INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

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

Judith Greif

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:	Dr. Nils gentschen Felde Christian Straube
Abgabetermin:	31. Mai 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. Mai 2014

. . . . . .
(Unterschrift der Kandidatin)

Abstract

Seit dem Beginn seiner Entwicklung punktet der Mini-Computer Raspberry Pi durch Flexibilität, Preis-Leistungs-Verhältnis niedrigschwelligen Zugang und geringen Stromverbrauch. Das macht ihn zum idealen Kandidaten für einen Beowulf-Cluster. Er kann z.B. an Universitäten zur Forschungszwecken eingesetzt werden oder in eingeschränktem Rahmen einen Supercomputer simulieren.

Tritt der Raspberry Pi in die Welt der Supercomputer ein, muss er sich auch mit ihren Spielregeln messen lassen. Die vorliegende Arbeit untersucht das Skalierungsverhalten eines Raspberry Pi-Clusters unter der Workload von Linpack, Whetstone und STREAM. Sie zeigt auf, ob und in welcher Form sich die ausgewählten HPC-Benchmarks auf dem Cluster auführen lassen und evaluiert die Ergebnisse. Ein Schwerpunkt liegt dabei auf dem Energieverbrauch des Clusters bei unterschiedlichen Versuchsaufbauten.

Inhaltsverzeichnis

1 Einleitung

Seit Beginn seiner Entwicklung im Jahr 2009 boomt der Minicomputer Raspberry Pi (im Folgenden als *RPi* bezeichnet). Er erhielt z.B. den Designpreis INDEX: award 2013 (vgl. `http://designtoimprovelife.dk/category/s15-award2013`), wurde als Innovation des Jahres bei den T3 Gadget Awards 2012 ausgezeichnet (vgl. `http://www.t3.com/news/t3-gadget-awards-2012-award-winners`) und zum Gadget of the Year 2012 des Linux Journal gewählt (vgl. `http://www.linuxjournal.com/slideshow/readers-choice-2012`). Im Februar dieses Jahres war das Modell B über 2,5 Millionen Mal verkauft worden.

1.1 Hintergrund

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 einzlnen Komponenten wie CPU, GPU und RAM zu evaluieren.

1.2 Fragestellung

Im wissenschaftlichen Umfeld werden immer häufiger mehrere RPis zu einem Beowulf-Cluster verschaltet, um parallele Berechnungen auszuführen. Vor diesem Hintergrund stellt sich die Frage: Wie verhält sich ein RPi bei der Ausführung von HPC-Benchmarks? Noch interessanter ist das Verhalten eines RPi-Clusters: 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 CPU-Performance und Skalierungsverhalten eines RPi-Clusters unter den Testbedingungen ausgewählter HPC-Benchmarks. Dazu wird zunächst die die Performance eines RPi-Einzelrechners ermittelt. Anschließend wird die Performance eines RPi-Clusters bei Ausführung der Benchmarks evaluiert. Im Fokus steht dabei der Energievebrauch, der mit Hilfe eines Strommessgeräts für die unterschiedlichen Versuchsaufbauten ermittelt wird.

1.3 Vorgehensweise

Grundsätzlich stellt sich die Frage, welche Benchmarks sich für das zu untersuchende System (RPi-Einzelrechner bzw. RPi-Cluster) und die zu untersuchende Komponente (CPU) eignen. Sie können nur herangezogen werden, wenn hierfür geeignete Implementierungen existieren.

Für den physischen Versuchsaufbau muss sicher gestellt sein, dass alle erforderlichen Komponenten (RPi-Cluster, RPi-Einzelrechner und Strommessgerät) zuverlässig arbeiten. Dies

muss regelmäßig überprüft werden, z.B. durch Kontrolle des Netzwerkstatus der Komponenten.

Die Ausführung der Benchmarks mit unterschiedlichen Anzahlen aktiver und angeschalteter RPi-Nodes erfolgt sinnvollerweise automatisiert durch entsprechende Shellskripte. Die Resultate müssen in geeigneter Form in eine Datenbank geloggt werden, was ebenfalls durch Shellskripte realisiert wird.

Der Stromverbrauch des RPi-Clusters mit unterschiedlichen Anzahlen aktiver und angeschalteter RPi-Nodes wird durch ein Strommessgerät ermittelt, das an die Stromversorgung des RPi-Clusters angeschlossen wird. Seine Messwerte müssen ebenfalls in geeigneter Form in eine Datenbank geloggt werden.

Die Messwerte der Benchmarks müssen mit denen des Strommessgeräts abgeglichen werden, um Aussagen über den Stromverbrauch treffen zu können. Für die Interpretation der ermittelten Versuchsergebnisse ist eine grafische Aufbereitung erforderlich.

1.4 Struktur

Um den Bezugsrahmen zu verdeutlichen, werden in Kapitel 2 zunächst grundlegende Definitionen geklärt. Insbesondere werden die ausgewählten Benchmarks vorgestellt (vgl. Kap. 2.4) und die Spezifikationen von RPi (vgl. Kap. 2.5) und RPi-Cluster erläutert (vgl. Kap. 2.6). Versuchsaufbau und -ablauf werden mit Blick auf Auswahl und Anpassung der RPi-spezifischen Parameter im folgenden Kapitel dargestellt (vgl. Kap. 3). Schließlich werden die Messergebnisse dargestellt (vgl. Kap. 3.4) 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 näherer Betrachtung 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. Auf solche Routinen darf sich die Untersuchung nicht stützen. Im Folgenden wird daher auf grundlegende Begriffe der Untersuchung eingegangen. Anschließend werden die verwendeten Benchmarks (vgl. 2.4) und ihre Ausführung auf dem 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:

> 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].

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

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

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

2.2 Definition: Performance

Die physikalische Größe *Leistung* ist als *Energie pro Zeit* definiert. In der Informatik wird der Begriff meist abweichend als Rechenleistung verwendet, d.h. das Zeitverhalten von Programmen und Geräten oder die Leistungsfähigkeit eines Computersystems. Eine Abgrenzung erscheint jedoch schwierig:

> The performance of a computer is a complicated issue, a function of many interrelated 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].

Das *Informatik-Handbuch* liefert folgende Aspekte:

> 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 soll die Performance eines RPi-Clusters bei der Ausführung von Benchmark-Programmen ermittelt werden. Curnow erläutert 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].

Mit *Performance* wird daher hier die ***Rechenleistung*** in diesem abgegrenzten Sinn bezeichnet, d.h. die Datenverarbeitungsgeschwindigkeit eines spezifischen Systems, nicht dessen Leistung im physikalischen Sinn. Dabei werden die ***Ausführungsrate*** und die ***Ausführungszeit*** als maßgebliche Zahlen betrachtet.

2.3 Definition: Energieverbrauch

Obwohl der Begriff *Energieverbrauch* (engl. *energy consumption* oder *power consumption*) weit verbreitet ist (vgl. z.B. [FH12] und [BL08]), stimmt er nicht mit der physikalischen Definition überein: Nach dem Energieerhaltungssatz bleibt die Gesamtenergie in einem geschlossenen System konstant. Dennoch wird der Begriff auch in der Informatik häufig verwendet, um die Leistungsfähigkeit eines Rechners oder Rechensystems in Abhängigkeit von der Energieaufnahme zu charakterisieren:

> Vom physikalischen Standpunkt aus betrachtet, ist der Ausdruck „Energieverbrauch“ falsch. Energie kann nicht verbraucht oder erzeugt, sondern nur umgewandelt werden. Trotzdem hat sich dieser Bergriff eingebürgert [...]. Bei elektronischen Systemen wird elektrische Energie verwendet, um Rechenleistung in einer bestimmten Zeitspanne zu generieren, physikalisch gesehen wird die elektrische Energie jedoch zu nahezu hundert Prozent in Wärmeenergie umgewandelt [LB12].

In dieser Arbeit liegt ein Schwerpunkt der Leistungsbewertung auf der Energieaufnahme eines RPi-Clusters bei der Ausführung von Benchmark-Programmen. Mit *Energieverbrauch* ist hier der Exergieverbrauch bzw. die Entropieerzeugung in Watt im physikalischen Sinn gemeint.

2.4 Benchmarks

Aus der Fülle an existierenden Benchmarks eine Auswahl zu treffen erweist sich als trickreich. Mögliche Kriterien für die Auswahl nennt Weicker:

> 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].

Der Benchmark muss zudem überhaupt auf das gewählte System anwendbar sein, d.h. das System muss dessen Spezifikationen erfüllen bzw. es muss eine geeignete Implementierung vorliegen. Aus diesem Grund musste z.B. SHOC ausscheiden. SHOC ist nicht lauffähig auf dem RPi, da Open CL nicht unterstützt wird.

Für die Untersuchung wurden drei etablierte HPC-Benchmarks vorgegeben, 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 zwischen der Linpack-Bibliothek und dem Linpack-Benchmark unterschieden werden. Erstere ist eine numerische Programmbibliothek zum Lösen von linearen Gleichungssystemen und galt darin lange Zeit als Standard. Der Linpack-Benchmark basiert auf zwei ihrer Routinen:

> Linpack was designed out of a real, purposeful program that is now used as a benchmark [Wei90].

Er wurde hauptsächlich von Jack Dongarra entwickelt und wird seit den 1970er Jahren zur Klassifizierung von Rechnern verwendet:

> Over the years additional performance data was added [...] and today the collection includes over 1300 different computer systems [DLP03].

Seit 1993 werden die leistungsfähigsten Supercomputer der Welt durch die Top500-Rankings ermittelt. Hierzu dient die Variante HPLinpack, auch $N \times N$ Linpack oder High Parallel Computing genannt:

> 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].

Dort findet sich auch eine vertiefte Darstellung des Linpack-Benchmarks.

Funktionsweise

Linpack dient der Ermittlung der CPU-Leistung. Dazu werden Fließpunkt-Operationen auf einer Matrix durchgeführt, die intern in eine lineare Darstellung umgewandelt wird. Das Ergebnis wird als *Performance* in FLOPs („Floating Point Operations Per second“) ermittelt. 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]). Die Linpack-Varianten Linpack 100, Linpack 1000 und HPLinpack verwenden Matrizen der Größe 100×100, 1000×1000 bzw. $n \times n$, d.h. eine Matrix variabler Größe, zur Berechnung. HPLinpack liefert Messergebnisse für R_{max} („Performance for the largest problem“), N_{max} (“Size of the largest problem run“), $N_{1/2}$ (“Size where half the R_{max} execution rate is achieved“) und R_{peak} („Theoretical peak performance“) in *FLOPs*. Entscheidend für die Positionierung in der Top500-Liste ist dabei der R_{max}-Wert. Dabei erreicht der derzeit leistungsfähigste Supercomputer, *Tianhe-2 (MilkyWay-2)*, 33862.7 TFLOPs. *SuperMUC*, der zuletzt auf Platz 10 der Top500 gerankt wurde, erzielte 2897.0 TFLOPs.

Beim Vergleich der Ergebnisse von Linpack ist die Matrixgröße von Bedeutung, da ihre Abänderung bei unterschiedlichen Rechnerarchitekturen zu geringerer Datenlokalität und damit zu starken Abweichungen der Ergebnisse führen kann (vgl. [Wei90]). In der Regel werden daher die oben genannten Matrixgrößen verwendet, obwohl der Quellcode eine Änderung 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. Somit liegen bereits Vergleichswerte für einen RPi-Einzelrechner vor. Auch Implementierungen von Linpack für verteilte Systeme unabhängig von den Top500-Rankings sind im Umlauf. Der hier verwendete Quellcode findet sich unter `http://www.netlib.org/benchmark/hpl`.

2.4.2 Whetstone

Auch Whetstone ist als Benchmark im HPC-Bereich und zur Klassifizierung von Einzelrechnern seit vielen Jahren etabliert:

> [T]he 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].

Er wurde 1976 von Roy Longbottom und anderen entwickelt und gilt als erstes Programm, das jemals explizit für das Benchmarking industrieller Standards designt wurde (vgl. [Wei90]). Wie Linpack misst er die CPU-Leistung.

Funktionsweise

Die Funktionsweise von Whetstone ähnelt Linpack. Allerdings werden nicht nur Fließkomma-Berechnungen ausgeführt. Auch mathematische Funktionen, Integer-Arithmetik, bedingte Anweisungen etc. kommen zur Anwendung, da die ursprüngliche Programmversion nicht komplex genug war, um ein durchschnittliches FORTRAN-Programm zu simulieren. Gegenüber der ursprünglichen Implementierung in ALGOL 60 war ein damals gängiger FORTRAN-Compiler in der Lage, Zwischenergebnisse in schnellen Registern abzuspeichern und darauf zurückzugreifen. So fiel die gemessene CPU-Leistung deutlich höher aus als erwartet (vgl. [CW76]).

Die einzelnen Teile werden als Module bezeichnet und sind jeweils in eine For-Schleife eingebettet, die viele Male hintereinander ausgeführt wird. Ein Rahmenprogramm steuert Aufruf der Module und Ausgabe der Ergebnisse. Auch die Ausgabe ist Teil des Benchmarks, allerdings bemerkt Curnow hierzu:

> 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 [CW76].

Die Ergebnisse wurden zunächst in *Whetstone Instructions per Second* gemessen, heutige Implementierungen liefern Ergebnisse in MFLOPs. Eine detaillierte Darstellung der Entwicklung und Implementierung von Whetstone findet sich bei [CW76], ebenso der ursprüngliche Quellcode in ALGOL 60.

Whetstone auf dem Raspberry Pi

Whetstone erschien wie für den RPi gemacht, da er sich als Standard für Mini-Computer etabliert hat: Der Benchmark war in den 70er Jahren für Maschinen mit deutlich geringerer Rechenleistung als heute entwickelt worden. Bereits 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 [CW76].

Er wurde vom Entwickler selbst für den RPi angepasst und und auf diesem getestet (vgl. `http://www.roylongbottom.org.uk/Raspberry%20Pi%20Benchmarks.htm`). Auch hier liegen also Vergleichswerte für einen RPi-Einzelrechner vor.

Wie für die meisten etablierten HPC-Benchmarks existieren Implementierungen in höheren Programmiersprachen wie C, C++, Fortran und Java (vgl. `http://freespace.virgin.net/roy.longbottom`). Ob eine lauffähige Implementierung für die Cluster-Architektur existiert, wird sich zeigen müssen.

2.4.3 STREAM

Mit zunehmender Rechenleistung der Prozessoren und der Zunahme an Prozessorkernen innerhalb eines Rechnersystems zeigte sich, dass es nicht ausreicht, die Leistungsfähigkeit lediglich an Hand der CPU-Leistung zu bewerten. Zudem lässt sich eine Tendenz beobachten, dass Rechnersysteme gezielt für die Ausführung bestimmter Benchmarks wie HPLinpack optimiert werden, um in Ranglisten bessere Plätze zu erreichen.

STREAM wurde mit dem Ziel entwickelt, das Verhältnis von CPU-Leistung zu Speichergeschwindigkeit in die Bewertung einzubeziehen. Mittlerweile ist STREAM auch Bestandteil der HPCChallenge Benchmark-Suite. Sie wurde mit der Zielsetzung entwickelt, Anforderungen aus der Anwendungspraxis in die Bewertung von HPC-Rechnersystemen einfließen zu lassen (vgl. [LDK+05]).

Funktionsweise

STREAM ist ein synthetischer Benchmark und basiert auf einem Programm aus der wissenschaftlichen Praxis von John McCalpin. Es ermittelt die dauerhafte Speicher-Bandbreite (engl. „sustainable memory bandwidth“ [McC95]) für angrenzende Speicherzugriffe langer Vektoren (engl. „sustainable memory bandwidth for contiguous, long-vector memory accesses“[McC95]). Die Vektoren müssen lang genug sein, dass sie nicht im Prozessor-Cache vorgehalten, sondern aus dem Arbeitsspeicher geladen werden müssen. STREAM ist somit Teil eines neuen Modells, das als Performance-Maßzahl die Gesamtzeit $T_{total} = T_{cpu}$ (CPU-Performance) + T_{memory} (Speicher-Performance, ermittelt durch STREAM) festlegt (vgl. [McC95] und [McC05]).

STREAM durchläuft bei seiner Ausführung vier Module (Copy, Scale, Add und Triad) und liefert für jedes Modul die durchschnittliche, maximale und minimale Ausführungszeit sowie die Ausführungsrate. Jedes Modul wird mehrfach ausgeführt, um verlässliche Mittelwerte zu erhalten. Folgende Berechnungen werden dabei durchgeführt:

1. Copy: $a(i) = b(i)$
2. Scale: $a(i) = q * b(i)$
3. Add: $a(i) = b(i) + c(i)$
4. Triad: $a(i) = b(i) + q * c(i)$

Zwischen dem Programmcode der Module bestehen Abhängigkeiten, um starke Optimierungen durch den Compiler zu verhindern (vgl. [McC05]). Quellcode und Ergebnisse für STREAM auf verschiedenen Systemen finden sich unter `http://www.cs.virginia.edu/stream`.

STREAM auf dem Raspberry Pi

Auch für STREAM wurden bereits Ergebnisse auf einem RPi-Einzelrechner bereits veröffentlicht (vgl. `http://www.cs.virginia.edu/stream/stream_mail/2012/0002.html`). Die auf dem RPi-Cluster verwendete Implementierung findet sich unter `http://www.cs.virginia.edu/stream/FTP/Code/Versions/stream_mpi.f`.

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

2.5.1 Physischer Aufbau

Der RPi ist mit einem Broadcom BCM2835 als System on a Chip ausgestattet. Es enthält CPU (ein ARM1176JZFS-Prozessor) und GPU (ein Broadcom VideoCore IV-Koprozessor, vgl. [Lan12]). Auffällig ist, dass die CPU des RPi (ein ARM-Chip, wie er auch in Mobiltelefonen zur Anwendung kommt), relativ schwach ist im Vergleich zur GPU. Diese unterstützt Full HD-Auflösung und das Hardware-beschleunigte Rendern verschiedener Videoformate. Das Modell B verfügt über 512 MB SDRAM. Er 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]).

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 RJ45-Buchse zum Anschluss einer 10/100 Base-T 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 700 mA/5 V benötigt (vgl. [Pow12]). Der Vollständigkeit halber sei das Vorhandensein einer Allzweckeingabe/-ausgabe mit sechs Anschlüssen und jeweils 26 Pins erwähnt. Darüber können, ähnlich wie bei einem Arduino-Board, z.B. LEDs, Sensoren und Displays angesteuert werden.

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 (vgl. [Pow12]). Daneben gibt es Implementierungen von RISC OS und Plan 9. Als Betriebssytem für die RPi-Einzelrechner war Raspbian gewählt worden, die offizielle Distribution der Raspberry Pi Foundation (vgl. `http://www.raspberrypi.org/faqs`). Das Abbild findet sich unter `http://downloads.raspberrypi.org/raspbian_latest`.

2.6 Raspberry Pi-Cluster als Testumfeld

Seit einiger Zeit lässt sich die Tendenz beobachten, eine größere Anzahl von Raspberry Pis zu einem Cluster zu koppeln (weitere Projekte werden in [CCB+13], [Kie13], [Bal12] und

[Ou13] dargestellt). Die hier verwendete Architektur wird im Folgenden kurz vorgestellt. Eine detaillierte Darstellung liefert [Kli13].

2.6.1 Physischer Aufbau

Der RPi-Cluster besteht aus 20 RPi Modell B-Einzelrechnern ((1), DNS-Namen `pi01` – `pi20`). Sie sind jeweils über ein Ethernet-Kabel (2) mit einem 24 Port Gigabit-Switch (3) verbunden, der an einen zentralen x86-Server (DNS-Name `careme`) angeschlossen ist.

Er besteht aus einem Mini-ITX-Mainboard (4) und Festplatten mit einem Software-RAID Level 5-System für den Netzwerkzugriff und einem RAID Level 1-System für das Betriebssystem (5).

Die Stromversorgung erfolgt über ein zentrales Netzteil (6). Daran sind zwei Verteiler (7) angeschlossen, an die jeweils 10 RPi-Einzelrechner über Mini-USB-Kabel (8) verbunden sind.

Alle Komponenten befinden sich in einem Metallgehäuse, in dessen Mitte drei Kühlgebläse angebracht sind. Ein solches System relativ kostengünstiger Rechner mit einem BSD- oder

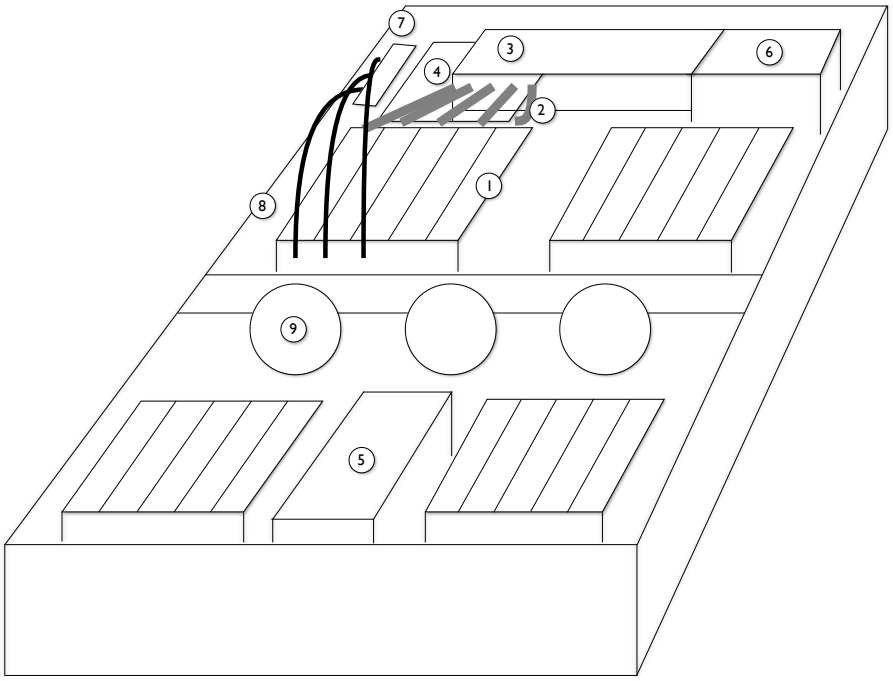

Abbildung 2.1: Schematischer Aufbau des hier verwendeten Bramble.

Linux-Betriebssystem, die über IP kommunizieren, wird im Allgemeinen als *Beowulf* bezeichnet (vgl. [Kie13]). Für einen Beowulf-Cluster aus RPis ist der Begriff *Bramble* gebräuchlich, der im Folgenden verwendet wird (vgl. `www.raspberrypi.org/archives/tag/bramble`).

2.6.2 Betriebssystem und Dateisystem

Auf den RPi-Knoten ist das oben genannte Betriebssytem Raspbian installiert (vgl. 2.5.2), auf `careme` eine Standard-Debian-Version. Ein Charakteristikum eines Beowulf-Clusters ist,

dass es kein Shared Memory-Interface und keine Cache-Kohärenz gibt. Somit stellt sich die Frage nach der Zeit- und Datensynchronisation der RPi-Knoten und des Servers sowie den Möglichkeiten des gemeinsamen Speicherzugriffs.

Die Datei-Synchronisation des Servers und der RPi-Knoten erfolgt über ein Netz-Dateisystem (hier: `nfs`). Dessen Verwaltung erfolgt auf dem Server über ein Union-Dateisystem (hier: `aufs`), das mehrere Schichten des Dateisystems anlegt: Die unteren Schichten haben nur Leseberechtigungen, die oberste Lese-Schreib-Berechtigungen. Werden Daten in einem geteilten Verzeichnis verändert, so bleiben die unteren Schichten unverändert. Die Änderungen werden nur auf der obersten Schicht geschrieben. Für gemeinsam genutzte Daten des Servers und der Clients steht auf dem Server das Verzeichnis `/srv/nfs-share` bzw. auf den Clients `/srv` zur Verfügung. Eine genaue Darstellung findet sich bei [Kli13].

2.6.3 Implikationen für den Versuchsaufbau

Welche Zugriffsmöglichkeiten auf die Komponenten des Bramble gibt es, und wie können verteilte Berechnungen darauf ausgeführt werden?

Netzzugriff

Der Zugriff auf den Bramble erfolgt aus dem internen Netz über IP und SSH. Auf die einzelnen RPi-Knoten wird von `careme` aus ebenfalls über IP und SSH zugegriffen (private Adressen `10.0.0.2` bis `10.0.0.21`).

Es gibt auf jedem RPi-Knoten den Raspbian-Standard-User `pi`, der nicht verändert wurde. Aus Sicherheitsgründen wurden für `careme` keine root-Rechte erteilt, nur für die einzelnen RPi-Knoten. Stattdessen wurde der Benutzer `rpi-user` eingerichtet. Das bedeutet, dass alle Änderungen im geteilten Verzeichnis `/srv` bzw. `/srv/nfs-share` von einem der RPi-Nodes aus vorgenommen werden müssen, da `rpi-user` hierauf keine Schreibberechtigungen hat. Somit wurde vorab RPi-Node `pi03` als Berechnungsknoten definiert, von dem aus alle Skripte ausgeführt, Programme installiert werden etc.. Die Datenbank zur Speicherung von Konfigurationen und Messwerten war hingegen auf dem Server angelegt worden. Dadurch wird häufiges Navigieren zwischen dem Server und dem Berechnungsknoten mit unterschiedlichen Benutzern notwendig.

Die Skripte zur Ausführung der Benchmarks sehen häufig das Herunterfahren einzelner RPi-Knoten vor, um den Energieverbrauch beim Abschalten nicht aktiver RPi-Knoten zu ermitteln. Alle Anwendungen wurden daher auf maximal n=19 RPi-Nodes ausgeführt und von `pi03` aus gesteuert.

Ausführung verteilter Anwendungen

Der Versuchsaufbau sieht vor, verteilte Anwendungen auf unterschiedlichen Anzahlen von RPi-Knoten auszuführen. 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-Knoten und die Anzahl der darauf auszuführenden Prozesse zu spezifizieren, muss ein entsprechendes Machinefile erstellt und im gesharten Verzeichnis abgelegt werden. Dieses wird der Ausführung von MPICH mit `mpiexec` als Parameter übergeben (vgl. Kap. 3.2.2).

3 Versuchsaufbau und -ablauf

Das folgende Kapitel beschreibt Versuchsaufbau und -durchführung sowie Ziele der Messung, von der Erkenntnisse über das Skalierungsverhalten eines RPi-Clusters unter der Arbeitslast der ausgewählten HPC-Benchmarks erwartet werden.

3.1 Zielsetzung

Für das Skalierungsverhalten des Bramble unter der Arbeitslast von Linpack und STREAM werden wie in Kap. 2 beschrieben zwei Maßzahlen betrachtet: Performance und Energieverbrauch.

Die augewählten Benchmarks werden auf n – 1 RPi-Knoten des Bramble mit n=19 RPi-Nodes ausgeführt (vgl. Kap. 2.6.3). Alle Messungen werden zweimal durchgeführt: Mit Stromanschluss der nicht beteiligten RPi-Nodes und ohne. Für beide Benchmarks werden zwei Ergebnisparameter betrachtet: Ausführungsrate in GFLOPs und Ausführungszeit in s für HPLinpack, Ausführungsrate in MB/s und durchschnittliche Ausführungszeit in s für STREAM. Die Ergebnisse der Messreihen werden anschließend gegenübergestellt (vgl. Kap. 3.4).

3.2 Aufbau und Art der Messung

Hier stellen sich zwei grundsätzliche Fragen: Welcher Art ist die Messung und welche Voraussetzungen müssen hierfür erfüllt sein?

Die Performance wird pro Benchmark durch zwei Ergebnisparameter ermittelt. Zwei Messreihen werden durchgeführt: Bramble mit n RPi-Knoten aktiv/20 RPi-Knoten angeschaltet, Bramble mit n RPi-Knoten aktiv/n RPi-Knoten angeschaltet. Für den Versuchsaufbau sind folgende Aspekte von Bedeutung: Mögliche Modifikationen der RPi-Knoten, Zeitsynchronisation der RPi-Knoten, Skalierung der Messung auf n – 1 RPi-Knoten, automatisierte Durchführung der Messung auf n – 1 RPi-Knoten sowie Einlesen der Messwerte in eine geeignete Datenstruktur.

3.2.1 Versuchsaufbau Ri-Einzelrechner

Viele Nutzer stellen sich nach Inbetriebnahme eines RPi-Einzelrechners die Frage nach Swap-Speicher und Übertakten (vgl. [Pow12]). Beides liegt nahe, da das Modell B des RPi nur über 512 MB Arbeitsspeicher verfügt. Die CPU-Leistung ist mit 700 MHz ebenfalls eher niedrig.

Im Praxisbetrieb wurde gezeigt, dass ein Übertakten der CPU auf bis zu 1 GHz gefahrlos möglich ist (vgl. z.B. `http://www.raspberrypi.org/introducing-turbo-mode-up-to-50-more-performance-for-free/`). Bei den hier verwendeten RPis wurde davon Abstand genommen, da es wenig zielführend erscheint, die Komponente zu manipulieren, deren Perfor-

mance man durch Benchmarking evaluieren möchte[1] Allerdings gibt es bereits Ergebnisse für Linpack 100 und Whetstone bei auf einem RPi-Einzelrechner bei Übertakten der CPU auf 1 MHz (vgl. `http://www.roylongbottom.org.uk/Raspberry%20Pi%20Benchmarks.htm`).

Zur Allokierung von Swap-Speicher auf dem RPi gibt es grundsätzlich drei Möglichkeiten: Swap-Datei, Swap-Partition oder zRAM.

Das Betriebssystem Raspbian verwendet standardmäßig eine Swap-Datei `/var/swap` auf der SD-Karte (vgl. `http://raspberrypi.stackexchange.com/questions/70/how-to-set-up-swap-space`). Hierbei zeigen sich Probleme: Erstens können ständige Schreibzugriffe auf Dauer die SD-Karte beschädigen. Zweitens sind Schreibzugriffe darauf sehr langsam, was die Performance des Systems bei hoher Arbeitsspeicherlast beeinträchtigen kann (vgl. [Pow12]).

Das gilt auch für die Allokierung einer auf Unix-Systemen üblicherweise genutzten Swap-Partition, weswegen diese Möglichkeit in der Praxis keine Rolle spielt.

Eine weitere Möglichkeit des Swapping ist die Verwendung von zRAM. Hierbei wird ein Teil des Arbeitsspeichers komprimiert und als Swap Space genutzt. Hierbei werden keine Zugriffe auf die SD-Karte notwendig (vgl. [Pow12]).

Auf dem Bramble-Server war kein Swap-Speicher allokiert worden (vgl. [Kli13]). Trotz der beschriebenen Schwierigkeiten wird auf den RPi-Knoten ein großer Teil des Speichers auf den SD-Karten als Swap-Speicher genutzt, um die Nutzung von Programmen mit mehr Speicherbedarf zu ermöglichen und das Netzwerk durch das Cachen häufig verwendeter Dateien zu entlasten (vgl. [Kli13]).

3.2.2 Versuchsaufbau Bramble

Das Komponentendiagramm 3.1 zeigt den Versuchsaufbau pro Messreihe auf dem Bramble mit Stromversorgung, Netzanschluss und Strommessgerät als phyischen Komponenten und der MySQL-Datenbank als logischer Komponente. Eine Messreihe wird als *ExperimentSuite* bezeichnet.

Modifikation der RPi-Knoten

Zu Beginn der Untersuchung zeigte sich, dass bei einige Mini-USB-Kabel zur Stromversorgung der RPis einen Wackelkontakt hatten oder ganz defekt waren. Um die in Kap. 1.3 genannte Zuverlässigkeit des Versuchsaufbaus sicherzustellen, wurden sie durch funktionsfähige Kabel ersetzt.

Zeitsynchronisation der RPi-Knoten

Der RPi-Einzelrechner besitzt aus Kostengründen keine Systemuhr (vgl. [Sch13]), sondern synchronisiert sich beim Booten gegen einen NTP-Server im Internet. Für die parallele Ausführung eines Programms auf mehreren Rechnerkernen ist die Zeitsynchronisation der RPi-Knoten und des Servers essentiell. Auf dem Bramble-Server gibt es daher einen OpenNTP-Server, gegen den sich die RPi-Knoten synchronisieren (vgl. [Kli13]).

[1] Für eine zukünftige Untersuchung wäre es interessant zu ermitteln, ob man den relativ hohen Stromverbrauch des Bramble bei Niedriglast (vgl. [Kli13]) durch Untertakten der einzelnen CPUs senken kann (vgl. Kap. 5).

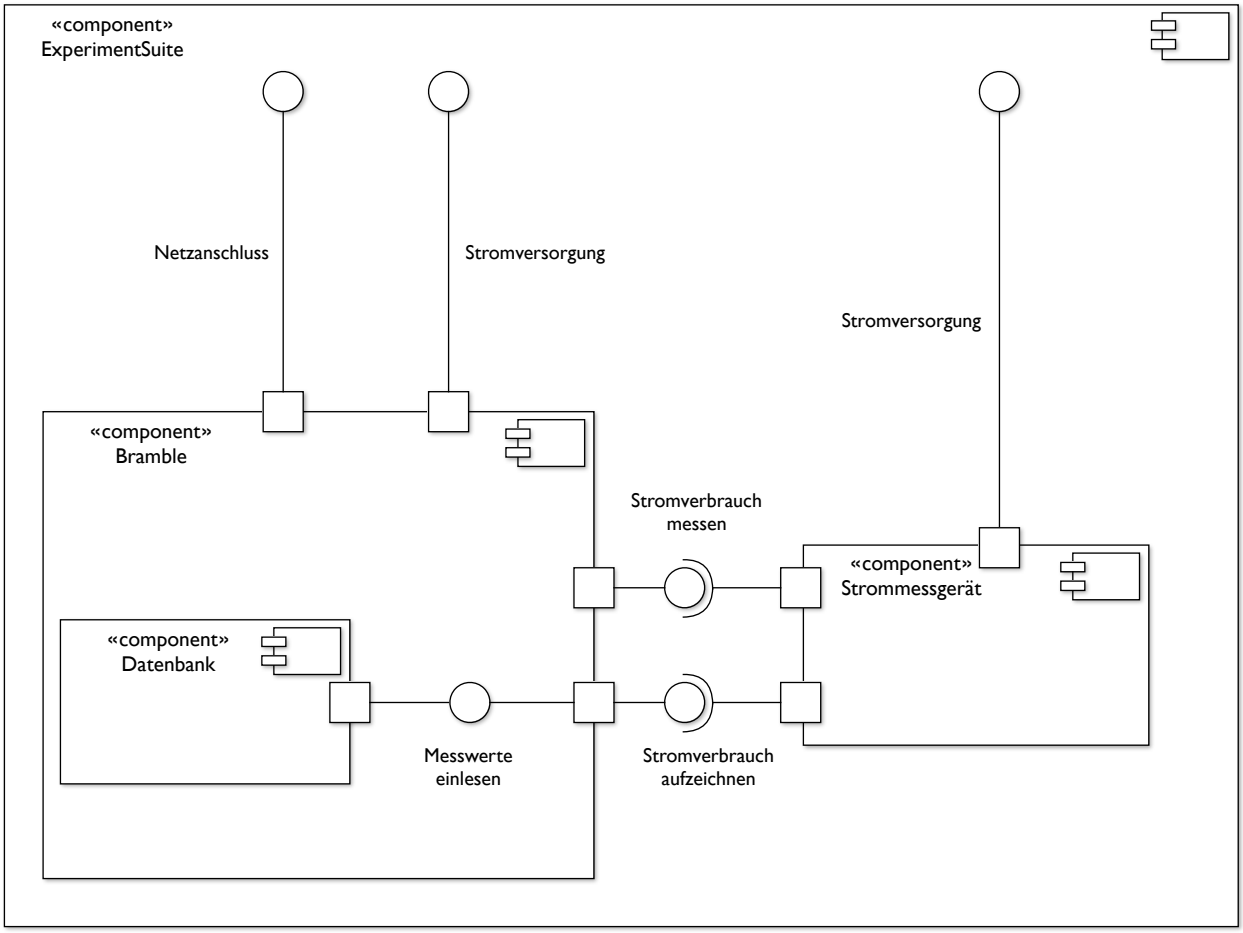

Abbildung 3.1: Komponentendiagramm des Versuchsaufbaus.

Skalierung der Messung auf n – 1 RPi-Knoten

Das Aktivitätsdiagramm 3.2 zeigt, welche Schritte aus Benutzersicht für die Durchführung einer ExperimentSuite, d.h. der Ausführung eines Benchmarks auf einer gewählten Anzahl aktiver und angeschalteter RPi-Knoten erforderlich sind. Es wird mit n=19 RPi-Knoten begonnen und einmal über alle Knoten von 19 – 1 (ohne den Ausführungsknoten `pi03`) iteriert. Danach wird die zweite Messung durchgeführt, wobei nach jedem Iterationsschritt der nicht mehr benötigte RPi-Knoten abgeschaltet wird. Jede Iteration der Benchmark-Ausführung läuft prinzipell gleich ab, während am Anfang und am Ende spezielle Vorkehrungen zu treffen sind.

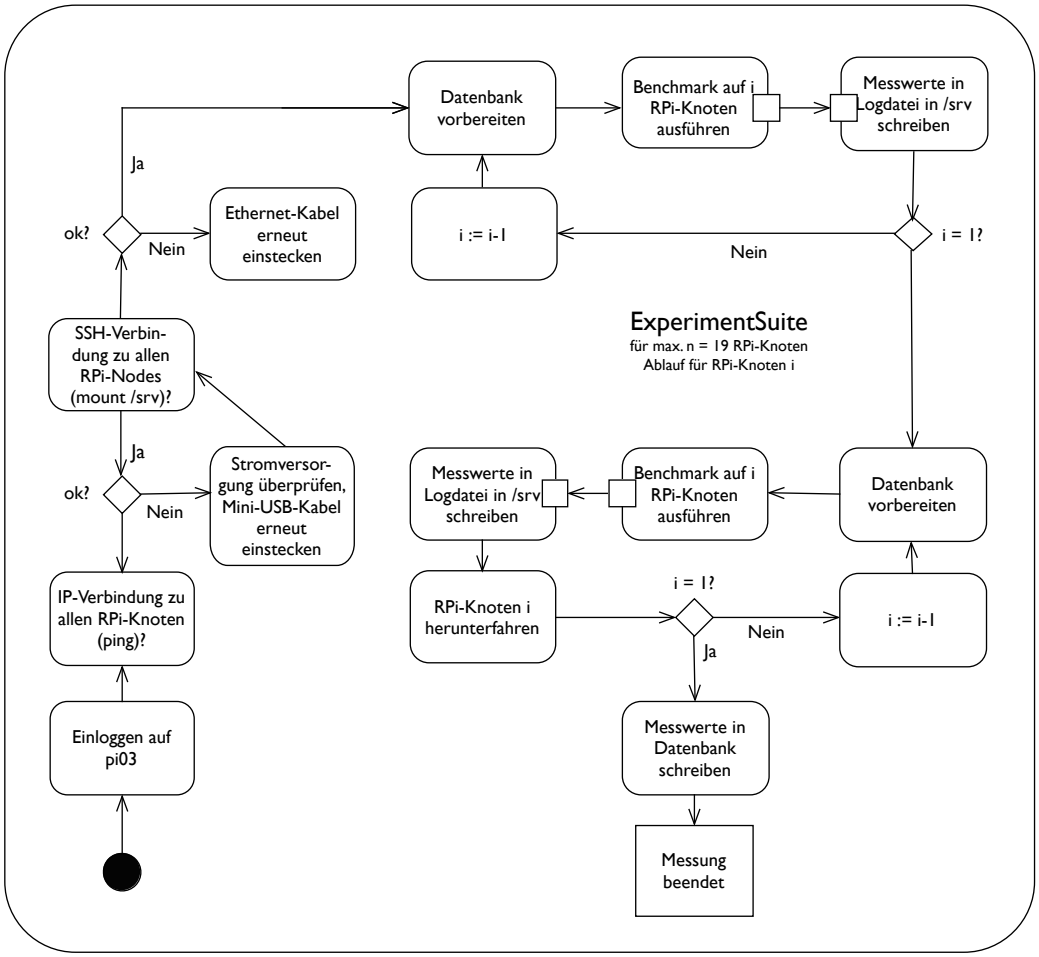

Abbildung 3.2: Aktivitätsdiagramm einer ExperimentSuite.

Konfiguration der Datenbank

Die Konfigurationen der Benchmarks und der ExperimentSuites sowie die Messergebnisse der Benchmarks werden in einer MySQL-Datenbank auf `careme` abgelegt. Dafür war ein Datenbankschema vorgegeben worden, das die Bramble-ExperimentSuites in einen größeren Versuchsaufbau integriert. Während der praktischen Arbeit wurde das Schema geringfügig an die tatsächlichen Erfordernisse angepasst. Z.B. wurde für jedes ausgeführte Teilmodul eines Benchmarks ein Messpaarameter definiert, der Unix-Timestamp seines Ausführungsendes ermittelt und zusammen mit dem jeweiligen Messwert in die Datenbank eingelesen.

Die Vorbereitung der Datenbank für den Versuchaufbau erfolgt in vier Schritten:

1. **Name und Beschreibung des Benchmarks.**
2. **Beschreibung des Versuchsaufbaus.**
3. **Konfiguration des Versuchsaufbaus.**

4. Verknüpfung von Benchmark-Konfiguration und Versuchsaufbau.

Das Aktivitätsdiagramm 3.3 visualisiert Schritt 1. Hier werden die statischen Konfigurationen für STREAM und HPLinpack festgelegt, was durch zwei Shellskripte `loadGeneratorConfigHpl.sh` und `loadGeneratorConfigStream.sh` realisiert wird. Sie werden zu Beginn des Versuchs einmal ausgeführt. Pro Benchmark muss in drei Tabellen jeweils ein Eintrag erstellt werden: In der Tabelle `LoadGenerator` muss ein Eintrag mit Werten für Name und Beschreibung des Benchmarks eingegeben werden. In der Tabelle `ENUM_LoadGeneratorConfigurationKey` muss Eintrag mit einem Wert für den Konfigurationsschlüssel des Benchmarks erstellt werden. In der Tabelle `LoadGeneratorConfiguration` muss ebenfalls ein Eintrag mit diesem Wert erstellt werden. Die dynamischen Schritte 2 – 4, die pro ExperimentSui-

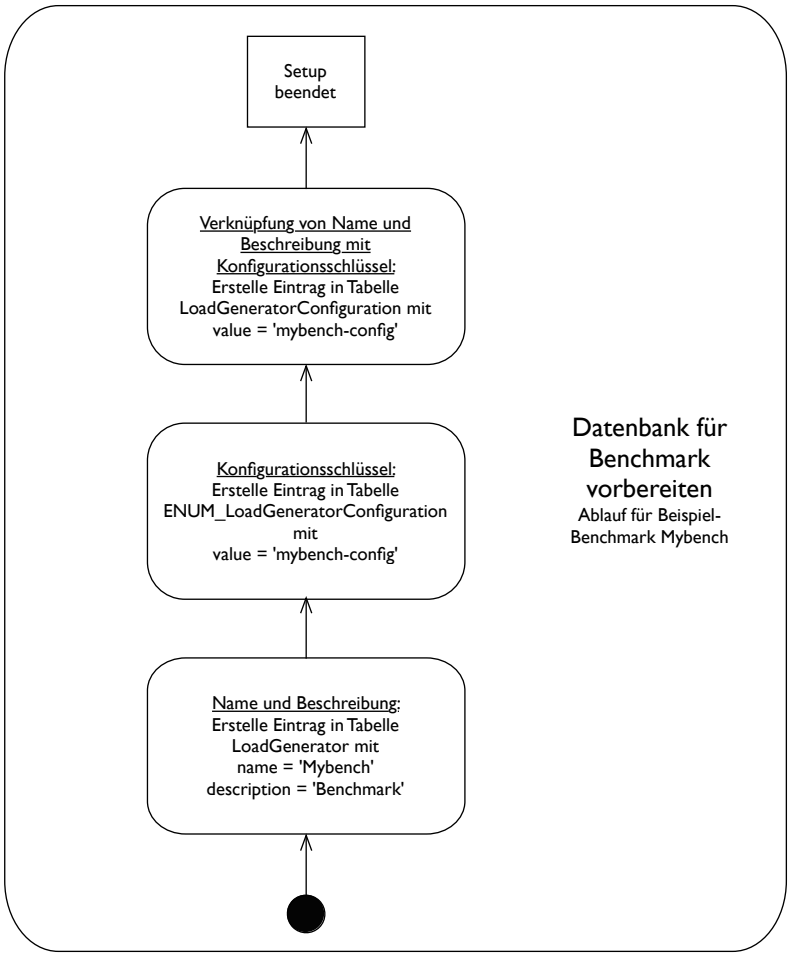

Abbildung 3.3: Aktivitätsdiagramm zur Vorbereitung der Datenbank für einen Benchmark.

te ausgeführt werden (Name, Beschreibung und Konfiguration des Versuchsaufbaus sowie Verknüpfung von ExperimentSuite und Benchmark-Konfiguration) müssen pro Benchmark-

Ausführung einmal erfolgen. Daher sind sie in die Ausführungsskripte für die Benchmarks integriert, die im Folgenden dargestellt werden. Sie werden in Diagramm 3.4 veranschaulicht.

Automatisierte Durchführung der Messung auf n – 1 RPi-Knoten

Die automatisierte Durchführung der Messung erfolgt durch Shellskripte. Sie werden im geteilten Verzeichnis abgelegt und können von `pi03` oder einem anderen Ausführungsknoten aus gestartet werden. Folgende Schritte werden durch die Skripte realisiert:

1. **Erstellen eines Machinefile zur Verteilung der Arbeitslast auf n RPi-Nodes, ggf. Löschen des alten.**
   Der Aufruf von MPICH mit `mpiexec` zur verteilten Ausführung eines Programms auf mehreren CPUs erfolgt mit dem Parameter `-machinefile`. Damit wird auf die Datei verwiesen, in der die Reihenfolge der zu nutzenden CPUs und die Anzahl der darauf auszuführenden Prozesse angegeben ist. Ist eine CPU nicht verfügbar, weil z.B. der entsprechende RPi-Knoten heruntergefahren wurde, wird der Prozess auf der nächsten verfügbaren CPU gestartet.

   Hierbei ist darauf zu achten, dass nur so viele Prozesse vergeben werden, wie auch angeschaltete RPi-Knoten zur Verfügung stehen. Die CPUs der RPi-Knoten müssen in der Reihenfolge angegeben werden, in der das Ausführungsskript das Herunterfahren der RPi-Knoten vorsieht. Das Machinefile muss in dem Verzeichnis abgelegt werden, von dem aus das auszuführende Programm gestartet wird. Um Verwechslungen vorzubeugen, wird ein eventuell dort liegendes, veraltetes Machinefile gelöscht.

2. **Einhängen des geteilten Verzeichnisses auf allen RPis.**
   Alle die ExperimentSuites betreffenden Dateien, z.B. die ausführbaren Dateien der Benchmark-Programme und das Machinefile, liegen im geteilten Verzeichnis `/srv` bzw. `/srv/nfs-share`. Um sicherzustellen, dass ein Benchmark-Programm auf einem RPi-Knoten ausgeführt werden kann, muss das geteilte Verzeichnis auf dem jeweiligen RPi-Knoten eingehängt sein.

3. **Navigation ins Arbeitsverzeichnis der ExperimentSuite.**
   Alle Shellskripte zur Konfiguration und Ausführung der ExperimentSuites wurden in einem Verzeichnis `experimentsuite` im geteilten Verzeichnis `/srv` abgelegt. Die Ergebnisdaten werden ebenfalls dort abgelegt.

4. **Iteration über n ausgewählte Benchmarks:**
   a) **Erstellen von Logdateien.**
      Für die Ergebnisse der Benchmarks werden Logdateien vorbereitet, in die später die Ausgabe der Benchmarks geschrieben wird.

   b) **Iteration über n RPi-Knoten:**
      i. **Datenbank für ExperimentSuite vorbereiten.**
         Das Aktivitätsdiagramm 3.4 zeigt das Vorbereiten der Datenbank `rpiWerte` für die aktuelle Iteration: Für jede Ausführung eines Benchmark-Programms muss ein Eintrag in der Tabelle `ExperimentSuite` mit Werten für Granularität, Ziel und Startzeitpunkt erstellt werden. In der Tabelle `Experiment-SuiteConfiguration` muss ein Eintrag mit Werten für die Anzahlen aktiver

und angeschalteter RPi-Knoten erstellt werden. Beide Tabellen werden über die Tabelle `N2M_loadConf3expSuite` miteinander verknüpft, in der ein Eintrag mit Werten für die ID des Benchmarks und die ID der ExperimentSuite erstellt werden muss.

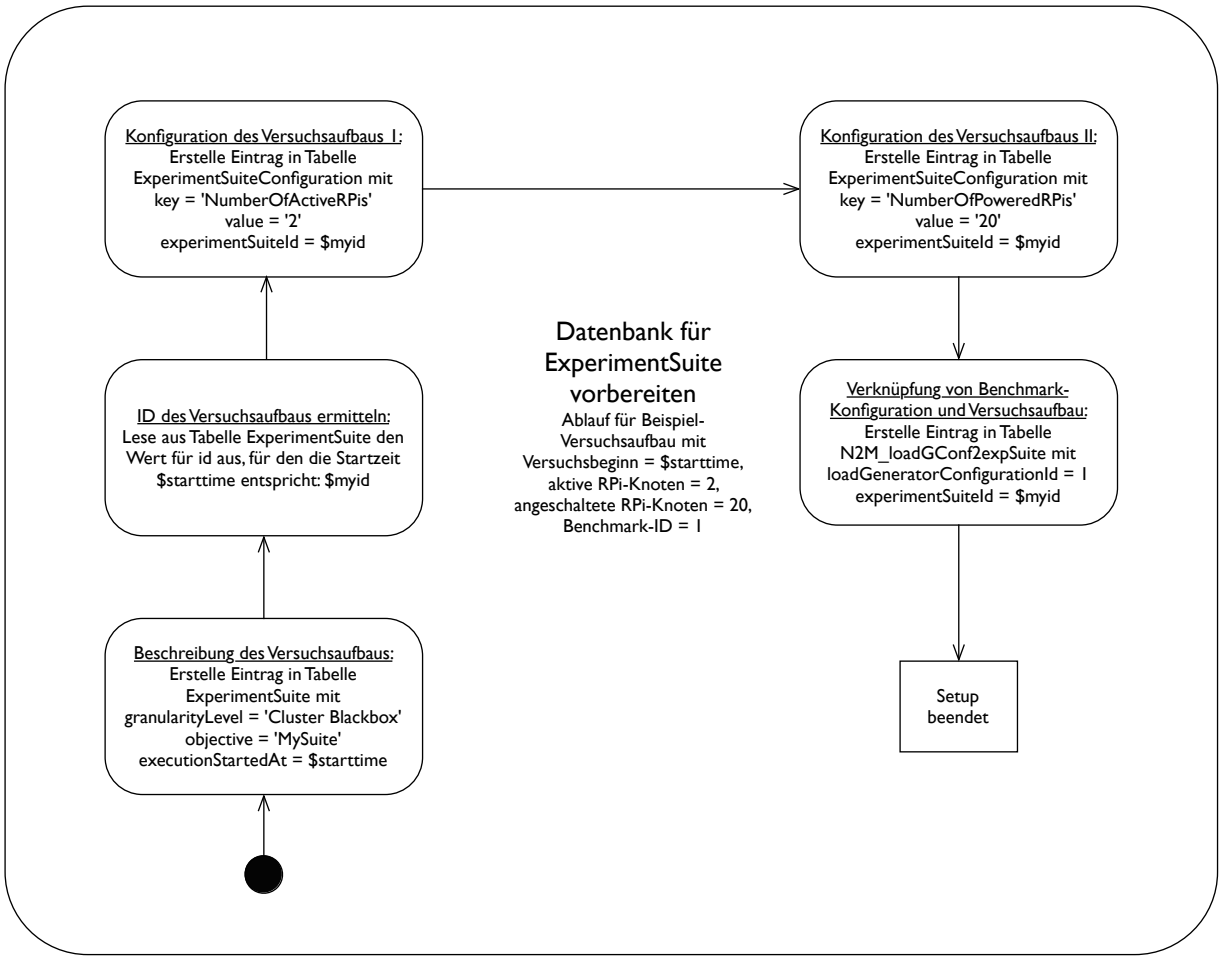

Abbildung 3.4: Aktivitätsdiagramm zur Vorbereitung der Datenbank für eine ExperimentSuite.

ii. Verteilte Ausführung des Benchmarks auf n RPi-Knoten.
Die verteilte Ausführung eines Benchmark-Programms wird mit dem Befehl

```
mpiexec -n -machinefile -wdir file
```

gestartet. Dabei spezifiziert `-n` die gewünschte Anzahl an parallelen Programmaufrufen, `-machinefile` den Pfad zum Machinefile, `-wdir` das Verzeichnis, in das zur Ausführung des Programms gewechselt werden soll, und `file` die auszuführende Datei (vgl. `http://www.mpich.org/static/docs/latest/www1/mpiexec.html`). Ein Beispielaufruf zur parallelen Ausführung

von HPLinpack auf vier RPi-Nodes mit der hier verwendeten Verzeichnisstruktur wäre also

```
mpiexec -n 4 -machinefile /srv/libraries/etc/mpich-3.0.4-shared
/machinefile -wdir /srv/benchmarks/bin
/hpl-2.1 /srv/benchmarks/bin/hpl-2.1/xhpl
```

iii. Schreiben der Ergebnisdaten.
Die Ausgabe des Benchmark-Programms wird in die vorbereitete Logdatei geschrieben. Falls das Programm in seiner Ausgabe keinen Zeitstempel vorsieht (z.B. STREAM), wird dieser ermittelt und der Ausgabe als zusätzliche Zeile hinzugefügt. Das ist wichtig, damit später für jeden Messwert ein Zeitstempel in die Datenbank eingegeben werden kann (vgl. Schritte d und e). Auch zum Abgleich der Messergebnisse der Benchmarks mit denen des Strommessgeräts ist ein Zeitstempel nötig.

c) Iteration über n RPi-Knoten:

i. Einrichten der Datenbank.
Die Datenbank wird wie in Schritt b i beschrieben vorbereitet. Der wesentliche Unterschied zu Schritt ist der Wert für die Anzahl angeschalteter RPi-Knoten. Es wird nicht mehr konstant 20 eingetragen, sondern n, da nicht mehr aktive RPi-Knoten heruntergefahren werden.

ii. Verteilte Ausführung des Benchmarks auf n RPi-Knoten.
Analog zu Schritt b ii.

iii. Herunterfahren von RPi-Knoten n.
Beim zweiten Lauf eines Benchmark-Programms auf n RPi-Knoten wird der nun nicht mehr benötigte bzw. aktive Knoten heruntergefahren. Das gilt nicht für den Ausführungsknoten `pi03`, der bis zum Schluss des Experiments zur Ausführung der Skripte und Eingabe der Messergebnisse in die Datenbank verwendet wird.

iv. Schreiben der Ergebnisdaten.
Analog zu Schritt b iii.

d) Parsen der Logdateien für die Datenbank-Eingabe.
Die Ausgabe der Benchmarks ist höchst unterschiedlich und entspricht natürlich nicht dem Eingabeformat für die Datenbank `rpiWerte`. Daher wird pro ausgeführtem Benchmark eine Eingabedatei für die Datenbank erstellt, die pro ExperimentSuite eine Zeile mit den Messergebnissen und Zeitstempel enthält, getrennt durch Leerzeichen. So enthält z.B. eine Zeile einer Datenbank-Eingabedatei für STREAM:

```
Copy: 3649.3 0.167595 0.166607 0.168846 Scale: 3428.0 0.178749
0.177361 0.180439 Add: 4749.8 0.192614 0.192010 0.193745 Triad:
4630.1 0.197780 0.196974 0.199436 Unixtime: 1396608095
```

Eine Beispielzeile einer Datenbank-Eingabedatei für HPLinpack:

```
WR00L2L2 29 1 2 2 0.09 1.965e-04 HPL_pdgesv() end time Wed Mar
26 15:39:05 2014
```

e) **Schreiben der Messergebnisse in die Datenbank.**
Falls die Zeitstempel in der Datenbank-Eingabedatei noch nicht dem Format des Unix-Zeitstempels UTC entsprechen (vergangene Sekunden seit dem 1. Januar 1970, vgl. `http://unixhelp.ed.ac.uk/CGI/man-cgi?date`) wie z.B. bei HPLinpack (vgl. Schritt d), werden sie in dieses Format konvertiert. Für jeden Messwert muss ein Eintrag in der Tabelle `MeasurementValue` mit Werten für Parameter, Messwert und Zeitstempel erstellt werden.

Zur Durchführung dieser Schritte wurden drei Shellskripte erstellt: `startBenchmarks.sh` zum Erstellen des Machinefile, Einhängen des geteilten Verzeichnisses und Aufruf der Skripte `STREAM.sh` und `hpl-2.1.sh` (Schritte 1–3). Schritt 4 wird für den jeweiligen Benchmark durch die Shellskripte `STREAM.sh` und `hpl-2.1.sh` ausgeführt, die von `startBenchmarks.sh` aufgerufen werden.

3.3 Strommessung

Die Messung des Stromverbrauchs erfolgt mit dem Strommessgerät Energenie EGM-PWM-LAN (vgl. `http://energenie.com/item.aspx?id=6736&lang=de`), im Folgenden als *Energenie* bezeichnet. Es wird zwischen die Steckdose und den Bramble gesteckt und über ein LAN-Kabel mit dem Netzwerk verbunden.

Für den verwendeten Versuchsaufbau war dem Energenie eine statische IP-Adresse zugewiesen und ein DNS-Eintrag erstellt worden. Damit ist es möglich, im lokalen Netzwerk auf das Gerät und die ermittelten Messwerte zuzugreifen. Für den Zugriff auf die Messergebnisse gibt es mehrere Möglichkeiten, darunter eine Webbrowser- und eine Windows-Anwendung. Damit die Software das Gerät erkennt, muss es sich im selben Subnetz befinden wie der zugreifende Rechner.

Die Software musste einmalig für die Benutzung des Energenie eingerichtet werden. Dazu mussten nach der Installation des Programms PowerManager IP-Adresse und ein Name für das Energenie eingetragen werden. In der Webbrowser-Anwendung wurde das Standardpasswort geändert und die NTP-Zeitsynchronisation aktiviert. Weitere Einstellungen wie die Angabe eines Strompreises können hier ebenfalls vorgenommen werden.

Das Energenie misst Spannung in Volt, elektrischen Strom in Ampère, Leistung in Watt und Arbeit in Kilowattstunden. Die Messgenauigkeit innerhalb des Messbereichs von $50W - -2500W$ beträgt $\pm 2\%$ bzw. $\pm 1W$.

Für den Versuchsaufbau erwies sich die Webbrowser-Anwendung als weniger geeignet, da sie hauptsächlich Konfigurationsfunktionen bietet. Zwar werden auf einer grafischen Oberfläche die jeweils aktuellen Werte für Spannung in Volt, elektrischen Strom in Ampère, Leistung in Watt und Arbeit in Kilowattstunden angezeigt, doch es können keine Messwerte für eine Zeitspanne abgerufen werden. Das Programm PowerManager bietet eine erweiterte Funktionalität, z.B. die Visualisierung der Leistung über eine Zeitspanne als Kurve.

Bei Verwendung von *PowerManager* werden die ermittelten Messwerte in der SQLite-Datenbank `database.sqlite` im Verzeichnis `C:\Program Data\PowerManagerDatabase` auf der zugreifenden Windows-Maschine abgelegt. Sie können auch als xls-Datei exportiert werden. Dabei werden pro Messung drei Werte angegeben: Beginn der Messung, Ende der Messung und Leistung in Watt. In der Datenbank werden für jede Messung zusätzliche Werte abgelegt wie Widerstand in Ohm und Frequenz in Hertz. Für die ExperimentSuites sind die Einträge `Stamp` (Zeitstempel als Integer-Wert) und `P` (Leistung in Watt als Double-Wert)

in der Tabelle `LogValues` entscheidend. Die Entscheidung für das Auslesen der SQLite-Datenbank gegenüber dem xls-Export fiel wegen der höheren Granularität der Zeitstempel in der Datenbank.

Das Aktivitätsdiagramm 3.5 zeigt, welche Schritte aus Benutzersicht pro ExperimentSuite für die Strommessung notwendig sind:

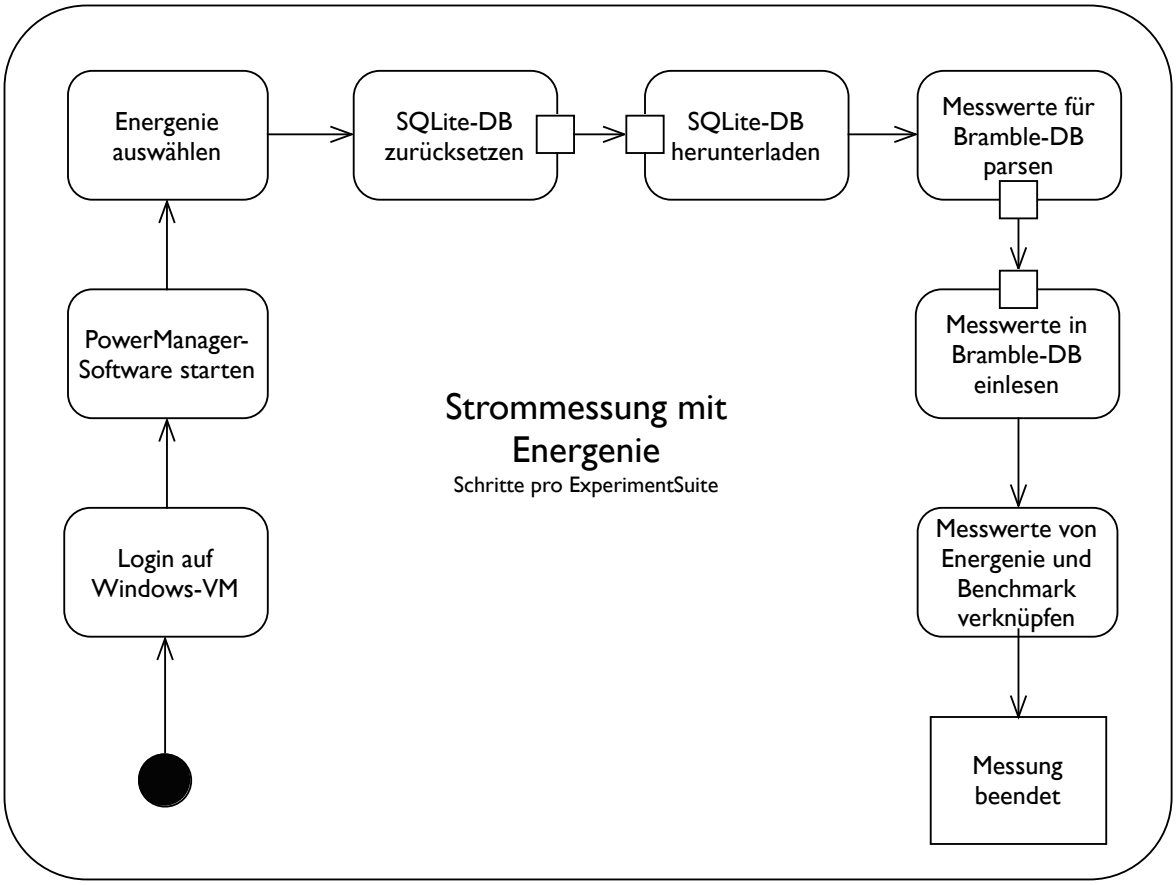

Abbildung 3.5: Aktivitätsdiagramm zur Durchführung einer Strommessung für eine ExperimentSuite.

1. **Login auf Windows-Maschine.** Das Programm PowerManager wird nur für ein Windows-Betriebssystem (Windows XP oder höher) angeboten. Daher war eine virtuelle Windows-Maschine (Windows 7) im lokalen Netzwerk zur Ausführung der PowerManager-Software zur Verfügung gestellt worden.
2. Starten der PowerManager-Software.
3. **Auswahl des Energenie.** Die Startoberfläche des Programms hat ein Menü mit den eingetragenen Geräten, aus denen der Name ausgewählt werden muss, der bei der erstmaligen Aktivierung eingetragen wurde.
4. **Zurücksetzen der SQLite-Datenbank.** Um möglichst wenige Messwerte zu erhalten, die nicht zur aktuellen ExperimentSuite gehören, ist es sinnvoll, die Datenbank

jeweils vor dem Start einer ExperimentSuite zurückzusetzen.

5. **Start der grafischen Aufzeichnung.** Optional ist es möglich, die aktuellen Messwerte für die Leistung in Watt in einer grafischen Oberfläche anzeigen zu lassen.

6. **Herunterladen der SQLite-Datenbank.** Zur weiteren Verarbeitung der Messwerte muss die SQLite-Datenbank von der Windows-Maschine heruntergeladen werden.

7. **Parsen der Messwerte für die Bramble-Datenbank.**

8. **Einlesen der Messwerte in die Bramble-Datenbank.**

9. **Verknüpfung der Messwerte von Energenie und Benchmark-Programm.**

3.4 Ergebnisse

Der folgende Abschnitt präsentiert die Untersuchungsergebnisse für HPLinpack und STREAM auf dem Bramble. Jeder Benchmark wurde zweimal ausgeführt. Bei Messreihe 1 waren alle RPi-Knoten angeschaltet, bei Messreihe 2 nur die, auf denen die Benchmark-Programme tatsächlich ausgeführt wurden. `pi03` als Berechnungsknoten blieb immer angeschaltet und wurde in der Untersuchung nicht berücksichtigt (vgl. Kap. 2.6.3).

Da keine Implementierung von Whetstone für eine verteilte Berechnung auf Raspbian existiert, war im Verlauf der Untersuchung vorgegeben worden, Whetstone nicht zu berücksichtigen. HPLinpack in der verwendeten Implementierung benötigt mindestens vier CPUs oder parallele Prozesse. Hier wurde jeder CPU, d.h. einem RPi-Knoten, genau ein Prozess zugewiesen, sodass zur Ausführung von HPLinpack mindestens vier angeschaltete und aktive RPi-Knoten benötigt werden.

Zur besseren Vergleichbarkeit wurde die verteilte Ausführung von STREAM daran angepasst, d.h. auch STREAM wurde auf mindestens vier RPi-Knoten parallel ausgeführt.

3.4.1 HPLinpack: Performance

HPLinpack in der verwendeten Implementierung (vgl. Kap. 2.4.1) liefert folgende Ausgabe:

Zunächst erfolgt eine Erläuterung der verwendeten Eingabe- und Ausgabeparameter. Ein Teil der Eingabeparameter ist invariabel (`eps` bzw. Maschinengenauigkeit und `T/V` bzw. Ausführungszeit). Die anderen Eingabeparameter sind in der Datei `HPL.dat` angegeben, die im selben Verzeichnis wie die ausführbare Datei des Benchmark-Programms liegen muss. Dazu zählen die Problemgröße bzw. Ordnung des zu lösenden linearen Gleichungssystems `N`, die Anzahl an verwendeten Problemgrößen `#N`, die Blockgröße`NB`[2] und die Größe des Prozessorennetzes (`P` und `Q`)[3] (vgl. `http://www.netlib.org/benchmark/hpl/algorithm.html`).

Die Ausgabeparameter sind invariabel. Die wichtigsten davon sind `Gflops` (Ausführungsrate in GFLOPs) und `Time` (Ausführungsdauer in Sekunden). Für jede Ausführung von

[2] Die Lösung des linearen Gleichungssystems erfolgt mittels L/U-Faktorisierung. Dazu wird zunächst eine $n \times n + 1$-Koeffizientenmatrix der Ausgangsmatrix A erzeugt. Diese wird zur weiteren Bearbeitung in Blöcke der Größe $NB \times NB$ aufgeteilt.

[3] Die Blöcke werden zur Bearbeitung einem Netz aus Prozessoren übergeben der Größe $P \times Q$ übergeben. `P` bezeichnet darin die Anzahl von Prozessoren in einer Spalte, `Q` die Anzahl von Prozessoren in einer Zeile des Netzes.

HPLinpack werden 864 Tests durchgeführt, d.h. 864 Messwerte für Ausführungsrate und Ausführungsdauer erzeugt. `Gflops` wird auf 8, `Time` auf 2 Nachkommastellen genau angegeben.

Die Diagramme 3.6 und 3.7 zeigen die Ergebnisse von Messreihe 1, jeweils skaliert auf 16–4 RPi-Knoten. Dabei sind jeweils n RPi-Knoten aktiv und alle 20 RPi-Knoten angeschaltet. Die Diagramme 3.8 und 3.9 stellen die Ergebnisse von Messreihe 1 denen von Messreihe

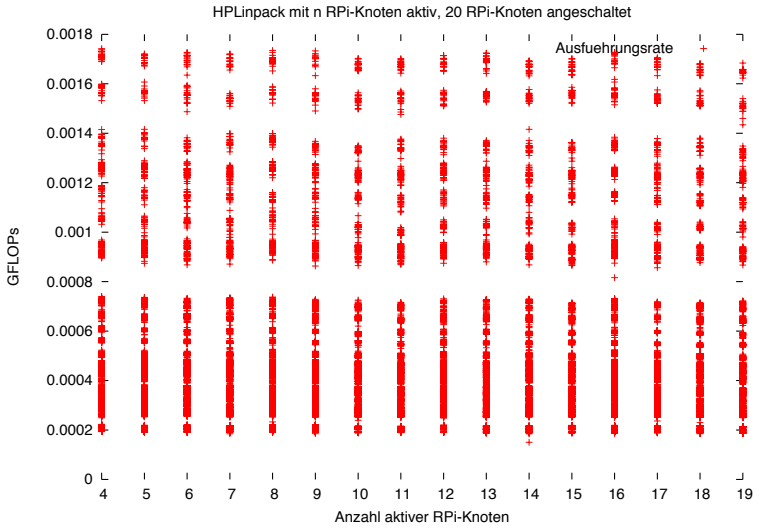

Abbildung 3.6: Ausführungsrate für HPLinpack, Messreihe 1.

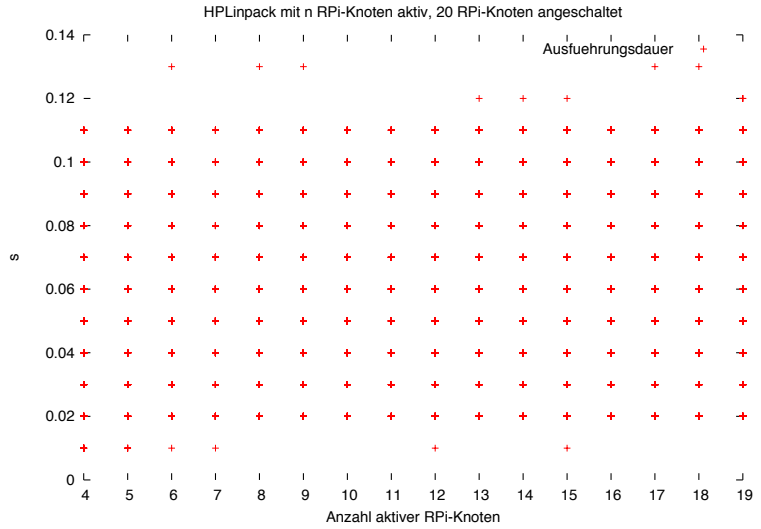

Abbildung 3.7: Ausführungsdauer für HPLinpack, Messreihe 1.

2 gegenüber, in der nicht mehr aktive RPi-Knoten nach der Ausführung des Benchmarks heruntergefahren werden. Die Ergebnisse von Messreihe 1 sind rot, die von Messreihe 2 blau dargestellt.

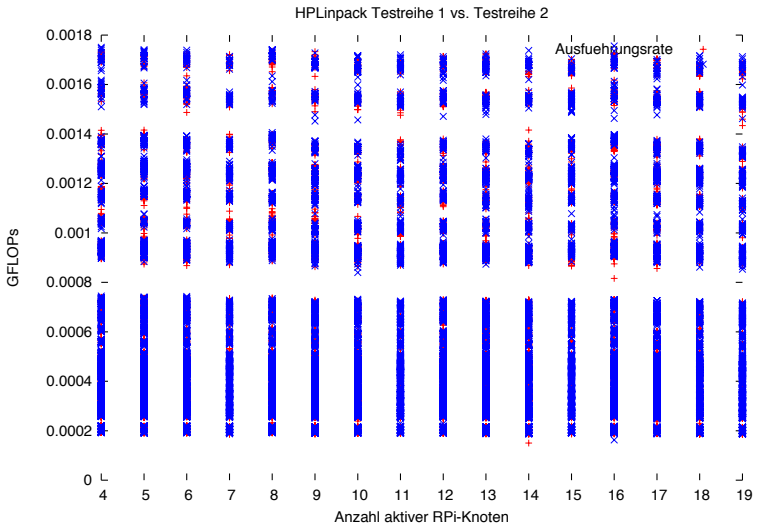

Abbildung 3.8: Ausführungsrate für HPLinpack, Messreihe 1 und Messreihe 2.

3.4.2 STREAM: Performance

In der Ausgabe von STREAM werden zunächst ebenfalls die verwendeten Parameter angegeben, darunter Anzahl an Prozessoren, Vektorlänge, Speicherbedarf und Anzahl an Iterationen für jedes Modul. Für die vier Module Copy, Scale, Add und Triad (vgl. Kap. 2.4.3) werden die beste erzielte Ausführungsrate in MB/s (`Rate`) und die durchschnittliche (`Avg time`), minimale (`Min time`) und maximale Ausführungsdauer (`Max time`) in Sekunden ausgegeben. Die Ausführungsrate wird auf eine, die Ausführungsdauer auf 6 Nachkommastellen genau angegeben.

Diagramme 3.10 und 3.11 zeigen die Ergebnisse von Messreihe 1 für STREAM. Ausgabeparameter sind Ausführungsrate in MB/s und durchschnittliche Ausführungsdauer in Sekunden für Copy, Scale, Add und Triad, skaliert auf 16–4 RPi-Knoten. Dabei sind jeweils n RPi-Knoten aktiv und alle 20 RPi-Knoten angeschaltet. Die Diagramme 3.12 und3.13 stellen die Ergebnisse von Messreihe 1 und Messreihe 2 für STREAM gegenüber. Auch hier sind die Ergebnisse von Messreihe 1 rot, die von Messreihe 2 blau markiert.

3.4.3 Stromverbrauch

Die Diagramme **??** und **??** zeigen die Ergebnisse der Strommessung für die Messreihen 1 und 2. Diagramm **??** zeigt den Energieverbrauch in Watt, wenn alle 20 RPi-Knoten angeschaltet sind. Diagramm **??** zeigt den Energieverbrauch, wenn nicht mehr aktive RPi-Knoten heruntergefahren werden.

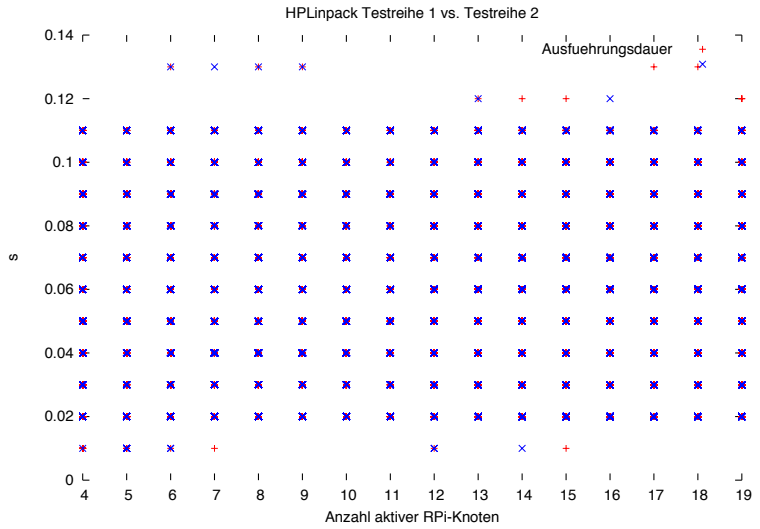

Abbildung 3.9: Ausführungsdauer für HPLinpack, Messreihe 1 und Messreihe 2.

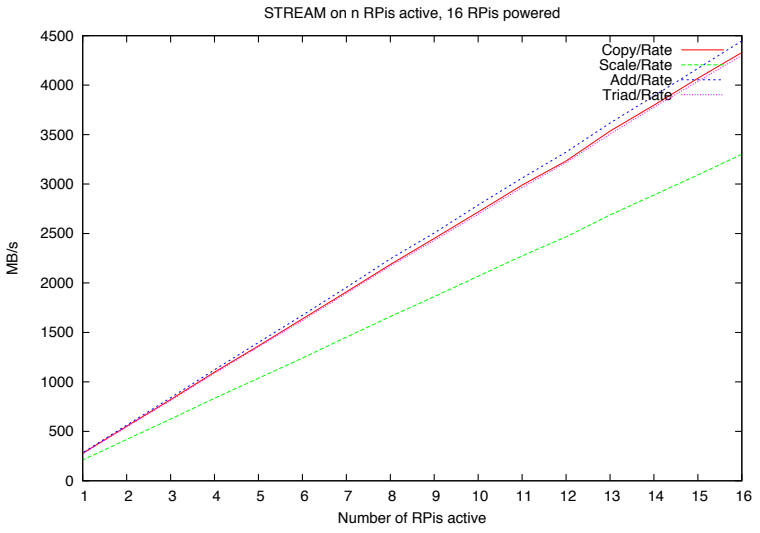

Abbildung 3.10: Ausführungsrate für STREAM, Messreihe 1.

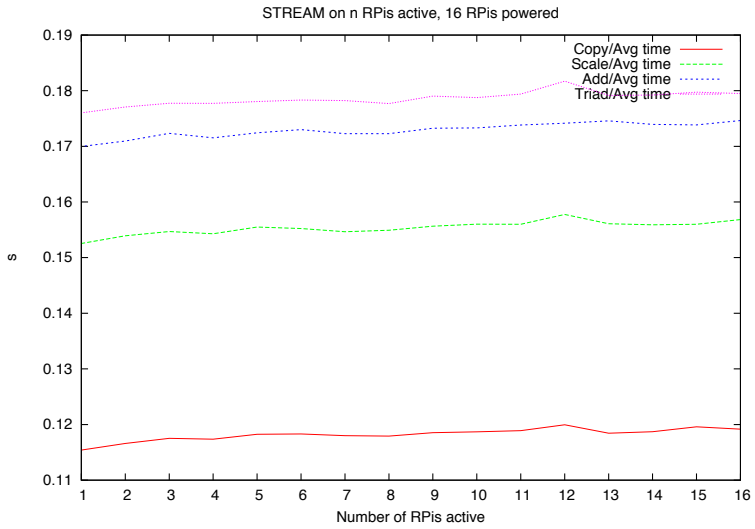

Abbildung 3.11: Ausführungsdauer für STREAM, Messreihe 1.

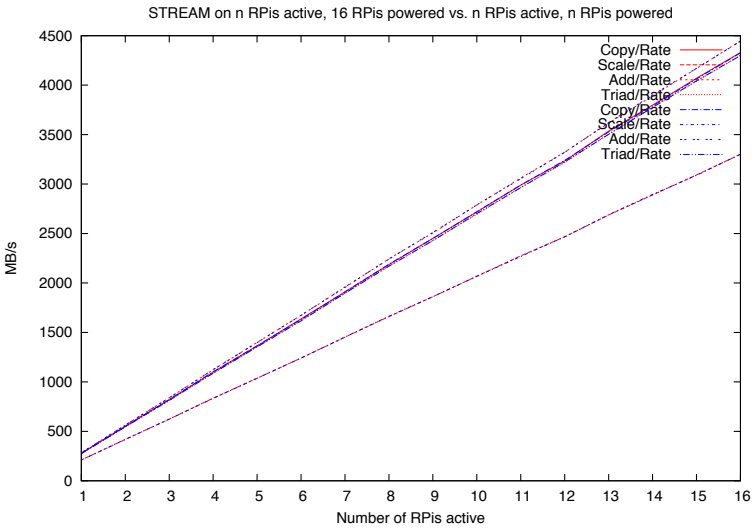

Abbildung 3.12: Ausführungsrate für STREAM, Messreihe 1 und Messreihe 2.

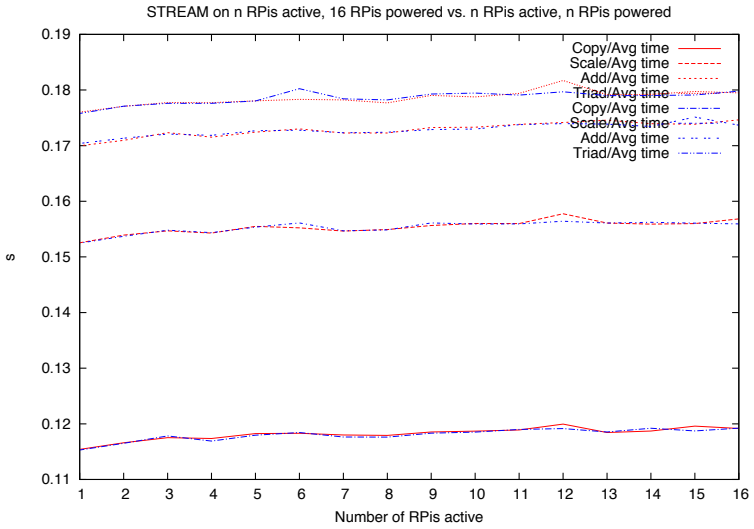

Abbildung 3.13: Ausführungsdauer für STREAM, Messreihe 1 und Messreihe 2.

4 Interpretation

Das folgende Kapitel dient der Bewertung und Einordnung der ermittelten Ergebnisse: Performance für HPLinpack und STREAM für Messreihe 1 gegenüber Messreihe 2 sowie Stromverbrauch in Messreihe 1 gegenüber Messreihe 2. Dabei ist das Skalierungsverhalten des Bramble bezüglich Performance und Stromverbrauch ausschlaggebend. Abschließend werden Grenzen des Versuchsaufbaus diskutiert.

4.1 HPLinpack: Performance

Der Versuchsaufbau und die Steuerung der verteilten Ausführung der Benchmark-Programme durch MPICH lassen vermuten, dass sich kein signifikanter Unterschied in der Performance zeigt, je nachdem, ob nicht mehr an der Programmausführung beteiligte RPi-Knoten heruntergefahren werden oder nicht. Wie in Kap. 3.2.2 dargestellt, ist im Machinefile genau festgelegt, welchen CPUs bzw. RPi-Knoten mit mpiexec eine die Ausführung des Benchmark-Programms zugewiesen wird und in welcher Reihenfolge sie für die Ausführung herangezogen werden.

Die Diagramme 3.8 und 3.9 bestätigen diese Erwartung: Messreihe 1 und Messreihe 2 werden in den Grafiken nahezu identisch dargestellt. Tabelle **??** stellt die minimalen, maximalen und durchschnittlichen Werte für Ausführungsrate und Ausführungsdauer beider Messreihen gegenüber:

Dabei überschreiten die Differenzen in keinem Fall die Größenordnung von , liegen also innerhalb des Messfehlers (vgl. Kap. 3.4.1). Somit lässt sich schlussfolgern, dass das Herunterfahren nicht mehr aktiver RPi-Knoten keine signifikanten Auswirkungen auf die Performance von HPLinpack auf dem Bramble hat. Das Skalierungsverhalten bezüglich der Performance ist somit für beide Messreihen gleich.

4.2 STREAM: Performance

Die Konzeption von STREAM lässt erwarten, dass sich das Hinzunehmen von Ressourcen, d.h. von RPi-Knoten, einen linearen Anstieg der Ausführungsrate zu Folge hat. Der Autor des Benchmarks schreibt hierzu:

> [...] unless something is very wrong, the performance of a cluster will be the performance of a node times the number of nodes (vgl. `http://www.cs.virginia.edu/stream/ref.html`).

Für die Ausführungsdauer werden hingegen annähernd gleiche Werte erwartet, wenn sich das System erwartungsgemäß verhält.

Die Diagramme 3.10 und 3.11 zeigen, dass sich der Bramble bei der skalierten Ausführung von STREAM auf 19 – 1 RPi-Knoten diesen Erwartungen gemäß verhält: Der Anstieg der

Ausführungsrate bei Hinzunahme von RPi-Knoten ist nahezu linear. Geringe Abweichungen zeigen sich bei der Ausführungszeit. Da die Ausführungsdauer auf zwei Stellen nach dem Komma genau gemessen wird und keine Abweichung eine Nachkommastelle überschreitet, liegen auch diese Abweichungen innerhalb des Messfehlers und können vernachlässigt werden. Das Skalierungsverhalten des Bramble bei der Ausführung von STREAM kann somit als erwartungsgemäß bezeichnet werden.

Es wird somit erwartet, dass auch das Herunterfahren nicht mehr aktiver RPi-Knoten keinen signifikanten Effekt auf die Performance des Bramble bei der Ausführung von STREAM haben wird. Die Diagramme 3.12 und 3.13 verdeutlichen dies: Die Ausführungsraten von STREAM in Messreihe 1 und Messreihe 2 sind nahezu identisch. Bei der durchschnittlichen Ausführungsdauer von Messreihe 1 gegenüber Messreihe 2 zeigen sich, wie schon innerhalb von Messreihe 1, leichte Unterschiede. Auch hier liegt die Abweichung von Messreihe 2 gegenüber Messreihe 1 bei maximal Sekunden. Sie ist also kleiner als der Messfehler und kann somit vernachlässigt werden.

Schlussfolgernd lässt sich festhalten, dass der Bramble bei der Ausführung von STREAM auch dann ein erwartungsgemäßes Skalierungsverhalten zeigt, wenn nicht aktive RPi-Knoten nach der Ausführung heruntergefahren werden.

4.3 Stromverbrauch

4.4 Grenzen des Versuchsaufbaus

Während der Versuchsdurchführung zeigte sich, dass die jetzige Architektur des Bramble in einigen Punkten an ihre Grenzen stößt. Bevor die Benchmarks bzw. die ExperimentSuites ausgeführt werden konnten, mussten einige Fehlerfälle untersucht und behoben werden:

1. **Defekte Hardware.**
   Einige Mini-USB-Kabel waren bereits zu Beginn defekt. Sie mussten durch funktionsfähige Kabel ersetzt werden.

2. **RPi-Knoten nicht erreichbar (`ping`).**
   Häufig reagierte ein einzelner RPi-Knoten nicht auf ein `ping` von einem anderen RPi-Knoten oder von `careme` aus (Fehlermeldung: `Destination Host Unreachable`), obwohl die Status-LED Netzwerkaktivität anzeigte. Als einzige Lösung erwies sich Ziehen und wieder Einstecken des Mini-USB-Kabels; war das nicht erfolgreich, musste der Vorgang mit dem Netzwerkkabel wiederholt werden (das Netzwerkkabel alleine reicht nicht aus). Nach einigen Minuten war der Zielknoten i.d.R. wieder mit `ping` erreichbar.

3. **RPi-Knoten nicht erreichbar (`ssh`).**
   Hier traten drei Fehlerfälle auf: Am häufigsten war die Fehlermeldung `No route to host` beim Versuch, von `careme` oder einem anderen RPi-Knoten aus eine SSH-Verbindung aufzubauen. Auch hier musste das Netzwerkkabel gezogen und wieder eingesteckt werden. Nach einigen Minuten war der Zielknoten i.d.R. wieder über `ssh` erreichbar.

   Ein weiterer Fehlerfall war ein überraschender Passwortprompt für `root` beim Versuch, eine SSH-Verbindung von einem RPi-Knoten aus zu einem anderen aufzubauen. Dieses Problem ließ sich durch Eintragen des RSA Public Key von `root` in die Datei `~/.ssh/authorized_keys` auf dem Zielknoten lösen.

Seltener trat die Fehlermeldung `WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!` auf. In diesem Fall musste der RSA Public Key des anfragenden RPi-Knotens in der Datei `~/.ssh/known_hosts` auf dem Zielknoten korrigiert werden.

4. **Geteiltes Verzeichnis nicht eingehängt.**
   Beim Neustart eines RPi-Knotens wurde häufig das geteilte Verzeichnis nicht eingehängt (Fehlermeldung z.B. `-bash: /srv/libraries/etc/.sharedprofile: No such file or directory`). Dann musste das Verzeichnis mit `mount /srv` auf dem betreffenden RPi-Knoten eingehängt werden.

5. **Bash-Befehle werden nicht erkannt.** Gelegentlich wurden auf einzelnen RPi-Knoten häufig verwendete Bash-Befehle nicht mehr erkannt (Fehlermeldung z.B. `mpiexec: command not found`). Als Lösung erwies sich ein Logout und erneuter Login auf dem betreffenden RPi-Knoten.

Der Ausschluss dieser Fehlerfälle wurde weitestgehend in die Vorbereitung einer ExperimentSuite integriert: Überprüfung der Netzverbindung aller RPi-Knoten mit `ping` vom Berechnungsknoten `pi03` aus, Einhängen des geteilten Verzeichnisses und damit gleichzeitig die Überprüfung der SSH-Verbindung zu allen RPi-Knoten (vgl. Kap. 3.2.2).

Spontan auftretende Fehler konnten nicht im Vorfeld ausgeschlossen werden. Die Durchführung einer ExperimentSuite erforderte daher immer die Anwesenheit einer Aufsichtsperson, um z.B. Kabel und Status-LEDs der RPi-Knoten zu überprüfen. Trat ein spontaner Fehler auf, musste die Durchführung der ExperimentSuite abgebrochen und von Neuem mit den Vorbereitungen und der Durchführung begonnen werden musste. Auch das Trennen nicht mehr aktiver RPi-Knoten vom Stromnetz musste manuell durch eine Aufsichtsperson erfolgen. Auch jeder Neustart eines RPi-Knoten musste von Hand veranlasst werden, da ein RPi nur durch Ziehen und erneutes Einstecken des Mini-USB-Kabels zum Hochfahren veranlasst werden kann.

Die physische Architektur des Bramble erwies sich in Teilen als Hindernis. Der Aufbau ist so beengt, dass die ständig benötigten Ethernet- und Mini-USB-Anschlüsse der RPi-Knoten `pi11` – `pi20` nur schwer zugänglich sind. Zudem sind manche Mini-USB-Kabel so kurz abgemessen, dass sie kaum bis zum entsprechenden Knoten reichen. Ein häufiges Trennen und wieder Einstecken von Mini-USB- und Netzwerkkabel, wie z.B. in den ExperimentSuites zum Messen des Stromverbrauchs einmal pro RPi-Knoten und Benchmark vorgesehen, wird dadurch erschwert.

Die vorliegenden Messergebnisse wurden durch diese Gegebenheiten nicht beeinflusst, da bei Störungen die ExperimentSuite abgebrochen und der Versuch von Neuem begonnen wurde. Für zukünftige Versuchsaufbauten erscheint eine entsprechende Anpassung der Cluster-Architektur sinnvoll (vgl. 5).

5 Zusammenfassung und Ausblick

> I have converted my Classic Benchmarks to run on the Linux based Raspberry Pi. These are Whetstone, [...], Linpack and Livermore Loops. [...] The Livermore Loops benchmark was used to accept the first supercomputer. So the main bragging rights are:
>
> In 1978, the Cray 1 supercomputer cost $7 Million, weighed 10,500 pounds and had a 115 kilowatt power supply. It was, by far, the fastest computer in the world. The Raspberry Pi costs around $70 (CPU board, case, power supply, SD card), weighs a few ounces, uses a 5 watt power supply and is more than 4.5 times faster than the Cray 1.
>
> My bragging rights are that I developed and ran benchmarks, including Whetstones, on Serial 1 Cray 1[1].

In der vorliegenden Arbeit wurde versucht, HPC-Benchmarks auf einem RPi-Cluster lauffähig zu machen und zu evaluieren. Dieser Proof of Concept war erfolgreich:

Die ausgewählten HPC-Benchmarks konnten mit Ausnahme von Whetstone nach dem Ausschluss vorhandener Störfaktoren auf dem Bramble ausgeführt werden. Die Experiment-Suite konnte mit kleinen Änderungen am bestehenden Datenbank-Schema in den übergeordneten Versuchsaufbau integeriert werden. Die erzielten Messwerte ergaben ein kohärentes Bild, wurden denen eines RPi-Einzelrechners gegenübergestellt sowie in den größeren Kontext der Top500-Rankings und der bisher erzielten Messwerte der Benchmark-Autoren eingeordnet.

Der Versuchsaufbau reiht sich damit Bestrebungen der letzten Monate, einen Beowulf-Cluster aus Raspberry Pi-Einzelrechnern für verteilte Berechnungen heranzuziehen. Wie Longbottom im obigen Zitat und die Projektleiter der Bramble-Projekte schreiben: Der Raspberry Pi ist in die Welt der Cluster und (zumindest ehemaligen) Supercomputer eingetreten[2] und erzielt in den betrachteten Kategorien CPU-Performance und Speicher-Bandbreite durchaus respektable Ergebnisse.

Schwierigkeiten zeigten sich vor allem in der vorhandenen Bramble-Infrastruktur, sowohl Hardware als auch IP/SSH-Kommunikation betreffend. Mögliche zukünftige Arbeiten umfassen daher zwei Felder:

Der vorhandene Bramble ist deutlich verbesserungsfähig. Vor allem die Hardware zeigt Schwächen, nicht nur die Stromversorgung betreffend. Der physische Aufbau ist extrem beengt, sodass Ziehen und erneutes Einstecken von Mini-USB- und Netzwerkkabeln nur eingeschränkt möglich ist und die Gefahr besteht, die übrige Hardware dabei zu beschädigen. Da die Unterbrechung der Stromversorgung die einzige Möglichkeit zum Reboot eines RPi

[1] Quelle: `http://www.raspberrypi.org/forum/viewtopic.php?f=31&t=44080`.
[2] Vgl. auch `http://www.cs.virginia.edu/stream/stream_mail/2012/0002.html`.

ist und eine unterbrochene Netzwerkverbindung nur durch Triggern des Netzwerkkabels wieder hergestellt werden kann, ist dieser Punkt systemkritisch für zukünftige Untersuchungen. Abhilfe ließe sich u.U. durch den Einbau von Reset-Knöpfen auf den einzelnen RPi-Nodes schaffen[3]. Auch die Platzierung der RPi-Nodes in einem aufrecht stehenden Rack statt eines liegenden Metallgehäuses wäre von Vorteil[4], damit die häufig benötigten Anschlüsse besser zugänglich sind.

In einem grösseren Rahmen wäre es interessant, früher oder später auf eine MPI-Implementierung von Whetstone für Raspbian zurückgreifen zu können. Auch die Evaluierung weiterer, bisher nicht betrachteter HPC-Benchmarks steht noch aus. Die wichtigsten Erkenntnisgewinne sind jedoch zu erwarten, wenn in naher Zukunft hoffentlich ein offener Treiber für die RPi-GPU vorliegt. Nachdem Broadcom vor wenigen Wochen erstmals die vollständige Spezifikation der GPU zur Verfügung gestellt hat[5], hat die Raspberry Pi Foundation einen entsprechenden Wettbewerb ausgeschrieben[6]. Die Ergebnisse und neuen Einsatzmöglichkeiten der bisher kaum anprogrammierbaren GPU, die deutlich leistungsfähiger ist als die hier schwerpunktmäßig untersuchte CPU, dürfen mit Spannung erwartet werden.

[3] Vgl. z.B. `http://raspi.tv/2012/making-a-reset-switch-for-your-rev-2-raspberry-pi`.
[4] Vgl. [Kie13] und [CCB+13].
[5] Vgl. `http://blog.broadcom.com/chip-design/android-for-all-broadcom-gives-developers-keys-\discretionary{-}{}{}to-the-videocore-kingdom/`.
[6] Vgl. `http://www.raspberrypi.org/competition-rules`.

Abbildungsverzeichnis

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

[BL08] Buhl, Hans ; Laartz, Jürgen: Warum Green IT nicht ausreicht – oder: Wo müssen wir heute anpacken, damit es uns übermorgen immer noch gut geht. In: *Wirtschaftsinformatik* (2008), S. 261–265. – http://dx.doi.org/10.1365/s11576-008-0058-5

[CCB+13] Cox, Simon ; Cox, James ; Boardman, Richard u. a.: 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://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

[FH12] Fichter, Klaus ; Hintermann, Ralph: Energieverbrauch und Energiekosten von Servern und Rechenzentren in Deutschland. Aktuelle Trends und Einsparpotenziale bis 2015 / Borderstep Institut für Innovation und Nachhaltigkeit. Berlin, Mai 2012. – Forschungsbericht. – http://www.bitkom.org/files/documents/Kurzstudie__Borderstep_l_Rechenzentren(1).pdf

[Kie13] Kiepert, Joshua: *Creating a Raspberry Pi-Based Bewoulf Cluster*. 2013. – 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. – http://www.nm.ifi.lmu.de/pub/Fopras

[Lan12] Langer, Alexander: *Raspberry Pi Handbuch*, Februar 2012. – http://raspberrycenter.de/handbuch/raspberry-pi-handbuch

[LB12] Lange, Walter ; Bogdan, Martin: *Entwurf und Synthese von Eingebetteten Systemen: Ein Lehrbuch*. München : Oldenbourg, 2012. – ISBN 3486718401

[LDK+05] Luszcek, Piotr ; Dongarra, Jack ; Koester, David u. a.: Introduction to the HPC Challenge Benchmark Suite / Lawrence Berkeley National Laboratory.

Berkeley, April 2005. – Forschungsbericht. – http://escholarship.org/uc/item/6sv079jp

[McC95] McCalpin, John: A Survey of Memory Bandwidth and Machine Balance in Current High Performance Computers. In: *IEEE TCCA Newsletter* (1995), S. 19–25. – http://www.cs.virginia.edu/~mccalpin/papers/balance

[McC05] McCalpin, John: *STREAM Benchmark*. 2005. – http://www.cs.virginia.edu/~mccalpin/STREAM_Benchmark_2005-01-25.pdf

[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

[Pre11] Prevezanos, Christoph: *Computer-Lexikon 2012*. München : Markt+Technik, 2011. – ISBN 9783827247285

[Rec06] Rechenberg, Peter: *Informatik-Handbuch*. München : Hanser, 2006. – ISBN 9783446401853

[Sch13] Schmidt, Maik: *Raspberry Pi. Einstieg, Optimierung, Projekte*. Heidelberg : dpunkt, 2013. – ISBN 9783864900327

[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