Cluster aus Einplatinencomputern

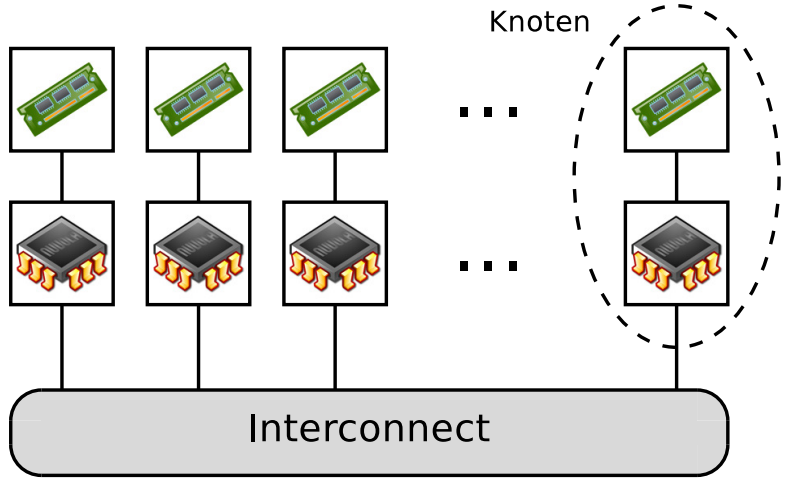
Prof. Dr. Christian Baun

christianbaun@fb2.fra-uas.de

Cluster-Computing

Cluster-Computing

Clustering ist paralleles Rechnen auf Systemen mit verteiltem Speicher

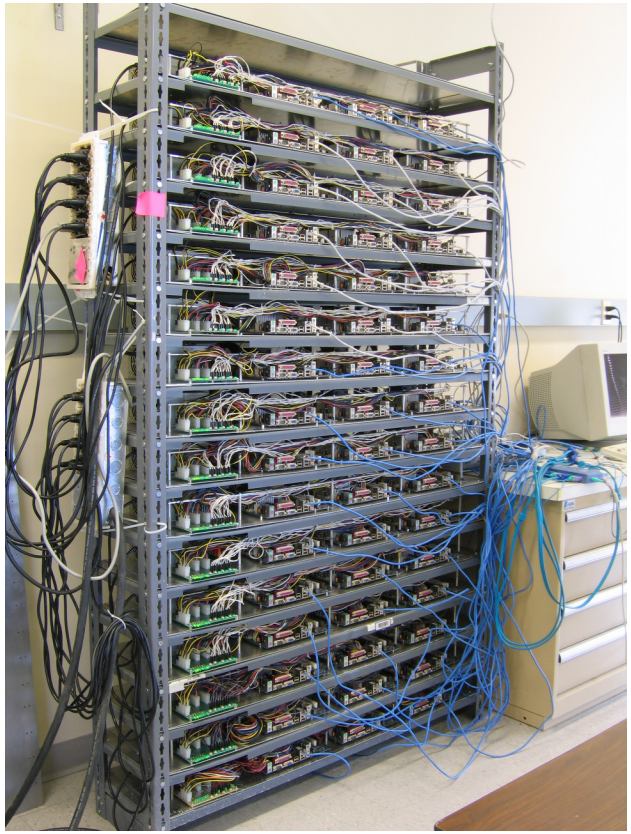


- Ein Cluster ist ein Verbund aus mindestens 2 Knoten
- Jeder Knoten ist ein eigenständiges Rechnersystem
- Die Knoten sind durch ein Netzwerk miteinander verbunden
- Knoten sind meist: PCs aus Standardkomponenten oder Server (seltener Workstations oder Supercomputer)

Gründe für den Einsatz von Clustern

- Verbesserung der Rechenleistung $\Longrightarrow$ High Performance Clustering
- Verbesserung der Ausfallsicherheit $\Longrightarrow$ High Availability Clustering
- Verbesserung des Datendurchsatzes $\Longrightarrow$ High Throughput Clustering

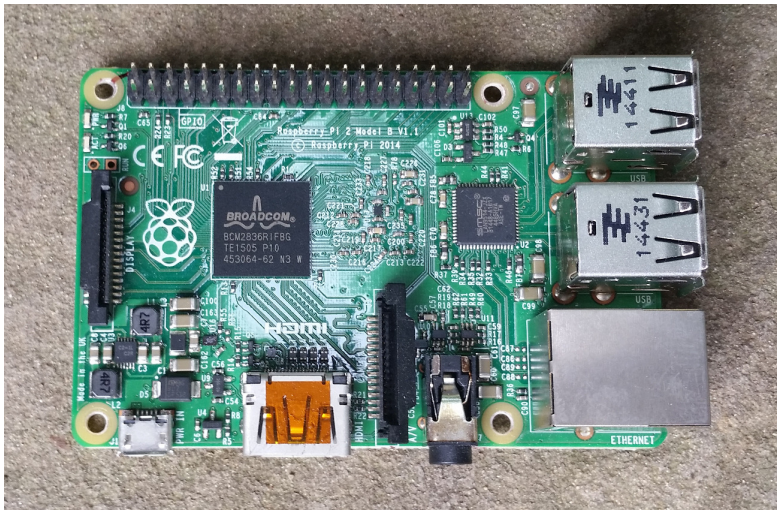
Stand der Technik beim Cluster-Computing



- Vorteile von Blades und Servern: Redundante Komponenten, leistungsfähige Verwaltungs- und Wartungswerkzeuge (z.B. HPE iLO)
- Vorteil von PC-Komponenten: Geringer Anschaffungspreis

Bildquelle (rechts): `http://physics.bu.edu/~sandvik/clusters.html`

Cluster aus Einplatinencomputern



- Beispiel: Cluster aus 8 Raspberry Pi 2

8 Einplatinencomputer	320 €
8 microSD-Karten	100 €
8 microUSB-Kabel	10 €
9 Ethernet-Kabel	10 €
Stromversorgung	50 €
Switch	40 €
Summe:	560 €

- Einplatinencomputer sind billiger in der Anschaffung und brauchen weniger Strom als PCs
  $\Longrightarrow$ Idee 2013/2014: **Cluster aus Einplatinencomputern aufbauen**
- Stromverbrauch unter Last: ca. 35 W

$$0,035\,\text{kW} * 24\,\frac{\text{h}}{\text{Tag}} * 365,25\,\frac{\text{Tag}}{\text{Jahr}} * 0,30\,\frac{€}{\text{kWh}}$$

$$\approx 92,05\,\frac{€}{\text{Jahr}}$$

- Dieser Cluster hat immerhin 32 CPU-Kerne
- Aber leider nur 8 GB RAM
  $\Longrightarrow$ Eher ungeeignet für In-Memory-Datenbanken

Sinnvolle Einsatzgebiete für Cluster aus Einplatinencomp.

- **Verteiltes/Paralleles Rechnen**
  - Viele physische CPU-Kerne für kleines Geld
- **Mobile Anwendungen**
  - Kompakte Bauweise ist möglich und das Gewicht ist vergleichsweise gering
- **Cloud-Plattformdienste für Webanwendungen (PaaS)**
- **Cloud-Infrastrukturdienste (IaaS)**
  - Nur Speicherdienste! VMs nicht sinnvoll möglich, da zu wenig RAM
- **Testumgebung für verteilte Dateisysteme**

Bewilligungen: Förderprogramm „Forschung für die Praxis“ (Ausschreibungsrunde 2014)

#	Antragsteller/innen	Hochschule	Titel des Vorhabens	Bewilligte Mittel
1	Baun, Prof. Dr. Christian	Frankfurt University of Applied Sciences	Entwicklung, Aufbau und Evaluation von Clustern, Cloud-Infrastrukturdiensten und -Plattformdiensten auf Basis moderner Einplatinencomputer	35.000,00 €

Verschiedene Einplatinencomputer wurden untersucht

	Rasp. Pi 1	**BananaPi 1** (ab Mitte 2014)	**ODROID-U3**	**Rasp. Pi 2** (ab Mitte 2015)	**Rasp. Pi 3** (ab Mitte 2016)
CPU	ARM 11	ARM A7	ARM A9	ARM A7	ARM A53
CPU-Kerne	1	2	**4**	**4**	**4**
Taktrate	700 MHz	900 MHz	**1700 MHz**	900 MHz	1200 MHz
RAM	512 MB	1024 MB	**2048 MB**	1024 MB	1024 MB
Ethernet	100 Mbit	**1000 Mbit**	100 Mbit	100 Mbit	100 Mbit
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	++	+	+

Die Bewertungen sind natürlich subjektiv

- Von diesen Einplatinencomputern wurden mehrere beschafft und mehrere Cluster aufgebaut

Verteiltes/Paralleles Rechnen (1/7) – HPL-Benchmark

¯_(ツ)_/¯ Wie schnell sind Cluster aus Einplatinencomputern?

- FLOPS mit dem High-Performance Linpack (HPL)
  - Der HPL löst ein lineares Gleichungssystem und misst dabei die FLOPS

    `http://www.netlib.org/benchmark/hpl/`

FLOPS/s = Gleitkommazahlen-Operationen (*floating-point operations*) pro Sekunde (Additionen oder Multiplikationen)

Einplatinencomputer	#Knoten	#CPU-Kerne	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

¯_(ツ)_/¯ Ist das gut?

Verteiltes/Paralleles Rechnen (2/7)

Screenshot von Wikipedia (2.9.2017)

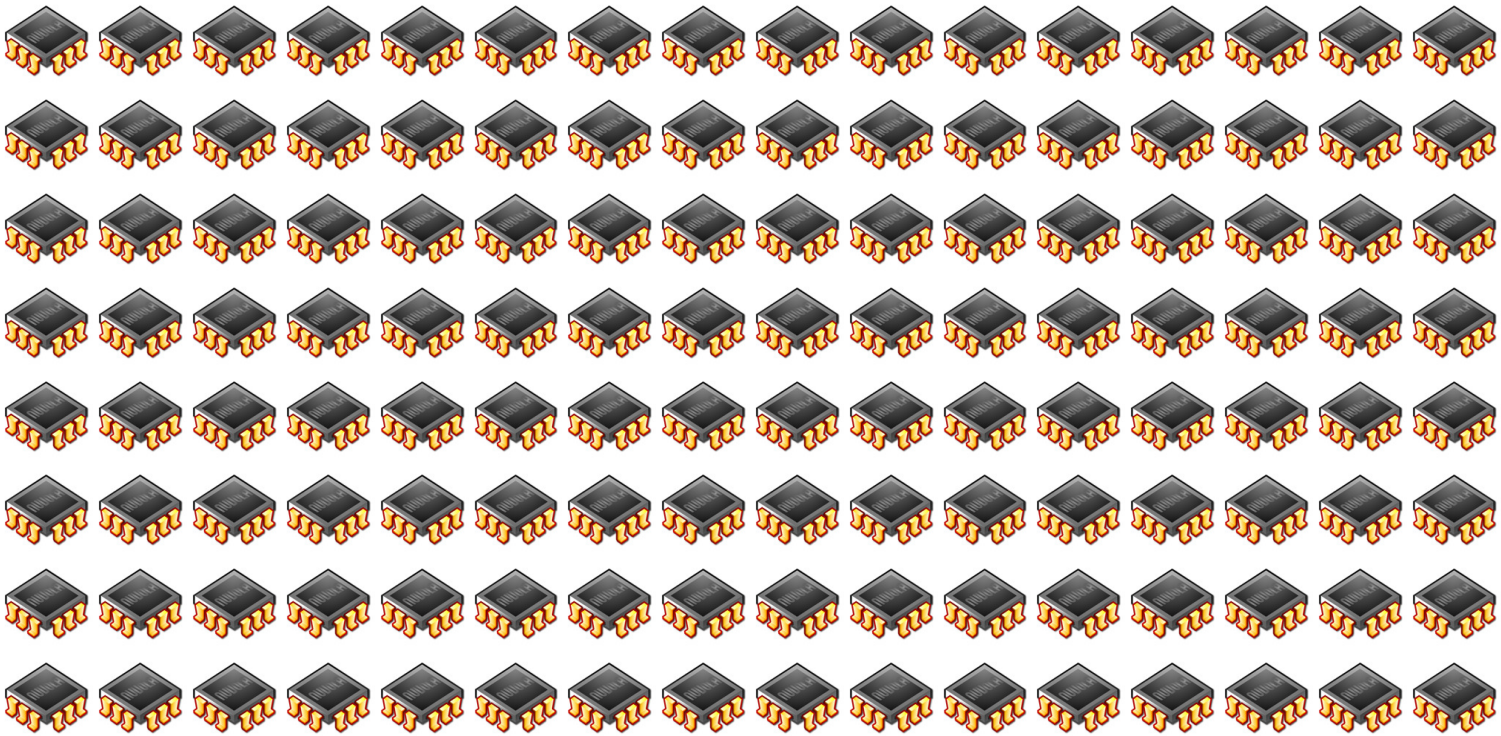
LINPACK-Benchmarkergebnisse (Stand Juni 2016)

Ausgewählte LINPACK-Benchmarks

Computer/CPU	Beschreibung	Position in den TOP500[1]	LINPACK Rmax in GigaFLOPS
Sunway TaihuLight	National Supercomputer Center, Jiangsu, China	1.	ca. 93.014.600
Tianhe-2	National University of Defense Technology, China	2.	ca. 33.862.700
Titan	DOE/SC/Oak Ridge National Laboratory	3.	ca. 17.590.000
Sequoia	DOE/NNSA/LLNL	4.	ca. 17.173.200
K Computer	RIKEN Advanced Institute for Computational Science (AICS), Japan	5.	ca. 10.510.000
Mira (Supercomputer)	Argonne National Laboratory	6.	ca. 8.586.600
2 × Intel Xeon DP X5680, 3,33 GHz	Workstation im Jahr 2010 (64 Bit)	–	ca. 94,8 [2]
Intel Core i7, 3,20 GHz, 4 Kerne	Standard-PC im Jahr 2009 (64 Bit)	–	ca. 33,0 [3]
Intel Core 2 Quad, 2,66 GHz	Standard-PC im Jahr 2007 (64 Bit)	–	ca. 23,5 [4]
Intel Core 2 Duo, 2,66 GHz	Standard-PC im Jahr 2007 (64 Bit)	–	ca. 12,5 [4]
AMD Athlon 64 X2 6000+, 3,00 GHz	Standard-PC im Jahr 2007 (64 Bit)	–	ca. 8,4 [4]
Intel Itanium 2, 1,6 GHz	Workstation (64 Bit)	–	ca. 6,4 [5]
Intel Pentium 4, 3,2 GHz	Standard-PC im Jahr 2003	–	ca. 3,1 [5]
Intel Pentium II, 450 MHz	Standard-PC im Jahr 1999	–	ca. 0,4
Raspberry Pi, 700 MHz	Educational Board	–	ca. 0,01625
Intel 386DX, 33 MHz	Standard-PC im Jahr 1989	–	ca. 0,008

¯_(ツ)_/¯ Wo ist der Sinn von Clustern aus Einplatinencomputern?

Verteiltes/Paralleles Rechnen (3/7)



Ein Cluster aus 32 Raspberry Pi 2 hat immerhin 128 CPU-Kerne

(und verbraucht nur ca. 110 W)

32 Raspberry Pi 2	1.280 €
32 microSD-Karten	400 €
32 microUSB-Kabel	30 €
33 Ethernet-Kabel	30 €
Stromversorgung	150 €
2 Switche	80 €
Summe:	1.970 €

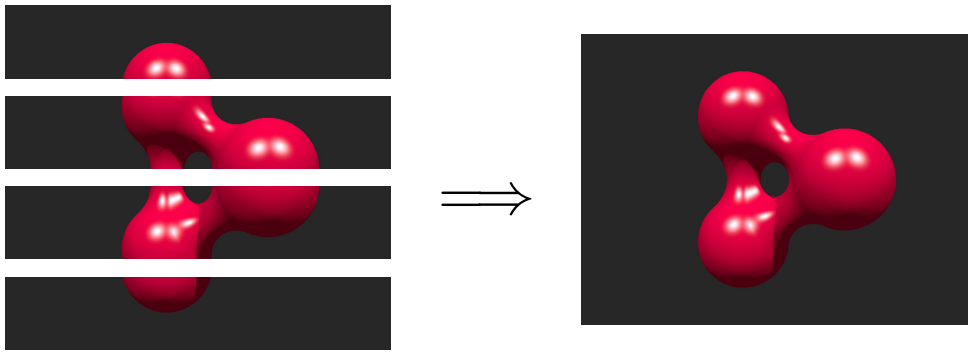
Mit physischen CPU-Kernen kann man spannende Dinge in den verteilten Systemen machen

Verteiltes/Paralleles Rechnen (4/7) – task-distributor

- 2015: Interessante Anwendung gesucht, die auch Studenten Spaß macht $\Longrightarrow$ Raytracing (mit POV-Ray)
- Problem: Es existierte keine funktionierende Lösung zur parallelen Bildberechnung im Cluster
- Lösung: Selbst implementieren $\Longrightarrow$ task-distributor

  `https://github.com/christianbaun/task-distributor`

Parallele Berechnung der Teilbilder auf den Rechenknoten

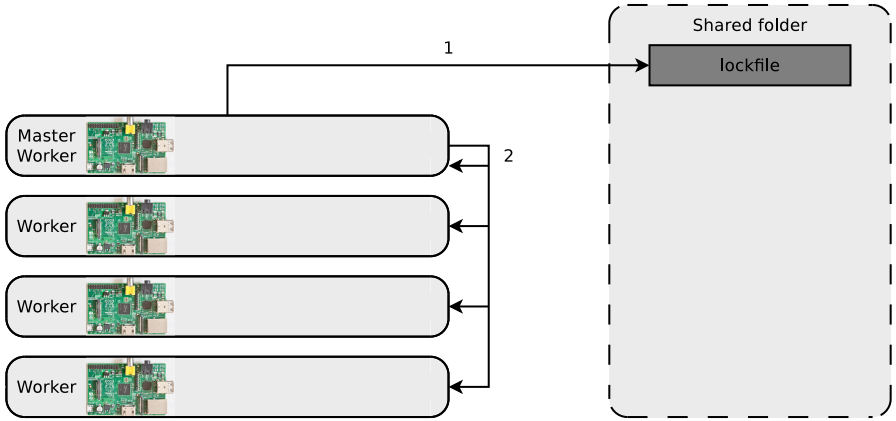


Kombination der Teilbilder zum endgültigen Bild auf einem Knoten

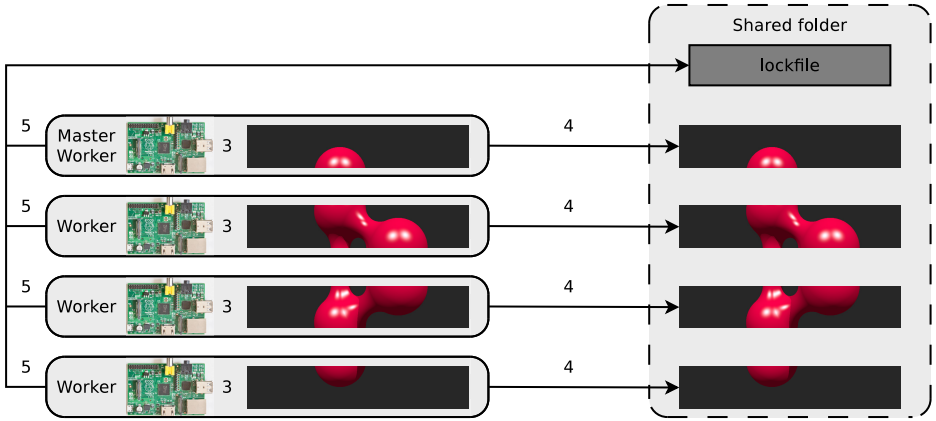
Parallel image computation in clusters with task-distributor. *Christian Baun*. SpringerPlus 2016 5:632.
`http://springerplus.springeropen.com/articles/10.1186/s40064-016-2254-x`

Verteiltes/Paralleles Rechnen (5/7) – task-distributor

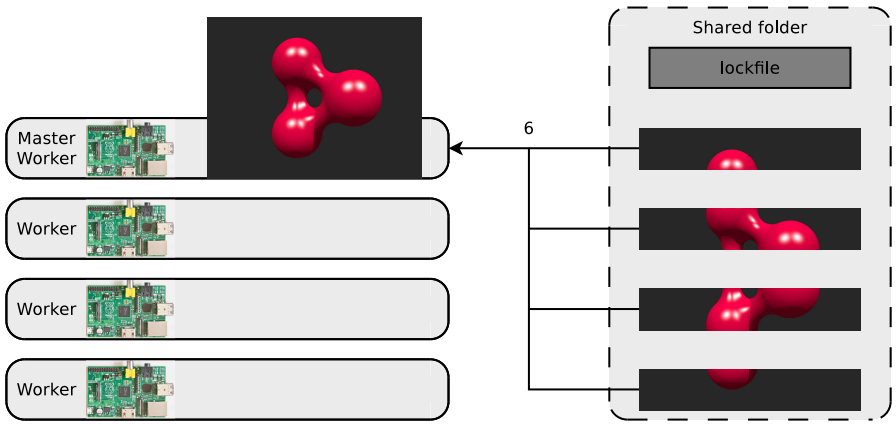
Der Master erzeugt eine Lockdatei (1) und startet die Raytracing-Jobs (2) $\Longrightarrow$ **sequentieller Anteil 1**



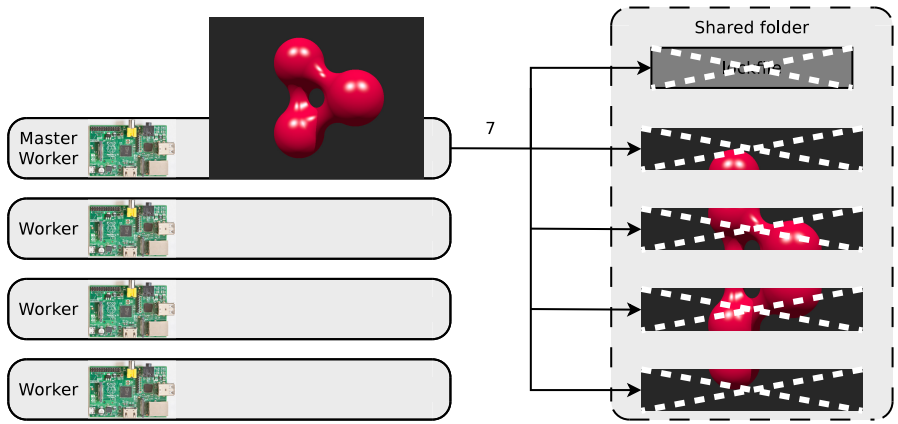
Die Worker berechnen die Teilbilder (3), kopieren diese in einen gemeinsamen Ordner (4) und schreiben ihre Hostnamen in die Lockdatei (5) $\Longrightarrow$ **paralleler Anteil**



Der Master kombiniert die Teilbilder zum endgültigen Bild (6) $\Longrightarrow$ **sequentieller Anteil 2**

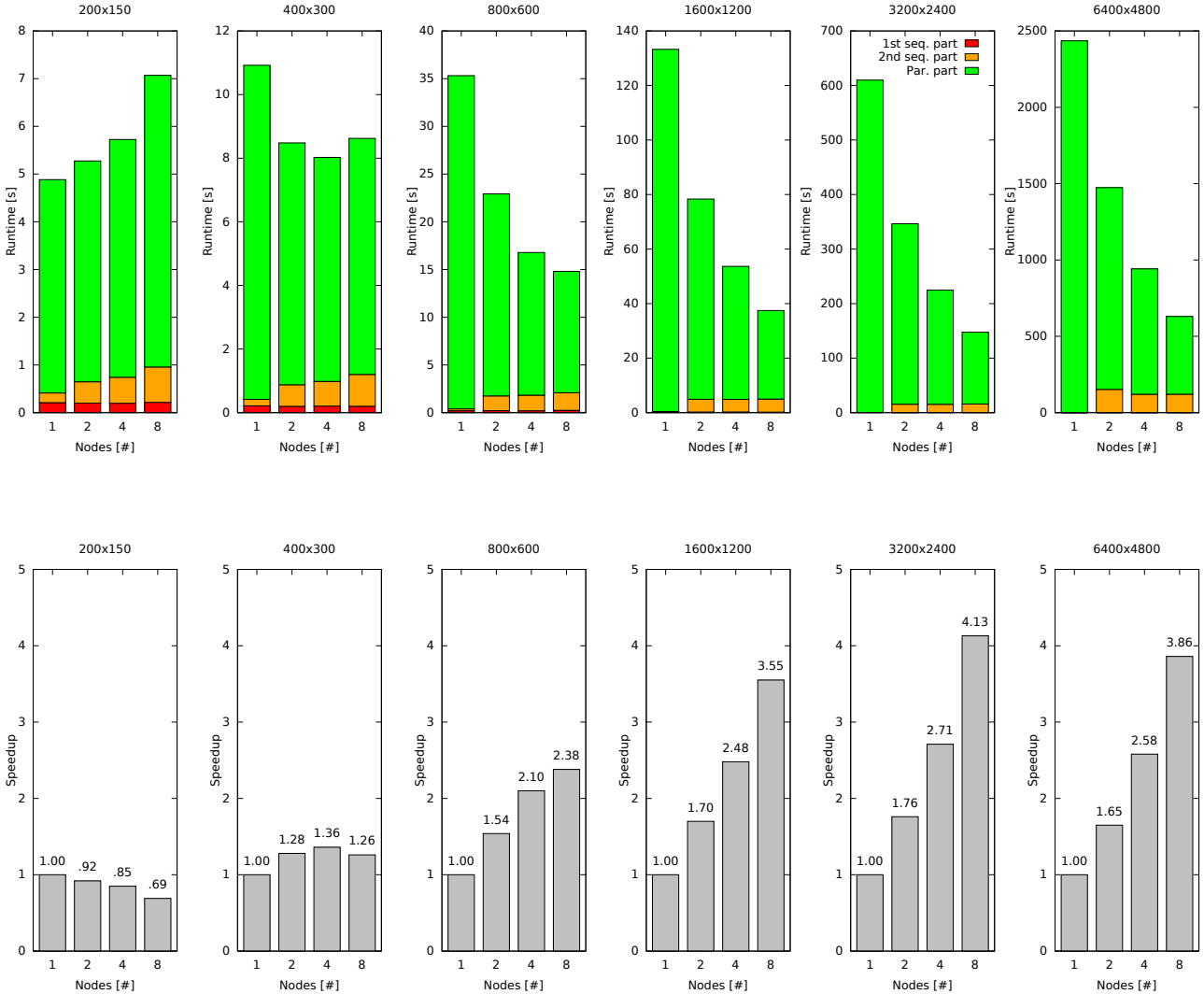


Der Master räumt den gemeinsamen Ordner auf (7) $\Longrightarrow$ **sequentieller Anteil 2**



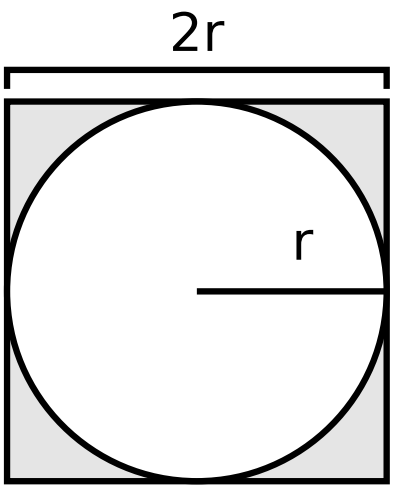
Verteiltes/Paralleles Rechnen (6/7) – Gesetze + Probleme

- Wir sehen: Gesetze und Probleme der verteilten Systeme

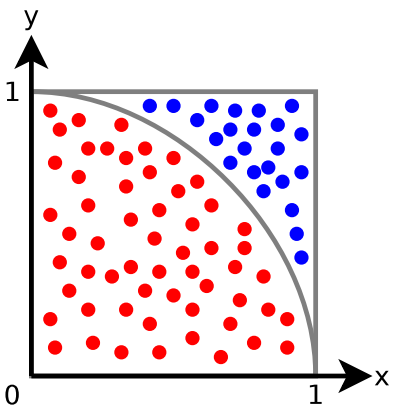


- **Amdahls Gesetz**: Der Geschwindigkeitszuwachs wird durch den sequentiellen Anteil des Problems beschränkt
  $\Longrightarrow$ **Tapezierbeispiel**
- **Gustafsons Gesetz**: Ein genügend großes Problem kann effizient parallelisiert werden
- Der sequentielle Teil wird mit zunehmender Anzahl an CPUs unbedeutender
- **Swap** bei 6400x4800 (`convert` braucht fast 500 MB RAM zum Zusammenfügen der Teilbilder. Wir hatten aber nur 512 MB
  - 16 MB für die GPU
  - Platz für Linux)

Exkurs: Berechnung von π via Monte-Carlo-Simulation



r	=	Radius
A	=	Flächeninhalt
K	=	Kreis
Q	=	Quadrat



- Einen Kreis mit Radius r in ein Quadrat mit Seitenlänge $2r$ einbeschreiben
- Zufällig Punkte im Quadrat erzeugen
  - Anzahl der Punkte auf A_K im Verhältnis zur Anzahl der Punkte auf A_Q ist gleich dem Flächenverhältnis

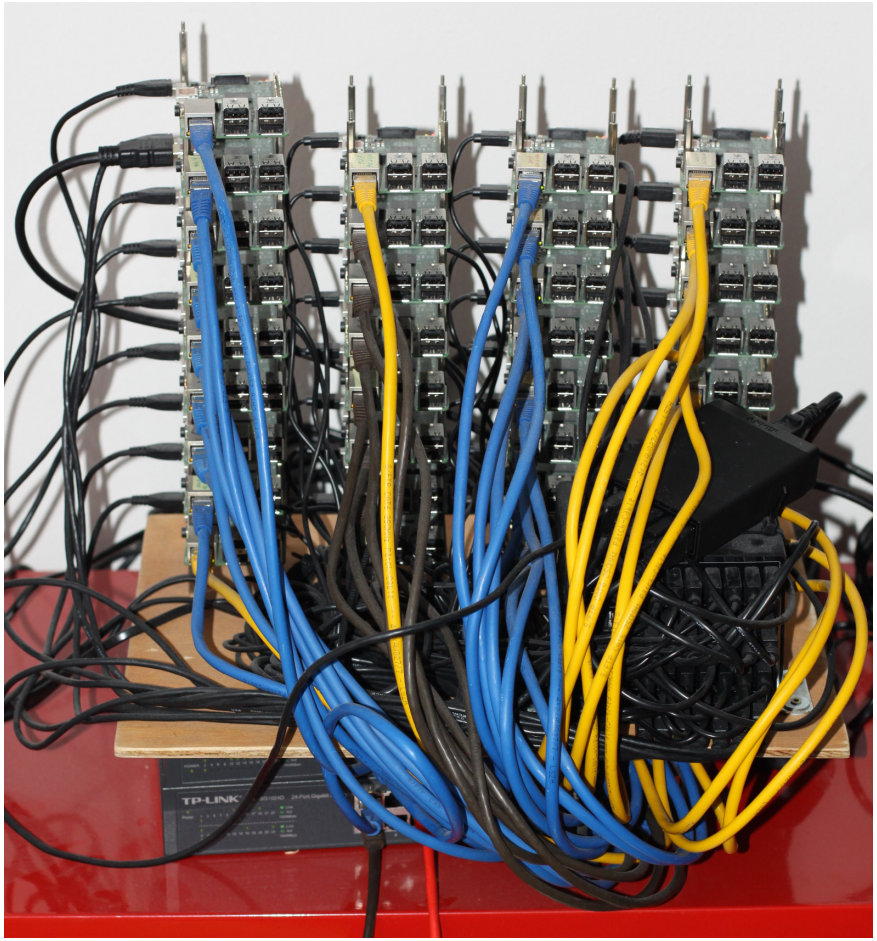
$$\frac{A_K}{A_Q} = \frac{\pi \cdot r^2}{(2 \cdot r)^2} = \frac{\pi \cdot r^2}{4 \cdot r^2} = \frac{\pi}{4}$$

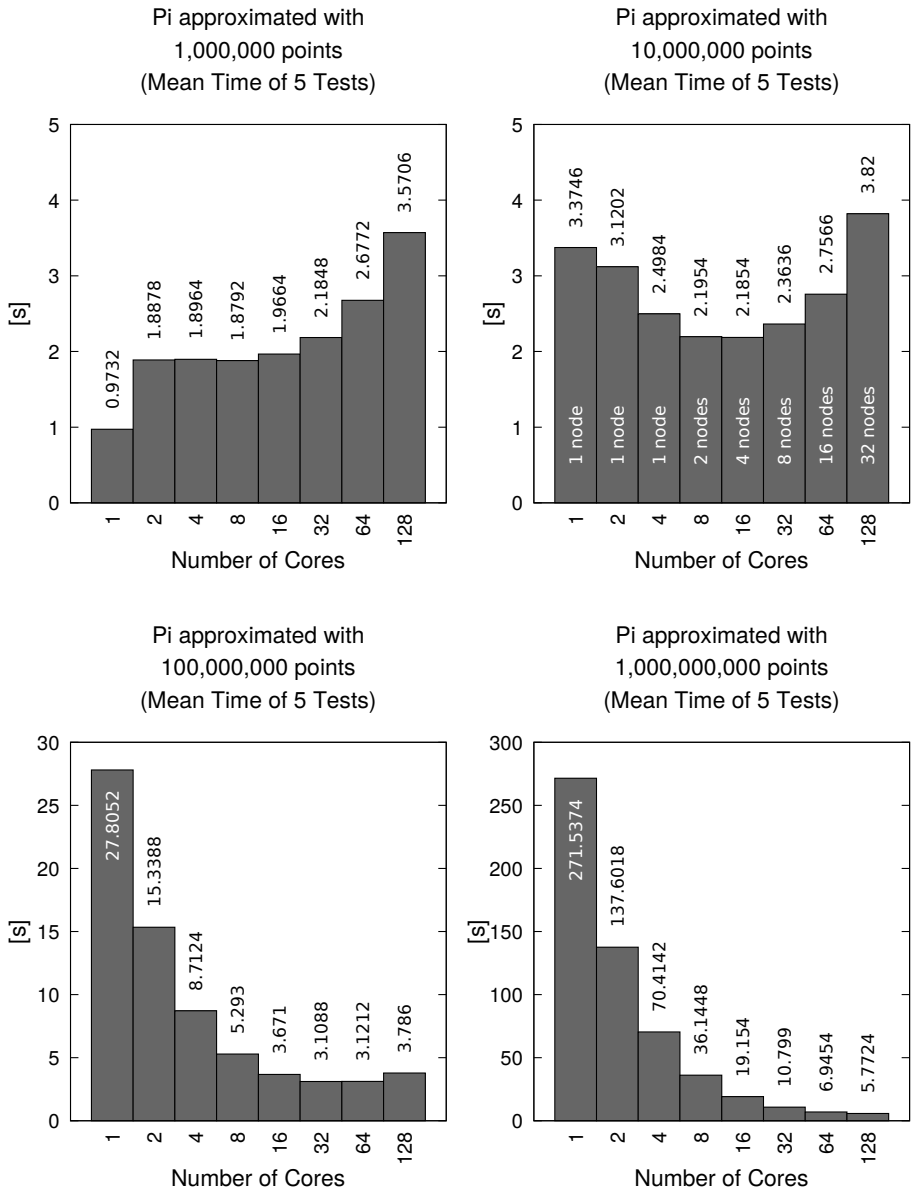
- Zufälliges Erzeugen der Punkte (X/Y-Achsenwerte via `random`) kann durch die Worker parallelisiert werden
- Der Master erhält von jedem Worker die Anzahl der erzeugten Punkte in A_K und berechnet:

$$\frac{4 \cdot \text{Punkte in } A_K}{\text{Punkte in } A_Q} = \pi$$

Schöne Quelle: `https://mathrule.wordpress.com/2011/01/27/mpi-calculation-of-pi-using-the-monte-carlo-method/`

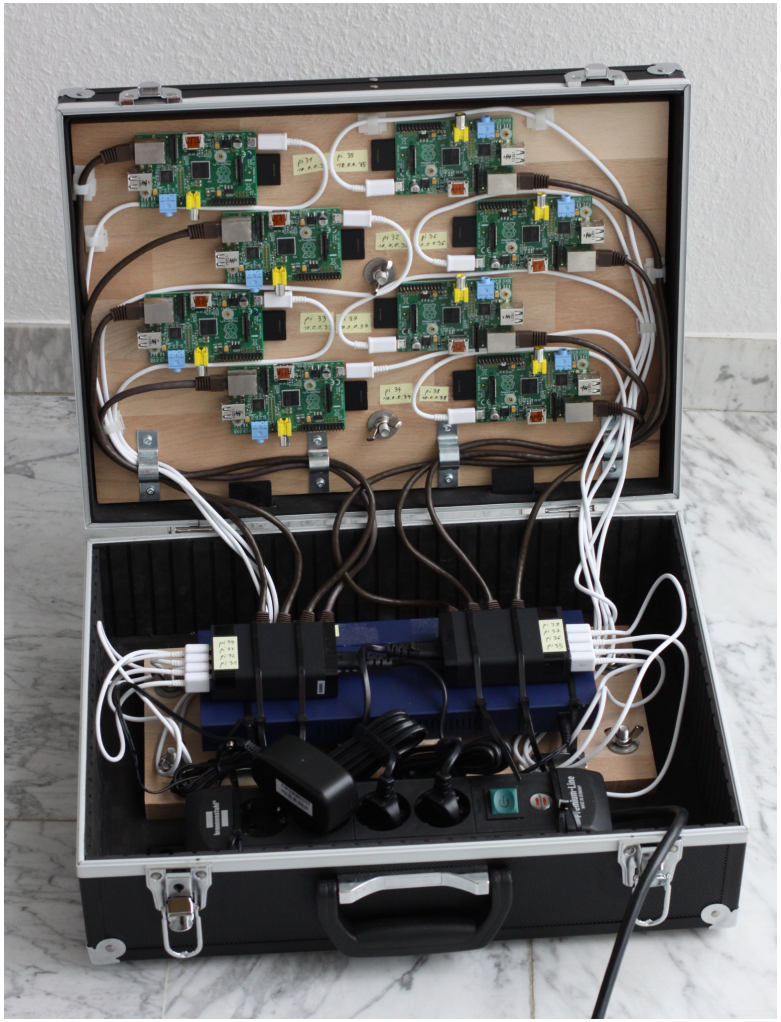
Verteiltes/Paralleles Rechnen (7/7) – MPI auf „Pinky“





Performance and Energy-Efficiency Aspects of Clusters of Single Board Computers. *Christian Baun*. International Journal of Distributed and Parallel Systems (IJDPS), Vol.7, No.2/3/4, 2016, S.13-22.
`http://aircconline.com/ijdps/V7N4/7416ijdps02.pdf`

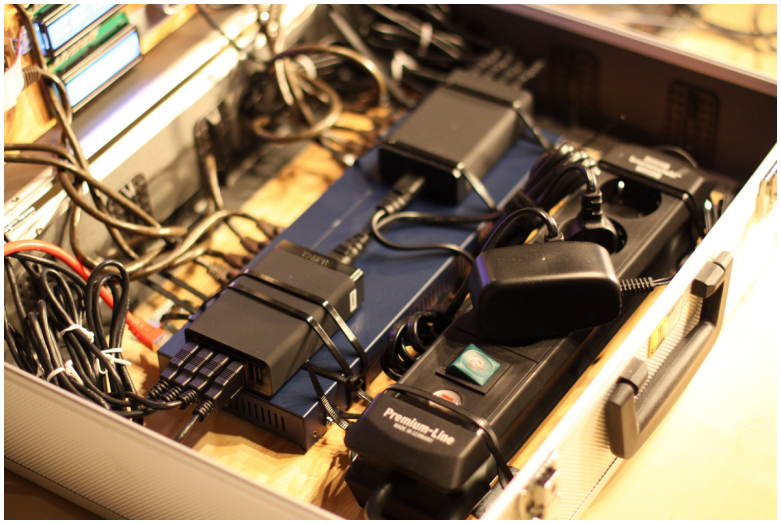
Mobile Anwendungen (1/2)

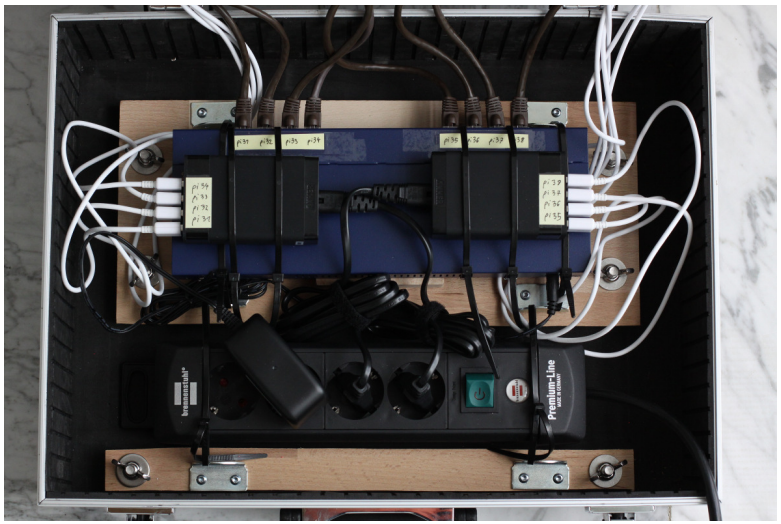


- Cluster mit 8 Knoten (RasPi 1) wurden in Alu-Koffer eingebaut
- Idee: Koffer an Studenten Semesterweise ausleihen

Mobile clusters of single board computers: an option for providing resources to student projects and researchers. *Christian Baun*. SpringerPlus 2016 5:360.
`http://springerplus.springeropen.com/articles/10.1186/s40064-016-1981-3`

Mobile Anwendungen (2/2)



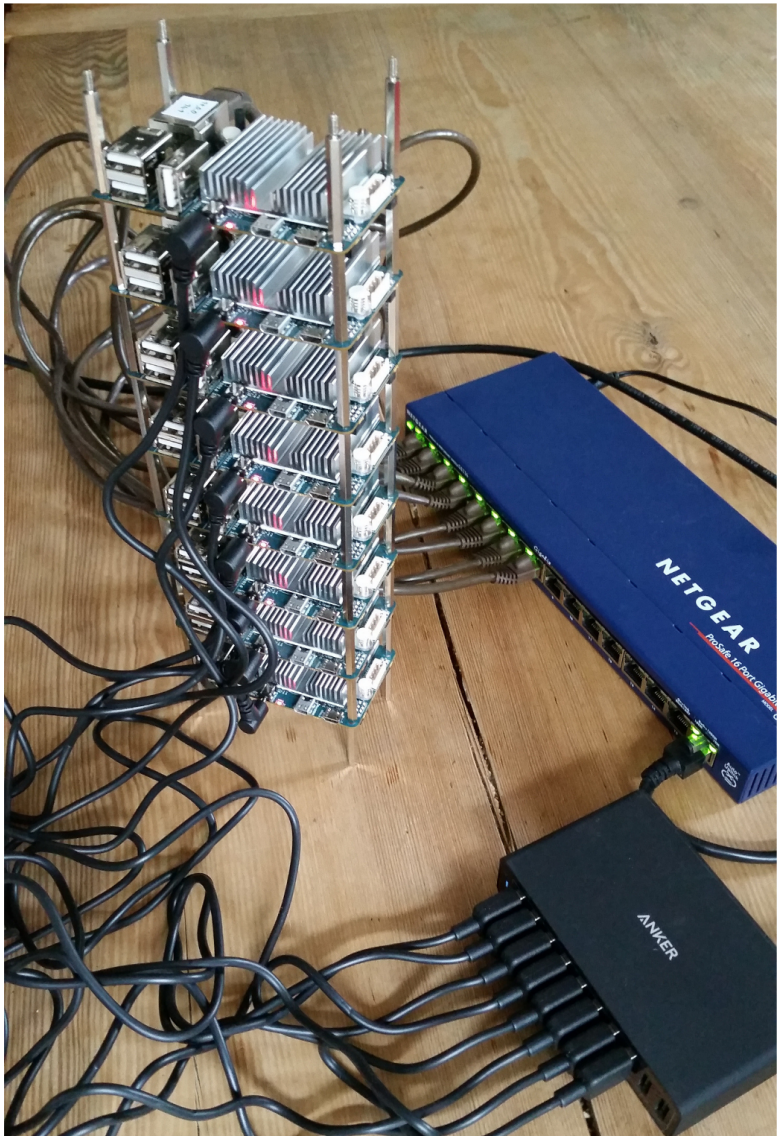


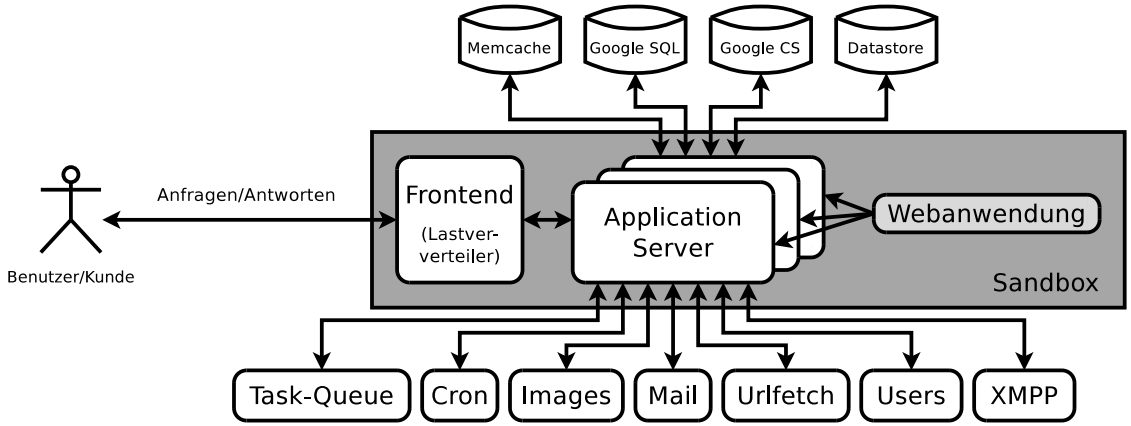
- Eine sinnvolle Einbindung ist in verschiedene LVs möglich
- Beispiele:
  - Cloud-Computing im HIS-Master
  - Distributed Systems
  - Realtime Systems
    - z.B. ein Programmierpraktikum mit den GPIO-Ports
  - Informatik Projekt im 5. Semester
    - Man könnte z.B. 3 bis 4 Gruppen a 4 Personen an einem Cluster arbeiten lassen

In der Praxis steht der Aufwand, mehrere mobile Cluster zu betreuen, zu warten und aktuell zu halten, in keinem Verhältnis zum Nutzen

Bessere Lösung: SSH-Zugänge zu Ressourcen in der FRA-UAS anbieten

Cloud-Plattformdienste für Webanwendungen (PaaS)



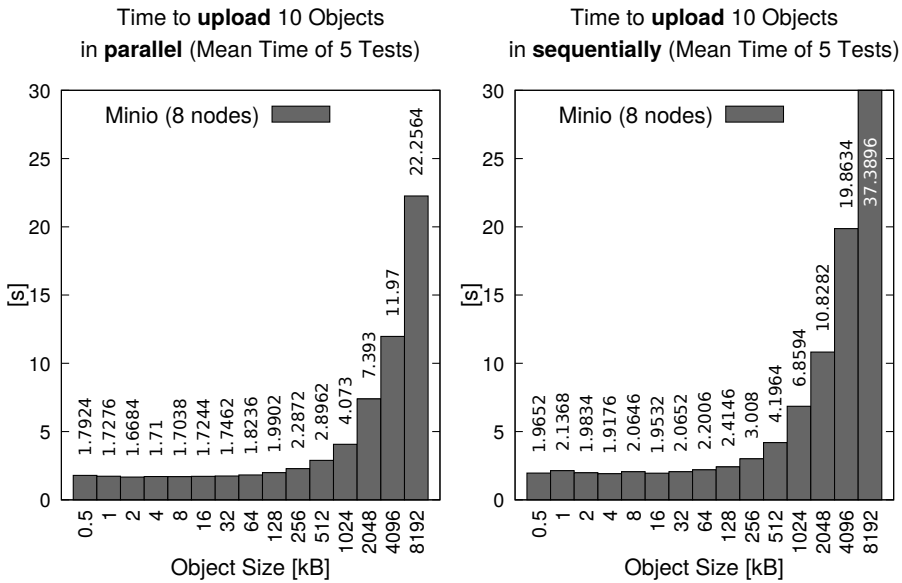


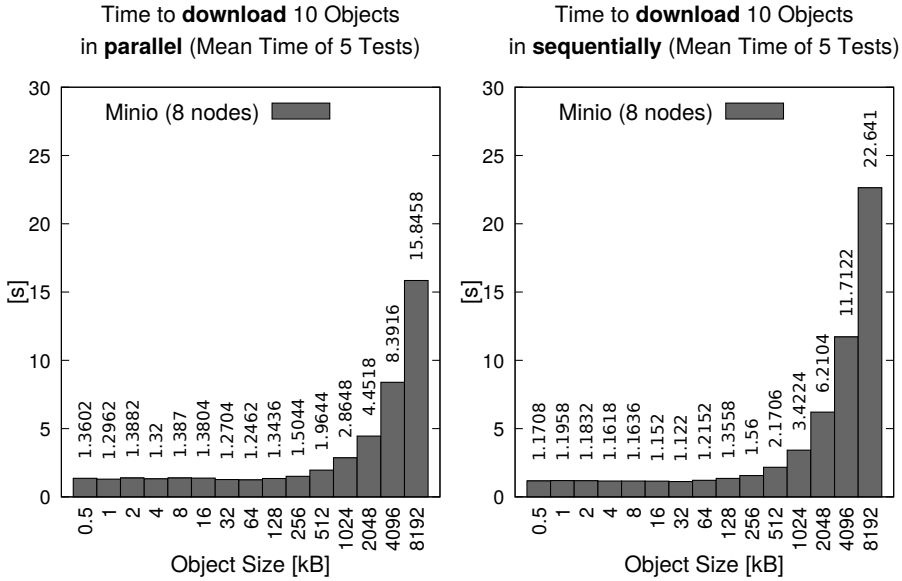
- Im Prinzip sind Einplatinencomputer für den Betrieb eines PaaS ausreichend
- In der Praxis lief nur ein Cluster mit 8 ODROID-U3 zufriedenstellend
  - Problem: Speicherbedarf der DB

AppScale als Alternative zu Googles App Engine. *Christian Baun*. iX 12/2016, S.72-75

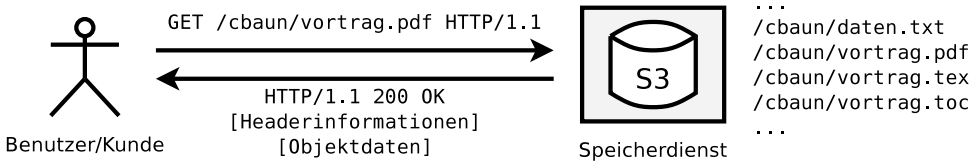
Lessons Learned From Implementing a Scalable PaaS Service by Using Single Board Computers. *Christian Baun*. International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.7, No.2, 2017, S.1-11. `http://aircconline.com/ijccsa/V7N2/7217ijccsa01.pdf`

Cloud-Infrastrukturdienste (IaaS)





- Objektbasierte Speicherdienste benötigen nur wenig CPU-Leistung
  $\Longrightarrow$ gut geeignet für den Betrieb auf Einplatinencomputern



- Ein neues Testwerkzeug wurde implementiert: `ossperf`

`http://github.com/christianbaun/ossperf`

Performance Aspects of Object-based Storage Services on Single Board Computers. *Christian Baun, Henry-Norbert Cocos, Rosa-Maria Spanou*. Open Journal of Cloud Computing (OJCC), Vol.4, Issue 1, 2017.
`https://www.ronpub.com/OJCC_2017v4i1n01_Baun.pdf`

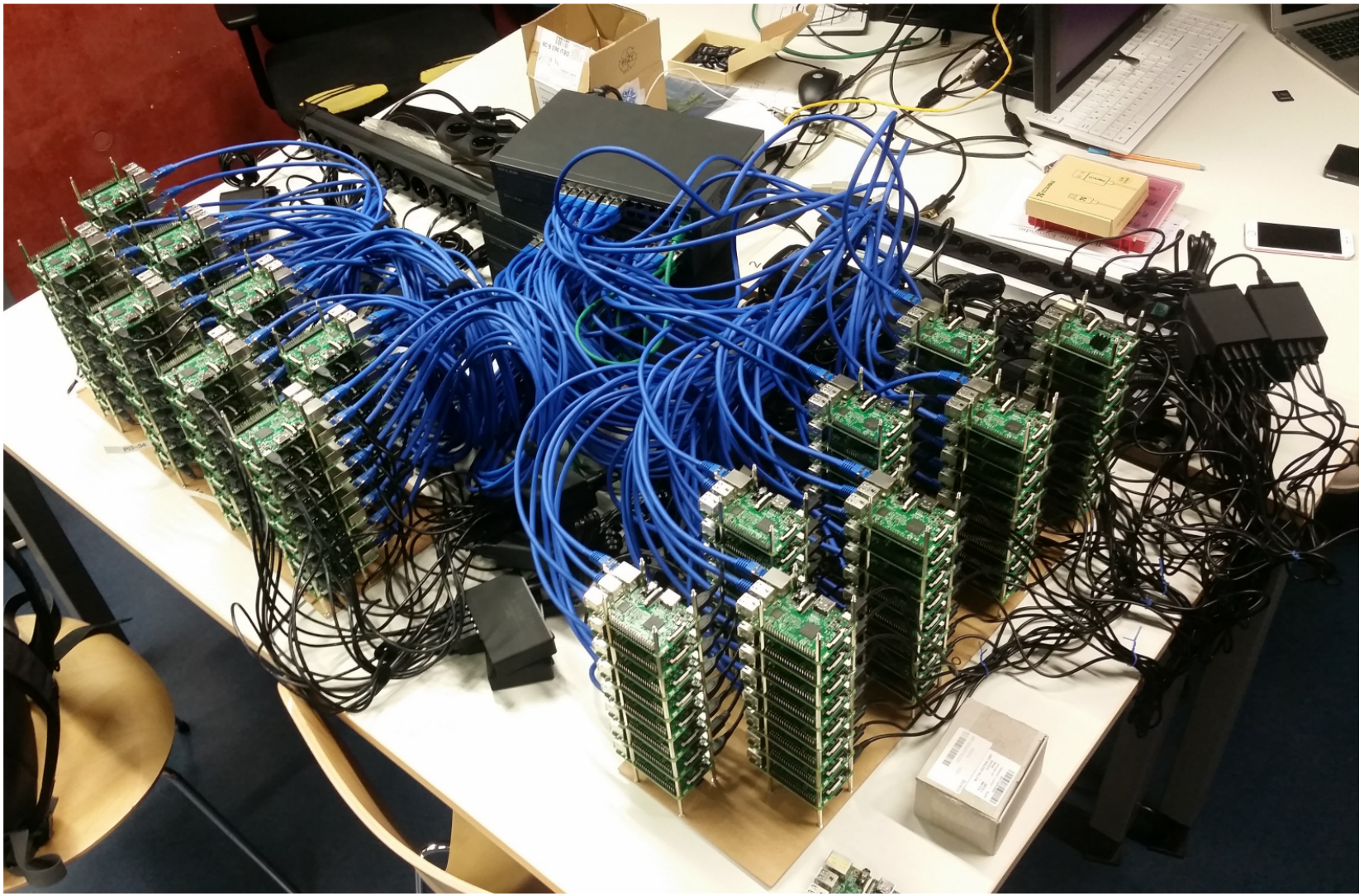
Testumgebung für verteilte Dateisysteme

- Problem: Es gibt nur wenige Anschlussmöglichkeiten für Speicher
  - USB 2.0, microSD, Ethernet (meist < 100 MBit/s)
- Idee: Datenzugriff durch verteilte Dateisysteme beschleunigen und gleichzeitig redundante Datenhaltung realisieren
- Erfolgreich installiert: GlusterFS, XtreemFS und NFS (zum Vergleich)
- Bislang erfolglos: Ceph, Lustre und OrangeFS/PVFS2

Eine systematische Untersuchung der Leistungsfähigkeit und die Publikation der Ergebnisse steht noch aus

```
Volume Name: volume32repl2
Type: Distributed-Replicate
Volume ID: f6d9ac9b-cbf2-4168-a197-272523e1478c
Status: Started
Number of Bricks: 16 x 2 = 32
Transport-type: tcp
Bricks:
Brick1: pi110:/mnt/gluster/brick3
Brick2: pi111:/mnt/gluster/brick3
Brick3: pi112:/mnt/gluster/brick3
Brick4: pi113:/mnt/gluster/brick3
Brick5: pi114:/mnt/gluster/brick3
Brick6: pi115:/mnt/gluster/brick3
Brick7: pi116:/mnt/gluster/brick3
Brick8: pi117:/mnt/gluster/brick3
Brick9: pi118:/mnt/gluster/brick3
Brick10: pi119:/mnt/gluster/brick3
Brick11: pi120:/mnt/gluster/brick3
Brick12: pi121:/mnt/gluster/brick3
Brick13: pi122:/mnt/gluster/brick3
Brick14: pi123:/mnt/gluster/brick3
Brick15: pi124:/mnt/gluster/brick3
...
Brick27: pi136:/mnt/gluster/brick3
Brick28: pi137:/mnt/gluster/brick3
Brick29: pi138:/mnt/gluster/brick3
Brick30: pi139:/mnt/gluster/brick3
Brick31: pi140:/mnt/gluster/brick3
Brick32: pi141:/mnt/gluster/brick3
```

Neuer Cluster: „Brain“mit 128 RPi 3 mit 512 CPU-Kernen



Maßgeblich am Aufbau beteiligt sind Henry-Norbert Cocos und Rosa-Maria Spanou

An der Optik müssen wir noch arbeiten. . . ;-)

Einige Erfahrungswerte (Energieversorgung)

- **Es gibt viele USB-Netzteile und USB-Kabel auf dem Markt**
  - Viele Netzteile entsprechen nicht den angegebenen Werten
  - Sind die USB-Kabel „zu billig“ (schlechte Schirmung) oder „zu lang“, kommt bei einigen Knoten zu wenig Strom an
- Gute Netzteile: **Anker** mit **60 W Leistung**

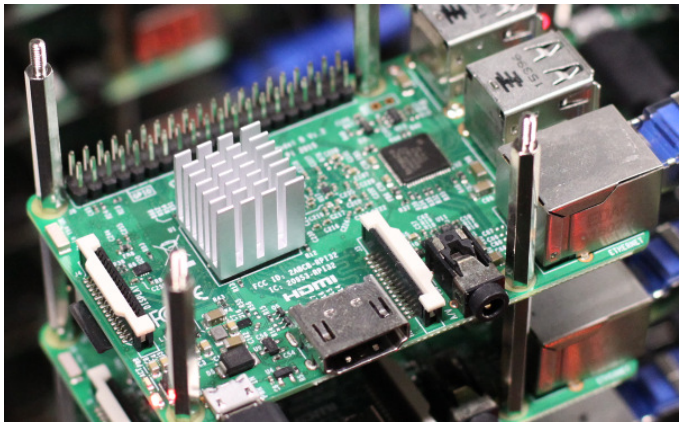
$$\frac{60\,\text{Watt}}{5\,\text{Volt}} = 12\,\text{Ampere}$$

- Alle 10 Ports belegt $\Longrightarrow$ 1,2 A pro Port
- Für RasPi 1, RasPi 2, BananaPi 1 und ODROID-U3 reicht das
- Für RasPi 3 reicht das nicht
- Ergebnis: Abstürze ohne ersichtlichen Grund in den Logdateien, zerstörte Dateisysteme und SD-Karten
  $\Longrightarrow$ 2 A pro Port und alles ist gut $\Longrightarrow$ 4 Ports pro Netzteil liegen brach

Einige Erfahrungswerte (Kühlung)

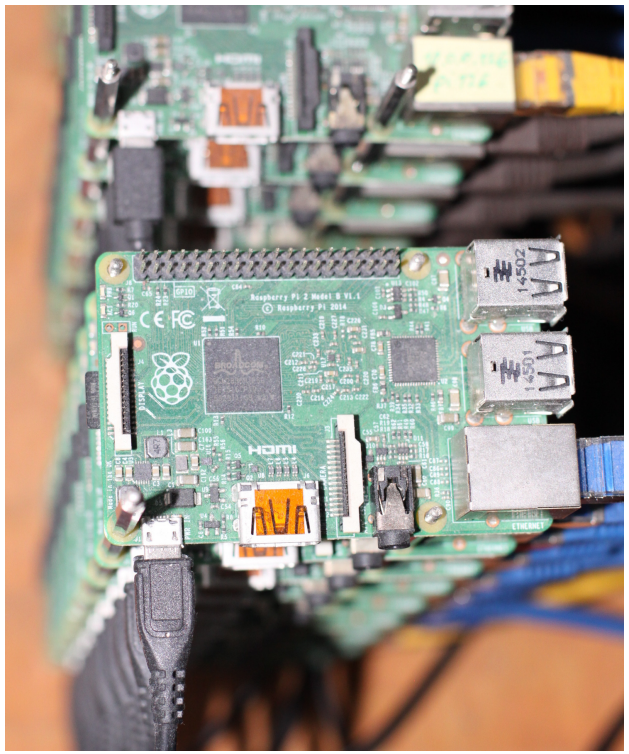
- **Die Knoten des Clusters sind passiv gekühlt**
  - Motivation: Keine beweglichen Teile $\Longrightarrow$ bessere Verfügbarkeit
  - Nachteil: Die Knoten werden unter Volllast $> 85°C$ heiß und stürzen ab
- Idee 1: **CPUs heruntertakten**
  - Bei 1000 MHz Taktrate bleibt die Temperatur knapp unter 85°C
    $\Longrightarrow$ Als Lösung ein bisschen unsexy
- Idee 2: **Kühlkörper kaufen**
  - Durchschnittl. Temp. im Ruhezustand:
    - 59°C ohne Kühlkörper
    - 56°C mit Kühlkörper
  - Durchschnittl. Temp. beim HPL-Test:
    - 82°C ohne Kühlkörper
    - 80°C mit Kühlkörper

  $\Longrightarrow$ Die Kühlkörper bringen fast nichts



Einige Erfahrungswerte (Positionierung der Knoten)

- Herausforderung: Wie bringt man die 32 oder 128 Knoten „unter“
  - **Einplatinencomputer haben meist an allen Seiten Anschlüsse**
    - Die Anschlüsse auf 3 Seiten müssen beim RasPi 2/3 frei bleiben



- Idee 1: Die Knoten auf ein Brett schrauben
  $\Longrightarrow$ unpraktische Lagerung und Handhabung
- Idee 2: Lego
  $\Longrightarrow$ sieht kindisch aus
- Idee 3: Fertige Gehäuse kaufen
  $\Longrightarrow$ teuer, verschlechtern die Luftzirkulation
- Idee 4: Fertige Cluster-Konstruktionen kaufen
  $\Longrightarrow$ viel zu teuer
- Lösung für den Moment: Die Knoten mit Abstandsbolzen (M2,5) zu Stapeln („Türmchen“) verschrauben
  - Die Stapel auf Bretter schrauben, damit sie nicht umfallen können

Einige Erfahrungswerte (Qualität der microSD-Karten)

- Die Hersteller der Karten werben mit tollen Schreibgeschwindigkeiten
  - Dummerweise sind das immer Angaben für sequentielles Schreiben
    - Im Multitasking-Betriebssystem kommt sequentielles Schreiben selten vor
  - Näher an der Praxis ist das Schreiben auf zufällige Speicherstellen

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`

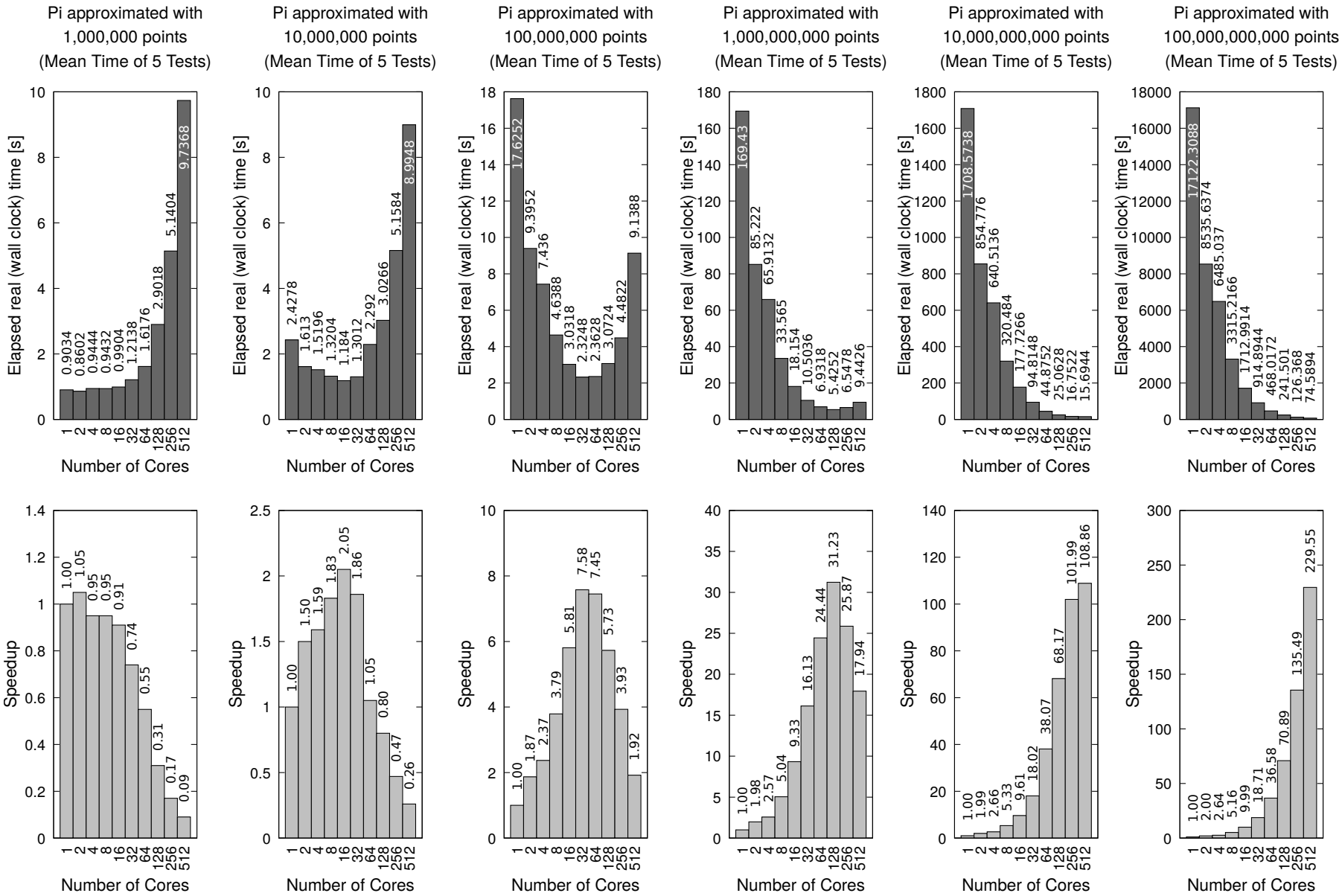
- Die Geschwindigkeit der Karten von Samsung und Verbatim ist gut
- Einige Karten sind offenbar für 4 kB-Random-Schreibzugriffe optimiert
  - Warum? Wie geht das? $\Longrightarrow$ Die Herstellerfirmen sagen es nicht
    `https://raspberrypi.stackexchange.com/questions/32884/why-does-the-sd-card-random-write-performance-for-record-si`
- Anekdote am Rande: Die Kingston-Karten starben wie die Fliegen...

Einige Details

Komponenten	Anschaffungskosten
129 RasPi3-Einplatinencomputer (128 Worker + 1 Master)	ca. 5.200 €
129 microSD-Karten (teilweise 16 GB, teilweise 32 GB)	ca. 1.800 €
129 USB-Kabel	ca. 160 €
136 Ethernet-Kabel	ca. 170 €
27 USB-Netzteile (teilweise 40 W, teilweise 60 W)	ca. 800 €
5 (ziemlich große) Steckdosenleisten	ca. 100 €
6 Ethernet-Switches (je 24 Anschlüsse)	ca. 570 €
129 Kühlkörper aus Aluminium	ca. 220 €
ca. 600 Abstandsbolzen (M2,5/30)	ca. 100 €
Summe:	ca. 9.120 €

- **Rechenleistung**
  - 107,1 Gflops (HPL 2.2, OpenBLAS 0.2.17 ARMv7, Kernel 4.4.50 v7+, N=112.000 MB, NB=192)
- **Stromverbrauch**
  - 345 W im Leerlauf (idle-Modus)
    - Davon benötigt alleine die Netzwerk-Infrastruktur ca. 165 W im Leerlauf!
  - ca. 430 W beim Überprüfen von Primzahlen mit `sysbench`
  - ca. 583 W beim Lösen eines linearen Gleichungssystems mit dem HPL
    - Dabei als Spitzenlast (zeitweise) bis zu 647 W

Mit 512 Prozessorkernen gehen sehr schöne Testreihen...



Gute Ideen gesucht!

- Gibt es Ideen für weitere sinnvolle Einsatzmöglichkeiten?

Im WS1718 kommt „Brain“ erstmals in der Cloud-Vorlesung (HIS-Master) zu Einsatz

Die Teilnehmer entwickeln in 3 Gruppen MPI-Anwendungen zum parallelen Sortieren, zur Primzahlbestimmung via Sieb des Eratosthenes und zur parallelen Matrizenmultiplikation

- Gibt es Interesse, die Ressourcen zu nutzen?

- Kennt jemand passende Fördermöglichkeiten?