Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Fachstudie Nr. 113

Vergleich von Speichersystemen für RDF-Datenbanken auf mobilen Geräten

Dirk Coehne,
Gregor Rothmaier,
Christoph Schulz

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Bernhard Mitschang
Betreuer:	Dipl.-Inf. Andreas Brodt
begonnen am:	16. Juli 2009
beendet am:	12. Februar 2010
CR-Klassifikation:	H.3.4, C.5.3

Inhaltsverzeichnis

I. Fachstudie **7**

1. Einleitung **9**
1.1. Ausschreibung . . . 9
1.2. Vorbereitungen . . . 10
1.3. Arbeitsthesen . . . 11
1.4. Terminplan . . . 11

2. Grundlagen **13**
2.1. RDF . . . 13
2.2. Anfragesprachen . . . 14
2.2.1. SQL . . . 14
2.2.2. SPARQL . . . 14
2.2.3. Bedeutung für die Fachstudie . . . 14
2.3. Mobiles Gerät: Nokia N810 Internet Tablet . . . 15
2.3.1. Technische Details . . . 15
2.3.2. Software aus der Maemo-Community . . . 16
2.3.3. Maemo SDK und Scratchbox . . . 16

3. Vorstellung der Datenbanken und Benchmarks **19**
3.1. Datenbanken . . . 19
3.1.1. MonetDB . . . 19
3.1.2. MySQL . . . 20
3.1.3. PostgreSQL . . . 20
3.1.4. RDF-3X . . . 21
3.1.5. SQLite . . . 21
3.2. Benchmarks . . . 22
3.2.1. BSBM . . . 22
3.2.2. hdparm . . . 24
3.2.3. YAGO . . . 24
3.3. Messwerkzeuge . . . 25
3.3.1. sysstat . . . 25
3.3.2. timechart . . . 28
3.3.3. FineToothComb . . . 28

4. Durchführung und Bewertung der Benchmarks **31**

4.1. Allgemeines . . . 31
4.2. Speichermedien . . . 31
4.3. MySQL . . . 32
4.3.1. BSBM . . . 32
4.3.2. YAGO . . . 47
4.4. RDF-3X . . . 47
4.4.1. BSBM . . . 48
4.4.2. YAGO . . . 48
4.5. Auswertung . . . 53
4.5.1. MySQL mit BSBM . . . 54
4.5.2. RDF-3X mit YAGO . . . 54

5. Zusammenfassung und Ausblick **57**

II. Anhang **59**

6. Queries **61**
6.1. BSBM-Queries für SQL . . . 61
6.2. RDF-3X-Queries für YAGO . . . 63

7. Anleitungen **67**
7.1. Flashen eines N810 / Neuinstallation von OS2008 . . . 67
7.2. Fernzugriff . . . 68
7.3. Programme auf dem mobilen Gerät . . . 68
7.3.1. SSH-Server . . . 68
7.3.2. Uptime, Sleep . . . 69
7.3.3. Killall . . . 69
7.3.4. Dash . . . 70
7.4. Programme auf dem Host-System . . . 70
7.4.1. Maemo SDK und Scratchbox . . . 70
7.4.2. Maemo SDK Virtual Image und VirtualBox . . . 71
7.4.3. sshfs . . . 72
7.4.4. gnuplot . . . 73
7.4.5. LaTeX . . . 73

Literaturverzeichnis **75**

Abbildungsverzeichnis

2.1. Beispiel mehrerer RDF-Tripel . . . 13
2.2. Vernetzung mehrerer RDF-Relation . . . 14
2.3. Nokia N810 . . . 15

3.1. Gerastertes timechart Diagramm . . . 29
3.2. Call Stack von OpenStreetMap auf Symbian[Balo9] . . . 30

4.1. BSBM Durchläufe für MySQL, Scalefactor 250 . . . 33
4.2. CPU-/Speicherauslastung während BSBM für MySQL, Scalefactor 250 . . . 37
4.3. Page-Informationen während BSBM für MySQL, Scalefactor 250 . . . 38
4.4. IO während BSBM für MySQL, Scalefactor 250 . . . 39
4.5. BSBM Durchläufe für MySQL, Scalefactor 2785 . . . 40
4.6. CPU-/Speicherauslastung während BSBM für MySQL, Scalefactor 2785 . . . 44
4.7. Page-Informationen während BSBM für MySQL, Scalefactor 2785 . . . 45
4.8. IO während BSBM für MySQL, Scalefactor 2785 . . . 46
4.9. CPU-/Speicherauslastung während des Benchmarks für RDF-3X mit YAGO . . . 51
4.10. Page-Informationen während des Benchmarks für RDF-3X mit YAGO . . . 52
4.11. IO während des Benchmarks für RDF-3X mit YAGO . . . 53

Tabellenverzeichnis

4.1. Ergebnisse von hdparm . . . 31
4.2. Größe der Speichermedien . . . 31
4.3. Ergebnisse mit MySQL für BSBM . . . 32
4.4. BSBM Queryzusammenfassung für MySQL, Scalefactor 250 . . . 33
4.5. BSBM Query-Ergebnisse für MySQL, Scalefactor 250 (1) . . . 34
4.6. BSBM Query-Ergebnisse für MySQL, Scalefactor 250 (2) . . . 35
4.7. BSBM Query-Ergebnisse für MySQL, Scalefactor 250 (3) . . . 36
4.8. BSBM Queryzusammenfassung für MySQL, Scalefactor 2785 . . . 40
4.9. BSBM Query-Ergebnisse für MySQL, Scalefactor 2785 (1) . . . 41
4.10. BSBM Query-Ergebnisse für MySQL, Scalefactor 2785 (2) . . . 42

4.11. BSBM Query-Ergebnisse für MySQL, Scalefactor 2785 (3) 43
4.12. RDF-3X Ergebnisse für BSBM 48
4.13. Reduzierte Facts der Ontologie, welche in den RDF-3X-Queries verwendeten werden. 49
4.14. RDF-3X Ergebnisse für YAGO 50
4.15. Vergleich der Datenbanksysteme 54
4.16. Vergleich zwischen Dell D620 und N810 für RDF-3X mit YAGO 55

Verzeichnis der Listings

6.1. BSBM Query 1 61
6.2. BSBM Query 2 61
6.3. BSBM Query 3 61
6.4. BSBM Query 4 61
6.5. BSBM Query 5 61
6.6. BSBM Query 6 62
6.7. BSBM Query 7 62
6.8. BSBM Query 8 62
6.9. BSBM Query 9 62
6.10. BSBM Query 10 62
6.11. BSBM Query 11 62
6.12. BSBM Query 12 62
6.13. RDF-3X Query a1 63
6.14. RDF-3X Query a2 63
6.15. RDF-3X Query a3 63
6.16. RDF-3X Query b1 63
6.17. RDF-3X Query b2 64
6.18. RDF-3X Query b3 64
6.19. RDF-3X Query c1 64
6.20. RDF-3X Query c2 64
6.21. RDF-3X Query c3 64

Teil I.

Fachstudie

1. Einleitung

> Habe nun, ach! Philosophie,
> Juristerei und Medizin,
> Und leider auch Theologie
> Durchaus studiert, mit heißem
> Bemühn.
> Da steh ich nun, ich armer Tor!
> Und bin so klug als wie zuvor.
>
> *(Johann Wolfgang v. Goethe)*

Diese Fachstudie befasst sich mit dem Thema „Vergleich von Speichersystemen für RDF-Datenbanken auf mobilen Geräten“.

Das Speicherformat RDF ist unter anderem ein Grundstein für das sogenannte *semantic web*, weshalb es in letzter Zeit immer populärer wurde; allerdings sind Datenbanksysteme, die dieses Format beherrschen, noch weitestgehend unbekannt und daher auch unerforscht. Ziel dieser Arbeit ist es, die Leistungsfähigkeit von konventionellen, relationalen Datenbanksystemen mit RDF-Datebanksystemen auf mobilen Geräten zu vergleichen. Der Schwerpunkt wird auf der RDF-3X-Datenbank liegen.

1.1. Ausschreibung

RDF ist ein Datenformat für schemalose, strukturierte Daten, das unter anderem in den Bereichen des Semantic Web, der Naturwissenschaften, aber auch im Content-Management und in kontextbasierten Systemen an Bedeutung gewinnt. Um mit großen Mengen an RDF-Daten umzugehen, sind spezielle Speicherstrategien notwendig, die die Tripeldarstellung der in RDF modellierten Daten effizient verwaltet. Neue Ergebnisse der Datenbankforschung schlagen verschiedene Strategien zur Implementierung nativer RDF-Datenbanken vor. Die dort vorgenommenen Messungen wurden jedoch auf leistungsstarken Computern vorgenommen.
Diese Fachstudie soll untersuchen, wie sich die derzeitigen Ansätze zur nativen Speicherung von RDF-Tripeln auf mobilen Geräten verhalten, die vergleichsweise wenig Arbeitsspeicher und Flash-Speicher anstelle von herkömmlichen Festplatten beinhalten. Dazu sollen wichtige Ansätze auf ein linuxbasiertes mobiles Gerät (Nokia Internet Tablet) portiert und in Messungen verglichen werden.

Arbeitsprogramm

1. Sichtung der einschlägigen Literatur zum Thema RDF Store, Triple Store und ggf. Implementierung von Datenbanken.
2. Identifikation geeigneter Kandidaten für effiziente Speichersysteme für RDF-Datenbanken.
3. Einarbeitung in die Programmierumgebung des mobilen Geräts und in Benchmark-Tools.
4. Aufstellung der Testszenarien.
5. Durchführung der Messungen auf dem mobilen Gerät.
6. Verfassen eines Abschlussberichts und Präsentation der Ergebnisse.

Voraussetzungen

- Kenntnisse in Linux und C/C++ oder die Bereitschaft sich einzuarbeiten.
- Kenntnisse der Vorlesungen Grundlagen der Datenbanken und Informationssysteme und Implementierung von Datenbanken und Informationssystemen sind von Vorteil.

1.2. Vorbereitungen

Zur Vorbereitung für diese Fachstudie wurden wissenschaftliche Veröffentlichungen gelesen, die Entwicklungsumgebung zu Maemo studiert und Arbeitsthesen aufgestellt.

Für das Nokia N810 existierten im Zeitraum der Fachstudie kaum fertige Softwarepakete, da es sich um ein mobiles Gerät mit einer speziellen Architektur handelt. Entsprechend musste untersucht werden welche Software portiert werden muss.

Wissenschaftliche Veröffentlichungen

- RDF Support in the Virtuoso DBMS[EM]
- The Berlin SPARQL Benchmark[BS09]
- Column-Store Support for RDF Data Management: not all swans are white[SGK+08]
- Scalable Join Processing on Very Large RDF Graphs[NW09b]
- RDF-3X: a RISCstyle Engine for RDF[NW08]
- Scalable Semantic Web Data Management Using Vertical Partitioning[AMMH07]
- The RDF-3X Engine for Scalable Management of RDF Data[NW09a]

1.3. Arbeitsthesen

Folgende Überleglegungen wurden vor der eigentlichen Arbeit angestellt, um abzuschätzen mit welchen Ergebnissen gerechnet werden kann.

Das RDF-Format ist langsam(er) RDF ist von der Idee her ein einfaches, dafür sehr flexibles Datenformat. Wie in Abschnitt 2.1 dargestellt, sind alle Datensätze im Format *Subjekt - Prädikat - Objekt* gespeichert, was es ermöglicht viele Beziehungen zwischen verschiedenen Objekten zu ermitteln. Bei sehr großen Datenbanken und einer naiven Implementierung müssten alle Datensätze auf Beziehungen untersucht werden. Indem auf Indices operiert wird, lässt sich die Menge der zu durchsuchenden Daten reduzieren. Das Zusammenstellen eines Ergebnisses bei einem so flexiblen Datenformat könnte zu Leistungsengpässen führen. Herkömmliche Datenbanksysteme könnten, im Vergleich zu RDF-Datenbanksystemen, auch besser abschneiden, da sie ausgereifter sind.

Flash-Speicher verhalten sich anders als herkömmliche Festplatten Flash-Speicher verhalten sich aufgrund ihrer Funktionsweise anders. Das Beschreiben von leeren Medien ist sehr schnell; das Ändern und Löschen von Daten dauert jedoch im Verhältnis um einiges länger als bei herkömmlichen Festplatten.
Diese Eigenschaft wird bei den hier verwendeten Datenbanksystemen nicht gezeigt, denn das zu vergleichende RDF-3X-Datenbanksystem unterstützt keine Änderungen und kein Löschen.

Probleme mit der Entwicklungsumgebung Schon am Anfang der Fachstudie zeigten sich einige Softwarefehler in der Scratchbox, wodurch die Erwartungshaltung an die gesamte Maemo Plattform stark vorgeprägt wurde.

1.4. Terminplan

Für den Zeitraum der Fachstudie wurde folgender Terminplan aufgestellt. Die Überlegung war frühestmöglich fertig zu werden, um bei Verzögerungen Engpässe zu vermeiden.

Meilenstein 0 Ende: 26.06.2009
geschätzter Gesamtaufwand: 49,0 Stunden, entspricht 10%

- Einrichtung der kompletten Arbeitsumgebung
  - Beantragung und Einrichtung eines SVN-Repository
  - Einrichtung des Ticketsystems Trac
  - Installation der Maemo SDK

- Standard Diplom-LaTeX-Vorlage für eine Fachstudie anpassen
- Bericht-Struktur anlegen

Meilenstein 1 Ende: 17.07.2009
geschätzter Gesamtaufwand: 88,2 Stunden, entspricht 18%

- Literatur-Recherche, Lesen von Papers, Einarbeitung in SPARQL und SQL
- Einarbeitung in die Werkzeuge, die Benchmark-Software und die Datenbanksysteme
- Auswahl der Software zu der Benchmarks erstellt werden sollen

Meilenstein 2 Ende: 14.08.2009
geschätzter Gesamtaufwand: 93,1 Stunden, entspricht 19%

- Software „zum Laufen“ bringen, Portierung auf mobiles Gerät

Meilenstein 3 (Hauptteil) Ende: 25.09.2009 [eventuell mit Prüfungszeitraum]
geschätzter Gesamtaufwand: 151,9 Stunden, entspricht 31%

- Bewertungskatalog erstellen
- Benchmarks durchführen
- Ergebnisse interpretieren und bewerten

Meilenstein 4 Ende: 30.10.2009
geschätzter Gesamtaufwand: 88,2 Stunden, entspricht 18%

- Bericht erstellen
- Präsentation erstellen

2. Grundlagen

> Was gesagt werden kann, kann genau gesagt werden. Worüber man nicht sprechen kann, darüber soll man schweigen.
>
> *(Ludwig Wittgenstein)*

In diesem Kapitel werden die Grundlagen dieser Fachstudie beschrieben. Hier wird das RDF-Format, die verwendeten Abfragesprachen und das Mobile Gerät beschrieben.

2.1. RDF

RDF (**R**esource **D**escription **F**ramework) ist ein W3C-Standard[w3c] zur Speicherung von Daten. Dieses Format wurde ursprünglich entwickelt um Web-Inhalte zu annotieren, die nicht nur von Menschen, sondern auch von Computern verstanden werden können. In diesem Zusammenhang wird oftmals vom Begriff des *semantic web* gesprochen.

In RDF werden Ressourcen als Tripple aus Subjekt, Prädikat und Objekt beschrieben. Dadurch können Relationen zwischen verschiedenen Objekten ausgedrückt werden. Auf diese Weise kann eine ganze (Wissens-)Datenbank erstellt werden.

Beispiel: Durch die folgenden drei Tripple *Teller is_on Tisch*; *Glas is_on Tisch* und *Tisch is_in Raum* lassen sich Gegenstände in einem Raum beschreiben.

Über die Prädikate *is_on* und *is_in*, lassen sich Relationen zwischen den Gegenständen herstellen - beispielsweise um herauszufinden, was sich alles in diesem Raum befindet.

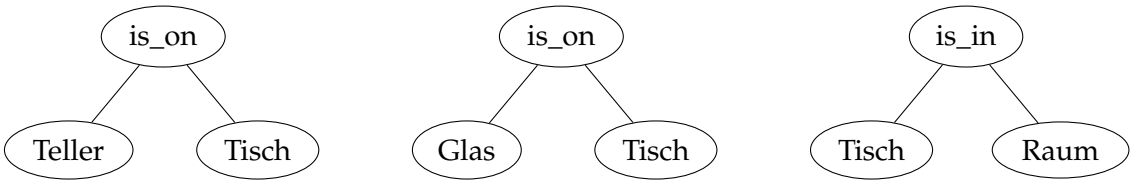

Abbildung 2.1.: Beispiel mehrerer RDF-Tripel

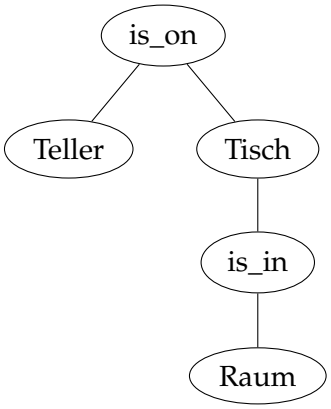

Abbildung 2.2.: Vernetzung mehrerer RDF-Relation

2.2. Anfragesprachen

In diesem Abschnitt werden die verschiedenen Anfragesprachen beschrieben, mit denen im Zuge dieser Fachstudie gearbeitet wurde.

2.2.1. SQL

SQL (**S**equential **Q**uery **L**anguage) ist eine sehr verbreitete Abfragesprache für relationale Datenbanksysteme. SQL ist eine deskriptive Sprache, die auf Operationen, welche auf Tabellen angewendet werden, ausgelegt ist. Die Reihenfolge der Operationen, also das „Wie“, ist dem Datenbanksystem überlassen. Im Gegensatz zu SPARQL ist SQL nicht für RDF-Datenbanksysteme ausgelegt.

2.2.2. SPARQL

SPARQL ist ein rekursives Akronym und steht für **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage. SPARQL ist ebenfalls eine deskriptive Sprache, die allerdings für RDF-Datenbanksysteme ausgelegt ist: die Anfrage bezieht sich nicht auf eine Tabelle sondern auf einen Graphen.

2.2.3. Bedeutung für die Fachstudie

Uns ist leider nicht bekannt, ob die Anfragesprache einen Einfluss auf die Messergebnisse ausübt. Prinzipiell muss jede Anfragesprache syntaktisch und semantisch analysiert werden, um anschließend in Datenbankoperationen transformiert zu werden. Für die Leistungsfähigkeit des Datenbanksystems spielt es sicherlich eine Rolle, wie leicht die jeweilige Sprache

in optimierte Datenbankoperationen transformierbar ist; allerdings lässt sich dieser Aufwand nicht ohne Weiteres messen, weshalb er bei beiden Sprachen als ähnlich angenommen wird.

2.3. Mobiles Gerät: Nokia N810 Internet Tablet

Abbildung 2.3.: Nokia N810

Als Muster für ein mobiles Gerät wurde von der Abteilung IPVS ein `Nokia N810 Internet Tablet` für die Dauer der Fachstudie bereitgestellt.

2.3.1. Technische Details

Nachfolgend werden die technischen Merkmale des Nokia N810 beschrieben:

Hersteller Nokia

CPU-Typ TI-OMAP 2420

CPU-Architektur ARMv6 (aus der ARM11-Familie)

Taktfrequent 400 MHz

Arbeitsspeicher 128 MB DDR-RAM

Interner Flash-Speicher 256 MB+2 GB (System + Nutzerdaten)

Externer Flash-Speicher Mini-SD-Card 4 GB

Speicherslots 1 MiniSDHC (extern)

Bluetooth Version 2.0

USB Micro-USB 2.0 (erreichbar durch Aufklappen des integrierten Ständers)

WLAN IEEE 802.11b/g

Länge 128 mm

Breite 72 mm

Höhe 14 mm

Akkulaufzeit 4 h (10 d Standby)

Betriebssystem OS2008 (Maemo Chinook 4.0 / Diablo 4.1.2)

Kernel-Version `2.6.21-omap1 #2 Tue Oct 14. 11:25:25 EST 2008`

2.3.2. Software aus der Maemo-Community

Nokia verwendet für einige seiner Geräte Software aus der Maemo-Community (`http://www.maemo.org`).

Das Betriebssystem *Maemo Diablo 4.1.2*[maeb] ist eine auf Debian basierende Linux-Distribution. Viele Linux-Werkzeuge und Befehle sind, mit gewissen Einschränkungen, auch auf Maemo verwendbar. Es stehen, wie bei Debian üblich, diverse vorkompilierte Softwarepakete zur Verfügung.

Die Nachfolgeversion *Maemo 5.0 (Freemantle)*[maea] basiert weitestgehend auf Qt. Allerdings ist diese Version einerseits noch im Beta-Stadium; andererseits ist sie inkompatibel mit dem N810, weshalb sie in dieser Studie nicht verwendet wird.

2.3.3. Maemo SDK und Scratchbox

Maemo SDK Zur Entwicklung von Software bietet Nokia das Maemo SDK 4 an. Dazu gehört auch die Scratchbox. Beides ist ausschließlich unter Linux lauffähig. Auf der Homepage finden sich hierzu die benötigten Links und Ressourcen (Hinweise zur Installation befinden sich im Anhang unter 7.4.1).

Scratchbox Die Scratchbox ist eine Cross-Compiler-Umgebung. In ihr ist es möglich Quellcode für die ARMv6-Archtiktur zu kompilieren, beziehungsweise vorhandene Debian-Pakete auf diese Architektur zu portieren.

Hinweis: Da die Scratchbox sich stark in das Gastgeber-System integriert und nicht für Windows verfügbar ist, gibt es auch ein Image für eine virtuelle Maschine. Das Image enthält ein vorbereitetes Ubuntu mit allen benötigten Programmen und Abhängigkeiten. Informationen darüber befinden sich im Anhang in Abschnitt 7.4.2.

3. Vorstellung der Datenbanken und Benchmarks

In diesem Kapitel wird die Auswahl der Datenbanken und Benchmarks beschrieben.

3.1. Datenbanken

Aufgrund nicht zur Verfügung stehender kommerzieller Produkte wurde die Auswahl ausschließlich auf Open Source Produkte begrenzt: eine Anfrage bei ORACLE Deutschland, ob Oracle Database für Maemo oder ARMv6-Prozessoren erhältlich sei, wurde mit aller Höflichkeit abgewiesen. Weitere wichtige Auswahlkriterien waren die Verbreitung der Produkte sowie der Grad der Spezialisierung auf RDF.

Für jedes Datenbanksystem folgt eine kurze Beschreibung mit Informationen über den Hersteller, Versionen, Lizenz sowie Gründe für und gegen die Auswahl.

3.1.1. MonetDB

MonetDB ist ein spaltenorientiertes Datenbanksystem, welches am niederländischen *Centrum Wiskunde & Informatica* (CWI) entwickelt wurde.

Bezugsquelle `http://monetdb.cwi.nl/`

Lizenz ähnlich der Mozilla Public License Version 1.1

Version MonetDB 5

Wieso? Mit MonetDB erhält man ein hoch skalierendes, Open Source Datenbankmanagementsystem. Open Source ist für das Cross-Compiling wichtig. MonetDB besitzt einige Optimierungen für RDF und es wird seitens der Entwickler auch in diese Richtung geforscht. Zudem kann auf MonetDB über TCP/IP Sockets zugegriffen werden, was eine leichte Anbindung an die verwendeten Benchmarks erlaubt und die Webseite verspricht eine leichte Portierbarkeit auf andere Systeme.

Wieso nicht? Aufgrund von Verzögerungen, resultierend aus den Problemen mit anderen Datenbanksystemen, hätte die Durchführung eines weiteren Benchmarks den Zeitrahmen der Fachstudie bei weitem gesprengt.

3.1.2. MySQL

MySQL ist ein sehr verbreitetes Datenbanksystem, das gerade im Online-Bereich gerne genutzt wird.

Bezugsquellen

- `http://www.mysql.de/`
- `http://maemo.org/downloads/OS2008/database/` (Maemo-Paket)

Lizenzen GPL oder eine proprietäre Lizenz mit kommerziellem Support

Version Die verwendete Version ist MySQL 4.1.22-1, obwohl eine Version 5 von MySQL existiert; allerdings kommt es laut Maemo-Homepage bei Version 5 zu Schwierigkeiten, da einige der benötigten Pakete für Maemo nicht existieren.

Wieso? Für MySQL existieren bereits viele Benchmarks für verschiedene Plattformen. Deswegen können die ermittelten Daten dieser Fachstudie mit Daten anderer Benchmarks verglichen werden. MySQL bietet eine Client/Server Architektur und ist über TCP/IP Sockets erreichbar. Damit erfüllt es auch die Voraussetzungen für einfach durchzuführende Benchmarks. Des Weiteren handelt es sich um ein zeilenorientiertes Datenbanksystem im Kontrast zu MonetDB.

3.1.3. PostgreSQL

PostgreSQL ist ein sehr bekanntes Open-Source Datenbanksystem, wenn auch nicht so verbreitet wie MySQL.

Bezugsquellen `http://www.postgresql.org/`

Lizenz BSD-Lizenz

Version PostgreSQL 8.4.0

Wieso? PostgreSQL bietet einige Funktionen, die MySQL erst in Version 5.x enthält oder dort vollständig fehlen. Beispielsweise unterstützt MySQL erst in Version 5.1 Randbedingungen für Foreign Keys - allerdings wird der Anwendungsfall „kaskadierendes Löschen“ im Rahmen dieser Fachstudie nicht betrachtet.

Wieso nicht? Weil mit MySQL bereits ein beliebtes, zeilenorientiertes Datenbanksystem abgedeckt ist und reichlich Vergleichsmaterial zu diesen beiden Datenbanksystemen existiert, wurde aus zeitlichen Gründen auf PostgreSQL verzichtet.

3.1.4. RDF-3X

Die RDF-3X-Datenbank (**R**esource **D**escription **F**ramework for **T**riple e**X**press) ist eine Triplestore-Datenbank, welche auf der Struktur von RDF arbeitet.

Bezugsquellen `http://www.mpi-inf.mpg.de/~neumann/rdf3x/`

Lizenz Creative Commons Attribution-Noncommercial-Share Alike

Version RDF-3X 0.3.3

Wieso? RDF-3X ist ein von Thomas Neumann entwickeltes Datenbanksystem. Es ist speziell für RDF von Grund auf entwickelt worden. Seine besonderen Eigenschaften sind die Optimierung der Indexierung und Speicherung von RDF Tripel, sowie einen Query Processor, der Merge Joins besonders gut einsetzt und einen Query Optimierer, welcher die Join Reihenfolge anhand eines Kostenmodells optimiert.

Es handelt sich bei RDF-3X um ein experimentelles Datenbanksystem, das sich noch in der Entwicklung befindet. So werden einige grundlegende Datenbankfunktionen nicht unterstützt (wie z.B. das nachträgliche Einfügen von Daten). Es vertritt zudem die RISC (**R**educed **I**nstruction **S**et **C**omputer) Philosophie, bei der versucht wird auf komplexe Befehle, bzw. Strukturen zu verzichten, um eine höhere Leistung zu erreichen.

RDF-3X, als der wohl schnellste Vertreter in Sachen RDF, stellt das grundlegende (und geforderte) Datenbanksystem dieser Fachstudie dar und dient als Referenz. Aufgrund des experimentellen Status, der nicht kompletten Umsetzung von SPARQL und dem Fehlen eines Client/Server Designs, ist RDF-3X nicht ohne Weiteres kompatibel zu BSBM.

3.1.5. SQLite

„SQLite ist das wohl meistgenutzte Datenbanksystem“ (Behauptung der Entwickler). Es ist ein sehr einfaches und vor allem bei C-Programmierern beliebtes Datenbanksystem.

Bezugsquellen `http://www.sqlite.org/`

Lizenz gemeinfrei

Version SQLite 3.6.19

Wieso? SQLite wird üblicherweise als einzelne C-Datei mit in ein Projekt kompiliert, was die Datenbank sehr schlank macht und das Datenbanksystem sehr klein erscheinen lässt. Die Quellen sind in der Public Domain und daher für alle Zwecke frei nutzbar.

Wieso nicht? SQLite besitzt keine Client/Server Architektur, was einen deutlichen Mehraufwand bedeutet hätte, besonders bei BSBM. Außerdem existieren bereits viele Vergleiche zwischen MySQL und SQLite.

3.2. Benchmarks

Der Begriff Benchmark (englisch für „Maßstab“) bezeichnet ein Verfahren um unterschiedliche Systeme, mit Hilfe von Referenzwerten nach festgelegten Kriterien, zu vergleichen. Um eine Vergleichbarkeit zu anderen Plattformen und Benchmark-Ergebnissen zu gewährleisten wurde in dieser Fachstudie auf das Erstellen und Rechtfertigen von eigenen Datenbankanfragen verzichtet. Folgende Eigenschaften sind eine notwendige Voraussetzung für ein erfolgreiches Benchmark:

- **Reproduzierbarkeit**: gleiche Vorbedingungen müssen zu gleichen, von Messfehlern bereinigten, Ergebnissen führen.
- **Aussagekraft:** jeder Messwert muss eine sinnvolle Interpretation haben, die in Zusammenhang mit dem Systemverhalten gebracht werden kann.

Für jedes verwendete Benchmark folgt eine kurze Beschreibung mit Informationen über den Aufbau und Messwerte des Benchmarks sowie Gründe für die Auswahl.

3.2.1. BSBM

Das ***B**erlin **S**PARQL **B**ench**m**ark*[BS09] (BSBM) hat es sich auf die Fahne geschrieben Leistungsdaten von Speichersystemen, die einen SPARQL-Endpunkt[w3-08] implementieren, zu messen. Das BSBM Benchmark bildet einen E-Commerce Anwendungsfall nach, in dem eine Menge von Produkten von verschiedenen Anbietern offeriert und mit Kundenrezessionen versehen wurden. Der Datensatz wird durch ein Generator-Werkzeug erzeugt und kann skaliert werden. Der Benchmark-Treiber bildet *Query Mixes* aus Datenbankanfragen für die Messung. Jede Datenbankanfrage simuliert einen Bestandteil der Suche eines Kunden

nach einem Produkt. Das Benchmark hat eine warm-up-Phase, falls das Speichersystem entsprechende Optimierungen beherrscht, und beugt so Messfehlern vor. BSBM wurde ausgewählt um einen Vergleich zu YAGO zu haben und RDF-3X auf Praxistauglichkeit zu untersuchen.

Average Query Execution Time	
Skala:	Sekunden
Skalenniveau:	Proportionalskala
Beschreibung:	Durchschnittliche Zeit, die eine mehrfach ausgeführte Anfrage vom Typ X mit verschiedenen Parametern benötigt hat.
Kommentar:	Sowohl das arithmetische als auch das geometrische Mittel wird bestimmt.

Queries per Second	
Skala:	Anfragen pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Durchschnittliche Anzahl an Anfragen des Typ X, die pro Sekunde ausgeführt wurden.

Min/Max Query Execution Time	
Skala:	Sekunden
Skalenniveau:	Proportionalskala
Beschreibung:	Obere und untere Grenze der benötigen Zeit für eine Anfrage des Typ X.

Queries Mixes per Hour	
Skala:	Query Mixes pro Stunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl der Query-Mixes, die pro Stunde ausgeführt werden können.

Overall Runtime	
Skala:	Sekunden
Skalenniveau:	Proportionalskala
Beschreibung:	Gesamte Zeit, die das Benchmark lief.

Composite Query Execution Time	
Skala:	Sekunden
Skalenniveau:	Proportionalskala
Beschreibung:	Durchschnittliche Zeit, die ein mehrfach ausgeführter Query Mix mit verschiedenen Parametern benötigt hat.

Average Query Execution Time over all Queries	
Skala:	Sekunden
Skalenniveau:	Proportionalskala
Beschreibung:	Durchschnittliche Zeit, die für die Durchführung von N Query Mixes bezogen auf die Gesamtzahl der Anfragen benötigt wird.

Price/Performance Metric	
Skala:	Dollar durch Query Mixes pro Stunde
Skalenniveau:	Proportionalskala
Beschreibung:	Preis-Leistungs-Verhältnis beziehungsweise die Kosten in Dollar innerhalb von 5 Jahren (siehe *TPC Pricing Specification*[tpc]).

3.2.2. hdparm

hdparm[hdp] ist ein Kommandozeilen-Programm zum Auslesen und Festlegen von Parametern einer Festplatte, mit dessen Hilfe auch Leistungsdaten ermittelt werden können. Bei diesem Benchmark handelt es sich nicht um ein Datenbankbenchmark, sondern um ein allgemeines System- und Festplattenbenchmark.

Cached reads	
Skala:	Megabyte pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Gibt die Geschwindigkeit, ohne Zugriff auf die Festplatte, an mit der direkt aus dem Linux buffer cache gelesen werden kann.
Kommentar:	Dieser Wert ist ein Indikator für den Datendurchsatz des Prozessors, des Arbeitsspeichers und der entsprechenden Caches.

Buffered disk reads	
Skala:	Megabyte pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Gibt die Geschwindigkeit an mit der ohne Einsatz von Caches von der Festplatte gelesen werden kann.
Kommentar:	Dieser Wert ist ein Indikator für den Datendurchsatz der Festplatte bei sequenziellem Lesen von Daten.

3.2.3. YAGO

Die **Y**et **A**nother **G**reat **O**ntology (YAGO) ist eine große, automatisch erstellte Wissensdatenbank mit über 20 Millionen Fakten von mehr als zwei Millionen Entitäten[SKW07]. Ein Werkzeug, um die Datebank in SQL oder Notation3 zu konvertieren wird auf der Webseite des Projekts bereitgestellt. Im Rahmen des Schwesterprojekts RDF-3X wurden Datenbankanfragen für YAGO[NW09a] erstellt.

Execution time	
Skala:	Sekunden
Skalenniveau:	Proportionalskala
Beschreibung:	Gibt die Suchzeit an, die eine SPARQL Anfrage zum Ausführen benötigt hat.

3.3. Messwerkzeuge

Gute Benchmark-Ergebnisse ohne gute Messwerkzeuge zu erhalten ist praktisch unmöglich. Im Folgenden werden einige Messwerkzeuge vorgestellt, die genutzt wurden oder zumindest interessant gewesen wären.

3.3.1. sysstat

Das *sysstat*-Paket[sys] dient zur Leistungsmessung unter Linux. Im Gegensatz zu einem regelmäßigen Auslesen von `/proc/*` hat es den Vorteil, dass alle Änderungen zwischen den Messungen, mit Hilfe einer entsprechenden Schnittstelle des Linux-Kernels, erfasst und berücksichtigt werden. Für diese Fachstudie wurden *iostat* und *sar* des Pakets genutzt. Diese erzeugen schwer in Diagrammen verwertbare Ausgaben, weshalb ein Werkzeug zur Extraktion der Messdaten entwickelt werden musste.

User	
Skala:	Prozent
Skalenniveau:	Proportionalskala
Beschreibung:	Prozentualer Anteil der CPU Auslastung, der für das User-Level (Anwendungen) genutzt wird.
Kommentar:	sar: %user

System	
Skala:	Prozent
Skalenniveau:	Proportionalskala
Beschreibung:	Prozentualer Anteil der CPU Auslastung, der für das System-Level (Kernel) genutzt wird.
Kommentar:	sar: %system

IO-Wait	
Skala:	Prozent
Skalenniveau:	Proportionalskala
Beschreibung:	Prozentualer Anteil der CPU Auslastung in dem die CPU aufgrund von ausstehenden IO-Anfragen leer läuft.
Kommentar:	sar: %iowait

Idle	
Skala:	Prozent
Skalenniveau:	Proportionalskala
Beschreibung:	Prozentualer Anteil der CPU Auslastung in dem die CPU leer läuft und keine IO-Anfragen ausstehen.
Kommentar:	sar: %idle

Prozesse/s	
Skala:	Prozesse pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl der Prozesse, die pro Sekunde erstellt/gestartet werden.
Kommentar:	sar: proc/s

Kontextwechsel/s	
Skala:	Kontextwechsel pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl der Kontextwechsel pro Sekunde.
Kommentar:	sar: cswch/s

Page-In/s	
Skala:	Kilobyte pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl an Pages in Kilobyte pro Sekunde, die das System vom physikalischen Speicher gelesen hat.
Kommentar:	sar: pgpgin/s

Page-Out/s	
Skala:	Kilobyte pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl an Pages in Kilobyte pro Sekunde, die das System auf den physikalischen Speicher geschrieben hat.
Kommentar:	sar: pgpgout/s

fault/s	
Skala:	Page-Faults pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl an Page-Faults, die das System pro Sekunde gemacht hat.
Kommentar:	sar: fault/s

Pagescan/s	
Skala:	Pages pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl der Pages, die durch den *kswapd deamon* pro Sekunde abgetastet wurden.
Kommentar:	sar: pgscank/s

Pagesteal/s

Skala:	Pages pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl der Pages, die das System aus dem Cache pro Sekunde wiedergewonnen hat, um seine Anfragen an den Speicher zu bedienen.
Kommentar:	sar: pgsteal/s

Speicherverwaltungseffizienz

Skala:	Prozent
Skalenniveau:	Proportionalskala
Beschreibung:	Metrik für Speicherverwaltungseffizienz, die sich aus *Pagesteal / Pagescan* berechnet. Der Wert beträgt 0% falls keine Pages abgetastet wurden.
Kommentar:	sar: %vmeff

Speicherauslastung

Skala:	Prozent
Skalenniveau:	Proportionalskala
Beschreibung:	Auslastung des Speichers in Prozent.
Kommentar:	sar: %memused

Benötigt/Vorhanden

Skala:	Prozent
Skalenniveau:	Proportionalskala
Beschreibung:	Anteil des Speichers, der für die momentane Arbeitslast, in Relation zum gesamten Speicher, benötigt wird (*Benötigt / Vorhanden*). Diese Zahl kann größer als 100% werden.
Kommentar:	sar: %commit

Lese-Anfragen/s

Skala:	Anfragen pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl der Lese-Anfragen pro Sekunde, die an eine Festplatte gestellt wurden.
Kommentar:	iostat: r/s

Schreib-Anfragen/s

Skala:	Anfragen pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl der Schreib-Anfragen pro Sekunde, die an eine Festplatte gestellt wurden.
Kommentar:	iostat: w/s

Gelesen/s	
Skala:	Kilobyte pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl an Kilobyte pro Sekunde, die von einer Festplatte gelesen wurden.
Kommentar:	iostat: rkB/s

Geschrieben/s	
Skala:	Kilobyte pro Sekunde
Skalenniveau:	Proportionalskala
Beschreibung:	Anzahl an Kilobyte pro Sekunde, die auf eine Festplatte geschrieben wurden.
Kommentar:	iostat: wkB/s

Wartezeit	
Skala:	Millisekunden
Skalenniveau:	Proportionalskala
Beschreibung:	Durchschnittliche Zeit, die eine IO-Anfrage in der Warteschlange verbracht und auf der Festplatte benötigt hat.
Kommentar:	iostat: await

Servicezeit	
Skala:	Millisekunden
Skalenniveau:	Proportionalskala
Beschreibung:	Durchschnittliche Zeit, die eine IO-Anfrage auf einer Festplatte benötigt hat.
Kommentar:	iostat: svctm

3.3.2. timechart

timechart[Ven09] wäre wohl mit Abstand das interessanteste Werkzeug gewesen, denn das auf *perf* (linux-kernel:`/tools/perf/`) basierende Werkzeug visualisiert in einer SVG Datei die Arbeit des Schedulers. Leider ist der entsprechende Kernel-Patch, im Zeitraum der Fachstudie, noch deutlich zu neu (21. September 2009) für Maemo. In Abbildung 3.1 sieht man wie lange welcher Prozess auf welcher CPU aktiv ist oder leer läuft, ob er schläft oder wegen IO-Operationen geblockt ist. In der SVG Datei kann man praktisch unbegrenzt heranzoomen, sodass man in einer Datei „das Ganze“ (Auslastung) überblickt, aber auch Details („welcher Prozess konsumiert wann Rechenzeit“) erkennen kann. Diese Informationen können sehr wertvoll sein, wenn man Software für das System optimieren möchte.

3.3.3. FineToothComb

Die Entwickler von Qt benutzen zur Optimierung ihrer Software ein Werkzeug namens ***F**ine**T**ooth**C**omb* (FTC)[Bal09], welches an die JTAG-Schnittstelle des ARM-Prozessors ange-

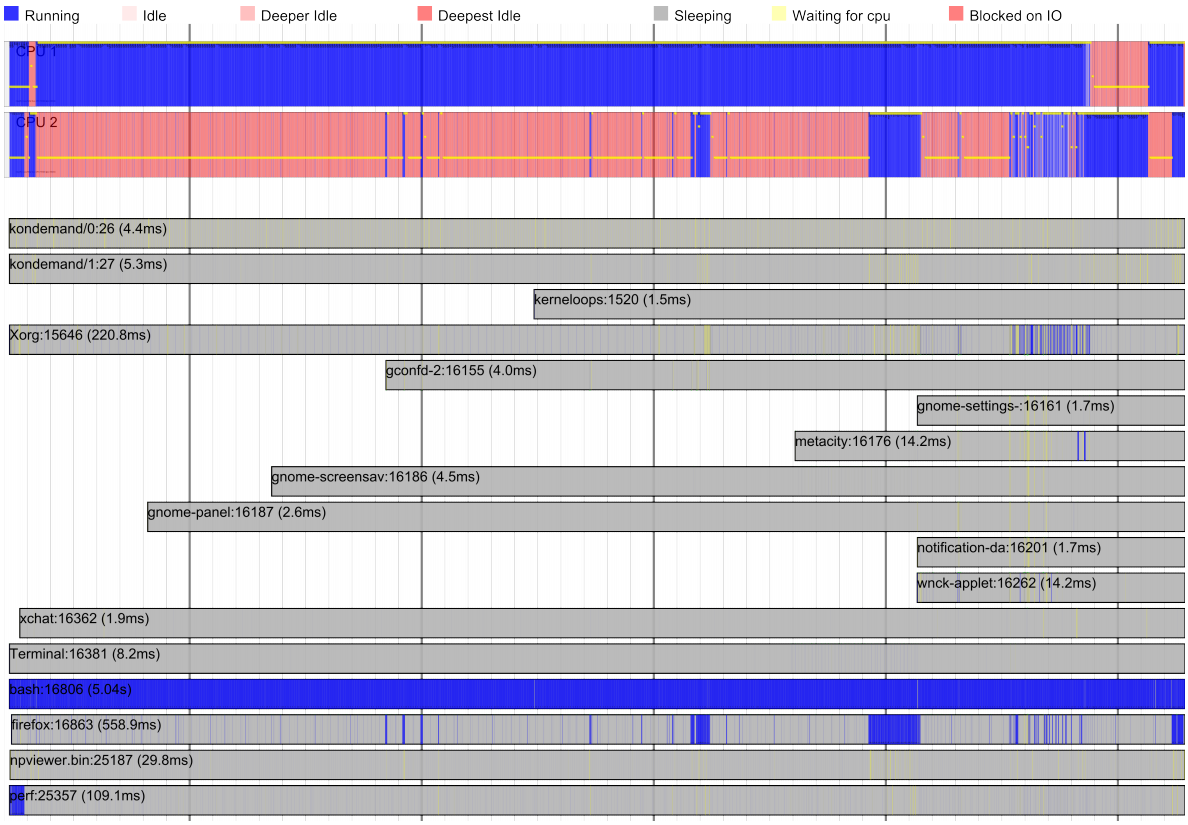

Abbildung 3.1.: Gerastertes timechart Diagramm

schlossen wird, um in Echtzeit Profiling-Informationen des ARM-Prozessors zu erhalten. Die Dabei anfallende Datenmenge (einige Gigabyte in ca. 10 Sekunden) setzt Nokia-interne Spezialhardware voraus, weshalb das Werkzeug nicht zur Verfügung steht. In Abbildung 3.2 lässt sich der Mehrwert gegenüber timechart erkennen: die gemessene Zeit in Nanosekunden, die ARM-Instruktion und der Zusammenhang mit dem Call Stack des Programms.

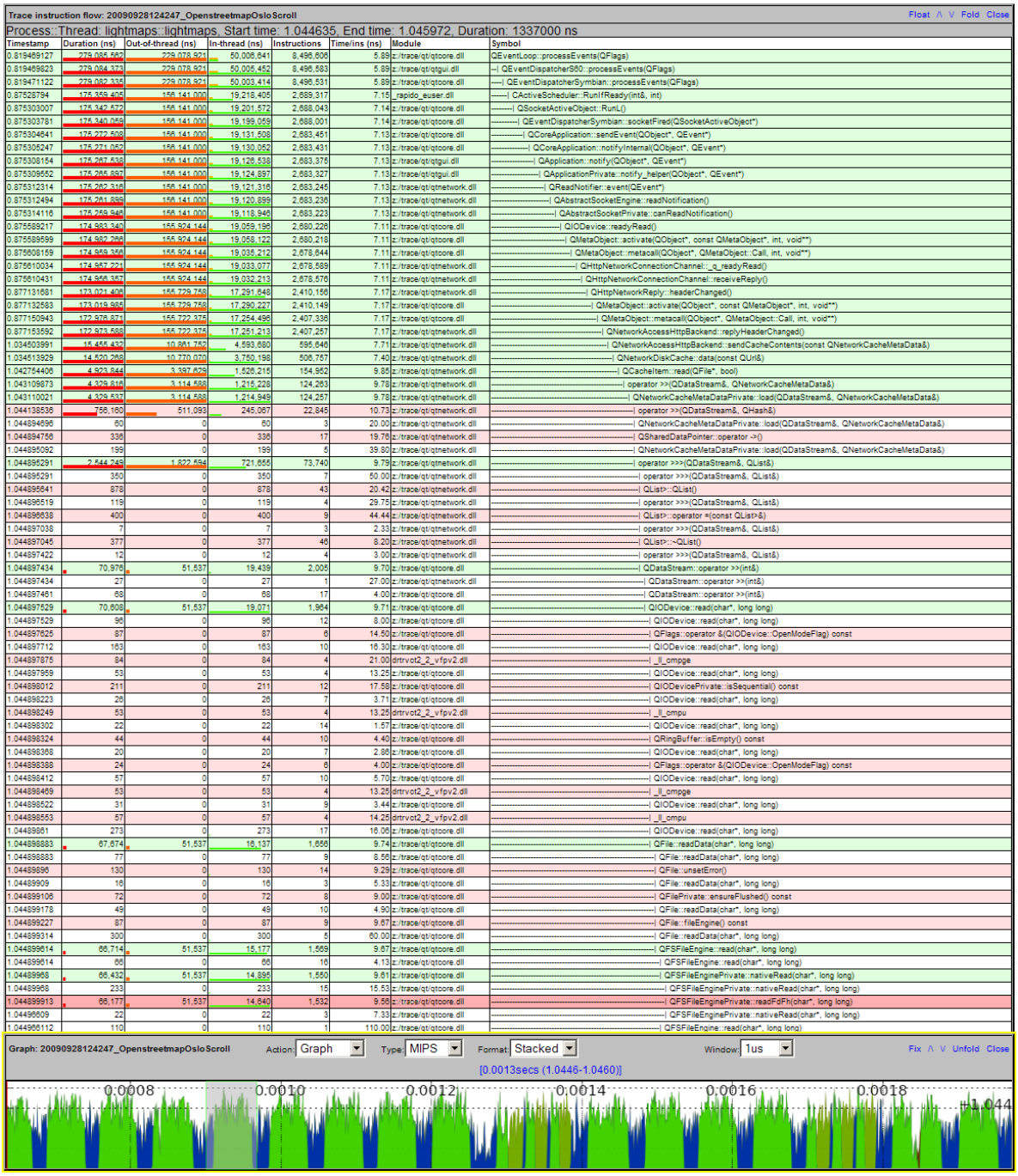

Abbildung 3.2.: Call Stack von OpenStreetMap auf Symbian[Balo9]

4. Durchführung und Bewertung der Benchmarks

In diesem Kapitel wird die Durchführung und Bewertung der Benchmarks beschrieben.

4.1. Allgemeines

Bei der Durchführung wurde darauf geachtet, dass sich das System vor jedem Benchmark in einem Zustand befindet, in dem es nicht unter Last steht und bereits „warm gelaufen“ ist. Das heißt, alle Ressourcen können für das Benchmark benutzt werden. Um eine gegenseitige Beeinflussung von Benchmarks auszuschließen wurde das System gegebenenfalls neu gestartet und nicht benötigte Softwarekomponenten deaktiviert.

4.2. Speichermedien

Alle drei internen Speichermedien wurden mit Hilfe von `hdparm` untersucht, um das schnellste Speichermedium des Systems zu identifizieren. Die Dokumentation empfiehlt die zwei- bis dreimalige Durchführung des folgenden Befehls, um ein verwertbares Ergebnis zu erhalten: `hdparm -tT [device]`

Speicher	Cached reads (MB/s)	Buffered disk reads (MB/s)
`/dev/mtdblock4` (intern)	101.41	18.03
`/dev/mmcblk0p1` (intern)	70.86	3.75
`/dev/mmcblk1p1` (extern)	31.77	3.76

Tabelle 4.1.: Ergebnisse von hdparm

Speicher	Größe (MB)
`/dev/mtdblock4` (intern)[a]	249
`/dev/mmcblk0p1` (intern)	1900
`/dev/mmcblk1p1` (extern)	3700

Tabelle 4.2.: Größe der Speichermedien

[a] etwa 60% sind durch Maemo vorbelegt

Tabelle 4.1 zeigt, dass `/dev/mtdblock4` das schnellste Speichermedium ist. Allerdings ist dieses Speichermedium auch das kleinste (siehe Tabelle 4.2) und bietet nicht genügend Platz für die Benchmarks. Deshalb wurden die Datenbank-Benchmarks auf `/dev/mmcblk0p1` durchgeführt. Die Protokolldateien des Benchmarks wurden auf `/dev/mtdblock4` abgelegt, um eine negative Beeinflussung der Leistung von `/dev/mmcblk0p1` zu vermeiden.

4.3. MySQL

MySQL war, wie gewohnt, leicht zu installieren.

4.3.1. BSBM

Das Einlesen des Datensatzes, der von BSBM generiert wurde, beanspruchte einige Zeit.

Anfrage	Ergebnis	Kommentar
Query 1	✓	
Query 2	✓	
Query 3	✓	
Query 4	✓	
Query 5	✓	
Query 6	✓	
Query 7	✗	Limitierung von MySQL 4.1[a].
Query 8	✓	
Query 9	✓	
Query 10	✓	
Query 11	✓	
Query 12	✓	

Tabelle 4.3.: Ergebnisse mit MySQL für BSBM

[a] existiert in MySQL 5 nicht mehr

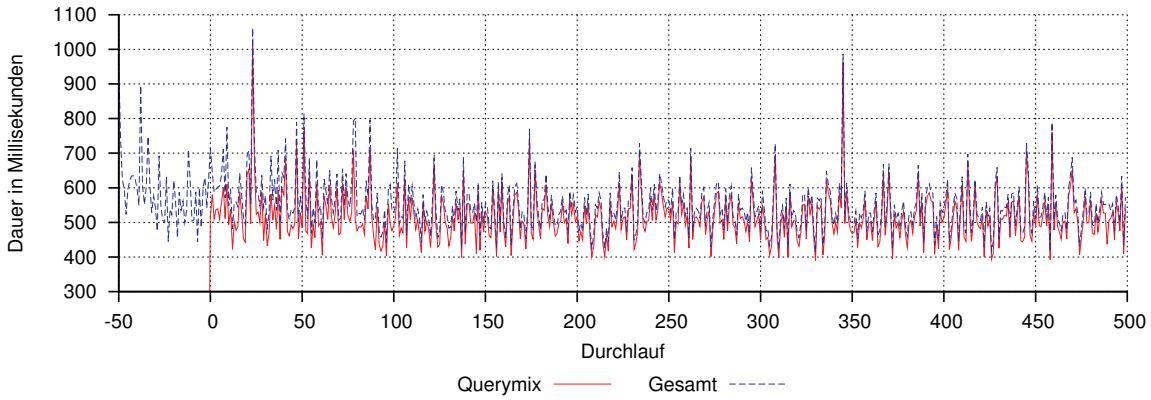

Abbildung 4.1.: BSBM Durchläufe für MySQL, Scalefactor 250

Scale factor:	250
Number of warmup runs:	50
Seed:	808080
Number of query mix runs (without warmups):	500 times
min/max Querymix runtime:	0.3901s / 1.0130s
Total runtime:	257.455 seconds
QMpH:	6991.51 query mixes per hour
CQET:	0.51491 seconds average runtime of query mix
CQET (geom.):	0.51046 seconds geometric mean runtime of query mix

Tabelle 4.4.: BSBM Queryzusammenfassung für MySQL, Scalefactor 250

Metrics for Query:	1
Count:	500 times executed in whole run
AQET:	0.034935 seconds (arithmetic mean)
AQET(geom.):	0.034748 seconds (geometric mean)
QPS:	28.62 Queries per second
minQET/maxQET:	0.03027344s / 0.12417603s
Average result count:	0.40
min/max result count:	0 / 6
Number of timeouts:	0
Metrics for Query:	2
Count:	3000 times executed in whole run
AQET:	0.010051 seconds (arithmetic mean)
AQET(geom.):	0.009663 seconds (geometric mean)
QPS:	99.50 Queries per second
minQET/maxQET:	0.00607300s / 0.21871948s
Average result count:	23.70
min/max result count:	11 / 46
Number of timeouts:	0
Metrics for Query:	3
Count:	500 times executed in whole run
AQET:	0.025973 seconds (arithmetic mean)
AQET(geom.):	0.025579 seconds (geometric mean)
QPS:	38.50 Queries per second
minQET/maxQET:	0.02032471s / 0.15341187s
Average result count:	0.21
min/max result count:	0 / 2
Number of timeouts:	0
Metrics for Query:	4
Count:	500 times executed in whole run
AQET:	0.039181 seconds (arithmetic mean)
AQET(geom.):	0.038909 seconds (geometric mean)
QPS:	25.52 Queries per second
minQET/maxQET:	0.03594971s / 0.14120483s
Average result count:	0.00
min/max result count:	0 / 0
Number of timeouts:	0

Tabelle 4.5.: BSBM Query-Ergebnisse für MySQL, Scalefactor 250 (1)

Metrics for Query:	5
Count:	500 times executed in whole run
AQET:	0.271272 seconds (arithmetic mean)
AQET(geom.):	0.264641 seconds (geometric mean)
QPS:	3.69 Queries per second
minQET/maxQET:	0.16052246s / 0.72329712s
Average result count:	2.31
min/max result count:	0 / 5
Number of timeouts:	0
Metrics for Query:	6
Count:	500 times executed in whole run
AQET:	0.022850 seconds (arithmetic mean)
AQET(geom.):	0.021663 seconds (geometric mean)
QPS:	43.76 Queries per second
minQET/maxQET:	0.020078247s / 0.43319702s
Average result count:	1.04
min/max result count:	1 / 20
Number of timeouts:	0
Metrics for Query:	8
Count:	1000 times executed in whole run
AQET:	0.006747 seconds (arithmetic mean)
AQET(geom.):	0.006544 seconds (geometric mean)
QPS:	148.22 Queries per second
minQET/maxQET:	0.00427246s / 0.03955078s
Average result count:	4.71
min/max result count:	0 / 17
Number of timeouts:	0
Metrics for Query:	9
Count:	2000 times executed in whole run
AQET:	0.006673 seconds (arithmetic mean)
AQET(geom.):	0.006549 seconds (geometric mean)
QPS:	149.85 Queries per second
minQET/maxQET:	0.00448608s / 0.04028320s
Average result (Bytes):	22.44
min/max result (Bytes):	7 / 37
Number of timeouts:	0

Tabelle 4.6.: BSBM Query-Ergebnisse für MySQL, Scalefactor 250 (2)

Metrics for Query:	10
Count:	1000 times executed in whole run
AQET:	0.006409 seconds (arithmetic mean)
AQET(geom.):	0.006303 seconds (geometric mean)
QPS:	156.04 Queries per second
minQET/maxQET:	0.004150395 / 0.04714966s
Average result count:	2.39
min/max result count:	0 / 10
Number of timeouts:	0
Metrics for Query:	11
Count:	500 times executed in whole run
AQET:	0.003181 seconds (arithmetic mean)
AQET(geom.):	0.003069 seconds (geometric mean)
QPS:	314.38 Queries per second
minQET/maxQET:	0.002868655 / 0.03671265s
Average result count:	1.00
min/max result count:	1 / 1
Number of timeouts:	0
Metrics for Query:	12
Count:	500 times executed in whole run
AQET:	0.004210 seconds (arithmetic mean)
AQET(geom.):	0.003900 seconds (geometric mean)
QPS:	237.50 Queries per second
minQET/maxQET:	0.003662115 / 0.08407593s
Average result (Bytes):	1.00
min/max result (Bytes):	1 / 1
Number of timeouts:	0

Tabelle 4.7.: BSBM Query-Ergebnisse für MySQL, Scalefactor 250 (3)

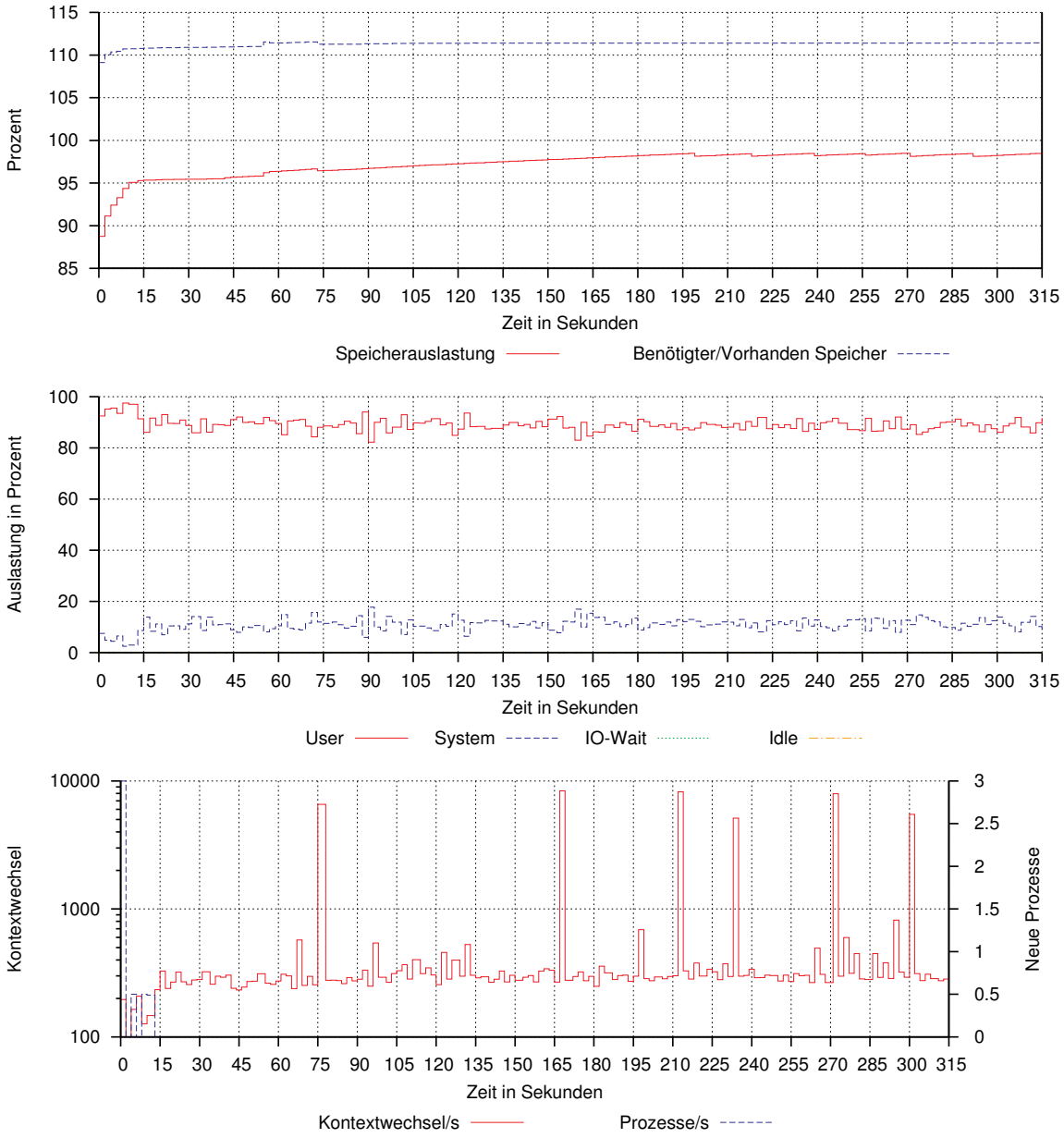

Abbildung 4.2.: CPU-/Speicherauslastung während BSBM für MySQL, Scalefactor 250

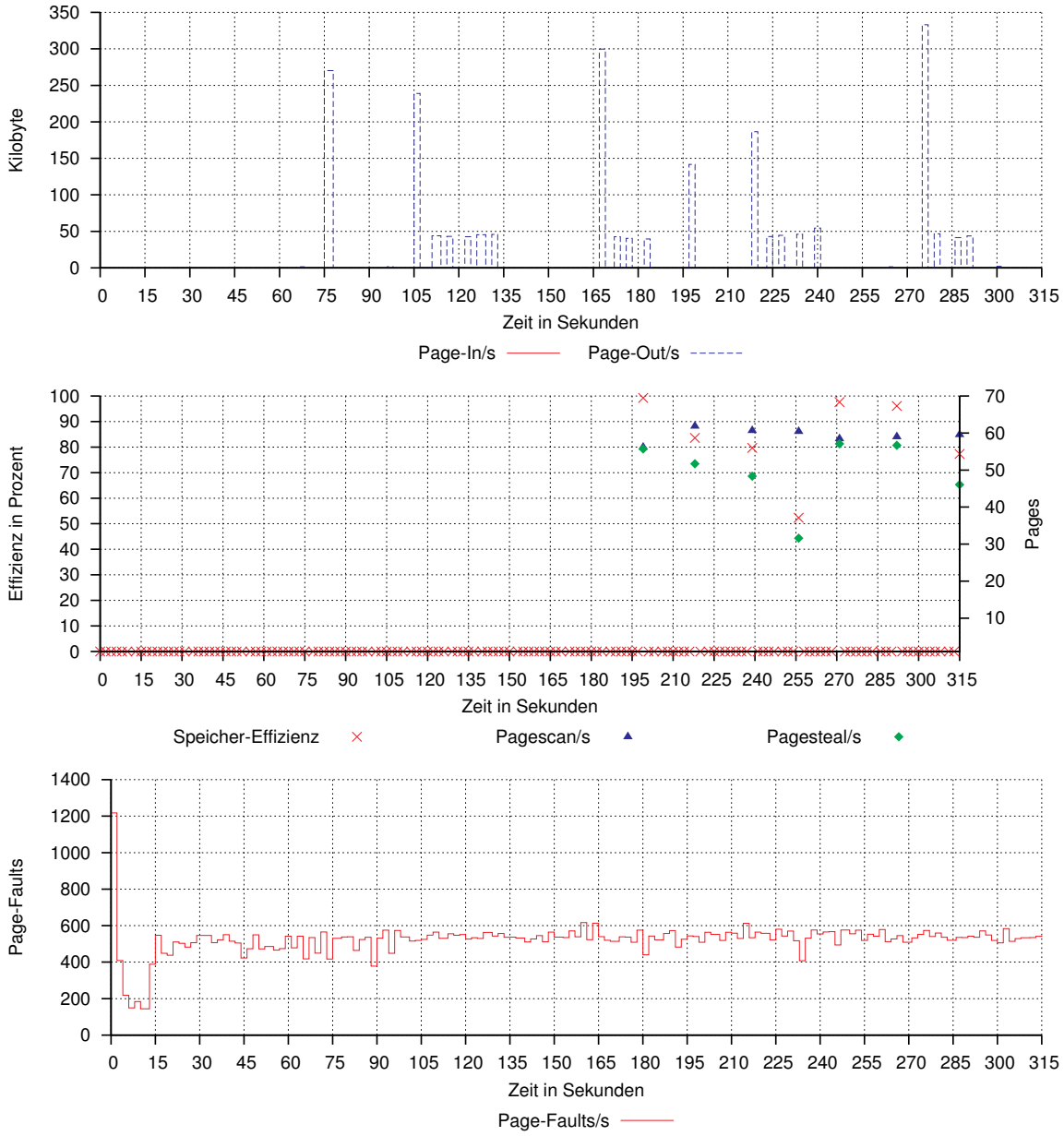

Abbildung 4.3.: Page-Informationen während BSBM für MySQL, Scalefactor 250

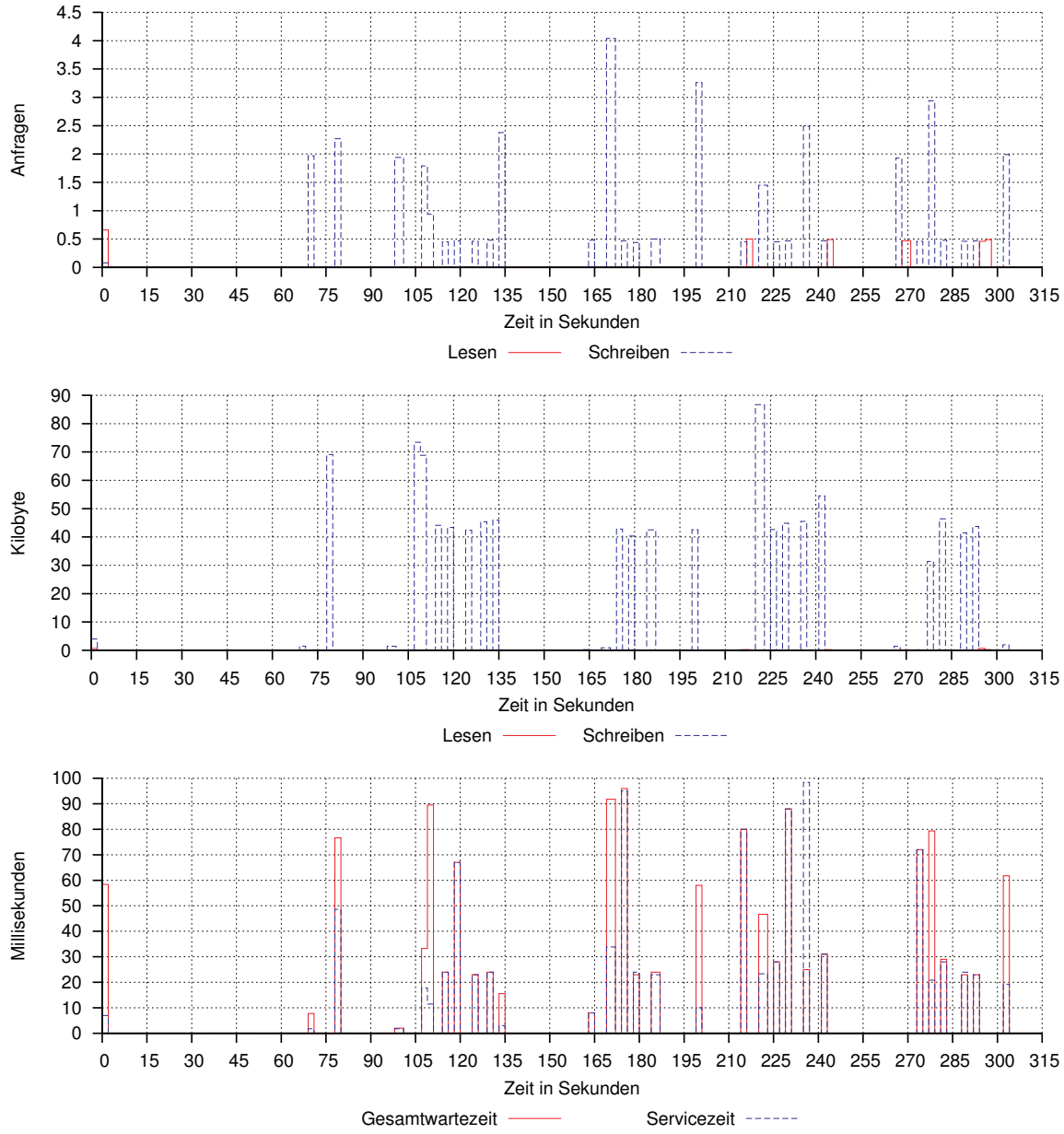

Abbildung 4.4.: IO während BSBM für MySQL, Scalefactor 250

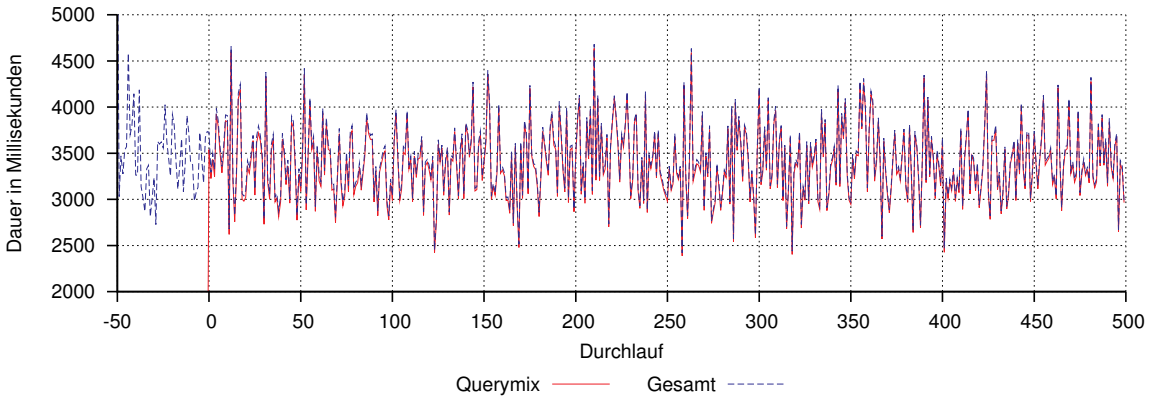

Abbildung 4.5.: BSBM Durchläufe für MySQL, Scalefactor 2785

Scale factor:	2785
Number of warmup runs:	50
Seed:	808080
Number of query mix runs (without warmups):	500 times
min/max Querymix runtime:	2.3865s / 4.6351s
Total runtime:	1693.079 seconds
QMpH:	1063.15 query mixes per hour
CQET:	3.38616 seconds average runtime of query mix
CQET (geom.):	3.36396 seconds geometric mean runtime of query mix

Tabelle 4.8.: BSBM Queryzusammenfassung für MySQL, Scalefactor 2785

Metrics for Query:	1
Count:	500 times executed in whole run
AQET:	0.377428 seconds (arithmetic mean)
AQET(geom.):	0.374839 seconds (geometric mean)
QPS:	2.65 Queries per second
minQET/maxQET:	0.286682135 / 0.75045776s
Average result count:	1.50
min/max result count:	0 / 10
Number of timeouts:	0
Metrics for Query:	2
Count:	3000 times executed in whole run
AQET:	0.010324 seconds (arithmetic mean)
AQET(geom.):	0.009989 seconds (geometric mean)
QPS:	96.86 Queries per second
minQET/maxQET:	0.005981455 / 0.09060669s
Average result count:	20.48
min/max result count:	8 / 41
Number of timeouts:	0
Metrics for Query:	3
Count:	500 times executed in whole run
AQET:	0.215681 seconds (arithmetic mean)
AQET(geom.):	0.213746 seconds (geometric mean)
QPS:	4.64 Queries per second
minQET/maxQET:	0.161163335 / 0.38400269s
Average result count:	0.98
min/max result count:	0 / 7
Number of timeouts:	0
Metrics for Query:	4
Count:	500 times executed in whole run
AQET:	0.357626 seconds (arithmetic mean)
AQET(geom.):	0.357300 seconds (geometric mean)
QPS:	2.80 Queries per second
minQET/maxQET:	0.338378915 / 0.50640869s
Average result count:	0.21
min/max result count:	0 / 7
Number of timeouts:	0

Tabelle 4.9.: BSBM Query-Ergebnisse für MySQL, Scalefactor 2785 (1)

Metrics for Query:	5
Count:	500 times executed in whole run
AQET:	1.855128 seconds (arithmetic mean)
AQET(geom.):	1.817285 seconds (geometric mean)
QPS:	0.54 Queries per second
minQET/maxQET:	0.887176515 / 3.20874023s
Average result count:	4.53
min/max result count:	0 / 5
Number of timeouts:	0
Metrics for Query:	6
Count:	500 times executed in whole run
AQET:	0.187529 seconds (arithmetic mean)
AQET(geom.):	0.187219 seconds (geometric mean)
QPS:	5.33 Queries per second
minQET/maxQET:	0.18112183s / 0.32949829s
Average result count:	1.07
min/max result count:	1 / 4
Number of timeouts:	0
Metrics for Query:	8
Count:	1000 times executed in whole run
AQET:	0.070660 seconds (arithmetic mean)
AQET(geom.):	0.048306 seconds (geometric mean)
QPS:	14.15 Queries per second
minQET/maxQET:	0.00527954s / 0.30023193s
Average result count:	5.04
min/max result count:	0 / 16
Number of timeouts:	0
Metrics for Query:	9
Count:	2000 times executed in whole run
AQET:	0.020367 seconds (arithmetic mean)
AQET(geom.):	0.017606 seconds (geometric mean)
QPS:	49.10 Queries per second
minQET/maxQET:	0.00442505s / 0.26486206s
Average result (Bytes):	21.67
min/max result (Bytes):	3 / 41
Number of timeouts:	0

Tabelle 4.10.: BSBM Query-Ergebnisse für MySQL, Scalefactor 2785 (2)

Metrics for Query:	10
Count:	1000 times executed in whole run
AQET:	0.045264 seconds (arithmetic mean)
AQET(geom.):	0.034298 seconds (geometric mean)
QPS:	22.09 Queries per second
minQET/maxQET:	0.00497437s / 0.517517709s
Average result count:	2.01
min/max result count:	0 / 10
Number of timeouts:	0
Metrics for Query:	11
Count:	500 times executed in whole run
AQET:	0.008395 seconds (arithmetic mean)
AQET(geom.):	0.007058 seconds (geometric mean)
QPS:	119.12 Queries per second
minQET/maxQET:	0.00289917s / 0.12243652s
Average result count:	1.00
min/max result count:	1 / 1
Number of timeouts:	0
Metrics for Query:	12
Count:	500 times executed in whole run
AQET:	0.009108 seconds (arithmetic mean)
AQET(geom.):	0.007876 seconds (geometric mean)
QPS:	109.79 Queries per second
minQET/maxQET:	0.00360107s / 0.209259035
Average result (Bytes):	1.00
min/max result (Bytes):	1 / 1
Number of timeouts:	0

Tabelle 4.11.: BSBM Query-Ergebnisse für MySQL, Scalefactor 2785 (3)

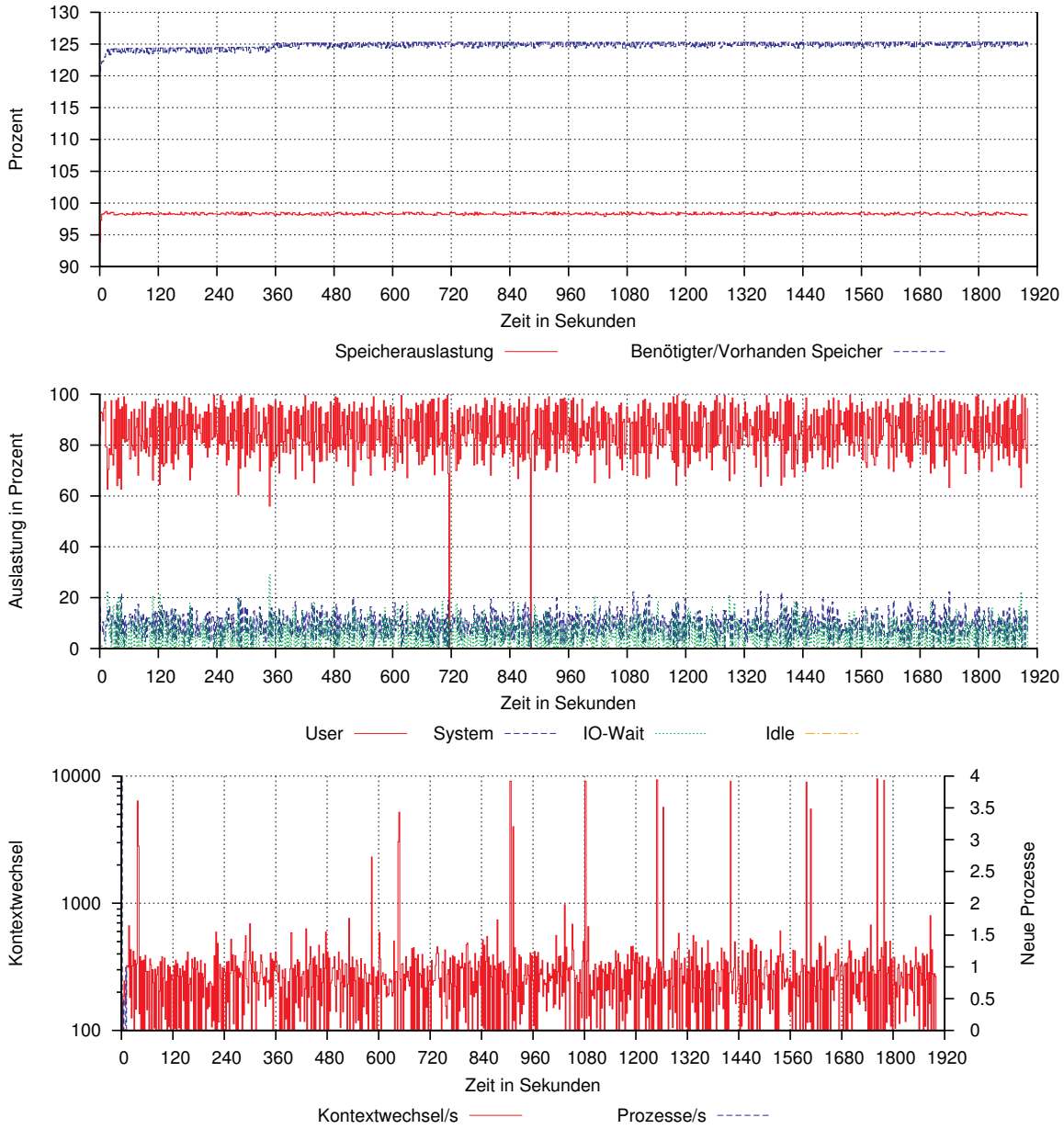

Abbildung 4.6.: CPU-/Speicherauslastung während BSBM für MySQL, Scalefactor 2785

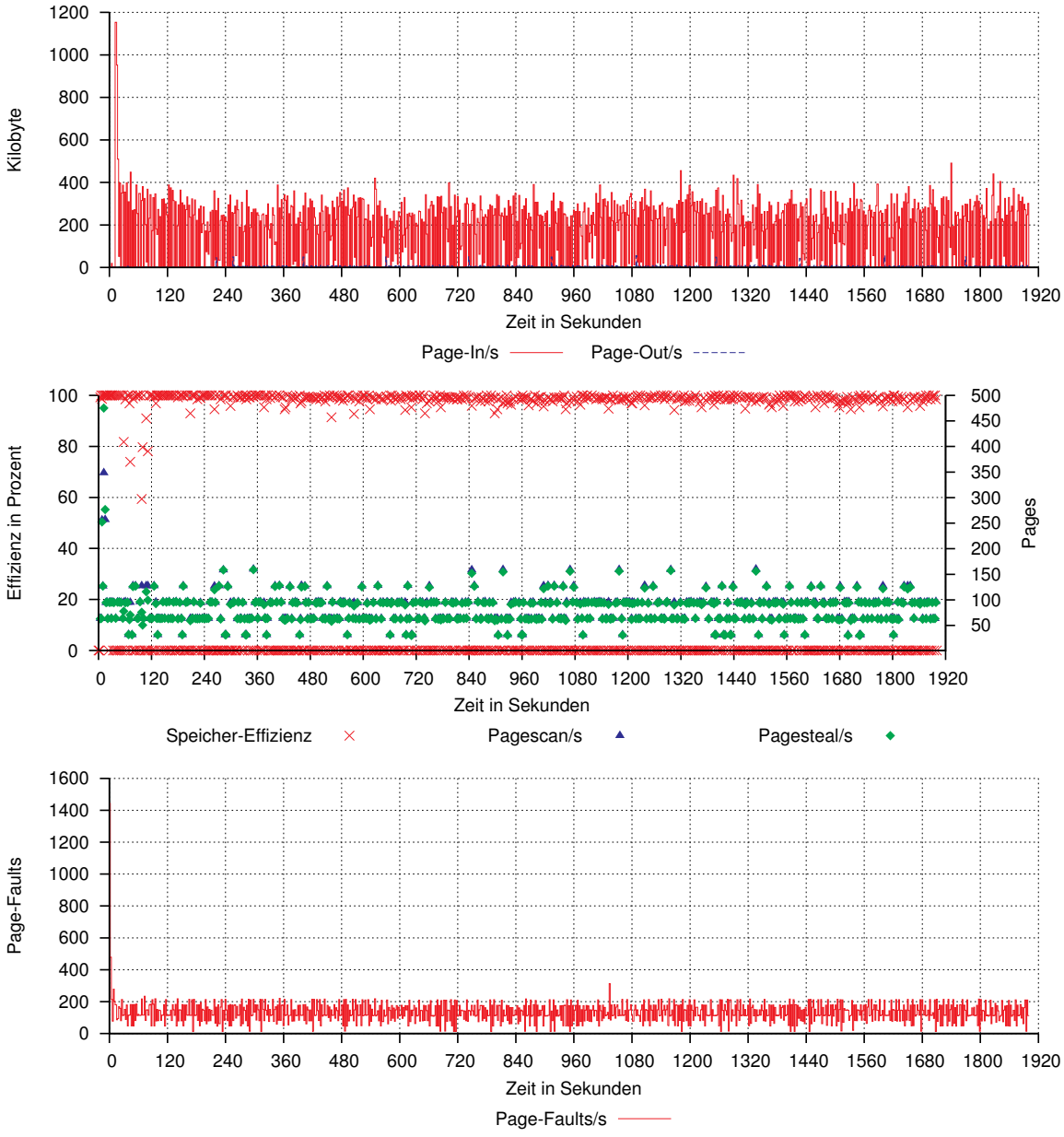

Abbildung 4.7.: Page-Informationen während BSBM für MySQL, Scalefactor 2785

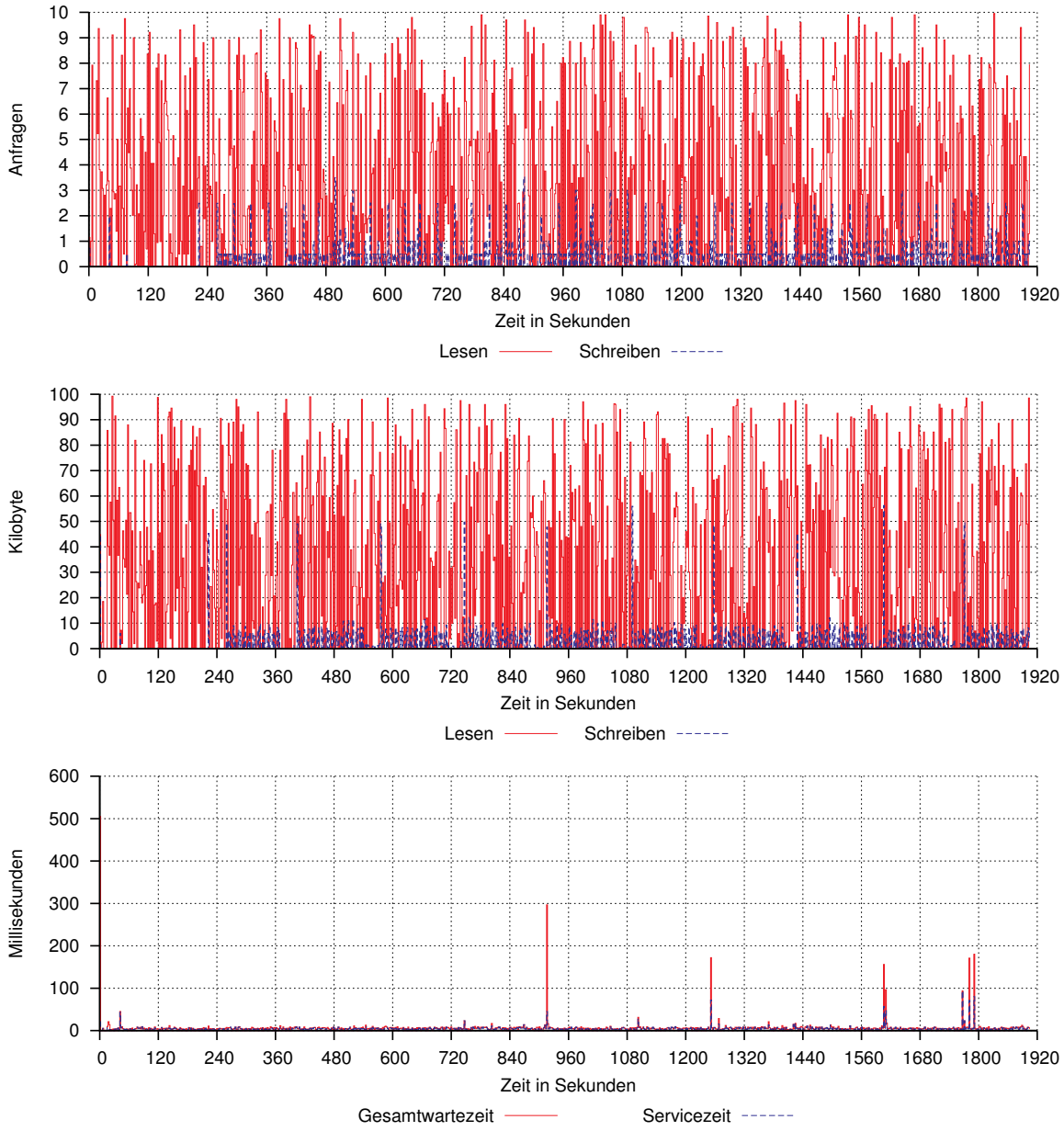

Abbildung 4.8.: IO während BSBM für MySQL, Scalefactor 2785

4.3.2. YAGO

Die Datensätze der YAGO-Datenbank erzeugten, trotz massiver Kürzung, eine unerwartet große Datenbank, bei der das Erstellen der Indices mehr als einen Tag dauerte. Die ungekürzte Datenbank benötigt auf einem handelsüblichen Laptop mehr als zwei Tage um die Indices zu erstellen:

```
mysql> CREATE INDEX factsarg1Index ON facts (arg1);
Query OK, 132263376 rows affected (2 days 17 hours 12 min 49.23 sec)
Records: 132263376 Duplicates: 0 Warnings: 0
```

Zudem funktionieren die von RDF-3X bereitgestellten SQL-Queries nicht mit der erstellten Datenbank. Eine intensive Fehlersuche wurde durch die langen Wartezeiten erschwert und führte letztendlich zu keiner Lösung.

4.4. RDF-3X

Die im Rahmen dieser Fachstudie verwendete Version von RDF-3X hat unsere Sorgen bezüglich Arbeitsaufwand bei weitem übertroffen.

Fehlermeldungen des Compilers: RDF-3X kompilierte nicht; es folgen einige Beispiele:

```
error: dereferencing type-punned pointer will break strict-aliasing rules
error: uint64_t was not declared in this scope
```

Der erste Fehler entsteht aufgrund von `-Werror`, weshalb die `CFLAGS` auf `-W -Wall -O3 -mcpu=arm1136jf-s` gesetzt wurden. Der zweite Fehler entsteht aufgrund einer fehlenden Definition eines Typs, weshalb folgende Zeile in `tools/rdf3xload/TempFile.hpp` eingefügt wurde:

```
typedef unsigned long long uint64_t;
```

32bit-Probleme: RDF-3X wurde für 64bit Systeme entwickelt. Prinzipiell wäre das kein Problem, weil der Compiler ein 64bit `unsigned long long` mit Hilfe von zwei 32bit Integern emulieren kann. Das größere Problem ist der Einsatz von Memory-Mapping, der die Dateigröße auf $2^{31} \approx 2.1$GB beschränkt. Zudem ist das Betriebssystem auf dem N810 nicht entsprechend kompiliert worden, sodass Dateien mit einer entsprechenden Größe nicht ohne weiteres verwendet werden können.

Mangelhafter SPARQL-Parser: Der von Hand geschriebene Parser ist nicht konform zur SPARQL-Spezifikation und hat eine mangelhafte Fehlerdiagnose: er gibt weder Zeilennummer mit Position des Zeichens in der Zeile noch eine verständliche Interpretation des Fehlers aus. Aufgrund der massiven Fehler beim BSBM-Benchmark wurde der Parser entsprechend modifiziert um Zeilennummer und Zeichenposition auszugeben.

Kein SPARQL-Endpoint: Das BSBM-Benchmark setzt einen SPARQL-Endpoint voraus, den RDF-3X nicht implementiert. Deshalb wurde im Rahmen der Fachstudie eine sehr experimentelle Implementierung mit Hilfe von *libmicrohttpd*[Gro] realisiert.

4.4.1. BSBM

Bei diesem Benchmark war von Anfang an unklar ob es funktioniert, denn RDF-3X ist ein Prototyp. Mit Hilfe eines selbst geschriebenen SPARQL-Endpoints, der `HTTP GET` Anfragen akzeptiert und für RDF-3X umwandelt, konnte festgestellt werden, dass keine BSBM Anfrage korrekt ausgeführt werden kann (siehe Tabelle 4.12). Entsprechend war es auch nicht nötig einen XML-Ausgabe-Operator zu entwickeln, der ein Ergebnis an BSBM übermittelt.

Anfrage	Ergebnis	Kommentar
Query 1	✗	`FILTER` mit Ungleichung.
Query 2	✗	`OPTIONAL`
Query 3	✗	`FILTER` mit Ungleichung und `bound()`, `OPTIONAL`, `ORDER BY`
Query 4	✗	`FILTER` mit Ungleichung, `ORDER BY`, `OFFSET`
Query 5	✗	`FILTER` mit Ungleichung und Rechenausdruck, `ORDER BY`
Query 6	✗	`FILTER` mit regulärem Ausdruck
Query 7	✗	`FILTER` mit Ungleichung, `OPTIONAL`
Query 8	✗	`FILTER` mit `langMatches()`, `ORDER BY DESC`
Query 9	✗	`DESCRIBE`[a]
Query 10	✗	`FILTER` mit Ungleichung, `ORDER BY` mit Uminterpretation
Query 11	✗	
Query 12	✗	`CONSTRUCT`[a]

Abkürzungen:

✓ funktioniert vollständig oder größtenteils

✗ funktioniert nicht

Tabelle 4.12.: RDF-3X Ergebnisse für BSBM

[a] nicht Bestandteil von `SELECT`

4.4.2. YAGO

Dieses Benchmark war, aufgrund der oben genannten 32bit Problematik, nicht wie in der Arbeit[NW08] von Thomas Neumann durchführbar. Eine Untersuchung der Quellen ergab, dass `mmap()` sowie der Zugriff auf Speicherseiten ungünstig abstrahiert wurden: ein Umstieg auf `mmap64()`, mit dem man das 2GB-Limit aufheben kann, oder eine in mehrere Dateien aufgeteilte Datenbank hätte umfassende Änderungen nach sich gezogen. Aufgrund der Limitierung der Dateigröße wurde der Datensatz gekürzt und die Datenbank auf einem

64bit System mit *rdf3xload* erstellt. Das Lesen/Schreiben der Datenbank macht keine 32/64bit-Probleme, wie ein Blick in die Quellen verriet.

Den RDF-3X-Quellen lag ein Programm namens *flushcaches* nicht bei, welches von *runbenchmark* benutzt wird. Mit der Annahme, dass das Programm die IO-Caches leert wurde es durch folgendes Shell-Script ersetzt:

```
#!/bin/sh
echo 3 > /proc/sys/vm/drop_caches
```

Um den Datensatz zu kürzen wurde die original YAGO Ontologie[yag] in der Version 2008-w40-2 herangezogen. Das Kürzen gestaltete sich als schwierig, da zunächst nicht klar war wie viel gekürzt werden musste, damit die Testdaten funktionieren. Zunächst wurde versucht einfach 10% der Ontologie zu verwenden. Dazu wurde mit den auf der YAGO Homepage erhältlichen Konvertierungswerkzeugen eine Notation3-Datei erstellt und nur jede zehnte Zeile verwendet. Leider wurde der Datensatz damit so ungünstig gekürzt, dass die Query-Anfragen keine Ergebnisse lieferten. Die Verwendung jeder vierten Zeile brachte ebenfalls nicht den gewünschten Effekt. Ein neuer und letztendlich auch funktionierender Ansatz war, alle RDF-3X-Queries durchzugehen und die verwendeten YAGO-Facts zu extrahieren. Aufgrund der etwas neueren Version der Ontologie mussten einige Facts der RDF-3X-Queries übertragen werden. Die Facts in den Queries wurden entsprechend angepasst (siehe Tabelle 4.13).

Extrahierte Facts	Verwendete Facts	Kommentar
actedIn	actedIn	
bornInLocation	bornIn	
directed	directed	
familyNameOf	familyNameOf	
givenNameOf	givenNameOf	
hasDoctoralAdvisor	hasAcademicAdvisor	
hasWonPrize	hasWonPrize	
	inLanguage	Diese Facts werden nicht explizit in den Queries referenziert werden jedoch trotzdem benötigt.
isCalled	isCalled	
isMarriedTo	isMarriedTo	
livesIn	livesIn	
locatedIn	locatedIn	
producedInCountry	produced	
subClassOf	subClassOf	
type	type	

Tabelle 4.13.: Reduzierte Facts der Ontologie, welche in den RDF-3X-Queries verwendeten werden.

Anfrage	Ergebnis	Cold-Caches (s)	Warm-Caches (s)
A1	✓	0.102	0.028
A2	✓	0.141	0.011
A3	✓	0.231	0.070
B1	✓	0.150	0.017
B2	✓	0.016	0.001
B3	✓	0.048	0.003
C1	✓	2.188	0.408
C2	✓	0.097	0.008
C3	✓	0.486	0.038

Abkürzungen:

✓	funktioniert vollständig oder größtenteils
✗	funktioniert nicht

Tabelle 4.14.: RDF-3X Ergebnisse für YAGO

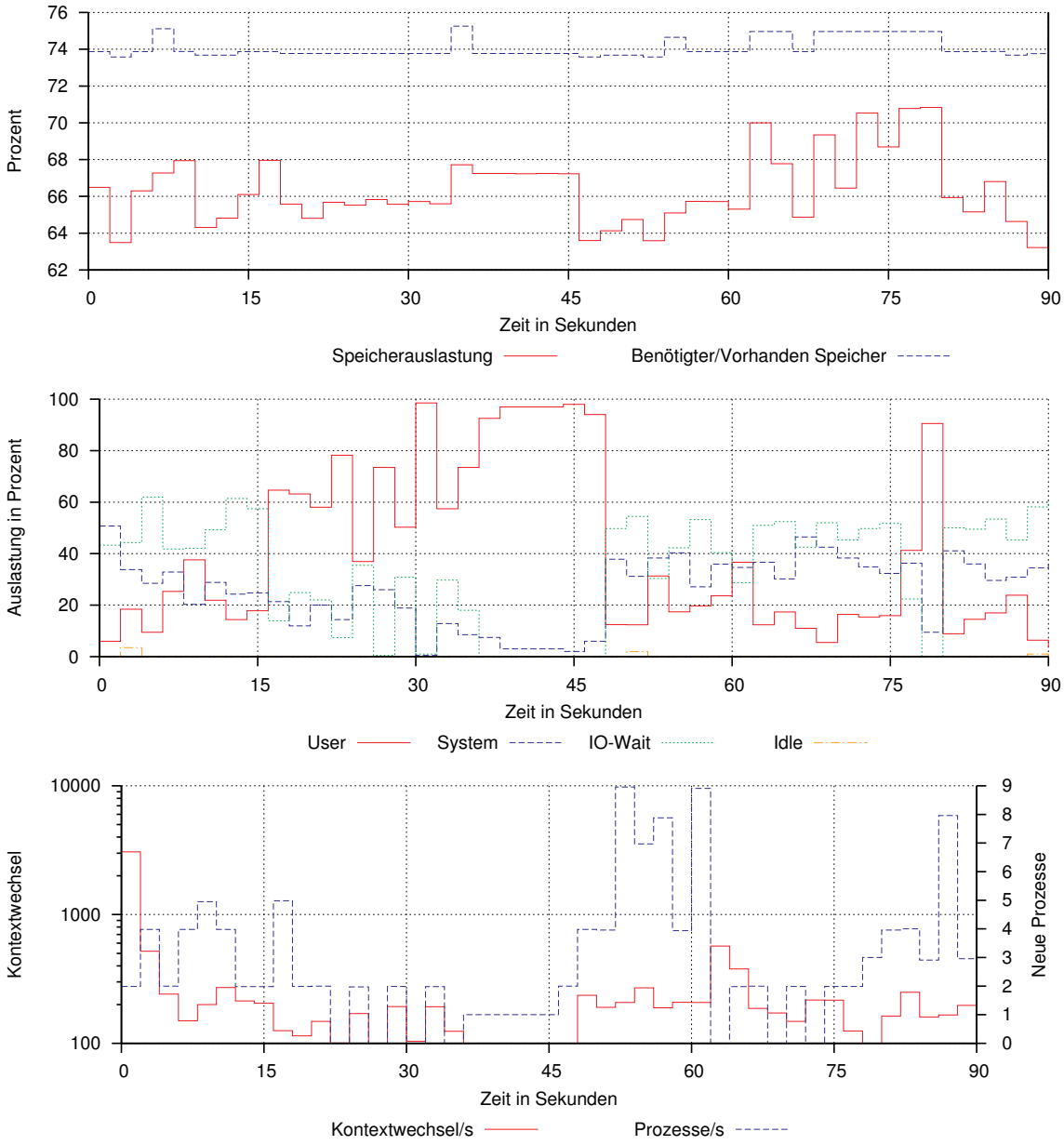

Abbildung 4.9.: CPU-/Speicherauslastung während des Benchmarks für RDF-3X mit YAGO

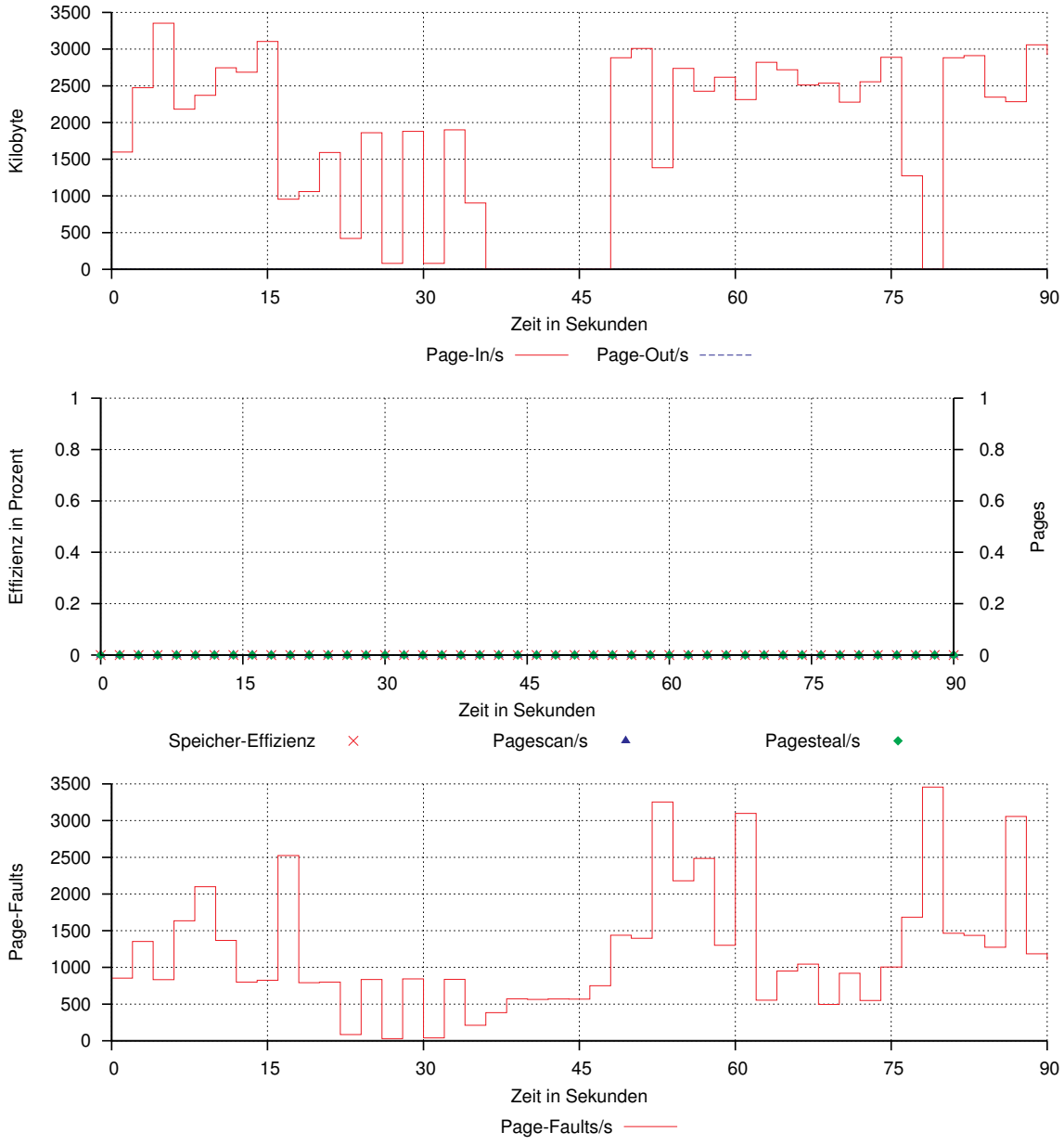

Abbildung 4.10.: Page-Informationen während des Benchmarks für RDF-3X mit YAGO

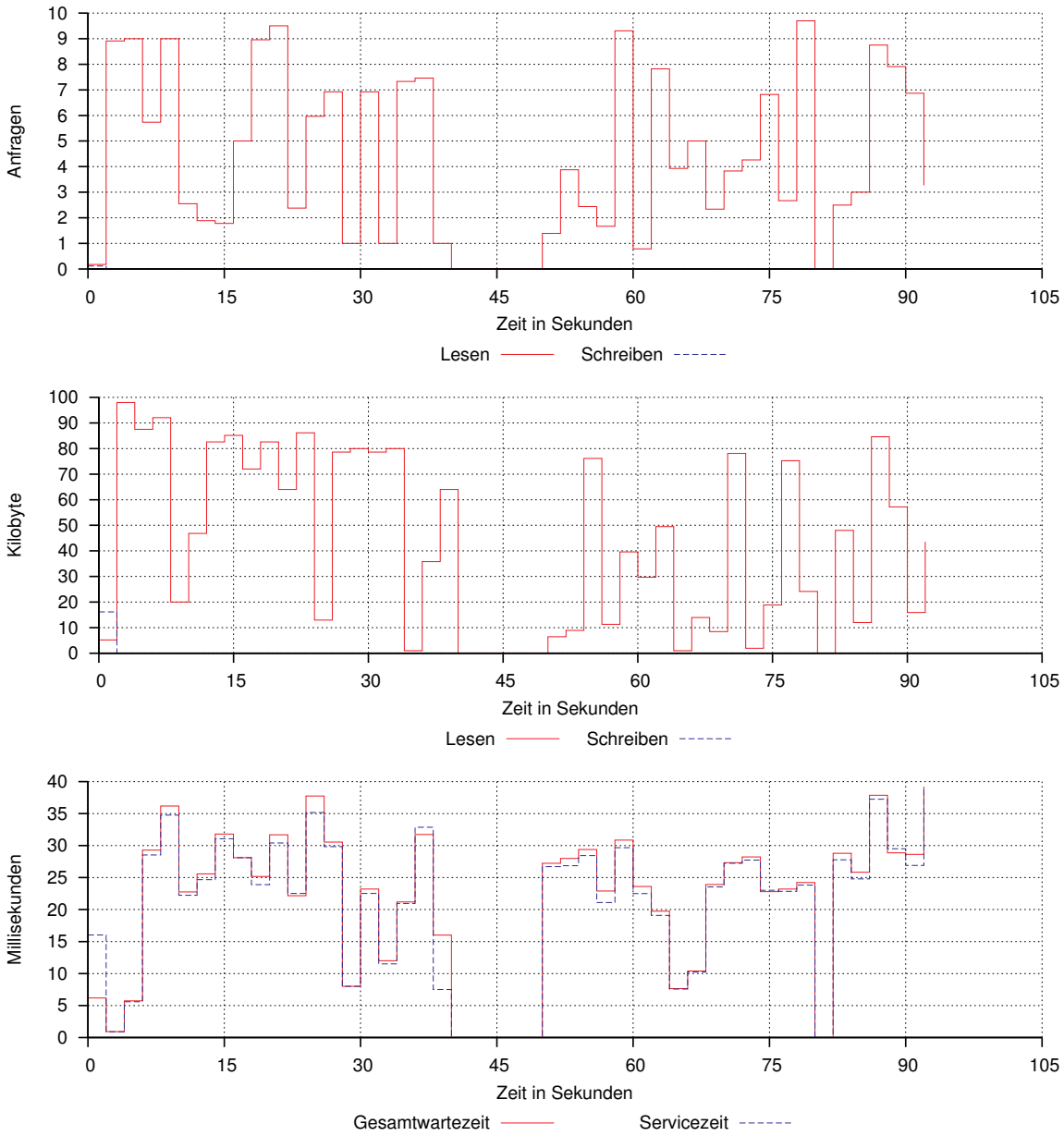

Abbildung 4.11.: IO während des Benchmarks für RDF-3X mit YAGO

4.5. Auswertung

Aufgrund der bereits erläuterten Probleme bei den verschieden Benchmarks gibt es leider keine Daten, die sowohl für MySQL als auch für RDF-3X eine einheitliche Basis bilden (siehe Tabelle 4.15) - ein Vergleich unter den Datenbanksystemen war nicht möglich. Die

Auswertung im Rahmen dieser Fachstudie beschränkt sich deshalb auf die einzelnen Datenbanksysteme mit ihren individuellen Messwerten.

Datenbanksystem	**Sprache**	**BSBM**	**YAGO**	Hardware
MySQL	SQL	✓	✗	N810
RDF-3X	SPARQL	✗	✓	N810

Abkürzungen:

✓ funktioniert vollständig oder größtenteils

✗ funktioniert nicht

Tabelle 4.15.: Vergleich der Datenbanksysteme

4.5.1. MySQL mit BSBM

Wie aus den Abbildungen 4.2 und 4.6 hervorgeht verursacht MySQL eine hohe Grundauslastung der CPU und des Speichers. Auch bei sehr kleinen Datenbanken (ca. 5MB) macht sich das bemerkbar. Obwohl nur wenige Anfragen an den physikalischen Speicher gerichtet werden, die CPU damit also nicht belastet sein sollte. Die Speichernutzung ist, erwartungsgemäß, bei der größeren Datenbank (ca. 50MB) höher. Aus den Abbildungen 4.3 und 4.7 geht hervor, dass die Speicherverwaltung bei der größeren Datenbank besser funktioniert. Obwohl sich die Datenbanken in der Größe um Faktor zehn unterscheiden, ist die Laufzeit der Anfragen nur um Faktor sieben verschieden. Bei einem Testlauf des Benchmarks ist zusätzlich aufgefallen, dass während des Benchmarks bei beiden Datenbankgrößen die Antwortzeit der Benutzeroberfläche des Gerätes spürbar leidet.

4.5.2. RDF-3X mit YAGO

Die Ergebnisse von RDF-3X haben uns aufgrund ihrer Antwortzeit überrascht (siehe Tabelle 4.16). Trotz der Größe von knapp 800MB liegen die Antwortzeiten bei den Warm-Caches im geometrischen Mittel unter 20 Millisekunden. Im Vergleich dazu liegen die Referenz-Zeiten bei einer Datenbankgröße von 2,7GB im geometrischen Mittel bei 40 Millisekunden[NW08]. Dies kann, neben der erheblichen Kürzung des YAGO-Datensatzes, vermutlich zum Teil auf die Version von RDF-3X zurückgeführt werden - die Referenzwerte wurden mit einer älteren Version erstellt.

In Abbildung 4.9 deuten sich, im Vergleich zu MySQL, Schwächen in der Implementierung an: 60% IO-Wait deuten auf eine schlechte Speicherverwaltung hin, die sich bei den Page-Faults (siehe Abbildung 4.10) zu bestätigen scheint - was allerdings durch die neu erstellten Prozesse (siehe Abbildung 4.9) leicht in Frage gestellt wird. RDF-3X verursacht zehn mal mehr Page-Faults als MySQL und außerdem auch einige Major-Page-Faults, die das System zusätzlich negativ beeinflussen. Allgemein scheint RDF-3X nicht in der Lage zu sein alle zu Verfügung stehenden Ressourcen voll auszulasten. Für eine größtmögliche Leistung ist

	A1	**A2**	**A3**	**B1**	**B2**	**B3**	**C1**	**C2**	**C3**
cold caches									
Dell D620[NW08]	0.29	0.28	1.20	0.28	0.99	0.33	2.23	4.23	n.v.
Nokia N810	0.10	0.14	0.23	0.15	0.02	0.03	2.19	0.10	0.49
warm caches									
Dell D620[NW08]	0.02	0.02	0.02	0.01	0.05	0.01	0.61	1.44	n.v.
Nokia N810	0.03	0.01	0.07	0.02	0.00	0.00	0.41	0.01	0.04

Tabelle 4.16.: Vergleich zwischen Dell D620 und N810 für RDF-3X mit YAGO

dieses Verhalten schlecht - allerdings haben andere Anwendungen so mehr Ressourcen zur Verfügung, was auf einem Alltagsgerät sicherlich erwünscht ist.

5. Zusammenfassung und Ausblick

Im Rahmen dieser Fachstudie wurden verschiedene Datenbanksysteme mit verschiedenen Benchmarks auf ihre RDF-Tauglichkeit untersucht; dazu wurde mit dem Nokia N810 und der Maemo-Plattform gearbeitet.

Zu Beginn der Fachstudie kristallisierte sich schnell heraus, dass das Risiko dieser Arbeit in den Datenbanksystemen und Benchmarks bestand, die nicht für mobile Geräte konzipiert waren. Zu Beginn blind vor Neugier wurden die möglichen Probleme ignoriert und mit der Arbeit begonnen: die Fachstudie war eine Herausforderung. Man konnte so gut wie jeden Softwarebestandteil als Quelle erhalten, modifizieren und selbst wieder kompilieren. Zunächst begann eine Einarbeitungsphase in Maemo und RDF. Im Anschluss wurden parallel Datenbanken, Benchmarks und geeignete Werkzeuge gesucht und ausprobiert.

Je mehr man sich mit RDF-3X auseinandersetze, desto mehr wurde klar, dass die damit einhergehenden Probleme unterschätzt wurden. Im November 2009 zeichnete sich endgültig ab, dass der Terminplan aufgrund von technischen Schwierigkeiten nicht mehr haltbar sein würde. In Folge der Verzögerung und der explodierenden Stundenzahlen wurde eine Verlängerung beantragt. Diese Verlängerung wirkte sich negativ auf andere Studienbereiche, wie Studienprojekte und Vorlesungen aus - der private Bereich blieb ebenfalls nicht verschont. Dank einer motivierten aber auch hemdsärmeligen Gruppe, sowie einem verständnisvollen Betreuer konnte die Fachstudie dann aber doch abgeschlossen werden.

Die Lehre, die sich für die Beteiligten, aus dieser Fachstudie ergeben hat, lässt sich am Besten in Form von (zu spät gestellten) Fragen formulieren:

1. Ist es experimentelle Software?
2. Befinden sich die Software und die dazugehörigen Werkzeuge in der Entwicklung?
3. Ist der Quellcode und alle benötigen Dateien verfügbar?
4. Ist die Software gut dokumentiert?

Obwohl so gut wie nichts glatt lief war es eine interessante Fachstudie. Wäre alles glatt gelaufen, dann hätte man sicher nicht so viel dabei gelernt. Um nachfolgende Arbeiten mit dem Nokia N810 zu erleichtern, befinden sich in Kapitel 7 eine umfangreiche Sammlung von Anleitungen und Hinweisen.

Ausblick

Datenbanksysteme, die graphenartige Informationen verwalten erhalten, bereits jetzt einige Aufmerksamkeit: erste Hersteller von mobilen Geräten werben mit „voll vernetzten Adressbüchern“. Diese Geräte lassen sicher noch viele Wünsche offen. Beispielsweise könnte man Dateien ohne Ordnerstruktur oder Dateinamen ablegen. Über entsprechende Meta-Informationen (wann ein Dokument angelegt, wie es bearbeitet wurde oder was es enthält) ließe sich die Datei wieder finden. Unabhängig von den psychologischen Problemen bei Mensch-Maschinenschnittstellen, muss die Suche nach diesen Informationen schnell ablaufen; alles andere empfindet ein Benutzer als unhöflich.

RDF-3X bietet sehr gute Ansätze. Aufgrund der Rechenzeit und Größe der Datenbanken sollte die weiterführende Entwicklung von RDF-3X allerdings zunächst auf normalen Desktop-PCs stattfinden. Der Fokus der Entwicklung sollte bei einer stärkeren Orientierung am SPARQL Standard liegen um eine größtmögliche Kompatibilität zu anderen Systemen zu schaffen.

Teil II.

Anhang

6. Queries

6.1. BSBM-Queries für SQL

```
SELECT distinct nr, label
FROM product p, producttypeproduct ptp
WHERE p.nr = ptp.product AND ptp.productType=@ProductType@
       AND propertyNum1 > @x@
       AND p.nr IN (SELECT distinct product FROM productfeatureproduct WHERE
           productFeature=@ProductFeature1@)
       AND p.nr IN (SELECT distinct product FROM productfeatureproduct WHERE
           productFeature=@ProductFeature2@)
ORDER BY label
LIMIT 10;
```

Listing 6.1: BSBM Query 1

```
SELECT pt.label, pt.comment, pt.producer, productFeature, propertyTex1, propertyTex2,
    propertyTex3,
       propertyNum1, propertyNum2, propertyTex4, propertyTex5, propertyNum4
FROM product pt, producer pr, productfeatureproduct pfp
WHERE pt.nr=@ProductXYZ@ AND pt.nr=pfp.product AND pt.producer=pr.nr;
```

Listing 6.2: BSBM Query 2

```
SELECT p.nr, p.label
FROM product p, producttypeproduct ptp
WHERE p.nr=ptp.product
       AND productType=@ProductType@
       AND propertyNum1>@x@
       AND propertyNum3<@y@
       AND @ProductFeature1@ IN (SELECT productFeature FROM productfeatureproduct WHERE
           product=p.nr)
       AND @ProductFeature2@ NOT IN (SELECT productFeature FROM productfeatureproduct WHERE
           product=p.nr)
ORDER BY p.label
LIMIT 10;
```

Listing 6.3: BSBM Query 3

```
SELECT pt.label, pt.comment, pt.producer, productFeature, propertyTex1, propertyTex2,
    propertyTex3,
       propertyNum1, propertyNum2, propertyTex4, propertyTex5, propertyNum4
FROM product pt, producer pr, productfeatureproduct pfp
WHERE pt.nr=@ProductXYZ@ AND pt.nr=pfp.product AND pt.producer=pr.nr;
```

Listing 6.4: BSBM Query 4

```
SELECT distinct p.nr, p.label, p.propertyTex1
FROM product p, producttypeproduct ptp
WHERE p.nr=ptp.product AND ptp.productType=@ProductType@
```

```
        AND p.nr IN (SELECT distinct product FROM productfeatureproduct WHERE
            productFeature=@ProductFeature1@)
        AND ((propertyNum1>@x@ AND p.nr IN (SELECT distinct product FROM
            productfeatureproduct WHERE productFeature=@ProductFeature2@)
) OR (propertyNum2>@y@ AND p.nr IN (SELECT distinct product FROM productfeatureproduct WHERE
    productFeature=@ProductFeature3@)))
ORDER BY label
LIMIT 10
OFFSET 5;
```

Listing 6.5: BSBM Query 5

```
SELECT nr, label
FROM product
WHERE label like "%@word1@%";
```

Listing 6.6: BSBM Query 6

```
Wurde aufgrund von Inkompatibilitäten mit MySQL auf die ignore-list gesetzt.
```

Listing 6.7: BSBM Query 7

```
SELECT r.title, r.text, r.reviewDate, p.nr, p.name, r.rating1, r.rating2, r.rating3,
    r.rating4
FROM review r, person p
WHERE r.product=@ProductXYZ@ AND r.person=p.nr
       AND r.language='en'
ORDER BY r.reviewDate desc
LIMIT 20;
```

Listing 6.8: BSBM Query 8

```
SELECT p.nr, p.name, p.mbox_sha1sum, p.country, r2.nr, r2.product, r2.title
FROM review r, person p, review r2
WHERE r.nr=@ReviewXYZ@ AND r.person=p.nr AND r2.person=p.nr;
```

Listing 6.9: BSBM Query 9

```
SELECT distinct o.nr, o.price
FROM offer o, vendor v
WHERE o.product=@ProductXYZ@
       AND o.deliveryDays<=3 AND v.country='US'
       AND o.validTo>'@currentDate@' AND o.vendor=v.nr
Order BY o.price
LIMIT 10;
```

Listing 6.10: BSBM Query 10

```
Select product, producer, vendor, price, validFrom, validTo, deliveryDays, offerWebpage,
    publisher, publishDate
from offer
where nr=@OfferXYZ@;
```

Listing 6.11: BSBM Query 11

```
Select p.nr As productNr, p.label As productlabel, v.label As vendorname, v.homepage As
    vendorhomepage,
       o.offerWebpage As offerURL, o.price As price, o.deliveryDays As deliveryDays,
           o.validTo As validTo
From offer o, product p, vendor v
Where o.nr=@OfferXYZ@ AND o.product=p.nr AND o.vendor=v.nr;
```

Listing 6.12: BSBM Query 12

6.2. RDF-3X-Queries für YAGO

```
select ?GivenName ?FamilyName
where {
   ?GivenName <givenNameOf> ?p .
   ?FamilyName <familyNameOf> ?p .
   ?p <type> "wordnet_scientist_110560637" .
   ?p <bornIn> ?city .
   ?city <locatedIn> "Switzerland" .
   ?p <hasAcademicAdvisor> ?a .
   ?a <bornIn> ?city2 .
   ?city2 <locatedIn> "Germany" .
}
```

Listing 6.13: RDF-3X Query a1

```
select ?name
where {
   ?a <isCalled> ?name .
   ?a <type> "wordnet_actor_109765278" .
   ?a <livesIn> ?city .
   ?city <locatedIn> ?state .
   ?state <locatedIn> "United_States" .
   ?a <actedIn> ?m1 .
   ?m1 <type> "wordnet_movie_106613686" .
   ?m1 <produced> "Germany" .
   ?a <directed> ?m2 .
   ?m2 <type> "wordnet_movie_106613686" .
   ?m2 <produced> "Canada" .
}
```

Listing 6.14: RDF-3X Query a2

```
select distinct ?name ?country
where {
   ?p <isCalled> ?name.
   { ?p <type> "wordnet_actor_109765278" }
   union
   { ?p <type> "wordnet_athlete_109820263" }
   ?p <bornIn> ?city.
   ?city <locatedIn> ?state.
   ?state <locatedIn> ?country.
   ?p <type> ?t.
   FILTER(?t reaches "wordnet_politician_110451263" via <subClassOf>)
}
```

Listing 6.15: RDF-3X Query a3

```
select distinct ?name1 ?name2
where {
   ?a1 <isCalled> ?name1.
   ?a2 <isCalled> ?name2.
   ?a1 <livesIn> ?city1.
   ?a2 <livesIn> ?city2.
   ?city1 <locatedIn> "England".
   ?city2 <locatedIn> "England".
   ?a1 <actedIn> ?movie.
   ?a2 <actedIn> ?movie.
```

```
   FILTER (?a1 != ?a2)
}
```

Listing 6.16: RDF-3X Query b1

```
select ?name1 ?name2
where {
   ?p1 <isCalled> ?name1.
   ?p2 <isCalled> ?name2.
   ?p1 <isMarriedTo> ?p2.
   ?p1 <bornIn> ?city.
   ?p2 <bornIn> ?city.
}
```

Listing 6.17: RDF-3X Query b2

```
select distinct ?name1 ?name2
where {
   ?name1 <familyNameOf> ?p1.
   ?name2 <familyNameOf> ?p2.
   ?p1 <type> "wordnet_scientist_110560637".
   ?p2 <type> "wordnet_scientist_110560637".
   ?p1 <hasWonPrize> ?award.
   ?p2 <hasWonPrize> ?award.
   ?p1 <bornIn> ?city.
   ?p2 <bornIn> ?city.
   FILTER (?p1 != ?p2)
}
```

Listing 6.18: RDF-3X Query b3

```
select distinct ?name1 ?name2
where {
   ?name1 <familyNameOf> ?p1.
   ?name2 <familyNameOf> ?p2.
   ?p1 <type> "wordnet_scientist_110560637".
   ?p2 <type> "wordnet_scientist_110560637".
   ?p1 [] ?city.
   ?p2 [] ?city.
   ?city <type> <wordnet_site_108651247>
   FILTER (?p1 != ?p2)
}
```

Listing 6.19: RDF-3X Query c1

```
select distinct ?name
where {
   ?p <isCalled> ?name.
   ?p [] ?c1.
   ?p [] ?c2.
   ?c1 <type> <wordnet_village_108672738>
   ?c2 <type> <wordnet_site_108651247>
   ?c1 <isCalled> "London".
   ?c2 <isCalled> "Paris".
}
```

Listing 6.20: RDF-3X Query c2

```
select ?p1 ?predicate ?p2
where {
   ?p1 [] ?city1.
```

```
   ?city1 <isCalled> "Paris".
   ?p1 ?predicate ?p2.
   ?p2 [] ?city2.
   ?city2 <isCalled> "Hong Kong".
}
```

Listing 6.21: RDF-3X Query c3

7. Anleitungen

It's a kind of magic,
Magic, magic, magic, MAGIC!

(Queen)

In diesem Kapitel werden verschiedene Vorgehensweisen beschrieben, die mit dem Betrieb und der Entwicklung für ein Nokia N810 verbunden sind. Dazu gehören u.a. auch die Installation der Entwicklungsumgebungen und der Werkzeuge.

Achtung: Benutzen auf eigene Gefahr!

7.1. Flashen eines N810 / Neuinstallation von OS2008

Wenn man einen schwerwiegenden Fehler gemacht hat, wie zum Beispiel das komplette Root-Verzeichnis rekursiv zu löschen, wird es notwendig sein, das N810 zu „flashen". Das bedeutet, auf das Gerät das Betriebssystem neu aufzuspielen und es damit auf den Auslieferungszustand zurückzusetzen.

Achtung: Dieses Vorgehen löscht ebenfalls die externe Speicherkarte. Enthält diese wichtige Daten, sollte diese zuvor entfernt werden.

Achtung: Vor diesem Vorgang sollte unbedingt sichergestellt sein, dass man die korrekte Hardware-Version hat. Verwendet man stattdessen die falsche Version, so wird das Gerät unbrauchbar. Es wird nicht mehr von dem angeschlossenen Computer erkannt und kann somit auch nicht mehr geflasht werden.

Schritt 1: Das Ladekabel entfernen und das Gerät ausschalten. Sollte dies nicht durch langes Drücken auf den Ausschaltknopf zum Erfolg führen, bleibt nur noch übrig den Akku herauszunehmen.

Schritt 2: Das aktuelle Image des Betriebssystems *Maemo OS2008* und das Flasher-Programm von der Seite `http://tablets-dev.nokia.com/nokia_N810.php` herunterladen. Wir verwendeten den Flasher in der Version 3.5. Dazu muss die Identifikationsnummer (im WLAN-Schlüssel kodiert) angegeben werden.

Schritt 3: Lokales Entpacken der Pakete.

Schritt 4: Starten des Flashers mit dem Befehl `> sudo ./flasher3.5 -F <Name des Images> -f -R -h <hardware:version>`. Die `hardware:version` befindet sich hinter dem Akku und nur diese sollte verwendet werden.

Schritt 5: Anschließen des mobilen Gerätes an den Computer.

Schritt 6: Einschalten des mobilen Gerätes. Um in den USB-Modus zu gelangen, muss die „Home"-Taste (zwei überlappende Fenster auf der Frontseite) und der Einschaltknopf gleichzeitig gedrückt werden. War der Vorgang erfolgreich, erscheint in der rechten oberen Ecke ein USB-Symbol.

Schritt 7: Nun sollte das mobile Gerät vom Computer erkannt und der Flash-Vorgang automatisch gestartet werden. Nach einem erfolgreichen Update wird das Gerät neu gestartet.

7.2. Fernzugriff

Ein mobiles Gerät ist sehr praktisch – zumindest solange man nicht damit entwickeln muss. Der Touchscreen und der beiliegende Stift, sowie die sehr eingeschränkte Tastatur, machen das Arbeiten mehr als unbequem. Es ist sehr von Vorteil, wenn man per „Fernbedienung" auf das mobile Gerät zugreifen kann!

Die einzelnen Bestandteile des Fernzugriffs werden nachfolgend erläutert.

ssh (Secure SHell) wird benötigt, um eine sichere Verbindung auf ein entferntes Gerät aufzubauen. (siehe 7.3.1)

scp (Secure CoPy) ermöglicht das, auf SSH basierende, Kopieren von Daten von lokalen zu entfernten System.

sshfs (SSH FileSystem) kann ein entferntes Dateisystem in das lokale einhängen.

vnc (Virtual Network Computing) kann die grafische Oberfläche von einem anderen Gerät aus steuern. Da das mobile Gerät allerdings keinen Cursor hat, kann man es nicht mit einem konventionellen Laptop bedienen, denn das Maussignal wird nicht interpretiert.

7.3. Programme auf dem mobilen Gerät

Hier werden die Programme und deren Installation beschrieben, die auf dem mobilen Gerät verwendet wurden.

7.3.1. SSH-Server

Leider ist beim Basissystem des N810 kein SSH-Server vorinstalliert. Um eine SSH-Verbindung auf ein N810 aufzubauen muss zuerst ein SSH-Server installiert werden. Der Aufruf erfolgt durch den Befehl `>ssh root@<IP des Geräts>`

Hinweis: Um auf den SSH-Server zugreifen zu können, muss zunächst vom mobilen Gerät aus der Befehl `> ping <IP des Hosts>` auf den aufrufenden Host ausgeführt werden, da das mobile Gerät sonst die Verbindungen ablehnt. Hierbei handelt es sich vermutlich um einen Bug.

Installation

Schritt 1: Internetverbindung herstellen.

Schritt 2: OpenSSH-Paket herunterladen. Auf der Seite `http://maemo.org` den Punkt *Downloads* auswählen, dann OpenSSH suchen und mit einem Klick auf *install* das Paket OpenSSH Client und Server installieren.

Schritt 3: Den Anweisungen am Bildschirm folgen. Dabei wird man auch aufgefordert ein Passwort für den Benutzer `root` zu setzen.

7.3.2. Uptime, Sleep

`uptime` ist ein Programm, das ausgibt, wie lange das System bereits gestartet ist. Das GNU `sleep` veranlasst, im Gegensatz zu dem vorinstallierten `sleep` von Nokia, das System *X* Millisekunden lang zu warten. Beide Programme sind in den *GNU Core Utils* kurz: *coreutils* (`http://www.gnu.org/software/coreutils/`) enthalten.

Installation

Schritt 1: Internetverbindung herstellen.

Schritt 2: Herunterladen des Pakets unter `http://ftp.gnu.org/gnu/coreutils/`.

Schritt 3: Entpacken in der Scratchbox.

Schritt 4: Kompilieren mit `make`.

Schritt 5: Auf mobiles Gerät übertragen.

7.3.3. Killall

`killall` ist ein Programm, das eine Reihe von Prozessen beendet. Es ist im Paket *PSmisc* enthalten (`http://psmisc.sourceforge`).

verwendete Version PSmisc 22.8 (aktuell)

Schritt 1: Internetverbindung herstellen.

Schritt 2: Herunterladen des Pakete unter `http://sourceforge.net/projects/psmisc/`.

Schritt 3: Entpacken in der Scratchbox.

Schritt 4: Kompilieren mit `make`.

Schritt 5: Auf mobiles Gerät übertragen.

7.3.4. Dash

dash (Debian-Almquist-SHell) ist ein Kommandozeileninterpreter. Dieser wird benötigt um „spezielle“ Shell-Skripte auszuführen, die die vorinstallierte Shell nicht interpretieren kann.

Bezugsquelle: `http://gondor.apana.org.au/~herbert/dash/`

7.4. Programme auf dem Host-System

In diesem Abschnitt finden sich Anleitungen zu den Programmen, die auf dem / den ausführenden Rechner(n) installiert wurden.

7.4.1. Maemo SDK und Scratchbox

Die Maemo SDK ist das eigentliche Herzstück bei der Entwicklung von Software für Maemo. Sie enthält ebenfalls einen Cross-Compiler für die ARMv6-Architektur des mobilen Geräts. Die Scratchbox ist Teil der Entwicklungsumgebung. Sie ist im Prinzip eine „Sandbox“ (sinngemäß und frei übersetzt: Wanderdüne), die das mobile Geräts simuliert. Das Host-System wird durch das Sandbox-Konzept (theoretisch) nicht verändert.

verwendete Versionen:

Maemo SDK 4.1.2 Diablo

Scratchbox Scratchbox R4 Apophis

Warnung Die Scratchbox bindet virtuelle Laufwerke ein und verbaut *hard-links* ins gesamte System, was beim manuellen, rekursiven Entfernen dieser doch beträchtlichen Schaden anrichtet. Für eine ungefährliche Lösung, siehe 7.4.2.

Installation Voraussetzung ist eine große Festplatte, und ein Rechner mit x86-Linux. Sollten Sie eine x64-Architektur und einen Kernel neuer als 2.6.25 besitzen, so sollten Sie zuerst den Computer mit dem Kernel Boot-Parameter `'vdso32=0'` neu starten, da die Scratchbox sonst den Dienst verweigert.

Referenzseite: `http://repository.maemo.org/stable/diablo/`

Schritt 1 Legen Sie ein Backup an!

Schritt 2 Herunterladen der Installer *maemo-sdk-install_4.1.2.sh* und *maemo-scratchbox-install_4.1.2.sh*, welche Sie anschließend mit `chmod a+x` ausführbar machen.

Schritt 3 Führen Sie zunächst den Scratchbox-Installer aus:
`sudo ./maemo-scratchbox-install_4.1.2.sh`

Schritt 4 Holen Sie sich einen Kaffee und warten Sie. Mit etwas Glück läuft dar Installer ohne Probleme durch und Scratchbox befindet sich im Verzeichnis `/scratchbox`. Ansonsten wird in der `Install.txt` in Abschnitt 1.2 beschrieben, wie man die Scratchbox manuell installieren kann.

Schritt 5 Fügen Sie den Benutzer `USER` der Gruppe „scratchbox“ hinzu, indem Sie `./scratchbox -u USER` ausführen.

Schritt 6 Führen Sie den Befehl `newgrp sbox` aus.

Schritt 7 Für die Installation des Maemo SDK führen Sie den Befehl `sh maemo-sdk-install_4.1.2.sh` aus. Wichtig ist hierbei, dies nicht als Root zu tun.

„...Happy Hacking!“ (Zitat: Scratchbox-Installationsskript)

7.4.2. Maemo SDK Virtual Image und VirtualBox

Als Alternative zur oben genannten Installation, gibt es auch die Möglichkeit ein virtuelles Ubuntu (genannt *Maemo SDK Virtual Image* aufzuspielen, das alle nötigen Pakete und eine vorinstallierte Maemo SDK, sowie eine Scratchbox bereits enthält. Das Image kann von der Seite `http://maemovmware.garage.maemo.org/2nd_edition/` heruntergeladen werden.

Die VirtualBox von Sun (`http://www.virtualbox.org/`) ist nach unserer Meinung die für diese Zwecke brauchbarste Virtualisierungssoftware.

Vorteile

- ein Image greift nur wenig, wenn überhaupt, in die Struktur des Hostsystems ein (es sei denn, man erlaubt es ihm)
- es sind alle Programme vorinstalliert, die benötigt werden
- man muss sich nicht über die falsche Architektur des Hostsystems kümmern: Scratchbox läuft nur unter einem 32bit-System ordentlich.

Nachteile

- durch die Virtualisierung langsamer
- hat man einen Computer mit 64bit-Architektur kann man nicht in dem simulierten 32bit-System die 64bit-Erweiterungen der 64bit-Architektur nutzen.

Installation Die Installation unter Windows geschieht über den Installer, den man von der Seite herunterladen kann. Unter Linux gibt es, auf derselben Seite für verschiedene Linux-Distributionen, entsprechende Pakete, die in entsprechender Weise installiert werden können.

Um das Image einzubinden, wird zuerst eine neue Session erstellt: als Betriebssystem `Ubuntu 32bit` auswählen und als Festplatte das Image verwenden.

Hinweis: Die OpenSource-Variante von VirtualBox (VirtualBox OSE) beherrscht keinen Zugriff auf USB-Ports, weshalb die proprietäre Variante verwendet werden sollte.

Hinweis: Möchte man tatsächlich den USB-Port unter Linux verwenden, muss man die Rechte der Scratchbox anpassen, damit sie auch Zugriff auf die USB-Geräte bekommt. Alternativ die Scratchbox mit `sudo VirtualBox &` ausführen. Ein genaueres Vorgehen wird in der gut dokumentierten Hilfe erläutert.

7.4.3. sshfs

sshfs wird dafür verwendet, ein entferntes Gerät als lokales Verzeichnis zu mounten.

Der `<Name des Einhängepunkts>` muss bereits vorhanden sein. Der Aufruf ist hierbei:

`sshfs root@<IP des Geräts> /media/<name des Einhängepunkts>`.

Installation über `sudo apt-get install sshfs`

7.4.4. gnuplot

Nachdem unsere anderen Versuche scheiterten (u.a. PGF und Ti*k*z) schöne Diagramme zu erstellen, sind wir letzten Endes auf Gnuplot umgestiegen.

Installation wie gewohnt.

7.4.5. LaTeX

Alle Dokumente wurden mit LaTeX erstellt – für schöne Arbeiten ein Muss. Allerdings konnten wir uns nicht einigen, welchen Editor oder welchen Paketmanager wir verwenden sollten. So kamen die Manager TeX Live und MikTeX, sowie die Editoren Kate, TeXnicCenter und TeXMaker zum Einsatz. Dank LaTeX war das aber kein Problem.

Installation wie gewohnt.

Literaturverzeichnis

[AMMH07] Abadi, Daniel J. ; Marcus, Adam ; Madden, Samuel R. ; Hollenbach, Kate: Scalable Semantic Web Data Management Using Vertical Partitioning / MIT. 2007. – Forschungsbericht (Zitiert auf Seite 10)

[Bal09] Baldwin, Andrew: *Exploring Qt Performance on ARM using FineToothComb*. http://labs.trolltech.com/blogs/2009/09/29/exploring-qt-performance-on-arm-using-finetoothcomb/. Version: 2009 (Zitiert auf den Seiten 5, 28 und 30)

[BS09] Bizer, Christian ; Schultz, Andreas: The Berlin SPARQL Benchmark / Freie Universität Berlin, Web-based Systems Group, Garystr. 21, 14195 Berlin, Germany. 2009. – Forschungsbericht (Zitiert auf den Seiten 10 und 22)

[EM] Erling, Orri ; Mikhailov, Ivan: RDF Support in the Virtuoso DBMS. – Forschungsbericht (Zitiert auf Seite 10)

[Gro] Grothoff, Christian: *GNU libmicrohttpd*. http://www.gnu.org/software/libmicrohttpd/ (Zitiert auf Seite 48)

[hdp] *hdparm at SourceForge.net*. http://sourceforge.net/projects/hdparm/ (Zitiert auf Seite 24)

[maea] *Maemo 5 Downloads*. http://maemo.org/downloads/Maemo5 (Zitiert auf Seite 16)

[maeb] *Maemo OS2008 Downloads*. http://maemo.org/downloads/OS2008/ (Zitiert auf Seite 16)

[NW08] Neumann, Thomas ; Weikum, Gerhard: RDF3X: a RISCstyle Engine for RDF / Max-Planck Institute for Informatics, Saarbrücken, Germany. 2008. – Forschungsbericht (Zitiert auf den Seiten 10, 48, 54 und 55)

[NW09a] Neumann, Thomas ; Weikum, Gerhard: The RDF-3X Engine for Scalable Management of RDF Data / Max-Planck-Institut für Informatik. Saarbrücken, Germany, March 2009 (MPI-I-2009-5-003). – Forschungsbericht. – 58 S. – ISBN 0946–011X (Zitiert auf den Seiten 10 und 24)

[NW09b] Neumann, Thomas ; Weikum, Gerhard: Scalable Join Processing on Very Large RDF Graphs / Max-Planck Institute for Informatics, Saarbrücken, Germany. 2009. – Forschungsbericht (Zitiert auf Seite 10)

[SGK+08] Sidirourgos, Lefteris ; Goncalves, Romulo ; Kersten, Martin ; Nes, Niels ; Manegold, Stefan: ColumnStore Support for RDF Data Management: not all swans are white / CWI, Amsterdam, The Netherlands. 2008. – Forschungsbericht (Zitiert auf Seite 10)

[SKW07] Suchanek, Fabian M. ; Kasneci, Gjergji ; Weikum, Gerhard: Yago: A Core of Semantic Knowledge. In: *16th international World Wide Web conference (WWW 2007)*. New York, NY, USA : ACM Press, 2007 (Zitiert auf Seite 24)

[sys] *SYSSTAT Homepage*. http://pagesperso-orange.fr/sebastien.godard/ (Zitiert auf Seite 25)

[tpc] *TPC Pricing Specification*. http://www.tpc.org/pricing/ (Zitiert auf Seite 24)

[Ven09] Ven, Arjan van d.: *Introducing "timechart"*. http://blog.fenrus.org/?p=5. Version: 2009 (Zitiert auf Seite 28)

[w3-08] *SPARQL Endpoint Description*. http://esw.w3.org/topic/SparqlEndpointDescription. Version: 2008 (Zitiert auf Seite 22)

[w3c] *Resource Description Framework*. http://www.w3.org/RDF/ (Zitiert auf Seite 13)

[yag] *YAGO Homepage*. http://www.mpi-inf.mpg.de/yago-naga/yago/ (Zitiert auf Seite 49)

Alle URLs wurden zuletzt am 10.02.2010 geprüft.

Erklärung

Hiermit versichern wir, diese Arbeit
selbständig verfasst und nur die angegebenen
Quellen benutzt zu haben.

_____

(Dirk Coehne,Gregor Rothmaier,Christoph Schulz)