

# INSTITUT FÜR INFORMATIK

DER LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN



Bachelorarbeit

## Skalierungsverhalten eines Raspberry Pi-Clusters unter der Workload ausgewählter HPC-Benchmarks

Judith Greif

Draft vom 22. März 2014



# INSTITUT FÜR INFORMATIK

DER LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN



# Bachelorarbeit

# Skalierungsverhalten eines Raspberry Pi-Clusters unter der Workload ausgewählter HPC-Benchmarks

Judith Greif

Aufgabensteller: Prof. Dr. Dieter Kranzlmüller  
Betreuer: MNM-Team-Betreuer Dr. Nils gentschen Felde  
                  MNM-Team-Betreuer Christian Straube  
Abgabetermin: 31. März 2014



Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 31. März 2014

.....  
*(Unterschrift der Kandidatin)*



## **Abstract**

Hier steht eine kurze Zusammenfassung der Arbeit. Sie darf auf gar keinen Fall länger als eine Seite sein, ca. eine drittel bis eine halbe Seite ist optimal.



# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
1.1 Hintergrund . . . . .	1
1.2 Fragestellung . . . . .	1
1.3 Herausforderungen . . . . .	2
1.3.1 Anpassung der Benchmarks . . . . .	2
1.3.2 Testumgebung . . . . .	2
1.3.3 Ausführung der Benchmarks auf einem RPi-Cluster . . . . .	2
1.4 Vorgehensweise und Struktur . . . . .	3
<b>2 Grundlagen und Begriffsbildung</b>	<b>5</b>
2.1 Definition: <i>Benchmarking</i> . . . . .	5
2.2 Definition: <i>Leistung</i> . . . . .	6
2.3 Definition: <i>Performance</i> . . . . .	6
2.4 Benchmarks . . . . .	7
2.4.1 Linpack . . . . .	7
2.4.2 Whetstone . . . . .	8
2.4.3 STREAM . . . . .	9
2.5 Spezifikation des Raspberry Pi Modell B . . . . .	9
2.5.1 Physischer Aufbau . . . . .	10
2.5.2 Betriebssystem . . . . .	11
2.6 Raspberry Pi-Cluster als Testumfeld . . . . .	11
2.6.1 Physischer Aufbau . . . . .	11
2.6.2 Betriebssystem und Filesystem . . . . .	12
2.6.3 Zugriff auf die Cluster-Komponenten . . . . .	12
<b>3 Versuchsaufbau und -ablauf</b>	<b>15</b>
3.1 Zielsetzung . . . . .	15
3.2 Aufbau und Art der Messung . . . . .	15
3.2.1 Versuchsaufbau Ri-Einzelrechner . . . . .	16
3.2.2 Versuchsaufbau Bramble . . . . .	16
3.2.3 Skalierung der Messung auf 1 – n RPi-Nodes . . . . .	17
3.2.4 Automatisierte Durchführung der Messung für 1 – n RPi-Nodes . . . . .	17
3.2.5 Zeitsynchronisierung der RPi-Nodes . . . . .	18
3.2.6 Einlesen der Messwerte in die Datenbank . . . . .	19
3.2.7 Ausgabe und Aufbereitung der Messwerte . . . . .	19
3.3 Ergebnisse . . . . .	19
3.3.1 Linpack . . . . .	19
3.3.2 Whetstone . . . . .	19

## *Inhaltsverzeichnis*

<b>4 Interpretation</b>	<b>21</b>
4.1 Linpack . . . . .	21
4.2 Whetstone . . . . .	21
4.3 Einordnung in TOP500 und Green500 . . . . .	21
<b>5 Zusammenfassung und Ausblick</b>	<b>23</b>
<b>6 Anhang</b>	<b>25</b>
6.1 Ergebnisse für Linpack 100 und Whetstone auf dem RPi-Einzelrechner . . . . .	25
6.2 Shellskript zur Ausführung der Benchmarks auf n RPi-Nodes . . . . .	27
<b>Abbildungsverzeichnis</b>	<b>29</b>
<b>Literaturverzeichnis</b>	<b>31</b>

# 1 Einleitung

Seit Beginn seiner Entwicklung im Jahr 2009 booms der Minicomputer Raspberry Pi<sup>1</sup>: Er erhielt z.B. den Designpreis INDEX: award 2013<sup>2</sup>, wurde als Innovation des Jahres bei den T3 Gadget Awards 2012 ausgezeichnet<sup>3</sup> und zum Product of the Year 2012 des Linux Journal gewählt. Im Februar dieses Jahres war das Modell B über 2,5 Millionen Mal verkauft worden. Was sind die Gründe für den Erfolg des RPi?

## 1.1 Hintergrund

Der RPi weist eine hohe Energieeffizienz und ein sehr gutes Kosten-Nutzen-Verhältnis auf, ist flexibel in Anpassung und Verwendung und bietet einen niederschwülligen Zugang. Das macht ihn z.B. für Projekte im pädagogischen Umfeld interessant, in denen Kinder und Jugendliche an die Grundlagen der Programmierung herangeführt werden.

Privatpersonen setzen den RPi z.B. als mobilen Video-Player, Überwachungskamera oder zur Steuerung von Lichtschaltern und Haushaltsgeräten ein. Im wissenschaftlichen Rahmen werden immer häufiger mehrere RPis zu einem Cluster verschaltet, um Rechenoperationen auf mehrere Knoten zu verteilen. Daraus ergeben sich Fragestellungen wie: Welche Rechtleistung erzielt ein RPi im Vergleich zu einem durchschnittlichen Desktop-Rechner oder einem Notebook? Welche CPU-Performance lässt sich mit einem RPi-Cluster im Verhältnis zu einem früheren oder aktuellen Supercomputer erreichen?

Die Leistung von Rechnern, seien es Großrechner, Desktop-Rechner oder Minicomputer, wird häufig durch Benchmarks ermittelt. Das ermöglicht die Vergleichbarkeit der Testergebnisse unterschiedlicher Systeme. Bekannte und erprobte Benchmark-Suites sind z.B. Linpack und Whetstone, mit denen seit den 70er Jahren die Performance von Supercomputern ermittelt wird. Für Einzelrechner mit Linux-Systemen gibt es z.B. die Phoronix Test Suite oder UnixBench, um die Performance der einzelnen Komponenten wie CPU, GPU und RAM zu evaluieren.

## 1.2 Fragestellung

Vor diesem Hintergrund stellt sich die Frage: Wie verhält sich ein RPi im Vergleich zu einem Supercomputer, wenn Benchmarks aus dem HPC-Bereich darauf ausgeführt werden? Noch bedeutsamer ist die Untersuchung eines RPi-Clusters: Wie ein Supercomputer implementiert er ein verteiltes System mit RPi-Einzelrechnern als Rechner-Nodes. Welche CPU-Performance lässt sich damit erzielen und wie verhält sich der Cluster bei Hinzunahme von Ressourcen, d.h. RPi-Rechenkernen? Im Zentrum dieser Arbeit stehen daher

---

<sup>1</sup>Im Folgenden als *RPi* bezeichnet.

<sup>2</sup>Vgl. <http://designtoimprovelife.dk/category/s15-award2013/index-award-winners-2013/>.

<sup>3</sup>Vgl. <http://www.t3.com/news/t3-gadget-awards-2012-award-winners/>.

## *1 Einleitung*

CPU-Performance und Skalierungsverhalten eines RPi-Clusters unter den Testbedingungen ausgewählter HPC-Benchmarks. Dazu soll zunächst die die Performance eines RPi-Einzelrechners unter den ausgewählten Benchmarks ermittelt werden. Anschließend wird versucht, dieselben Benchmarks auf einem RPi-Cluster lauffähig zu machen und zu evaluieren.

### **1.3 Herausforderungen**

Die Anpassung von HPC-Benchmarks an einen RPi und einen RPi-Cluster stellen besondere Anforderungen an den Versuchsaufbau: Es müssen geeignete Implementierungen der Benchmarks für den RPi-Einzelrechner und den RPi-Cluster verwendet werden und die Vergleichbarkeit der Ergebnisse muss durch eine geeignete Testumgebung sicher gestellt werden.

#### **1.3.1 Anpassung der Benchmarks**

Bei der Auswahl der Implementierungen ist eine Unterscheidung zwischen RPi-Einzelrechner und dem Cluster notwendig, vor allem hinsichtlich des Betriebssystems. Ist eine passende Implementierung gefunden, muss diese möglicherweise auf einem anderen Rechner kompiliert und übertragen werden, da die CPU des RPi vergleichsweise schwach gegenüber einem Desktop-PC ist und die Kompilierung sehr lange dauern könnte. Für die verteilte Ausführung der Benchmark auf dem Cluster ist die Installation eines Message Passing Interfaces (MPI) wie OpenMP oder MPICH notwendig.

#### **1.3.2 Testumgebung**

Die Testumgebung muss an die Testsituation angepasst werden. Es muss sicher gestellt sein, dass keine weiteren Prozesse im Hintergrund ablaufen, die Rechenleistung von den Benchmark-Programmen abziehen und die Untersuchungsergebnisse verfälschen können. Auch die Systemzeiten der Rechner-Nodes muss beachtet und gegebenenfalls synchronisiert werden, da ein RPi keine eingebaute Echtzeituhr besitzt.

#### **1.3.3 Ausführung der Benchmarks auf einem RPi-Cluster**

Der verwendete Cluster wurde nicht mit dem primären Fokus auf Benchmarking entwickelt. Bevor er dafür eingesetzt werden kann, muss ein grundsätzlicher Überblick über seine Architektur, die Zugriffsmöglichkeiten auf seine Komponenten, insbesondere das Filesystems, geschaffen werden. Dann kann ermittelt werden, wie und ob sich die gewählten Benchmarks auf der bestehenden Struktur ausführen lassen. Hierbei ist mit Hindernissen zu rechnen.

## 1.4 Vorgehensweise und Struktur

Um den Bezugsrahmen zu verdeutlichen, werden in Kapitel 2 zunächst grundlegende Definitionen geklärt. Insbesondere werden die Spezifikationen des RPi (vgl. Kap. 2.5) und des Clusters erläutert (vgl. Kap. 2.6). Anschließend werden die ausgewählten Benchmarks vorgestellt (vgl. Kap. 2.4). Versuchsaufbau und -ablauf werden mit Blick auf Auswahl und Anpassung der RPi-spezifischen Parameter im folgenden Kapitel erläutert (vgl. Kap. 3). Schließlich werden die Messergebnisse auf dem RPi-Einzelrechner und dem Cluster dargestellt (vgl. Kap. 3.3) und interpretiert (vgl. Kap. 4). Den Abschluss bilden eine Zusammenfassung der Untersuchungsergebnisse und ein Ausblick (vgl. Kap. 5).



## 2 Grundlagen und Begriffsbildung

Es gibt zahlreiche Benchmarks, die bereits an den RPi angepasst und auf diesem ausgeführt wurden. Dabei steht oft die Performance verschiedener Betriebssysteme (z.B. Fedora vs. Debian) oder einzelner Hardware-Komponenten im Vordergrund. Zu diesem Zweck werden hauptsächlich Linux-spezifische Benchmarks verwendet wie Sysbench CPU Benchmark (CPU), PyBench (Python-Implementierung), Apache Benchmark (Webserver), OpenSSL (CPU) oder ioquake3 (GPU).

Bei näherem Hinschauen erscheint es schwierig, sich einen Überblick über die existierenden Benchmarks zu verschaffen. Vieles, was von den Anwendern als „Benchmark“ bezeichnet wird, stellt sich als selbst geschriebene Routine heraus, mit der z.B. die Performance der Grafikkarte getestet werden soll. Eine solche Routine kann für einen Vergleich mit HPC-Rechnern herangezogen werden. Im Folgenden wird daher auf grundlegende Begriffe eingegangen, die für die nachfolgende Untersuchung von Bedeutung sind. Anschließend werden die verwendeten Benchmarks (vgl. 2.4) und ihre Anpassung an den RPi erläutert (vgl. 3.2).

### 2.1 Definition: Benchmarking

Unter *Benchmarking* oder „Maßstäbe vergleichen“ versteht man im Allgemeinen die vergleichende Analyse von Ergebnissen oder Prozessen mit einem festgelegten Bezugswert oder Vergleichsprozess. *Computer-Benchmarks*, die hier von Bedeutung sind, dienen dem Vergleich der Rechenleistung von Computer-Systemen, wozu in der Regel Software verwendet wird<sup>1</sup>. Das *Computer Lexikon 2012* kennt folgende Definition:

Mit einem Benchmark-Programm werden Hardwarekomponenten meist auf Geschwindigkeit getestet, wie z.B. die CPU, das Mainboard, die Festplatte (Schreib-Lese-Geschwindigkeit), die Grafikkarte (Frames/s) usw. Verschiedene Benchmark-Programme liefern oft unterschiedliche Ergebnisse, so dass ein direkter Vergleich zwischen den erreichten Werten kaum aussagekräftig ist [Pre11].

Hieran wird deutlich, dass die Aussagekraft von Benchmarks eng mit der jeweiligen Testumgebung und der Zielsetzung des Benchmarks zusammenhängt. Zwei Benchmarks, die die CPU-Performance evaluieren, liefern möglicherweise unterschiedliche Ergebnisse, weil unterschiedliche Parameter oder sogar Messgrößen zu Grunde liegen. Das ist insbesondere bei der Auswahl der Implementierungen und Gestaltung der Testumgebung zu berücksichtigen (vgl. Kap. 2.5, 2.6 und 3.2).

In dieser Arbeit soll die Leistung von Hardware-Komponenten eines oder mehrerer parallel arbeitender Rechenkerne mit standardisierten Verfahren ermittelt werden. Rechenberg

---

<sup>1</sup> „A simple method of measuring performance is by means of a benchmark program. [...] The intention is that by running it upon a new type of machine one may learn something of the performance the machine would have if it ran the original programs [CW76].“

## 2 Grundlagen und Begriffsbildung

bezeichnet „*Analyse*, *Auswahl* und *Konfiguration* von Gesamtsystemen aus Hardware und Software [Rec06]“ als eine Hauptaufgabe der Leistungsbewertung:

[F]ür diese Aufgaben, die etwa im Zuge einer Rechnerbeschaffung anfallen, wurde in Form von standardisierten Meßprogrammen und -methoden (*benchmarking*) eine solide Basis geschaffen [Rec06].

Daher wird hier mit *Benchmarking* kurz das **Standardisieren von Arbeit** bezeichnet.

### 2.2 Definition: Leistung

Die physikalische Größe *Leistung* ist als *Energie pro Zeit* definiert. In der Informatik versteht man darunter meist die *Rechenleistung*. Wichtige Aspekte bei der Leistungsbewertung eines Rechnersystems sind unter anderem

[...] die *Leistungskenngrößen* oder *-maßzahlen* (*performance metrics*), die für das vorliegende Rechnersystem und die Ziele der Leistungsanalyse relevant sind [Rec06].

Weitere Aspekte der Leistungsbewertung sind das zu untersuchende System, die Workload und die Methode zur Leistungsermittlung<sup>2</sup>. Übertragen auf die vorliegende Untersuchung bedeutet dies: Das zu untersuchende System besteht aus einem RPi-Einzelrechner und einem RPi-Cluster (vgl. Kap. 2.5 bzw. 2.6). Methoden zur Leistungsbewertung sind HPC-Benchmarks (vgl. Kap. 2.4.1, 2.4.2 und 2.4.3). Die Workload ist die Ausführung dieser Programme. Die hier verwendete Definition greift den ersten Aspekt auf: Mit *Leistung* ist die **Time to completion** gemeint, d.h. die Zeit, die ein Prozess oder ein Programm(teil) bis zum erfolgreichen Abschluss benötigt. Bei Rechenberg wird dies als „Programmlaufzeit“ [Rec06] bezeichnet<sup>3</sup>.

### 2.3 Definition: Performance

Für den Begriff *Performance* im Zusammenhang mit Rechnern existieren verschiedene Definitionen, häufig gleichbedeutend mit der *Rechenleistung* oder nicht klar davon abgegrenzt. Erklärungen wie „Zeitverhalten von Programmen und Geräten“ oder „Leistungsfähigkeit eines Computersystems“ scheinen der Fragestellung nicht ganz gerecht zu werden:

The performance of a computer is a complicated issue, a function of many inter-related quantities. These quantities include the application, the algorithm, the size of a problem, the high-level language, the implementation, the human level of effort used to optimize the program, the compiler's ability to optimize, the age of the compiler, the operating system, the architecture of the computer and the hardware characteristics [DLP03].

---

<sup>2</sup>Vgl. [Rec06].

<sup>3</sup>Andere Klassen von Kenngrößen, die neben der Zeit zur Leistungsbewertung herangezogen werden, sind *Durchsatz* und *Auslastung*. Abweichende zeitliche Kenngrößen sind z.B. *Bedienzeit* (*Service time*) oder *TTR* (*Time to Response*) (vgl. ebd.).

Das *Informatik-Handbuch* liefert eine genauere Definition:

Quantitative Leistungsanalysen (*performance analyses*) ermitteln Leistungskenngrößen von Rechenanlagen. [...] Leistungsbewertung kann sich auf Teilschaltungen, Komponenten (wie Prozessor, Speichersystem oder periphere Geräte), gesamte Rechnerverbünde beziehen [Rec06].

In dieser Arbeit werden Performance eines RPis und eines RPi-Clusters ermittelt und der Leistungsfähigkeit eines Supercomputers gegenübergestellt. Der Maßstab für die Performance ist dabei die *Time to Completion* eines gegebenen Benchmark-Programms<sup>4</sup>. *Performance* soll daher hier als ***Arbeit pro Zeit*** verstanden werden.

## 2.4 Benchmarks

Aus der Fülle an existierenden Benchmarks wurden drei für die Untersuchung ausgewählt. Kriterien waren dabei Erprobtheit, Verlässlichkeit und Angemessenheit für das zu untersuchende System<sup>5</sup>. Außerdem muss gewährleistet sein, dass der Benchmark überhaupt auf das gewählte System anwendbar und an dieses anpassbar ist<sup>6</sup>.

Mit dem Ziel, das Skalierungsverhalten eines RPi-Clusters zu untersuchen, wurden drei etablierte HPC-Benchmarks ausgewählt, die sowohl auf Cluster-Architekturen als auch Einzelrechnern zur Anwendung kommen: Linpack, Whetstone und STREAM. Sie werden im Folgenden vorgestellt.

### 2.4.1 Linpack

Grundsätzlich muss man zwischen der Linpack-Library und dem Linpack-Benchmark unterscheiden: Die Library ist eine numerische Programmbibliothek zum Lösen von linearen Gleichungssystemen. Sie wurde inzwischen von anderen Bibliotheken abgelöst, galt aber lange als Standard<sup>7</sup>. Der Linpack-Benchmark basiert auf zwei Programmen dieser Bibliothek<sup>8</sup>. Er wurde hauptsächlich von Jack Dongarra, einem der Autoren der Linpack-Bibliothek, entwickelt und wird seit den 1970er Jahren zur Klassifizierung von Rechnern verwendet<sup>9</sup>. Seit 1993 werden die leistungsfähigsten Supercomputer der Welt durch die Top500-Rankings

---

<sup>4</sup>Vgl. hierzu Curnow in Bezug auf Whetstone: „We are not claiming [to reflect] the overall performance of a given system. On the contrary, we believe that no single number ever can. It does, however, reflect the performance of a dedicated machine for solving a dense system of linear equations [CW76]“.

<sup>5</sup>Vgl. Weickers Kriterien für die Auswahl eines Benchmarks: „the [...] best benchmark (1) is written in a high-level language, making it portable across different machines, (2) is representative for some kind of programming style (for example, systems programming, numerical programming, or commercial programming), (3) can be measured easily, and (4) has wide distribution [Wei90].“

<sup>6</sup>Aus diesem Grund mussten z.B. SHOC und LLNL IOR ausscheiden. SHOC ist nicht lauffähig auf dem RPi, da Open CL nicht unterstützt wird. LLNL IOR wiederum benötigt ein POSIX-, MPIIO- oder HDF5-Interface. Zum letztendlichen Ausschluss von Whetstone vgl. Kap. 5.

<sup>7</sup>Für die genaue Spezifikation von Linpack vgl. [DLP03]. Der Sourcecode für Linpack 1000 in Fortran findet sich unter <http://www.netlib.org/benchmark/1000d>.

<sup>8</sup>„Linpack was designed out of a real, purposeful program that is now used as a benchmark[Wei90].“

<sup>9</sup>„Over the years additional performance data was added [...] and today the collection includes over 1300 different computer systems [DLP03].“

## 2 Grundlagen und Begriffsbildung

ermittelt. Hierzu dient die Variante HPLinpack, auch  $N \times N$  Linpack oder High Parallel Computing genannt<sup>10</sup>.

### Funktionsweise

Linpack ist ein Benchmark zur Ermittlung der CPU-Performance. Dazu werden Fließpunkt-Operationen auf einer Matrix<sup>11</sup>, die intern in eine lineare Darstellung umgewandelt wird, durchgeführt und das Ergebnis in *FLOPS* („Floating Point Operations Per Second“) ausgegeben<sup>12</sup>. Dabei erreicht der derzeit leistungsfähigste Supercomputer, *Tianhe-2 (Milky Way-2)*, z.B. einen *Rmax*-Wert („Maximal LINPACK performance achieved“) von 33862.7 TFLOPS. SuperMUC, der jüngst auf Platz 10 der Top500 gerankt wurde, erreicht einen *Rmax*-Wert von 2897.0 TFLOPS.

Beim Vergleich Linpack-Ergebnissen ist die Größe des Arrays bzw. der Matrix von Bedeutung, da eine Änderung wegen geringerer Datenlokalität zu starken Abweichungen der Ergebnisse führen kann<sup>13</sup>. Wichtig ist das beim Vergleich der Ergebnisdaten verschiedener Rechnerarchitekturen. I.d.R. werden daher o.g. standardisierte Größen verwendet, auch wenn der Quellcode eine Veränderung der Matrixgröße zulässt.

### Linpack auf dem Raspberry Pi

Bereits kurz nach Verkaufsstart des RPi wurden Implementierungen von Linpack für den RPi bereit gestellt und Ergebnisse von Testläufen im Internet veröffentlicht<sup>14</sup>. Somit liegen bereits Vergleichswerte für einen RPi-Einzelrechner vor. Auch Implementierungen von Linpack für verteilte Systeme unabhängig von den Top500-Rankings sind im Umlauf<sup>15</sup>. Zu prüfen wird sein, wie diese für den RPi-Cluster nutzbar gemacht werden können.

#### 2.4.2 Whetstone

Auch Whetstone ist als Benchmark im HPC-Bereich und zur Klassifizierung von Einzelrechnern seit vielen Jahren etabliert<sup>16</sup>. Er wurde 1976 von Roy Longbottom u.A. entwickelt und gilt als erstes Programm, das jemals explizit für das Benchmarking industrieller Standards designt wurde<sup>17</sup>. Wie Linpack misst er die CPU-Performance.

<sup>10</sup> „Over recent years, the LINPACK Benchmark has evolved from a simple listing for one matrix problem to an expanded benchmark describing the performance at three levels of problem size on several hundred computers. The benchmark today is used by scientists worldwide to evaluate computer performance, particularly for innovative advanced-architecture machines [DLP03].“ Eine detaillierte Beschreibung des Sourcecode von HPLinpack findet sich ebd..

<sup>11</sup> LINPACK 100:  $100 \times 100$ -Matrix; LINPACK 1000:  $1000 \times 1000$ -Matrix; HPLinpack:  $n \times n$ -Matrix (vgl. ebd.).

<sup>12</sup> Genau genommen ist es so, dass zwar hauptsächlich, aber nicht ausschließlich Floating Point-Operationen ausgeführt werden. Der Anteil der Nicht-Fließpunkt-Operationen wie Berechnungen auf Integer-Werten werden bei der Auswertung entweder vernachlässigt oder in die Fließpunkt-Operationen integriert (vgl. [Wei90]).

<sup>13</sup> Vgl. ebd..

<sup>14</sup> Die hier verwendete Implementierung in C für den RPi-Einzelrechner findet sich unter [http://www.roylongbottom.org.uk/Raspberry\\_Pi\\_Benchmarks.zip](http://www.roylongbottom.org.uk/Raspberry_Pi_Benchmarks.zip).

<sup>15</sup> Vgl. <http://www.netlib.org/benchmark/hpl/>.

<sup>16</sup> „[...] the most common ‘stone age’ benchmarks (CPU/memory/compiler benchmarks only) [are] in particular the Whetstone, Dhrystone, and Linpack benchmarks. These are the benchmarks whose results are most often cited in manufacturers’ publications and in the trade press [Wei90].“

<sup>17</sup> Vgl. ebd..

### Funktionsweise

Die Funktionsweise von Whetstone ähnelt Linpack. Allerdings werden nicht nur Fließkomma-Berechnungen ausgeführt, sondern auch mathematische Funktionen, Integer-Arithmetik, If-Statements etc. kommen zur Anwendung<sup>18</sup>. Jedes dieser als genuin betrachteten Module wird in eine For-Schleife eingebettet und viele Male hintereinander ausgeführt. Die Ausgabe der Berechnungsergebnisse ist Teil des Programms<sup>19</sup>. Ein Framework bildet den äußeren Programmrahmen und steuert die Ausgaben der einzelnen Module. Die Ergebnisse wurden zunächst in *Whetstone Instructions per Second* gemessen, heutige Implementierungen liefern Ergebnisse in MFLOPS<sup>20</sup>.

### Whetstone auf dem Raspberry Pi

Whetstone erschien wie für den RPi gemacht, da er sich als Standard für Mini-Computer etabliert hat<sup>21</sup>. Er wurde vom Entwickler selbst für den RPi adaptiert und auf diesem getestet<sup>22</sup>. Auch hier liegen also bereits Vergleichswerte für einen RPi-Einzelrechner vor.

Wie für die meisten etablierten HPC-Benchmarks existieren Implementierungen in zahlreichen höheren Programmiersprachen wie C, C++, Fortran und Java<sup>23</sup>. Auch hier wird aus nahe liegenden Gründen auf eine Implementierung in C für den RPi-Einzelrechner zurückgegriffen<sup>24</sup>. Ob eine lauffähige Implementierung für die Cluster-Architektur existiert, wird sich zeigen müssen.

### 2.4.3 STREAM

#### Funktionsweise

#### STREAM auf dem Raspberry Pi

## 2.5 Spezifikation des Raspberry Pi Modell B

Was dem Nutzer zuerst auffällt, ist sicherlich die Größe des RPi. Er wird manchmal als „Scheckkarten-Computer“ bezeichnet und überschreitet diese Maße mit 85.60 mm × 53.98

---

<sup>18</sup>Ein Grund hierfür ist, dass die ursprüngliche Programmversion nicht komplex genug war, um ein durchschnittliches FORTRAN-Programm zu simulieren. Gegenüber der ursprünglichen Implementierung in ALGOL 60 war ein damals üblicher FORTRAN-Compiler in der Lage, Zwischenergebnisse in schnellen Registern abzuspeichern und darauf zurückzugreifen, sodass die gemessene CPU-Leistung deutlich höher ausfiel als erwartet (vgl. [CW76]).

<sup>19</sup>Allerdings: „This output is only required to ensure that the calculations are logically necessary; it is not intended to represent the output from a typical program“ (ebd.).

<sup>20</sup>Für eine detaillierte Darstellung der Entwicklung und Implementierung von Whetstone vgl. ebd.. Hier findet sich auch der ursprüngliche Sourcecode in ALGOL 60.

<sup>21</sup>Ein Grund hierfür ist, dass Whetstone in den 70er Jahren für Maschinen mit signifikant geringerer Rechenleistung entwickelt worden war. Auch damals gingen die Entwickler von notwendigen Anpassungen für neue Speicherhierarchien aus: „When more is known about the characteristics of programs running on these multi-level store machines it may be possible to produce a typical program for particular types of machine. [...] Despite these limitations the program described should be of some value, particularly in relation to smaller machines (ebd.).“

<sup>22</sup>Vgl. <http://www.roylongbottom.org.uk/Raspberry%20Pi%20Benchmarks.htm>.

<sup>23</sup>Vgl. z.B. <http://freespace.virgin.net/roy.longbottom/>.

<sup>24</sup>Der hier verwendete Sourcecode findet sich ebenfalls unter [http://www.roylongbottom.org.uk/Raspberry\\_Pi\\_Benchmarks.zip](http://www.roylongbottom.org.uk/Raspberry_Pi_Benchmarks.zip).

## 2 Grundlagen und Begriffsbildung

mm × 17 mm tatsächlich nur geringfügig. Auf dieser Platine sind alle Komponenten verbaut, die den RPi zu einem voll funktionsfähigen Rechner machen:

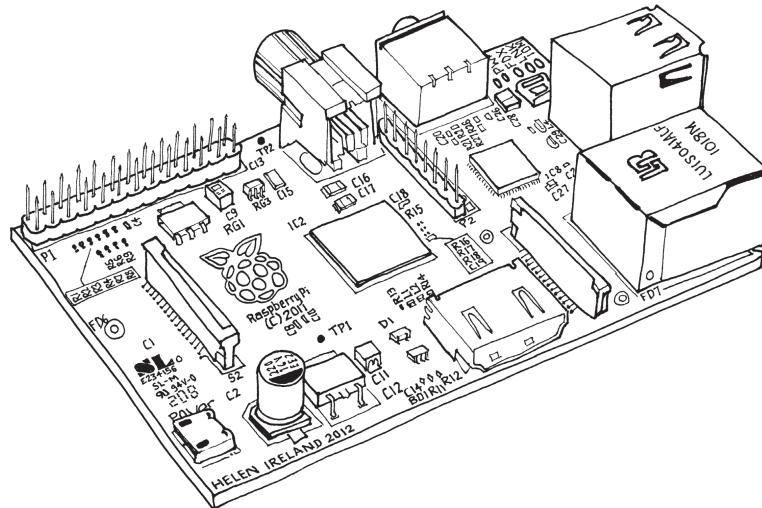


Abbildung 2.1: Ein Raspberry Pi Modell B (Quelle: [BCH<sup>+</sup>12]).

### 2.5.1 Physischer Aufbau

#### SoC, CPU, GPU und RAM

Der RPi ist mit einem Broadcom BCM2835 als SoC ausgestattet. Es enthält CPU (ein ARM1176JZFS-Prozessor) und GPU (ein Broadcom VideoCore IV-Koprozessor)<sup>25</sup>. Auffällig ist, dass die CPU des RPi (ein ARM-Chip, wie er häufig in Mobiltelefonen verbaut ist), relativ schwach ist im Vergleich zur GPU, die Full HD-Auflösung und das hardwarebeschleunigte Rendern verschiedener Videoformate unterstützt. Das Modell B verfügt über 512 MB SDRAM<sup>26</sup>.

#### Schnittstellen

Der RPi verfügt über einen HDMI-Ausgang, einen Cinch-Ausgang („RCA Jack“) und einen analogen Tonausgang. Er hat zwei USB 2.0-Schnittstellen und eine Ethernet-Schnittstelle. Ein Steckplatz ist für eine SD-Karte vorgesehen, auf der der nicht flüchtige Speicher liegt. Die Stromversorgung erfolgt über einen Mini-USB-Eingang. Zum Betrieb werden mindestens 700 mA/5 V benötigt<sup>27</sup>.

<sup>25</sup>Vgl. [Lan12].

<sup>26</sup>Gegenüber dem Modell A mit 256 MB (vgl. [Lan12]). Dieser kann nicht erweitert werden, doch die Aufteilung zwischen CPU und GPU ist variabel. Da der RPi kein BIOS hat, muss dazu die Datei `/boot/start.elf` manipuliert werden (vgl. [Pow12]).

<sup>27</sup>Vgl. ebd.. Der Vollständigkeit halber sei auch das Vorhandensein einer GPIO mit sechs Anschlüssen und jeweils 26 Pins erwähnt. Daüber können, vergleichbar einem Arduino-Board, z.B. LEDs, Sensoren und Displays angesteuert werden.

Für den Versuchsaufbau sind vor allem Ethernetanschluss, Mini-USB und SD-Karte von Bedeutung. Beim RPi-Einzelrechner erfolgt der Netzanschluss über einen DSL-Router. Die Stromversorgung geschieht über die Steckdose, während beim Cluster Stromversorgung und Netzanschluss zentral über den Server des RPi-Clusters erfolgen. RPi-Einzelrechner und jeder RPi-Node verfügen über eine eigene 4 GB Class 10 SD-Karte (auf die Cluster-Architektur wird in Kap. 2.6 genauer eingegangen).

## 2.5.2 Betriebssystem

Für den RPi existieren mehrere Varianten bzw. Derivate verbreiteter Betriebssysteme. Am verbreitetsten sind Linux/Unix-basierte Systeme wie FreeBSD und NetBSD (BSD-Varianten), Raspbian (Debian-Variante), Pidora (Fedora-Variante) oder eine Variante von Arch Linux<sup>28</sup>. Als Betriebssystem für den RPi-Einzelrechner wurde Raspbian gewählt, die offizielle Distribution der Raspberry Pi Foundation<sup>29</sup>. Auf den Einsatz von NOOBS<sup>30</sup> oder anderer zusätzlicher Bootloader wurde verzichtet<sup>31</sup>.

# 2.6 Raspberry Pi-Cluster als Testumfeld

In den letzten Monaten lässt sich die Tendenz beobachten, eine größere Anzahl von Raspberry Pis zu einem Cluster zu koppeln<sup>32</sup>. Die hier verwendete Architektur wird im Folgenden wird umrissen<sup>33</sup>.

## 2.6.1 Physischer Aufbau

Der RPi-Cluster besteht aus 20 RPi Modell B-Einzelrechnern (DNS-Namen `pi01 – pi20`), die jeweils über ein Ethernet-Kabel mit dem zentralen x86-Server (im Folgenden mit seinem DNS-Namen `careme` bezeichnet) verbunden sind. Er besteht hauptsächlich aus einem Mini-ITX-Mainboard und einer Gigabit Ethernet-Netzwerkkarte. Alle Komponenten befinden sich in einem modifizierten Tower-Metallgehäuse, das auf die Seite gedreht und oben offen gelassen wurde. Darin sind außerdem ein 24 Port Gigabit-Switch, die Stromversorgung des Servers und der RPi-Nodes sowie die Kühlung integriert<sup>34</sup>.

---

<sup>28</sup>Vgl. [Pow12]. Daneben gibt es Implementierungen von RISC OS und Plan 9.

<sup>29</sup>Vgl. <http://www.raspberrypi.org/faqs>. Das verwendete Image findet sich unter [http://downloads.raspberrypi.org/raspbian\\_latest](http://downloads.raspberrypi.org/raspbian_latest).

<sup>30</sup>Die Raspberry Pi-Foundation empfiehlt besonders für Einsteiger die *New Out Of Box Software* und bietet entsprechend präparierte SD-Karten zum Kauf an (vgl. <http://www.raspberrypi.org/archives/tag/noobs>).

<sup>31</sup>In diesem Fall fungiert die ausführbare Datei `start.elf`, die nach einer ausführbaren Datei `kernel.img` sucht und diese lädt, als Bootloader (vgl. [Kli13]).

<sup>32</sup>Weitere Projekte werden in [CCB<sup>+</sup>13], [Kie13], [Bal12] und [Ou13] dargestellt.

<sup>33</sup>Für eine detaillierte Darstellung vgl. [Kli13].

<sup>34</sup>Ein solches System relativ kostengünstiger Rechner mit einem BSD- oder Linux-Betriebssystem, die über IP kommunizieren, wird im Allgemeinen als *Beowulf* bezeichnet (vgl. [Kie13] und [Kli13]). Häufig dient es dem Ersatz oder der Simulation eines Supercomputers. Im Zusammenhang mit RPis ist auch der Begriff *Bramble* gebräuchlich, der im Folgenden verwendet wird (vgl. [www.raspberrypi.org/archives/tag/bramble](http://www.raspberrypi.org/archives/tag/bramble)).

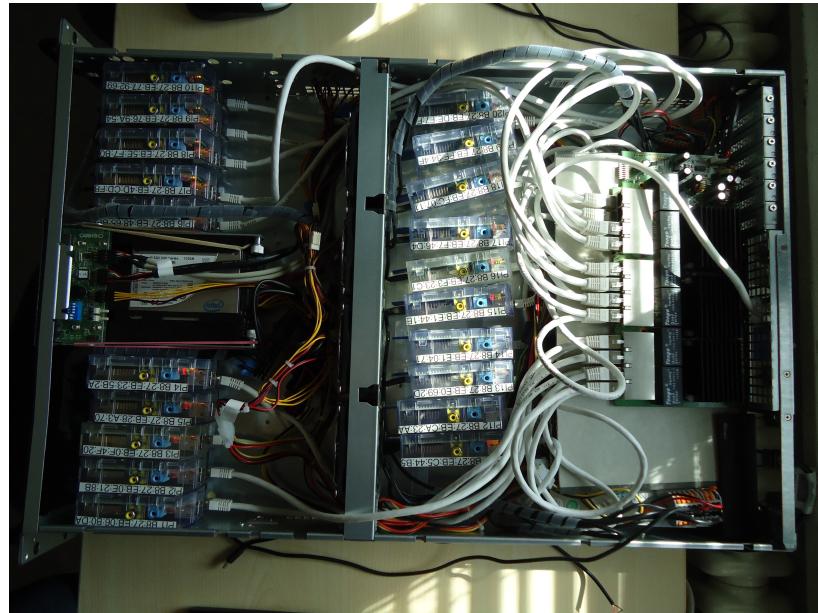


Abbildung 2.2: Der hier verwendete Bramble.

### 2.6.2 Betriebssystem und Filesystem

Auf den RPi-Nodes läuft das o.g. Betriebssystem Raspbian, auf `careme` eine Standard-Debian-Version.

Ein Charakteristikum eines Beowulf-Clusters bzw. Bramble ist, dass es im Gegensatz zu einem Supercomputer kein Shared Memory-Interface und keine Cache-Kohärenz gibt. Die Kommunikation zwischen den einzelnen Komponenten via Ethernet ist zudem relativ langsam. Es stellt sich daher die Frage nach dem Filesystem und der Synchronisierung der einzelnen Nodes.

Um das Filesystem der einzelnen Nodes mit dem NFS des Servers zu synchronisieren, wurde hier eine Read-Only-Schicht für die Slaves (Nodes) und eine Read-Write-Schicht für den Master (Server) verwendet<sup>35</sup>. Um trotzdem den gemeinsamen Zugriff von Server und RPi-Nodes auf ein Speichermedium zu ermöglichen, wurde ein gesharedes Verzeichnis `/srv` bzw. `/srv/nfs-share` eingerichtet.

### 2.6.3 Zugriff auf die Cluster-Komponenten

Welche Möglichkeiten gibt es nun, verteilte Anwendungen auf dem Bramble auszuführen bzw. überhaupt die einzelnen Komponenten anzusprechen? Bei einem RPi-Einzelrechner nutzt man normalerweise einen USB-Port und den HDMI-Ausgang als stdin und stdout. Eine andere Möglichkeit besteht darin, den RPi über den Ethernet-Port mit einem Router zu verbinden und über SSH darauf zuzugreifen<sup>36</sup>.

---

<sup>35</sup>Vgl. ebd..

<sup>36</sup>Diese Variante wurde für das hier verwendeten Testgerät gewählt auf Grund von Problemen beim Anschluss älterer Monitore ohne HDMI-Eingang (vgl. z.B. <http://www.raspberrypi.org/forum/viewtopic.php?f=91&t=34061>).

Der Zugriff auf den Bramble erfolgt aus dem internen Netzwerk über IP und SSH. Die einzelnen RPi-Nodes werden von `careme` aus über SSH adressiert (private Adressen 10.0.0.2 bis 10.0.0.21). Es gibt auf jedem RPi-Node den Raspbian-Standard-User `pi`, der hier nicht verändert wurde<sup>37</sup>. Um z.B. auf das gesharte Verzeichnis `/srv` zuzugreifen, sind `root`-Berechtigungen erforderlich<sup>38</sup>.

Beim hier verwendeten Versuchsaufbau wurde auf `careme` mit eingeschränkten Rechten als User `rpi-user` gearbeitet, auf den einzelnen RPis als `root`. Das bedeutet, dass alle Änderungen im gesharten Verzeichnis notwendigerweise von einem der RPi-Nodes aus vorgenommen werden müssen. Dazu wurde vorab RPi-Node `pi03` als Berechnungs-Node definiert, von dem aus alle Skripte ausgeführt, Programme installiert werden etc.. Da die Skripte zur Ausführung der Benchmarks häufig das Herunterfahren einzelner RPi-Nodes vorsehen (vgl. Kap. 6), wurde `pi03` wurden die Benchmarks grundsätzlich auf maximal  $n=19$  RPi-Nodes ausgeführt (vgl. Kap. 3). Zur verteilten Berechnung von Anwendungen auf mehreren CPUs steht auf dem Bramble die MPI-Implementierung MPICH in der Version 3.0.4 zur Verfügung. Um die CPUs bzw. RPi-Nodes und die Anzahl der darauf auszuführenden Prozesse zu spezifizieren, muss ein entsprechendes Machinefile erstellt und im gesharten Verzeichnis abgelegt werden, das der Ausführung von MPICH mit `mpiexec` als Parameter übergeben wird (vgl. auch Kap. 3 und 5).

---

<sup>37</sup>Auf dem RPi-Einzelrechner wurden aus Sicherheitsgründen Username und Passwort für den Standard-User geändert.

<sup>38</sup>Auf Schwierigkeiten bei dieser Einteilung und Troubleshoots wird in Kap. 5 kurz eingegangen.



# 3 Versuchsaufbau und -ablauf

Das folgende Kapitel beschreibt die Ziele der Messung, Versuchsaufbau und -durchführung. Ziel der Messung ist die Untersuchung des Skalierungsverhaltens der ausgewählten HPC-Benchmarks auf dem Bramble. Die Ergebnisse sollen unter Vergleichswerte des Bramble gegenüber einem Supercomputer liefern. Dazu wird der Bramble an Hand der Messwerte in die TOP500- und Green 500-Listen eingeordnet (vgl. Kap. 4.3).

## 3.1 Zielsetzung

Von der Messung werden detaillierte Erkenntnisse über das Skalierungsverhalten von HPLinpack und Whetstone auf dem Bramble mit 20 RPi-Nodes erwartet. Alle Messungen werden auf 1 – n RPi-Nodes des Bramble durchgeführt mit  $n = 20$ . Die Messungen werden zwei Mal durchgeführt: Zuerst mit Netzwerkanschluss der gerade nicht beteiligten RPi-Nodes an den Bramble-Server, danach ohne.

Als Output-Parameter werden der Energieverbrauch des Bramble und die jeweils Benchmark-spezifischen Load-Generatoren betrachtet. Der Stromverbrauch wird durch ein Strommessgerät ermittelt, das an den Bramble-Server angeschlossen wird. Die Benchmark-spezifischen Load-Generatoren sind natürlich an den vom jeweiligen Programm vorgegebenen Parametern: HPLinpack liefert 4 Maßzahlen für die CPU-Performance eines Rechners/Rechnersystems.  $R_{max}$  entspricht der Performance in GFLOPS bei Ausführung des größten Problems,  $N_{max}$  bezeichnet das grösste ausgeführte Problem,  $N_{1/2}$  die Problemgröße bei halber  $R_{max}$ -Rate und  $R_{peak}$  die theoretische Peak-Performance in GFLOPS. Whetstone liefert drei Kennzahlen für die CPU-Performance: MWIPS (Millions of Whetstone Instructions per Second), davon abgeleitet MFLOPS für die Ausführung der Fließpunkt-Operationen und MIPS (Millions of Instructions per Second) für die Integer-Tests.

## 3.2 Aufbau und Art der Messung

Für den Versuchsaufbau stellen sich zwei grundlegende Fragen: Welcher Art ist die Messung und welche Voraussetzungen müssen hierfür gelten?

Die durchzuführende Messung hat zwei Ausgabeparameter: Energieverbrauch und die jeweils Benchmark-spezifischen Ausgabewerte. Für den Versuchsaufbau sind daher folgende Aspekte von Bedeutung: Zeitsynchronisation der RPi-Nodes, Skalierung der Messung auf 1 – n RPi-Nodes, automatisierte Durchführung der Messung für 1 – n RPi-Nodes, Einlesen der Messwerte in eine geeignete Datenstruktur, Ausgabe und Aufbereitung der Messwerte.

Um sich mit den grundlegenden Funktionalitäten vertraut zu machen, wurden die ausgewählten Benchmarks zunächst auf einem RPi-Einzelrechner ausgeführt. Dabei wurden schnell die grundlegenden Unterschiede zum Bramble deutlich, vor allem hinsichtlich des Filesystems. Dies wirkt sich auf die Auswahl der Benchmark-Implementierungen aus: Z.B.

### 3 Versuchsaufbau und -ablauf

ist die Ausführung von HPLinpack auf einem Einzelrechner nicht möglich. Auch der Versuchsaufbau und die Ausführung der Benchmarks sind hiervon betroffen: Zeitsynchronisation und Skalierung der Messung auf 1 – n Nodes sind bei einem RPi-Einzelrechner nicht notwendig, wohl aber bei einem Cluster.

Da der Schwerpunkt auf dem Skalierungsverhalten der Benchmarks auf dem Bramble liegt, wird der Versuchsaufbau für den RPi-Einzelrechner im Folgenden nur kurz skizziert. Die Ergebnisse von Linpack 100 und Whetstone auf dem RPi-Einzelrechner sind im Anhang dokumentiert (vgl. Kap. 6.1).

#### 3.2.1 Versuchsaufbau Ri-Einzelrechner

Für die meisten RPi-User stellen sich nach Auswahl, Installation und Konfiguration des Betriebssystems zwei Fragen nach Swap Space und Übertakten. Beides liegt auf der Hand, da der RPi Modell B nur über einen Arbeitsspeicher von 512 MB verfügt. Die CPU-Leistung ist mit 700 MHz ebenfalls nicht eher niedrig.

Im Praxisbetrieb zahlreicher User zeigte sich, dass ein Übertakten der CPU auf bis auf 1 GHz anscheinend gefahrlos möglich ist. In der Konsequenz entschloss sich die Raspberry Pi Foundation, den Verlust der Garantieleistung bei Übertaktung aufzuheben. Beim hier verwendeten Testgerät wurde dennoch auf das Übertakten verzichtet, um die Vergleichbarkeit mit anderen RPi-Einzelrechnern und dem Bramble sicherzustellen<sup>1</sup>.

Das gewählte Betriebssystem Raspbian verwendet standardmäßig eine Swap-Datei (`/var/swap`), die den Adressraum des Arbeitsspeichers bei Überlast erweitert. Hierbei gibt es zwei Probleme: Erstens könnten ständige Schreibzugriffe auf Dauer die SD-Karte beschädigen<sup>2</sup>. Außerdem sind Schreibzugriffe auf die SD-Karte sehr langsam, was die Performanz des Systems bei hoher Arbeitsspeicherlast beeinträchtigen kann. Aus diesem Grund wurde entschieden, zunächst auf die Nutzung der Swap-Datei zu verzichten<sup>3</sup>.

#### 3.2.2 Versuchsaufbau Bramble

Das folgende Komponentendiagramm (3.1) zeigt den Versuchsaufbau pro Messreihe, der als *ExperimentSuite* bezeichnet wird. Das Diagramm orientiert sich an den Aspekten, an die systemkritisch für die Versuchsdurchführung sind: *Stromversorgung* und *Netzwerkanschluss* des Bramble, *Messung des Stromverbrauchs* mit dem Strommessgerät, *Loggen des gemessenen Stromverbrauchs* und *Upload der Messergebnisse* in eine MySQL-Datenbank auf dem Bramble-Server.

---

<sup>1</sup>Es erscheint wenig zielführend, die Komponente zu tunen, deren Performance man mittels Benchmarking evaluieren möchte. Für eine spätere Untersuchung ist dies jedoch nicht ausgeschlossen. Z.B. wäre es interessant zu ermitteln, ob man den relativ hohen Stromverbrauch des Bramble bei Niedriglast (vgl. [Kli13]) durch Untertakten der einzelnen CPUs senken kann.

<sup>2</sup>Vgl. [Pow12].

<sup>3</sup>Die Swap-Datei kann mit dem Befehl `sudo update-rc.d dphys-swapfile remove` deaktiviert werden. Eine weitere Möglichkeit des Swapping ist die Verwendung von zRAM, sodass ein Teil des Arbeitsspeichers komprimiert und als Swap Space genutzt wird (vgl. [Pow12]). Die Allokierung einer auf Unix-Systemen üblicherweise genutzten Swap-Partition ist nicht sinnvoll, da auch hierdurch die Lebensdauer der SD-Karte durch häufige Schreibzugriffe verkürzt wird.

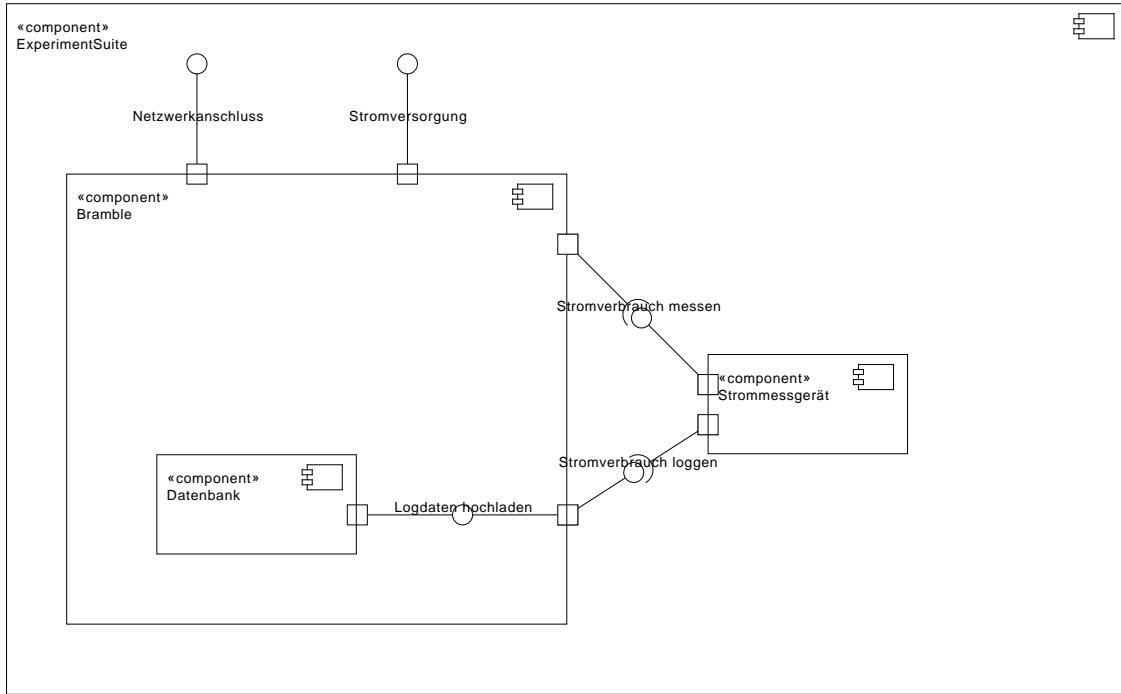


Abbildung 3.1: Komponentendiagramm des Versuchsaufbaus.

### 3.2.3 Skalierung der Messung auf 1 – n RPi-Nodes

Die Benchmark-generierte Workload und die Messung dieser Ergebnisse findet im Inneren der Komponente *Bramble* statt. Auf dem Bramble-Server wird ein Prozess angestoßen, der den jeweiligen Benchmark auf 1 – 20 RPi-Nodes ausführt, zunächst ohne und dann mit Stromanschluss der gerade nicht beteiligten RPi-Nodes. Das folgende Aktivitätsdiagramm zeigt, welche Schritte aus Benutzersicht für die Durchführung einer ExperimentSuite erforderlich sind. Dabei wird mir  $n = 20$  RPi-Nodes begonnen und einmal über alle Nodes von 20 – 1 iteriert. Danach wird die zweite Messung ebenfalls als Iteration über die RPi-Nodes 20 – 1 durchgeführt, wobei nach jedem Iterationsschritt der RPi-Node abgeschaltet wird. Dieser Aufbau ist in einem Aktivitätsdiagramm dargestellt (vgl. 3.2).

### 3.2.4 Automatisierte Durchführung der Messung für 1 – n RPi-Nodes

Die automatisierte Ausführung der Benchmarks erfolgt mit einem Shellskript auf dem Ausführungs-Node `pi03` (vgl. 6.2). Es ist im gescharfen Verzeichnis der RPi-Nodes `/srv` abgelegt und enthält unter anderem folgende Schritte:

1. Lösche altes Machinefile.
2. Erstelle ein neues Machinefile mit allen RPi-Nodes von 1 – 20 in `/srv/libraries/etc/mpich-3.0.4-shared/`.
3. Mounte das gesharte Verzeichnis `/srv` auf allen RPi-Nodes.

### 3 Versuchsaufbau und -ablauf

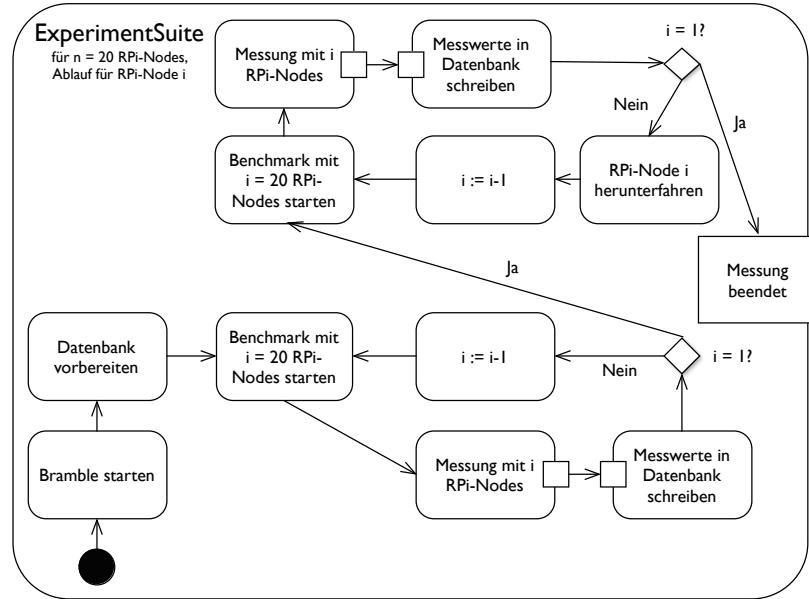


Abbildung 3.2: Aktivitätsdiagramm der ExperimentSuite.

4. Benutzereingabe für den auszuführenden Benchmark, das Arbeitsverzeichnis und ein optionales Parameterfile.
5. Erstelle zwei Ausgabedateien.
6. Für alle RPi-Nodes von 20 – 1: Führe den Benchmark auf n Nodes aus und logge die Ausgabe in die Ausgabedatei.
7. Wiederhole Schritt 6 und schalte RPi-Node n nach dem Loggen der Ausgabe aus.

#### 3.2.5 Zeitsynchronisierung der RPi-Nodes

Ein wichtiger Aspekt bei der parallelen Ausführung eines Programms ist die Zeitsynchronisation. Alle angestoßenen Berechnungen müssen auf allen beteiligten Knoten zum exakt selben Zeitpunkt starten, um verlässliche Messergebnisse zu liefern. Auf einem RPi ist das kritisch, da er keine Systemuhr hat. Auf dem Bramble wird die Zeitsynchronisation der einzelnen RPi-Nodes durch einen OpenNTP-Server realisiert, der auf dem Bramble-Server installiert wurde. Die einzelnen RPi-Nodes synchronisieren sich gegen diesen Server<sup>4</sup>. Die Zeitsynchronisation der RPi-Nodes ist somit auch für die verteilte Ausführung der Benchmarks gewährleistet.

---

<sup>4</sup>Vgl. [Kli13].

### 3.2.6 Einlesen der Messwerte in die Datenbank

### 3.2.7 Ausgabe und Aufbereitung der Messwerte

## 3.3 Ergebnisse

Auch hier liegt der Schwerpunkt auf den Ergebnissen der Messungen von HPLinpack und Whetstone auf dem Bramble. Der RPi-Einzelrechner erreichte für Linpack 100 41.31 MFLOPS. Für Whetstone wurden 255.154 MWIPS erreicht<sup>5</sup>.

### 3.3.1 Linpack

### 3.3.2 Whetstone

---

<sup>5</sup>Die Ergebnisdateien für beide Benchmarks finden sich im Anhang unter 6.1. Die vom Autor der verwendeten Implementierung erzielten Ergebnisse auf einem RPi-Einzelrechner sind unter <http://www.roylongbottom.org.uk/Raspberry%20Pi%20Benchmarks.htm#anchor4> zu finden.



## **4 Interpretation**

**4.1 Linpack**

**4.2 Whetstone**

**4.3 Einordnung in TOP500 und Green500**



## **5 Zusammenfassung und Ausblick**

Zum Vergleich: Der erste Supercomputer, der mit der Linpack-Benchmarksuite getestet wurde, war Cray 1 im Jahr 1987. Er kostete 7 Mio. US-Dollar und hatte eine Energiezufuhr von 115 Kilowatt. Ein Raspberry Pi mit allen notwendigen Komponenten kostet nur ca. 70 US-Dollar, benötigt 5 Watt Stromzufuhr und ist viereinhalb Mal schneller als Cray 1.



# 6 Anhang

## 6.1 Ergebnisse für Linpack 100 und Whetstone auf dem RPi-Einzelrechner

```
#####
Unrolled Double Precision Linpack Benchmark - Linux Version in 'C/C++'

Optimisation Opt 3 32 Bit

norm resid      resid      machep      x[0]-1      x[n-1]-1
 1.7    7.41628980e-14  2.22044605e-16 -1.49880108e-14 -1.89848137e-14

Times are reported for matrices of order          100
1 pass times for array with leading dimension of 201

      dgefa      dgesl      total      Mflops      unit      ratio
 0.01613    0.00057    0.01669     41.14     0.0486     0.2981

Calculating matgen overhead
      10 times   0.01 seconds
      100 times   0.14 seconds
      200 times   0.28 seconds
      400 times   0.57 seconds
      800 times   1.13 seconds
Overhead for 1 matgen   0.00141 seconds

Calculating matgen/dgefa passes for 1 seconds
      10 times   0.17 seconds
      20 times   0.35 seconds
      40 times   0.70 seconds
      80 times   1.40 seconds
Passes used      57

Times for array with leading dimension of 201

      dgefa      dgesl      total      Mflops      unit      ratio
 0.01609    0.00054    0.01663     41.29     0.0484     0.2970
 0.01603    0.00054    0.01658     41.43     0.0483     0.2960
 0.01610    0.00054    0.01664     41.25     0.0485     0.2972
 0.01609    0.00054    0.01663     41.29     0.0484     0.2970
```

## 6 Anhang

0.01603	0.00061	0.01663	41.28	0.0484	0.2970
Average			41.31		

Calculating matgen2 overhead  
Overhead for 1 matgen 0.00137 seconds

Times for array with leading dimension of 200

dgefa	dgesl	total	Mflops	unit	ratio
0.01447	0.00054	0.01502	45.73	0.0437	0.2682
0.01437	0.00051	0.01489	46.13	0.0434	0.2658
0.01447	0.00051	0.01498	45.84	0.0436	0.2675
0.01445	0.00051	0.01496	45.89	0.0436	0.2672
0.01441	0.00051	0.01492	46.02	0.0435	0.2665
Average			45.92		

#####

From File /proc/cpuinfo  
Processor : ARMv6-compatible processor rev 7 (v6l)  
BogoMIPS : 697.95  
Features : swp half thumb fastmult vfp edsp java tls  
CPU implementer : 0x41  
CPU architecture: 7  
CPU variant : 0x0  
CPU part : 0xb76  
CPU revision : 7  
  
Hardware : BCM2708  
Revision : 000f  
Serial : 00000000e98379f1

From File /proc/version  
Linux version 3.6.11+ (dc4@dc4-arm-01) (gcc version 4.7.2 20120731 (prerelease) (crosstool-NG linaro-1.13.1+bzr2458 - Linaro GCC 2012.08) )  
#538 PREEMPT Fri Aug 30 20:42:08 BST 2013

Unrolled Double Precision 41.31 Mflops

#####

#####  
Single Precision C Whetstone Benchmark Opt 3 32 Bit, Sat Nov 30 15:22:14 2013

Calibrate

## 6.2 Shellskript zur Ausführung der Benchmarks auf n RPi-Nodes

0.04 Seconds	1	Passes (x 100)
0.19 Seconds	5	Passes (x 100)
0.96 Seconds	25	Passes (x 100)
4.80 Seconds	125	Passes (x 100)

Use 260 passes (x 100)

```
From File /proc/cpuinfo
Processor : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS  : 697.95
Features   : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xb76
CPU revision   : 7

Hardware     : BCM2708
Revision     : 000f
Serial       : 00000000e98379f1
```

```
From File /proc/version
Linux version 3.6.11+ (dc4@dc4-arm-01) (gcc version 4.7.2 20120731 (prerelease)
(crosstool-NG linaro-1.13.1+bzr2458 - Linaro GCC 2012.08) )
#538 PREEMPT Fri Aug 30 20:42:08 BST 2013
```

### Single Precision C/C++ Whetstone Benchmark

Loop content	Result	MFLOPS	MOPS	Seconds
N1 floating point	-1.12475013732910156	97.643		0.051
N2 floating point	-1.12274742126464844	100.883		0.346
N3 if then else	1.0000000000000000		690.831	0.039
N4 fixed point	12.0000000000000000		423.573	0.193
N5 sin,cos etc.	0.49911010265350342		5.050	4.284
N6 floating point	0.99999982118606567	86.081		1.629
N7 assignments	3.0000000000000000		498.602	0.096
N8 exp,sqrt etc.	0.75110864639282227		2.724	3.551
MWIPS		255.154		10.190

## 6.2 Shellskript zur Ausführung der Benchmarks auf n RPi-Nodes



# **Abbildungsverzeichnis**

2.1 Ein Raspberry Pi Modell B (Quelle: [BCH <sup>+</sup> 12]). . . . .	10
2.2 Der hier verwendete Bramble. . . . .	12
3.1 Komponentendiagramm des Versuchsaufbaus. . . . .	17
3.2 Aktivitätsdiagramm der ExperimentSuite. . . . .	18



# Literaturverzeichnis

- [Bal12] BALAKRISHNAN, Nikilesh: *Building and benchmarking a low power ARM cluster.* 2012. – <http://www.epcc.ed.ac.uk/sites/default/files/Dissertations/2011-2012/Submission-1126390.pdf>
- [BCH<sup>+</sup>12] BEALE, Clive ; CROSTON, Ben ; HAGUE, Andrew ; HASTINGS, Graham ; KÖLLING, Michael ; LOCKWOOD, Brian ; OLDFNOW, Adrian: *The Raspberry Pi Education Manual*, Dezember 2012. – [http://downloads.raspberrypi.org/Raspberry\\_Pi\\_Education\\_Manual.pdf](http://downloads.raspberrypi.org/Raspberry_Pi_Education_Manual.pdf)
- [CCB<sup>+</sup>13] COX, Simon ; COX, James ; BOARDMAN, Richard ; JOHNSTON, Steven ; SCOTT, Mark ; O'BRIEN, Neil: Iridis-pi: a low-cost, compact demonstration cluster. In: *Cluster Computing* (2013), S. 1–10. – <http://dx.doi.org/10.1007/s10586-013-0282-7>
- [CW76] CURNOW, Harold ; WICHMANN, Brian: A Synthetic Benchmark. In: *The Computer Journal* (1976), S. 43–49. – <http://comjnl.oxfordjournals.org/content/19/1/43.full.pdf>
- [DLP03] DONGARRA, Jack ; LUSZCZEK, Piotr ; PETITET, Antoine: The LINPACK Benchmark: past, present and future. In: *Concurrency and Computation: Practice and Experience* (2003), S. 803–820. – <http://onlinelibrary.wiley.com/doi/10.1002/cpe.728/pdf>
- [Kie13] KIEPERT, Joshua: *Creating a Raspberry Pi-Based Beowulf Cluster.* 2013. – [http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster\\_v2.pdf](http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster_v2.pdf)
- [Kli13] KLINGER, Maximilian: *Evaluating the Feasibility and Performance of a Model Raspberry Pi Beowulf Cluster.* 2013. – zzz
- [Lan12] LANGER, Alexander: *Raspberry Pi Handbuch*, Februar 2012. – <http://raspberrycenter.de/handbuch/raspberry-pi-handbuch>
- [Ou13] OU, Jun: *Pi and the Sky.* 2013. – <https://www.duo.uio.no/bitstream/10852/37445/4/Ou-Jun.pdf>
- [Pow12] POWERS, Shawn: The Open-source Classroom: Your First Bite of Raspberry Pi. In: *Linux Journal* (2012), S. 48–54. – [http://dl.acm.org/ft\\_gateway.cfm?id=2422332&ftid=1329297&dwn=1&CFID=403460446&CFTOKEN=89580898](http://dl.acm.org/ft_gateway.cfm?id=2422332&ftid=1329297&dwn=1&CFID=403460446&CFTOKEN=89580898)
- [Pre11] PREVEZANOS, Christoph: *Computer-Lexikon 2012.* München : Markt+Technik, 2011. – ISBN 9783827247285

## *Literaturverzeichnis*

- [Rec06] RECHENBERG, Peter: *Informatik-Handbuch*. München : Hanser, 2006. – ISBN 9783446401853
- [Wei90] WEICKER, Reinhold: An Overview of Common Benchmarks. In: *Computer* (1990), S. 65–75. – [http://dlojewski.dl.funpic.de/download/Diplomarbeit/Andere\\_Dok/Dhry\\_Whet/OverviewOfCommonBenchmarks.pdf](http://dlojewski.dl.funpic.de/download/Diplomarbeit/Andere_Dok/Dhry_Whet/OverviewOfCommonBenchmarks.pdf)