 beschriebener Cluster aus acht Raspberry Pi 2-Einplatinencomputern verbraucht unter Vollast ca. 35 Watt. Daraus ergeben sich gemäß der

folgenden Formel Stromkosten von unter 100 Euro pro Jahr.

$$0,035 \text{ kW} \times 24 \frac{\text{h}}{\text{Tag}} \times 365,25 \frac{\text{Tag}}{\text{Jahr}} \times 0,3 \frac{\text{€}}{\text{kWh}} \\ \approx 92,05 \frac{\text{€}}{\text{Jahr}}$$

Da bereits ein einziger Server oder ein PC aus Standardkomponenten in der Regel mit einer elektrischen Leistung von mehreren hundert Watt läuft, sind die Energiekosten für einen Cluster aus

Tabelle 2

Anschaffungskosten für einen Cluster aus acht Raspberry Pi 2/3-Einplatinencomputern	
Komponente	Anschaffungspreis
8 Raspberry Pi 2 oder 3 Einplatinencomputer	320 €
8 microSD-Karten	100 €
8 microUSB-Kabel	10 €
9 Ethernet-Kabel	10 €
Stromversorgung	50 €
Netzwerk-Switch	40 €
Summe	560 €

Einplatinencomputern sehr niedrig im Vergleich zu anderen Lösungen.

Alle in Tab. 1 präsentierten Einplatinencomputer sind hinsichtlich ihrer Rechenleistung und Hauptspeicherkapazität leistungsfähig genug, um mit einem Linux-Betriebssystem Serverdienste zu betreiben. Da die Einplatinencomputer alle mit einer 100 Mbit-Ethernet oder Gigabit-Ethernet-Netzwerkschnittstelle ausgestattet sind, ist mit ihnen der Aufbau von Clustern prinzipiell möglich.

Wegen ihrer eingeschränkten Hardwareausstattung und Erweiterungsmöglichkeiten eignen sich Einplatinencomputer nicht für alle Anwendungen aus dem Bereich der parallelen und verteilten Systeme. So hat beispielsweise der in Tab. 1 beschriebene Cluster aus acht Knoten immerhin 32 Rechenkerne, aber nur 8 GB Hauptspeicher, was selbst für einen einzelnen modernen Server oder eine Workstation ein niedriger Wert wäre.

Im Rahmen dieser Forschungsarbeit wurden unterschiedliche potenzielle Einsatzgebiete für Cluster aus Einplatinencomputern hinsichtlich ihrer Nützlichkeit für Forschung und Lehre untersucht. Konkret handelt es sich um diese Themen:

- Paralleles Rechnen,
- Mobile Anwendungen,
- Cloud-Plattformdienste und -Infrastrukturdienste.

Paralleles Rechnen

Wegen ihrer geringen Anschaffungs- und Betriebskosten eignen sich Cluster aus Einplatinencomputern potenziell sehr gut für Anwendungen aus dem Bereich des parallelen Rechnens. Mit relativ geringen finanziellen Mitteln ist es möglich, ein paralleles System mit vielen physischen Rechenkernen zu beschaffen. Ein solches System eignet sich ideal für Anwendungen in der Lehre und, je nach Einsatzzweck, auch für die Forschung.

Die Entwicklung eigener Programme ist beispielsweise mit der Bibliothek *Message Passing Interface* (MPI) [10] möglich. Diese stellt Funktionen zur Entwicklung eigener Cluster-Programme bereit und ist der De-facto-Standard zur Entwicklung paralleler Programme in Clustern. Lösungen mit vergleichbarer Funktionalität wie zum Beispiel OpenSHMEM [9], PVM und Unified Parallel C existieren zwar, haben aber keine vergleichbare Verbreitung.

Implementierungen der MPI-Bibliothek existieren u. a. für die Programmiersprachen C++, C, Python, Fortran und Java. Die enthaltenen Funktionen ermöglichen Interprozesskommunikation, Prozesssynchronisierung, Auslesen von Umgebungsvariablen, Prozessgruppen, Dateiein- und -ausgabe sowie Speicherzugriff.

Der Aufbau von Clustern aus Einplatinencomputern mit MPI ist mit geringem Aufwand möglich, da entsprechende MPI-Pakete bereits bei allen gängigen Linux-Distributionen enthalten sind und der Konfigurationsaufwand von MPI gering ist. Zahlreiche andere Projekte mit Einplatinencomputern haben solche Systeme in den vergangenen Jahren realisiert. Beispiele sind die Installationen an den Universitäten Southampton [8], Boise [16], Bozen [1] und Maine [7] und an der Frankfurt University of Applied Sciences.

Neben kleineren Clustern mit 4, 8 (siehe Tab. 2), 16 oder 32 Knoten, wurde im Rahmen dieser Arbeit auch ein Cluster aus 128 Raspberry Pi 3 (siehe Abb. 1) für unter 10.000 € Anschaffungskosten aufgebaut. Ein solches System hat immerhin 512 physische Rechenkerne und verbraucht im Leerlauf ca. 350 Watt und bei Spitzenlast bis zu ca. 650 Watt.

Eine interessante Frage ist, welche Rechenleistung ein solcher Cluster im Vergleich zu anderen Lösungen erbringen kann. Ein etablierter Weg, die Rechenleistung eines Clusters zu bestimmen, ist der High-Performance Linpack (HPL) Benchmark [11]. Dieser löst ein lineares Gleichungssystem und misst dabei die Gleitkommazahlenoperationen (Additionen oder Multiplikationen) pro Sekunde. Das Ergebnis wird in FLOPS (floating-point operations per second) angegeben. Tabelle 3 zeigt eine Auswahl der Messergebnisse, die im Rahmen dieser Arbeit auf verschiedenen selbst aufgebauten Clustern gemessen wurden.

Zum Vergleich: In der TOP500-Liste (Stand: Juni 2017) der schnellsten Supercomputer hatten alle Systeme in den Top 10 eine Rechenleistung von mehreren PetaFLOPS. Die von uns gemessenen Werte liegen noch unter den Leistungsdaten eines einzelnen modernen Servers oder PCs. Damit ist klar, dass beim aktuellen Leistungsstand der einzelnen Knoten, ein Cluster aus Einplatinencomputern niemals eine sinnvolle Lösung sein kann, wenn das Ziel die Erreichung einer möglichst hohen Rechenleistung ist. Soll allerdings die Skalierbarkeit von Programmen untersucht werden, kann ein solcher

Abb. 1 Der Cluster aus 128 Raspberry Pi 3-Knoten an der Frankfurt University of Applied Sciences

Cluster eine ideale Lösung darstellen, weil für vergleichsweise geringe Kosten eine sehr große Zahl physischer Rechenkern beschafft werden kann.

Zu den elementaren Fähigkeiten, die im Rahmen einer Lehrveranstaltung zu parallelen und verteilten Systemen in der Informatik vermittelt werden, gehört die Erkenntnis, dass Softwareprozesse durch parallele Ausführung auf mehreren unabhängigen Rechnern nicht unendlich beschleunigt werden können.

Die Auseinandersetzung mit den Möglichkeiten und Einschränkungen von Parallelrechnern findet in Wissenschaft und Alltag seit mehreren Jahrzehnten statt und wird u. a. in Form der Gesetze von Gene Amdahl und John Gustafson in den entsprechenden Lehrveranstaltungen im Rahmen einer umfangreichen Ausbildung in Informatik gelehrt.

Das im Kern pessimistisch formulierte Amdahlsche Gesetz aus den 1960er-Jahren beschreibt ein Modell zur Berechnung der potenziellen Beschleunigung („SpeedUp“) von Programmen durch parallele Ausführung. Es sagt aus, dass der Geschwindigkeitszuwachs vor allem durch den sequenziellen Anteil eines Programms beschränkt ist. In erster Linie handelt es sich hierbei um diejenigen Programmabschnitte, die sich mit Prozessinitialisierung, Speicherallokation, Ein-/Ausgabe und dem Verarbeiten von Zwischenergebnissen zum Endergebnis beschäftigen.

Das Amdahlsche Gesetz sagt zudem aus, dass durch den Aufwand für Kommunikation und zur Synchronisierung zwischen den Prozessoren die Laufzeit eines Programms bei paralleler Abarbeitung bis zu einem gewissen Maximum beschleunigt

Gemessene FLOPS bei verschiedenen Clustern aus Einplatinencomputern

Einplatinencomputer	Knoten [Anzahl]	Rechenkern [Anzahl]	FLOPS
RasPi 1 (800 MHz)	8	8	1,3 Gflops
BananaPi 1 (900 MHz)	8	16	4,0 Gflops
RasPi 2 (900 MHz)	8	32	7,1 Gflops
RasPi 3 (1150 MHz)	8	32	12,4 Gflops
RasPi 2 (900 MHz)	16	64	14,0 Gflops
RasPi 2 (900 MHz)	32	128	24,7 Gflops
RasPi 3 (1000 MHz)	128	512	107,1 Gflops

Tabelle 3

werden kann, aber die Hinzunahme weiterer Prozessoren zu einer erneuten Vergrößerung der Laufzeit führt. Einfach gesagt: Zusätzliche Prozessoren beschleunigen die Abarbeitung nicht mehr, sondern verlangsamen sie. Dieser im Modell vorhergesagte Effekt kann auch in der Praxis beobachtet werden.

Eine in [2] exzellent beschriebene Alltagsversion des Amdahlschen Gesetzes ist wie folgt: Man stelle sich vor, ein Maler benötigt zum Tapezieren eines Zimmers 60 Minuten. Eine realistische Annahme ist, dass zwei Maler diese Aufgabe in 30 Minuten bewältigen können. Es ist aber unrealistisch zu glauben, dass 60 Maler das Zimmer in einer Minute tapezieren. Der Grund dafür ist, dass sich die Maler gegenseitig im Weg stünden und sich wegen des Zugriffs auf knappe Ressourcen (Tapeziertisch, Leiter, etc.) gegenseitig blockieren. Vermutlich würde das Tapezieren des Raumes mit 60 Malern länger als 30 Minuten dauern. Ganz anders ist die potenzielle Beschleunigung, wenn man nicht von einem einzelnen Zimmer, sondern von einem Haus mit 60 Zimmern ausgeht.

Überträgt man diese Erkenntnis auf Parallelrechner, bedeutet das, dass bei zunehmender Zahl der Prozessoren auch die Problemgröße wachsen sollte. Das zu bearbeitende Problem muss mit der Anzahl der Prozessoren „skalieren“. Diese Erkenntnis führte in den 1980er-Jahren zu Gustafsons Gesetz. Dieses besagt, dass ein genügend großes Problem effizient parallelisiert werden kann, denn der sequenzielle Anteil wird mit zunehmender Anzahl an Prozessoren unbedeutender.

Die Vermittlung von solchen elementaren Gesetzmäßigkeiten verteilter Systeme anhand textueller Beschreibungen und mathematischer Modelle ist in der Lehre der Standard. Der Grund dafür ist u. a., dass eine sinnvolle Darstellung der potenziellen Beschleunigung bei paralleler Programmausführung nur mit einer sehr großen Anzahl an Prozessoren bzw. Rechenkernen möglich ist. Einzelne Prozessoren haben aber in der Praxis selten mehr als vier, sechs oder acht Rekerne. Ausnahmen sind beispielsweise einzelne Xeon-Prozessoren von Intel mit 22 Rechenkernen (Xeon E5-2699 v4) oder die Ryzen-Prozessoren von AMD (Ryzen Threadripper 1950X) mit bis zu 16 Kernen. Einzelne solche Systeme mögen eine hohe Rechenleistung aufweisen, kosten aber mehrere tausend Euro und eignen sich wenig um die Skalierbarkeit von Pro-

grammen bei paralleler Programmausführung zu untersuchen.

Ein Beispiel für eine Untersuchung der Beschleunigung eines Programms bei paralleler Ausführung zeigt Abb. 2. Diese zeigt die Laufzeit und den SpeedUp (im Vergleich zur Ausführung auf nur einem Rechenkern) eines Programms zur Bestimmung der Kreiszahl Pi via einer Monte-Carlo-basierten Annäherung. Bei diesem in der Lehre häufig präsentem Verfahren werden zufällig Punkte auf der Fläche eines Quadrats erzeugt, in das ein Viertelkreis einbeschrieben ist. Aus dem Verhältnis der Anzahl der Punkte innerhalb der Fläche des Viertelkreises und der Gesamtzahl der erzeugten Punkte lässt sich Pi herleiten. Da es sich bei dem Verfahren um eine Näherungslösung handelt, wird das Resultat exakter, je mehr Punkte erzeugt werden.

Das Erzeugen der Positionen der Punkte via Pseudozufallszahlen ist eine prozessorintensive Aufgabe, die parallel auf den Knoten eines Parallelrechners erfolgen kann. Der sequenzielle Anteil des Programms besteht aus dem Einsammeln der Zwischenergebnisse von den Knoten und deren Kombination zum Endergebnis auf einem Master-Knoten. Abbildung 2 zeigt, dass bei nur 1.000.000 Punkten die Problemgröße zu gering ist, um bei paralleler Programmausführung eine Beschleunigung zu erreichen. Jede Verzehnfachung der Problemgröße verbessert die Eignung des Programms für eine parallele Abarbeitung. Bei 10.000.000 und bei 100.000.000 Punkten wird eine maximale Beschleunigung erreicht und es kommt zum Abfallen der Leistung bei Hinzunahme weiterer Rekerne. Beeindruckend sind der SpeedUp und die Laufzeit bei 100.000.000.000 Punkten. Auf einem einzelnen Rechenkern ist die Laufzeit über 4 Stunden und 45 Minuten. Bei 512 Rechenkernen ist die Laufzeit gerade noch knapp über 1 Minute und 14 Sekunden. Es ergibt sich eine Beschleunigung von Faktor 229.

Ein anderes Beispiel, das Möglichkeiten und Grenzen paralleler Programmausführung zeigt und gleichzeitig ein greifbares Ergebnis liefert, ist die parallele Berechnung einzelner Bilder oder ganzer Filme mithilfe einer Raytracing-Anwendung wie zum Beispiel POV-Ray im Cluster. Auch ein solches Beispiel ist für die Lehre sehr gut geeignet, da es auch hier klar definierbare sequenzielle und parallelisierbare Programmanteile gibt [4].

{ AUFBAU VON CLUSTERN AUS EINPLATINENCOMPUTERN

Abb. 2 Untersuchung der Skalierbarkeit eines MPI-Programms zur Bestimmung der Kreiszahl π via Monte-Carlo-basierter Annäherung im Cluster mit 512 Rechenkernen

Abb. 3 Ein mobiler Cluster aus acht Raspberry Pi-Einplatinencomputern

Mobile Anwendungen

Wegen ihrer kompakten Bauweise, dem vergleichsweise geringen Gewicht und dem geringen Energieverbrauch, eignen sich Cluster aus Einplatinencomputern prinzipiell gut für mobile Anwendungen. Im Rahmen dieser Arbeit wurden verschiedene Cluster mit acht Knoten in Koffer aus Aluminium eingebaut. Diese enthielten außer den Knoten auch die komplette Stromversorgung und Netzwerkinfrastruktur (siehe Abb. 3).

Die Intention hinter diesem Vorhaben war es, den Studierenden semesterweise Cluster für Laborübungen auszuleihen [3]. Eine sinnvolle Einbindung ist in verschiedene Lehrveranstaltungen möglich.

Naheliegende Themengebiete sind verteilte/parallele Systeme und Echtzeitsysteme. Ein mobiles System, wie das in Abb. 3 dargestellte, kann für unter 600 Euro konstruiert werden. Zu den in Tab. 2 gezeigten Anschaffungskosten kommt lediglich noch der Preis für den Koffer.

Cloud-Plattformdienste und -Infrastrukturdienste

Cluster aus Einplatinencomputern können theoretisch eine gut geeignete Lösung zum Betrieb von Cloud-Plattformdiensten – Platform as a Service (PaaS) – sein, denn deren Hardwareressourcen sind für den Betrieb eines solchen Dienstes ausreichend. Bei einem Cloud-Plattformdienst handelt es sich im Prinzip um eine skalierbare Laufzeitumgebung für Webanwendungen. Der Dienst startet beim Import einer Webanwendung einen Application-Server und einen Lastverteiler. Steigt die Menge der Anfragen auf den Application-Server auf ein Maß, sodass weitere Ressourcen nötig werden, startet der Plattformdienst automatisch einen oder mehrere weitere Application-Server. Üblicherweise hat die Webanwendung in einem Plattformdienst auch noch Zugriff auf Infrastrukturdienste des Anbieters, wie beispielsweise Speicherdienste (Objektspeicher, Datenbankdienste, etc.), einen Email-Gateway, eine Benutzerverwaltung und einen Gateway für Kurznachrichten. Exemplarisch für die Funktionalität von Plattformdiensten zeigt Abb. 4 die Komponenten des öffentlichen Dienstangebots Google App Engine [13].

Eine Besonderheit der Google App Engine ist, dass es freie Softwarelösungen gibt, die dieses öffentlich verfügbare Dienstangebot reimplementieren. Somit ist es möglich, einen eigenen Plattformdienst mit der gleichen API und vergleichbarer Funktionalität zu betreiben. Die bekannteste Reimplementierung der Google App Engine ist AppScale [17].

Abb. 4 Komponenten der Google App Engine als Beispiel für einen Cloud-Plattformdienst

Um herauszufinden, ob ein Cluster aus Einplatinencomputern eine sinnvolle Hardwareplattform für den Betrieb eines AppScale-Plattformdienstes ist, wurden verschiedene Cluster aus Raspberry Pi 2 und ODROID-U3 (siehe Tab. 1) mit vier und acht Knoten mit AppScale 3.1 unter dem Betriebssystem Ubuntu 14.04 LTS installiert. Eine Herausforderung sind die Ressourcenanforderungen der verwendeten verteilten Datenbank Apache Cassandra. Nur auf dem Cluster aus ODROID-U3 war wegen seines doppelten Hauptspeichers im Gegensatz zum Raspberry Pi 2 ein vollständiger, wenn auch etwas zäher Betrieb des Plattformdienstes möglich [5].

Infrastrukturdienste – Infrastructure as a Service (IaaS) – eignen sich teilweise für einen Betrieb auf Clustern aus Einplatinencomputern. Dienste zum Betrieb virtueller Server sind ungeeignet, weil vollständige Virtualisierungslösungen via Typ-1-Hypervisor (z. B. VMware Workstation, Oracle VirtualBox oder KVM) sowie Paravirtualisierungslösungen via Typ-2-Hypervisor (z. B. XEN oder VMware ESX) auf Einplatinencomputern nicht sinnvoll betrieben werden können, da die verwendeten Prozessoren keine Funktionen zur Hardwarevirtualisierung bieten. Der Betrieb von Containern (z. B. Docker oder LXC) ist auf Einplatinencomputern möglich, spielt aber für Infrastrukturdienste keine große Rolle.

Besser eignen sich objektbasierte Speicherdienste für einen Betrieb auf Einplatinencomputern. Diese Dienste gehören auch zur Gruppe der Infrastrukturdienste und haben nur sehr geringe Anforderungen an die Hardwareressourcen. Häufig bieten solche Dienste die Schnittstelle des Amazon Simple Storage Service (S3) [18] und dessen Kernfunktionalität. Beispiele für solche Dienste, die einen Betrieb im Cluster ermöglichen, sind Ceph-RGW, Mino und RiakCS [6].

Herausforderungen beim Aufbau eines Clusters aus Einplatinencomputern

Beim Aufbau der verschiedenen Cluster im Rahmen dieser Arbeit traten zahlreiche Herausforderungen zutage. Exemplarisch geht dieser Artikel auf die Eigenarten der Energieversorgung der Cluster-Knoten, die Kühlung, die Positionierung der Knoten und die Qualität des persistenten Speichers ein.

Energieversorgung der Knoten

Die allermeisten untersuchten Einplatinencomputer, wie zum Beispiel die Vertreter der Raspberry Pi-

Familie, aber auch die Banana Pi verfügen zu diesem Zweck über einen micro-USB-Anschluss. Seltener kommt bei Einplatinencomputern, wie zum Beispiel beim ODROID-U3, ein Hohlstecker zum Einsatz. Alternativ ist zum Beispiel bei den Raspberry Pi die Stromversorgung über die GPIO-Pins (General Purpose Input Output) möglich. Ein Cluster aus mehreren Knoten kann somit beispielsweise über die 5V-Schiene eines PC-Netzteils versorgt werden. Da solche Lösungen in der Praxis häufig einen Bastelcharakter haben, erschienen USB-Netzteile als die bessere Wahl bei der Planung des großen Clusters.

Das Anschließen der USB-Kabel an einzelne Netzteile, wie sie von Smartphones bekannt sind, ist nicht zu empfehlen, da sie die Lösung mit dem schlechtesten Preis-Leistungs-Verhältnis sind, sehr viele Steckdosen benötigen und einen geringeren Wirkungsgrad als andere Lösungen haben [3]. Aus diesem Grund wurde beim Aufbau des Clusters auf USB-Netzteile mit mehreren Schnittstellen zurückgegriffen.

Zwar gleichen sich die untersuchten Einplatinencomputer in der benötigten elektrischen Spannung (5 V), aber in der benötigten Stromstärke unterscheiden sie sich teilweise deutlich.

Als gute Lösung für den Aufbau von Clustern haben sich Netzteile der Firma Anker mit 60 Watt Leistung und sechs oder zehn USB-Schnittstellen erwiesen. Bei diesen Netzteilen ist der maximal lieferbare Strom $60 \text{ Watt} / 5 \text{ Volt} = 12 \text{ Ampere}$. Der Hersteller des Raspberry Pi 3 empfiehlt für den Betrieb eine Stromversorgung, die bei 5 Volt 2,5 Ampere Strom abgeben kann [14], wobei hier davon ausgegangen wird, dass auch zusätzliche Hardware via USB angeschlossen wird, was bei den Knoten eines Clusters selten zu erwarten ist. Da im Cluster-Betrieb üblicherweise keine umfangreiche Verwendung von den GPIO-Pins und den USB-Anschlüssen gemacht wird, ist eine Stromaufnahme von 12 Ampere / 6 Schnittstellen = 2 Ampere pro Schnittstelle und Knoten ausreichend.

Wird alternativ ein Netzteil mit zehn USB-Schnittstellen und 60 W Leistung verwendet und sind an allen zehn Schnittstellen Cluster-Knoten angeschlossen, reduziert sich die Stromversorgung pro Schnittstelle auf nur noch 1,2 Ampere. Nach den im Rahmen dieser Arbeit gemachten Erfahrungen genügt dieser Strom zwar für den Betrieb des Raspberry Pi 1 und 2 sowie für den Banana Pi und den ODROID-U3, aber nicht für den

Abb. 5 Kühlkörper aus Aluminium können die Betriebstemperatur der Prozessoren der Raspberry Pi 3-Knoten nur um 2-3 Grad Celsius reduzieren

Raspberry Pi 3. In der Praxis kommt es bei Last auf den Knoten zu Abstürzen einzelner Knoten sowie zu zerstörten Dateisystemen und in einigen Fällen kam es auch zu zerstörten SD-Karten. Bei einem Netzteil mit zehn USB-Schnittstellen und 60 W Leistung müssen somit vier Schnittstellen ungenutzt bleiben.

Kühlung der Knoten

Die allermeisten Einplatinencomputer und alle in Tab. 1 dargestellten Systeme verfügen über keinerlei bewegliche Teile. Auch die Kühlung der Prozessoren erfolgt passiv und meist sogar ohne Kühlkörper. Für den Betrieb eines Clusters ist das von Vorteil, denn bewegliche Teile haben immer einen negativen Einfluss auf die Verfügbarkeit von Computern.

Die im Rahmen dieser Arbeit intensiv eingesetzten Systeme der Raspberry Pi-Familie zeigen je nach Versionsnummer ein ganz unterschiedliches Verhalten hinsichtlich Abwärme und Kühlungsbedarf. Der Raspberry Pi 1 sowie der Raspberry Pi 2 sind auch unter Volllast unauffällig, was die Abwärme angeht. Der Raspberry Pi 3 wird allerdings unter Volllast bei der standardmäßigen Taktrate mehr als 85 Grad Celsius heiß und stürzt nach wenigen Minuten ab. Da bewusst auf Lüfter verzichtet wurde, musste die Temperatur der Prozessoren durch ein Heruntertakten von 1,2 GHz auf 1 GHz abgesenkt werden. Durch diese Reduktion der Taktrate bleibt die Temperatur der Prozessoren auch unter Volllast unter 85 Grad Celsius.

Versuche, die Betriebstemperatur der Prozessoren der Raspberry Pi 3 mit zugekauften Kühlkörpern aus Aluminium (siehe Abb. 5) signifikant zu reduzieren, waren wenig erfolgreich. Die durchschnittliche

Abb. 6 Die Positionierung der Knoten mit Abstandsbolzen ist eine einfach zu handhabende und kostengünstige Option bei gleichzeitig guter Belüftung der Knoten und Erreichbarkeit der Anschlüsse

Temperatur im Ruhezustand konnte dadurch nur von 59 auf 56 Grad Celsius reduziert werden. Die durchschnittliche Temperatur unter Volllast beim High-Performance Linpack (HPL) Benchmark konnte nur von 82 auf 80 Grad Celsius reduziert werden.

Positionierung der Knoten

Wegen der kompakten Bauweise haben die gängigen Einplatinencomputer an fast allen Seiten Anschlüsse, von denen im Betrieb als Cluster-Knoten zwar die allermeisten nicht verwendet werden, aber dennoch fast alle Seiten frei zugänglich sein müssen. Am Beispiel des Raspberry Pi 3 müssen wegen der microSD-Karte, dem Stromanschluss und der Ethernet-Schnittstelle drei Seiten erreichbar sein (siehe Abb. 6). Dieser Umstand macht die Positionierung der Knoten eines großen Clusters zur Herausforderung.

Verschiedene Optionen zur Positionierung und Befestigung der Knoten sind denkbar. Die Positionierung der Knoten auf einem Brett (siehe Abb. 3) oder auf einem vergleichbaren Träger ist nur bei Clustern aus wenigen Knoten sinnvoll machbar.

Tabelle 4

Gemessene Schreibgeschwindigkeiten bei zufälligen Schreibzugriffen auf verschiedene microSD-Karten

Hersteller	Format	Kapazität [GB]	Speed Class	Random Schreibgeschwindigkeit ¹ [kB/s] mit Datensatzgröße [kB] ...								
				4	8	16	32	64	128	256	512	1024
Verbatim	microSD	16	4	121	257	586	962	1272	1780	2310	2894	3417
Samsung	microSD	16	6	133	272	631	1055	1485	2028	2592	2994	3473
Kingston	microSD	16	10	224	13	26	53	108	219	446	917	1939
Samsung	microSD	16	10	128	263	581	1006	1440	2018	2589	3043	3430
SanDisk	microSD	16	10	334	419	41	83	167	336	672	1345	2778
SONY	microSD	16	10	216	12	25	51	102	207	420	845	1807

¹ Gemessen mit: `iozone -RaeI -i 0 -i 1 -i 2 -y 4k -q 1M -s 500m -o -f /tmp/testfile`

Bei einem großen Cluster mit 128 oder mehr Knoten ist das keine Option. Eine häufig verwendete Möglichkeit bei Einplatinencomputern ist die Konstruktion eines Gehäuses aus Lego oder mit einem vergleichbaren System. Diese Möglichkeit wurde aber verworfen, weil die Belüftung in einem solchen Fall nicht optimal ist. Den gleichen Nachteil haben fertige Gehäuse für Einplatinencomputer. Diese verschlechtern die Luftzirkulation signifikant. Fertige Konstruktionen zum Aufbau von Clustern aus Einplatinencomputer existieren zwar, sind aber kostspielig in der Anschaffung.

Die im Rahmen dieses Projekts gewählte Lösung ist der Aufbau von Stapeln aus bis zu acht Knoten und deren Befestigung mit Abstandsbolzen im Format M2,5, entsprechend den Bohrlöchern. Diese Stapel wurden auf Bretter geschraubt, um sie gegen ein Umfallen abzusichern. Diese Lösung ist einfach in der Handhabung, kostengünstig und erlaubt gleichzeitig eine gute Belüftung der Knoten und Erreichbarkeit der Anschlüsse.

Realisierung eines leistungsfähigen persistenten Blockspeichers

Es existieren nur wenige Optionen zur Anbindung eines leistungsfähigen persistenten Blockspeichers an die Knoten. Die Knoten selbst haben keinen eingebauten Blockspeicher, dafür aber einen Steckplatz für eine (micro)SD-Karte (siehe Tab. 1). Auf dieser Karte befindet sich auch das Betriebssystem. Die Leistungsdaten von microSD-Karten bei sequenziellen Lese- und Schreibzugriffen können je nach Blockgröße bis zu 20 MB pro Sekunde erreichen. Die Geschwindigkeit bei den, besonders bei Multitasking-Betriebssystemen in der Praxis rele-

vanten, zufälligen Schreibzugriffen ist üblicherweise sehr gering [3]. Tabelle 4 enthält einige Messergebnisse, die mit dem Dateisystem-Benchmark IOzone und dem Dateisystem Journaling-Dateisystem ext4 (fourth extended filesystem) gemessen wurden.

Bei den Messergebnissen in Tab. 4 fällt auf, dass einige Karten offenbar für zufällige Schreibzugriffe auf 4 kB große Blöcke optimiert sind. Nur wenige Speicherkarten liefern eine Steigerung der Schreibgeschwindigkeit bei wachsender Blockgröße, wie es eigentlich zu erwarten ist.

Eine weitere Möglichkeit, persistenten Speicher an die Knoten anzubinden, bieten die USB-Schnittstellen. Diese arbeiten allerdings nach dem USB-Standard 2.0 und liefern in der Praxis abhängig vom verwendeten Speicher und Dateisystem nur maximal 25 bis 30 MB pro Sekunde bei sequenziellen Zugriffen. Bei zufälligen Schreibzugriffen, z. B. auf USB-Sticks, werden in der Praxis üblicherweise auch nur unter 1 MB pro Sekunde Datenübertragungsrate erreicht.

Abschließend bleibt noch die Möglichkeit, persistenten Blockspeicher über die Ethernet-Schnittstelle der Knoten zu erreichen. Diese ist zwar bei der Raspberry Pi-Familie auch nur intern via USB 2.0 verbunden, dennoch können verteilte Dateisysteme, wie zum Beispiel GlusterFS, XtremFS, Ceph oder OrangeFS, oder alternativ das Protokoll NFS, je nach Konfiguration und verwendetem Speicher einen leistungsfähigeren Speicher bereitstellen, als das mit den anderen genannten Optionen möglich ist. Ein positiver Nebeneffekt der verteilten Dateisysteme ist, dass diese häufig schon selbst eine redundante Datenhaltung rea-

lisieren können und somit die Datensicherheit verbessern.

Fazit zu Clustern aus Einplatinencomputern

Einplatinencomputer wie die Vertreter der Raspberry Pi-Familie sind nicht unbedingt die erste Wahl beim Aufbau eines Clusters und sie können mit Ihren Leistungsparametern nicht mit Servern, Workstations oder PCs aus Standardkomponenten konkurrieren. Sie sind aber wegen der vergleichsweise geringen Anschaffungs- und Betriebskosten überall dort eine kostengünstige Alternative, wo viele physische Rechenkerne hilfreich sind. Typische Anwendungsgebiete sind die Lehre, aber auch überall dort, wo die Skalierbarkeit von Programmen untersucht wird. Weitere sinnvolle Anwendungsgebiete sind der Aufbau mobiler Systeme und der Betrieb von Clustern zur Realisierung skalierbarer Speicherdienste.

Open Access. Dieser Artikel wird unter der Creative Commons Namensnennung 4.0 International Lizenz (<http://creativecommons.org/licenses/by/4.0/deed.de>) veröffentlicht, welche die Nutzung, Vielfältigung, Bearbeitung, Verbreitung und Wiedergabe in jeglichem Medium und Format erlaubt, sofern Sie den/die ursprünglichen Autor(en) und die Quelle ordnungsgemäß nennen, einen Link zur Creative Commons Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden.

Literatur

1. Abrahamsson P, Helmer S, Phaphoom N, Nicolodi L, Preda N, Miori L, Bugoloni S (2013) Affordable and Energy-Efficient Cloud Computing Clusters: The Bolzano Raspberry Pi Cloud Cluster Experiment. In: IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), 2013, Vol. 2, pp 170–175
2. Bauke H, Mertens S (2006) Cluster Computing. Springer, Berlin
3. Baun C (2016) Mobile Clusters of Single Board Computers: An Option for providing Resources to Student Projects and Researchers. SpringerPlus 2016 5:360, <http://springerplus.springeropen.com/articles/10.1186/s40064-016-1981-3>, last access: 4.12.2017
4. Baun C (2016) Parallel Image computation in Clusters with task-distributor. SpringerPlus 2016, 5:632, <http://springerplus.springeropen.com/articles/10.1186/s40064-016-2254-x>, last access: 4.12.2017
5. Baun C (2017) Lessons learned from implementing a scalable PaaS service by using single board computers. Int J Cloud Comput Serv Archit 7(2):1–11, <http://aircconline.com/ijccsa/V7N2/7217ijccsa01.pdf>, last access: 4.12.2017
6. Baun C, Cocos HN, Spanou RM (2017) Performance aspects of object-based storage services on single board computers. Open J Cloud Comput 4(1):1–16, https://www.ronpub.com/OJCC_2017v4i1n01_Baun.pdf, last access: 4.12.2017
7. Cloutier MF, Paradis C, Weaver VM (2014) Design and Analysis of a 32-bit Embedded High-Performance Cluster Optimized for Energy and Performance. In: Proceedings of the 1st International Workshop on Hardware-Software Co-Design for High Performance Computing 2014, IEEE Press, pp 1–8
8. Cox S, Cox J, Boardman R, Johnston S, Scott M, O'Brien N (2014) Iridis-Pi: A low-cost, compact demonstration cluster. Cluster Comput 17(2):349–358
9. Dinan J, Balaji P, Lusk E, Sadayappan P, Thakur R (2010) Hybrid Parallel Programming with MPI and Unified Parallel C. In: Proceedings of the 7th ACM International Conference on Computing Frontiers 2010, pp 177–186
10. <http://www.mcs.anl.gov/research/projects/mpi/>, last access: 4.12.2017
11. <http://www.netlib.org/benchmark/hpl/>, last access: 4.12.2017
12. <http://www.raspbian.org/>, last access: 4.12.2017
13. <https://cloud.google.com/appengine/>, last access: 4.12.2017
14. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>, last access: 4.12.2017
15. <https://wiki.ubuntu.com/ARM/RaspberryPi>, last access: 4.12.2017
16. Kiepert J (2013) Creating a Raspberry Pi-Based Beowulf Cluster, http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster_v2.pdf, last access: 4.12.2017
17. Krintz C (2013) The AppScale Cloud Platform: Enabling portable, scalable Web Application Deployment. IEEE Internet Comput 17(2):72–75
18. Palankar MR, Iamnitchi A, Ripeanu M, Garfinkel S (2008) Amazon S3 for Science Grids: A Viable Solution? In: Proceedings of the 2008 International Workshop on Data-Aware Distributed Computing, ACM, pp 55–64