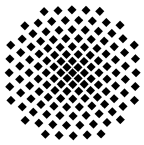

H L R S

Institut für
Höchstleistungsrechnen

FORSCHUNGS- UND ENTWICKLUNGSBERICHTE

ENERGIEEFFIZIENZ VON PROZESSOREN IN HIGH PERFORMANCE COMPUTING-ANWENDUNGEN DER INGENIEUR-WISSENSCHAFTEN

Dmitry Khabi

ISSN 0941 - 4665 November 2018 HLRS-19

Höchstleistungsrechenzentrum
Universität Stuttgart
Prof. Dr.-Ing. Dr. h.c. Dr. h.c. Prof. E.h. Michael M. Resch
Nobelstrasse 19 - 70569 Stuttgart
Institut für Höchstleistungsrechnen

ENERGIEEFFIZIENZ VON PROZESSOREN IN HIGH PERFORMANCE COMPUTING-ANWENDUNGEN DER INGENIEUR-WISSENSCHAFTEN

von der Fakultät Energie-, Verfahrens- und Biotechnik
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von

Dmitry Khabi
aus Moskau, Russland

Hauptberichter:	Prof. Dr.-Ing. Dr. h.c. Dr. h.c. Prof. E.h. Michael M. Resch
MItberichter:	Prof. Dr. Thomas Ludwig
Tag der Einreichung:	19.12.2018
Tag der mündlichen Prüfung:	07.10.2019
CR-Klassifikation:	I.3.2, I.6.6

D93

ISSN 0941 - 4665 November 2018 HLRS-19

Danksagung

Ohne die Unterstützung von vielen Seiten wäre diese Arbeit nicht möglich gewesen, und so möchte ich vor allem meinen Eltern, Olga Godunova und Villi Khabi, und meiner Schwester Polina Khabi, für ihre hilfreiche Unterstützung und ihr Verständnis sowohl während meines Studiums der Angewandten Mathematik in Moskau und der Informatik in Stuttgart, als auch während der Anfertigung dieser Doktorarbeit danken.

Mein herzlicher Dank gilt Herrn Dipl.-Math. Uwe Küster, der ein hervorragender „Chef“ war und mich in jeder Art unterstütze und wissenschaftliche Freiheit ermöglichte, die zu den vorliegenden Ergebnissen geführt hat. Seine stets positive und motivierende Betreuung hat mich in meinem Schreib- und Forschungsprozess unterstützt und bekräftigt.

Ebenso danke ich Herrn Professor Michael M. Resch, der mir durch die Anstellung am HLRS als wissenschaftlichen Mitarbeiter die Möglichkeit und Freiheit eröffnete, mich dem in dieser Arbeit dargestellten Forschungsbereich zu widmen.

Ebenso geht mein Dank an meine Kollegen und Kolleginnen, die mich in den vergangenen Jahren mit bereichernden Tipps und Diskussionsbeiträgen wiederholt in neue fruchtbare thematische Bahnen gelenkt haben.

Dmitry Khabi Stuttgart im November 2018

Inhaltsverzeichnis

Inhaltsverzeichnis v

Abbildungsverzeichnis xi

Tabellenverzeichnis xvii

Abkürzungsverzeichnis xxiv

Zusammenfassung xxv

Abstract xxvi

1. Einleitung 1

1.1. Motivation 1

1.2. Stand der Technik 1

1.2.1. Elektrische Leistung eines Halbleiterbauelements 1

1.2.2. Dynamic Voltage und Frequency Scaling 2

1.2.3. Frequency Scaling und Stromverbrauch 5

1.2.4. Messung vom Stromverbrauch 6

1.2.5. Anwendungen im Hochleistungsrechner 7

2. Messsystem für Testcluster 9

2.0.1. Prozessor- und Speicherkonfiguration 10

2.1. Messsystem 11

2.1.1. Messsystem-Integration in Testcluster-Umgebung 11

2.1.2. Messsystem-Integration in Rechenknoten 12
2.1.3. Aufzeichnung der elektrischen Leistung mit A/D-Wandler . . . 12
2.1.4. Filterung 14
2.2. Pulsweitenmodulation eines Schaltnetzteiles 14
2.2.1. Spannungsverlauf mit Oszilloskop 15
2.2.2. Verlauf der elektrischen Leistung 16

3. Elektrische Leistung im Betrieb ohne Rechenlast 19
3.1. Spannungs- und Stromverlauf 19
3.1.1. Spannungs- und Stromverlauf (Haswell) 20
3.1.2. Spannungs- und Stromverlauf (Ivy Bridge) 21

4. Elektrische Leistung und Rechenleistung der Kernel-Operation *Add* 23
4.1. Grundlagen der Approximation der elektrischen Leistung 24
4.2. Bestimmung der Koeffizienten 26
4.3. Grundlagen der Approximation der Rechenleistung 27
4.4. Kernel-Operation *Add* 29
4.4.1. Implementierung 29
4.5. Approximation der Kernel-Operation *Add* 32
4.5.1. First Level Cache 32
4.5.2. Second Level Cache 40
4.5.3. Last Level Cache 45
4.5.4. Hauptspeicher 55

5. Modellierung der Kernel-Ausführung 61
5.1. Execution-Cache-Memory Modell (ECM) 61
5.1.1. Grundlagen des ECM-Modells 61
5.1.2. Modellierung der Ausführungszeit mit ECM 62

5.1.3. Vergleich der theoretischen und experimentellen Ergebnisse . . 70
5.1.4. Bandbreite im Falle des Haltens der Daten im Cache 71
5.2. Prozessorkomponenten und Frequenzen 75
5.2.1. Prozessorkern und Frequenz f_{CPU} 76
5.2.2. Ringbus und Frequenz f_{RING} 76
5.2.3. Speichercontroller und Frequenz f_{IMC} 77
5.2.4. Speicher und Frequenz f_{MEM} 77
5.3. Data-Transfer-Memory Modell (DTM) 78
5.3.1. Anwendung des DTM-Modells 81
5.3.2. Relativer Fehler und Komponente des DTM-Modells 82

6. Energieverbrauch eines Höchstleistungsrechners 87

6.1. Höchstleistungsrechner Cray XC40 (Hazel Hen) 87
6.1.1. Messsystem in Cray XC40 89
6.2. Kernel-Operation Add im Hauptspeicher auf Hazel Hen 90
6.2.1. Rechenleistung auf Hazel Hen 90
6.2.2. Elektrische Leistung von „Hazel Hen“ 91
6.3. Kernel-Operation PETsC-CG auf „Hazel Hen“ 93
6.3.1. Problembeschreibung der Kernel-Operation PETsC-CG 94
6.3.2. Matrix- und Vektorverteilung 94
6.3.3. Matrix-Vektor-Multiplikation 95
6.3.4. Rechenleistung der Kernel-Operationen Add und PETsC-CG auf „Hazel Hen“ 96
6.3.5. Elektrische Leistung der Kernel-Operationen Add und PETsC-CG auf „Hazel Hen“ 98

7. Zusammenfassung und Ausblick 101

7.1. Messsystem für Testcluster 101

7.2. Elektrische Leistung im Betrieb ohne Rechenlast 101
7.3. Elektrische Leistung und Rechenleistung der Kernel-Operation *Add* . 102
7.4. Modellierung der Kernel-Ausführung 105
7.5. Energieverbrauch eines Höchstleistungsrechners 106

Literaturverzeichnis **109**

Anhänge 117

A. Testcluster **117**

I. Hardwarekomponente 117
I.I. Technische Beschreibung des Testclusters 118
I.II. Messgruppen des Testclusters 119
I.III. Schaltnetzteile des Testclusters 120

B. Messeinrichtung **123**

I. A/D-Wandler 123
II. Strommessung mit einem Messwiderstand 124
III. Spannungsmessung mit einem Spannungsteiler 125
IV. Hochlast-Widerstand RB50 125
V. Messung mit Akku 126
VI. Messsystem-Integration in Rechenknoten 127
VII. Messung in Wechselstrom 128
VIII. Kalibrierungsfehler 131
IX. Spannungsverlauf am Hochlast-Widerstand 131
IX.I. Spannungsverlauf und Schaltnetzteil 132
IX.II. Spannungsverlauf und Akkumulator 133

IX.III. Einfluss der Messfrequenz auf Spannungsmessung 135
X. Stromverlauf 136
X.I. Stromverlauf und Schaltnetzteil 137
X.II. Stromverlauf und Akkumulator 138

C. Oszillogramme **141**

I. Oszillogramm für Spannungssignal am Schaltnetzteil 141
II. Oszillogramme für Spannungssignal am Spannungsteiler 144
III. Oszillogramme für Stromsignal 147

D. Betrieb ohne Rechenlast **151**

I. Zähler für Betrieb ohne Rechenlast 151
II. Spannungs- und Stromverlauf (Haswell) 152
III. Spannungs- und Stromverlauf (Ivy Bridge) 154
IV. Akkumulator als Stromquelle 156
IV.I. Prozessor Ivy Bridge 156
IV.II. Prozessor Haswell 158
V. Elektrischen Leistung der Rechenknoten 159
VI. Vergleich zwischen RAPL-Zähler und A/D-Messung 164

E. Approximation der Kernel-Operation *Add* **165**

I. Fall des Haltens der Daten im Cache 165
II. Hauptspeicher 169
II.I. Speicherzugriff im „Temporal“-Modus 169
II.II. Rechenleistung 169
II.III. Elektrische Leistung 173

F. Optimierter Pipeline-Verlauf (ECM) **175**

G. Speichermodul-Architektur **177**

I. SDRAM-Modul 177

II. SDRAM-Chip 179

III. SDRAM-Steuerung 180

IV. Burst-Modus 182

V. Erweiterung für DDR4-SDRAM 183

VI. Speicherzellen-Auffrischung 184

H. Modellierte Bandbreite im Falle des Haltens der Daten im Cache **185**

I. Effizienz des Datentransfers im Falle des Haltens der Daten im Cache 186

II. DTM-Modell im Falle des Haltens der Daten im L2-Cache 187

III. DTM-Modell im Falle des Haltens der Daten im L3-Cache 188

I. Hauptspeicher-Frequenz **191**

I. Einfluss der Hauptspeicher-Frequenz auf die Bandbreite 191

II. Einfluss der Hauptspeicher-Frequenz auf die elektrische Leistung . . . 192

J. DTM-Modell **195**

I. DTM-Modell für die Kernel Add3 und Copy 196

Abbildungsverzeichnis

1.1. Abhängigkeit zwischen der Core-Spannung und der CPU-Frequenz . . 4

2.1. Funktionales Blockschema der Messung der elektrischen Leistung. . . 13

2.2. Oszillogramme mit den Verlaufsgraphen für die Spannung 16

2.3. Histogramm der elektrischen Leistungsverlauf 17

2.4. Verlauf der elektrischen Leistung und das Histogramm der Messdaten für einen Akkumulator 17

3.1. Betrieb ohne Rechenlast: Spannungsverlauf und Stromverlauf für *Haswell* 20

3.2. Betrieb ohne Rechenlast: Spannungs- und Stromverlauf für *Ivy Bridge* 21

3.3. Betrieb ohne Rechenlast: Stromverlauf für *Ivy Bridge* 22

4.1. Rechenleistung im Falle *L1-Cache* 33

4.2. *Ivy Bridge*: Elektrische Leistung im Falle *L1-Cache* 35

4.3. *Haswell*: Elektrische Leistung im Falle *L1-Cache* 35

4.4. *Ivy Bridge*: Die Energiekosten im Falle *L1-Cache* 39

4.5. *Haswell*: Die Energiekosten im Falle *L1-Cache* 39

4.6. Rechenleistung im Falle *L2-Cache* 40

4.7. *Ivy Bridge*: Elektrische Leistung im Falle *L2-Cache* 42

4.8. *Haswell*: Elektrische Leistung im Falle *L2-Cache* 42

4.9. Energiekosten im Falle *L2-Cache* 44

4.10. Rechenleistung im Falle *L3-Cache* 46

4.11. Rechenleistung im Falle *L3-Cache* 46
4.12. Erste Ableitung der Rechenleistung nach der CPU-Frequenz 47
4.13. *Ivy Bridge*: Elektrische Leistung im Falle *L3-Cache* 49
4.14. *Haswell*: Elektrische Leistung im Falle *L3-Cache* 49
4.15. „Uncore“-Frequenz und CPU-Frequenz im Falle *L3-Cache* 50
4.16. *Ivy Bridge*: Energiekosten im Falle *L3-Cache* 54
4.17. *Haswell*: Energiekosten im Falle *L3-Cache* 54
4.18. *Ivy Bridge*: Rechenleistung im Falle *Hauptspeicher* 55
4.19. *Haswell*: Rechenleistung im Falle *Hauptspeicher* 56
4.20. *Ivy Bridge*: Elektrische Leistung im Falle *Hauptspeicher* 57
4.21. *Haswell*: Elektrische Leistung im Falle *Hauptspeicher* 57
4.22. Energiekosten im Falle *Hauptspeicher* 59
4.23. Energiekosten und Rechenleistung im Falle *Hauptspeicher* 60

5.1. Prinzipieller Aufbau einer Pipeline im *Haswell*-Kern 63
5.2. *Single-Instruction-Multiple-Data* (SIMD) Addition 65
5.3. Speicher-Hierarchiestufen in *ECM*-Modell 68
5.4. *Ivy Bridge*: Bandbreite der Kernel-Operation *Add* 74
5.5. *Haswell*: Bandbreite der Kernel-Operation *Add* 74
5.6. Speicher-Hierarchiestufen in *DTM*-Modell 75
5.7. Vergleich zwischen ECM- und DTM-Modellen 82
5.8. Fehler des DTM-Modells 83
5.9. Komponenten des DTM-Modells für 1.2 GHz, 1.9 GHz und Turbo-Mode 85

6.1. *Cray XC40*-Blade 89
6.2. Hazel Hen: Rechenleistung der Kernel-Operation *Add* 90
6.3. Hazel Hen: Elektrische Leistung der Kernel-Operation *Add* 91

6.4. Vergleich: „Hazel Hen“und „node03“ 93

6.5. Hazel Hen: Rechenleistung der Kernel-Operation *PETsC* 96

6.6. Hazel Hen: Elektrische Leistung der Kernel-Operation *PETsC* 98

A.1. Testcluster 117

B.1. Elektrisches Schema der Messung am Messwiderstand 124

B.2. Elektrisches Schema der Messung am Spannungsteiler 125

B.3. Messeinrichtung mit Hochlast-Widerstand *RB50-2R2-J* 2.2 Ω 126

B.4. Anwendung eines Lithium-Polymer-Akkumulators für CPU-Versorgung127

B.5. Stromversorgung und Messeinrichtung für CPU und Speicher 128

B.6. Messeinrichtung am AC-Kaltgerätekabel 129

B.7. Spannungs- und Stromverlauf am AC-Kaltgerätekabel 129

B.8. Messeinrichtung für Spannungsverlauf am AC-Kaltgerätekabel 130

B.9. Messeinrichtung für Stromverlauf am AC-Kaltgerätekabel 131

B.10. Spannungsverlauf: Vergleich zwischen zwei Schaltnetzteilen 132

B.11. Histogramm des Spannungsverlaufs: Vergleich zwischen zwei Schaltnetzteilen 133

B.12. Spannungsverlauf und das Histogramm der Messdaten für einen Akkumulator 134

B.13. Spannungsverlauf: Vergleich zwischen zwei Messfrequenzen 135

B.14. Stromverlauf: Vergleich zwischen zwei Schaltnetzteilen 137

B.15. Histogramm des Stromsverlaufs im Messkreis mit einem Schaltnetzteil 138

B.16. Stromverlauf und das Histogramm der Messdaten für einen Akkumulator139

C.1. Oszillogramm für Spannungssignal im Mode „Peak detect“ 141

C.2. Oszillogramm für Spannungssignal im Mode „Peak Detect Mode to Find a Glitch“ 142

C.3. Oszillogramm für Spannungssignal im Mode „High resolution“ 142

C.4. Oszillogramm für Spannungssignal im Mode „Average over 16 samples“ 143

C.5. Oszillogramm für Spannungssignal im Mode „Average over 4096 samples“ 143

C.6. Oszillogramm für Spannungssignal am Spannungsteiler im Mode „Peak detect“ 144

C.7. Oszillogramm für Spannungssignal am Spannungsteiler im Mode „Peak Detect Mode to Find a Glitc“ 145

C.8. Oszillogramm für Spannungssignal am Spannungsteiler im Mode „High resolution“ 145

C.9. Oszillogramm für Spannungssignal am Spannungsteiler im Mode „Average over 16 samples“ 146

C.10. Oszillogramm für Spannungssignal am Spannungsteiler im Mode „Average over 4096 samples“ 146

C.11. Oszillogramm für Stromsignal im Mode „Peak detect“ 147

C.12. Oszillogramm für Stromsignal im Mode „Peak Peak Detect Mode to Find a Glitch“ 148

C.13. Oszillogramm für Stromsignal im Mode „High resolution“ 148

C.14. Oszillogramm für Stromsignal im Mode „Average over 4 samples“ . . 149

C.15. Oszillogramm für Stromsignal im Mode „Average over 4096 samples“ 149

D.1. Histogramme für Spannungs- und Stromverläufe 152

D.2. Betrieb ohne Rechenlast: Verlauf der elektrischen Leistung der zwei Prozessoren *Haswell* ohne Filterung 153

D.3. Betrieb ohne Rechenlast: Verlauf der elektrischen Leistung der zwei Prozessoren *Haswell* 153

D.4. Betrieb ohne Rechenlast: Verlauf der elektrischen Leistung der zwei Prozessoren *Haswell* 154

D.5. Stromverlauf des ersten Prozessors E5-2690v2 auf *node02* während des Betriebes ohne Rechenlast. Der durchschnittliche Stromwert beträgt 1.64 A. Die Messfrequenz beträgt 50 kHz. 155

D.6. Betrieb ohne Rechenlast: Stromverlauf für *Ivy Bridge* 155

D.7. Betrieb ohne Rechenlast: Verlauf der elektrischen Leistung der zwei Prozessoren *Ivy Bridge* 156

D.8. Histogramme für Spannungsverlauf: Vergleich zwischen Schaltnetzteil und Akkumulator für *Ivy Bridge* 157

D.9. Histogramme für Stromverlauf: Vergleich zwischen Schaltnetzteil und Akkumulator für *Ivy Bridge* 157

D.10. Histogramme für Spannungsverlauf: Vergleich zwischen Schaltnetzteil und Akkumulator für *Haswell* 158

D.11. Histogramme für Stromverlauf: Vergleich zwischen Schaltnetzteil und Akkumulator für *Haswell* 159

D.12. Leistungsverlauf der Rechenknoten *node01*, *node02* und *node03* während des Betriebes ohne Rechenlast. Die Messfrequenz beträgt 12.5 kHz. . 162

E.1. Rechenleistungs-Approximation im Falle *L2-Cache* 167

E.2. „Uncore“-Frequenz und CPU-Frequenz im Falle *Hauptspeicher* 170

E.3. Erste Ableitung der Rechenleistung nach der CPU-Frequenz im Falle *Hauptspeicher* 171

E.4. Approximations-Koeffizienten im Falle *Hauptspeicher* 173

G.1. Aufbau eines DDR3-SDRAM-Speichermoduls 178

G.2. Zeitdiagramm für Burst-Modus 183

I.1. Bandbreite und SDRAM-Frequenz 192

I.2. *Ivy Bridge*: Elektrische Leistung und SDRAM-Frequenz 193

I.3. *Haswell*: Elektrische Leistung und SDRAM-Frequenz 193

J.1. *Ivy Bridge*: Überlappungskoeffizient des DTM-Modells 195

J.2. *Haswell*: Überlappungskoeffizient des DTM-Modells 196

J.3. Vergleich zwischen ECM- und DTM-Modellen für den Kernel *Copy* . 198

J.4. *Haswell*: Überlappungskoeffizient des DTM-Modells 198

J.5. Vergleich zwischen ECM- und DTM-Modellen für den Kernel *Add3* . 199

J.6. Vergleich zwischen ECM- und DTM-Modellen für den Kernel *Add3* . 199

J.7. Komponenten des DTM-Modells für *Copy* 201

J.8. Komponenten des DTM-Modells für *Add3* 202

Tabellenverzeichnis

1.1. CPU-Frequenz und Betriebsspannung eines Intel Pentium M Prozessors 3

1.2. Betriebsspannungswerte eines Prozessors vom Typ *Intel E5-2600v2.* . 4

4.1. Approximation der elektrischen Leistung im Falle *L1-Cache* 36

4.2. Approximation der elektrische Leistung mit einer fixierten Potenz im Falle *L1-Cache* 36

4.3. Approximation der elektrischen Leistung im Falle *L3-Cache* 51

5.1. Pipeline im Falle *L1-Cache* 67

5.2. Pipeline im Falle *L2-Cache* (Teil I) 69

5.3. Pipeline im Falle *L2-Cache* (Teil II) 69

5.4. Vergleich zwischen dem ECM-Model und der Messung 71

6.1. Hazel Hen (Cray XC40) 88

6.2. Hazel Hen: Vergleich zwischen *Add* und *PETsC-CG* 97

7.1. *Ivy Bridge*: Approximation der elektrischen Leistung für L1-, L2-, L3-Caches und Hauptspeicher 104

7.2. *Haswell*: Approximation der elektrischen Leistung für L1-, L2-, L3-Caches und Hauptspeicher 104

A.1. Messgruppen des Rechenknotens „node01“ 119

A.2. Messgruppen des Rechenknotens „node02“ 120

A.3. Messgruppen des Rechenknotens „node03“ 120

B.1. Spannungsverlauf: Vergleich zwischen Messfrequenzen 136

D.1. Betrieb ohne Rechenlast: System-Laufzeitparameter für *Haswell* . . . 151

D.2. Betrieb ohne Rechenlast: System-Laufzeitparameter für *Ivy Bridge* . . 152

D.3. Elektrische Leistung der Rechenknotenkomponenten 163

D.4. Vergleich zwischen der Ausgabe von RAPL-Zählern und den Messwerten während des Betriebes ohne Rechenlast auf *node03* und *node02*. 164

E.1. Rechenleistungs-Approximation im Falle *L1-Cache* 166

E.2. Rechenleistungs-Approximation im Falle *L2-Cache* 166

E.3. Approximation der elektrischen Leistung im Falle *L2-Cache* 168

E.4. Rechenleistungs-Approximation im Falle *L3* 168

E.5. Ungenauere Rechenleistungs-Approximation im Falle *Hauptspeicher* . 170

E.6. Verbesserte Rechenleistungs-Approximation 171

E.7. Koeffizienten der Rechenleistungs-Approximation über f für *Ivy Bridge* im Falle *Hauptspeicher* 172

E.8. Koeffizienten der Rechenleistungs-Approximation über f für *Haswell* im Falle *Hauptspeicher* 172

E.9. Approximation der elektrischen Lesitung im Falle *Haptspeicher* 174

E.10. Approximation der elektrischen Lesitung in Gruppen im Falle *Haptspeicher* 174

F.1. Optimierte Pipeline im Falle *L1-Cache* 176

G.1. Latenzzeiten eines DDR3-SDRAM-Chips 181

G.2. Speicherzellen-Auffrischungszeit 184

J.1. Minimalen und Maximalen Energiekosten der Kernel *Copy*, *Add* und *Add3* 200

Abkürzungsverzeichnis

AC engl. Alternating Current; AC ist eine Bezeichnung für Wechselstrom.

ALU engl. Arithmetic Logic Unit; eine arithmetisch-logische Einheit ist ein Rechenwerk der CPU.

ATX engl. Advanced Technology Extended; eine Normierung der Bauform der Hauptkomponenten eines Computers.

AVX engl. Advanced Vector Extension; ein AVX-Register ist 256-Bit breit und kann bis zu 4 Fließkommazahlen mit der doppelten Genauigkeit speichern.

BIOS engl. Basic Input/Output System; BIOS ist eine Firmware, die auf der Hauptplatine in einem nichtflüchtigem Chip abgelegt ist. Die Firmware wird beim Einschalten und bei der Konfiguration der Basishardware verwendet.

BMC engl. Baseboard Management Controller; BMC dient zur entfernten Verwaltung eines Servers.

CA engl. Caching Agent; Caching Agent ist eine funktionale Einheit eines Prozessors, die den Ringbus mit dem LLC-Cache und dem Prozessorkern verbindet. CA ist eine Uncore-Komponente. CA werden in den neusten Prozessoren mit der Gitter-Topologie für LLC verwendet.

CG engl. Conjugate Gradients; Verfahren der konjugierten Gradienten ist eine numerische Methode zur Lösung von linearen Gleichungssystemen der Form A x = b. Die Matrix A muss symmetrisch und positive definiert sein.

CL engl. Cache Line; eine Cache-Zeile ist die kleinste physikalische Verwaltungseinheit innerhalb des Caches und Hauptspeichers.

CMOS engl. Complementary Metal-Oxide-Semiconductor; CMOS besteht aus Halbleiterbauelementen mit p-Kanal- und n-Kanal-Feldeffekttransistoren.

CSR engl. Compressed Sparse Row; CSR ist ein Format für die Speicherung einer dünn-besetzten Matrix.

DC engl. Alternating Current; AC ist eine Bezeichnung für Gleichstrom.

DDR-SDRAM engl. Double Data Rate Synchronous Dynamic Random Access Memory; mit DDR-Verfahren werden die Daten mit der doppelten Datenrate, sowohl bei den aufsteigenden als auch absteigenden Flanken des Taktsignals, übertragen. Somit wird die doppelte Frequenz in der DDR-Speicher-Spezifikationen angegeben. Die Bezeichnung Synchronous beruht auf der Tatsache, dass alle Zustandsänderungen in Speichermodulen durch die gemeinsame für alle Speicherkomponente geltende Taktfrequenz synchronisiert werden.

DDR1-SDRAM engl. Double Data Rate First-generation Synchronous Dynamic Random Access Memory.

DDR2-SDRAM engl. Double Data Rate Second-generation Synchronous Dynamic Random Access Memory.

DDR3-SDRAM engl. Double Data Rate Third-generation Synchronous Dynamic Random Access Memory.

DDR4-SDRAM engl. Double Data Rate Fourth-Generation Synchronous Dynamic Random Access Memory.

DIMM engl. Dual Inline Memory Module; im Speichermodul werden die Daten aus dem Arbeitsspeicher eines Prozessors flüssig gehalten. Das Modul wird am Speicher-Controller des Prozessors angeschlossen.

DNL engl. Differential nonlinearity; differenzielle Nichtlinearität beschreibt die maximale Abweichung der Stufenbreite einer idealen Stufenfunktion für die Umwandlung des Analog-Signales in ein Digital-Signal von der tatsächlich realisierten Stufenbreite im A/D Wandler.

DMA engl. Direct Memory Access; DMA ist eine Speicherzugriffsart, die die modernen Akzeleratoren, Netzwerkkarten und andere Erweiterungshardware nutzen können, um effizient über ein dediziertes Bussystem auf den Hauptspeicher eines Rechenknotens zuzugreifen.

DRAM engl. Dynamic Random Access Memory; DRAM ist eine Bezeichnung für einen auf den Halbleiterelementen basierten Arbeitsspeicher eines Prozessors.

DTM engl. Data Transfer Model; Datentransfer Modell ist eine Erweiterung des ECM Modells mit Unterstützung der unterschiedlichen Frequenzen der CPU und des Speichers.

DVFS engl. Dynamic Voltage and Frequency Scaling; dynamische Änderung der Frequenz und Versorgungsspannung von Prozessoren.

DVS engl. Dynamic Voltage Scaling; dynamische Änderung der Versorgungsspannung von Prozessoren

ECC engl. Error-Correcting Code Memory; hochwertige Speichermodule sind mit einer Fehlerkorrektur ausgestattet. Üblicherweise können alle 1-Bit-Fehler, die bei der Übertragung zwischen einem Speichermodule und CPU entstehen, erkannt und korrigiert werden. Die Mehr-Bit-Fehler können nur erkannt werden.

ECM engl. Execution-Cache-Memory; ECM ist ein Modell, das für die Ermittlung und Erklärung der Rechenleistung der verschiedenen Kernel-Operationen auf einem idealisiertem Mehrkernprozessor verwendet wird.

FDR engl. Fourteen Data Rate; FDR ist eine der Spezifikationen der InfiniBand-Technologie (IB), die die Frequenz des Datensignales über die Infiniband-Leitungen bestimmt. Diese beträgt 14.025 Gbit/s. Typischerweise werden in einem IB-Interface 4 Leitungen verwendet. Damit ergibt sich die theoretische Bandbreite eines Interfaces von 54.54 GB/s.

FEM engl. Finite Element Method; Finite Element Methode.

FLOP engl. Floating Point Operations; Anzahl der Gleitkommazahl-Operationen.

FLOPS engl. Floating Point Operations Per Second; Anzahl der Gleitkommazahl-Operationen pro Sekunde.

FMA engl. Fused Multiply-Add; FMA ist eine Befehlssatzerweiterung, die die Addition und Multiplikation in einer Mikroinstruktion durchführen kann.

GDDR5 engl. Graphics Double Data Rate memory; dieser Typ des Speichers basiert auf DDR3-SDRAM-Technologie und ist für den Einsatz als Grafikspeicher modifiziert.

GPR engl. General Purpose Register; GPRs können als Daten- oder Adressregister eingesetzt werden.

GPU engl. Graphics Processing Unit; eine GPU wird sowohl für die Visualisierung als auch für die Berechnung benutzt. In den meisten Fällen wird eine GPU über die PCI-Schnittstelle in ein System integriert. Zusätzlich kann ein Desktop-Prozessor eine integrierte GPU enthalten.

HA engl. Home Agent; HA überwacht zusammen mit Cache Agenten (CA) die Cache-Kohärenz des Speichers und verwaltet die Zugriffe im Hauptspeicher.

HPC engl. High Performance Computing; Hochleistungsrechnen.

HLRS engl. The High Performance Computing Center Stuttgart; Höchstleistungsrechenzentrum in Stuttgart.

IC engl. Integrated circuit; ein integrierter Schaltkreis ist eine auf einem Halbleiter aufgebrachte elektronische Schaltung.

IMC engl. Integrated Memory Controller; IMC ist ein Speichercontroller, der im Prozessorchip integriert ist. Der Controller ist eine Uncore-Komponente.

INL engl. Integral nonlinearity; integrale Nichtlinearität beschreibt die

Abweichung einer idealen Disktretesierungtransformation vom tatsächlichen Transformationsresultat eines A/D Wandlers.

I/O Server engl. Input/Output Server; I/O ist eine Bezeichnung für einen Server, dessen Hauptaufgabe die Speicherung der Daten ist.

IPMI engl. Intelligent Platform Management Interface; Ein Satz von den Schnittstellen, die für die Wartung und Verwaltung eines Computers benutzt wird.

JEDEC engl. Joint Electron Devices Engineering Council; die JEDEC Solid State Technology Association ist eine Gesellschaft zur Standardisierung von Halbleitern.

LSB engl. Least significant bit; die letzte Stelle einer binären Zahldarstellung (die am weitesten rechts stehende Ziffer) leistet den geringsten Beitrag für die Wertigkeit der Zahl.

LLNL engl. Lawrence Livermore National Laboratory; das Forschungszentrum LLNL ist unter anderem für Pionierarbeit auf dem Gebiet von Hochleistungsrechnern bekannt.

MESI engl. Modified Exclusive Shared Invalid; MESI ist ein Protokoll zur Wahrung der Cache-Kohärenz zwischen den Hierarchiestufen des gemeinsamen Speichers.

MVM engl. Matrix Vector Multiplication; Matrix-Vektor-Multiplikation ist in der linearen Algebra das Produkt einer Matrix mit einem Vektor. Das Resultat der Operation ist ein Vektor.

MPI engl. Message Passing Interface; MPI ist ein Standard für den Nachrichtenaustausch bei parallelen Berechnungen auf dem verteilter Speicher.

NUMA engl. Non-Uniform Memory Access; wenn ein Computersystem mit einem gemeinsamen Speicheradressraum aus mehreren Prozessoren besteht und jeder Prozessor an die eigenen Speichermodule angeschlossen ist, dauert ein Zugriff auf den direkt angeschlossenen Speicher viel länger als ein Zugriff auf den Speicher des anderen Prozessors im System.

OpenMP engl. Open Multi-Processing; OpenMP ist eine Programmierschnittstelle für die Shared-Memory-Programmierung.

OS engl. Operation System; im HPC-Bereich wird überwiegend ein Linux basierendes Betriebsystem verwendet.

PAPI engl. Performance Application Programming Interface; PAPI definiert die plattformübergreifende uniforme Methode zum Auslesen der

unterschiedlichen Hardwarezähler der Hardwarekomponenten.

PBS engl. Portable Batch System; es ist eine Softwareumgebung, die für die Ablaufplanung des Rechners auf einem verteilten System benutzt wird.

PCIe engl. Peripheral Component Interconnect Express; es ist eine standardisierte Schnittstelle zur Verbindung der Peripheriegeräte mit dem Hauptprozessor. PCIe besitzt eine relativ hohe Durchsatzrate und unterstützt die DMA-Speicherzugriffsart.

PCU engl. Power Control Unit; die Logik für die Betriebsspannungsregelung des Prozessors ist in PCU implementiert.

PETsC engl. Portable, Extensible Toolkit for Scientific Computation; die Bibliothek PETsC wird im Argonne National Laboratory entwickelt. Die PETsC dient zur Lösung der partiellen Differentialgleichung in der HPC-Umgebung.

PFlops engl. Billiard Floating Point Operations Per Second; Anzahl der Billiarden von Gleitkommazahl-Operationen pro Sekunde.

PTP engl. Precision Time Protocol; das Precision Time Protocol wird für die Synchronisierung der Uhrzeiteinstellungen mehrerer Geräte in einem Netzwerk verwendet. Die Genauigkeit der Synchronisierung liegt deutlich höher als beim Network Time Protocol (NTP).

PWM engl. Pulse-Width Modulation; Pulsweitenmodulation ist ein Verfahren, das zum Wechseln einer physikalischen Größe zwischen zwei Werten verwendet wird. Die Spannungsreglermodule nutzen die Pulsweitenmodulation, um einen Prozessor mit einer variablen Betriebspannung zu versorgen.

QR Eine QR-Zerlegung der permutierten Matrix A besteht aus einer orthogonalen Matrix Q und einer oberen Dreiecksmatrix R. Mit den Matrizen Q und R lässt sich unter anderem die beste Approximation eines über-definierten linearen Systems Ax=b finden.

RAPL engl. Running Average Limit; RAPL ist eine Bezeichnung für die in Intel-Prozessoren integrierten Zähler. Die Hauptanwendung der Zähler besteht darin, den Stromverbrauch des Prozessors zu kontrollieren.

SDRAM engl. Synchronous Dynamic Random Access Memory; SDRAM ist die Bezeichnung für einen auf Halbleiterelementen basierten Arbeitsspeicher eines Prozessors. Zur Synchronisierung mit dem Speichercontroller werden das Taktsignal und die Kenntnis über die Dauer einer Speicher-Operation verwendet.

SIMD engl. Single-Instruction-Multiple-Data; SIMD ist ein Paradigma für die parallele Bearbeitung der Daten, indem auf unterschiedliche Daten die

gleiche Instruktion angewendet wird.

SM engl. Streaming Multiprozesior; eine GPU besteht aus mehreren Streaming Multiprocessors, die der CUDA-Kernel parallel ausführen.

TSC engl.; Time Stamp Counter; der Zähler wird in jedem Takt im Register TSC inkrementiert. Abhängig von der Prozessorarchitektur kann die Frequenz des Zählers unterschiedlich sein. Im Intel-Prozessoren *Haswell* und *Ivy Bridge* wird der Zähler mit der Grundfrequenz des Prozessors inkrementiert.

VRM engl. Voltage Regulator Modul; ein Spannungsreglermodul ist ein Spannungsregler, der einen Prozessor mit der geeigneten Betriebsspannung versorgt.

Zusammenfassung

Im Mittelpunkt dieser Arbeit steht die Frage nach Energieeffizienz im Hochleistungsrechnen (HPC) mit Schwerpunkt auf Zusammenhänge zwischen der elektrischen Leistung der Prozessoren und deren Rechenleistung.
In Kapitel 1, Einleitung der folgenden Abhandlungen, werden die Motivation und der Stand der Technik auf dem Gebiet der Strommessung und der Energieeffizienz im HPC und dessen Komponenten erläutert.
In den Folgenden Kapiteln 2 und 3 wird eine am Höchstleistungsrechenzentrum Stuttgart (HLRS) entwickelte Messtechnik detailliert diskutiert, die für die Strommessungen im Testcluster angewendet wird. Das Messverfahren der unterschiedlichen Hardwarekomponenten und die Abhängigkeit zwischen deren Stromversorgung, Messgenauigkeit und Messfrequenz werden dargelegt.
Im Kapitel 4 der Arbeit beschreibe ich, welchen Zusammenhang es zwischen dem Stromverbrauch eines Prozessors, dessen Konfiguration und darauf ausgeführten Algorithmen gibt. Der Fokus liegt dabei auf den Zusammenhängen zwischen CPU-Frequenz, Grad der Parallelisierung, Rechenleistung und elektrischer Leistung.
Für den Effizienzvergleich zwischen den Prozessoren und Algorithmen benutze ich ein Verfahren, das auf eine Approximation in der analytischen Form der Rechen- und der elektrischen Leistung der Prozessoren basiert. In diesem Kapitel wird außerdem gezeigt, dass die Koeffizienten der Approximation, die mehrere Hinweise auf Software- und Hardware-Eigenschaften geben, als Basis für die Ausarbeitung eines erweiterten Modells dienen können.
Wie im weiteren Verlauf gezeigt wird, berücksichtigen die existierenden Modelle der Rechen- und der elektrischen Leistung nur zum Teil die unterschiedlichen Frequenz-Domains der Hardwarekomponenten. Im Kapitel 5 wird eine Erweiterung des existierenden Modells der Rechenleistung erläutert, mit dessen Hilfe die entsprechenden neuen Eigenschaften der CPU-Architektur teilweise erklärt werden könnten.
Die daraus gewonnenen Erkenntnisse sollen helfen, ein Modell zu entwickeln, das sowohl die Rechen- als auch die elektrische Leistung beschreibt. In Kapitel 6 beschreibe ich die Problemstellung der Energieeffizienz eines Hochleistungsrechners. Unter anderem werden die in dieser Arbeit entwickelten Methoden auf eine HPC-Platform evaluiert.

Abstract

At the forefront of the presented work is the question of energy efficiency in high-performance computing (HPC) with a focus on correlations between the electrical power of the processors and their performance .
In chapter 1, Einleitung of the following essay the motivation, the state of the art in the field of power measurement and energy efficiency in HPC as well as the targets of this work are stated.
The next two chapters 2 and 3, Messsystem für Testcluster and Elektrische Leistung im Betrieb ohne Rechenlast, a measuring technique developed in the High Performance Computing Center Stuttgart (HLRS) is discussed in detail. The developed measurement system has been used for the power measurements in a test cluster. The measuring method of the different hardware components and the relationship between the power supplies, measuring accuracy and measuring frequency is explained.
In the following chapter 4, Elektrische Leistung und Rechenleistung der Kernel-Operation *Add*, I describe the close connection between the power dissipation of a processor, its configuration and algorithms running on it. The focus is on the correlation between CPU frequency, degree of parallelisation, performance and power dissipation.
For the efficiency comparison between the processors and algorithms, I use a method that is based on an approximation in the analytical form of the performance and power dissipation of the processors. This chapter also shows that the coefficients of approximation, which give several hints on software and hardware properties, can serve as the basis for the elaboration of an extended performance and power model. As will be shown below, the existing models only partially account for the different frequency domains of the hardware components.
Chapter 5, Modellierung der Kernel-Ausführung, discusses an extension of the existing performance model that could be used to partially explain the new features of the CPU architecture. The lessons learned will help to develop a model that describes both computing and electrical performance.
In Chapter 6, Energieverbrauch eines Höchstleistungsrechners, I describe the problem of energy efficiency of a high performance computer. Among other things, the methods developed in this thesis are evaluated on a HPC-platform.

1. Einleitung

1.1. Motivation

Höchstleistungsrechner werden überwiegend zur Simulation von komplexen Prozessen eingesetzt. Die rechnergestützte Wissenschaft hilft dabei, die praktischen Probleme in unterschiedlichen Bereichen zu lösen. Die Einsatzgebiete sind zum Beispiel Molekulardynamik, Aerodynamik, Biomechanik, Wettervorhersage, Struktur und Dynamik in Festkörpern. Die modernen und zukünftigen nummerischen Simulationen verlangen hoch auflösende Modelle mit fast unbegrenzter Anzahl von Unbekannten. Um solche Simulationen in einer geforderten Zeit durchzuführen, werden immer größere Supercomputer gebaut. Die zwei stark limitierenden Faktoren dagegen sind technischer und ökonomischer Natur: die gesamte elektrische Leistung der Höchstleistungsrechner und deren Energieverbrauch. In dieser Arbeit wird die Frage nach der Energieeffizienz von verschiedenen Rechnerkomponenten insbesondere von den Prozessoren und dem Hauptspeicher untersucht.

1.2. Stand der Technik

1.2.1. Elektrische Leistung eines Halbleiterbauelements

Ein Prozessor ist ein integrierter Schaltkreis, der aus mehreren Millionen Halbleiterbauelementen besteht, kurz IC, und der auf der Basis von der CMOS-Technologie hergestellt wird. Die Halbleiter bestehen aus den Materialien, derer Leitfähigkeit zwischen den elektrischen Leitern ($\rho < 10^{-4}\ \Omega \times \mathrm{cm}$) und den Dielektrikan ($\rho > 10^{9}\ \Omega \times$ cm) liegen. „Die COS/MOS- oder auch die CMOS-Technik ist eine MOS-Technik, die in jeder Schaltung sowohl p-Kanal-MOS-FET-Systeme als auch n-Kanal-MOS-FET-Systeme verwendet “ (Beuth et al.'Elementare Elektronik', S. 315 [6]). Die Approximation-Modelle für die elektrische Leistung von CMOS werden in mehreren Arbeiten

untersucht. Im Jahre 1992 untersuchten Chandrakasan et al. [7] die elektrische Leistung von den Halbleiterbauelementen in den eingebetteten Systemen: „Motivated by emerging battery-operated applications that demand intensive computation in portable environments, techniques are investigated which reduce power consumption in CMOS digital circuits while maintaining computational throughput.“. Die Autoren heben drei Hauptkomponenten für die elektrische Leistung P_{cmos} hervor:
$P_{cmos} = P_{static} + P_{dynamic} + P_{shortcircuit}$

$P_{static} = I_{leakage} * V_{dd}$ - statischer Anteil der elektrischen Leistung wird durch die unerwünschten Stromflüsse (Leckströme) innerhalb der Transistoren verursacht, wenn die Transistoren in einem gesperrten Zustand sind.

$P_{dynamic} = \alpha * C_L * V_{dd}^2 * f_{clk}$ - dynamischer Anteil der elektrischen Leistung wird durch die Ladung und Entladung von Halbleiterelementen verursacht.

$P_{shortcircuit} = I_{sc} * V_{dd}$ - Kurzschluss-Anteil der elektrischen Leistung wird durch den Kurzschlussstrom verursacht, wenn ein NMOS-Transistor und ein PMOS-Transistor für eine kurze Zeit gleichzeitig geöffnet sind.

Der Term V_{dd} bezeichnet die Betriebsspannung. Der Koeffizient α drückt die Wahrscheinlichkeit aus, dass die beiden Transistoren ihre Zustände im nächsten Takt ändern. Das kann auch als Anzahl der Transistoren in einem Prozessor betrachtet werden, die in einem Zeitfenster umgeschaltet werden. Der Term C_L ist die Lastkapazität eines CMOS-Elementes. Der Term f_{clk} ist eine Frequenz, die für den Betrieb von IC benutzt wird.
In [7] wird gezeigt, dass die dynamische Spannungsanpassung (engl. Dynamic voltage scaling oder kurz DVS) eine erfolgversprechende Technik zur Reduzierung der Verlustleistung ist. Wenn die Betriebsspannung V_{dd} reduziert wird, wird die $P_{dynamic}$ quadratisch reduziert. Allerdings bedeutet die Reduzierung der Versorgungsspannung eine Erhöhung der Umschaltzeit des Transistors. Das führt dazu, dass die Taktfrequenz auch reduziert werden muss.

1.2.2. Dynamic Voltage und Frequency Scaling

In den modernen Prozessoren werden sowohl die Betriebsspannung, als auch die CPU-Frequenz dynamisch angepasst (engl. Dynamic voltage and frequency scaling oder kurz DVFS). Die Firma Intel war einer der ersten Prozessorhersteller, der diese Technologie zur potenziellen Reduzierung des Stromverbrauchs und der Betriebstem-

peratur eingeführt hatte. Es ist als Enhanced Intel-SpeedStep-Technologie bekannt. In der Produktbeschreibung vom Intel Pentium M Prozessor des Jahres 2004 [1] werden die möglichen Frequenz- und Betriebsspannungsbereiche aufgelistet (siehe Tabelle Tab. 1.1). Die entsprechenden Werte für die Spannung und Frequenz sind einander zugeordnet.

Tab. 1.1.: CPU-Frequenz und Betriebsspannung eines Intel Pentium M Prozessors

Frequenz(GHz)	Spannung(V_{CC})
1.6	1.484
1.4	1.420
1.2	1.276
1.0	1.164
0.8	1.036
0.6	0.956

In den nachfolgenden Intel CPU Modellen ist die Anzahl der Spannungs- und Frequenzstufen erhöht. Die Höhe der Betriebsspannung V_{CC} wird vom Prozessor automatisch abhängig von den unterschiedlichen Parametern, wie zum Beispiel von der aktuellen Temperatur und den physikalischen Eigenschaften des Silikon-Chips ausgewählt. Außerdem wurden mehrere Stromleitungen mit unabhängigen Spannungswerten eingefügt, die für die diversen Komponenten benutzt werden. In Tab. 1.2 sind die Stromleitungen mit absolut minimalen und maximalen Werten für die Betriebsspannung und Stromstärke des *Ivy Bridge* Prozessors mit zehn Prozessorkernen (E5-2690v2) aufgelistet. Die Werte sind aus der technischen Dokumentation der Firma Intel entnommen [24].
Die Logik der Spannungsregelung ist im „*Power Control Unit*“ des Prozessors (PCU) definiert. Über eine Kontrollverbindung zwischen dem PCU und dem Spannungsreglermodul (VRM: engl. Voltage Regulator Module, siehe Abb. B.5) wird der ausgewählte Spannungswert an das VRM gesendet.[1] Wenn die Spannung vom VRM eingestellt ist, wird die Frequenz des Prozessors umgeschaltet.
Die aktuelle Frequenz dagegen kann vom Benutzer mit einem Systemaufruf eingestellt werden. Die *Ivy Bridge* Prozessorkerne besitzen eine gemeinsame Stromleitung. Aus diesem Grund werden sie mit der gleichen CPU-Frequenz getaktet. Die Intel

[1] Für die Spannungsauswahl stehen 179 Spannungsstufen von 0.0 V bis zum 1.52 V zur Verfügung: 0.000 V, 0.500 V, 0.505 V, ..., 1.520 V. Die weiteren Anforderungen an die VRM-Module sind im Dokument „VR12/IMVP7 Pulse Width Moduleation (PWM) Specification“ definiert [26].

Tab. 1.2.: Maximale und minimale Spannungswerte für die unterschiedlichen Stromleitungen eines Prozessors vom Typ *Intel E5-2600v2.*

Symbol	Beschreibung	Max(V)	Max(A)
V_{CC}, I_{CC}	Spannung und Strom für Prozessorkern	1.4	165.0
V_{CCPLL}, I_{CCPLL}	... für Frequenz-Multiplikator (PLL)	2.0	2.0
V_{CCD}, I_{CCD}	... für Standard DDR3-Module	1.85	8.0
V_{CCD}, I_{CCD}	... für Low-Voltage DDR3-Module	1.7	8.0
V_{SA}, I_{SA}	... für Integrated Memory Controller, QPI-Agent, I/O Controller etc.	1.4	24.0
V_{TT},I_{TT}	... für IO-,QPI-, Display- und andere Analogteile im Prozessor	1.4	24.0

Prozessoren der nächsten *Hasswell* Generation sind mit den zusätzlichen internen Spannungsreglern ausgestattet. Das ermöglicht, die Frequenz per Prozessorkern auszuwählen. Trotz interner Spannungsregler benötigt der *Hasswell* die externen VRM-Module.[2]

Die Latenzzeit für die Umschaltung der CPU-Frequenz ist für die beiden Prozessoren relativ groß und kann mehrere Hundert Mikrosekunden erreichen [30].[3] Wie in Abb. 1.1 zu erkennen ist, steigt die Betriebsspannung eines Prozessorkerns mit der Frequenz annähernd linear. Das Diagramm zeigt die Abhängigkeit zwischen der Core-Spannung V_{cc} und der CPU-Frequenz in GHz während der Ausführung einer Kernel-Operation *ADD*, die in weiteren Abschnitten detailliert beschrieben wird.

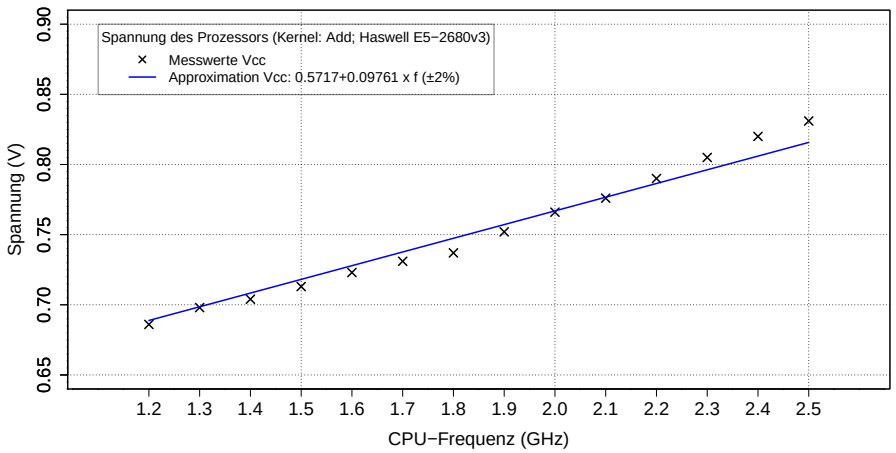

Abb. 1.1.: Abhängigkeit zwischen der Core-Spannung V_{cc} und der CPU-Frequenz in GHz während der Ausführung des Kernels *Add* auf dem Pozessor *Haswell E5-2680v3*

[2] Die Spannung der externen VRM-Modulen V_{CCIN} kann sich im Bereich von 0.0 V bis zum 2.73 V ändern. Die Spannung V_{CCIN} wird im Prozessor weiter angepasst, um eine optimale Betriebsspannung V_{cc} zu erreichen. Für die Spannung V_{CCIN} stehen 210 diskrete Werte zur Verfügung: 0.00 V, 1.00 V, 1.01 V, ..., 2.73 V [22].

[3] Die Betriebsspannung des *Ivy Bridge* Prozessors stellt sich mit 20.0 $\frac{\text{mV}}{\mu\text{s}}$ stufenweise ein.

1.2.3. Frequency Scaling und Stromverbrauch

Der Stromverbrauch eines Rechenknotens für eine numerische Simulation hängt von der Rechenleistung und der elektrischen Leistung ab. Die Prozessoren mit den Speichermodulen tragen einen wesentlichen Anteil dazu bei.
Sowohl die Reduzierung als auch die Erhöhung der CPU-Frequenz können die Energieeffizienz eines Rechenknotens steigern. Einerseits bedeutet die höhere Frequenz eine Erhöhung des dynamischen Anteils der elektrischen Leistung. Anderseits kann sich die Rechenleistung durch die Erhöhung der CPU-Frequenz erhöhen. Es führt zur Reduzierung der Laufzeit und somit zu einer Senkung der Energiekosten, die durch den statischen Anteil der elektrischen Leistung verursacht werden.
Neben den Algorithmen spielen dabei die Prozessorarchitektur und die für die Herstellung der integrierten Schaltkreisen verwendeten Technologien eine wichtige Rolle. Etienne Le Sueur and Grenot Heiser verglichen in ihrer Arbeit „Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns“[49] drei Prozessoren mit unterschiedlichen CMOS-Technologien miteinander: *Sledgehammer* (Jahr 2003, 130 nm), *Santa Rosa* (Jahr 2006, 90 nm) und *Shanghai*(Jahr 2009, 45 nm) [4]. Der Betrieb vom *Shanghai* Prozessor war für alle getesteten Szenarien am effizientesten mit der höchsten Frequenz. Das galt unter anderem für den Benchmark „SPEC CPU2000 181.mcf“ [34], der durch eine hohe Rate von *Cache misses* charakterisiert ist. Basierend auf experimentelle Ergebnisse haben die Autoren angenommen, dass die Reduzierung der Transistorengröße zur Limitierung der Frequenzskalierungsanwendung führen sollte: “Smaller transistor have a lower threshold voltage and, because sub-threshold leakage grows exponentially, more current is lost into the transistor substrate. Processors with smaller transistors can run at higher frequencies with lower supply voltages. The net effect is a reduction in the dynamic range of power consumption that DVFS can utilise and an increase in static power consumption.“. Etienne Le Sueur and Grenot Heiser haben nicht berücksichtigt, dass die CMOS-Technologie nicht nur in der Größe sondern auch bei ihrer Gestaltung erfolgreich weiterentwickelt wird. Die im Jahre 2011 eingeführten Tri-Gate Transistoren [21] reduzierten gegenüber Planar Transistoren nicht nur die dynamische sondern auch die statische Verlustleistung der Halbleiter.
Die kürzeren Ein- und Ausschaltzeiten verringern zusätzlich den Kurzschlussanteil $P_{shortcircuit}$. Im Gegensatz dazu wurden von uns im Jahr 2011 die Power- und Performance-Messungen für den Prozessor Intel Core i7 [39] durchgeführt. Dabei wurde ein einfacher Test aus dem Benchmark-Test STREAM [53] verwendet. Die Resultate zeigten, dass die Energieeffizienz eines Intel Core 7 Prozessors durch die

Reduzierung der Taktfrequenz um bis zu 50 Prozent erhöht werden kann [47].[4]
Im Jahr 2013 haben Markus Wittmann, Gregor Hager und Thomas Zeiser eine Lattice-Bolzmann CFD Simulation auf bis zu 128 Prozessoren auf einem parallelen Rechner durchgeführt [77]. Zwar wurde im Artikel auch darauf hingewiesen, dass Energieeffizienz durch die statische elektrische Leistung stark limitiert ist, aber dabei konnten einige positive Resultate erzielt werden.

1.2.4. Messung vom Stromverbrauch

Als erste Aufgabe war ein Verfahren für die Messung und Aufzeichnung des zeitlichen Verlaufs der elektrischen Leistung eines Testclusters zu entwickeln. Im Artikel "Power measurement techniques on standard compute nodes: A quantitative comparison" [31] werden verschiedene Methoden zur Messung der elektrischen Leistung von Hardwarekomponenten betrachtet. Die Autoren haben gezeigt, dass es immer einen Ausgleich zwischen der Messgenauigkeit, der zeitlichen Auflösung und den zusätzlichen Kosten gibt. Messgeräte, Software und einige Erweiterungen der Infrastruktur müssen besorgt werden und an die bestehende Cluster-Hardware angepasst werden.
Eine der Varianten ohne Hardwareerweiterung ist die Verwendung von den im CPU eingebauten Hardware-Zählern, falls der Prozessor diese hat. Zum Beispiel sind die modernen Intel Prozessoren mit den „Running Average Power Limit" Zählern (RAPL) ausgestattet, aus denen der Stromverbrauch der Prozessoren und der Speichermodule ausgelesen werden kann. Die RAPL-Zähler können unter anderem aus dem laufenden Programm ausgelesen werden.[5] Die zeitliche Auflösung der RAPL-Zähler beträgt laut Intel-Spezifikation eine Millisekunde. Die Genauigkeit der Messung hängt stark von der Prozessorgeneration und den Rechenaufgaben [30] ab.
Ein anderer Weg die RAPL-Zähler auszulesen ist die Verwendung von „Intelligent Platform Management Interface Policy" (IPMI). Am IPMI wird ein „Baseboard Management Controller" (BMC) angeschlossen. Ein BMC-Manager-Modul kann die RAPL-Zähler autonom auslesen und über ein Netzwerk die Messdaten an einen „Node Manager" versenden. Wenn das verwendete Schaltnetzteil einen IPMI kompatiblen integrierten Sensor für den Stromverbrauch besitzt, kann die Information aus dem Sensor des Schaltnetzteiles ausgelesen werden. Die im Schaltnetzteil integrier-

[4] Für die Messung der elektrischen Leistung habe ich ein im Schaltnetzteil integriertes Leistungsmessgerät verwendet.

[5] In vielen HPC-Platformen muss der Benutzer über die Root-Rechte verfügen oder die besonderen Tools dafür verwenden.

ten Temperatur- und Stromverbrauchsensoren haben typischerweise die Auflösung von 1 s. Die Verwendung vom BMC für die Strommessung der Intel-Plattformen präsentierte Andrey Semin auf der Konferenz "Energy-Aware High Performance Computing 2014"[69].
Eine andere im [31] beschriebene Methode ist die Verwendung eines Leistungsmessgerätes *LMG450*, das sowohl in Wechsel- als auch in Gleichstromkreisen verwendet werden kann. Im Unterschied zu den Hardware-Zählern bietet diese Methode eine bessere Genauigkeit, eine höhere zeitliche Auflösung und keinen zusätzlichen Aufwand für die zu messenden Komponenten. Allerdings gibt es dabei einige Nachteile: Preis, Größe und eingeschränkte Zeitsynchronisierung zwischen dem Messgerät und den Rechenknoten: Die Zeitverschiebung liegt weit über der zeitlichen Auflösung des Messgerätes. Die Autoren haben für die Zeitsynchronisierung zwischen den Rechenphasen und der elektrischen Leistung ein zusätzliches Signal benutzt, das mit einer definierten Arbeitslast bzw. einem festgelegten Algorithmus auf dem Prozessor generiert wird. Dies erzeugt ein erkennbares Muster in den aufgezeichneten elektrischen Leistungsprofilen, das für die Markierung des Anfangs und des Endes einer Rechenphase verwendet wird. Zwar löst das Verfahren ein Synchronisierungsproblem, verlangt aber für die Vorbereitung eines Experimentes viele zusätzliche Schritte. Außerdem dürfen die Rechenphasen nicht sehr groß sein, da die Quarzuhren (vom Messgerät und von den Rechenknoten) innerhalb von wenigen Minuten stark auseinander gehen können.
Eine HPC-Plattform, wie zum Beispiel *Cray XC40*, kann mit einer eigenen Messeinrichtung ausgestattet werden [52]. Die zeitliche Auflösung ist der von den RAPL-Zählern ähnlich. Abhängig von der Version des Messsystems werden die unterschiedlichen Hardwarekomponenten gemessen. In dieser Arbeit werden die Hauptmerkmale eines *Cray XC40* Messsystems in Abschnitt 6.1.1 erläutert.

1.2.5. Anwendungen im Hochleistungsrechner

Die Frage nach der Energieeffizienz eines Hochleistungsrechners beschränkt sich nicht nur auf den Stromverbrauch der einzelnen Hardwarekomponenten. Wenn auch die Stromkosten während der gesamten Betriebslaufzeit des Hochleistungsrechners mit den Kosten von dessen Hardwarekomponenten vergleichbar sind, stellt die bestehende Infrastruktur eines Rechenzentrums zusätzliche Anforderungen an den Betrieb des Hochleistungsrechners. Eine der mächtigsten dieser Anforderungen ist die maximale mögliche elektrische Leistung, die der Hochleistungsrechner aufnehmen darf.

Trotz dieser Limitierung kann ein Hochleistungsrechner mit einer höheren als der unterstützten maximalen elektrischen Leistung im Betrieb genommen werden, wenn das zur Verfügung stehende Power-Budget nicht übersteigert wird. Obwohl keine solchen Systeme zum Zeitpunkt des Schreibens dieser Arbeit produktiv im Betrieb genommen worden sind, laufen mehrere Untersuchungen nach möglichen Lösungen. Ein Prototyp der sogenannten „Over-Provisioned“ Systeme wurde in Rechenzentrum des „Lawrence Livermore National Laboratory (LLNL)“ installiert [28]. Das entwickelte Verfahren wurde in einem Produktionssystem der Universität Kyūshū [6] erfolgreich getestet [66].
Der Fokus der Arbeit war, den weitverbreiteten Job-Scheduler „SLURM“ mit einer neuen Funktionalität zu erweitern. Es wird versucht, das zur Verfügung stehende Power-Budget über die Rechenaufträge, die sogenannten Jobs, fair zu verteilen. Die elektrische Leistung kann durch die Abschaltung der nicht benutzen Hardwarekomponenten oder durch die Konfiguration der Prozessoren und des Speichers limitiert werden. Die Modelle für die Rechen- und die elektrische Leistung der Rechenaufträge erlauben dem Job-Scheduler den Durchsatz des Systems zur erhöhen, besonders wenn die Rechenkomponenten fortlaufend überwacht werden.
Die Autoren weisen außerdem darauf hin, dass es Mängel an den Messtechniken in HPC-Systemen gibt. Die weiterverbreiteten RAPL-Zähler können die elektrische Leistung der Prozessoren und des Speichers und keinen anderen Komponenten liefern. Außerdem ist die zeitliche Auflösung der Zähler nicht genug hoch. Ein eingebautes Messsystem, wie es bei den Hochleistungsrechnern der Firma *Cray* der Fall ist, ist eher eine Ausnahme.
Ein anderer Mangel vieler existierender Messeinrichtungen ist, dass die Genauigkeit der Messung nicht spezifiziert ist. Besonders bei größeren Systemen kann dies dazu führen, dass die tatsächliche elektrischen Leistung der Hardwarekomponenten stark von der gemessenen elektrischen Leistung abweicht.

[6] Die HPC-Platform „HA8000“ besteht aus 965 Dual-Socket-Rechenknoten.

2. Messsystem für Testcluster

In diesem Kapitel wird das Vorgehen zur Bestimmung der elektrischen Leistung und des Stromverbrauchs der Hardware-Komponenten eines Testclusters beschrieben. Dieses wurde im Rahmen des Projektes "EXCESS" [11] aufgebaut. Das primäre Ziel des Clusters ist die Unterstützung der Forschung im Bereich von „Energy-Aware HPC".[1]
Das im HLRS in der Abteilung „Numerical Methods & Libraries" entwickelte und in den Cluster integrierte Messsystem besteht aus einem Server mit mehreren Analog-Eingabekarten. Eine Eingabekarte besitzt acht differenzielle Kanäle für die Digitalisierung von acht voneinander unabhängigen Analogsignalen, die, wie im betrachteten Fall die Information sowohl über die Betriebsspannung als auch über die elektrische Leistung der Hardwarekomponenten kontinuierlich übertragen.
Das Messsystem hat folgende Vorteile gegenüber den in Abschnitt 1.2.4 beschriebenen Methoden:

- **Genauigkeit und Auflösung**: Wie im Folgenden gezeigt wird, hat das Messsystem eine hohe absolute und relative Messgenauigkeit. Der Messfehler kann unter einem Prozent gehalten werden. Das System zeichnet die Messdaten mit einer hohen zeitlichen Auflösung auf. Mit den verwendeten A/D-Wandlern kann die Auflösung bis zu 10 µs erreicht werden.

- **Erweitbar und universell einsetzbar**: Das Messsystem zeichnet den Stromverbrauch nicht nur von den Prozessoren, sondern auch von Netzwerkkarten, Akzeleratoren und anderen Hardwarekomponenten auf. Es kann ohne großen Aufwand in eine Arbeitsumgebung, wie zum Beispiel "*Portable Batch System*" (PBS), integriert werden.

- **Stabil**: Das Messsystem läuft stabil und kann ohne Unterbrechungen im Betriebsmodus verwendet werden.

Um dem Leser einen Überblick über die Funktionsweise des Messsystems zu geben,

[1] Der Cluster wurde außerdem in anderen Projekten, wie zum Beispiel im Projekt "DreamCloud" [10], verwendet.

werden in diesem Kapitel die Hauptmerkmale des Testclusters und dessen Messsystems erläutert. Die Einzelheiten über die Aufbau des Testclusters, des Messsystems und dessen Kalibrierung sind in Anhang A auf Seite 117 und Anhang B auf Seite 123 beschrieben.

2.0.1. Prozessor- und Speicherkonfiguration

Auf den Rechenknoten des Testclusters sind zwei Typen von Prozessoren installiert: *Ivy Bridge* auf *node01* und *node02* und *Haswell* auf *node03*.

2.0.1.1. Ivy Bridge

Der *Ivy Bridge* Prozessor *E5-2690 v2* wurde im Jahr 2013 in den Markt eingeführt. Dieser Prozessor verfügt über 10 Prozessorkerne, 20 HW-Threads, 25 MB *L3*-Cache, 256 kB *L2*-Cache und 32 kB *L1*-Cache für die Daten.[2] Der *L3*-Cache ist in zehn Cache-Segmente aufgeteilt, die miteinander durch einen bidirektionalen Bus, sogenanntem Ring, verbunden sind. Außerdem sind am Ring vier DDR3-Speicherkanäle, QPI- und PCIe-Links und andere Komponenten angeschlossen. Der Prozessor kann mit 16 unterschiedlichen CPU-Frequenzen von 1.2 GHz bis zu 3.6 GHz im *Turbo*-Modus getaktet werden.[3] Wie auch sein Nachfolger *Haswell* ist der Prozessor aus Tri-Gate Transistoren mit der Technologie von 22 nm hergestellt.

2.0.1.2. Haswell

Der *Haswell* Prozessor *E5-2680 v3* wurde im Jahre 2014 in den Markt eingeführt. Ein *Haswell E5-2680 v3* verfügt über 12 Prozessorkerne, 24 HW-Threads, 30 MB *L3*-Cache, 256 kB *L2*-Cache und 32 kB *L1*-Cache für die Daten. Im Vergleich zum *Ivy Bridge* verfügt der *Haswell* über vier DDR4-Speicherkanäle, einen erweiterten Ring und einige andere Modifikationen. Der Prozessor kann mit 15 unterschiedlichen

[2]Zwei Metriken für die Datenmenge kB und kiB werden im Script als identisch betrachtet. In einem Kilobyte der Daten sind es 2^{10} B. Das gleiche gilt für MB, GB etc.

[3]Von *Ivy Bridge E5-2690 v2* Unterstützte CPU-Frequenzen: 1.2 GHz, 1.3 GHz, 1.5 GHz, 1.6 GHz, 1.7 GHz, 1.8 GHz, 2.0 GHz, 2.1 GHz, 2.2 GHz, 2.4 GHz, 2.5 GHz, 2.6 GHz, 2.7 GHz, 2.9 GHz, 3.0 GHz und *Turbo*. Im *Turbo*-Modus kann der Prozessor mit einer Frequenz bis zu 3.6 GHz arbeiten.

CPU-Frequenzen von 1.2 GHz bis zu 3.3 GHz im *Turbo*-Modus getaktet werden.[4] Die Turbo-Frequenz ist zwischen 2.8 GHz und 3.3 GHz abhängig von den Vektorinstruktionen und der Anzahl der aktiven Prozessorkerne spezifiziert [25]. Die weitere Information über die beiden Prozessoren ist im Online-Artikel [70] oder in der technischen Beschreibung der beiden Prozessoren zu finden. Außerdem wird die Architektur des *Haswell* Prozessors in Kapitel 5 detailliert betrachtet.

2.1. Messsystem

Im folgenden Abschnitt werden wesentliche Einzelheiten erläutert, wie die elektrische Leistung der Hardwarekomponenten gemessen werden kann.
Der Kern des Messsystems besteht aus einem Linux-Server mit vier Analog-Eingabekarten[5], die den Verlauf der elektrischen Leistung der Cluster-Komponenten aufzeichnen. Der Server ist in einem 19" Servergehäuse installiert und über Ethernet-Netzwerk mit anderen Services des Testclusters verbunden.

2.1.1. Messsystem-Integration in Testcluster-Umgebung

Auf dem Testcluster ist ein PBS-System installiert, damit die Anwendungen auf den Rechenknoten in einer isolierten Umgebung ausgeführt werden. Ein Jobscript besteht aus mehreren PBS-Direktiven und den Shell-Befehlen der Anwendung. Mit den PBS-Direktiven werden die bestimmten Rechenknoten für eine exklusive Nutzung gefordert, auf denen die Anwendung später ausgeführt wird.[6]
PBS-System ruft am Start und am Ende eines Jobs zwei Scripte *prologue* und *epilogue* auf. Die Messung der reservierten Rechenknoten wird im PBS-Script *prologue* gestartet. Und im PBS-Script *epilogue* werden die aufgezeichneten Spannungs- und Stromwerte auf einem I/O Server gespeichert.[7] Das Messsystem markiert die aufgezeichneten Messwerte mit den Zeitstempeln und mit der Job-ID-Nummer. Daher

[4] Von *Haswell E5-2680 v3* Unterstützte CPU-Frequenzen: 1.2 GHz, 1.3 GHz, 1.4 GHz, 1.5 GHz, 1.6 GHz, 1.7 GHz, 1.8 GHz, 1.9 GHz, 2.0 GHz, 2.1 GHz, 2.2 GHz, 2.3 GHz, 2.4 GHz, 2.5 GHz und *Turbo*.

[5] Die Konfiguration mit 6 Analog-Eingabekarten wurde getestet. Mit einer entsprechenden Skalierung des Messsystems und dem I/O-Server kann die Anzahl der Karten erhöht werden.

[6] Der Zeitpunkt der Ausführung wird vom PBS-Scheduler ausgewählt, nachdem der Benutzer seinen Job-Script an den PBS-Server überreicht.

[7] Optional können die Messdaten mit einer Verzögerung von einigen Sekunden an einen „ATOM Monitoring Framework" Server [36] gesendet werden. Der Server „ATOM" speichert die Mess- und die Performance-Daten in einer Datenbank und kann diese über ein Web-Interface visualisieren.

ist es möglich, die unterschiedlichen Phasen eines Jobs den Messwerten zuzuordnen. Die Genauigkeit der Synchronisation bestimmt das „Precision Time Protocol“ (PTP), das die Rechenknoten und das Messsystem zeitlich synchronisiert [27].[8]

2.1.2. Messsystem-Integration in Rechenknoten

Eine Hauptplatine[9] eines Rechenknotens verfügt über mehrere Buchsen zur Verbindung eines Schaltnetzteiles. Zum Beispiel verfügen die gängigen „Dual Socket“ Hauptplatinen über einen 24-poligen ATX- und zwei 8-polige CPU-Stecker, die an einem Schaltnetzteil angeschlossen sind. Sowohl die Spannung als auch die Stromstärke werden an diesen Stromanschlüssen gemessen. Die dafür angebrachten Signalleitungen übertragen die Spannungs- und Stromverläufe an eine Analog-Eingabekarte, wie dies in Anhang B auf Seite 127 gezeigt ist. Daher werden die Hardwarekomponenten zusammen gemessen, die die gleiche Stromquelle besitzen. Somit teilen sich die Hauptkomponenten in mehreren Messgruppen auf. Zum Beispiel besitzt ein Rechenknoten mit zwei Prozessoren mindestens zwei Messgruppen: „CPU1“ und „CPU2“. Die weiteren Messgruppen im Testcluster sind in Anhang A auf Seite 119 zusammengefasst.

2.1.3. Aufzeichnung der elektrischen Leistung mit A/D-Wandler

Für die Gleichstrommessung im Testcluster wird per Rechenknoten eine Analog-Eingabekarte mit acht Messkanälen verwendet. Das Messsystem zeichnet die Betriebsspannung und die elektrische Stromstärke der einzelnen Messgruppen auf. Das 12 Volt-Signal eines Schaltnetzteiles kann nicht direkt erfasst werden, weil die maximale Spannung des Signales an einer Analog-Eingabekarte auf 10.0 V beschränkt ist. Aus diesem Grund werden die Spannungsteiler mit zwei Widerständen R_1 und R_2 verwendet. Damit wird das Signal um $c_{divider} = \frac{R_1+R_2}{R_2}$-mal geschwächt. Die verwendeten Spannungsteiler wurden jeweils aus zwei Chip-Widerständen mit R_1=10 kΩ und R_2=220 Ω gebaut (siehe Anhang B auf Seite 125). Abb. 2.1 stellt schematisch den Messablauf für eine Messgruppe dar.

[8] Mit dem in IEEE 1588 [50] definierten PTP ist es möglich, die verteilten Uhren über Ethernet-Netzwerke auf wenige Mikrosekunden genau zu synchronisieren. Die Uhren des Testclusters sind dadurch mit der Genauigkeit von 10 µs synchronisiert.

[9] In der englischsprachigen Literatur wird von „Mainboard“ gesprochen.

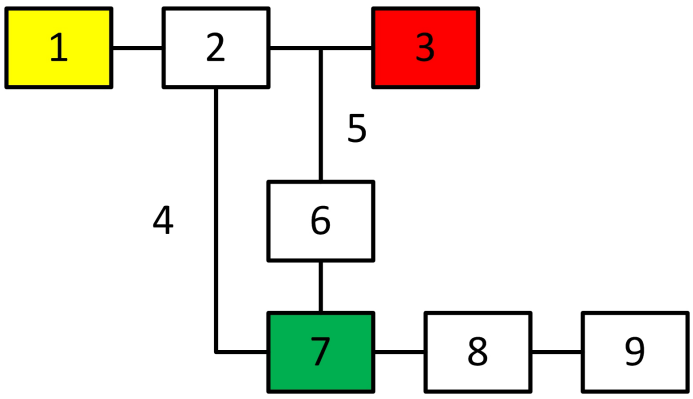

Abb. 2.1.: Funktionales Blockschema der Messung einer Messgruppe: 1 - Schaltnetzteil, 2 - Shunt bzw. Messwiderstand für die Strommessung, 3 - Hardwarekomponenten der Messgruppe, 4 - Stromkreis für die Strommessung, 5 - Stromkreis für die Spannungsmessung, 6 - Spannungsteiler, 7 - A/D-Wandler, 8 - digitale Bearbeitung der Rohmessdaten, 9 - unformatierte I/O Datenausgabe der Messdaten.

Im Block 8 wird die momentane elektrische Leistung P_i einer Messgruppe nach Gl. (2.1) bestimmt. Die Variable $v_i^{divider}$ bezeichnet den Spannungswert am Spannungsteiler während der i-ten Messung. Die Koeffizienten $c_{divider}$ und c_{shunt} können aus den vom Hersteller angegebenen Widerstandswerten ausgerechnet oder durch die Kalibrierung mit einem Multimeter und einem Widerstandsmessgerät bestimmt werden. Die Variable v_i^{shunt} bezeichnet die gemessene abfallende Spannung am Messwiderstand während der i-ten Messung.

$$\begin{aligned} &P_i = I_i * V_i; \\ &V_i = c_{divider} \times v_i^{divider}; \qquad I_i = c_{shunt} \times v_i^{shunt}; \\ &c_{divider} = \tfrac{R_1+R_2}{R_2}; \qquad c_{shunt} = \tfrac{1}{R_{shunt}}; \end{aligned} \tag{2.1}$$

Die elektrische Leistung eines gesamten Rechenknotens wird am AC-Kaltgerätekabel mit zwei Präzision-Stromwandlern gemessen (siehe Anhang B auf Seite 128). Die durchschnittliche elektrische Leistung am AC-Kaltgerätekabel wird nach Gl. (2.2) für ein Zeitintervall T durch ein Summenprodukt der Messwerte $v_j^{AC_V}$ und $v_j^{AC_I}$ berechnet, die während des Zeitintervalls an den Stromwandlern N_T-mal aufgezeichnet werden. Die Koeffizienten c_{AC_V} und c_{AC_I} werden durch die Kalibrierung bestimmt.

$$\begin{aligned} &P_i^T = \tfrac{1}{N_T} \sum_j v_j * i_j; \\ &v_j = c_{AC_V} * v_j^{AC_V}; \quad i_j = c_{AC_I} * v_j^{AC_I}; \end{aligned} \tag{2.2}$$

2.1.4. Filterung

Die aufgezeichneten Roh-Messwerte $v_i^{divider}$, v_i^{shunt}, $v_j^{AC_V}$ und $v_j^{AC_I}$ werden im Block 8 (siehe Abb. 2.1) vor der Berechnung der elektrischen Leistung gefiltert. Die zwei verwendeten Filter sind in Gl. (2.3) beschrieben. Der i-ten Roh-Messewert ist in Gl. (2.3) mit x_i bezeichnet. Die beiden Größen n_l und n_r bestimmen die Größe vom „Patch-Window“ eines Messwertes. Anders gesprochen, definieren diese Parameter, wie viele Messwerte bei der Berechnung der i-ten gefilterten Messwert berücksichtigt werden. n_l steht für die Anzahl der Messwerte vor und n_r nach dem Messwert x_i.[10] Am Anfang und am Ende der Messreihe werden die Daten nicht gefiltert:

- Für die Anwendung vom „Mittelwert-Filter“ wird der durchschnittliche Wert $\boldsymbol{x_i'}$ im „Patch-Window“ berechnet.
- Für die Anwendung vom „Median-Filter“ werden die Werte im „Patch-Window“ sortiert. Der mittlere Wert ist der gesuchte Medianwert $\boldsymbol{x_i'}$.

$$
\begin{aligned}
&\textit{Patch-Window:}\\
&\qquad (x_{i-n_l}, ..., x_{i-1}, x_i, x_{i+1}, ..., x_{i+n_r});\\
&\textit{Median-Filter:}\\
&\qquad (x'_{i-n_l}, ..., x'_{i-1}, \boldsymbol{x_i'}, x'_{i+1}, ..., x'_{i+n_r}) :\ x'_j \leq x'_{j+1};\\
&\textit{Mittelwert-Filter:}\\
&\qquad \boldsymbol{x_i'} = \tfrac{1}{n_l+n_r+1} \sum_{j=i-n_l}^{i+n_r} x_j;
\end{aligned}
\tag{2.3}
$$

Um die Störungen in der Messung zu glätten und dabei so wenig wie möglich von der Information zu verlieren, wird der „Median-Filter“ auf die Messwerte für den Messwiderstand angewendet. Die Messung am Spannungsteiler filtert man mit dem „Mittelwert-Filter“. Im weiteren Verlauf werden die Gründe dafür erläutert.

2.2. Pulsweitenmodulation eines Schaltnetzteiles

In diesem Abschnitt wird gezeigt, dass die Pulsweitenmodulation eines Schaltnetzteiles eine wesentliche Fehlerquelle für die Messung der elektrischen Leistung der Hardwarekomponenten ist. Der Kalibrierungsfehler einer Messgruppe ist dagegen einfach zu bestimmen und beträgt weniger als ±0.6 % (siehe Anhang B auf Seite 131). Die Schaltvorgänge im PWM-Controller verursachen die Störungen, die als

[10] In der englischsprachigen Literatur wird von „window half-size“ gesprochen.

„Ripple-/Noise“ bezeichnet werden. Die detaillierte Beschreibung der Arbeitsweise eines Schaltnetzes kann der interessierte Leser im Buch von Valter Quercioli „Width Modulated (Pwm) Power Supplies“ [65] finden.
Zur Analyse der Störungen werden zwei unterschiedliche Schaltnetzteile (siehe Anhang A auf Seite 120) und ein Lithium-Polymer-Akkumulator *LiPo* verwendet. Ein Lithium-Polymer-Akkumulator gibt die Energie durch elektrochemische Vorgänge ab und ist keine Quelle für die „Ripple-/Noise“ -Störungen. Eine Beschreibung der Akkumulator-Anwendung kann der Leser in Anhang B auf Seite 126 finden.
Im Folgenden wird die Spannungsmessung an den oben erwähnten Stromquellen analysiert. Für die Analyse wurde ein Hochlast-Widerstand vom Type *R850-2R2-J* 2.2 Ω an die 12V-Schiene (+12VCD) der beiden Schaltnetzteile und des Akkumulators angeschlossen.

2.2.1. Spannungsverlauf mit Oszilloskop

Zwei Oszillogramme sind in Abb. 2.2 dargestellt. Im linken Oszillogramm ist die zeitlich variable Komponente des Spannungsverlaufs des Schaltnetzteils *Intel* und im rechten Oszillogramm des Schaltnetzteils *Antec* zu sehen. In den Oszillogrammen ist das sich periodisch wiederholende Impuls-Signal zu erkennen, das von der Pulsweitenmodulation verursacht und durch die parasitäre Induktivität der Messleitungen verstärkt wird. Die Oszillogramme wurden im Mode „Average over 4096 samples“ bei 500 *MSa/s* aufgenommen. Aus diesem Grund sind die Spitzen der Impulssignale relativ kurz. Es ist deutlich sichtbar, dass die Signalformen der beiden Oszillogramme unterschiedlich sind und dass die Oszillation vom *Intel* viel höher als die vom *Antec* ist: ±0.2% gegen ±0.05% der Gesamtspannung. Das ähnliche Oszillogramm für den Spannungsverlauf des Akkumulators *LiPo* beim angeschlossenen Hochlast-Widerstand sieht wie eine schmale gerade Linie aus. Die Frequenzen der beiden Signale sind unterschiedlich: 116 kHz beim *Intel* und 70 kHz beim *Antec*. Der oszillierende Spannungspegel im Messkreis kann eine erhebliche Verzerrung in die Messergebnisse bringen. Deshalb wurde bei der Auswahl der Messeinrichtungskomponenten darauf geachtet, dass deren Blindwiderstände minimal sind.

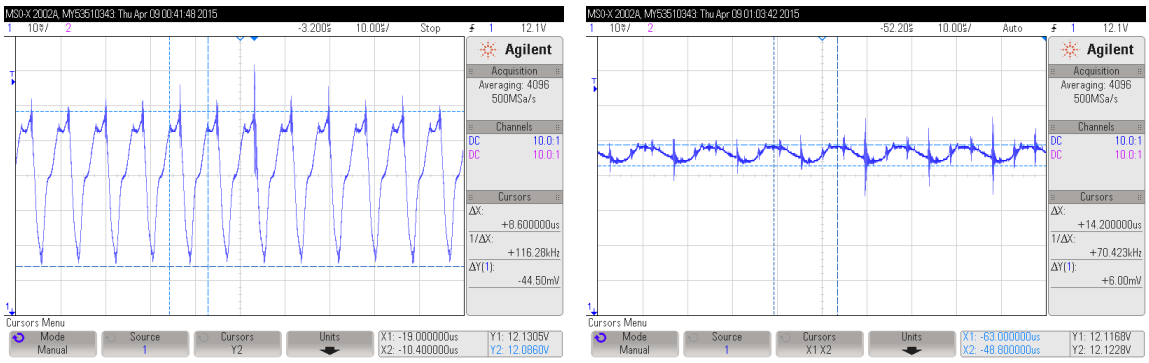

Abb. 2.2.: Oszillogramme mit den Verlaufsgraphen für die Spannung. Die Spannung wurde parallel zum Hochlast-Widerstand 2.2 Ω gemessen, der direkt am +12VCD Ausgang vom Schaltnetzteil *Intel DPS-750XB* (links) oder *Antec EartWatts EA-650* (rechts) angeschlossen wurde. Die Messfrequenz ist 500 *MSa/s*. Und der Messmodus des Oszilloskops ist „Average over 4096 samples“.

In Anhang C auf Seite 141 sind weitere Oszillogramme sowohl für das Spannungssignal als auch für das Stromsignal in unterschiedlichen Modi dargestellt.

2.2.2. Verlauf der elektrischen Leistung

Desweiteres wird statt eines Oszilloskops eine Analog-Eingabekarte *APCIe-3021* [2] für die Aufzeichnung der Signale verwendet. Bei Gleichstrom wird die tatsächlich umgesetzte elektrische Leistung aus dem Produkt der elektrischen Spannung und der Stromstärke berechnet (siehe Gl. (2.1)). In Abb. 2.3 sind die Histogramme des elektrischen Leistungsverlaufs für zwei Schaltnetzteile dargestellt.[11] Zum Vergleich sind außerdem zwei Dichtefunktionen der Normalverteilung gezeichnet. Die Normalverteilung ist durch den Mittelwert und durch die Standardabweichung parametrisiert, die nach Gl. (2.4) berechnet sind.

$$\mu = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i;\ \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{2.4}$$

[11] Die Histogrammwerte sind mit der Anzahl der Messdaten (1000) und der Histogramm-Klassenbreite normiert, so dass die gesamte Fläche der Balken gleich eins ist.

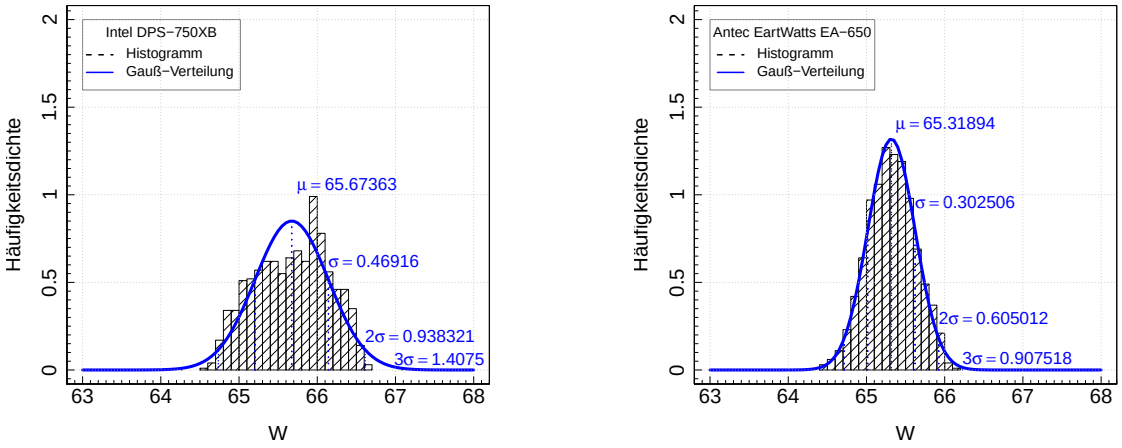

Abb. 2.3.: Auf den Diagrammen sind der Mittelwert, das Maximum, Minimum und die Grenzwerte $\mu \pm 3 * \sigma$ für die fünfzig unabhängigen Stichproben mit je 1000 Messungen dargestellt. Die elektrischen Leistungswerte sind aus den Spannungs- und Strommessungen für die Schaltnetzteile *Intel* (links) und *Antec* (rechts) und angeschlossenem Hochlast-Widerstand 2.2 Ω aufgenommen. Die Messfrequenz beträgt 50.0 kHz.

Wie zu erwarten war, streuen sich die Messwerte für das Schaltnetzteil *Antec* deutlich geringer. In Abb. 2.4 sind ein Leistungsverlauf und ein entsprechendes Histogramm dargestellt, wenn der Hochlast-Widerstand am Akkumulator angeschlossen ist. Es ist deutlich zu erkennen, dass die beiden Schaltnetzteile viel größere Streuung der Messdaten als der Akkumulator verursachen.

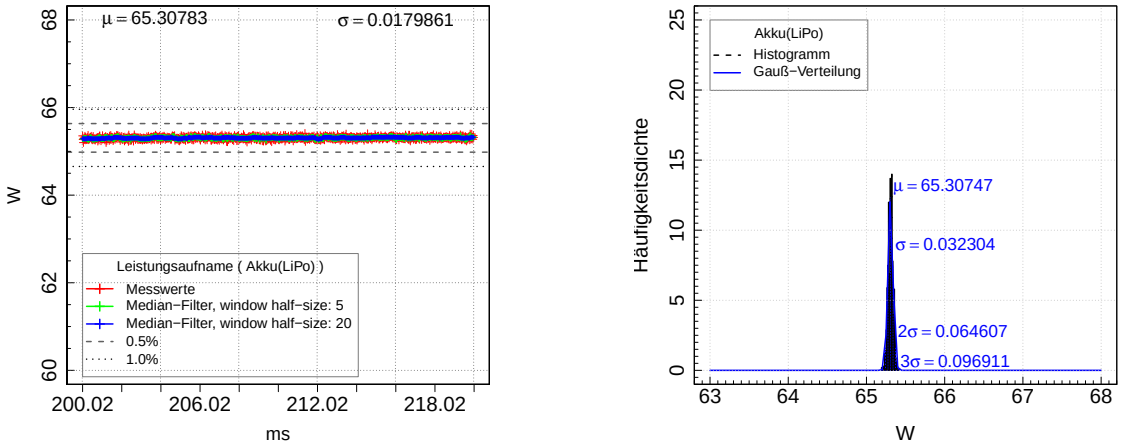

Abb. 2.4.: **Links:** Verlauf der elektrischen Leistung des Hochlast-Widerstandes 2.2 Ω im Messkreis mit Lithium-Polymer-Akkumulator. **Rechts:** Histogramm der Leistungswerte. Zusätzlich ist die Dichtefunktion der Normalverteilung dargestellt. Die Messfrequenz beträgt 50.0 kHz.

Zum Vergleich: Die Rohmesswerte bei der Verwendung eines Akkumulators streuen sich innerhalb des Intervalls von 65.2000 W bis 65.4113 W. Die Differenz beträgt dabei $\Delta_{min,max} = 0.2113$ W. Das ist nun um eine Ordnung kleiner als bei der Verwendung des Schaltnetzteiles *Antec* mit der maximalen Differenz von 1.95 W. Eine detaillierte Untersuchung der Störsignale kann man in Anhang B ab Seite 131 finden.

Bei der Verwendung des Lithium-Polymer-Akkumulators wurde eine Reduzierung des durchschnittlichen Stromverbrauchs registriert. Die Messungen zeigten, dass der Stromverbrauch vom Prozessor *Ivy Bridge* um $\sim 10\%$ senkte und vom Prozessor *Haswell* um $\sim 5\%$. Erste Versuche laufen bereits, die Stromversorgung eines Hochleistungsrechners mit dem DC-Strom vollständig zu realisieren, wie es zum Beispiel im Jahr 2008 in „Lawrence Berkeley National Laboratory“ gemacht wurde [73]. Die Resultate zeigten eine Erhöhung der Energieeffizienz eines Versuchssystems um bis zu 28 %. Die Autoren weisen darauf hin, dass die bessere Energieeffizienz durch die Abschaffung der Verluste bei der AC/DC-Konvertierung erreicht wurde. Die Gründe für Erhöhung der Energieeffizienz, die in dieser Arbeit erläutert wurde, müssen in dieser Arbeit folgenden Untersuchungen geklärt werden.

3. Elektrische Leistung im Betrieb ohne Rechenlast

Im Kapitel 2, „Messsystem für Testcluster“, wurde gezeigt, wie die elektrische Leistung im Messkreis mit einem passiven Lastwiderstand verläuft. Anstatt eines Lastwiderstands werden im Testcluster die unterschiedlichen Hardwarekomponenten an die Stromquellen angeschlossen, die in Messgruppen aufgeteilt sind (siehe Abschnitt 2.1.2). Daher produzieren sie in den Messkreisen nicht nur die zusätzlichen Störungssignale, sondern ändern auch dynamisch ihre elektrischen Eigenschaften, wie zum Beispiel den Widerstand. Um die Messmethode statistisch zu evaluieren, ist eine Untersuchung der elektrischen Leistung der Hardwarekomponenten unter den unterschiedlichen Lasten erforderlich. In diesem Kapitel wird die elektrische Leistung der Hardware-Komponenten des Testclusters während des Betriebes ohne Rechenaufgabe untersucht.
Wenn einem Rechenknoten keine Rechenaufgabe zugewiesen ist, wird dies im Folgenden als ein „Betrieb ohne Rechenlast“ gekennzeichnet. Obwohl es keine direkte Rechenlast gibt, laufen mehrere Dienstprogramme des Betriebssystems auf dem Prozessor. Die Dienste aktivieren sich in unterschiedlichen Zeitabständen durch den Interrupt-Mechanismus. Der interessierte Leser kann mehr über die grundlegenden Prinzipien eines Betriebssystems, zum Beispiel im Buch von Jürgen Nehmer „Systemsoftware - Grundlagen moderner Betriebssysteme“ [58], finden.

3.1. Spannungs- und Stromverlauf

In diesem Abschnitt werden die Spannungs- und Stromverläufe der CPU-Messgruppen der untersuchten Rechenknoten „node03“ und „node02“ betrachtet. Mit Hilfe des direkten Vergleichs mit den in Abschnitt 2.2 beschriebenen Experimenten für einen Hochlast-Widerstand werden die besonderen Merkmale der Messungen im Messkreis mit den Hardwarekomponenten erläutert.

3.1.1. Spannungs- und Stromverlauf (Haswell)

In Abb. 3.1 ist der Spannungsverlauf des ersten *Haswell* Prozessor des Rechenknotens „node03“ dargestellt. Das gefilterte Spannungssignal des Prozessors hat zum Unterschied vom gefilterten Spannungssignal eines Hochlast-Widerstandes eine doppelte Standardabweichung (siehe Abb. B.10 in Anhang B auf Seite 132). Und das hier nicht gezeigte Histogramm des Spannungssignals zeigt eine Symmetrie und entspricht der Normalverteilung, wie es im Falle des Hochlast-Widerstandes ist (siehe Abb. D.1 links in Anhang D auf Seite 152).

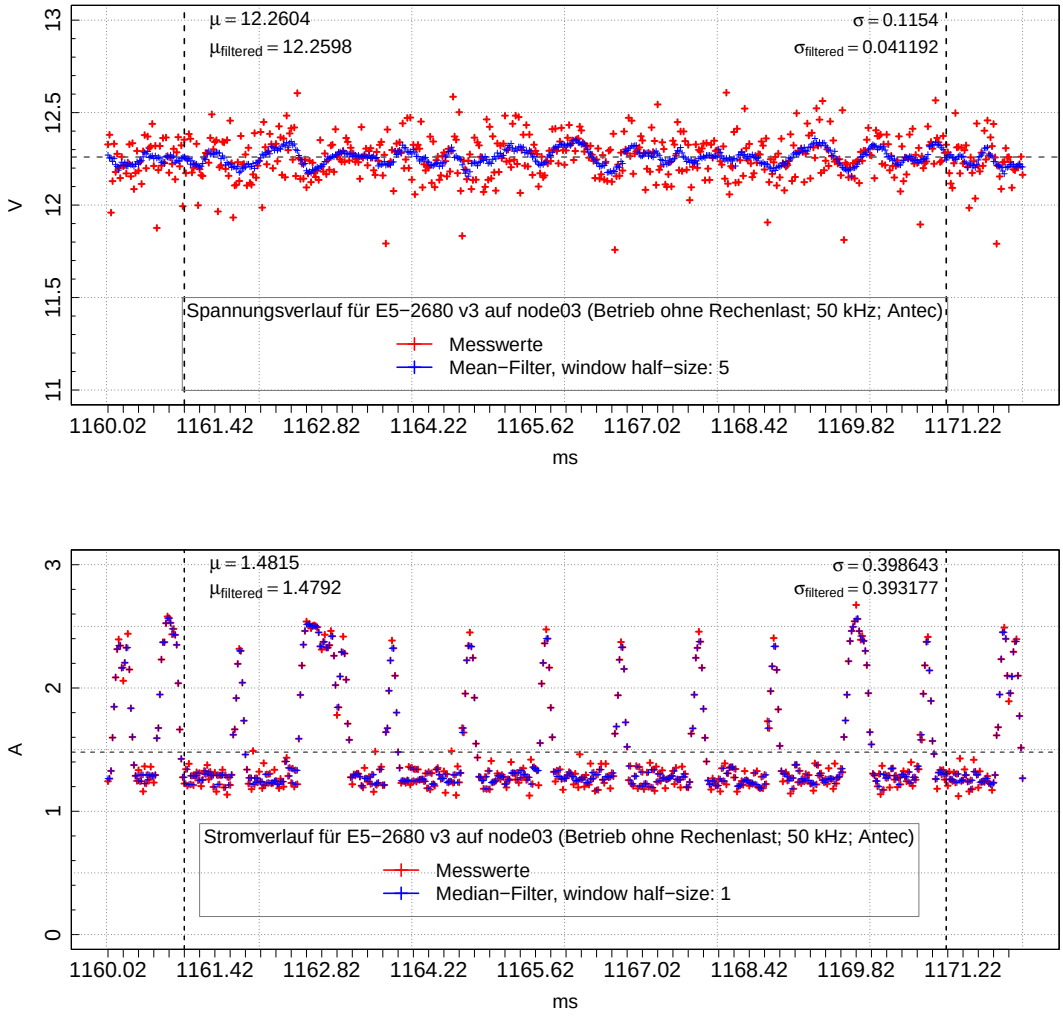

Abb. 3.1.: Spannungsverlauf (oben) und Stromverlauf (unten) im Messkreis mit einem *Haswell* Prozessor des Rechenknotens „node03“ und mit dem Schaltnetzteil *Antec* während des Betriebes ohne Rechenlast.

Zunächst wurde angenommen, es handele sich um die zusätzlichen Störungen, die

durch die Spannungsreglermodule verursacht wurden. Das Einbeziehen des Stromsignales in die Analyse zeigte jedoch, dass die Ursache einer größeren Standardabweichung grade Änderungen in der elektrischen Leistung des Prozessors sind. Es sind eindeutig die Phasen des energiesparenden Zustandes im Stromverlauf in Abb. 3.1 durch die minimal benötigte Stromstärke zu erkennen. Sowohl die Spannung als auch die Stromstärke wurden gleichzeitig während des Betriebs ohne Rechenlast aufgezeichnet. Sobald der Prozessor aus dem energiesparenden Zustand geweckt wird, fällt die Spannung ab. Nach einer kurzen Zeit regelt sich das Schaltnetzteil und stabilisiert die abgefallene Spannung.

3.1.2. Spannungs- und Stromverlauf (Ivy Bridge)

In Abb. 3.2 ist der Spannungsverlauf für den ersten Prozessor *Ivy Bridge* des Rechenknotens „node02“ dargestellt.

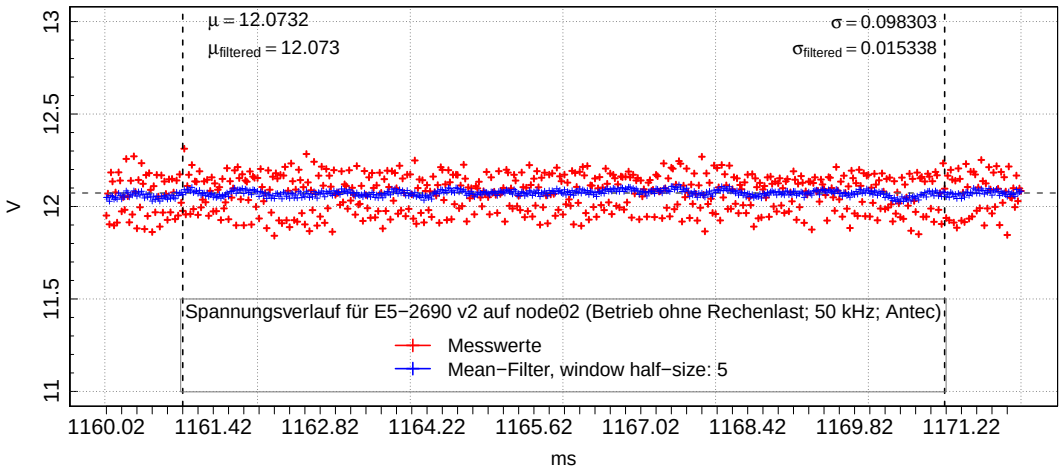

Abb. 3.2.: Spannungsverlauf im Messkreis mit einem Prozessors *Ivy Bridge* und mit dem Schaltnetzteil *Antec* während des Betriebes ohne Rechenlast.

Das gefilterte Spannungssignal für den Prozessor *Ivy Bridge* hat sogar eine kleinere Standardabweichung als das gefilterte Spannungssignal für den Hochlast-Widerstand (siehe Abb. B.10). Wegen der näherungsweise konstanten elektrischen Leistung des Prozessors sind keine regelmäßigen Oszillationen des Spannungssignals in Abb. 3.2 zu erkennen.
Der Stromverlauf änderte sich dementsprechend und ist in Abb. 3.3 dargestellt. Die

regelmäßige Oszillationen des Stromverlaufs werden durch die Funktionsweise des Schaltnetzteiles verursacht. Die Oszillationen sind nicht mehr vorhanden, wenn man einen Akkumulator statt eines Schaltnetzteiles als Stromquelle für die Prozessoren verwendet.[1] Dabei gilt es anzumerken, dass sich die Frequenz der Oszillationen erhöht und dass sich die Amplitude reduziert, wenn der Verbraucher eine höhere Stromstärke benötigt (siehe Abb. D.6 in Anhang D auf Seite 154). In Anhang D auf Seite 156 sind weitere Beispiele der Anwendung vom Akkumulator im Vergleich zum Schaltnetzteil *Antec* beschrieben.

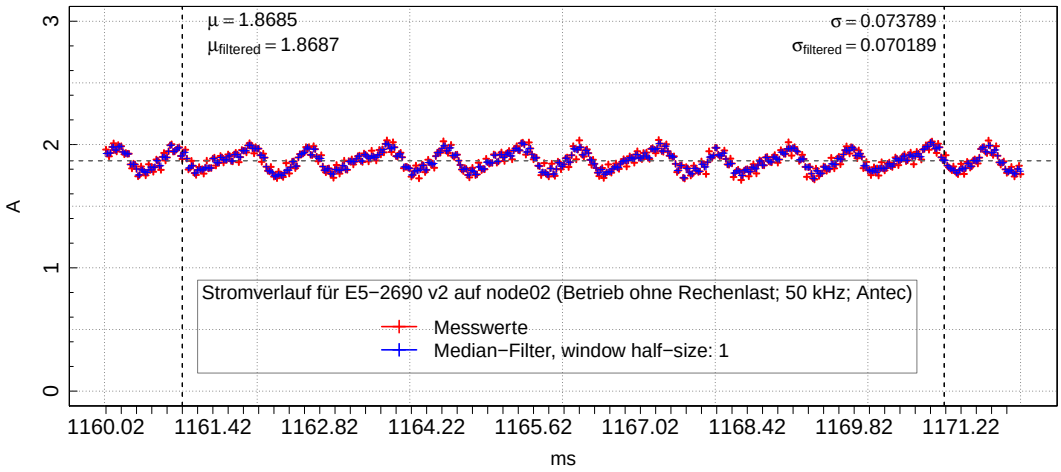

Abb. 3.3.: Stromverlauf im Messkreis mit einem Prozessors *Ivy Bridge* und mit dem Schaltnetzteil *Antec* während des Betriebes ohne Rechenlast. Der Stromverlauf wurde gleichzeitig mit dem Spannungsverlauf in Abb. 3.2 aufgezeichnet.

[1] Die Stromversorgung der Hardwarekomponenten über das ATX-Stromkabel wurde weiterhin mit dem Schaltnetzteil realisiert.

4. Elektrische Leistung und Rechenleistung der Kernel-Operation *Add*

Die Prozessoren und der Speicher bestehen aus mehreren Milliarden von CMOS-Elementen. Die einzelnen Gruppen dieser Elemente funktionieren mit unterschiedlichen physikalischen Parametern, wie zum Beispiel der Betriebsspannung und der Frequenz. Wie in Abschnitt 1.2.1 erwähnt, kann die elektrische Leistung eines Halbleiters P_{cmos} in drei Termen zusammengefasst werden:

$$\begin{cases} P_{cmos} & = P_{static} + P_{dynamic} + P_{shortcircuit}; \\ P_{static} & = I_{leakage} \times V_{dd}; \\ P_{dynamic} & = \alpha \times C_L \times V_{dd}^2 \times f_{clk}; \\ P_{shortcircuit} & = I_{sc} \times V_{dd}; \\ I_{leakage}, I_{sc}: & \textit{ Leck- und Kurzschlussstrom;} \\ f_{clk}: & \textit{Betriebsfrequenz von IC;} \\ V_{dd}: & \textit{Betriebsspannung;} \\ \alpha: & \textit{Wahrscheinlichkeit der Zustandsänderung;} \\ C_L: & \textit{Lastkapazität des CMOS-Elementes;} \end{cases} \tag{4.1}$$

Die Frequenz f_{clk} und die Spannung V_{dd} aus Gl. (4.1) können mit Hilfe der Zähler festgestellt werden. Die anderen Koeffizienten sind entweder unbekannt oder schwer zu schätzen. Aus diesem Grund ist es wünschenswert, eine einfache Formel zur Verfügung zu haben, mit derer Hilfe sowohl eine quantitative als auch eine qualitative Analyse der Hardwarekomponenten durchgeführt werden kann.
Nach einer Reihe der Messungen für unterschiedliche *Kernel-Operationen* und CPU-Frequenzen schlagen Hager et al. [32] die Verwendung eines Polynoms zweiten Grades vor, um die elektrische Leistung zu approximieren: „Note that one would naively expect a cubic dependence in f, if the core voltage is adjusted to always reflect the lowest possible setting at a given frequency. Since we cannot look into the precise algorithm that the hardware uses to set the core voltage, we use the observed

quadratic function as phenomenological input to the power model below, without questioning its exact origins.“. Die elektrische Leistung P wird im Artikel wie folgt approximiert:

$$\begin{cases} P(f,p) & = \alpha_0 + (\alpha_1 \times f + \alpha_2 \times f^2) * p; \\ p: & \textit{Anzahl der aktiven Prozessorkerne}; \\ \alpha_0, \alpha_1, \alpha_2: & \textit{Approximationskoeffizienten}; \end{cases} \tag{4.2}$$

Die Autoren weisen darauf hin, dass diese Gleichung nicht für eine qualitative Analyse gedacht ist. Die quadratische Abhängigkeit der elektrischen Leistung von der Frequenz ist nach Gl. (4.1) physikalisch möglich, wenn die Betriebsspannung V_{dd} durch die Quadratwurzel der Frequenz f_{clk} beschrieben werden kann. Die Messungen zeigen allerdings, dass die Abhängigkeit nahezu linear ist (siehe Abb. 1.1).
Eine andere Form der Interpolation für die elektrische Leistung eines Prozessors wurde von Herrn Uwe Küster vorgeschlagen [48]. Zum ersten Mal wurde das Verfahren im Artikel „Power consumption of kernel operationen“ [45] beschrieben. Im Folgenden wird eine Approximation der elektrischen Leistung eines Mehrkernprozessors und dessen Speichermodule während der Ausführung einer Kernel-Operation beschrieben. Als Kernel-Operation ist ein Algorithmus bezeichnet, der aus in einer Schleife stehenden elementaren Operationen besteht. Ein Beispiel dafür ist die Berechnung der Summe von zwei Feldern und die Speicherung des Resultates in einem dritten Feld. Die Anzahl der aktiven Prozessorkerne wird in diesem Fall durch die Anzahl der Threads bestimmt, die bei der Ausführung auf den Prozessorkernen synchron ausgeführt werden.

4.1. Grundlagen der Approximation der elektrischen Leistung

Die Hardwarekomponenten eines Mehrkernprozessors und dessen Speichermodule können in zwei Gruppen aufgeteilt werden: „Uncore“- und „Core“-Komponenten. Die „Uncore“-Komponenten, wie zum Beispiel der *L3*-Cache, der *L3*-Ring und der integrierte Speichercontroller (IMC), können mit einer eigenen Frequenz getaktet werden, wie im Falle des *Haswell* Prozessors. Außerdem kann die „Uncore“-Frequenz auf einigen Plattformen aus dem „User-Space“ geändert werden, wie es zum Beispiel Hackenbert et al. in [30] gemacht haben. Die Höhe der „Uncore“-Frequenz hängt von mehreren Faktoren ab. Hackenbert et al. schreiben dazu: “The results ... indicate

that uncore frequencies in addition to EPB[1] and stall cycles depend on the core frequency of the fastest active core on the system.“.
Auf dem getesteten Rechenknoten *node03* mit zwei *Haswell* Prozessoren wird die „Uncore“-Frequenz automatisch gesteuert, mit dem Ziel die Energieeffizienz im Durchschnitt zu erhöhen.
Die Menge der „Core“-Komponenten wird aus den Prozessorkernen und den dazugehörigen *L1*- und *L2*-Caches gebildet. Die Frequenz der Prozessorkerne kann innerhalb der spezifizierten Grenzen eingestellt werden. Im Weiteren werden Fälle betrachtet, in denen die Frequenz für alle Prozessorkerne gleich groß ist. Diese Frequenz wird als die CPU-Frequenz bezeichnet.
Aus Gl. (4.1) folgt, dass wenn die CPU-Frequenz eines Prozessors erhöht wird, dominiert der dynamische Term $P_{dynamic}$ über die zwei anderen Terme. Als Grundlage für eine „elementare“ Approximationsformel, im Weiteren als die 1. Approximation bezeichnet, benutzen wir die Interpolation aus Gl. (4.3):

$$
\begin{aligned}
P_I(f)_{l\,p} &= \alpha_{0\,l,p} + \alpha_{1\,l,p} \times f^{\lambda_{l,p}};\\
\\
\epsilon_{L_2} &= \sqrt{\sum_{i=0}^{n_f}(P_{f_i,l,p} - P_I(f_i)_{l,p})^2};\\
\epsilon_{rel} &= \max_{0\le i<m_f} |\tfrac{P_{f_i,l,p}-P_I(f_i)_{l,p}}{P_{f_i,l,p}}|;\\
\epsilon_{abs} &= \max_{0\le i<m_f} |P_{f_i,l,p} - P_I(f_i)_{l,p}|;
\end{aligned}
\tag{4.3}
$$

f : *CPU-Frequenz*;
p : *Anzahl der Threads*;
l : *Hierarchiestufe des Speichers* $l \in \{L1, L2, L3, RAM\}$;
m_f : *Anzahl der CPU-Frequenzstufen*;
$P_{f_i,l,p}$: *gemessene elektrische Leistung*;

In Gl. (4.3) sind zusätzlich der Interpolationsfehler in der L2-Norm ϵ_{L_2}, der relative Fehler ϵ_{rel} und der absolute Fehler ϵ_{abs} definiert. Die Potenz λ der CPU-Frequenz und die entsprechenden Koeffizienten α_0 und α_1 definieren mit einem minimalen Fehler ϵ_{L_2} die „beste“ 1. Approximation in L2-Norm.
Der Koeffizient α_0 [W] kann als der statische Anteil der elektrischen Leistung angesehen werden. Und der Term $\alpha_1 \times f^{\lambda}$ $[\frac{\mathrm{W}}{\mathrm{GHz}^{-\lambda}}]$ zeigt an, in welcher Abhängigkeit der dynamische Anteil der elektrischen Leistung von der CPU-Frequenz ist.

[1]Energy Performance Bias ist die Bias-Einstellung für einen Betriebsmodus des Prozessors (performance, energy saving, balanced).

Somit ergeben sich $n_p \times m_f$ Koeffizienten $\alpha_{0\,l,p}$, $\alpha_{1\,l,p}$ und λ. Zur Minimierung ihrer Anzahl führe ich eine komplexere Form der Approximation ein, die im weiteren Verlauf als die 2. Approximation bezeichnet wird. Der Grundgedanke wäre, die Koeffizienten wiederum über die Threads-Anzahl p zu interpolieren. Somit ergibt sich Gl. (4.4). Die neuen Koeffizienten $\tilde{\alpha}_{0,0,l}$, $\tilde{\alpha}_{0,1,l}$ beschreiben den Verlauf der Koeffizienten der 1. Approximation $\alpha_{0,l,p}$ über die Threads-Anzahl p. Entsprechend wird der zweite Term $\alpha_{1\,l,p} \times f^{\lambda_{l,p}}$ mit $\tilde{\alpha}_{1,0,l}$, $\tilde{\alpha}_{1,1,l}$ und $\tilde{\lambda}_l$ interpoliert.

$$
\begin{aligned}
P_{II}(f,p)_l &= (\tilde{\alpha}_{0,0,l} + \tilde{\alpha}_{0,1,l} \times p) + (\tilde{\alpha}_{1,0,l} + \tilde{\alpha}_{1,1,l} \times p) \times f^{\tilde{\lambda}_l};\\
\\
\epsilon_{L_2} &= \sqrt{\sum_{p=1}^{n_p} \sum_{i=0}^{m_f-1} (P_{f_i,l,p_i} - P_{II}(f_i)_{l,p})^2};\\
\epsilon_{rel} &= \max_{0\le i<m_f;1\le p_i\le n_p} |\tfrac{P_{f_i,l,p_i} - P_{II}(p_i,f_i)_l}{P_{f_i,l,p_i}}|;\\
\epsilon_{abs} &= \max_{0\le i<m_f;1\le p_i\le n_p} |P_{f_i,l,p_i} - P_{II}(p_i,f_i)_l|;
\end{aligned}
\tag{4.4}
$$

f : *CPU-Frequenz*;
p : *Anzahl der Threads*;
l : *Hierarchiestufe des Speichers* $l \in \{L1, L2, L3, RAM\}$;
f_i : *gesetzte CPU-Frequenz* $f_i \in \{f_0, f_1, .., f_{m_f-1}\}$;
m_f : *Anzahl der CPU-Frequenzstufen*;
n_p : *Anzahl der Prozessorkerne*;
P_{f_i,l,p_i} : *gemessene elektrische Leistung für* $f = f_i \wedge p = p_i$;

4.2. Bestimmung der Koeffizienten

Für alle Hierarchiestufen des Speichers l und die Threads-Anzahl p bestimmen wir die Koeffizienten α_0, α_1 und λ auf folgende Weise. Für jede Kernel-Operation werden N lineare Gleichungssysteme aufgestellt. N ist ein Produkt aus der Anzahl der Prozessorkerne N_p und der Hierarchiestufen des Speichers (*L1*, *L2*, *L3*, *RAM*). Jedes der N linearen Systeme besteht aus n_f Gleichungen. n_f ist die Anzahl der CPU-Frequenzstufen, mit denen die CPU-Frequenz vor der Ausführung einer Kernel-Operation gesetzt wird:

$$
\begin{bmatrix} 1 & f_0^\lambda \\ 1 & f_1^\lambda \\ \dots & \dots \\ 1 & f_{m_f-1}^\lambda \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} P_0 \\ P_1 \\ \dots \\ P_{m_f-1} \end{bmatrix}
\tag{4.5}
$$

$P_0 \dots P_{m_f-1}$ sind die gemessenen Werte für die elektrische Leistung des Prozessors und der Speichermodule während der Ausführung einer Kernel-Operation mit den CPU-Frequenzen $f_0 \dots f_{n-1}$. Es ist deutlich zu erkennen, dass das lineare System überdefiniert ist. Mit einem QR-Algorithmus aus [64] für die kleinsten Quadrate können alle Koeffizienten α_0 und α_1 für ein fixiertes λ näherungsweise gefunden werden, so dass der Interpolationsfehler in der L_2-Norm ϵ_{L_2} minimal ist. Um die beste Approximation zu finden, wird ϵ_{L_2} für alle λ aus einem vordefinierten Bereich $[\lambda_{min}, \lambda_{max}]$ mit einem fixierten Schritt ausgerechnet. Danach wird die Potenz λ mit einem minimalen Fehler ϵ_{L_2} ausgewählt.

4.3. Grundlagen der Approximation der Rechenleistung

Um die Energiekosten einer Kernel-Operation auf einem Prozessor zu berechnen, müssen nicht nur die elektrische Leistung der Hardwarekomponenten, sondern auch die Ausführungszeit bekannt sein. Das hängt von der Rechenleistung und der Anzahl der Operationen während der Ausführung ab. In den weiteren Kapiteln wird die Metrik FLOPS eingeführt. Die Rechenleistungsmetrik FLOPS wird als die Anzahl der ausgeführten Gleitkommazahl-Operationen geteilt durch die Ausführungszeit definiert.
Die beiden betrachteten Prozessoren *Ivy Bridge* und *Hasswell* sind mit L1-, L2- und L3-Caches ausgestattet. Falls die Daten komplett im Cache gehalten werden, erwartet man die beste Rechenleistung. Zusätzlich vermuten wir, dass die Abhängigkeit zwischen der Rechenleistung und der CPU-Frequenz in diesem Fall linear sein soll. Falls der Datenzugriff über den Speicher erfolgt, ist die Rechenleistung in vielen Fällen durch die Bandbreite des Speichers begrenzt. Dabei muss man beachten, dass die Bandbreite einer vektorisierten Operation viel höher als die Bandbreite der einzelnen Zugriffe auf den Speicher ist (siehe Anhang G auf Seite 177). Im weiteren Verlauf der Arbeit untersuche ich die Cache- und Speicherzugriffe und ihre Abhängigkeit von der CPU-Frequenz.
Als Grundlage für die Approximationsformel wird Gl. (4.6) benutzt. Mit dieser Gleichung kann sowohl eine lineare Abhängigkeit der Rechenleistung von der Thread-Anzahl als auch eine mit der CPU-Frequenz degradierte Rechenleistung dargestellt

werden.

$$
\begin{aligned}
R_I(f)_{l\,p} &= \beta_{1\,l,p} \times f^{\gamma_{l,p}};\\
\\
\epsilon_{L_2} &= \sqrt{\sum_{i=0}^{n_f}(R_{f_i,l,p} - R_I(f_i)_{l,p})^2};\\
\epsilon_{rel} &= \max_{0\le i<n_f} |\tfrac{R_{f_i,l,p} - R_I(f_i)_{l,p}}{R_{f_i,l,p}}|;\\
\epsilon_{abs} &= \max_{0\le i<n_f} |R_{f_i,l,p} - R_I(f_i)_{l,p}|;
\end{aligned}
\tag{4.6}
$$

f : *CPU-Frequenz*;
p : *Anzahl der Threads*;
l : *Hierarchiestufe des Speichers* $l \in \{L1, L2, L3, RAM\}$;
n_p : *Anzahl der Prozessorkerne*;
$R_{f_i,l,p}$: *gemessene Rechenleistung*;

Der Koeffizienten $\beta_{1\,l,p}$ wird im weiteren Verlauf der Arbeit über die Threads-Anzahl p linear interpoliert. Der $\gamma\, l, p$ wird so ausgewählt, dass der Fehler ϵ_{L_2} minimal wird. Gl. (4.7) definiert die entsprechende Approximationsformel $R_{II}(f,p)_l$:

$$
\begin{aligned}
R_{II}(f,p)_l &= (\tilde{\beta}_{1,1,l} \times p) \times f^{\tilde{\gamma}_l};\\
\\
\epsilon_{L_2} &= \sqrt{\sum_{p=1}^{n_p}\sum_{i=0}^{m_f-1}(R_{f_i,l,p_i} - R_{II}(f_i)_{l,p})^2};\\
\epsilon_{rel} &= \max_{0\le i<m_f;1\le p_i\le n_p} |\tfrac{R_{f_i,l,p_i} - R_{II}(p_i,f_i)_l}{R_{f_i,l,p_i}}|;\\
\epsilon_{abs} &= \max_{0\le i<m_f;1\le p_i\le n_p} |R_{f_i,l,p_i} - R_{II}(p_i,f_i)_l|;
\end{aligned}
\tag{4.7}
$$

f : *CPU-Frequenz*;
p : *Anzahl der Threads*;
l : *Hierarchiestufe des Speichers* $l \in \{L1, L2, L3, RAM\}$;
f_i : *gesetzte CPU-Frequenz* $f_i \in \{f_0, f_1, .., f_{m_f-1}\}$;
m_f : *Anzahl der CPU-Frequenzstufen*;
n_p : *Anzahl der Prozessorkerne*;
R_{f_i,l,p_i} : *gemessene Rechenleistung für* $f = f_i \wedge p = p_i$;

4.4. Kernel-Operation *Add*

Die Kernel-Operation wird in mehreren PBS-Jobs durchgeführt. Eine Job-Ausführung wird im Folgenden als ein Experiment bezeichnet. In einem Experiment fixiert man die Domaingröße bzw. die Länge der Felder und variiert die Anzahl der OpenMP-Threads und die Höhe der CPU-Frequenz. Für die Statistik werden die Resultate von zehn Experimenten mit der jeweils gleichen Länge ausgewertet. Die folgenden Resultate wurden auf den im ersten Socket installierten Prozessoren der Rechenknoten *node01* und *node03* gewonnen, falls nicht anders erwähnt.

4.4.1. Implementierung

Die Kernel-Operation *Add* berechnet die Summe von zwei Feldern $b[]$ und $c[]$ und schreibt das Resultat in das dritte Feld $a[]$:

$$c[i] = a[i] + b[i]; 0 <= i < n; \tag{4.8}$$

Die Berechnung erfolgt in einer Schleife, die mehrmals wiederholt wird.

Quelltext 4.1: Quelltext der Kernel-Operation *Add*

```
#pragma omp parallel num_threads(num_threads)\
        shared(aa,bb,cc,length_per_thread,num_tests,num_it)
{
  const long length=length_per_thread;
  const long tests = num_tests;
  const long iterations = num_it;
  int thread_num = omp_get_thread_num();
  double* __restrict local_a=a[thread_num];
  double* __restrict local_b=b[thread_num];
  double* __restrict local_c=c[thread_num];
  #pragma omp master
  {
    PAPI_START
    get_timestamp(start_time);
  }
  for(jj=0; jj<tests;jj++)
  {
    #ifdef OMPBARRIER    #pragma omp barrier    #endif
    local_ticks_start=read_tsc_register();
    for(kk=0; kk<iterations;kk++)
      for(jj=0; jj<length;jj+=32)
```

```
      {
        __m256d AAVX = _mm256_load_pd(local_a+ii);
        __m256d BAVX = _mm256_load_pd(local_b+ii);
        __m256d CAVX = _mm256_add_pd(AAVX,BAVX);
        ...
        __m256d AAVX8 = _mm256_load_pd(local_a+ii+28);
        __m256d BAVX8 = _mm256_load_pd(local_b+ii+28);
        __m256d CAVX8 = _mm256_add_pd(AAVX8,BAVX8);

        _mm_store_pd(local_c+ii, CAVX)
        ...
        _mm_store_pd(local_c+ii+28, CAVX8)
      }
    thread_ticks[jj]=read_tsc_register()-local_ticks_start;
  }
  #pragma omp master
  {
    get_timestamp(end_time);
    PAPI_END
  }
}
```

Die Schleife ist achtmal entrollt und mit OpenMP-Threads parallelisiert. Die Operation ist mit intrinsischen Funktionen programmiert. Es ist in Quelltext 4.1 zu erkennen, dass die AVX-Register für die Rechenoperationen benutzt werden.
Zwei äußere Schleifen lassen die Operation *Add* mehrfach wiederholen, damit eine minimale, maximale und durchschnittliche Dauer der Operation bestimmt werden kann. Mit der Funktion `read_tsc_register()` wird die genaue Dauer einer Phase bestimmt. Die Funktion liefert den Inhalt des TSC-Registers. Der Zähler im TSC-Register wird bei jedem Takt mit der Grundfrequenz des Prozessors inkrementiert. In unserem Fall sind es entsprechend 2.5 und 3.0 GHz für die Prozessoren *Ivy Bridge E5-2690v2* und *Haswell E5-2680v3*. Dabei spielt es keine Rolle, ob *Haswell* in AVX- oder *non*-AVX Modus ist (siehe Abschnitt 2.0.1.2).
Die Dauer des Zeitabschnittes in Ticks wird für jeden OpenMP-Thread unabhängig voneinander in den privaten Feldern `thread_ticks` gespeichert. Die OpenMP-Operation *Barrier* in der äußeren Schleife synchronisiert die Threads. Für die nachfolgenden Diagramme wird die durchschnittliche Dauer aller Threads benutzt, da die Lastbalancierung zwischen den Threads für den betrachteten Kernel und dessen Konfiguration praktisch perfekt ist. Für die Messung des Stromverbrauchs werden die Zeitstempel vor und nach der Berechnung mit dem Aufruf von der Prozedur `void get_timestamp(..)` aus dem Master-Thread bestimmt.

Die Variablen `num_threads` und `length_per_thread` legen die Anzahl der aktiven *Threads* und die Länge der Felder fest. Die gemeinsamen (engl. *shared*) Zeiger `**a`, `**b` und `**c` zeigen auf die zweidimensionalen Felder. Die erste Dimension hat die Länge `num_threads` und die zweite die Länge `length_per_thread`. Jeder *Thread* initialisiert die lokalen Zeiger `*local_a`, `*local_b` und `*local_c` mit den entsprechenden Speicherbereichen aus `**a`, `**b` und `**c`.
Mit dem Schlüsselwort `__restrict` wird der verwendete Compiler informiert, dass die Speicherbereiche separate Adressräume besitzen [29]. In dem betrachteten Fall wird dies keine Vorteile bringen, weil die verwendete Optimierungsstufe `-O1` dem Compiler untersagt, eigene Optimierung des Codes bezüglich der Vektorisierung durchzuführen: Dieser Kernel ist manuell optimiert.
Vor dem Aufruf des Kernels *Add* werden die Threads erzeugt und zu den entsprechenden Prozessorkernen zugewiesen (*thread pinning*). Nach der Zuweisung initialisiert jeder Thread mit der ID-Nummer `thread_num` die Felder und speichert ihre virtuellen Adressen in den Zeigern `*a[thread_num]`, `*b[thread_num]` und `*c[thread_num]`. Dies garantiert, dass die Daten physikalisch in den eigenen Speichermodulen liegen und dass keine NUMA-Effekte vorhanden sind. Die Anzahl der Threads und die gewünschte Frequenz für die Prozessorkerne werden vor der Kernel-Ausführung im Benchmark gesetzt.
Die gesetzte CPU-Frequenz muss nicht unbedingt der tatsächlichen Frequenz entsprechen. Die PAPI-Zähler dienen unter anderem dazu, dass die durchschnittliche CPU-Frequenz ausgerechnet werden kann. Dafür wird der Zähler `UNHALTED_CORE_CYCLES` benutzt. Standardgemäß werden die PAPI-Zähler für die Zeit angehalten, während der Prozess mit der Kontextumschaltung angehalten wird. Für meine Tests habe ich die PAPI-Zähler so umgestellt, dass sie unabhängig von der Kontextumschaltung arbeiten. Auf solche Weise kann man die tatsächliche Frequenz des Prozessors ausrechnen. Diese Option steht normalerweise nur begrenzt zur Verfügung, weil sie die Root-Rechte benötigt.
Zusätzlich zu den Registern `UNHALTED_CORE_CYCLES` werden die RAPL-Register ausgelesen. Die RAPL-Register liefern die Menge der verbrauchten Energie für die verwendeten Speichermodule und die Prozessoren. Der Leser kann die zusätzliche Information in der technischen Beschreibung [23] finden.

4.5. Approximation der Kernel-Operation *Add*

Im weiteren Verlauf der Arbeit wird die Approximation sowohl für die Rechen- als auch für die elektrische Leistung der Kernel-Operation *Add* auf den Prozessoren *Ivy Bridge* und *Haswell* für vier Domain-Größen gesucht:

- Die ersten zehn Experimente wurden für die Länge `length=1088` durchgeführt. Die Größe des L1-Caches (32 kB) ist für alle Elemente der Felder $a[]$, $b[]$ und $c[]$ ausreichend: Während einer Ausführung der Kernel-Operation liest jeder Thread 17 kB, schreibt 8.5 kB und führt 1088-mal die Fließkommazahladdition aus.

- Mit der Länge `length=6528` werden sowohl der L1-Cache als auch der L2-Cache benutzt. Jeder Thread liest in einer Kernel-Operation 102 kB und schreibt 51 kB. Während einer Ausführung der Kernel-Operation wird knapp die Hälfte vom L2-Cache (256 kB) ausgelesen und ein Viertel davon beschrieben. Der L1-Cache wird mehrmals aktualisiert.

- Mit der Länge `length=57664` werden in jeder Operation sowohl die L1- und L2-Caches als auch der L3-Cache (*Ivy Bridge*: 25 MB, *Haswell*: 30 MB) benutzt. Jeder Thread liest in einer Kernel-Operation 901 kB und schreibt 450.5 kB.

- Mit der Länge `length=42949664` werden die DDR-SDRAM-Speichermodule aktiviert. Somit werden 655.4 MB gelesen und 327.7 MB geschrieben.

4.5.1. First Level Cache

Der Fall des Haltens der Daten im L1-Cache ist der einfachste Fall, weil die Prozessorkerne die Daten während der Kernel-Operation unabhängig voneinander bearbeiten können. Es ist zu vermuten, dass die Rechenleistung mit der steigenden CPU-Frequenz und der Thread-Anzahl nach einem linearen Gesetz steigt.

4.5.1.1. Rechenleistung im First Level Cache

In Abb. 4.1 ist die Rechenleistung der beiden Prozessoren für die unterschiedliche Thread-Anzahl dargestellt. Für jede Konfiguration der Kernel-Operation *Add* ist die minimal erreichte Rechenleistung mit einem Kreuz angezeigt. Dementsprechend

werden der Mittelwert mit einem Kreis und der maximale Wert mit einem Stern dargestellt.
Die Rechenleistungs-Approximation nach Gl. (4.6) ist mit einer durchgehenden Linie unter der Legende *1.Apprx.* dargestellt. Die Gl. (4.6) Gleichung approximiert die Rechenleistung abhängig von f. Die entsprechenden Koeffizienten sind in Tab. E.1 des Anhangs E auf Seite 166 angegeben.
Die Rechenleistungs-Approximation nach Gl. (4.7) ist mit einer gestrichelten Linie unter der Legende *2.Apprx.* dargestellt. Diese Gleichung approximiert die Rechenleistung abhängig von f und p.
Es hat sich gezeigt, dass die beiden Prozessoren nicht mit den exakt spezifizierten CPU-Frequenzen arbeiten (siehe Abschnitt 2.0.1). Dies gilt insbesonders für die Turbo-Frequenz und lässt vermuten, dass die Temperaturbedingungen für den Turbo-Modus nicht optimal sind. In Abb. 4.1 ist zu erkennen, dass die erreichte Turbo-Frequenz mit einem Thread auf dem *Ivy Bridge* die maximale CPU-Frequenz und auf dem *Haswell* die minimale CPU-Frequenz ist. Inwiefern die Temperaturbedingungen die Resultate für den Turbo-Modus beeinflussen, muss mit zusätzlichen Experimenten untersucht werden. Die Bestimmung und die Beeinflussung der umgebenden Temperaturbedingungen werden im Rahmen dieser Arbeit nicht durchgeführt.

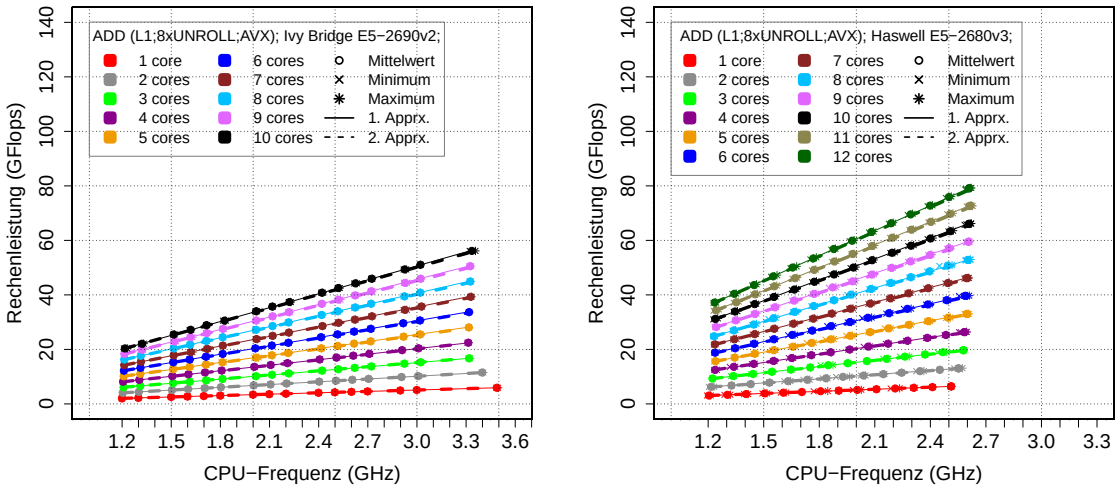

Abb. 4.1.: Rechenleistung auf *Ivy Bridge E5-2690v2* (links) und auf *Haswell E5-2680v3* (rechts) im Falle des Haltens der Daten im L1-Cache. Die Rechenleistung hängt offenbar linear von Frequenz und Anzahl der Threads ab. Der *Ivy Bridge* Prozessor erreicht mit einem Thread die höchste CPU-Frequenz. Der *Haswell* Prozessor ist mit einem Thread am langsamsten.

Die Rechenleistung kann für die Kernel-Operation *Add* im Falle des Haltens der Daten im L1-Cache wie folgt approximiert werden:

Ivy Bridge (E5-2690v2) :

$$\begin{aligned} R_{II}(f,p)_{L1} &= 1.699572 \times p \times f; \\ \epsilon_{L^2} &= 1.433\,188; \\ \epsilon_{rel} &= 0.016\,071; \\ \epsilon_{abs} &= 0.372\,891; \\ 1 \leq & p \leq 10 \wedge f \geq 0.0; \end{aligned} \tag{4.9}$$

Haswell (E5-2680v3) :

$$\begin{aligned} R_{II}(f,p)_{L1} &= 2.556957 \times p \times f; \\ \epsilon_{L^2} &= 3.187\,384; \\ \epsilon_{rel} &= 0.018\,783; \\ \epsilon_{abs} &= 0.886\,022; \\ 1 \leq & p \leq 12 \wedge f \geq 0.0; \end{aligned} \tag{4.10}$$

Der Skalierungsfaktor des *Haswell* Prozessors von 2.56 [FLOP] ist deutlich höher als der Skalierungsfaktor des *Ivy Bridge* Prozessors von 1.7 [FLOP]. Der Grund dafür ist, dass der *Ivy Bridge* in einem Takt der CPU-Frequenz f aus L1-Cache in die AVX-Register 2x16 B laden und 1x16 B schreiben kann. Der *Haswell* verdoppelte die Interfacebreite zwischen dem Cache und den AVX-Registern. Allerdings ist die erreichte Rechenleistung kleiner als die theoretische um 15% für den *Ivy Bridge* und um 37% für den *Haswell*. Einer der Gründe dafür ist die Funktionsweise der Pipeline der Prozessoren. Dies wird im Abschnitt 5.1.2.3 im Detail diskutiert.

4.5.1.2. Elektrische Leistung im First Level Cache

Im folgenden Abschnitt betrachte ich die elektrische Leistung der Prozessoren *Ivy Bridge E5-2690v2* und *Haswell E5-2680v3* und deren Speichermodule während der Ausführung der Kernel-Operation *Add* im Falle des Haltens der Daten in L1-Cache. Die Interpolierung der elektrische Leistung wird nach Gl. (4.3) und Gl. (4.4) durchgeführt.
In Abb. 4.2 und Abb. 4.3 ist die elektrische Leistung der Prozessoren und Speichermodule abhängig von der CPU-Frequenz und von der Thread-Anzahl dargestellt. Es ist zu erkennen, dass der Prozessor *Haswell* mehr elektrische Leistung als der Prozessor *Ivy Bridge* braucht. Der Unterschied bei einer minimalen Frequenz von 1.2 GHz beträgt bis zu 10 W. Wesentlich ist zu erwähnen, dass die Prozessoren *Ivy*

Bridge und *Haswell* mit unterschiedlichen Speichermodulen ausgestattet sind (4 × 4 GB DDR3-SDRAM und 4 × 16 GB DDR4-SDRAM).

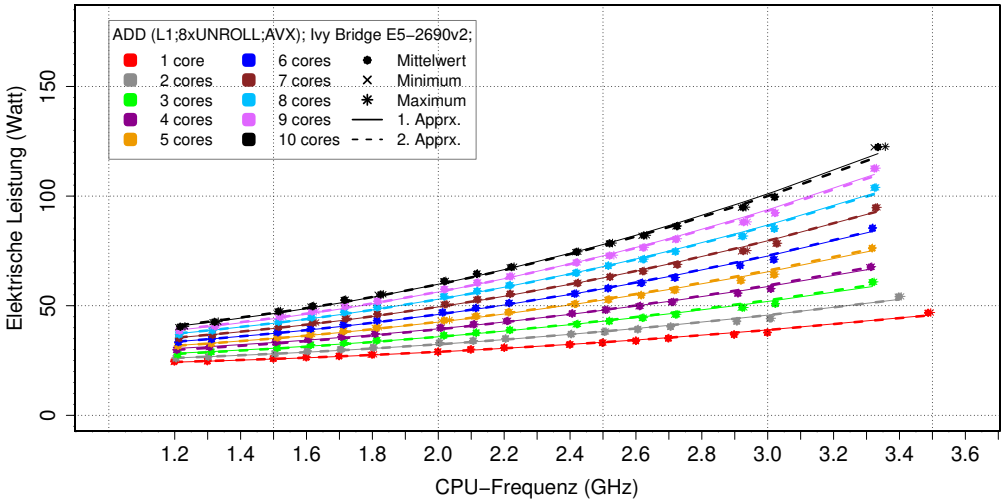

Abb. 4.2.: Elektrische Leistung des *Ivy Bridge E5-2690v2* Prozessors im Falle des Haltens der Daten im L1-Cache

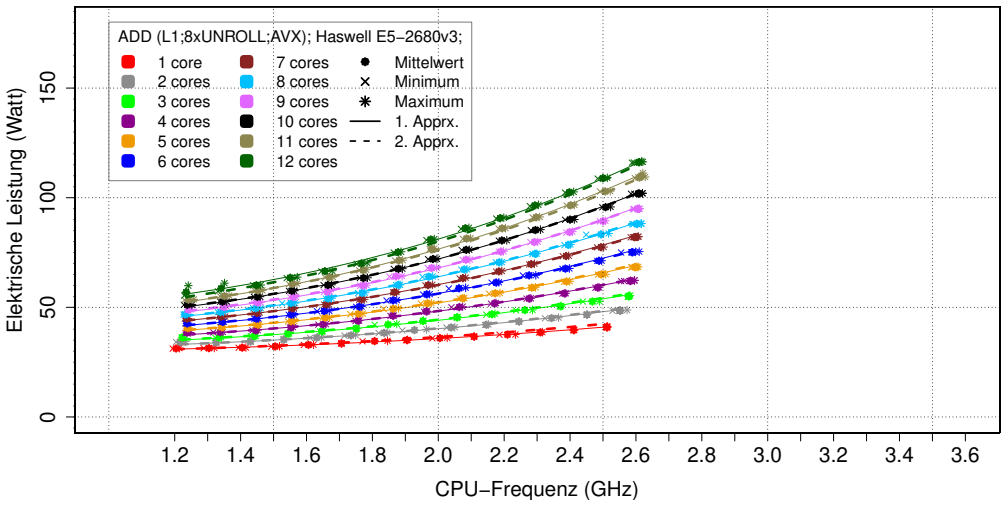

Abb. 4.3.: Elektrische Leistung des *Haswell E5-2680v3* Prozessors im Falle des Haltens der Daten im L1-Cache

Nach der Anwendung der QR-Methode ergeben sich die in der Tab. 4.1 aufgeführten Koeffizienten der 1. Approximation $P_I(f)_{L1,p}$. Die im L2-Norm optimale Potenz $\lambda_{L1,p}$ wurde für alle p einzeln gefunden.

Im Unterschied zur Rechenleistung steigt die elektrische Leistung zur CPU-Frequenz überproportional und konnte mit einer hohen Genauigkeit sowohl nach Gl. (4.3) als auch nach Gl. (4.3) über f und p interpoliert werden. Die in Tab. 4.1 angegebenen Potenzkoeffizienten sind dabei deutlich anders als in [32] angenommen.

Tab. 4.1.: Koeffizienten der Approximation der elektrischen Leistung $P_I(f)_{L1,p} = \alpha_{0\,L1,p} + \alpha_{1\,L1,p} \times f^{\lambda_{L1,p}}$ im Falle des Haltens der Daten im L1-Cache

p	*Ivy Bridge E5-2690v2*					*Haswell E5-2680v3*				
	$\alpha_{0,L1,p}$	$\alpha_{1,L1,p}$	$\lambda_{L1,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L_2\,p}$	$\alpha_{0,L1,p}$	$\alpha_{1,L1,p}$	$\lambda_{L1,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L_2\,p}$
1	22.2327	1.3452	2.30	0.332×10^{-1}	0.204×10^{1}	28.8715	1.2871	2.43	0.113×10^{-1}	0.64
2	23.1133	2.0350	2.19	0.332×10^{-1}	0.292×10^{1}	29.7667	2.0355	2.38	0.380×10^{-1}	0.141×10^{1}
3	24.8474	2.2944	2.25	0.310×10^{-1}	0.322×10^{1}	30.4725	2.8298	2.29	0.376×10^{-1}	0.157×10^{1}
4	25.9241	3.0462	2.16	0.195×10^{-1}	0.218×10^{1}	32.1356	3.1821	2.36	0.280×10^{-1}	0.149×10^{1}
5	26.2018	3.7580	2.13	0.236×10^{-1}	0.308×10^{1}	33.263	3.7309	2.35	0.275×10^{-1}	0.133×10^{1}
6	27.1976	4.2641	2.15	0.272×10^{-1}	0.364×10^{1}	34.5375	4.2716	2.37	0.264×10^{-1}	0.149×10^{1}
7	28.3504	4.6634	2.18	0.263×10^{-1}	0.416×10^{1}	36.3208	4.5060	2.43	0.233×10^{-1}	0.157×10^{1}
8	29.5097	5.1069	2.20	0.286×10^{-1}	0.461×10^{1}	37.8808	4.7386	2.47	0.216×10^{-1}	0.173×10^{1}
9	30.3508	5.5942	2.21	0.262×10^{-1}	0.501×10^{1}	38.4628	5.5683	2.42	0.233×10^{-1}	0.163×10^{1}
10	31.4306	6.0755	2.22	0.247×10^{-1}	0.529×10^{1}	40.6381	5.6285	2.49	0.181×10^{-1}	0.156×10^{1}
11	-	-	-	-	-	43.3376	6.6376	2.43	0.205×10^{-1}	0.186×10^{1}
12	-	-	-	-	-	46.5980	5.2736	2.69	0.172×10^{-1}	0.277×10^{1}

Die Approximation kann vereinfacht werden, wenn die optimale Potenz $\lambda_{L1,p}$ über alle p gesucht wird. So ergeben sich die neuen Koeffizienten in der Tab. 4.2. Sowohl der Fehler der Interpolation als auch die Koeffizienten ändern sich nur unwesentlich für die weitere Analyse.

Tab. 4.2.: Koeffizienten der Approximation der elektrischen Leistung $P_I(f)_{L1,p} = \alpha_{0\,L1,p} + \alpha_{1\,L1,p} \times f^{\lambda_{L1,p}}$. Die optimale Potenz $\lambda_{L1,p}$ wurde über alle p gesucht.

p	*Ivy Bridge E5-2690v2*					*Haswell E5-2680v3*				
	$\alpha_{0,L1,p}$	$\alpha_{1,L1,p}$	$\lambda_{L1,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L_2\,p}$	$\alpha_{0,L1,p}$	$\alpha_{1,L1,p}$	λ_{L1}	$\epsilon_{rel\,p}$	$\epsilon_{L_2\,p}$
1	22.0426	1.4776	2.22	0.310×10^{-1}	0.208×10^{1}	29.2601	1.0236	2.66	0.120×10^{-1}	0.753
2	23.2134	1.9645	2.22	0.37×10^{-1}	0.292×10^{1}	30.5263	1.5314	2.66	0.350×10^{-1}	0.155×10^{1}
3	24.7292	2.3765	2.22	0.336×10^{-1}	0.322×10^{1}	31.8235	1.9320	2.66	0.3278×10^{-1}	0.195×10^{1}
4	26.2173	2.8389	2.22	0.210×10^{-1}	0.222×10^{1}	33.4198	2.3376	2.66	0.241×10^{-1}	0.188×10^{1}
5	26.7389	3.3793	2.22	0.276×10^{-1}	0.318×10^{1}	34.8262	2.7083	2.66	0.228×10^{-1}	0.190×10^{1}
6	27.6782	3.9264	2.22	0.304×10^{-1}	0.372×10^{1}	36.2342	3.1674	2.66	0.214×10^{-1}	0.210×10^{1}
7	28.6584	4.4486	2.22	0.282×10^{-1}	0.419×10^{1}	37.7846	3.5578	2.66	0.192×10^{-1}	0.209×10^{1}
8	29.6800	4.9881	2.22	0.296×10^{-1}	0.462×10^{1}	39.1828	3.8992	2.66	0.180×10^{-1}	0.212×10^{1}
9	30.4458	5.5287	2.22	0.267×10^{-1}	0.501×10^{1}	40.3643	4.3458	2.66	0.184×10^{-1}	0.236×10^{1}
10	31.4306	6.0755	2.22	0.247×10^{-1}	0.529×10^{1}	42.0511	4.7259	2.66	0.1451×10^{-1}	0.205×10^{1}
11	-	-	-	-	-	43.3376	5.2321	2.66	0.153×10^{-1}	0.272×10^{1}
12	-	-	-	-	-	46.5980	5.4366	2.66	0.178×10^{-1}	0.278×10^{1}

Durch die Anwendung der QR-Methode auf das überdefinierte lineare System mit

den Koeffizienten aus Tab. 4.2 kann die elektrische Leistung mit Gl. (4.11) und Gl. (4.12) approximiert werden.

$$\begin{aligned}
&\textbf{Ivy Bridge (E5-2690v2)}: \\
P_{II}(f,p)_{ADD,L1} &= (21.0374 + 1.0993 \times p) + (0.967\,57 + 0.4969 \times p) \times f^{2.22}; \\
p &\in (1,2,3,4,5,6,7,8,9,10); \\
1.199 &\leq f \leq 3.487; \\
\epsilon_{rel} = 0.02;\ \epsilon_{L^2} &= 2.352;\ \epsilon_{abs} = 1.417;
\end{aligned} \tag{4.11}$$

$$\begin{aligned}
&\textbf{Haswell (E5-2680v3)}: \\
P_{II}(f,p)_{ADD,L1} &= (27.5085 + 1.4783 \times p) + (0.782\,90 + 0.3911 \times p) \times f^{2.66}; \\
p &\in (1,2,3,4,5,6,7,8,9,10,11,12); \\
1.204 &\leq f \leq 2.615; \\
\epsilon_{rel} = 0.036;\ \epsilon_{L^2} &= 4.246;\ \epsilon_{abs} = 1.470;
\end{aligned} \tag{4.12}$$

Interessant sind folgende Unterschiede:

- Der statische Anteil ${\alpha_0}^{Haswell} = 27.5085 + 1.4783 \times p$ [Watt] ist deutlich größer als ${\alpha_0}^{IvyBridge} = 21.0374 + 1.0993 \times p$ [Watt]. Dafür ist unter anderem der größer L3-Cache des *Haswell* Prozessors verantwortlich.

- Die Potenz $\lambda^{Haswell} = 2.66$ ist deutlich größer als die Potenz $\lambda^{IvyBridge} = 2.22$. Allerdings sind die beiden Skalierungskoeffizienten des dynamischen Anteils der elektrischen Leistung für *Haswell* um ~ 20 % kleiner. Dies kann damit erklärt werden, dass der L3-Ring des *Haswell* Prozessors während der Ausführung mit einer niedrigen Frequenz getaktet war. Somit steigt die elektrische Leistung des *Haswell* Prozessors mit der CPU-Frequenz deutlich schneller als die elektrische Leistung des *Ivy Bridge* Prozessors. Dies erklärt, warum die maximale CPU-Frequenz des *Ivy Bridge* Prozessors deutlich höher als die vom *Hawell* ist.

4.5.1.3. Energiekosten im First Level Cache

Unter Energiekosten wird eine Menge der Energie in Joule verstanden, die für die Durchführung einer Gleitkommazahl-Operation verbraucht wird. Damit ergibt sich Gl. (4.13) für die Energiekosten.

$$E = \frac{P[Watt]}{R[GFlop/s]} = \frac{P}{R}\frac{[Joule]}{[GFlop]} \tag{4.13}$$

Mit Gl. (4.4) und Gl. (4.7) in Gl. (4.13) eingesetzt folgt Gl. (4.14). Bei der Berechnung des relativen Fehlers der Energiekosten-Approximation werden die entsprechenden

Fehler der Rechenleistungs-Approximation und der Approximation der elektrischen Leistung nach [78] summiert.

$$E = \frac{P_{II}(f,p)_l}{R_{II}(f,p)_l} = \frac{(\tilde{\alpha}_{0,0,l} + \tilde{\alpha}_{0,1,l} \times p) + (\tilde{\alpha}_{1,0,l} + \tilde{\alpha}_{1,1,l} \times p) \times f^{\tilde{\lambda}_l}}{(\tilde{\beta}_{1,1,l} \times p) \times f^{\tilde{\gamma}_l}} \tag{4.14}$$

Werden Gl. (4.12), (4.11) und Gl. (4.10), (4.9) in Gl. (4.14) eingesetzt folgt Gl. (4.15). Diese Gleichung beschreibt den Energie-Verbrauch der beiden Prozessoren und deren Speichermodule abhängig von f und p. Die grünen Pfeile neben den Energieanteilen zeigen, dass die Erhöhung von p immer die Kosten reduziert. Der Einfluss der CPU-Frequenz ist nicht so eindeutig: Die Erhöhung der CPU-Frequenz erhöht ebenso die Effizienz. Aber zu hohe CPU-Frequenzen können die Effizienz auch reduzieren. Der Vergleich der letzten Termen der beiden Gleichungen zeigt, dass der Energieverbrauch des *Haswell* Prozessors mit der CPU-Frequenz deutlich schneller wächst.

Ivy Bridge (E5-2690v2) :

$$\begin{aligned} E(f,p)_{L1} &= 12.337806 && \times \tfrac{1}{p \times f} && \downarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\ &+ 0.64681 && \times \tfrac{1}{f} && \rightarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\ &+ 0.56930 && \times \tfrac{f^{1.22}}{p} && \downarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \\ &+ 0.292368 && \times f^{1.22} && \rightarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \end{aligned}$$

$$\epsilon_{rel} = 0.02;$$

(4.15)

Haswell (E5-2680v3) :

$$\begin{aligned} E(f,p)_{L1} &= 10.7583 \times \tfrac{1}{p \times f} && \downarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\ &+ 0.57815 \times \tfrac{1}{f} && \rightarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\ &+ 0.30618 \times \tfrac{f^{1.66}}{p} && \downarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \\ &+ 0.15296 \times f^{1.66} && \rightarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \end{aligned}$$

$$\epsilon_{rel} = 0.036;$$

Die oben erwähnten Verhältnisse zwischen der CPU-Frequenz, der Thread-Anzahl und den Energiekosten sind in Abb. 4.4 für den Prozessor *Ivy Bridge* und in Abb. 4.5 für den Prozessor *Haswell* zu erkennen:

- Die Berechnung mit wenigen Threads braucht deutlich mehr Energie als die Berechnung mit der maximalen Anzahl der Threads.
- Die minimalen Energiekosten des *Ivy Bridge* Prozessors werden mit 10 Threads

und der CPU-Frequenz von $\sim$ 2.2 GHz erreicht. Der *Haswell* erreicht die minimalen Energiekosten mit 12 Threads und der CPU-Frequenz von nur $\sim$ 1.8 GHz. Mit einer höheren CPU-Frequenz steigen die Kosten der beiden Prozessoren. Hierbei sei erwähnt, dass wenn die statischen Energiekosten der anderen Komponenten berücksichtigt werden, sich das Minimum nach rechts schiebt.

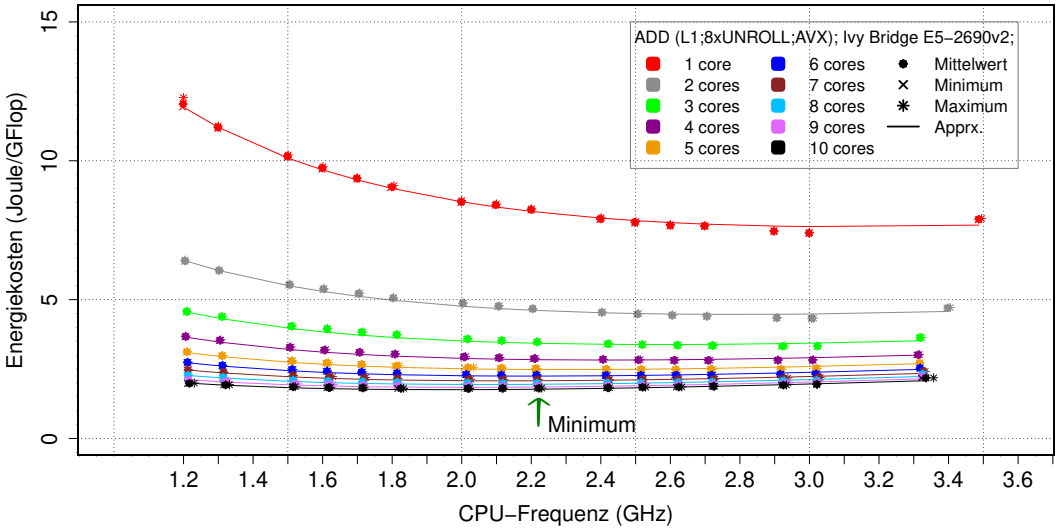

Abb. 4.4.: Energiekosten der Kernel-Operation *Add* auf *Ivy Bridge E5-2690v2* im Falle des Haltens der Daten im L1-Cache

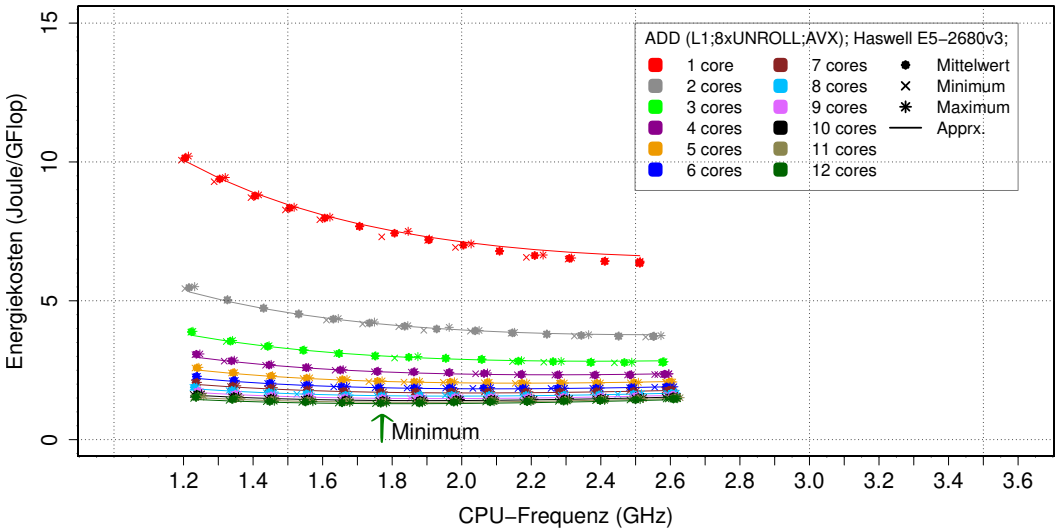

Abb. 4.5.: Energiekosten der Kernel-Operation *Add* auf *Haswell E5-2680v3* (unten) im Falle des Haltens der Daten im L1-Cache

4.5.2. Second Level Cache

Im Falle des Haltens der Daten im L2-Cache bearbeiten die Prozessorkerne die Daten sowohl im L1-Cache als auch im L2-Cache.

4.5.2.1. Rechenleistung im Second Level Cache

Zwei Diagramme in Abb. 4.6 zeigen die Rechenleistung der beiden Prozessoren. Die Rechenleistung wächst linear mit der Anzahl der Threads und der CPU-Frequenz. Wie im Fall des Haltens der Daten im L1-Cache können die Operationen unabhängig voneinander auf den aktiven Prozessorkernen ausgeführt werden. Der Unterschied zwischen der Rechenleistung der beiden Prozessoren ist im Vergleich zum Falle des Haltens der Daten im L1-Cache deutlich kleiner geworden und hat sich von 33 % auf 23 % gesenkt. In Abschnitt 5 werden die Gründe dafür mit Hilfe eines vereinfachten Prozessormodells erläutert.

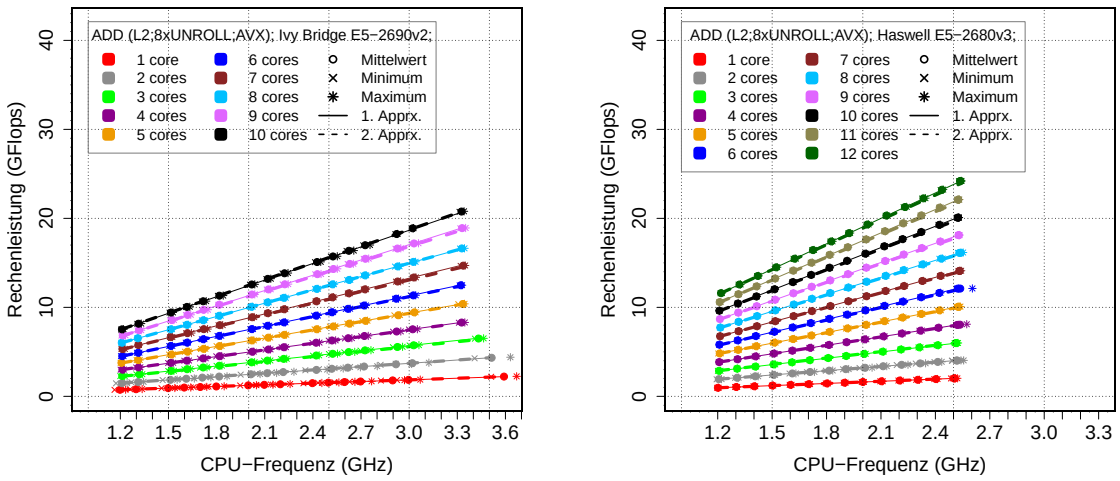

Abb. 4.6.: Rechenleistung auf *Ivy Bridge E5-2690v2* (links) und auf *Haswell E5-2680v3* (rechts) im Falle des Haltens der Daten im L2-Cache

Die Koeffizienten der Approximation über f sind in Tab. E.2 des Anhangs E auf Seite 166 für alle p zusammengefasst. Da die Rechenleistung linear skaliert ist, konnten diese mit einer hohen Genauigkeit auch über p interpoliert werden. Gl. (4.16) und Gl. (4.17) zeigen das Resultat der Interpolation der Messdaten für beide Prozessoren. Obwohl die Potenz der CPU-Frequenz für *Haswell* um einen geringen Betrag von

0.01 größer als 1.0 ist, ist der entsprechende Beitrag der Potenzfunktion der gleichen Ordnung wie der Interpolationsfehler. Der Skalierungsfaktor des *Haswell* Prozessors von 0.8 [FLOP] ist höher als der Skalierungsfaktor des *Ivy Bridge* Prozessors von 0.62 [FLOP]. Im Vergleich zum Fall des Haltens der Daten im L1-Cache ist der Unterschied in der Rechenleistung kleiner geworden. Der Grund dafür liegt in der Funktionsweise der Pipeline. Dies wird im Abschnitt 5.1.2.5 im Detail diskutiert.

Ivy Bridge (E5-2690v2) :

$$\begin{aligned} R_{II}(f,p)_{L2} &= 0.61678 \times p \times f; \\ \epsilon_{L^2} &= 1.098\,618; \\ \epsilon_{rel} &= 0.028\,178; \\ \epsilon_{abs} &= 0.455\,637; \\ 1 \leq &p \leq 10 \wedge f \geq 0.0; \end{aligned} \tag{4.16}$$

Haswell (E5-2680v3) :

$$\begin{aligned} R_{II}(f,p)_{L2} &= 0.80054 \times p \times f^{1.01}; \\ \epsilon_{L^2} &= 1.083\,144; \\ \epsilon_{rel} &= 0.018\,633; \\ \epsilon_{abs} &= 0.408\,560; \\ 1 \leq &p \leq 12 \wedge f \geq 0.0; \end{aligned} \tag{4.17}$$

4.5.2.2. Elektrische Leistung im Second Level Cache

In Abb. 4.7 und 4.8 ist die elektrische Leistung der Prozessoren und der Speichermodule in Abhängigkeit von der CPU-Frequenz dargestellt. Der relative Fehler der elektrischen Leistung-Interpolation ist sowohl für den *Ivy Bridge* Prozessor als auch für den *Haswell* kleiner als drei Prozent. Die einzelnen Koeffizienten der Approximation sind in Tab. E.3 angegeben.
Die elektrische Leistung über f und p der beiden Prozessoren ist mit Gl. (4.18) und Gl. (4.19) approximiert. Der *Ivy Bridge* Prozessor zeigt kaum Unterschiede in der elektrischen Leistung im Falle des Haltens der Daten im L1- und im L2-Cache. Dies kann ebenso im Vergleich von Gl. (4.11) und Gl. (4.18) erkannt werden.
Der *Haswell* Prozessor reduziert dagegen leicht die elektrische Leistung im Falle des Haltens der Daten im L2-Cache. Ebenso lassen sich die Unterschiede im Vergleich der beiden Gleichungen für den *Haswell* erkennen. Während die von f und p un-

abhängigen statischen Anteile der elektrischen Leistung gleich sind, ist der von p abhängige Anteil $1.4783 \times p$ im Falle des Haltens der Daten im L1-Cache wesentlich größer als der entsprechende Anteil $0.7763 \times p$ im Falle des Haltens der Daten im L2-Cache. Welche Eigenschaften des Prozessors oder der Umgebung dafür verantwortlich sind, muss zusätzlich untersucht werden. Eine weitere Analyse wurde nicht durchgeführt, da diese den zeitlichen Rahmen der Arbeit zu stark ausgedehnt hätte.

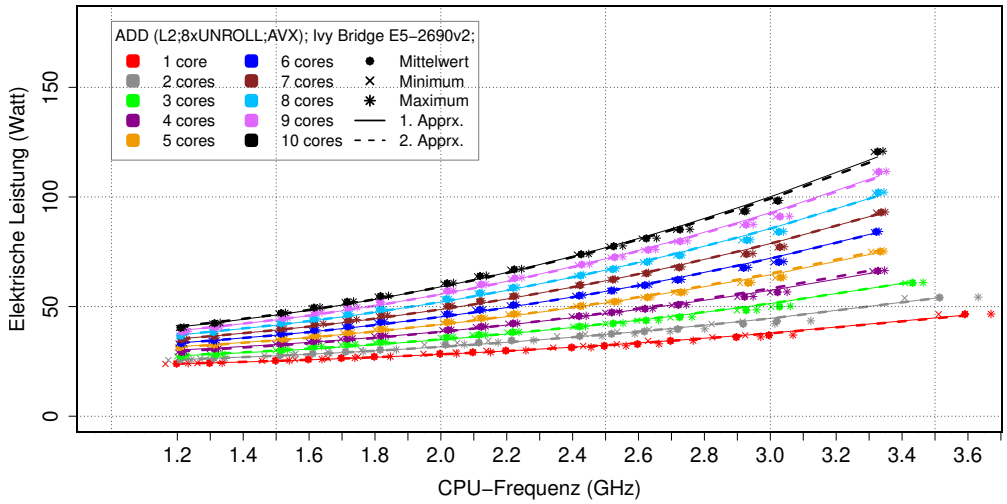

Abb. 4.7.: Elektrische Leistung des *Ivy Bridge* Prozessors im Falle des Haltens der Daten im L2-Cache

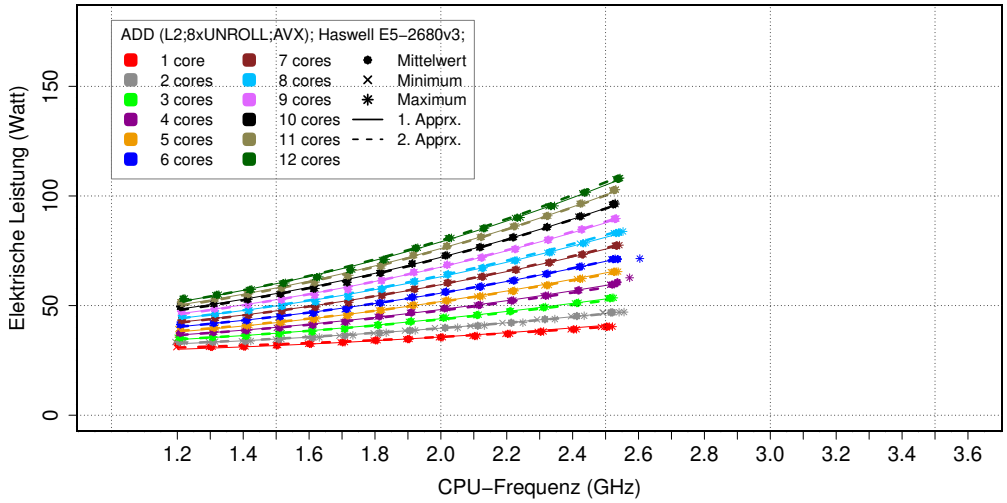

Abb. 4.8.: Elektrische Leistung des *Haswell* Prozessors im Falle des Haltens der Daten im L2-Cache

Ivy Bridge (E5-2690v2) :

$$P_{II}(f,p)_{ADD,L2} = (20.7740 + 1.1701 \times p) + (0.8470 + 0.4707 \times p) \times f^{2.26};$$
$$p \in (1,2,3,4,5,6,7,8,9,10); \quad (4.18)$$
$$1.198 \leq f \leq 3.591;$$
$$\epsilon_{rel} = 0.02;\ \epsilon_{L^2} = 2.208;\ \epsilon_{abs} = 1.311;$$

Haswell (E5-2680v3) :

$$P_{II}(f,p)_{ADD,L2} = (27.5505 + 0.7763 \times p) + (0.9583 + 0.7446 \times p) \times f^{2.12};$$
$$p \in P(1,2,3,4,5,6,7,8,9,10,11,12); \quad (4.19)$$
$$1.201 \leq f \leq 2.540;$$
$$\epsilon_{rel} = 0.0262;\ \epsilon_{L^2} = 2.956;\ \epsilon_{abs} = 1.3328;$$

Interessant sind folgende Unterschiede:

- Der statische Anteil der elektrischen Leistung hat sich im Vergleich zum Fall des Haltens der Daten im L1-Cache wenig geändert. Der dynamische Anteil für den *Ivy Bridge* Prozessor hat sich ebenfalls nur gering geändert.
- Die Potenz ${\lambda_{L2}}^{Haswell} = 2.12$ ist deutlich niedriger als die Potenz ${\lambda_{L1}}^{Haswell} = 2.66$ im Falle des Haltens der Daten im L1-Cache. Allerdings verdoppelte sich dafür der erste Skalierungskoeffizient des dynamischen Anteils der elektrischen Leistung und stieg auf 0.7446.

4.5.2.3. Energiekosten im Second Level Cache

Trotzdem reduzieren sich die Energiekosten mit der Erhöhung der Threads-Anzahl. Im Unterschied zum L1-Cache unterscheidet sich kaum die Energieeffizienz der beiden Prozessoren. Allerdings erweist sich der Prozessor *Haswell* effizienter wegen mehr zur Verfügung stehender Prozessorkerne. Die Diagramme in Abb. 4.9 zeigen die Energiekosten der Prozessoren und der Speichermodule in Abhängigkeit von der CPU-Frequenz.

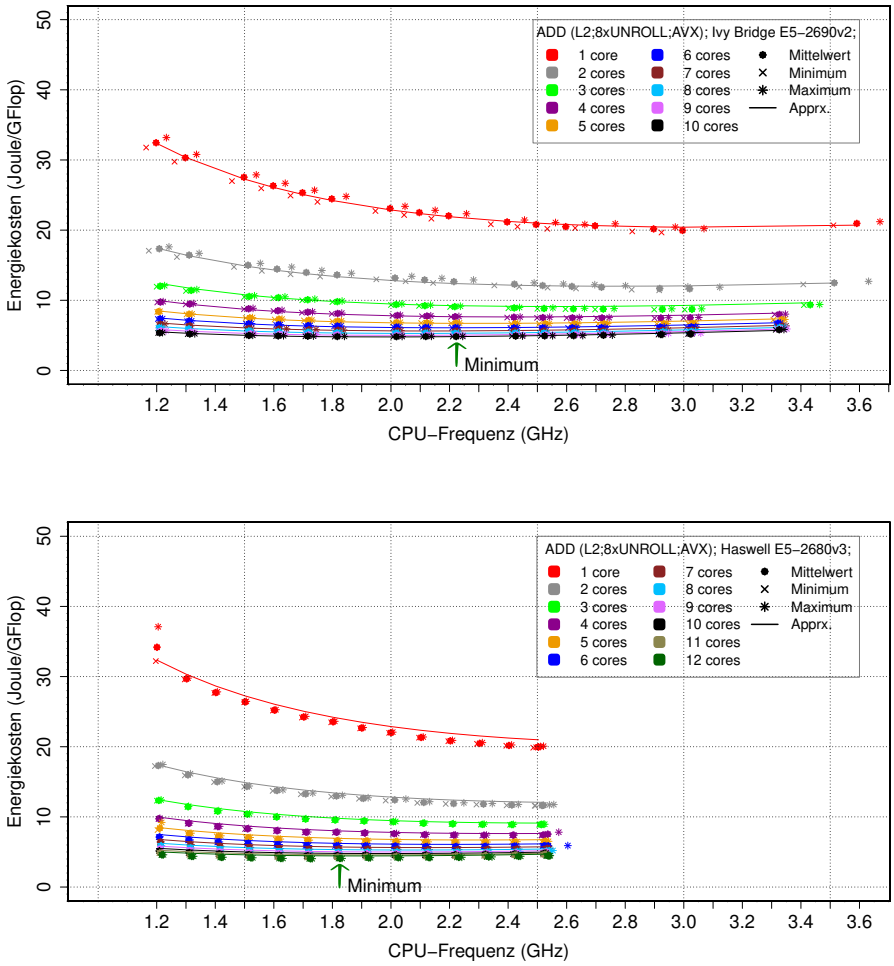

Abb. 4.9.: Energiekosten der beiden Prozessoren *Ivy Bridge E5-2690v2* (oben) und *Haswell E5-2680v3* (unten) im Falle des Haltens der Daten im L2-Cache

Werden Gl. (4.18), Gl. (4.19), Gl. (4.16) und (4.17) in Gl. (4.14) eingesetzt folgt Gl. (4.20) und Gl. (4.21).

Ivy Bridge (E5-2690v2) :

$$
\begin{aligned}
E(f,p)_{L2} &= 33.6814 \times \frac{1}{p \times f} \quad \downarrow_{\mathrm{p}} \quad \downarrow_{\mathrm{f}} \\
&+ 1.8971 \times \frac{1}{f} \quad \rightarrow_{\mathrm{p}} \quad \downarrow_{\mathrm{f}} \\
&+ 1.3748 \times \frac{f^{1.26}}{p} \quad \downarrow_{\mathrm{p}} \quad \uparrow_{\mathrm{f}} \\
&+ 0.7632 \times f^{1.26} \quad \rightarrow_{\mathrm{p}} \quad \uparrow_{\mathrm{f}}
\end{aligned}
\tag{4.20}
$$

$\epsilon_{rel} = 0.0282;$

Haswell (E5-2680v3) :

$$
\begin{aligned}
E(f,p)_{L2} &= 34.4148 \times \tfrac{1}{p \times f} \quad \downarrow_{\mathrm{p}} \quad \downarrow_{\mathrm{f}} \\
&+ 0.9697 \times \tfrac{1}{f} \quad \rightarrow_{\mathrm{p}} \quad \downarrow_{\mathrm{f}} \\
&+ 1.1971 \times \tfrac{f^{1.19}}{p} \quad \downarrow_{\mathrm{p}} \quad \uparrow_{\mathrm{f}} \\
&+ 0.9301 \times f^{1.19} \quad \rightarrow_{\mathrm{p}} \quad \uparrow_{\mathrm{f}}
\end{aligned}
\qquad (4.21)
$$

$\epsilon_{rel} = 0.0262$;

Diese Gleichungen beschreiben den Energie-Verbrauch der beiden Prozessoren und deren Speichermodule abhängig von der CPU-Frequenz f und der Thread-Anzahl p im Falle des Haltens der Daten im L2-Cache.
Der Vergleich der Gleichungen zeigt die ähnlichen Energiekosten der beiden Prozessoren, wenn f und p in den zulässigen Bereichen sind. Im Vergleich zum Fall des Haltens der Daten im L1-Cache sind die Kosten um etwa dreimal gestiegen.

4.5.3. Last Level Cache

Im Unterschied zu den L1- und L2-Caches bildet der L3-Cache einen gemeinsamen Zwischenspeicher für alle Prozessorkerne. Das Cache-Kohärenz-Protokoll und die Bandbreite des L3-Ringes können die Rechenleistung stören. Zur Beschreibung des L3-Ringes siehe Abschnitt 5.2.2. Dies kann dazu führen, dass die Rechenleistung sowohl nach der CPU-Frequenz f als auch nach der Thread-Anzahl p nicht mehr linear skaliert.

4.5.3.1. Rechenleistung im Last Level Cache

In Abb. 4.10 und 4.11 wird die Rechenleistung der beiden Prozessoren im Falle des Haltens der Daten im L3-Cache gezeigt. Die Rechenleistung des Prozessors *Ivy Bridge* konnte mit einer Genauigkeit von 3.6 % sowohl über die CPU-Frequenz f als auch über die Thread-Anzahl p linear interpoliert werden, wie dies im vorangegangenen Abschnitt gezeigt wurde. Die Rechenleistung des Prozessors *Haswell* konnte dagegen nicht auf die gleiche Weise interpoliert werden, weil die Potenzen der CPU-Frequenz für die verschiedene Thread-Anzahl p ungleich zueinander und kleiner als Eins sind. Die Koeffizienten für die Interpolation der Rechenleistung über die CPU-Frequenz f sind in Tab. E.4 des Anhangs E auf Seite 168 für die beiden Prozessoren angegeben.

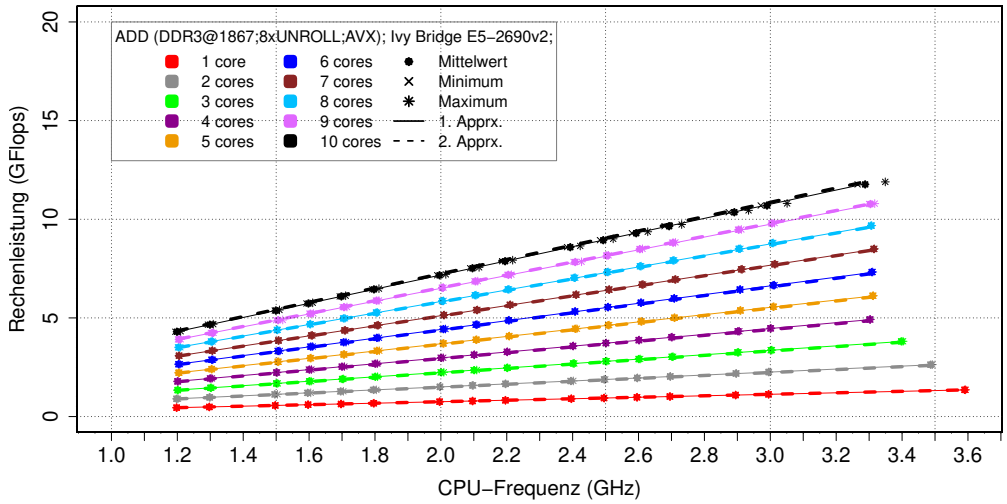

Abb. 4.10.: Rechenleistung auf *Ivy Bridge E5-2690v2* im Falle des Haltens der Daten im L3-Cache. Die Rechenleistung skaliert mit der CPU-Frequenz und der Prozessorkernanzahl linear.

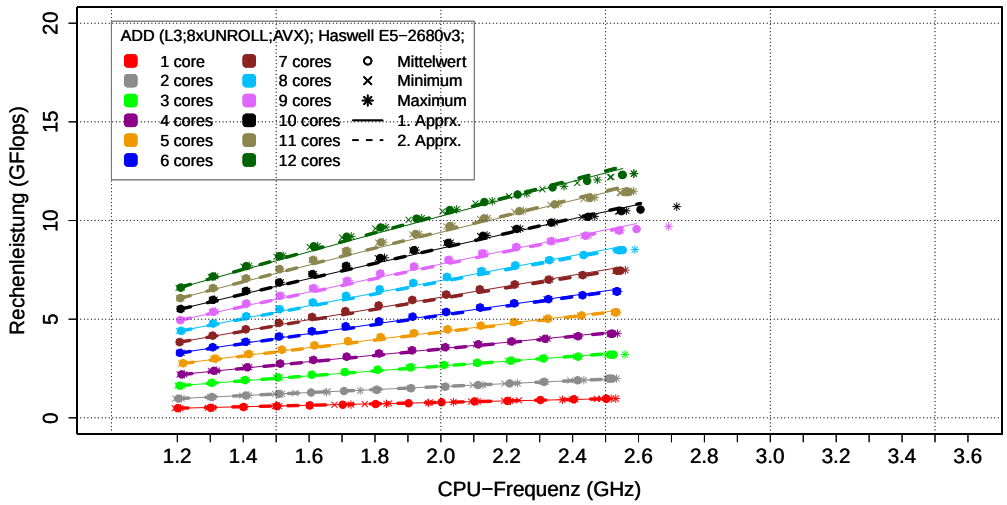

Abb. 4.11.: Rechenleistung auf *Haswell E5-2680v3* im Falle des Haltens der Daten im L3-Cache. Die Rechenleistung degradiert bei den höheren CPU-Frequenzen.

Inwieweit dabei die Rechenleistung vom linearen Verhalten abweicht, kann durch die Analyse der Ableitung der Rechenleistung gezeigt werden. Unter Verwendung von Gl. (4.22) lässt sich die erste diskrete Ableitung der Rechenleistung nach der

CPU-Frequenz berechnen.

$$\begin{aligned}
\frac{dR_{l,p}}{df}(f_i) &= \frac{R_{f_{i+1},l,p} - R_{f_i,l,p_i}}{f_{i+1} - f_i}; \\
f: &\quad \textit{CPU-Frequenz}; \\
p: &\quad \textit{Anzahl der Threads}; \\
l: &\quad \textit{Hierarchiestufe des Speichers } l \in \{L1, L2, L3, RAM\}; \\
f_i: &\quad \textit{gesetzte CPU-Frequenz } f_i \in \{f_0, f_1, .., f_{m_f-1}\}; \\
m_f: &\quad \textit{Anzahl der CPU-Frequenzstufen}; \\
n_p: &\quad \textit{Anzahl der Prozessorkerne}; \\
R_{f_i,l,p}: &\quad \textit{gemessene Rechenleistung für } f = f_i;
\end{aligned} \tag{4.22}$$

Die in Abb. 4.12 gezeigte Ableitung bestätigt, dass die Rechenleistung auf dem Prozessor *Ivy Bridge* mit der CPU-Frequenz annähernd linear wächst: Mit wenigen Ausnahmen ist die Ableitung $\frac{dR(f)}{df}$ konstant. Es ist deutlich zu erkennen, dass die diskrete Ableitung für den Prozessor *Haswell* in zwei Gruppen aufgeteilt wird. Die Ableitung für die wenigen aktiven Prozessorkerne ist verhältnismäßig konstant. Und die Ableitung für die mehreren aktiven Prozessorkerne senkt sich mit der wachsenden CPU-Frequenz ab.

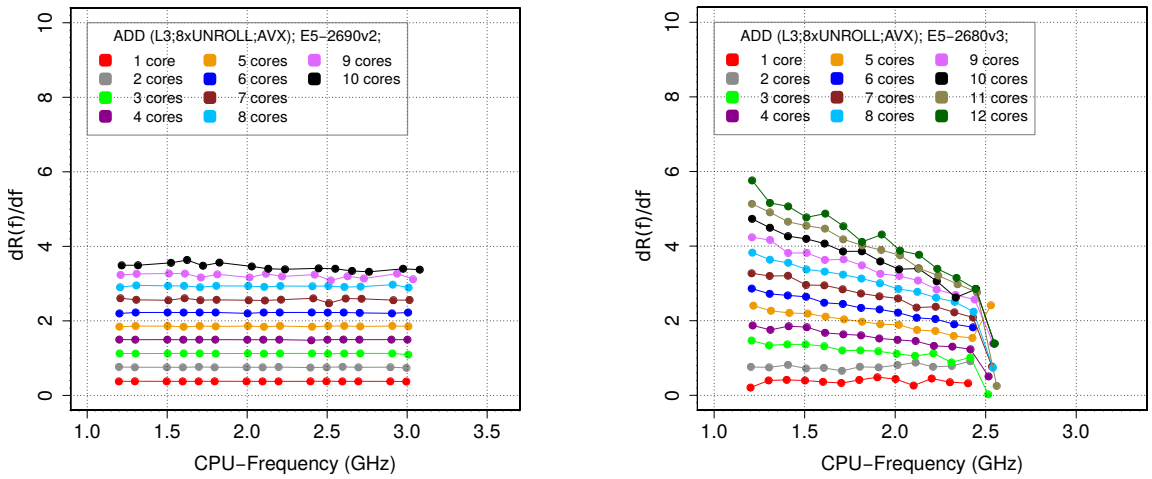

Abb. 4.12.: Erste Ableitung der Rechenleistung nach der CPU-Frequenz für *Ivy Bridge E5-2690v2* (links) und für *Haswell E5-2680v3* (rechts) dargestellt

Im Folgenden wird die Approximation der Rechenleistung R_{II} betrachtet, um die Skalierungseigenschaften der Prozessoren über p und f zu untersuchen. Um die Genauigkeit und die Aussagekraft der Approximation zu erhöhen, werden die Messergebnisse über die Thread-Anzahl p für die unterschiedlichen Bereiche unabhängig voneinander interpoliert:

Ivy Bridge (E5-2690v2) :

$$\begin{aligned} R_{II}(f,p)_{ADD,L3} &= (0.3763 \times p) \times f; \\ p &\in (1,2,3,4); \\ \\ R_{II}(f,p)_{ADD,L3} &= (0.3690 \times p) \times f; \\ p &\in (5,6,7,8); \\ \\ R_{II}(f,p)_{ADD,L3} &= (0.3610 \times p) \times f; \\ p &\in (9,10); \\ \\ \epsilon_{rel} &\leq 0.0126; 1.2 \leq f \leq 3.6; \end{aligned} \tag{4.23}$$

Haswell (E5-2680v3) :

$$\begin{aligned} R_{II}(f,p)_{ADD,L3} &= (0.4113 \times p) \times f^{0.95}; \\ p &\in (1,2); \\ \\ R_{II}(f,p)_{ADD,L3} &= (0.4587 \times p) \times f^{0.94}; \\ p &\in (3,4,5,6,7,8); \\ \\ R_{II}(f,p)_{ADD,L3} &= (0.465 \times p) \times f^{0.9}; \\ p &\in (9,10); \\ \\ R_{II}(f,p)_{ADD,L3} &= (0.4675 \times p) \times f^{0.88}; \\ p &\in (11,12); \\ \\ \epsilon_{rel} &\leq 0.03; 1.2 \leq f \leq 2.6 \end{aligned} \tag{4.24}$$

Der Skalierungsfaktor des *Ivy Bridge* Prozessors sinkt von 0.3763 [FLOP] auf 0.3610 [FLOP], wenn die Anzahl der Threads steigt. Die Skalierung nach der CPU-Frequenz ist im ganzen Frequenzbereich linear. Dies ist anders, als die Approximation der Rechenleistung des *Haswell* Prozessors zeigt. Während die Rechenleistung mit der Anzahl der Threads linear steigt, besitzt diese leichte Degradierung mit der Erhöhung der CPU-Frequenz. Die Gründe für das unterschiedliche Verhalten der beiden Prozessoren liegen im Aufbau des L3-Caches. In Kapitel 5 wird die Prozessorarchitektur im Detail diskutiert.

4.5.3.2. Elektrische Leistung im Last Level Cache

Im Vergleich Abb. 4.13 zu Abb. 4.2 und zu Abb. 4.7 ist zu erkennen, dass sich die elektrische Leistung des Prozessors *Ivy Bridge* im Falle des Haltens der Daten im L3-Cache zu den Fällen des Haltens der Daten in den L1- und L2-Caches sehr ähnlich verhält. Die elektrische Leistung des Prozessors *Haswell* in Abb. 4.14 unterscheidet sich jedoch stark davon.

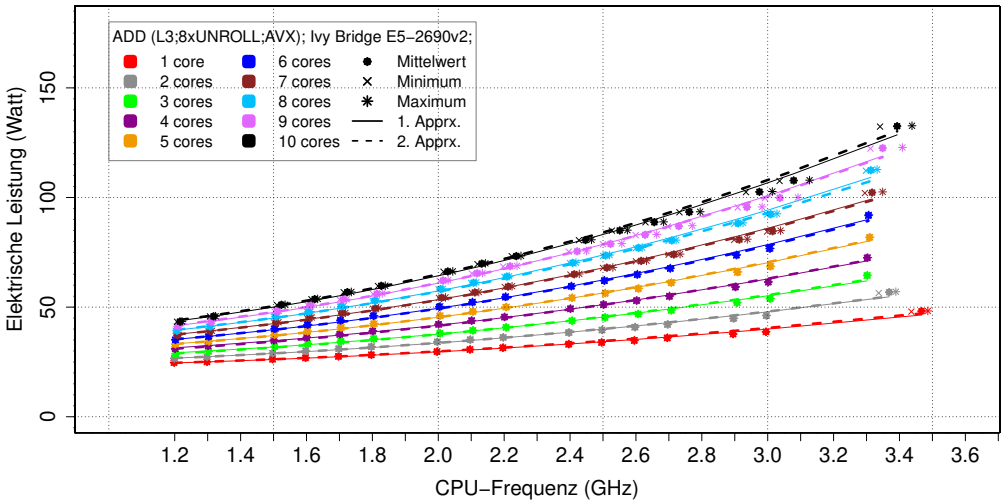

Abb. 4.13.: Elektrische Leistung des *Ivy Bridge* Prozessors *E5-2690v2* im Falle des Haltens der Daten im L3-Cache

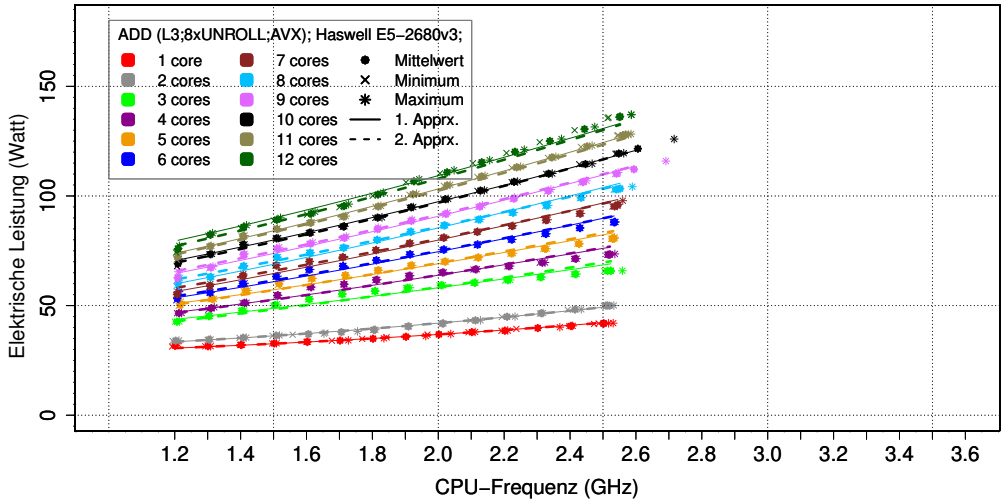

Abb. 4.14.: Elektrische Leistung des *Haswell* Prozessors *E5-2680v3* im Falle des Haltens der Daten im L3-Cache

Wie in Abb. 4.14 zu erkennen ist, verhält sich die elektrische Leistung des Prozessors bei der Berechnung mit einem und zwei Threads anders als bei der Berechnung mit mehr als zwei Threads. Dies habe ich im vorangegangenen Abschnitt mit einem kleineren Ausmaß für die Rechenleistung beobachtet. Der Grund dafür ist, dass sich die „Uncore"-Frequenz des Prozessors *Haswell* dynamisch ändert.
In Abb. 4.15 ist die Abhängigkeit zwischen der CPU-Frequenz und der „Uncore"-Frequenz für den betrachteten Fall dargestellt. Während die „Uncore"-Frequenz für mehr als zwei Threads annähernd konstant ist, steigt die „Uncore"-Frequenz in allen anderen Fällen zusammen mit der CPU-Frequenz.

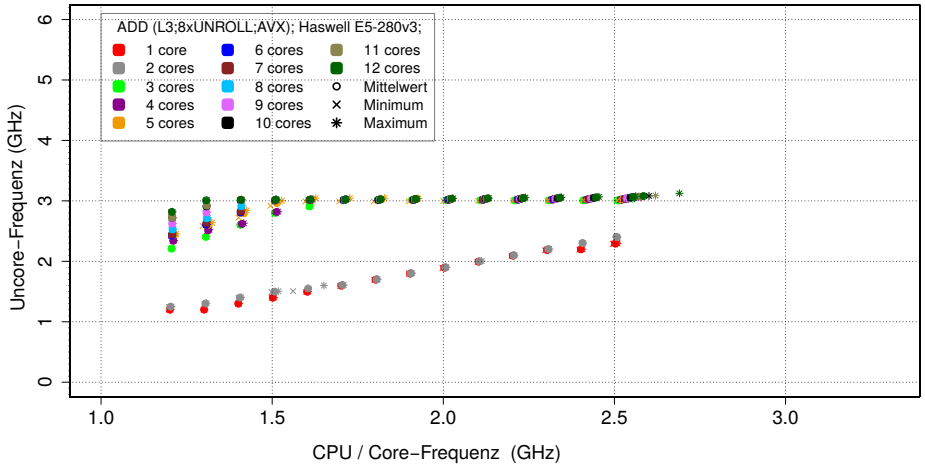

Abb. 4.15.: Abhängigkeit zwischen der „Uncore"-Frequenz und der CPU-Frequenz des Prozessors *Haswell E5-2680v3* im Falle des Haltens der Daten im L3-Cache

Bei genauerer Betrachtung der Approximations-Koeffizienten der elektrischen Leistung in Tab. 4.3 ist zu erkennen, dass sich die Koeffizienten und die Potenzen der CPU-Frequenz für $p = 1$ und $p = 2$ für die beiden Prozessoren nicht sehr stark unterscheiden. Wenn jedoch die „Uncore"-Frequenz unabhängig von der CPU-Frequenz hoch gesetzt wird, verringert sich die Potenz und dementsprechend vergrößert sich der Skalierungsfaktor β_1. Außerdem ist es zu erkennen, dass sich die statischen Anteile der elektrischen Leistung des Prozessors *Haswell* größer sind. Die durchgeführte Analyse liefert die Erkenntnis, dass die „Uncore"-Komponenten des Prozessors *Ivy Bridge* höchstwahrscheinlich mit der CPU-Frequenz und nicht mit einer konstanten Frequenz getaktet sind.[2]

[2]Keine explizite Angaben dazu in der Literatur gefunden

Tab. 4.3.: Koeffizienten der Approximation der elektrischen Leistung $P_I(f)_{L3,p} = \alpha_{1\,L3,p} \times f^{\lambda_{L3,p}}$ im Falle des Haltens der Daten im L3-Cache

p	*Ivy Bridge E5-2690v2*					*Haswell E5-2680v3*				
	$\beta_{0,L3,p}$	$\beta_{1,L3,p}$	$\lambda_{L3,p}$	ϵ_{rel_p}	$\epsilon_{L_{2_p}}$	$\beta_{0,L3,p}$	$\beta_{1,L3,p}$	$\lambda_{L3,p}$	ϵ_{rel_p}	$\epsilon_{L_{2_p}}$
1	21.9436	1.7751	2.11	0.386×10^{-1}	0.247×10^{1}	26.8210	2.667	1.91	0.371×10^{-1}	0.138×10^{1}
2	23.1555	2.4324	2.11	0.39×10^{-1}	0.327×10^{1}	28.3247	3.6344	1.91	0.127×10^{-1}	0.136×10^{1}
3	24.764	2.9928	2.11	0.398×10^{-1}	0.336×10^{1}	30.0406	10.5698	1.41	0.524×10^{-1}	0.729×10^{1}
4	26.1482	3.6125	2.11	0.23×10^{-1}	0.269×10^{1}	30.6848	12.4762	1.41	0.591×10^{-1}	0.780×10^{1}
5	26.9921	4.2593	2.11	0.27×10^{-1}	0.338×10^{1}	33.4652	13.4085	1.41	0.418×10^{-1}	0.647×10^{1}
6	27.965	4.9783	2.11	0.255×10^{-1}	0.366×10^{1}	33.4512	15.4801	1.41	0.429×10^{-1}	0.678×10^{1}
7	29.3379	5.5756	2.11	0.296×10^{-1}	0.443×10^{1}	34.0990	17.2232	1.41	0.539×10^{-1}	0.712×10^{1}
8	30.1296	6.3129	2.11	0.3×10^{-1}	0.507×10^{1}	35.6544	18.7293	1.41	0.413×10^{-1}	0.623×10^{1}
9	31.4544	6.8272	2.11	0.31×10^{-1}	0.566×10^{1}	39.8218	19.1627	1.41	0.339×10^{-1}	0.447×10^{1}
10	32.8913	7.2664	2.11	0.31×10^{-1}	0.651×10^{1}	45.0086	19.7059	1.41	0.238×10^{-1}	0.257×10^{1}
11	-	-	-	-	-	45.9798	21.5262	1.41	0.248×10^{-1}	0.303×10^{1}
12	-	-	-	-	-	51.1722	21.8637	1.41	0.491×10^{-1}	0.765×10^{1}

Durch die Anwendung der QR-Methode auf die Approximations-Koeffizienten aus Tab. 4.3 ergeben sich Gl. (4.25) und Gl. (4.26) für die *2. Approximation* der elektrischen Leistung über die CPU-Frequenz f und die Thread-Anzahl p. Die Koeffizienten werden aus oben genannten Gründen für $p < 3$ und $p > 2$ für den Prozessor *Haswell* separat berechnet.

Ivy Bridge (E5-2690v2):

$$\begin{aligned} P_{II}(f,p)_{ADD,L3} &= (20.67272 + 1.23735 \times p) + (1.21206 + 0.61658 \times p) \times f^{2.11}; \\ p &\in (1,2,3,4,5,6,7,8,9,10); \\ 1.198 &\leq f \leq 3.393; \\ \epsilon_{rel} &= 0.02; \\ \epsilon_{L^2} &= 3.573; \\ \epsilon_{abs} &= 1.664; \end{aligned} \tag{4.25}$$

Haswell (E5-2680v3):

$$\begin{aligned} P_{II}(f,p)_{ADD,L3} &= (25.31736 + 1.50367 \times p) + (1.6995 + 0.96749 \times p) \times f^{1.91}; \\ p &\in (1,2); \\ 1.201 &\leq f \leq 2.540; \\ P_{II}(f,p)_{ADD,L3} &= (21.3318 + 2.2141 \times p) + (7.74370 + 1.23611 \times p) \times f^{1.41}; \\ p &\in (3,4,...,12); \\ 1.201 &\leq f \leq 2.606; \\ \epsilon_{rel} &\leq 0.03; \\ \epsilon_{L^2} &\leq 6.054; \\ \epsilon_{abs} &\leq 2.338; \end{aligned} \tag{4.26}$$

Interessant sind folgende Unterschiede:

- Der statische Anteil für den *Ivy Bridge* Prozessor hat sich im Vergleich zu den Fällen des Haltens der Daten im L1- und L2-Cache wenig geändert.
- Die Potenz $\lambda_{L3}^{IvyBridge} = 2.11$ hat sich nur um einen kleinen Betrag von 0.11 verringert und beide Skalierungs-Koeffizienten des dynamischen Anteils sind um ~ 40 % gestiegen. Dies sind die zusätzlichen Kosten für den L3-Cache des *Ivy Bridge* Prozessors.
- Die Potenz $\lambda_{L3,p>2}^{Haswell} = 1.41$ ist deutlich niedriger als die Potenz $\lambda_{L1}^{Haswell} = 2.66$ im Falle des Haltens der Daten im L1-Cache geworden. Allerdings vervierfachte sich dafür der erste Skalierungs-Koeffizient des dynamischen Anteils der elektrischen Leistung und stieg von 0.9583 auf 7.743 70. Der wahrscheinliche Grund dafür ist, dass die Frequenz des *L3*-Ringes für mehrere Threads auf den maximalen Wert gesetzt war (siehe Abb. 4.15). Der entsprechende dynamische Anteil hängt nicht von der Anzahl der Threads p ab, weil der gesamte *L3*-Ring schon mit wenigen Threads aktiviert werden muss, um die Daten zwischen dem L3-Cache und dem L2-Cache zu transportieren (siehe Abschnitt 5.2.2). Der zweite Skalierungskoeffizient des dynamischen Anteils der elektrischen Leistung vom *Haswell* ist von 0.7446 im Falle des L2-Caches auf 1.236 11 gestiegen. Dies sind die zusätzliche Kosten für das Halten der Daten im L3-Cache.

4.5.3.3. Energiekosten im Last Level Cache

Werden Gl. (4.25), Gl. (4.26), Gl. (4.23) und (4.24) in Gl. (4.14) eingesetzt folgt Gl. (4.27). Die Approximation beschreibt die Energiekosten der beiden Prozessoren im Falle des Haltens der Datem im L3-Cache. Wie in Fällen des Haltens der Daten im L1-Cache und im L2-Cache reduzieren sich die Energiekosten mit der Erhöhung der Threads-Anzahl. Es ist deutlich zu sehen, dass die Energiekosten des *Haswell* Prozessors viel langsamer mit der CPU-Frequenz steigen als die Energiekosten des *Ivy Bridge* Prozessors, wenn mit mehreren Threads berechnet wird. Der dritte Term der Approximation des *Haswell* Prozessors für $p \in (11, 12)$ ist deutlich höher als in anderen Fällen. Dieses deutet darauf, dass die hohe „Uncore“-Frequenz des *Haswell*

Prozessors gegenüber dem *Ivy Bridge* die Energiekosten stark erhöhen.

Ivy Bridge (E5-2690v2) :

$$
\begin{aligned}
&E(f,p)_{L3} && = 53.1491 && \times \tfrac{1}{p \times f} && \downarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\
& && + 3.2882 && \times \tfrac{1}{f} && \rightarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\
& && + 3.2221 && \times \tfrac{f^{1.11}}{p} && \downarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \\
& && + 1.6385 && \times f^{1.11} && \rightarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \\
&p \in (1,2); \\
&\dots \\
&E(f,p)_{L3} && = 57.265 && \times \tfrac{1}{p \times f} && \downarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\
& && + 3.4276 && \times \tfrac{1}{f} && \tilde{\rightarrow}_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\
& && + 3.3575 && \times \tfrac{f^{1.11}}{p} && \downarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \\
& && + 1.708 && \times f^{1.11} && \rightarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \\
&p \in (9,10); \\
&\epsilon_{rel} \leq 0.03;
\end{aligned}
\tag{4.27}
$$

Haswell (E5-2680v3) :

$$
\begin{aligned}
&E(f,p)_{L3} && = 61.5545 && \times \tfrac{1}{p \times f^{0.95}} && \downarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\
& && + 3.6559 && \times \tfrac{1}{f^{0.95}} && \rightarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\
& && + 4.1211 && \times \tfrac{f^{0.96}}{p} && \downarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \\
& && + 2.3523 && \times f^{0.96} && \rightarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \\
&p \in (1,2); \\
&\dots \\
&E(f,p)_{L3} && = 45.6295 && \times \tfrac{1}{p \times f^{0.88}} && \downarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\
& && + 4.7360 && \times \tfrac{1}{f^{0.88}} && \rightarrow_{\mathrm{p}} && \downarrow_{\mathrm{f}} \\
& && + 16.5641 && \times \tfrac{f^{0.53}}{p} && \downarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \\
& && + 2.6441 && \times f^{0.53} && \rightarrow_{\mathrm{p}} && \uparrow_{\mathrm{f}} \\
&p \in (11,12); \\
&\epsilon_{rel} \leq 0.06;
\end{aligned}
$$

Abb. 4.16 und Abb. 4.17 zeigen die Energiekosten abhängig von der CPU-Frequenz für die beiden Prozessoren.

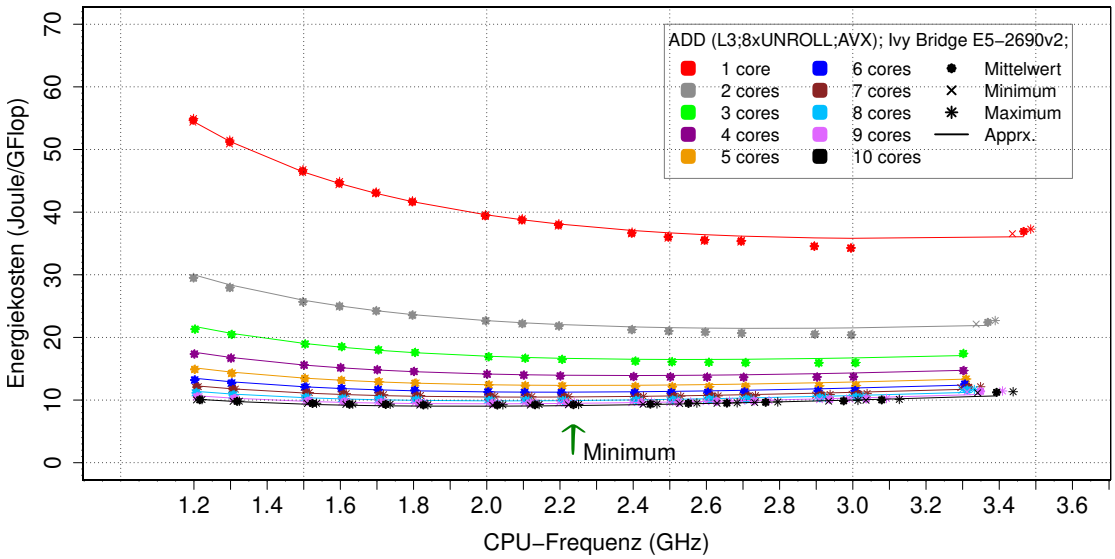

Abb. 4.16.: Energiekosten des Kernels *Add* auf *Ivy Bridge E5-2690v2* im Falle des Haltens der Daten im L3-Cache

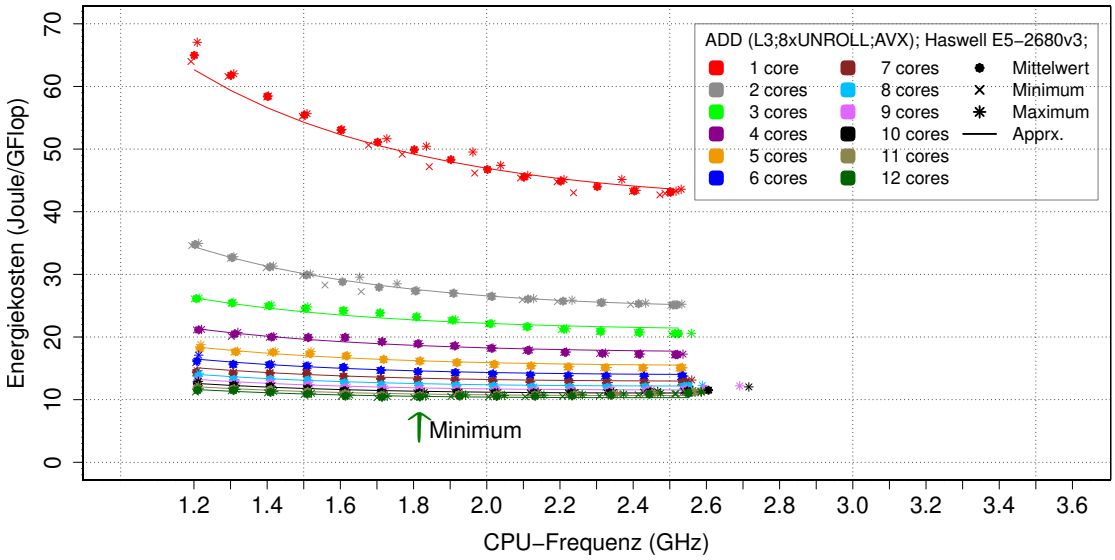

Abb. 4.17.: Energiekosten des Kernels *Add* auf *Haswell E5-2680v3* im Falle des Haltens der Daten im L3-Cache

Interessant ist, dass sich die optimale CPU-Frequenz nicht ändert, wenn die Daten im Cache (L1, L2 und L3) gehalten werden.

4.5.4. Hauptspeicher

Im Unterschied zum Cache bietet der Hauptspeicher eines Prozessors einen größeren gemeinsamen Speicherplatz. Ein Nachteil dieser Architektur besteht darin, dass die Zugriffszeiten auf den Inhalt des Speichers um mehr als eine Ordnung höher als auf den Inhalt des Caches sind.

4.5.4.1. Rechenleistung im Hauptspeicher

In Abb. 4.18 und Abb. 4.19 sind die Rechenleistung der beiden Prozessoren im Falle des Haltens der Daten im Hauptspeicher dargestellt. Es ist deutlich zu erkennen, dass die Rechenleistung der Prozessoren den Sättigungsbereich auf unterschiedliche Weise erreicht. Dies wird in Anhang II.II auf Seite 169 detailliert betrachtet. Die Analyse der ersten diskreten Ableitung der gemessenen Werte nach Gl. (4.22) zeigt, dass die CPU-Frequenz des Prozessors *Ivy Bridge* in zwei disjunkte Teilmengen aufgeteilt werden muss, um eine genaue Interpolation durchzuführen: $F_1 \equiv [f_0; \ldots; f_4]$ und $F_2 \equiv [f_5; \ldots; f_{15}]$.

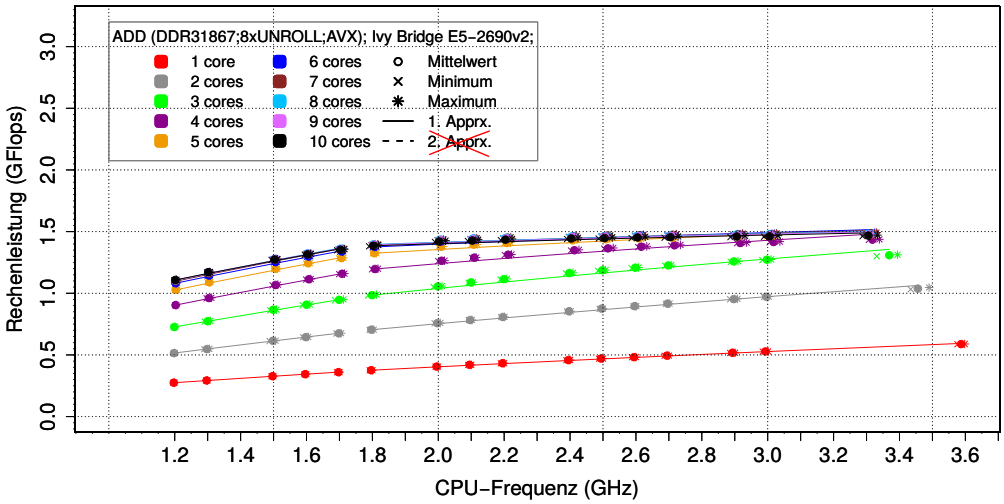

Abb. 4.18.: Rechenleistung auf *Ivy Bridge E5-2690v2* im Falle des Haltens der Daten im Hauptspeicher. Die gemessene Rechenleistung muss über zwei disjunkte CPU-Frequenz-Definitionsbereiche getrennt interpoliert werden. Daher ist die Approximation stückweise dargestellt und bei 1.8 GHz unterbrochen.

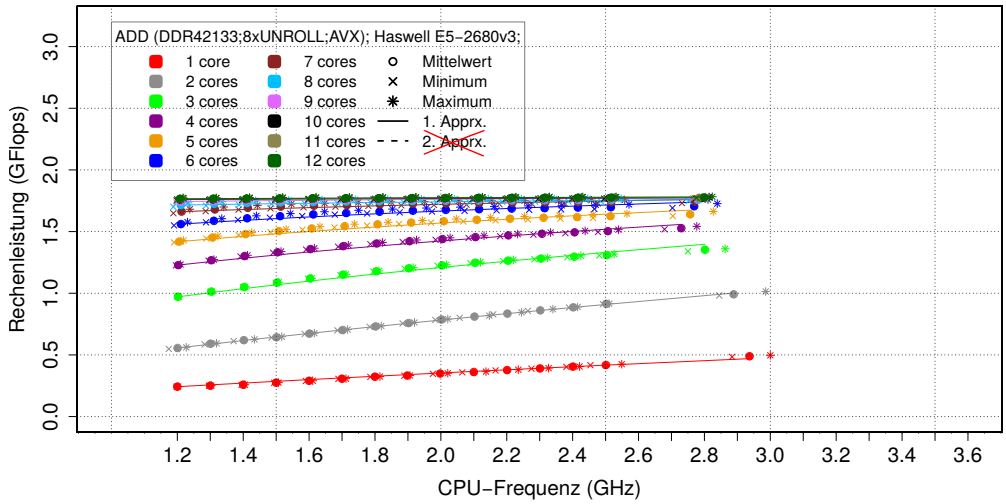

Abb. 4.19.: Rechenleistung auf *Haswell E5-2680v3* im Falle des Haltens der Daten im Hauptspeicher

Die Approximation der Rechenleistung über die CPU-Frequenz-Definitionsbereiche F_1 und F_2 werden in Abb. 4.18 über zwei getrennte Kurven angezeigt.
Die Koeffizienten der Approximation konnten nicht mit einer ausreichenden Genauigkeit und in einer gewünschten Form über die Thread-Anzahl p interpoliert werden (siehe Abschnitt II.II des Anhangs E auf Seite 165). Wie die Rechenleistung im Falle des Haltens der Daten im Hauptspeicher beschrieben werden kann, wird unter anderem im Kapitel 5 gezeigt.

4.5.4.2. Elektrische Leistung im Hauptspeicher

Es zeigte sich, dass die Rechenleistung schnell einen Sättigungsbereich erreicht. Dies gilt nicht für die elektrische Leistung. Wie in sonstigen Fällen steigt die elektrische Leistung überlinear mit der CPU-Frequenz-Erhöhung, dabei gibt es jedoch einige Unterschiede. Die Speichermodule müssen die Daten einer Kernel-Operation lesen und schreiben. Diese Daten tauschen die Prozessoren mit den Speichermodulen über die IMCs. An jedem der vier Speicherkanäle eines Prozessors ist ein Speichermodul angeschlossen: Dual Rank DDR3-SDRAM 4 GB Modul für den Prozessor *Ivy Bridge* und ein wesentliches größeres Dual Rank DDR4-SDRAM 16 GB Modul für den Prozessor *Haswell*. Deshalb ist der statische Anteil der elektrischen Leistung des Prozessors *Haswell* größer als der statische Anteil der elektrischen Leistung des Prozessors *Ivy Bridge*. Abb. 4.20 und Abb. 4.21 zeigen die elektrische Leistung der

beiden Prozessoren und derer Speichermodule abhängig von der CPU-Frequenz an. Wie in vorherigen Fällen, konnte die elektrische Leistung mit einer hohen Genauigkeit approximiert werden.

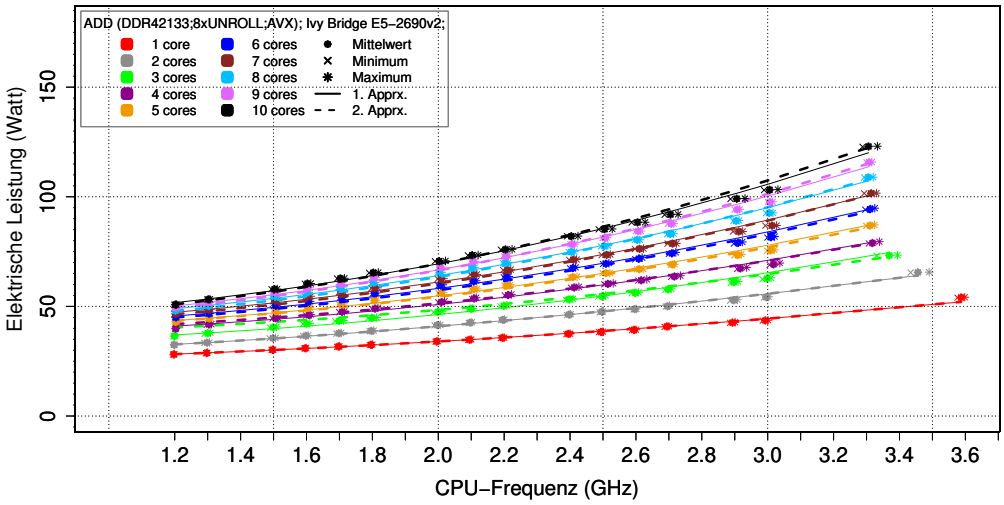

Abb. 4.20.: Elektrische Leistung des Prozessors *Ivy Bridge E5-2690v2* im Falle des Haltens der Daten im Hauptspeicher

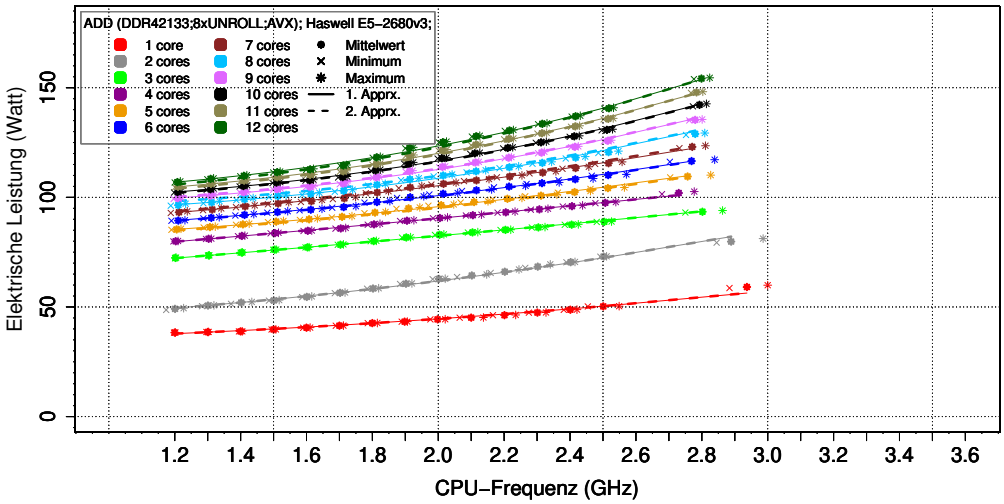

Abb. 4.21.: Elektrische Leistung des Prozessors *Haswell E5-2680v3* im Falle des Haltens der Daten im Hauptspeicher. Der statische Anteil der elektrischen Leistung des Prozessors *Haswell* ist wegen des größeren Speichers deutlich größer als der statische Anteil der elektrischen Leistung des Prozessors *Ivy Bridge*.

Es ist zu erkennen, dass die elektrische Leistung des Prozessors *Haswell* für $p < 3$ ein besonderes Verhalten aufweist. Wie im Falle des L3-Caches, unterscheiden sich die „Uncore"- und die CPU-Frequenzen für $p > 2$ (siehe Abbildung Abb. E.2 in Anhang E auf Seite 170). Durch die Anwendung der QR-Methode auf die Koeffizienten ergibt sich Gl. (4.28). Die Gleichung beschreibt die elektrische Leistung der beiden Prozessoren. Die Interpolationskoeffizienten sind in Anhang E auf Seite 173 aufgeschrieben.

Ivy Bridge (E5-2690v2):

$$
\begin{aligned}
P_{II}(f,p)_{ADD,RAM} &= (21.58571 + 2.282459 \times p) + (1.56876 + 1.16752 \times p) \times f^{1.81}; \\
&\quad p \in (1,2); \\
P_{II}(f,p)_{ADD,RAM} &= (34.13481 + 0.72235 \times p) + (1.21357 + 0.52912 \times p) \times f^{2.11}; \\
&\quad p \in (3,4,...,10); \\
&\quad \epsilon_{rel} \leq 0.093;
\end{aligned}
$$

Haswell (E5-2680v3):

$$
\begin{aligned}
P_{II}(f,p)_{ADD,RAM} &= (25.46588 + 8.26171 \times p) + (0.49423 + 2.39527 \times p) \times f^{1.91}; \\
&\quad p \in (1,2); \\
P_{II}(f,p)_{ADD,RAM} &= (65.52072 + 2.02131 \times p) + (2.02131 + 0.32525 \times p) \times f^{2.11}; \\
&\quad p \in (3,4,...,12); \\
&\quad \epsilon_{rel} \leq 0.077;
\end{aligned}
\tag{4.28}
$$

Interessant sind folgende Unterschiede:

- Die Speichermodule der beiden Prozessoren werden nicht voll belastet, wenn wenige Threads aktiv sind: die Koeffizienten des statischen Anteils für $p \in (1,2)$ sind deutliche kleiner als für $p > 2$.
- Die kleineren Speichermodule des *Ivy Bridge* Prozessors verbrauchen deutlich weniger Strom als die Speichermodule des *Haswell* Prozessors.
- *Ivy Bridge*: Außer des statischen Anteils sind die Koeffizienten ähnlich zu den Koeffizienten im Falle des Haltens der Daten im L3-Cache.
- *Haswell*: Die Koeffizienten unterscheiden sich stark von den Koeffizienten im Falle des Haltens der Daten im L3-Cache.

4.5.4.3. Energiekosten im Hauptspeicher

Die Energiekosten im Hauptspeicher können nicht mit einer einzigen Formel ausgedrückt werden, weil die Rechenleistungs-Approximation in einer komplizierten Form vorliegt (siehe Abb. E.4). Die Energiekosten werden für die Messpunkte mit Gl. (4.13) berechnet. Abb. 4.22 stellt die Ergebnisse dar. Wenn die berechneten Werte der Energiekosten für die entsprechende Anzahl der aktiven Prozessorkerne verbunden wären, würden sich einige Kurven überschneiden. Diese sind zum Beispiel die Kurven der Messwerte für drei Prozessorkerne (in grün).

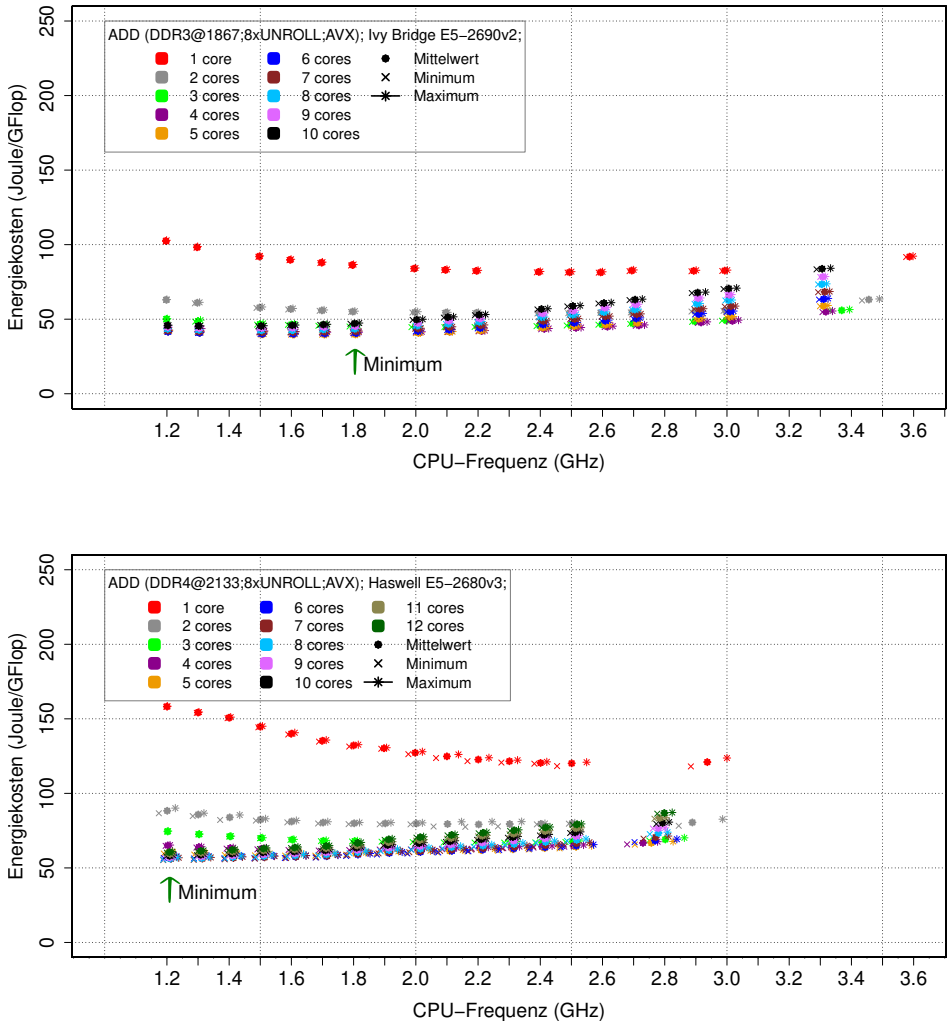

Abb. 4.22.: Energiekosten des Prozessors *Ivy Bridge E5-2690v2* (oben) und *Haswell E5-2680v3* (unten) im Falle des Haltens der Daten im Hauptspeicher

Es ist deutlich zu erkennen, dass sich die Energiekosten schneller mit der Zunahme

der CPU-Frequenz erhöhen. Der Grund dafür ist, dass die elektrische Leistung mit der CPU-Frequenz viel schneller als die Rechenleistung wächst. Die Diagramme in Abb. 4.23 zeigen die Abhängigkeit zwischen der Rechenleistung (X-Achse) und den Energiekosten (Y-Achse). Sowohl der Prozessor *Ivy Bridge* als auch der Prozessor *Haswell* sind in der Rechenleistung stark limitiert. Mit der Erhöhung der CPU-Frequenz steigen die Energiekosten aber die Rechenleistung bleibt fast unverändert.

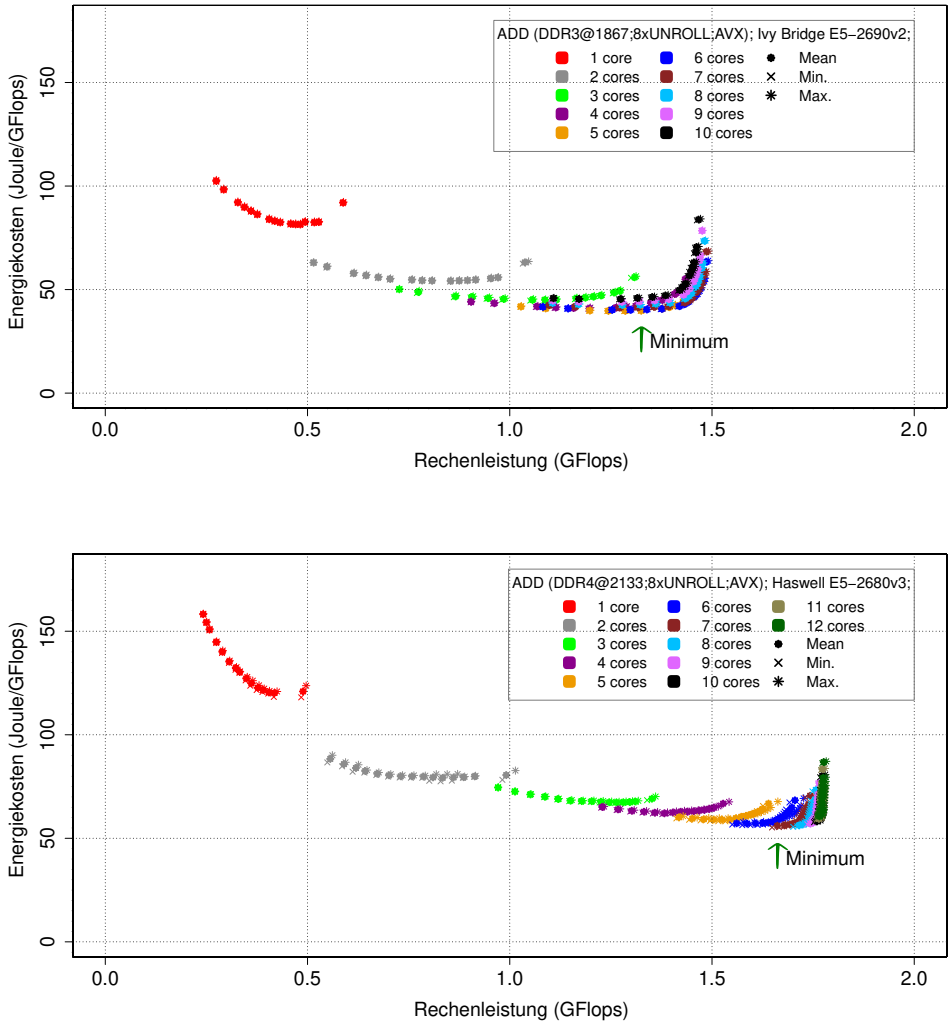

Abb. 4.23.: Energiekosten der Prozessoren *Ivy Bridge E5-2690v2* (oben) und *Haswell E5-2680v3* (unten) abhängig von der Rechenleistung im Falle des Haltens der Daten im Hauptspeicher

Zusätzlich sind die optimalen CPU-Frequenzen mit den minimalen Energiekosten der Prozessoren dargestellt. Im Vergleich zu den Fällen des Haltens der Daten im Cache ist das Minimum stark nach links geschoben.

5. Modellierung der Kernel-Ausführung

Die Rechenleistung der Kernel-Operation *Add* hängt stark davon ab, wie weit der Weg der Daten a, b und c zwischen den AVX-Registern und der letzten Hierarchiestufe des physikalisch gelesenen und geschriebenen Speichers wie L1-, L2-, L3-Caches und Hauptspeicher ist. Im ersten Teil dieses Kapitels wird dies im Anhang der Ausführungsmodellierung der Kernel-Operation *Add* mit „Execution-Cache-Memory Modell“ (ECM) nach [32] und [35] für den Prozessor *Haswell* gezeigt. Während des Modellaufbaus werden die relevanten Einheiten der Prozessorarchitektur und darüber hinaus die Rolle der verschiedenen Frequenzdomänen für die Rechenleistung eines Mehrkernprozessors erläutert. Ich zeige, welche Einschränkungen das existierte ECM-Modell aufweist und versuche teilweise diese Einschränkungen im zweiten Teil mit einer Erweiterung des ECM-Modells zu überwinden. Das neue Modell werde ich als „Daten-Transfer Modell“ (DTM) bezeichnen. Der Name deutet darauf hin, dass die Effizienz des Datentransfers zwischen den verschiedenen Hierarchiestufen des Speichers den bedeutendsten Faktor für die Effizienz der High Performance Computing Anwendungen der Ingenieurwissenschaften darstellt. Für die Validierung der beiden Modelle werden die Modell-Resultate mit den realen Messungen verglichen.

5.1. Execution-Cache-Memory Modell (ECM)

5.1.1. Grundlagen des ECM-Modells

Bei der ECM-Modellierung nimmt man an, dass der Datentransfer während der Ausführung einer Kernel-Operation, wie z.B. *Add* auf einem Prozessorkern, durch alle Speicher-Hierarchiestufen sequenziell durchläuft. Somit rechnet sich die gesamte Ausführungszeit wie die Summe der einzelnen Datenübertragungen zwischen den Hierarchiestufen. Erst die Ausführung der arithmetischen Operationen in der Recheneinheit (ALU) kann zum Datentransfer zwischen den beliebigen Hierarchiestufen überlappend erfolgen. Gleichung Gl. (5.1) beschreibt nach [33] die Ausführungszeit

einer Kernel-Operation auf einem Prozessorkern mit drei Cache-Hierarchiestufen im Falle des Haltens der Daten im Hauptspeicher. Für die Beschreibung der Rechenleistung wird angenommen, dass die Prozessorarchitektur eine lineare Skalierung hinsichtlich der Kernelanzahl aufweist. Eine Ausnahme von dieser Regel ist die Speicherbandbreite, die in vielen Fällen ein Flaschenhals ist, was im Abschnitt 5.1.2.4 ausführlich betrachtet wird.

$$
\begin{aligned}
T^{ECM} &= \max(T^{OL}, T^{nOL} + T_{L1\leftrightarrow L2} + T_{L2\leftrightarrow L3} + T_{L3\leftrightarrow MEM}); \\
\\
T^{ECM} &\text{ - } \textit{ECM-Ausführungszeit einer Kernel-Operation mit einem Thread;} \\
T^{OL} &\text{ - } \textit{Ausführungszeit in ALU mit der Überlappung zum Datentransfer;} \\
T^{nOL} &\text{ - } \textit{Ausführungszeit in ALU ohne Überlappung zum Datentransfer;} \\
T_{L1\leftrightarrow L2} &\text{ - } \textit{Übertragungszeit der Daten zwischen L1 und L2;} \\
T_{L2\leftrightarrow L3} &\text{ - } \textit{Übertragungszeit der Daten zwischen L2 und L3;} \\
T_{L2\leftrightarrow L3} &\text{ - } \textit{Übertragungszeit der Daten zwischen L3 und MEM;}
\end{aligned}
\tag{5.1}
$$

Um die einzelnen Terme der Gleichung Gl. (5.1) für eine der Kernel-Operation zu bestimmen, werden die für die Rechenleistung relevanten Merkmale des Prozessors in das Modell übernommen.

5.1.2. Modellierung der Ausführungszeit mit ECM

Ein guter Ausgangspunkt zur Modellierung ist eine Formulierung der Kernel-Operation in der hardwarenahen Programmiersprache *Assembler*. Die gesammelte Information über die Prozessorarchitektur, die Compilerausgabe, die Erfahrung und eine Reihe der begründeten Annahmen unterstützen einen Modellierer bei der Formulierung eines Quellcodes.
Die Assemblerbefehle werden vom Compiler in die Maschinenbefehle übersetzt, die während der Ausführung im Prozessorkern in mehrere Mikrobefehle aufgeteilt werden. Diese werden in einer sogenannten *Out-of-Order* Einheit gesammelt und mit einem Scheduler an die Ausführungseinheit der Prozessorkern-Pipeline einzeln über die Ports weitergereicht.

5.1.2.1. Out-of-Order Einheit und Pipeline

Die komplette Ausführung eines einzigen Mikrobefehls umfasst mehrere Takte. Ein Mikrobefehl wird in mehrere Suboperationen oder weitere Mikrobefehle aufgeteilt

und schrittweise ausgeführt. Falls der auszuführende Code aus sich wiederholenden und teilweise unabhängigen Mikrobefehlen besteht, wie es oft in Schleifen der Fall ist, können die entsprechenden Suboperationen in der Pipeline überlappend ausgeführt werden. Dies ermöglicht es die Latenzzeiten der einzelnen Mikrobefehle zu verstecken. Zusätzlich zur Latenzzeit werden die Mikrobefehle durch den Durchsatz charakterisiert, dessen Dauer seinerseits zu der längsten elementaren Suboperation umgekehrt proportional ist. Ohne die Allgemeinheit der weiteren Aussagen zu limitieren, werden die Latenzzeiten der einzelnen Mikrobefehle nicht berücksichtigt, da die Anzahl der sich wiederholenden Mikrooperatione für die Füllung der Pipeline ausreichend ist.
Die „Out-out-Order“ Einheit ist das Front-End einer Pipeline[1] im *Haswell*-Prozessorkern, das die Mikrobefehle über die sogenannten „Ports“ in der Pipeline verteilt. Insgesamt können gleichzeitig bis zu 198 Mikrobefehle auf eine Portzuweisung warten. Ein *Haswell*-Kern verfügt über 8 Ports: 4 für die Rechen- und 4 für die Speicheroperationen. Alle Ports können, mit einigen Ausnahmen, gleichzeitig genutzt werden. In Abb. 5.1 ist ein Teil des Prozessorkerns mit den Ports und den entsprechenden Mikrobefehlen schematisch dargestellt.

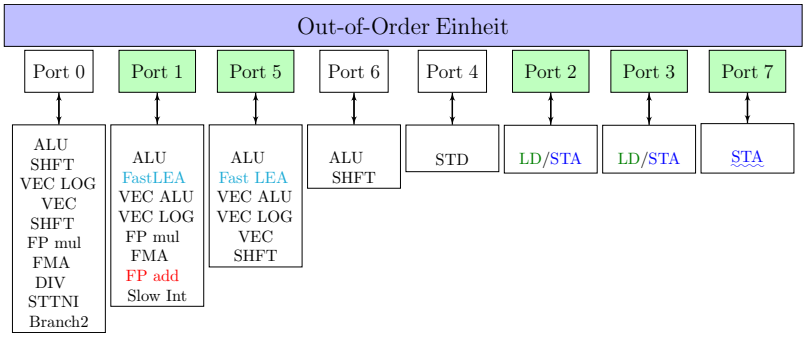

Abb. 5.1.: Prinzipieller Aufbau einer Pipeline im *Haswell*-Kern [13].

In das ECM-Modell werden nur die Prozessoreinheiten übernommen, die von Bedeutung für die Ausführung der Kernel-Operation *Add* sein könnten:

- *Port 1:* Dieser Port übernimmt die Aufträge für die Vektoroperation Add auf den AVX-Registern. Der Mikrobefehl lautet „*FP add*“. Die Latenzzeit von „*FP add*“ beträgt drei Takte und der Durchsatz ist nach [14] eine Operation pro Takt. Somit gleicht die Ausführungszeit des Mikrobefehls im Modell einem Takt der CPU-Frequenz.

- *Port 2:* Dieser Port übernimmt die Aufträge für das Laden der Daten aus dem

[1] In der englischsprachigen Literatur wird von „pipeline front end“ gesprochen.

Speicher in ein AVX-Register oder die Speicherung des AVX-Registerinhaltes in den Speicher. Es gibt keine Beschränkungen für die Art der Datenadressierung. Die Mikrobefehle lauten „*LD*“ und „*ST*“. Nach [15] hängen der Durchsatz und die Latenzzeit überwiegend davon ab, in welcher Speicher-Hierarchiestufe die Daten bearbeitet werden:

  - *L1:* Die Latenzzeit beträgt vier Takte und der Durchsatz beträgt ein Ladebefehl pro Taktzyklus.
  - *L2:* Die Latenzzeit beträgt 11 Takte. Keine Angaben für den Durchsatz sind mir bekannt.
  - *L3:* Die Latenzzeit beträgt 34 Takte. Keine Angaben für den Durchsatz sind mir bekannt.

- *Port 3:* Dieser Port ist mit Port 2 identisch.
- *Port 7:* Dieser Port übernimmt die Aufträge für die Speicherung eines AVX-Registerinhaltes. Es ist nur die direkte Adressierungsart möglich.
- *Port 5:* Dieser Port übernimmt die Aufträge für die Umwandlung einer indirekten Adresse in die entsprechende direkte Adresse. Der Mikrobefehl lautet „*Fast LEA*“. Der Unterschied zwischen den indirekten und direkten Adressierungsarten wird im weiteren Verlauf erklärt. Die Latenzzeit des Mikrobefehls beträgt nach [16] drei Takte. Es wurden keine genaueren Angaben zum Durchsatz gefunden. Die Ausführungszeit des Mikrobefehls wird wie in den vorherigen Fällen für einen Takt angenommen.

Mehr Details über die *Haswell*-Pipeline und ihre Funktionalität kann der interessierte Leser in [17] finden.

5.1.2.2. Implementierung des Kernels im Assembler

In diesem Abschnitt wird eine einfache Implementierung der Kernel-Operation *Add* in der Assemblersprache demonstriert. Um eine Vektor-Operation durchzuführen, müssen die Fließkommazahlen in den AVX-Registern gehalten werden. Das ermöglicht eine gleichzeitige Ausführung von vier identischen arithmetischen Fließkommazahlen-Operationen in doppelter Genauigkeit.
Eine schematische Darstellung des Mikrobefehls „*FP add*“ ist in Abb. 5.2 dargestellt.

Ein AVX-Register, zum Beispiel `%ymm0`, enthält vier Fließkommazahlen und hat die Länge von 32 Bytes.

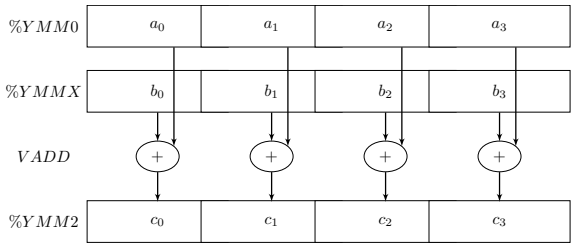

Abb. 5.2.: Schematische Darstellung einer *Single-Instruction-Multiple-Data* (SIMD) Addition

In Quelltext 5.1 ist eine Implementierung der Kernel-Operation *Add* in der Assemblersprache ohne Initialisierung, Schleifeninkrement und Schleifenbedingung gezeigt. Der erste Befehl `vmovapd` wird vom Prozessor als eine Anweisung zum Laden der Daten aus dem Speicher in ein AVX-Register interpretiert, wobei die Anfangsadresse der zu ladenden Daten aus dem Ausdruck `(%r13,%rax,8)` entnommen werden muss. Diese Art der Adressierung bezeichnet man als indirekt. Vor dem Laden der Daten wird die indirekte Adresse in eine absolute umgewandelt. Die Formel dazu ist `%r13+8×%rax`. Der aktuelle Schleifenindex `i` und die Basisadresse des Feldes $a[]$ müssen während der Befehlsausführung in den allgemeinen Registern (GPR) `%rax` und `%r13` gehalten werden. Der Skalierungsfaktor 8 ist die Größe eines Fließkommazahlen-Elementes des Feldes $a[]$ in Bytes. Somit lädt der Prozessor vier nacheinander folgende Elemente $a[i], a[i+1], a[i+2], a[i+3]$ in das AVX-Register `%ymm0`.
Der zweite Befehl `vaddpd` führt paarweise die Addition von acht Fließkommazahlen aus dem AVX-Register `%ymm0` und dem Feld $b[]$ aus. Die zweiten Summanden der Addition liegen im Speicher mit der Adresse `(%r12,%rax,8)`. Das GPR-Register `%r12` enthält dabei die Adresse des ersten Elements des Feldes $b[]$. Somit besteht der Befehl aus zwei nacheinander folgenden Mikrobefehlen. Der erste Mikrobefehl wandelt die indirekte Adresse und lädt die Daten in ein Register `%ymmX` und der zweite Mikrobefehl addiert paarweise den Inhalt von den Registern `%ymm0` und `%ymmX`, wie es in Abbildung 5.2 dargestellt ist.
Der dritte Befehl `vmovapd` schreibt den Inhalt des AVX-Registers `%ymm2` in den Speicher mit der Adresse `%r14+8×%rax`, wobei das GPR-Register `%r14` die Basisadresse des Feldes $c[]$ enthält.

Quelltext 5.1: Anweisungs-Block der Kernel-Operation *Add* in der Assemblersprache mit einer indirekten Adressierung sowohl für das Laden als auch für die Speicherung der Daten.

```
vmovapd    (%r13,%rax,8), %ymm0
vaddpd     (%r12,%rax,8), %ymm0, %ymm2
vmovapd    %ymm2, (%rb14,%rax,8)
```

5.1.2.3. Pipeline im Falle des Haltens der Daten im L1-Cache

Der Pipeline-Scheduler versucht in jedem Takt die Mikrobefehle zu den entsprechenden Ports zu verteilen, um die Pipeline möglichst voll zu besetzen. Tab. 5.1 zeigt einen Pipeline-Verlauf für acht nacheinander folgende Takte. Das Kürzel „$LD(a_i)$“ bezeichnet einen Mikrobefehl zum Laden von vier nacheinander folgenden Elementen $a_i,\ldots,a_{i+3}$ aus dem L1-Cache in ein AVX-Register. Das Kürzel „$ADD(c_{i-(i+3)})$“ bezeichnet einen Mikrobefehl zur Addition von $a_i,\ldots,a_{i+3}$ und $b_i,\ldots,b_{i+3}$, die vorher in zwei AVX-Registern geladen wurden. Das Resultat der Operation wird im dritten AVX-Register gehalten, bis es mit dem Mikrobefehl „$ST(c_{i-(i+3)})$“ im L1-Cache gespeichert wird. Die Adressierungsart ist indirekt, deshalb kann der Port 7 nicht verwendet werden.
Somit ergibt sich, dass zwei Vektoroperationen „ADD“, vier Ladebefehle „LD“ und zwei Speicherbefehle „ST“ in drei Takten ausgeführt werden. In einem Takt werden durchschnittlich $\frac{8}{3} = 2.6(6)$ Gleitkommazahl-Operationen durchgeführt (Vergleiche mit dem Skalierungsfaktor des *Haswell* Prozessors von 2.56 [FLOP] in Gl. (4.10).
Im ECM-Modell wird zusätzlich die Metrik $cycle/CL$ verwendet. Die Abkürzung CL ist eine Bezeichnung für eine Cache-Zeile[2], die im Prozessor *Haswell* 64 B lang ist. Die Länge der zwei AVX-Register entspricht der Länge einer CL. Somit ergibt sich die sogenannte Ausführungszeit (engl. „execution time“) von 3 $cycle/CL$.
Der modellierte Verlauf der Ausführung der Kernel-Operation *Add* enthält die Lücken, die sogenannten „Bubbles“. Diese können nach dem ECM-Modell des Prozessors *Haswell* eliminiert werden. Der optimierte Quellcode und dessen entsprechender Verlauf in der Pipeline ist im Anhang F auf Seite 175 zu finden. Hierbei handelt es sich um ein theoretisches Modell. Eine entsprechende Implementierung zeigte auf dem Prozessor *Haswell E5-2680v3* keine Vorteile gegenüber der hier betrachteten Implementierung mit den Lücken in der Pipeline.

[2]In der englischsprachigen Literatur wird von „cacheline“ gesprochen.

Tab. 5.1.: Verlauf der Kernel-Operation *Add* im Falle des Haltens der Daten im L1-Cache in einer *Haswell*-Pipeline in Takten der CPU-Frequenz.

	Verlauf von Mikrobefehlen in der Pipeline über die Zeit						
	1	2	3	4	5	6	7
Port 2	$LD(a_{0-3})$	$LD(a_{4-7})$	$ST(c_{0-3})$	$ST(c_{4-7})$	$LD(a_{12-15})$	$ST(c_{8-11})$	$ST(c_{12-15})$
Port 3	$LD(b_{0-3})$	$LD(b_{4-7})$	$LD(a_{8-11})$	$LD(b_{8-11})$	$LD(b_{12-15})$	$LD(a_{16-19})$	$LD(b_{16-19})$
Port 1		$ADD(c_{0-3})$	$ADD(c_{4-7})$		$ADD(c_{8-11})$	$ADD(c_{12-15})$	
Port 7							
Port 5							

5.1.2.4. Modellierung der Hierarchiestufen des Speichers

Die im Abschnitt 5.1.1 betrachteten Modelle beschreiben das Verhalten eines Prozessors im Falle des Haltens der Daten im L1-Cache. Wenn die Daten in einer höheren Hierarchiestufe des Speichers gehalten werden, wird die theoretische Bandbreite zwischen den Stufen für die Modellierung der Rechenleistung entscheidend. Im Unterschied zur Übertragung der Daten zwischen den Registern und dem L1-Cache mit einer Granularität von einer halben Cache-Zeile (CL) erfolgt der Datenaustausch zwischen den L1- und L2-Caches in ganzer CL. Für die L2- und L3-Caches modelliert man weiterhin die Rechenleistung für einen Prozessorkern pro Takt der CPU-Frequenz.
Die Übertragungszeit zwischen dem SDRAM-Speicher und dem L3-Cache wird nicht für einzelne Prozessorkerne, sondern für eine Speicherdomain modelliert. Im betrachteten Fall ist der Prozessor mit einer Domain konfiguriert. Das bedeutet, dass alle Prozessorkerne einen uniformen Weg zu jeder Cache-Zeile im Hauptspeicher haben. Die Kerne kommunizieren mit dem SDRAM-Speicher über zwei integrierte Speichercontroller (IMC). Ein IMC verfügt über zwei Speicherkanäle, die je mit einem SDRAM-Modul verbunden sind (andere Konfigurationen sind möglich). Die theoretische Hauptspeicher-Bandbreite wird durch das Produkt von drei Parametern berechnet:

$$
\begin{aligned}
B_{MEM}[B/s] \quad &= N_{mem\_channel} \times W_{bus}[\mathrm{B}] \times f_{clock\_mem}[MHz]; \\
\\
N_{mem\_channel} \quad &= 4 \ \textit{(Anzahl der Speicherkanäle);} \\
W_{bus} \quad &= 8 \ \mathrm{B} \ \textit{(Bus-Breite eines Speicherkanals);} \\
f_{clock\_mem} \quad &= 2133 \ \mathrm{MHz} \ \textit{(Speicher-Taktfrequenz);}
\end{aligned}
\tag{5.2}
$$

Abb. 5.3 zeigt eine schematische Darstellung der Übertragungswege zwischen den Speicher-Hierarchiestufen. Zusätzlich sind in der Mitte der Abbildung die Frequenzen angegeben, mit denen die Komponenten im Prozessor *Haswell E5-2680v3* getaktet sind (links vom Takt-Symbol).
Der bi-direktionale Bus vom L2- zum L1-Cache hat die Breite von 64 Bytes. Der Bus zur Übertragung der Daten vom L3- zum L2-Cache hat die Busbreite von 32 Bytes. Eine Cache-Zeile braucht somit einen Takt für den Weg zwischen den L1- und L2-Caches und zwei Takte zwischen den L2- und L3-Caches, da die Latenzzeiten der entsprechenden Mikrobefehle in der Pipeline versteckt werden.

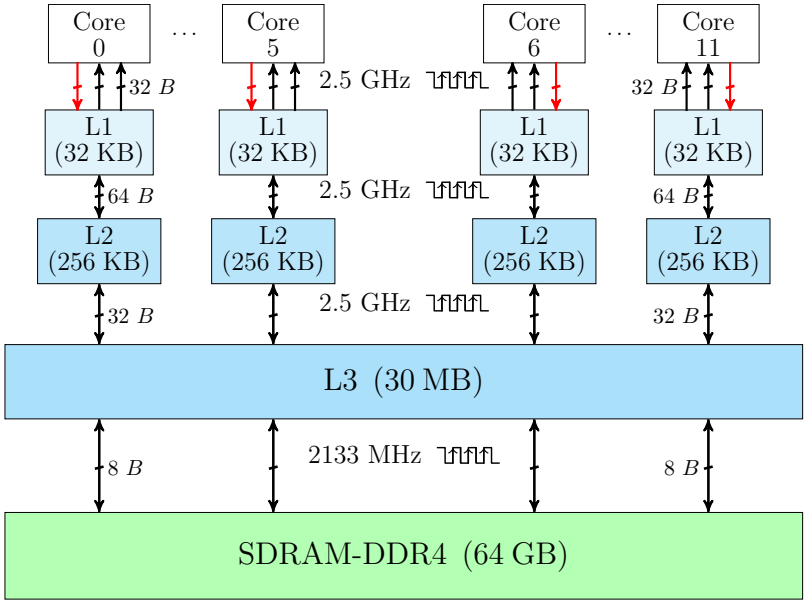

Abb. 5.3.: Die Speicher-Hierarchiestufen in einem *ECM*-Modell für den Prozessor *Haswell E5-2680v3*. Die Breite der Interfaces zwischen den Speicher-Hierarchiestufen, die Größe des Caches und die Frequenz der Komponenten (links vom Takt-Symbol) sind in der Skizze angegeben.

Eine andere wichtige Eigenschaft der Speicherorganisation eines Intel Prozessors ist hervorzuheben. Wenn die zu schreibende CL nicht im L1-Cache oder keinem passenden MESI-Status markiert ist, wird der Prozessor diese mit einem zulässigen MESI-Status aus einer höheren Hierarchiestufe des Speichers holen[3]. Diese Eigenschaft bezeichnet man als „write-allocate transfer“.
Die Durchsatzanalyse der unterschiedlichen Stream-Operationen mit ECM-Model in [32] bestätigt, dass die Datenübertragung zwischen den verschiedenen Hierarchiestufen des Speichers nicht überlappend erfolgt, wenn die CPU-Frequenz maximal ist (siehe Abschnitt 5.1.1).

[3]Für eine detaillierte Beschreibung der Speicheroperation *Store* siehe [18].

5.1.2.5. Pipeline im Falle des Haltens der Daten in L2- und L3-Caches

Die Tabellen Tab. 5.2 und Tab. 5.3 zeigen den Verlauf des Kernels *Add* in einem *ECM*-Modell für einen *Haswell*-Kern. Der Mikrobefehl „RCL“ dient zur Übertragung einer CL vom L2- zum L1-Cache. Mit dem Befehl „WCL“ wird eine CL in die entgegengesetzte Richtung übertragen, wenn es im L1-Cache für die neuen Daten keinen Platz mehr gibt.

Tab. 5.2.: Verlauf der Kernel-Operation *Add* im Falle des Haltens der Daten im L2-Cache in einer *Haswell*-Pipeline in Takten der CPU-Frequenz (Teil I).

	Verlauf von Mikrobefehlen in der Pipeline über die Zeit						
	1	2	3	4	5	6	7
Port 2			$LD(a_{0-3})$	$LD(a_{4-7})$			$ST(c_{0-3})$
Port 3			$LD(b_{0-3})$	$LD(b_{4-7})$			$LD(a_{8-11})$
Port 1				$ADD(c_{0-3})$			$ADD(c_{4-7})$
L2->L1	$RCL(a_{0-7})$	$RCL(b_{0-7})$			$RCL(c_{0-7})$	$RCL(a_{8-15})$	
L1->L2							

Tab. 5.3.: Fortsetzung des Verlaufs der Kernel-Operation *Add* im Falle des Haltens der Daten im L2-Cache in einer *Haswell*-Pipeline in Takten der CPU-Frequenz (Teil II).

	Verlauf von Mikrobefehlen in der Pipeline über die Zeit						
	8	9	10	11	12	13	...
Port 2		$ST(c_{4-7})$		$LD(a_{12-15})$			...
Port 3		$LD(b_{8-11})$		$LD(b_{12-15})$			...
Port 1				$ADD(c_{8-11})$			...
L2->L1	$RCL(b_{8-15})$				$RCL(c_{8-15})$	$RCL(a_{16-20})$	...
L1->L2			$WCL(c_{0-7})$				...

In den ersten drei Takten des Verlaufs in Tab. 5.2 wird die Pipeline des Prozessors aufgefüllt. Alle sieben Takte danach schreibt der Prozessor eine Cache-Zeile mit den Elementen des Feldes c im L2-Cache aus. Die Übertragungszeit im Falle des Haltens der Daten im L2-Cache beträgt 7 $cycle/CL$. Somit werden in einem Takt durchschnittlich $\frac{8}{7} = 1.14$ Gleitkommazahl-Operationen durchgeführt. Der experimentell bestimmte Skalierungsfaktor des *Haswell* Prozessors von 0.8 [FLOP] (siehe Gl. (4.17)) ist deutlich kleiner. Der Grund dafür ist mir unbekannt.
Im Falle des Haltens der Daten im L3-Cache verdoppelt sich die Übertragungszeit,

weil die Bandbreite des Busses zwischen dem L2- und dem L3-Cache halb so groß wie zwischen dem L1- und dem L2-Cache ist (siehe Abb. 5.3).

5.1.2.6. Pipeline im Falle des Haltens der Daten im Hauptspeicher

Für die Modellierung der Übertragungszeit im Falle des Haltens der Daten im Hauptspeicher muss berücksichtigt werden, dass sich die Speicher- und Prozessor-Frequenzen unterscheiden. Für eine Einheitlichkeit müssen die Takte umgerechnet werden. Die theoretische Bandbreite von 63.56 GB/s entspricht 29.8023 B/cycle im Takt der Speicherfrequenz oder 25.4273 B/cycle im Takt der CPU-Frequenz. Als CPU-Frequenz gilt die Basis-Frequenz des Prozessors 2.5 GHz. Hierbei sei erwähnt, dass die Bandbreite zwischen den aktiven Prozessorkernen geteilt wird. In einem ECM-Modell wird angenommen, dass die Aufteilung einem linearen Gesetz folgt.
Somit wird modelliert, wie hoch die Ausführungszeit einer Kernel-Operation ist. Die Zeit gilt als eine gute Abschätzung der maximalen Ausführungszeit abhängig vom Prozessortyp, von der Implementierung und von der Größe der Daten.

5.1.3. Vergleich der theoretischen und experimentellen Ergebnisse

5.1.3.1. Unterschiede im Quellcode

Der Intel Compiler[4] erzeugt aus einer Formulierung der Kernel-Operation in der C-Sprache eine vektorisierte Implementierung in der Assemblersprache. Der einzige Unterschied zum Quelltext 5.1 besteht darin, dass die Schleife viermal vom Compiler entrollt wird. Das Schleifen-Aufrollen reduziert die Anzahl der Indexoperationen, füllt die *Out-out-Order* Einheit mit mehr unabhängigen Mikrobefehlen aus und lässt während der Ausführung noch mehr Register benutzen.

5.1.3.2. Evaluierung der Ergebnisse

Tab. 5.4 zeigt sowohl die modellierten als auch die gemessenen Zeiten für den Kernel *Add* in cycle/CL auf dem Prozessor *Haswell*. Die Transferzeiten im Falle des Haltens der Daten im Hauptspeicher sind für die Ausführung mit zwölf Threads und 2.5 GHz

[4]Version 17.0.4, Compileroptionen: -O3 -openmp -restrict;

berechnet und in der Spalte „L123+RAM“ aufgelistet. In der Zeile „Insgesamt“ sind die Endergebnisse gezeigt. Die Zeile „Per Stufe“ enthält die intermediären Zeiten der Datenübertragung, derer Summe dem Endergebnis entspricht, da die Übertragung zwischen den Speicher-Hierarchiestufen nicht überlappend läuft (siehe den Abschnitt 5.1.1).
Das *ECM*-Modell liefert eine gute Abschätzung für die Transfer- und Ausführungszeiten. Allerdings liefert das Modell keine Information über die Ausführungszeit für die verschiedene Anzahl der Threads und der CPU-Frequenz im Falle des Haltens der Daten im Hauptspeicher. Zusätzlich wird angenommen, dass die Rechenleistung im Falle des Caches mit der Anzahl der Threads und der CPU-Frequenz linear wächst. Es wird vorausgesetzt, dass es keinen Flaschenhals während der Übertragung gibt. Wie gezeigt wurde, stimmt das im Falle des L3-Caches nicht exakt.
Im Folgenden wird ein Modell präsentiert, das die Ausführungszeit auch für die verschiedene Anzahl der Threads und für einen kompletten Bereich der CPU-Frequenzen modelliert.

Tab. 5.4.: Die Tabelle gibt sowohl die modellierte als auch die gemessene Ausführungszeit der Kernel-Operation *Add* auf dem Prozessor *Haswell E5-2680v3* in cycle/CL an.

Kernel Add	ECM-Modell [cycle/CL]				Messung [cycle/CL]			
	L1	L1+L2	(L12)+L3	(L123)+RAM	L1	L1+L2	(L12)+L3	(L123)+RAM
Per Stufe	3	3+4	7+8	15+120.81	3.13	3.13 + 6.77	9.9 + 9.13	19.03+115.94
Insgesamt	3	7	15	135.81	3.13	9.9	19.03	134.97

Interessant ist, dass die gemessene Ausführungszeit im Falle des Haltens der Daten im L3-Cache („(L12)+L3“) deutlich höher als die modellierte Zeit (19.3 und 15) ist. Trotzdem stimmen die Resultate im Falle „(L123)+RAM“ praktisch völlig überein. Eine Erklärung über das unterschiedliche Verhalten des L3-Caches liegt in der Funktionsweise des Caches, die im folgenden Abschnitt 5.2 Anhang der *Haswell*-Prozessorarchitektur erklärt wird.

5.1.4. Bandbreite im Falle des Haltens der Daten im Cache

Zusätzlich zu den Metriken [FLOP] und [cycle/CL] wird die Metrik „Bandbreite“ [B/s] für die Messung einer Kernel-Operation verwendet. Diese zeigt eine durchschnittliche Transferrate der Daten während der Ausführung. Man unterscheidet

zwischen einer tatsächlichen und einer effektiven Bandbreite. Für eine Gleitkommazahl-Operation *Plus* müssen zum Beispiel zwei Fließkommazahlen mit doppelter Genauigkeit gelesen (N_{read}) und eine Fließkommazahl geschrieben (N_{write}) werden. Somit ergibt sich, dass das Verhältnis der Gleitkommazahl-Operationen zu der tatsächlichen Bandbreite im Falle des Haltens der Daten im L1-Cache 1 zu 24 ist und für alle anderen Fälle 1 zu 32. Begründet ist es dadurch, dass jede Cache-Zeile vor dem Schreiben zunächst gelesen werden muss (Siehe Abschnitt 5.1.2.4 und [18]). Im Gegensatz dazu ist eine effektive Bandbreite in allen Fällen mit dem Verhältnis 1 zu 24 zu berechnen. Dieses Verhältnis wird in der Gleichung 5.3 als r bezeichnet. Solange die Bandbreite der limitierende Faktor ist, stehen die Ausführungszeit T, die Datenmenge Q, die Rechenleistung R und die Bandbreite B einer Kernel-Operation in folgenden Beziehungen zueinander:

$$
\begin{aligned}
&T = \tfrac{Q}{B} \quad [\text{s}];\\
&B = r \times R \quad [\text{B/s}];
\end{aligned}
$$

Für tatsächliche Bandbriete: $r = (N_{read} + 2 \times N_{write}) \times 8$ B;

Für effiktive Bandbriete: $r = (N_{read} + N_{write}) \times 8$ B;

N_{read}, N_{write} - *Anzahl der Variablen zum Lesen/Schreiben für ein* FLOP;
T - *Ausführungszeit in* s; R - *Rechenleistung in* FLOPS;
B - *Bandbreite in* B/s; Q - *Datenmenge in* B; (5.3)

Die in Kapitel 4 berechnete Interpolation der Rechenleistung kann in [B/s] umgerechnet werden.[5] Nach dem Einsetzen von Gl. (5.3) in die Gleichungen der Interpolation der Rechenleistung (Gl. (4.9), (4.10), 4.16, 4.17, 4.23 und (4.24)) ergibt sich Gl. 5.4, die die gemessene Bandbreite B^{MESS} abhängig von der CPU-Frequenz f_{cpu} und der Anzahl der Threads p interpoliert.

[5] Im weiteren Verlauf der Arbeit wird die tatsächliche Bandbreite betrachtet.

$$
\begin{aligned}
&\textbf{Ivy Bridge (E5-2690v2)} && : \\
&B_{L1}^{MESS}(f_{cpu}, p) && = 40.77 \times p \times f_{cpu}\ \text{B/s}\ (\pm 1.6071\%); \\
&B_{L2}^{MESS}(f_{cpu}, p) && = 19.737 \times p \times f_{cpu}\ \text{B/s}\ (\pm 2.8178\%); \\
&B_{L3,1\leq p\leq 4}^{MESS}(f_{cpu}, p) && = 12.0416 \times p \times f_{cpu}\ \text{B/s}\ (\pm 1.2\%); \\
&\dots \\
&B_{L3,9\leq p\leq 10}^{MESS}(f_{cpu}, p) && = 11.5520 \times p \times f_{cpu}\ \text{B/s}\ (\pm 1.2\%); \\
&\textbf{Haswell (E5-2680v3)} && : \\
&B_{L1}^{MESS}(f_{cpu}, p) && = 61.366 \times p \times f_{cpu}\ \text{B/s}\ (\pm 1.8783\%); \\
&B_{L2}^{MESS}(f_{cpu}, p) && = 25.617 \times p \times f_{cpu}\ \text{B/s}\ (\pm 1.8633\%); \\
&B_{L3,1\leq p\leq 2}^{MESS}(f_{cpu}, p) && = 13.1616 \times p \times {f_{cpu}}^{0.95}\ \text{B/s}\ (\pm 3.0\%); \\
&\dots \\
&B_{L3,11\leq p\leq 12}^{MESS}(f_{cpu}, p) && = 14.96 \times p \times {f_{cpu}}^{0.88}\ \text{B/s}\ (\pm 3.0\%);
\end{aligned}
\tag{5.4}
$$

Anhand dieser Gleichungen lassen sich die beiden Prozessoren für die unterschiedlichen Fälle leicht miteinander und mit den Spezifikationen der Prozessoren vergleichen. Der interessierte Leser kann eine solche Analyse im Falle des Haltens der Daten im Cache in Anhang H auf Seite 185 finden.

5.1.4.1. Bandbreite im Falle des Haltens der Daten im Hauptspeicher

Analog zum Cache kann die Bandbreite für den Fall des Hauptspeichers aus der gemessenen Rechenleistung ausgerechnet werden. Die Resultate für die Prozessoren *Ivy Bridge* und *Haswell* sind in Abb. 5.4 und in Abb. 5.5 dargestellt. Für eine bessere Übersicht sind die einzelnen Messungen in einer Kurve für die gleiche Anzahl der Threads zusammengefasst. Für die unterschiedlichen CPU-Frequenzen benutze ich verschiedene Farben. Zum Beispiel zeigt die rote Kurve die Bandbreite, die mit der niedrigsten Frequenz f_0 erreicht wurde. Die maximal erreichbare theoretische Bandbreite der beiden Prozessoren ist mit gestrichelten Linien dargestellt.

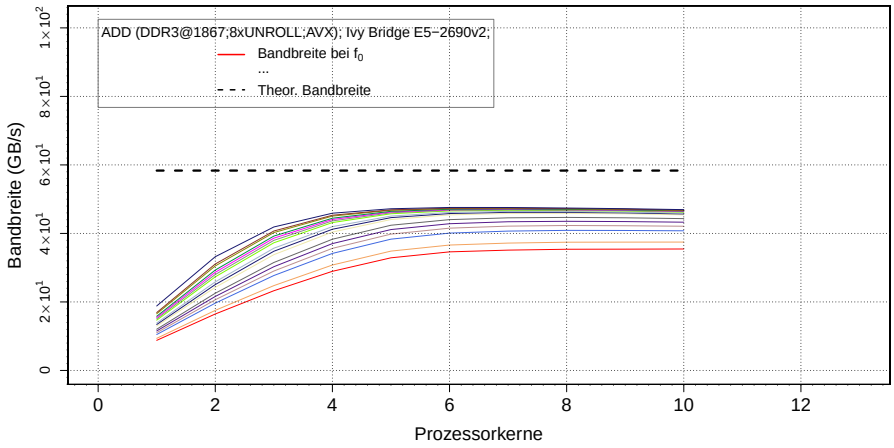

Abb. 5.4.: Bandbreite der Kernel-Operation *Add* auf dem Prozessor *Ivy Bridge E5-2690v2* im Falle des Haltens der Daten im Hauptspeicher.

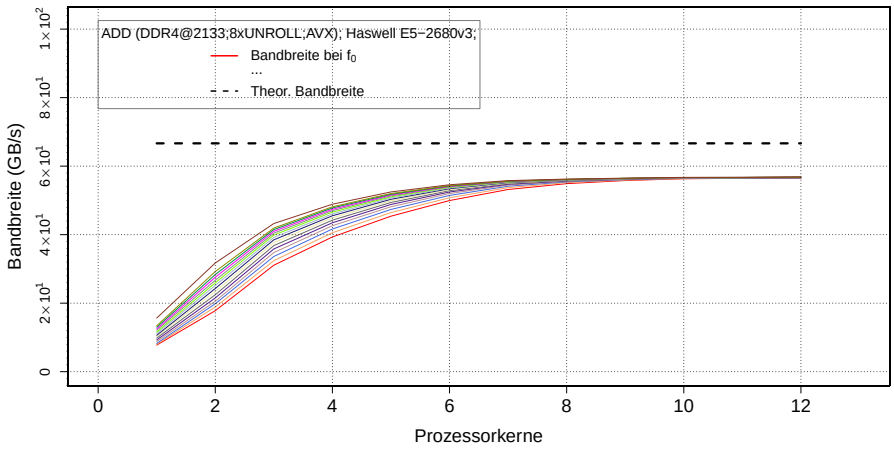

Abb. 5.5.: Bandbreite der Kernel-Operation *Add* auf dem Prozessor *Haswell E5-2680v3* im Falle des Haltens der Daten im Hauptspeicher.

Das unterschiedliche Verhalten der Bandbreite ist deutlich in den Diagrammen zu sehen. Zum Beispiel erreicht der Prozessor *Haswell* die maximale Bandbreite nicht mit allen, sondern mit weniger Prozessorkernen und niedrigeren CPU-Frequenzen. Der Prozessor *Ivy Bridge* braucht dagegen alle Prozessorkerne, um die maximale Rechenleistung zu erreichen. Dieses Verhalten kann nicht mit einem ECM-Modell erklärt werden. Es wird ausschließlich die Bandbreite vorhergesagt, wenn alle Prozessorkerne mit der maximalen CPU-Frequenz getaktet sind. Um den Grund dafür verstehen zu können und eine Hypothese für das unterschiedliche Verhalten der beiden Prozessoren aufzustellen, muss die Prozessorarchitektur näher betrachtet werden.

5.2. Prozessorkomponenten und Frequenzen

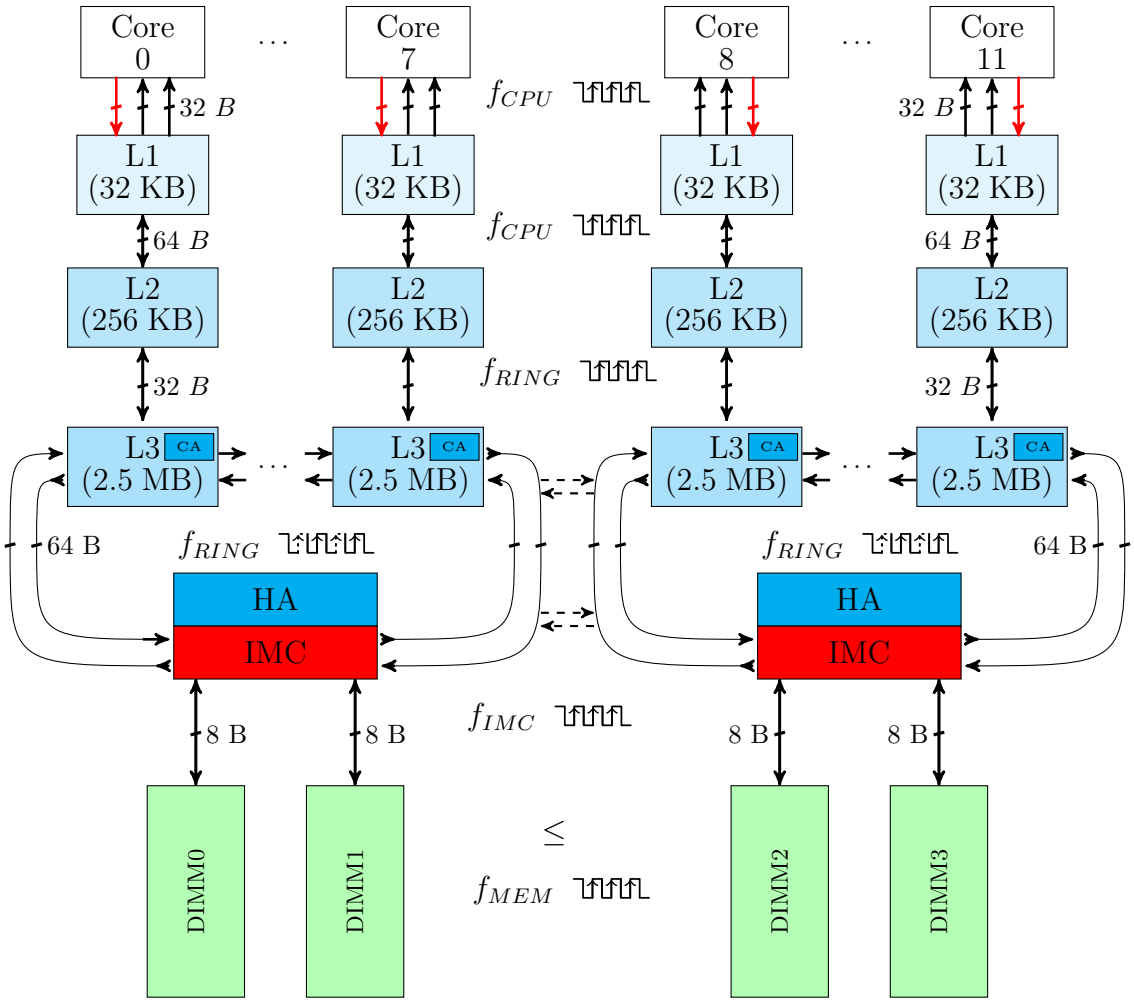

Abb. 5.6.: Speicher-Hierarchiestufen in einem *DTM*-Modell für den Prozessor *Haswell E5-2680v3*. Die Breite der Interfaces zwischen den Speicher-Hierarchiestufen, die Größe des Caches und die Frequenz der Komponenten (Links vom Takt-Symbol) sind in der Skizze angegeben.

Abb. 5.6 stellt das prinzipielle Schema eines erweiterten ECM-Modells für den Prozessor *Haswell* dar. Der Unterschied zum ursprünglichen ECM liegt in der Berücksichtigung der unterschiedlichen Frequenzen, mit denen die Prozessorkomponenten getaktet sind. Zur Unterscheidung wird das von mir erweiterte ECM-Modell „Daten-Transfer-Memory“ Modell oder kurzt DTM genannt. In weiteren Abschnitten folgt eine ausführliche Beschreibung der Prozessorkomponenten in Abb. 5.6.

5.2.1. Prozessorkern und Frequenz f_{CPU}

Die Frequenz der Prozessorkerne sowie die Frequenz der L1- und L2-Caches wird von mir als f_{CPU} bezeichnet. Das Interface zum Datatransfer zwischen einem Kern und seinen beiden Cache-Hierarchiestufen arbeitet mit der gleichen Frequenz. Im Vergleich zum *Ivy Bridge* können die Kerne vom *Haswell* mit unterschiedlichen CPU-Frequenzen f_{CPU} getaktet werden. Ich betrachte den Fall, bei dem alle Prozessorkerne sowohl die gleiche Arbeit als auch die gleiche CPU-Frequenz haben.

5.2.2. Ringbus und Frequenz f_{RING}

Der gesamte L3-Cache ist in L3-Segmenten, den sogenannten „*Multiple Cache Slices*“, zwischen allen Prozessorkernen aufgeteilt. Eine gesamte Cache-Zeile wird abhängig von derer Adresse und möglicherweise anderen Parametern in einem der L3-Segmente gespeichert. Eine Beschreibung, wie die Abbildung des Adressraums auf den L3-Cache aussieht, konnte ich in keiner der öffentlichen Quellen finden. Die entsprechende unbekannte Relation wird als Hashfunktion bezeichnet. Nach [19] werden die Adressen „gleichmäßig“ verteilt. Falls ein Prozessorkern auf eine Cache-Zeile aus einem der L3-Segmente zugreifen muss, wird die Anfrage zuerst von einem lokalen *Cache Agent* (CA) bearbeitet. Wenn die Daten nicht im nächstliegenden L3-Segment vorhanden sind, werden sie über den Ring, den sogenannten „Ringbus“, zum Adressaten geschoben. Für eine genauere Darstellung von den möglichen Cache-Kohärenz-Protokoll-Modi eines Prozessors siehe [57].

5.2.2.1. Vor- und Nachteile eines Ringbusses

Die L3-Segmente sind mit zwei Ringbussen im Prozessor *Haswell* und mit einem Ringbus im Prozessor *Ivy Bridge* miteinander verbunden. Die Verteilung der Prozessorkerne auf den Ringen hängt von deren Anzahl ab. Diese Verteilung muss nicht symmetrisch sein. Die Reihenfolge der Prozessorkerne in Abb. 5.6 kann in der Realität abweichen. Wenn eine Cache-Zeile aus einem Ringbus in einen anderen geschickt wird, passiert diese über einen von zwei zusätzlichen Puffern, die die beiden Ringe miteinander verbinden. Die Puffer sind in Abb. 5.6 mit vier gestrichelten Pfeilen dargestellt.
Die Ring-Topologie hat folgende Vorteile:

- Die Cachegröße skaliert linear mit der Anzahl der Prozessorkerne.
- Der Prozessor kann mit ausgeschalteten oder fehlerhaften Kernen funktionsfähig sein, solange mindestens einer funktioniert.
- Der gesamte Cache steht auch einem Prozessorkern zur Verfügung.
- Die Last kann sich gleichmäßig über alle Ringbus-Komponenten verteilen.

Allerdings hat diese Topologie wesentliche Nachteile. Die durchschnittlichen Latenzzeiten erhöhen sich mit der Erhöhung der Anzahl der Kerne. Die Bandbreite pro Prozessorkern kann sich dadurch reduzieren, so dass die gleichmäßige Verteilung der Last nicht für alle möglichen parallelen Zugriffsmuster realisiert werden kann.
Eine Cache-Zeile kann zwischen den Segmenten bei jedem zweiten Takt der Ringfrequenz in eine der Richtungen geschoben werden. Der Ring, der CA und die L3-Segmente sind mit der Ring-Frequenz getaktet. Die Ring-Frequenz f_{RING} des Prozessors *Ivy Bridge* ist der CPU-Frequenz f_{CPU} gleich. Die Ring-Frequenz des Prozessors *Haswell* kann unabhängig von der CPU-Frequenz gesetzt werden (siehe Abschnitt 4.5.3.1).

5.2.3. Speichercontroller und Frequenz f_{IMC}

Wenn die von CA angefragte Cache-Zeile in keinem der L3-Segmente vorhanden ist, muss diese aus dem SDRAM-Speicher über die integrierten Speichercontroller (IMC) geholt werden. Ein IMC arbeitet mindestens mit zwei Frequenzen. Mit der Frequenz f_{RING} spricht der Controller mit einem der L3-Segmente über den Ring. Die Speicherkanäle zwischen dem IMC und den Speichermodulen (DIMM) sind mit der Frequenz f_{MEM} getaktet (in Abb. 5.6 als „DIMM0“ ,. . . „DIMM3“ gekennzeichnet). Die interne IMC-Frequenz kann aus einem Programm für einige der *Haswell* Prozessormodelle sowohl ausgelesen, als auch geändert werden. Im betrachteten Fall wird der IMC autonom vom Prozessor gesteuert. Ein „Prefetcher“ wird im Modell nicht berücksichtigt.

5.2.4. Speicher und Frequenz f_{MEM}

Wie auch der IMC arbeitet der Speicher mit mehreren Frequenzen. Als f_{MEM} bezeichne ich die Frequenz des IO-Interfaces zum IMC. Der maximale Wert für f_{MEM}

kann im BIOS eingestellt werden.[6] Die Speicherfrequenz f_{MEM} kann im laufenden Betrieb nach [8] geändert werden, dabei müsste der Speicher aber in einem bestimmten Zustand sein, der eine Unterbrechung in der Ausführung der Kernel-Operation verursacht hätte. Die in der Arbeit betrachteten Prozessoren unterstützen keine dynamische Änderung der Speicherfrequenz.
Gleichung Gl. (5.2) definiert die theoretische Hauptspeicher-Bandbreite als das Produkt der Anzahl der Speicherkanäle $N_{mem\_channel} = 4$ mit einer effektiven Speicher-Taktfrequenz f_{MEM}. Die maximale Taktfrequenz des verwendeten DDR3-SDRAM-Moduls ist mit 1866 MHz spezifiziert. Die maximale Taktfrequenz der verwendeten DDR4-SDRAM-Module beträgt 2133 MHz. Die effektive Busbreite der beiden Typen des Speichers beträgt 8 Bytes.
Ein Speichermodul wird in den mir bekannten Rechenleistungsmodellen als „Blackbox“ angesehen. Einer der Gründe dafür ist die Komplexität des Speicheraufbaus. Das DTM-Modell ist keine Ausnahme von dieser Regel. Trotzdem fasse ich in Anhang G auf Seite 177 die Basisprinzipien des Aufbaus eines SDRAM-Modules zusammen, weil die dahinterstehende Technologie einen großen Einfluss auf die Rechenleistung der Prozessoren hat.

5.3. Data-Transfer-Memory Modell (DTM)

Das wichtigste Merkmal eines ECM-Modells ist, dass der Datentransfer zwischen den verschiedenen Hierarchiestufen des Speichers nicht überlappend angenommen wird. Ein von mir entwickeltes „Data-Transfer-Memory Modell“ (DTM) dagegen erlaubt eine Überlappung des Datentransfers, während die Daten zwischen dem IMC und dem Speicher transportiert werden. Somit ergibt sich Gl. (5.6) für die Ausführungszeiten der Kernel-Operation *Add* im Falle des Haltens der Daten im

[6] Die meisten Hauptplatinen ermöglichen sowohl eine automatische Auswahl der Speicherfrequenz als auch das Setzen eines fixierten Wertes. Wenn der Menüeintrag „Automatic“ ausgewählt wird, wird die maximal mögliche Speicherfrequenz gesetzt, die sowohl vom IMC als auch vom SDRAM unterstützt wird.

Hauptspeicher (Vergleiche mit Gl. 5.1 für ECM-Modell).

$$
\begin{aligned}
T^{DTM} \quad & = \max(T^{OL}_{L2\leftrightarrow AVX}, T^{nOL}_{L2\leftrightarrow AVX} + T_{L3\leftrightarrow L2} + T_{L3\leftrightarrow IMC} + \\
& \qquad + T_{MEM\leftrightarrow IMC}); \\
T^{DTM} \quad & \text{- } \textit{DTM-Ausführungszeit einer Kernel-Operation;} \\
T^{OL}_{L2\leftrightarrow AVX} \quad & \text{- } \textit{Ausführungszeit in einem Prozessorkern (ALU,L1,L2)} \\
& \qquad \textit{mit Überlappung zum Datentransfer zwischen den} \\
& \qquad \textit{Uncore-Komponenten (Hauptspeicher, IMC und L3);} \\
T^{nOL}_{L2\leftrightarrow AVX} \quad & \text{- } \textit{Ausführungszeit in einem Prozessorkern ohne Überlappung;} \\
T_{A\leftrightarrow B} \quad & \text{- } \textit{Ausführungszeit der Datentransfer zwischen den Komponenten} \\
& \qquad \textit{A und B ohne Überlappung;}
\end{aligned}
\tag{5.5}
$$

Für die Anwendung dieses Modells wird anstatt der Ausführungszeit die Bandbreite einer Kernel-Operation betrachtet. Nach Gl. (5.3) kann das DTM-Modell für die Bandbreite umformuliert werden. Die IO-Interfaces zwischen den Hierarchiestufen sind im Prozessor bi-direktional. Um das Modell zu vereinfachen, wird der Datentransfer in einer Richtung modelliert, nämlich vom Hauptspeicher zu den AVX-Registern. Die Datenmenge ist dabei die Summe der Daten in beiden Richtungen. Daher kann die Bandbreite mit Gl. (5.6) modelliert werden. Die einzelnen Terme der Gleichung sind für den Prozessor *Haswell* aufgeschrieben, wie im Weiteren erklärt wird.

Zur Erklärung: Der Koeffizient K wird im Bereich zwischen 0 und 1 gesucht, so dass die Abweichung zwischen der modellierten und der gemessenen Bandbreite minimal wird. Wenn K gleich 0 ist, werden die Daten zwischen den AVX-Registern und dem L2-Cache komplett parallel zum Datentransfer zwischen dem Speichercontroller und dem L3-Cache transportiert.

Im ersten Schritt werden die Daten zwischen dem Hauptspeicher und dem IMC über vier Speicherkanäle mit einer Busbreite von 8 B und mit einer Frequenz f_{IMC} transportiert. Im zweiten Schritt werden 64 B in jedem zweiten Takt der Ring-Frequenz f_{RING} zu einem der L3-Segmente über den Ringbus geschoben. Im dritten Schritt werden 32 B in jedem Takt der Ring-Frequenz f_{RING} vom L3-Segment zum L2-Cache weiter verschickt.

Weiter müssen die Daten bis zu den AVX-Registern geliefert werden. Die Bandbreite vom L2-Cache zu den AVX-Registern gleicht im Falle des Haltens der Daten im Hauptspeicher der Bandbreite im Falle des Haltens der Daten im L2-Cache. Somit kann die Transferleistung aus Gl. (4.16) und Gl. (4.17) mit dem Vorfaktor 32 in Gl.

(5.6) eingesetzt werden.

$$
\begin{aligned}
&\frac{1}{B^{DTM}_{MEM\leftrightarrow AVX}} &&= \frac{1}{B^{TH}_{MEM\leftrightarrow IMC}} + \frac{1}{B^{TH}_{IMC\leftrightarrow L3}} + \frac{1}{B^{TH}_{L3\leftrightarrow L2}} + \frac{K}{B^{MESS}_{L2\leftrightarrow AVX}};\\
&B^{TH}_{MEM\leftrightarrow IMC} &&= 8 \times N_{mem\_channel} \times f_{IMC};\\
&B^{TH}_{IMC\leftrightarrow L3} &&= 64 \times \tfrac{f_{RING}}{2} \times p;\\
&B^{TH}_{L3\leftrightarrow L2} &&= 32 \times f_{RING} \times p;\\
&B^{MESS}_{L2\leftrightarrow AVX} &&= 8 \times (N_{read} + 2 \times N_{write}) \times (\tilde{\beta}_{0,L2} \times p \times f_{CPU});\\
&K: &&\begin{cases} 0 \leq K \leq 1.0; \\ \min\limits_{K}(B^{DTM}_{MEM\leftrightarrow AVX} - B^{MESS}_{MEM\leftrightarrow AVX}); \end{cases}\\
&B^{DTM}_{MEM\leftrightarrow AVX} && \text{- Modellierte Bandbreite einer Kernel-Operation;}\\
&B^{TH}_{MEM\leftrightarrow IMC} && \text{- Theoretische Bandbreite zwischen DRAM und IMC;}\\
&N_{mem\_channel} && \text{- Anzahl der 8 B breiten Speicherkanälen;}\\
&B^{TH}_{IMC\leftrightarrow L3} && \text{- Theoretische Bandbreite zwischen IMC und L3;}\\
&B^{TH}_{L3\leftrightarrow L3} && \text{- Theoretische Bandbreite zwischen L3 und L2;}\\
&B^{MESS}_{L2\leftrightarrow AVX} && \text{- Gemesse Bandbreite im Falle L2-Cache;}\\
&N_{read}, N_{write} && \text{- Anzahl der Variablen zum Lesen/Schreiben für ein FLOP;}\\
&B^{MESS}_{MEM\leftrightarrow AVX} && \text{- Gemesse Bandbreite im Falle Hauptspeicher;}\\
&\tilde{\beta}_{0,L2} && \text{- Skalierungsfaktor für die Rechenleistung der Kernel-Operation im Falle L2-Cache;}\\
&K && \text{- Überlappungskoeffizient;}
\end{aligned}
\tag{5.6}
$$

Falls die Ring-Frequenz f_{RING} hoch genug ist, werden die Daten zum L3-Segment und zum L2-Cache schneller geliefert, als die Daten vom Hauptspeicher zum IMC. Deshalb ist anzunehmen, dass die Mikrobefehle in der „Out-of-Order“ Einheit, die für den Datentransfer zwischen den L2-Caches und den AVX-Registern zuständig sind, in der Zwischenzeit für die früher geladenen Daten im L2-Cache ausgeführt werden können.
Der Datentransfer zwischen den L3- und L2-Caches ist nicht als überlappend angenommen. Das liegt daran, dass dieser Datentransfer, wie auch der Datentransfer zwischen den höheren Hierarchiestufen, von dem *Cache Agent* (CA) und dem *Home Agent* (HA) kontrolliert wird (siehe Abschnitt 5.2.3).
Der Überlappungskoeffizient K wird anhand der gemessenen Bandbreite im Falle des Haltens der Daten im Hauptspeicher bestimmt, sodass die mit dem DTM-Modell berechnete Bandbreite, soweit es möglich ist, mit den gemessenen Daten übereinstimmt.

5.3.1. Anwendung des DTM-Modells

Die Werte für f_{RING}, f_{CPU} und B_{L2}^{MESS} wurden in den vorherigen Kapiteln bestimmt. Die Frequenz f_{IMC} hängt von der Häufigkeit der Anfragen des IMC nach den Daten ab. Die maximale Häufigkeit der Anfragen kann mit den Resultaten der Kernel-Operation im Falle des Haltens der Daten im L3-Cache ermittelt oder theoretisch berechnet werden, wie im Abschnitt 5.1.4 gezeigt wird. Anderseits ist f_{IMC} durch die Speicherfrequenz beschränkt:

$$\begin{aligned} f_{IMC} &= min(N_{mem\_channel} \times f_{MEM}, \tfrac{B_{L3}[\mathrm{B/s}]}{N_{mem\_channel} \times 8[\mathrm{B}]}); \\ N_{mem\_channel} &= 4 \ \text{(siehe Gl. (5.2))}; \\ B_{L3} &: \ \text{Bandbreite [B/s] im Falle des L3-Caches (siehe Gl. (5.4))}; \end{aligned} \tag{5.7}$$

In Abb. 5.7 zeigen die farbigen Kurven die berechnete Bandbreite nach dem DTM-Modell für die Prozessoren *Ivy Bridge* und *Haswell*. Zum Beispiel zeigt die rote Kurve die Bandbreite mit der minimalen CPU-Frequenz $f_{CPU} = 1.2$ GHz. Für einen besseren Überblick werden die Werte für die Ausführung mit einem, drei und der maximalen Anzahl der Prozessorkerne gezeigt. Die restlichen Werte haben einen ähnlichen Verlauf. Die gestrichelten Linien zeigen den Verlauf für die Bandbreite zwischen dem IMC und den Speichermodulen $B_{MEM \leftrightarrow IMC}^{TH}$ abhängig von der Anzahl der Prozessorkerne. Die entsprechenden Messwerte sind zum Vergleich mit Kreuzchen angegeben.

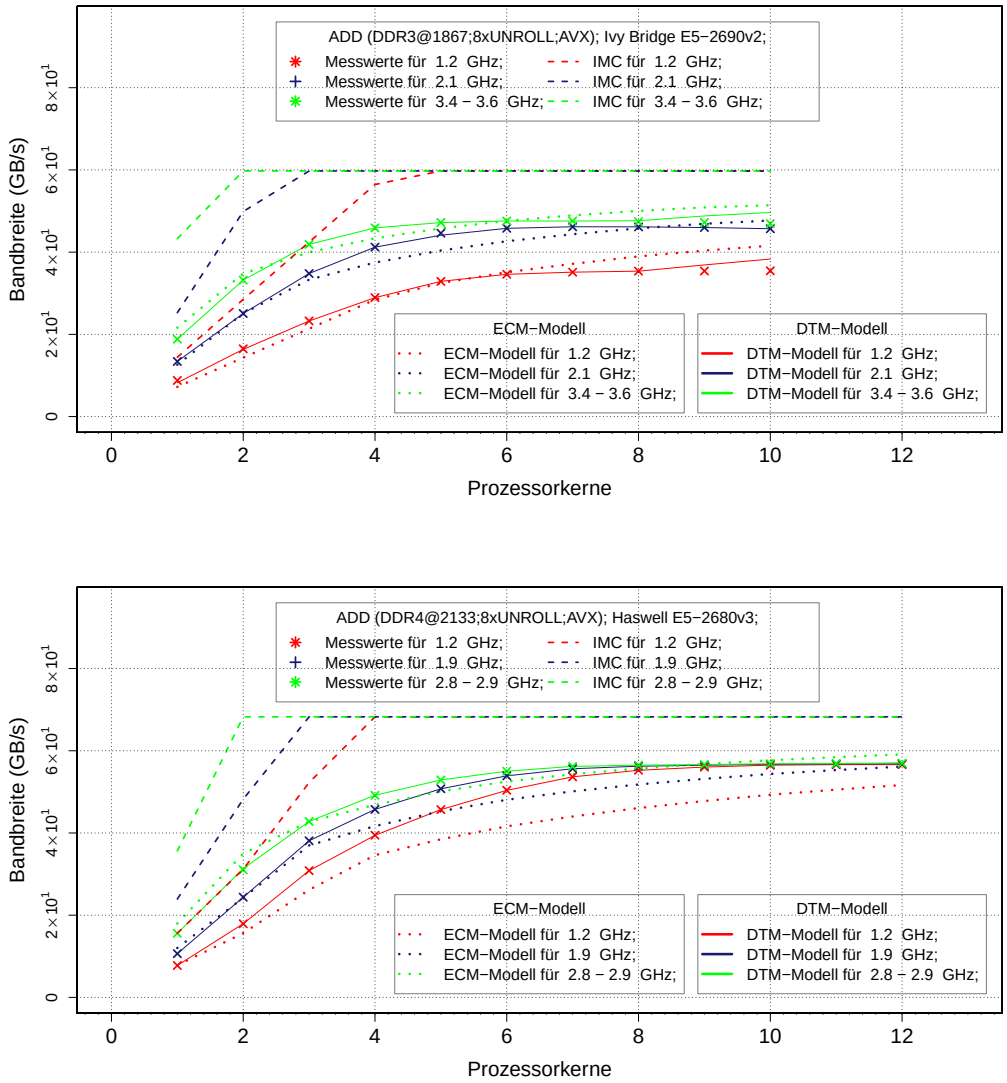

Abb. 5.7.: Mit ECM- und DTM-Modell berechnete Bandbreite der Kernel-Operation *Add* auf den Prozessoren *Ivy Bridge E5-2690v2* (oben) und *Haswell E5-2680v3* (unten) im Fall des Haltens der Daten im Hauptspeicher.

5.3.2. Relativer Fehler und Komponente des DTM-Modells

In Abb. 5.8 ist der Fehler des DTM-Modells relativ zu den Messwerten dargestellt. Das Modell liefert eine gute Bandbreitenapproximation für die Ausführung der Kernel-Operation *Add* auf dem Prozessor *Haswell E-2680v3*. Der Fehler liegt unter einem Prozent.

Die Bandbreitenapproximation für die Ausführung auf dem Prozessor *Ivy Bridge E5-2690v2* zeigt dagegen starke Abweichungen für bestimmte Konfigurationen. Die

Modellierung der Ausführung mit mehreren Threads bei den niedrigeren und höheren CPU-Frequenzen berücksichtigt offensichtlich nicht alle wichtigen Prozesse im Prozessor zeigt aber teilweise gute Übereinstimmungen. Aus dieser Tatsache ergibt sich Grund zu der Annahme, dass der Datentransfer auch zwischen den anderen Speicher-Hierarchiestufen überlappend stattfinden könnte.

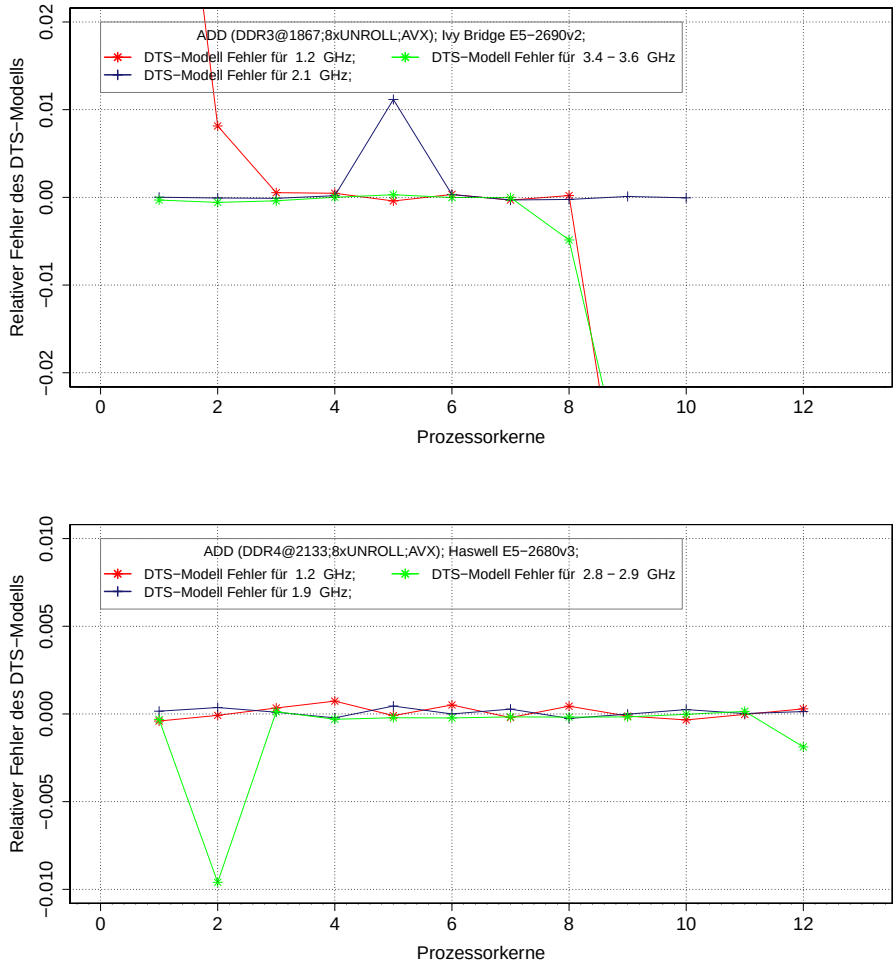

Abb. 5.8.: Fehler des DTM-Modells für die Kernel-Operation *Add* auf den Prozessoren *Ivy Bridge E5-2690v2* (oben) und *Haswell E5-2680v3* (unten) im Falle des Haltens der Daten im Hauptspeicher

In Abb. 5.9 ist die Bandbreite zwischen den einzelnen Komponenten des DTM-Modells für den *Haswell* aus Gl. (5.6) dargestellt, wenn die CPU-Frequenz auf 1.2 GHz, 1.9 GHz und Turbo-Mode gesetzt war. Zusätzlich ist der Überlappungsanteil des Datentransfers zwischen den L2-Cache und den AVX-Registern unter $1 - K$ gezeichnet. Wenn der Koeffizient K gleich eins ist, bedeutet es, dass der Datentransfer zu 100% Prozent überlappend durchgeführt wird, während die Daten aus dem Spei-

cher zu den L3-Segmenten transportiert werden.
Ein sehr ähnlicher Verlauf zeigt der Überlappungsanteil $1 - K$ für die Ausführung der Kernel-Operation auf dem Prozessor *Ivy Bridge* (nicht angezeigt). Der Fehler im DTM-Modell für den *Ivy Bridge* könnte eliminiert werden, wenn der überlappende Datentransfer zwischen den restlichen Hierarchiestufen des Speichers zugelassen wäre. Eine Evaluierung der Modell-Erweiterung bleibt dieser Arbeit folgenden Untersuchungen vorbehalten. In Anhang J auf Seite 195 sind zusätzliche Diagramme für den Verlauf des Koeffizienten K zu finden. Außerdem sind die Resultate des DTM-Modells für die Ausführung von zwei weiteren Kernel-Operationen im Anhang erläutert.

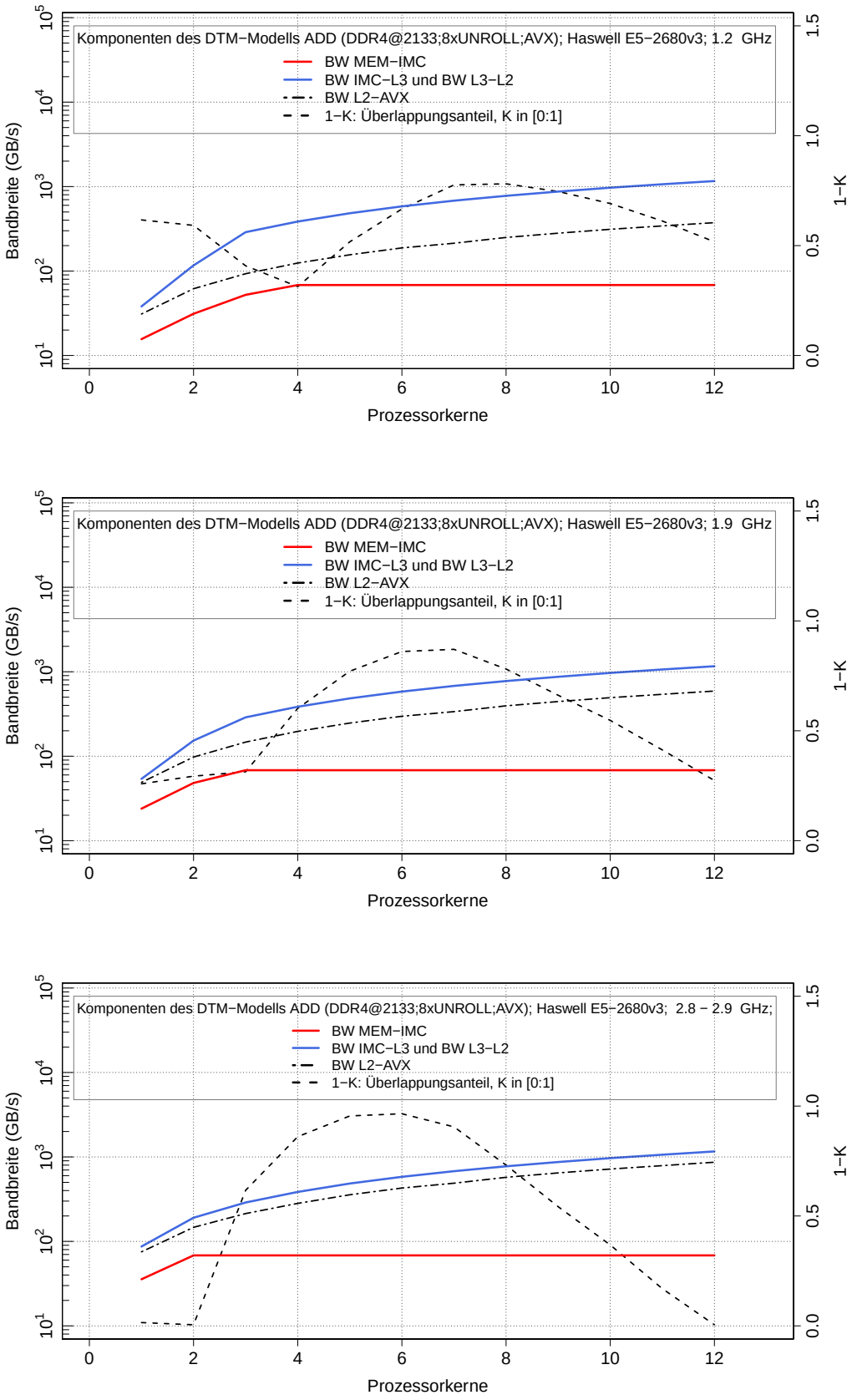

Abb. 5.9.: Komponenten des DTM-Modells für die Ausführung der Kernel-Operation *Add* auf dem Prozessor *Haswell E5-2680v3* im Falle des Haltens der Daten im Hauptspeicher, wenn die CPU-Frequenz auf 1.2 GHz, 1.9 GHz und Turbo-Mode gesetzt war.

6. Energieverbrauch eines Höchstleistungsrechners

Der Energieverbrauch eines Höchstleistungsrechners ist ein komplexes Thema, auf das im Rahmen dieser Arbeit aus Zeitgründen nicht tief eingegangen wird. Dieses hätte den zeitlichen Rahmen der Arbeit stark ausgedehnt. In vorangegangenen Abschnitten wurden die Rechen- und die elektrische Leistung der Prozessoren *Ivy Bridge* und *Haswell* betrachtet, die in den Rechenknoten des Testclusters installiert sind (siehe Anhang A auf Seite 118). Im weiteren Verlauf wird versucht, die entwickelten Methoden der Analyse der Rechenleistung und der elektrischen Leistung auf einem Höchstleistungsrechner anzuwenden. Zusätzlich zur Kernel-Operation *Add* wird das Verfahren der konjugierten Gradienten betrachtet, das in den Anwendungen der Ingenieurwissenschaften weit verbreitet ist.

6.1. Höchstleistungsrechner Cray XC40 (Hazel Hen)

Der Höchstleistungsrechner *Cray XC40* namens „Hazel Hen“ [71] wurde im Oktober 2015 im High Performance Computing Center Stuttgart (HLRS) in den Betrieb genommen. Die theoretische Peak-Performance dieser Maschine beträgt 7.4 Billiarden (10^{15}) Gleitkommazahl-Operationen pro Sekunde (PFlops).[1] In der Tabelle 6.1 sind die „Hazel Hen“ Kernkomponenten aufgelistet. Der Rechenknoten „node03“ des Testclusters ist mit Prozessoren und Speichermodulen der gleichen Version wie die Prozessoren und die Speichermodule von „Hazel Hen“ ausgestattet.[2] Während

[1] Diese Zahl ist das Produkt der maximal theoretisch erreichbaren Rechenleistung eines Prozessorkerns und deren Anzahl im Höchstleistungsrechner. Jeder aus 185 088 Prozessorkernen könnte theoretisch in einem Takt der Basis-CPU-Frequenz zwei vektorisierte Gleitkommazahl-Operationen FMA (siehe Abb. 5.1.2.1) durchführen. Dies ermöglicht in einem Takt acht Additionen und acht Multiplikationen. Die CPU-Frequenz wird durch die spezifizierte Basis-Frequenz des Prozessors ersetzt, wie zum Beispiel 2.5 GHz für den Prozessor *Haswell E5-2680v3*.

[2] Die Revisionsnummer können sich unterscheiden.

"node03" auf einer gewöhnlichen Server Hauptplatine basiert, verwendet die Firma Cray besondere Hauptplatinen: Zur Reduzierung der Ausfälle, der elektrischen Leistung, des Platzes und der Kosten werden ausschließlich die benötigten Chips und Bauelemente eingesetzt. Vier der Hauptplatinen sind in einem „Blade" montiert und benutzen gemeinsam eine Netzwerkkomponente, die mit den Interfaces zu den *Aries*[3]- und Gigabit[4]-Netzwerken ausgestattet ist. Abb. 6.1 stellt die Ausstattung eines „Blades" dar. Die vier „DC-DC" Wandler sind zusätzliche Gleichspannungswandler. Knapp fünfzig „Blades" werden in einem „Kabinett" zusammenmontiert und an ein Schaltnetzteil angeschlossen. Da dieses Schaltnetzteil eine Gleichspannung von 52 V hat, wird die Spannung mit „DC-DC" Wandler auf die benötigten 12 V gesenkt.

Tab. 6.1.: Hauptkomponente des Höchstleistungsrechners „Hazel Hen" (Cray XC40)

Bezeichnung von Komponenten	Menge
Zwei Socket Rechenknoten	7712
Prozessor E5-2680v3 [42]	$2 \times 7712 = 15424$
Prozessor-Kern (Core)	$12 \times 15424 = 185088$
DDR4 Speicher pro Prozessor	64 GB
DDR4 Speicher insgesamt	15424×64 GB= 964 TB
Workspace	10 PB

Sowohl der Testcluster als auch die „Hazel Hen" werden mit Luft gekühlt. Während die Rechenknoten des Testclusters die gekühlte Luft direkt aus dem Serverraum ziehen, kühlt „Hazel Hen" die Luft zusätzlich innerhalb der Maschine mit kaltem Wasser.[5]
Für eine detaillierte Beschreibung des Höchstleistungsrechners sei auf [71] verwiesen.

[3] *Aries* ist ein Hochleitungsnetzwerk, das für die schnelle Kommunikation zwischen den Prozessen während der verteilten Berechnung benutzt wird.

[4] Das Gigabit-Netzwerk wird überwiegend für die administrativen Zwecke benötigt, wie zum Beispiel für die Ausspannung der Prozesse zwischen den Rechenknoten und zur Überwachung der Gesundheit der Komponenten.

[5] Die Kühlung und deren Auswirkung sowohl auf die Rechenleistung als auch auf die elektrische Leistung ist ein nicht zu vernachlässigendes Thema im Bereich der Energieeffizienz eines Höchstleistungsrechners.

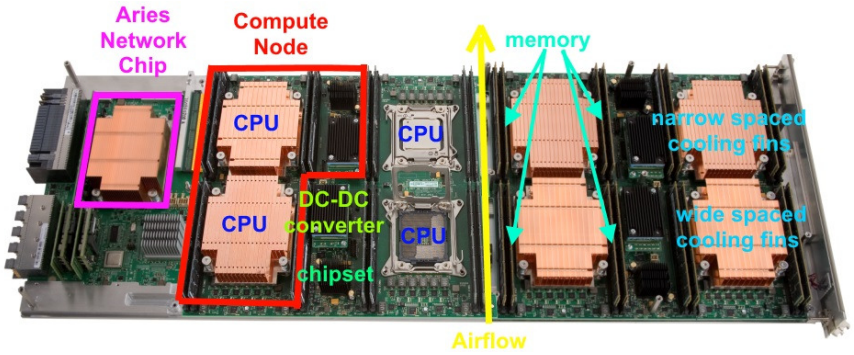

Abb. 6.1.: Ein *Cray XC40* „Blade“ ist mit vier Rechenknoten und einer gemeinsamen Netzwehrkomponente mit den Interfaces zu den *Aries*- und Gigabit-Netzwerken ausgestattet. Ein Rechenknoten besteht aus zwei *Haswell E5-2680 v3* Prozessoren und 8 DDR4-SDRAM Speichermodulen. ©2015 HPCN Production, HLRS

6.1.1. Messsystem in Cray XC40

Jeder Rechenknoten des Höchstleistungsrechners „Hazel Hen“ ist mit einer Messeinrichtung ausgestattet. Die Hauptmerkmale des Messsystems sind in [74] erläutert. Eine detaillierte Beschreibung des Messsystems einer neueren als in der „Hazel Hen“ installierten Version ist in [52] zu finden. Die gemessenen Werte können über die besonderen Hardware-Zähler, die sogenannten *Cray PM-Zähler*, ausgelesen werden. Der Zähler liefert die letzte gemessene momentane elektrische Leistung des Rechenknotens. Der zweite Zähler dient zur Aufzeichnung des Energieverbrauchs. Die Zähler werden alle 100 ms aktualisiert. Die Genauigkeit der neueren Version des Messsystems beträgt 1.5 %.

Wegen der niedrigen Messfrequenz müssen die zu messenden Phasen einer Anwendung mehrere Sekunden dauern. Der andere Unterschied zum Messsystem des Testclusters besteht darin, dass die Messwerte nicht aufgezeichnet werden. Die Messwerte für die elektrische Leistung der Rechenknoten können dagegen in der laufenden Anwendung abgelesen werden.[6]

[6] Aus Sicherheitsgründen ist der Zugriff mit den regulären Rechten auf die RAPL-Zähler nicht gestattet.

6.2. Kernel-Operation Add im Hauptspeicher auf Hazel Hen

Im weiteren Verlauf werden sowohl die Rechenleistung als auch die elektrische Leistung während der Ausführung der Kernel-Operation *Add* im Falle des Haltens der Daten im Hauptspeicher auf dem Rechenknoten „node03“ und auf einem durchschnittlichen Rechenknoten vom „Hazel Hen“[7] verglichen.

6.2.1. Rechenleistung auf Hazel Hen

Wie bereits erwähnt, ist der Rechenknoten „node03“ mit Prozessoren und Speichermodulen der gleichen Version wie die von der „Hazel Hen“ ausgestattet. Außerdem wurden die BIOS-Einstellungen so angepasst, dass der Rechenknoten „node03“ eine Rechenleistung für den Kernel-Operation *Add* zeigt, die mit der Rechenleistung eines durchschnittlichen Rechenknotens von „Hazel Hen“ vergleichbar ist.
In Abb. 6.2 sind die Resultate der Kernel-Operation *Add* auf einem durchschnittlichen „Hazel Hen“ Rechenknoten dargestellt.

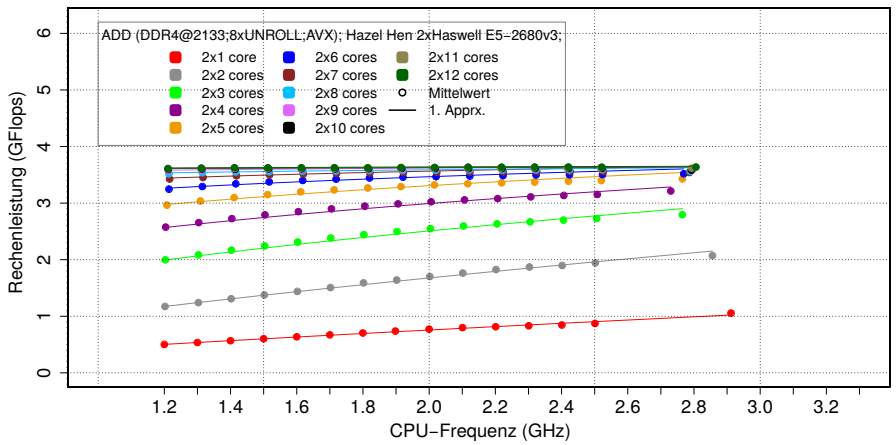

Abb. 6.2.: Durchschnittliche Rechenleistung der Kernel-Operation *Add* im Falle des Haltens der Daten im Hauptspeicher auf zwei Prozessoren eines „Hazel Hen“ Rechenknotens. Die roten Messpunkte zeigen die Rechenleistung während der Berechnung auf zwei Kernen, die je einem der beiden Prozessoren gehören. Wie zu erwarten, sind die Messergebnisse auf „Hazel Hen“ und „node03“ näherungsweise gleich groß. Der größte Unterschied beträgt weniger als 3%.

[7]Der Test wurde zehnmal auf den unterschiedlichen Rechenknoten ausgeführt.

Während der Ausführung der Kernel-Operation war die Berechnung über zwei Prozessoren und nicht, wie auf „node03“, über einen gleichmäßig verteilt. Die roten Messpunkte zeigen zum Beispiel die Rechenleistung während der Berechnung auf zwei Kernen, die je einem der beiden Prozessoren gehören. Im Vergleich mit den auf „node03“ erzielten Ergebnissen aus dem vorhergegangen Abschnitt 4.5.4.1 ist zu erkennen, dass die Rechenleistung tatsächlich doppelt so hoch wie die Rechenleistung eines Prozessors ist. Entsprechend gleich sind die Ergebnisse, wenn die Kernel-Operation *Add* auf den beiden Prozessoren vom „node03“ ausgeführt wird. Der größte Unterschied beträgt weniger als 3%.

6.2.2. Elektrische Leistung von „Hazel Hen“

Weiterhin setzen wir voraus, dass sich die elektrische Leistung der Prozessoren und der Speichermodule der beiden Rechenknoten nicht stark voneinander unterscheidet. Ein direkter Vergleich zwischen der elektrischen Leistung der Prozessoren und deren Speichermodulen im Testcluster und im „Hazel Hen“ ist nicht möglich, weil die *Cray PM-Zähler* die Leistung eines gesamten Rechenknotens zählen.
Das Diagramm in der Abbildung Abb. 6.3 zeigt die durchschnittliche elektrische Leistung eines Rechenknotens von „Hazel Hen“ während der Ausführung der Kernel-Operation *Add* im Falle des Haltens der Daten im Cache.

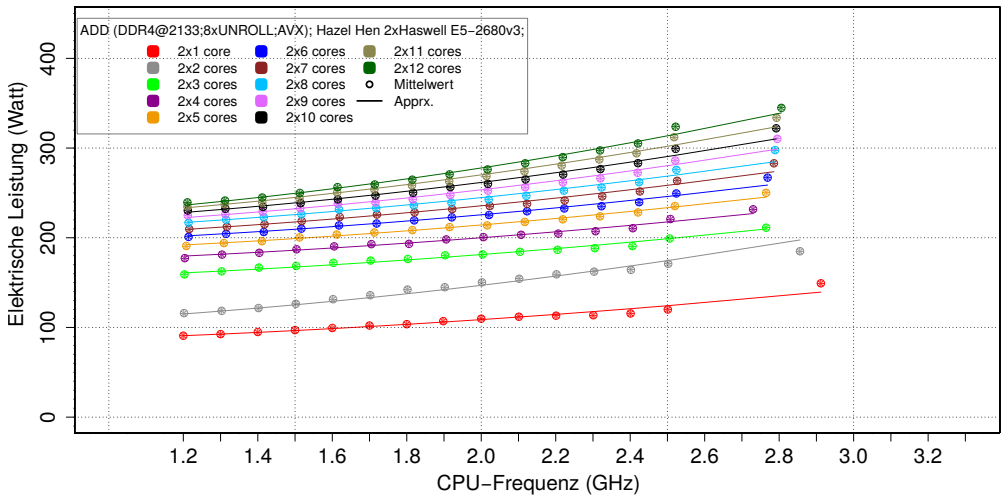

Abb. 6.3.: Durchschnittliche elektrische Leistung eines „Hazel Hen“ Rechenknotens während der Ausführung des Kernels *Add* im Falle des Haltens der Daten im Hauptspeicher. Die elektrische Leistung der Netzwerkkomponenten und des Kühlungssystems ist nicht mitberechnet.

Im Unterschied zur Rechenleistung ist die elektrische Leistung eines „Hazel Hen“ Rechenknotens mehr als doppelt so hoch wie die elektrische Leistung von zwei Prozessoren und derer Speichermodulen, die im vorangegangenen Abschnitt 4.5.4.2 auf dem Rechenknoten „node03“ gemessen wurde. Dies ist kein unerwartetes Ergebnis, da die *Cray PM-Zähler* alle Komponenten auf der Hauptplatine messen (siehe Abschnitt 6.1.1). Zum Vergleich der Messsysteme werden die *Cray PM-Zähler* mit den Messwerten des Testclusters interpoliert. Die elektrische Leistung der beiden Prozessoren des Rechenknotens „node03“ ist dafür in Gl. (6.1) als P_{node03} bezeichnet.

$$P_{node03} = P_{CPU1+RAM} + P_{CPU2+RAM} \tag{6.1}$$

Weiterhin ist die von den *Cray PM-Zählern* ermittelte elektrische Leistung als $P_{HazelHen}$ bezeichnet. Mit Hilfe der linearen Regression kann $P_{HazelHen}$ durch P_{node03} ausgedrückt werden. Die neuen interpolierten Werte sind somit in Gl. (6.2) als $\tilde{P}_{HazelHen}$ bezeichnet:

$$\begin{aligned} P_{node03} &= P_{CPU1+RAM} + P_{CPU2+RAM}; \\ \tilde{P}_{HazelHen} &= p_0 + (1 + p_1) \times P_{node03}; \\ P_{HazelHen} &\simeq \tilde{P}_{HazelHen}; \\ \epsilon_{rel} &= \max \left| \tfrac{P_{HazelHen} - \tilde{P}_{HazelHen}}{P_{HazelHen}} \right|; \\ \epsilon_{abs} &= \max \left| P_{HazelHen} - \tilde{P}_{HazelHen} \right|; \end{aligned} \tag{6.2}$$

Die Koeffizienten p_0 und p_1 werden mit der Methode der kleinsten Quadraten berechnet, so dass der Approximationsfehler im L_2-Norm minimal ist (siehe Abschnitt 4.2). Somit ergibt sich die Gleichung Gl. (6.3) für einen Vergleich zwischen den Messergebnissen der beiden Plattformen.

$$\begin{aligned} \tilde{P}_{HazelHen} &= 6.916\,009 + 1.120\,461 \times P_{node03}; \\ \epsilon_{rel} &\leq 3.34\%; \\ \epsilon_{abs} &\leq 11.51W; \end{aligned} \tag{6.3}$$

Abb. 6.4 stellt die Messwerte P_{node03} gegen die Messwerten $P_{HazelHen}$ mit den gefüllten Kreisen dar. Mit den Kreuzchen sind die Interpolationswerte $\tilde{P}_{HazelHen}$ gegen die gemessenen Messwerte $P_{HazelHen}$ dargestellt.
Der Unterschied zwischen P_{node03} und $P_{HazelHen}$ erreicht bis zu 40.82 Watt. Die elektrische Leistung eines Rechenknotens $\tilde{P}_{HazelHen}$ von „Hazel Hen“ kann mit einem relativen Fehler von weniger als 3.34% durch die Messwerte P_{node03} interpoliert werden.
Hier ist zu bemerken, dass die Kühlung von „Hazel Hen“ $\sim$ 12.12 % der gesamten elektrischen Leistung des Hochleistungsrechners benötigt [62].

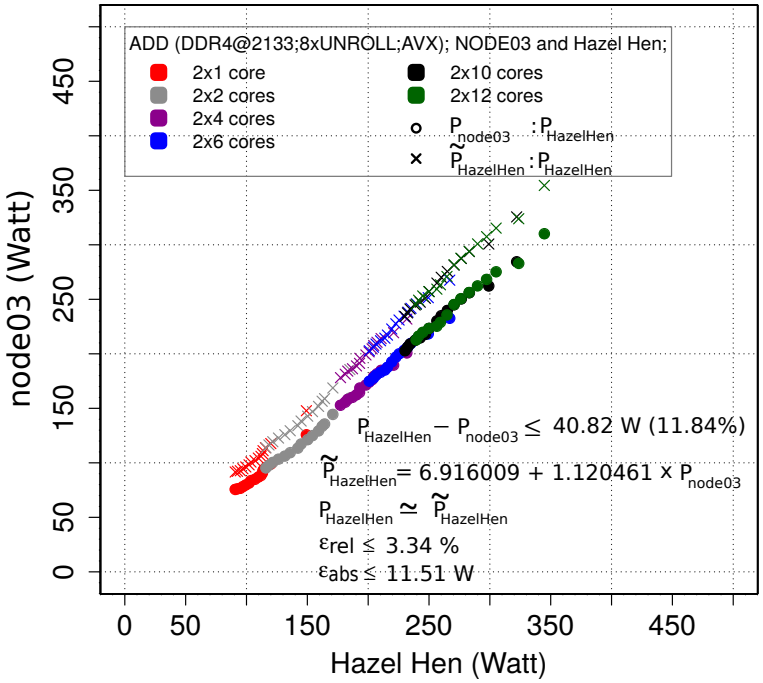

Abb. 6.4.: Vergleich zwischen zwei unterschiedlichen Messverfahren der elektrischen Leistung. P_{node03} ist gegen $P_{HazelHen}$ mit einem gefüllten Kreis gezeichnet. Mit dem Kreuzchen ist $P_{HazelHen}$ gegen $\tilde{P}_{HazelHen}$ dargestellt.

6.3. Kernel-Operation PETsC-CG auf „Hazel Hen“

Wie bereits erwähnt, gehört die Kernel-Operation *Add* zu der Gruppe der sogenannten „STREAM-Benchmarks“. Diese sind durch die unabhängigen Speicherzugriffssequenzen charakterisiert, die aus den nacheinander folgenden virtuellen Adressen bestehen. Wie in Abschnitt IV des Anhangs G auf Seite 177 erläutert, werden die Daten im „Burst“-Modus zwischen dem Speicher und dem L3-Cache am effektivsten übertragen.

Eine andere Gruppe der Algorithmen beschäftigt sich mit den großen linearen schwachbesetzten Gleichungssystemen, die zum Beispiel durch die Anwendung der Finite Element Methode (FEM) auf einer komplexeren Geometrie entstehen, wie das in der Arbeit von Herrn Dr. Ralf Schneider für die Modellierung der Knochenstruktur gemacht wurde [68]. Die Lösung der entstandenen großen linearen Gleichungssysteme wurde mit Hilfe der Bibliothek PETsC [5] und deren parallelen Implementierung des Verfahrens der konjugierten Gradienten (CG) durchgeführt.

Das CG-Verfahren besteht aus unterschiedlichen Operationen auf Skalaren, Vektoren und schwachbesetzten Matrizen. Während die Vektorelemente mit einem direkten Speicherzugriff referenziert werden, müssen die Elemente einer schwachbesetzten Matrix mit einem indirekten Zugriff angesprochen werden.
Das Ziel des folgenden Experimentes war zu überprüfen, ob die beobachteten Eigenschaften eines „STREAM“-Benchmarks auch für einen komplexen Algorithmus in realen Anwendungen beobachtbar ist. Im weiteren Verlauf der Arbeit wird kurz über die Komplexität und über die wichtigsten Merkmale der Kernel-Operation *PETsC-CG* diskutiert. Für eine ausführliche Beschreibung des Verfahrens der konjugierten Gradienten siehe [76] und die dort angegebenen Quellen.

6.3.1. Problembeschreibung der Kernel-Operation PETsC-CG

Die diskretisierte Poisson-Gleichung ist ein Standardproblem für einen Test des CG-Verfahrens. In dieser Arbeit benutze ich die 3-dimensionale Poisson-Gleichung mit homogenen Dirichlet-Randbedingungen:

$$\begin{cases} \Delta\varphi(x) = 6, \quad \overline{x} = \overline{(x_1, x_2, x_3)} \in \Omega \subset \mathbb{R}^3; \\ \varphi(\overline{x}) = x_1^2 + x_2^2 + x_3^2, \quad \overline{x} \in \delta\Omega; \end{cases} \tag{6.4}$$

Der Laplace-Operator Δ wird mit einem 27-Punkt-Stern Diskretisierungsschema auf einem äquidistanten Gitter des 3-dimensionalen Gebietes $\Omega \equiv [0;1] \times [0;1] \times [0;1]$ angewendet. Das Diskretisierungsschema ist in [61] beschrieben. Somit ergibt sich ein lineares schwachbesetztes Gleichungssystem $A\overline{x} = \overline{b}$. Die Randelemente aus $\delta\Omega$ werden entsprechend den Dirichlet-Randbedingungen aus der Matrix eliminiert und von der rechten Seite $\overline{b}$ abgezogen. Die Größe der Matrix und der Vektoren hängt von der Schrittweite h zwischen den benachbarten Gitterpunkten ab. An allen inneren Punkten besitzt die Matrix 27 Nicht-Null-Elemente pro Zeile. Die Nummerierung der Matrix-Elemente ist in lexikographischen Reihenfolge zeilenweise durchgeführt.

6.3.2. Matrix- und Vektorverteilung

Die PETsC Bibliothek verwendet das „Message Passing Interface“ (MPI) für die Parallelisierung. Sowohl die Vektoren als auch die Matrizen werden über die MPI-Prozesse verteilt, so dass jeder Prozess eine gleiche Menge Arbeit hat. Im Falle der

Poisson-Gleichung werden die Matrixzeilen und die entsprechenden Vektorelemente gleichmäßig über die MPI-Prozesse verteilt. Analog zum Fall der Kernel-Operation *Add* wurde die Kernel-Operation *PETsC-CG* mit der unterschiedlichen Anzahlen der Prozessorkernen eines „Hazel Hen“ Rechenknotens ausgeführt. Anstatt der Threads werden die MPI-Prozesse verwendet, die gleichmäßig über zwei Prozessoren zu verteilen sind. Die lokale Größe der Matrix A ist fixiert, so dass ein MPI-Prozess $128 \times 128 \times 128$ Elemente des globalen Gitters speichert. Jeder MPI-Prozess benötigt damit ~ 0.45 GB Speicherplatz.

6.3.3. Matrix-Vektor-Multiplikation

Die Matrix-Vektor-Multiplikation (MVM) $\overline{y} = A\overline{x}$ hat die höchste Komplexität von den in CG verwendeten Operationen der linearen Algebra. Quelltext 6.1 zeigt einen Pseudocode für MVM, wenn die Matrix im Format „Compressed Sparse Row“ (CSR) gespeichert ist. CSR besteht aus drei Feldern. Das erste Feld `aa` beinhaltet die Werte der Matrixelemente. Die entsprechenden Spaltenindizien werden im Feld `adjncy` gespeichert. Im dritten Feld `xadj` sind die Startindizien einer Zeile in den Feldern `aa` und `xadj` gespeichert. Die verwendete Installation der Bibliothek PETsC auf „Hazel Hen“ benutzt für die Speicherung der Indizes und der Matrixelemente die 64-Bit Datenformate. Während einer MVM führt ein MPI-Prozess näherungsweise $2 \times 27 \times 2.097\,152 \times 10^6$ Gleitkommazahl-Operationen durch.

Quelltext 6.1: Der Pseudocode für die Matrix-Vektor-Multiplikation im CSR-Format.

```
double mat_value, vec_value, tmp_value;
int64_t col_idx;
for(ii=0; ii<length; ii++){
      const int64_t first_col=xadj[ii];
      const int64_t next_first_col=xadj[ii+1];
      tmp_value=0.0;
      for(jj=first_col; jj<next_first_col; jj++)
      {
              col_idx=adjncy[jj];
              mat_value=aa[jj];
              vec_value=xx[col_idx];
              tmp_value+=mat_value*vec_value;
      }
      yy[ii]=tmp_value;
}
```

Sowohl GNU- als auch Intel-Compiler konnten den Code nicht vektorisieren, so dass erst ein Viertel jedes verwendeten AVX-Registers während der Ausführung belegt war.
Die verteilte MVM verwendet die MPI-Nachrichtenroutinen, um sich mit den benötigten Vektorelementen zwischen den Prozessen auszutauschen.

6.3.4. Rechenleistung der Kernel-Operationen Add und PETsC-CG auf „Hazel Hen“

Für die Messungen der durchschnittlichen Rechen- und elektrischen Leistung wurde die Kernel-Operation *PETsC-CG* mit hundert Iterationen für die unterschiedliche Anzahl der MPI-Prozesse ausgeführt. Der Test wurde auf zehn unterschiedlichen „Hazel Hen“ Rechenknoten wiederholt. In einer Iteration wurden zusätzlich zur MVM sechs Vektor-Vektor-Operationen und drei MPI-Routinen ausgeführt. Allerdings ist die Größe der Matrix so ausgewählt, dass der Kommunikationsaufwand viel kleiner als der Berechnungsaufwand war.
Abb. 6.5 stellt die durchschnittliche Rechenleistung der Kernel-Operationen *PETsC-CG* auf einem „Hazel Hen“ Rechenknoten dar.

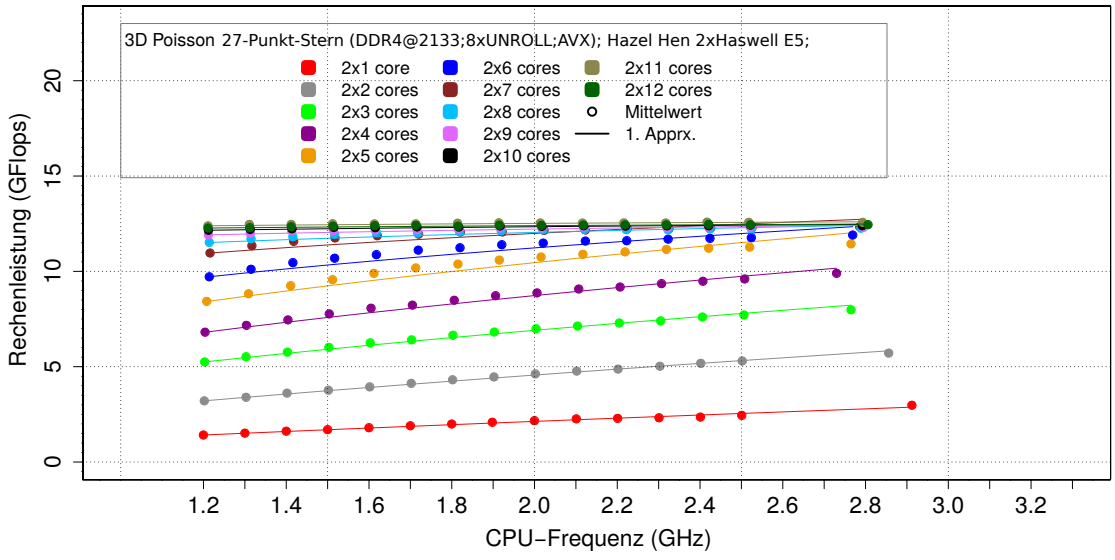

Abb. 6.5.: Durchschnittliche Rechenleistung der Kernel-Operation *PETsC* auf einem „Hazel Hen“ Rechenknoten. Die Rechenleistung ist doppelt so hoch wie auf einem Prozessor des gleichen Modells *Haswell E5-2680 v3* im Testcluster.

Der Verlauf der Rechenleistung abhängig von der CPU-Frequenz und von der Anzahl der aktiven Prozessorkerne sieht auf den ersten Blick dem Verlauf ähnlich, der für die Kernel-Operation *Add* zu beobachten ist (siehe Abb. 6.2). Der Grund für die generell höhere Rechenleistung besteht darin, dass die Daten im Cache teilweise wiederverwendet werden.
Allerdings zeigen die Koeffizienten der Rechenleistungsapproximation $R(f) = \beta_1 \times f^{\gamma}$ (siehe Abschnitt 4.3) der beiden Kernel-Operationen in der Tab. 6.2 einige Unterschiede. Der Exponent γ zeigt darauf an, wie stark die Erhöhung der CPU-Frequenz die Rechenleistung beeinflusst. Die Werte in der Tabelle zeigen, dass die Rechenleistung der Kernel-Operation *PETsC-CG* stärker von der CPU-Frequenz als die Rechenleistung der Kernel-Operation *Add* abhängt. Dies ist ein guter Hinweis, dass der Cache teilweise wiederverwendet wird. Allerdings strebt der Exponent in beiden Fällen wegen der beschränkten Speicherleistung zur Null, wenn die Anzahl der MPI-Prozesse und auf solche Weise die Häufigkeit der Speicherzugriffe erhöht werden.
Für die genauere Analyse der Unterschiede bietet sich eine weitere Erweiterung des DTM-Modells unter Berücksichtigung der Cache-Wiederverwendung an.

Tab. 6.2.: Die Koeffizienten der Rechenleistungsapproximation $R(f) = \beta_1 \times f^{\gamma}$ für die Kernel-Operation *Add* im Falle des Haltens der Daten im Hauptspeicher und für die Kernel-Operation *PETsC-CG* im betrachteten Fall. Die Rechenleistung ist für die unterschiedliche Anzahl der aktiven Prozessorkerne auf einem Rechenknoten vom „Hazel Hen“ angegeben.

Cores	Add_{RAM}			$PETsC-CG_{128x128x182}$		
	β_1	γ	ϵ_{rel}	β_1	γ	ϵ_{rel}
2x1	0.426	0.75	0.45×10^{-1}	0.122×10^{1}	0.8	0.45×10^{-1}
2x2	0.101×10^{1}	0.67	0.56×10^{-1}	0.283×10^{1}	0.69	0.21×10^{-1}
2x3	0.181×10^{1}	0.44	0.9×10^{-1}	0.475×10^{1}	0.54	0.29×10^{-1}
2x4	0.24×10^{1}	0.3	0.64×10^{-1}	0.622×10^{1}	0.49	0.29×10^{-1}
2x5	0.284×10^{1}	0.21	0.81×10^{-1}	0.776×10^{1}	0.43	0.48×10^{-1}
2x6	0.317×10^{1}	0.13	0.76×10^{-1}	0.987×10^{1}	0.29	0.36×10^{-1}
2x7	0.339×10^{1}	0.07	0.54×10^{-1}	0.106×10^{2}	0.18	0.30×10^{-1}
2x8	0.351×10^{1}	0.03	0.21×10^{-1}	0.114×10^{2}	0.09	0.19×10^{-1}
2x9	0.357×10^{1}	0.02	0.32×10^{-1}	0.118×10^{2}	0.05	0.1×10^{-2}
2x10	0.360×10^{1}	0.01	0.34×10^{-1}	0.121×10^{2}	0.03	0.55×10^{-2}
2x11	0.369×10^{1}	0.01	0.24×10^{-1}	0.123×10^{2}	0.02	0.44×10^{-2}
2x12	0.361×10^{1}	0.01	0.9×10^{-2}	0.122×10^{2}	0.02	0.45×10^{-2}

6.3.5. Elektrische Leistung der Kernel-Operationen Add und PETsC-CG auf „Hazel Hen“

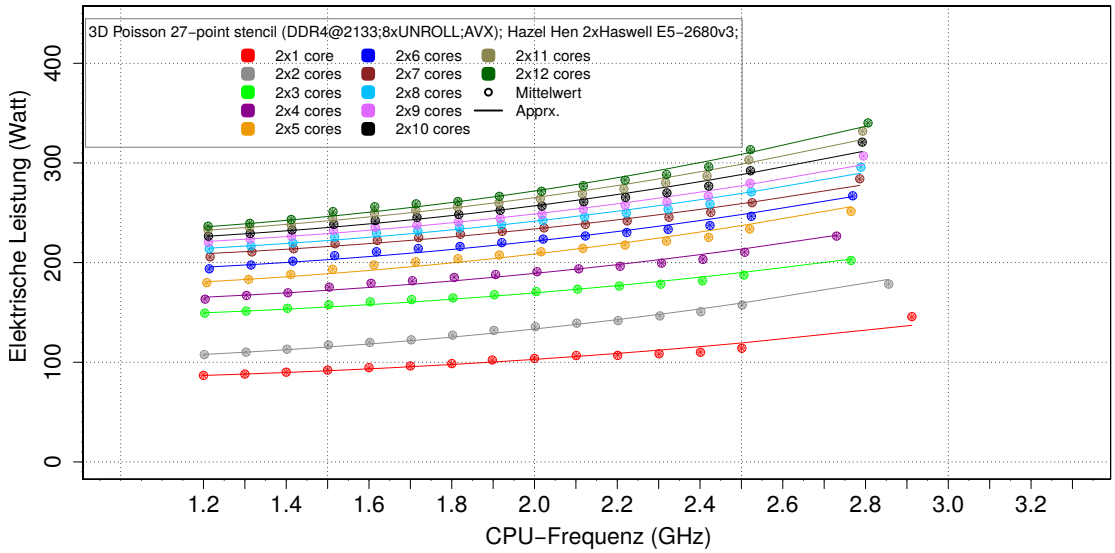

Abb. 6.6.: Das Diagramm zeigt die durchschnittliche elektrische Leistung eines „Hazel Hen“ Rechenknotens während der Ausführung der Kernel-Operation *PETsC-CG*. Die elektrische Leistung der Netzwerkkomponenten und des Kühlungssystems ist nicht mitberechnet.

Abb. 6.6 stellt die elektrische Leistung eines durchschnittlichen „Hazel Hen“ Rechenknotens für die Kernel-Operation *PETsC-CG* dar. Wie im Falle der Rechenleistung sind die Unterschiede zwischen den beiden Kernel-Operationen schwierig festzustellen. Der Vergleich kann durch die Approximationsgleichungen Gl. (6.5) für die elektrische Leistung deutlich gemacht werden. Gl. (6.5) beinhaltet die Messungen für drei Fälle:

- Elektrische Leistung des *Haswell* Prozessors und des Hauptspeichers für die Kernel-Operation *Add* im Falle des Haltens der Daten im Hauptspeicher (siehe Abschnitt 6.2.2). Die Berechnungen wurden auf den Prozessorkernen eines Prozessors durchgeführt.

- Elektrische Leistung eines „Hazel Hen“ Rechenknotens für die Kernel-Operation *Add* im Falle des Haltens der Daten im Hauptspeicher. Die Berechnungen wurden auf den Prozessorkernen der beiden Prozessoren paarweise durchgeführt.

- Elektrische Leistung eines „Hazel Hen“ Rechenknotens für die Kernel-Operation

PetsC-CG im Falle des Haltens der Daten im Hauptspeicher. Die Berechnungen wurden auf den Prozessorkernen der beiden Prozessoren paarweise durchgeführt.

Add (node03 1xHaswell):

$$P_{II}(f,p)^{ADD}_{RAM} = (25.46588 + 8.26171 \times p) + (0.49423 + 2.39527 \times p) \times f^{1.91};$$
$$p \in (1 \times 1, 1 \times 2);\ \epsilon_{rel} \leq 0.05;$$
$$P_{II}(f,p)^{ADD}_{RAM} = (65.52072 + 2.02131 \times p) + (2.02131 + 0.32525 \times p) \times f^{2.11};$$
$$p \in (1 \times 3, 1 \times 4, ..., 1 \times 12);\ \epsilon_{rel} \leq 0.077;$$

Add (Hazel Hen 2xHaswell):

$$P_{II}(f,p)^{ADD}_{RAM} = (63.74772 + 7.19333 \times p) + (2.01264 + 3.51754 \times p) \times f^{1.77};$$
$$p \in (2 \times 1, 2 \times 2);\ \epsilon_{rel} \leq 0.057;$$
$$P_{II}(f,p)^{ADD}_{RAM} = (152.047746 + 3.19220 \times p) + (1.931612 + 0.242121 \times p) \times f^{2.65};$$
$$p \in (2 \times 3, 2 \times 4, ..., 2 \times 12);\ \epsilon_{rel} \leq 0.028;$$

PETsC-CG (Hazel Hen 2xHaswell):

$$P_{II}(f,p)^{CG}_{128x128x128} = (63.68315 + 18.3396 \times p) + (1.2453 + 1.74137 \times p) \times f^{2.77};$$
$$p \in (2 \times 1, 2 \times 2);\ \epsilon_{rel} \leq 0.053;$$
$$P_{II}(f,p)^{CG}_{128x128x128} = (117.2081 + 9.939 \times p) + (5.52700 + 0.212965 \times p) \times f^{2.45};$$
$$p \in (2 \times 3, 2 \times 4, ..., 2 \times 12);\ \epsilon_{rel} \leq 0.03; \tag{6.5}$$

Der statische Anteil der elektrischen Leistung während der Berechnung mit mehr als zwei MPI-Prozessen ist deutlich höher als während der Berechnung mit zwei MPI-Prozessen. Dies gilt für die Leistung eines „Hazel Hen“ Rechenknotens. Im Vergleich zur Messung eines Prozessors und des Hauptspeichers (*Add (node03 1xHaswell)*) ist der statische Anteil des Rechenknotens mehr als zweimal größer.
Die Berechnung mit mehr als zwei MPI-Prozessen zeigt zuerst einen deutlich höheren statischen Anteil für die Kernel-Operation *Add* $(152.047746 + 3.19220 \times p)$ als der entsprechende statische Anteil für die Kernel-Operation *PETsC-CG*. Erst bei der Berechnung mit 18 MPI-Prozessen gleichen sich die beiden statischen Anteile. Dies passiert, wenn die obere Grenze der Rechenleistung erreicht ist. Wie im vorangegangenen Abschnitt 4.5.4.2 erläutert, steigt der statische Anteil wegen der Erhöhung

der Anfragen zu den Speichermodulen und durch die dynamisch steigende Frequenz des L3-Caches. Die genaueren Gründe für die Unterschiede können erst durch eine detaillierte Analysis der Kernel-Operation *PETsC-CG* gefunden werden, wie es in den vorangegangenen Abschnitten für die Kernel-Operation *Add* gezeigt wurde. Eine weitere Analyse wurde nicht gemacht, da diese den zeitlichen Rahmen der Arbeit zu stark ausgedehnt hätte. Hier ist zu bemerken, dass die Approximation der elektrischen Leistung eines gesamten Rechenknotens und die Approximation der elektrischen Leistung eines Prozessors und seines Hauptspeichers schwer zu vergleichen sind. Die elektrische Leistung der zusätzlichen Hardware-Komponenten des Rechenknotens hängt scheinbar nicht linear von der Anzahl der Threads und der CPU-Frequenz ab. Daher unterscheiden sich die Potenzen in der Approximation für die Kernel-Operation *Add* in den Fällen *Add (node03 1xHaswell)* und *Add (Hazel Hen 2xHaswell)*.

7. Zusammenfassung und Ausblick

Im Folgenden sind wesentliche Ergebnisse der Arbeit zusammengefasst. Dabei wird über das mögliche weitere Vorgehen im Hinblick auf die neue gestellten Fragen hingewiesen.

7.1. Messsystem für Testcluster

In Kapitel 2 dieser Arbeit wurde gezeigt, wie ein Messsystem basierend auf mehreren Analog-Eingabekarten die elektrische Leistung der Hardwarekomponenten eines Clusters aufzeichnen kann. Dabei sind die Kosten und die Genauigkeit der Messung zwei wichtige limitierende Faktoren. Anhand der Messungen an einem Hochlast-Widerstand wurde gezeigt, dass die Störsignale während der Messung größtenteils durch die Pulsweitenmodulation eines Schaltnetzteiles zustande kommen. Die Störungen unterliegen dabei der Normalverteilung. Daher können diese mit den Mean- und Median-Filtern stark reduziert werden. Der Wiederholungsfehler in den gefilterten Messdaten sinkt von $\sim$ 1.95 W auf 0.2 W bei einer Last von 65 W.
Ein viel kleinerer Wiederholungsfehler ohne Filterung konnte durch die Verwendung vom Lithium-Polymer-Akkumulator erreicht werden (0.2 W). Obwohl es äußerst aufwendig ist, einen leistungsstarken Rechner mit einem Akkumulator zu betreiben, erweist sich diese Methode als beste Wahl. Dies wiederum findet seine Bestätigung in weiteren Untersuchungen nach Energieeffizienz der Hardwarekomponenten.

7.2. Elektrische Leistung im Betrieb ohne Rechenlast

Mit der ersten Anwendung des Messsystems in Kapitel 3 wurde dessen Fähigkeit bestätigt, zeitlich hochauflösende Messungen der Prozessoren und des Speichers durchzuführen (bis zu 10 µs). Es wurden dabei die relevanten Eigenschaften des

Testclusters im Bereich der Energieeffizienz geprüft. Es wurde gezeigt, dass ein Rechner auch ohne Rechenaufgabe viel Strom verbraucht. Die elektrische Leistung von zwei Prozessoren und den Speichermodulen beträgt durchschnittlich 30 Watt. Dies ist ≈ 30 % der elektrischen Leistung eines Rechenknotens. Außerdem sind die beiden Prozessoren in einem Dual-Socket-Rechner stark miteinander gekoppelt. Wenn ein Prozessor wegen der Betriebssystemdienste regelmäßig den energiesparenden Zustand *C6-PState* verlässt, muss der zweite Prozessor diesem folgen. Innerhalb weniger Mikrosekunden ändert sich die elektrische Leistung der beiden Prozessoren bis zu 50 W. Die Messung der Spannung ist dabei für eine hohe Genauigkeit notwendig, denn es hat sich erwiesen, dass die vom Schaltnetzteil erzeugte Spannung nicht konstant bleibt, wenn sich die Höhe des Stroms rasant ändert.
Bei der testweisen Verwendung des Lithium-Polymer-Akkumulators als Energiequelle wurde außerdem eine leichte Reduzierung des durchschnittlichen Stromverbrauchs der Prozessoren und des Speichers registriert (zwischen 5 % und 10 %). Dies findet eine indirekte Bestätigung auch in den Versuchen, die in „Lawrence Berkeley National Laboratory“ durchgeführt wurden [73]. Die Gründe für die Erhöhung der Energieeffizienz müssen in dieser Arbeit folgenden Untersuchungen geklärt werden.

7.3. Elektrische Leistung und Rechenleistung der Kernel-Operation *Add*

In Kapitel 4 untersuchte ich am Beispiel der Stream-Operation *Add*, welche Zusammenhänge es zwischen dem Stromverbrauch eines Prozessors und des Hauptspeichers, CPU-Frequenz, Grad der Parallelisierung und Rechenleistung gibt. Diese Kernel-Operation ist einerseits einfach zu analysieren und anderseits ausreichend komplex, um die wichtigen Prozessorkomponente unter großer und teilweise kontrollierter Last zu setzen.
Es hat sich erwiesen, dass die elektrische Leistung eines Prozessors und dessen Speichermodulen in allen betrachteten Fällen nach Gl. 7.1 mit einer hohen Genauigkeit approximiert werden kann.[1] Die berechneten Potenzen $\tilde{\lambda}_l$ der CPU-Frequenz sind dabei deutlich anders als in den bekannten Modellen, wie [32], angenommen. Wie in Kapitel 6 gezeigt wurde, gelten die folgenden Überlegungen gleichermaßen nicht nur

[1] Die einzelnen Ausnahmen können durch Turbo-Mode verursacht werden.

für die Stream-Operationen, sondern auch für die komplexeren Kernels.

$$
\begin{aligned}
P(f_{cpu}, p)_l &= (\tilde{\alpha}_{0,0,l} + \tilde{\alpha}_{0,1,l} \times p) + (\tilde{\alpha}_{1,0,l} + \tilde{\alpha}_{1,1,l} \times p) \times {f_{cpu}}^{\tilde{\lambda}_l}; \\
R(f_{cpu}, p)_l &= (\tilde{\beta}_{1,1,l} \times p) \times {f_{cpu}}^{\tilde{\gamma}_l};
\end{aligned}
$$

$$
\begin{aligned}
P(f_{cpu}, p): &\quad \textit{Approximation der elektrischen Leistung}; \\
R(f_{cpu}, p)_l: &\quad \textit{Approximation der Rechenleistung}; \\
f_{cpu}: &\quad \textit{CPU-Frequenz}; \\
p: &\quad \textit{Anzahl der Threads}; \\
l: &\quad \textit{Hierarchiestufe des Speichers } l \in \{L1, L2, L3, RAM\}; \\
\alpha, \lambda, \beta, \gamma: &\quad \textit{Koeffizienten der Approximationen};
\end{aligned}
\tag{7.1}
$$

Vier Fälle sind zu unterscheiden. In den ersten zwei Fällen wurden die Prozessorkerne, der L1-Cache und der L2-Cache unter Last gesetzt. Die elektrische Leistung der Prozessoren *Ivy Bridge* und *Haswell* konnte über die gesamten Definitionsbereiche für die CPU-Frequenz und Thread-Anzahl analytisch beschrieben werden. Die Rechenleistung zeigte eine lineare Abhängigkeit von der CPU-Frequenz und von der Thread-Anzahl.
Im dritten Fall wurden die „Uncore“-Komponenten der beiden Prozessoren zusätzlich unter die Last gesetzt. Der Unterschied in den Approximationskoeffizienten der elektrischen Leistung wies implizit auf die unterschiedlichen Taktungen der Komponenten hin. Der Definitionsbereich für die Thread-Anzahl p musste daher wegen der besonderen Taktung des L3-Caches des *Haswell* Prozessors in zwei disjunkte Teilmengen aufgeteilt werden ($p \in (1, 2)$ und $p \in (3, 4, ..., 12)$), um für jede Teilmenge die eigenen Approximationskoeffizienten zu finden. Durch eine hohe Frequenz des L3-Caches wird die elektrische Leistung des Prozessors wesentlich erhöht. Es war außerdem deutlich zu erkennen, dass der Durchsatz des L3-Caches des *Haswell* Prozessors unter der maximalen Last nicht perfekt über die CPU-Frequenz skaliert und bei der höheren CPU-Frequenz deutlich degradiert ($\tilde{\gamma}_{L3} = 0.88$).
Im vierten Fall wurde die Kernel-Operation *Add* für den Fall des Haltens der Daten im Hauptspeicher ausgeführt. Hier war deutlich zu erkennen, dass die statischen Anteile der Approximation stark angestiegen waren. Außerdem zeigten die beiden Prozessoren eine schlechte Energieeffizienz, falls diese trotz des erreichten Sättigungsbereiches noch höher getaktet wurden.
Zusätzlich zur Aufteilung des Thread-Anzahl-Definitionsbereiches musste man auch den Definitionsbereich für die CPU-Frequenz aufteilen. Der Grund hierfür ist die Arbeitsweise der Speichermodule und deren Verbindung mit dem Prozessor. Daher

ist es ersichtlich, dass die Approximation der Rechenleistung im Falle des Haltens der Daten im Hauptspeicher einen anderen Ansatz braucht, bei dem die komplexe Speicheranbindung abgebildet wird. Ein weitergehender Ansatz dazu wurde im folgenden Teil der Arbeit verfolgt.
Die Tabellen Tab. 7.1 und Tab. 7.2 fassen die Koeffizienten der Approximation der elektrischen Leistung in den verschiedenen Fällen zusammen. Es ist wünschenswert, die beschriebenen Modelle zur Approximation der Rechen- und der elektrischen Leistung zu erweitern, so dass die Zusammenhänge zwischen der Kernel-Operation, der Prozessorarchitektur und den Koeffizienten der Approximation genau erklärt werden können. Als Zwischenschritt zum Erreichen dieses Zieles wurde die Architektur eines Prozessors in Details in Kapitel 5 betrachtet und ein von mir entwickeltes „Data-Transfer-Memory“ Modell (DTM) für die Rechenleistung, insbesondere im Falle des Haltens der Daten im Hauptspeicher, vorgestellt.

Tab. 7.1.: *Ivy Bridge*: Koeffizienten der Approximation der elektrischen Leistung während der Ausführung der Kernel-Operation *Add*:
$P_{II}(f,p)_l = (\tilde{\alpha}_{0,0,l} + \tilde{\alpha}_{0,1,l} \times p) + (\tilde{\alpha}_{1,0,l} + \tilde{\alpha}_{1,1,l} \times p) \times f^{\tilde{\lambda}_l}$;

l	p	$\tilde{\alpha}_{0,0,l}$	$\tilde{\alpha}_{0,1,l}$	$\tilde{\alpha}_{1,0,l}$	$\tilde{\alpha}_{1,1,l}$	$\tilde{\lambda}_l$	**Bemerkung**
L1	1-10	21.04	1.099	0.9676	0.4969	2.22	$\tilde{\lambda} > 2$; El. Leistung steigt rasant mit f_{CPU}
L2	1-10	20.77	1.170	0.8470	0.4707	2.26	$\sim$ L1, obwohl mehr Komponenten aktiv sind
L3	1-10	20.67	1.237	1.212	0.6166	2.11	$\sim$ L1 und L2; $f_{CPU} \cong f_{UNCORE}$;
MEM	1,2	21.59	2.282	1.569	1.168	1.81	$\tilde{\lambda} < 2$, $\tilde{\alpha}_{1,0-1,MEM}$ ist höher als bei L1-L3;
MEM	3-10	34.13	0.7224	1.214	0.5291	2.11	$\sim$ L3 aber $\tilde{\alpha}_{0,0,MEM} >> \tilde{\alpha}_{0,0,L3}$;

Tab. 7.2.: *Haswell*: Approximation der elektrischen Leistung während der Ausführung der Kernel-Operation *Add*:
$P_{II}(f,p)_l = (\tilde{\alpha}_{0,0,l} + \tilde{\alpha}_{0,1,l} \times p) + (\tilde{\alpha}_{1,0,l} + \tilde{\alpha}_{1,1,l} \times p) \times f^{\tilde{\lambda}_l}$;

l	p	$\tilde{\alpha}_{0,0,l}$	$\tilde{\alpha}_{0,1,l}$	$\tilde{\alpha}_{1,0,l}$	$\tilde{\alpha}_{1,1,l}$	$\tilde{\lambda}_l$	**Bemerkung**
L1	1-,12	27.51	1.478	0.7829	0.3911	2.66	$\tilde{\lambda} > 2$; El. Leistung steigt rasant mit f_{CPU};
L2	1-,12	27.55	0.7763	0.9583	0.7446	2.12	$\approx$ L1;
L3	1,2	25.32	1.504	1.8	0.9675	1.91	$\tilde{\lambda} < 2$; $f_{CPU} \cong f_{UNCORE}$;
L3	3-12	21.33	2.214	7.744	1.236	1.41	$f_{CPU} \ncong f_{UNCORE}$; Großer dyn. Anteil $\tilde{\alpha}_{1,0,l}$;
MEM	1,2	25.47	8.262	0.4942	2.396	1.91	$\approx L3_{p\in(1,2)}$; Großer stat. Anteil $\tilde{\alpha}_{0,1,l}$;
MEM	3-12	65.52	2.021	2.021	0.3253	2.11	$\approx L3_{p>2}$; $\tilde{\alpha}_{0,0,MEM} >> \tilde{\alpha}_{0,0,L3}$;

7.4. Modellierung der Kernel-Ausführung

Im Kapitel 5 „Modellierung der Kernel-Ausführung“ wurden zwei Modelle erläutert, die sich mit der Modellierung von Von-Neumann-Prozessoren beschäftigen. Am Beispiel des Kernels *Add* wurde gezeigt, wie die beiden Modelle, ECM und von das von mir entwickelte DTM-Modell, funktionieren und welche Vor- und Nachteile diese haben.
Trotz seiner Einfachheit erklärt das ECM-Modell (siehe Gl. (7.2)) die wichtigsten Eigenschaften eines Prozessors, solche wie Pipeline, Hierarchiestufen des Speichers und die Überlappungsmöglichkeit. Das Modell zeigt eine hohe Genauigkeit für die Rechenleistung vieler Kernel-Operationen. Erst wenn Eigenschaften des Prozessors in den Vordergrund treten, die nicht im Modell berücksichtigt werden können, zeigen sich große Abweichungen zu den Messergebnissen.

$$
\begin{aligned}
T^{ECM} &= \max(T^{OL}, T^{nOL} + T_{L1\leftrightarrow L2} + T_{L2\leftrightarrow L3} + T_{L3\leftrightarrow MEM});\\
\\
T^{ECM} &\text{ - } \textit{ECM-Ausführungszeit einer Kernel-Operation mit einem Thread;}\\
T^{OL} &\text{ - } \textit{Ausführungszeit in ALU mit der Überlappung zum Datentransfer;}\\
T^{nOL} &\text{ - } \textit{Ausführungszeit in ALU ohne Überlappung zum Datentransfer;}\\
T_{L1\leftrightarrow L2} &\text{ - } \textit{Übertragungszeit der Daten zwischen L1 und L2;}\\
T_{L2\leftrightarrow L3} &\text{ - } \textit{Übertragungszeit der Daten zwischen L2 und L3;}\\
T_{L2\leftrightarrow L3} &\text{ - } \textit{Übertragungszeit der Daten zwischen L3 und MEM;}
\end{aligned}
\tag{7.2}
$$

In den betrachteten Beispielen können die Prozessoren die Frequenz separat für die Prozessorkerne, den L3-Cache und die Speichermodule ändern. Wie gezeigt wurde, verursacht dies und die limitierte Speicherbandbreite im Falle des Haltens der Daten im Hauptspeicher eine große Abweichung zwischen dem ECM-Modell und der beobachteten Rechenleistung. Aus diesem Grund habe ich das ECM-Modell erweitert und dies „Data-Transfer-Memory“ Modell (DTM) genannt.
Das DTM-Modell (siehe 7.3) basiert auf der Modellierung der Datenströme zwischen den verschiedenen Einheiten des Prozessors, die unterschiedlich getaktet sein können. Das wichtigste Merkmal eines ECM-Modells ist, dass der Datentransfer zwischen den verschiedenen Hierarchiestufen des Speichers nicht überlappend angenommen wird. Nur die Ausführung der arithmetischen Operationen in der Recheneinheit (ALU) kann zum Datentransfer überlappend erfolgen. Ein DTM-Modell dagegen erlaubt eine Überlappung des Datentransfers innerhalb eines Prozessorkerns, während die Daten zwischen dem IMC und dem Speicher transportiert werden. Dies ermöglicht das Verhalten des Prozessors im Falle des Haltens der Daten im Hauptspeicher zu

beschreiben, wenn sich dabei die CPU-Frequenz und die Anzahl der aktiven Prozessorkerne ändern. Daher wurde das Modell mit dem Ringbus und dem Speicherkontrolleur erweitert.

$$
\begin{aligned}
&T^{DTM} && = \max(T^{OL}_{L2\leftrightarrow AVX}, T^{nOL}_{L2\leftrightarrow AVX} + T_{L3\leftrightarrow L2} + T_{L3\leftrightarrow IMC} + \\
& && \quad + T_{MEM\leftrightarrow IMC});\\
&T^{DTM} && \text{- } \textit{DTM-Ausführungszeit einer Kernel-Operation;}\\
&T^{OL}_{L2\leftrightarrow AVX} && \text{- } \textit{Ausführungszeit in einem Prozessorkern (ALU,L1,L2)}\\
& && \quad \textit{mit Überlappung zum Datentransfer zwischen den}\\
& && \quad \textit{Uncore-Komponenten (Hauptspeicher, IMC und L3);}\\
&T^{nOL}_{L2\leftrightarrow AVX} && \text{- } \textit{Ausführungszeit in einem Prozessorkern ohne Überlappung;}\\
&T_{A\leftrightarrow B} && \text{- } \textit{Übertragungszeit der Daten zwischen den Komponenten}\\
& && \quad \textit{A und B ohne Überlappung;}
\end{aligned}
\tag{7.3}
$$

Die Resultate zeigen eine gute Übereinstimmung mit den Messergebnissen, was eine begründete Erklärung für das Verhalten der betrachteten Prozessoren bedeutet. Der Leser konnte sehen, dass nicht alle Merkmale des Prozessors berücksichtigt waren, so dass eine erkennbare Abweichung in der modellierten Bandbreite und der tatsächlichen Bandbreite der Kernel-Operation auf dem Prozessor *Ivy Bridge* aufgetreten ist. Als eine vielversprechende Lösung zur Korrektur der Ergebnisse sehe ich eine weitere Modellierung des Ringbusses und der Speichermodule.
Das Modell kann prinzipiell für viele andere Prozessoren und nicht nur für die hier betrachteten angewendet werden, weil deren Architektur oft aus den vergleichbaren Komponenten weiterentwickelt wird.

7.5. Energieverbrauch eines Höchstleistungsrechners

In Kapitel 6 wurde gezeigt, wie ein Messsystem im Hochleistungsrechner „Hazel Hen“ funktioniert und wie weit dies für die Analyse der Energieeffizienz angewendet werden kann. Die Analyse ist durch eine niedrige zeitliche Auflösung des Messsystems und ohne die Möglichkeit die wichtigsten Hardwarekomponenten separat zu messen stark erschwert. Obwohl die Differenz zwischen der elektrischen Leistung der Prozessoren mit dem Hauptspeicher und den anderen Hardware-Komponenten mit einer hohen

Genauigkeit linear interpoliert werden kann, beeinflusst derer elektrische Leistung die Approximation.
Eine andere wichtige Beobachtung ist, dass sowohl die Rechenleistung als auch die elektrische Leistung einer komplexeren Kernel-Operation *PETsC-CG* ein ähnliches Verhalten zu den Stream-Operationen (z.B. *Add*) zeigen. Für den betrachten Fall hat eine hohe CPU-Frequenz bei der Berechnung auf allen Prozessorkernen des Rechenknotens keine oder sehr geringe zusätzliche Rechenleistung gebracht, während sich die elektrische Leistung stark erhöht. Mit der niedrigen CPU-Frequenz verkleinert sich der statische Anteil der elektrischen Leistung um mehr als 60 W per Rechenknoten (siehe Gl. (6.5)). Dies ist $\sim 20\%$ des gesamten Verbrauchs eines Rechenknotens.

Literaturverzeichnis

[1] Intel Corparation (2004). *Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor White paper March 2014*. Accessed: 2014-11-26. URL: ftp://download.intel.com/design/network/papers/30117401.pdf.

[2] ADDI-DATA. *Technisches Referenzhandbuch; APCIe-3021, APCIe-3121 und APCIe-3521 Multifunktionskarte, galvanisch getrennt*. Ausgabe: 02.07-11/2013. URL: http://www.addi-data.de.

[3] Inc. Advanced Micro Devices. *Technical Description AMD FirePro S1000 High-Density, High-Performance Server Graphics*. Ausgabe: 02.07-11/2013. URL: https://www.amd.com/documents/FirePro_S10000_Data_Sheet.pdf.

[4] *AMD Opteron™ Processor Quick Reference Guide*. Accessed: 2018-10-10. URL: https://www.amd.com/Documents/Opteron_6000_QRG.pdf.

[5] Satish Balay u. a. *PETSc Users Manual*. Techn. Ber. ANL-95/11 - Revision 3.8. Argonne National Laboratory, 2017. URL: http://www.mcs.anl.gov/petsc.

[6] Klaus Beuth und Olaf Beuth. *Elementare Elektronik mit Grundlagen der Elektrotechnik*. 5. Auflage. Würzburg: Vogel, 1997, 378 S. ISBN: 3-8023-1524-6.

[7] Anantha P. Chandrakasan, Samuel Sheng und Robert W. Brodersen. „Low Power CMOS Digital Design“. In: *IEEE JOURNAL OF SOLID STATE CIRCUITS* 27 (1995), S. 473–484.

[8] Samsung Electronics Co. „Datasheet 288pin Registered DIMM based on 4Gb D-die (Model: M393A2G40DB0-CPB)“. In: Accessed: 2018-10-28. Juli 2016. Kap. Input clock frequency change, S. 36. URL: https://gzhls.at/blob/ldb/5/c/4/c/91e6f1069eb7490a6261e87d6b17856c1b91.pdf.

[9] Samsung Electronics Co. „Datasheet 288pin Registered DIMM based on 4Gb D-die (Model: M393A2G40DB0-CPB)“. In: Accessed: 2018-10-28. Juli 2016. URL: https://gzhls.at/blob/ldb/5/c/4/c/91e6f1069eb7490a6261e87d6b17856c1b91.pdf.

[10] European Commission. *Dynamic Resource Allocation in Embedded and High-Performance Computing - DreamCloud*. Sep. 2013. URL: http://www.dreamcloud-project.org/.

[11] European Commission. *Execution Models for Energy-Efficient Computing Systems - EXCESS*. Sep. 2013. URL: http://excess-project.eu/.

[12] Intel Corporation. *ATX12V Power Supply Design Guide*. Accessed: 2018-10-28. März 2005. URL: http://www.ieca-inc.com/images/ATX12V_PSDG2.0_Ratified.pdf.

[13] Intel Corporation. „Intel 64 and IA-32 Architectures Optimization Reference Manual“. In: Juni 2016. Kap. Execution Engine, S. 2–15. URL: http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html.

[14] Intel Corporation. „Intel 64 and IA-32 Architectures Optimization Reference Manual“. In: Juni 2016. Kap. Issue Ports and Execution Units, S. 2–38. URL: http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html.

[15] Intel Corporation. „Intel 64 and IA-32 Architectures Optimization Reference Manual“. In: Juni 2016. Kap. Cache and Memory Subsystem, S. 2–17. URL: http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html.

[16] Intel Corporation. „Intel 64 and IA-32 Architectures Optimization Reference Manual“. In: Juni 2016. Kap. Using LEA, S. 3–22. URL: http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html.

[17] Intel Corporation. „Intel 64 and IA-32 Architectures Optimization Reference Manual“. In: Juni 2016. Kap. The Haswell Microarchitecture, S. 2–7. URL: http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html.

[18] Intel Corporation. „Intel 64 and IA-32 Architectures Optimization Reference Manual“. In: Juni 2016. Kap. Load and Store Operation Overview, S. 2–31. URL: http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html.

[19] Intel Corporation. „Intel 64 and IA-32 Architectures Optimization Reference Manual“. In: Juni 2016. Kap. Ring interconnect and Last Level Cache, S. 2–29. URL: http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html.

[20] Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual - Volume 1*. Sep. 2015. Kap. Caching of Temporal vs. Non-Temporal Data, S. 10–12. URL: https://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-1-manual.html.

[21] Intel Corporation. *Intel Announces New 22nm 3D Tri-gate Transistors*. Apr. 2011. URL: http://www.intel.com/content/www/us/en/silicon-innovations/standards-22nm-3d-tri-gate-transistors-presentation.html.

[22] Intel Corporation. *Intel Xeon 64 Processor E5-1600 and E5-2600 v3 Product Families,Volume 1 of 2, Electrical*. 330783-001. Sep. 2014.

[23] Intel Corporation. *Intel Xeon 64 Processor E5-1600 and E5-2600 v3 Product Families,Volume 2: Registers, Electrical*. 330784-003. Juni 2015.

[24] Intel Corporation. *Intel Xeon 64 Processor E5-1600/E52600/ E5-2600 v2 Product Families, Volume 1*. 329187-003. März 2014.

[25] Intel Corporation. *Intel Xeon Prozessor E5-1600 and E5-2600 v3 Product Families Specification Update*. Accessed: 2015-09-01. Aug. 2015. URL: http://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/xeon-e5-v3-spec-update.pdf.

[26] Intel Corporation. *VR12/IMVP7 Pulse Width Modulation (PWM) Specification*. 397113. Dez. 2009.

[27] Björn Dick. *Mündliche Disskusionen über die Synchronisierungsmöglichkeit der verteilten Uhren mit Björn Dick, „Numerical Methods & Libraries, High Performance Computing Center Stuttgart (HLRS)*. 2013.

[28] Daniel A. Ellsworth u. a. „A Unified Platform for Exploring Power Management Strategies“. In: *4th International Workshop on Energy Efficient Supercomputing, E2SC@SC 2016, Salt Lake City, UT, USA, November 14, 2016*. 2016, S. 24–30. DOI: 10.1109/E2SC.2016.009. URL: https://doi.org/10.1109/E2SC.2016.009.

[29] Free Software Foundation, Inc. *A GNU Manual*. Accessed: 2015-08-31. URL: https://gcc.gnu.org/onlinedocs/gcc/.

[30] Daniel Hackenberg u. a. „An energy efficiency feature survey of the intel haswell processor“. In: *High-Performance, Power-Aware Computing (HP-PAC)*. 2015.

[31] Daniel Hackenberg u. a. „Power measurement techniques on standard compute nodes: A quantitative comparison.“ In: *ISPASS*. IEEE, 2013, S. 194–204. URL: http://dblp.uni-trier.de/db/conf/ispass/ispass2013.html#HackenbergISMSN13.

[32] Georg Hager u. a. „Exploring performance and power properties of modern multicore chips via simple machine models“. In: *CoRR* abs/1208.2908 (2012). URL: http://arxiv.org/abs/1208.2908.

[33] Julian Hammer u. a. „Automatic Loop Kernel Analysis and Performance Modeling With Kerncraft“. In: *CoRR* abs/1509.03778 (2015). URL: http://arxiv.org/abs/1509.03778.

[34] John L. Henning. „SPEC CPU2000: Measuring CPU Performance in the New Millennium“. In: *Computer* 33.7 (Juli 2000), S. 28–35. ISSN: 0018-9162. DOI: 10.1109/2.869367. URL: http://dx.doi.org/10.1109/2.869367.

[35] Johannes Hofmann, Jan Eitzinger und Dietmar Fey. „Execution-Cache-Memory Performance Model: Introduction and Validation“. In: *CoRR* abs/1509.03118 (2015). URL: http://arxiv.org/abs/1509.03118.

[36] Dennis Hoppe, Yosandra Sandoval und Michael Gienger. „ATOM: A Near-Real Time Monitoring Framework for HPC and Embedded Systems.“ In: Donostia-San Sebastián, Spain, Juni 2015. URL: http://www.ena-hpc.org/2014/pdf/intel.pdf.

[37] *HP™ Memory 4 GB DDR3, Technische Beschreibung*. Accessed: 2015-01-18. URL: http://h30094.www3.hp.com/product/sku/10715899.

[38] SK hynix Inc. *DDR3 SDRAM Unbuffered DIMMs Based on 2Gb E-Die (Model: HMT351U7EFR8C-RD)*. Accessed: 2017-11-01. Juli 213. URL: https://www.skhynix.com/product/filedata/fileDownload.do?seq=3750.

[39] *Intel™ Core Processor i7-860, Technische Beschreibung*. Accessed: 2014-11-26. URL: https://ark.intel.com/De/products/41316/Intel-Core-i7-860-Processor-8M-Cache-2_80-GHz.

[40] *Intel™ Xeon Processor E5-2680 v3, Technische Beschreibung*. Accessed: 2015-06-30. URL: https://ark.intel.com/products/81908/Intel-Xeon-Processor-E5-2680-v3-30M-Cache-2_50-GHz.

[41] *Intel™ Xeon Processor E5-2690 v2, Technische Beschreibung*. Accessed: 2014-11-26. URL: http://ark.intel.com/products/75279/Intel-Xeon-Processor-E5-2690-v2-25M-Cache-3_00-GHz.

[42] *Intel ™ Xeon Processor E5-2699 v3, Technische Beschreibung*. Accessed: 2014-11-26. URL: https://ark.intel.com/products/81061/Intel-Xeon-Processor-E5-2699-v3-45M-Cache-2_30-GHz.

[43] Thomas Willhalm (Intel), Roman Dementiev (Intel) und Patrick Fay (Intel). *Intel Performance Counter Monitor - A better way to measure CPU utilization*. Accessed: 2015-04-07. Aug. 2012. URL: http://www.intel.com/software/pcm.

[44] JEDEC. „DDR4 SDRAM STANDARD Specification (Revision of JESD79-4)“. In: (Nov. 2013). URL: https://www.jedec.org/standards-documents/docs/jesd-79-3d.

[45] Dmitry Khabi und Uwe Küster. „Power Consumption of Kernel Operations“. In: *Sustained Simulation Performance 2013*. Springer, Juni 2013, S. 27–45.

[46] Chulwoo Kim, Hyun-Woo Lee und Junyoung Song. *High-Bandwidth Memory Interface*. Springer Briefs in Electrical and Computer Engineering. Springer, 2014. ISBN: 978-3-319-02380-9. DOI: 10.1007/978-3-319-02381-6. URL: http://dx.doi.org/10.1007/978-3-319-02381-6.

[47] Alexander Kipp u. a. „Approach towards an Energy-Aware and Energy-Efficient High Performance Computing Environment“. In: *Proceedings of the IEEE 7th International Conference on Intelligent Computer Communication and Processing*. Cluj-Napoca, Romania, Okt. 2011.

[48] Uwe Küster. *Mündliche Disskusionen über ein Verfahren zur Interpolation der elektischen Leistung eines Prozessors mit Uwe Küster, „Numerical Methods & Libraries , High Performance Computing Center Stuttgart (HLRS)"*.

[49] Etienne Le Sueur und Gernot Heiser. „Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns“. In: *Proceedings of the 2010 Workshop on Power Aware Computing and Systems (HotPower'10)*. Vancouver, Canada, Okt. 2010. URL: https://www.usenix.org/legacy/events/hotpower/tech/full_papers/LeSueur.pdf.

[50] Kang Lee und John Eidson. „IEEE-1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems“. In: *In 34 th Annual Precise Time and Time Interval (PTTI) Meeting*. 2002, S. 98–105.

[51] *Leistungswiderstand FPA100 2.2 kOhm 100 W 5%, Technische Beschreibung.* Accessed: 2015-03-5. URL: http://www.mercateo.at/p/115A-16057225/Leistungswiderstand_2_2_kOhm_100_W_5_FPA100_2K2_J.html.

[52] Steven J. Martin u. a. „Cray XC40 Power Monitoring and Control for Knights Landing“. In: Cray User Group. URL: https://cug.org/proceedings/cug2016_proceedings/includes/files/pap112s2-file1.pdf.

[53] John D. McCalpin. *STREAM: Sustainable Memory Bandwidth in High Performance Computers.* Techn. Ber. A continually updated technical report. Charlottesville, Virginia: University of Virginia, 1991-2007. URL: http://www.cs.virginia.edu/stream/.

[54] *Mellanox™ FDR Single/Dual-Port InfiniBand Host Channel Adapter Card, Technische Beschreibung.* Accessed: 2015-01-18. URL: http://www.mellanox.com/related-docs/prod_adapter_cards/PB_Connect-IB.pdf.

[55] *Messwiderstand 0.01 Ohm 10 W (L x B x H) 22 x 4 x 17 mm Isabellenhütte PBV 0,01, Technische Beschreibung.* Accessed: 2015-01-19. URL: http://www.produktinfo.conrad.com/datenblaetter/425000-449999/447323-da-01-en-Praezisionswiderstand_PBV.pdf.

[56] *MLP/DFN-6 to DIP-6 SMT Adapter (0.5 mm pitch, 2 x 2 mm body), Technische Beschreibung.* Accessed: 2015-01-18. URL: http://www.proto-advantage.com/store/product_info.php?products_id=2200053.

[57] Daniel Molka u. a. „Cache Coherence Protocol and Memory Performance of the Intel Haswell-EP Architecture“. In: *2015 44th International Conference on Parallel Processing* (2015), S. 739–748.

[58] Jürgen Nehmer. *Systemsoftware - Grundlagen moderner Betriebssysteme.* 2. aktualis. Aufl. Köln: Dpunkt-Verlag, 2001. ISBN: 978-3-898-64115-9.

[59] *NVIDIA ™ TeslaK40c Graphik-Accelerator, Technische Beschreibung.* Accessed: 2015-01-18. URL: http://www.nvidia.com/object/tesla-servers.html.

[60] *NVIDIA™ TeslaK80 GPU Accelerator, Board Specification.* Accessed: 2015-08-04. URL: http://images.nvidia.com/content/pdf/kepler/Tesla-K80-BoardSpec-07317-001-v05.pdf.

[61] Randall C. O'Reilly. *A Family of Large-Stencil Discrete Laplacian Approximations in Three Dimensions.* University of Colorado Boulder 345 UCB. URL: ftp://grey.colorado.edu/pub/oreilly/misc/disc_lapl.3.pdf.

[62] Ursula Paul. *Mündliche Disskusionen über die Kühlung von Hazel Hen mit Ursula Paul, Infrastructure, High Performance Computing Center Stuttgart (HLRS)*. 2018.

[63] *Präzisions-Stromwandler TZ 77 mit Linearität 0,2% (0,25 A - 20 A), Technische Beschreibung*. Accessed: 2015-01-18. URL: http://www.produktinfo.conrad.com/datenblaetter/400000-424999/415707-da-01-en-Praezisions_Stromwandler.pdf.

[64] William H. Press u. a. *Numerical Recipes in FORTRAN; The Art of Scientific Computing*. 2nd. New York, NY, USA: Cambridge University Press, 1993. ISBN: 0521437164.

[65] Valter Quercioli. *Pulse Width Modulated (PWM) Power Supplies*. Amsterdam: ELSEVIER SCIENCE PUBLISHERS B.V., 1993. ISBN: 0-444-89790-9 (Vol. 45).

[66] Ryuichi Sakamoto u. a. „Production Hardware Overprovisioning: Real-World Performance Optimization Using an Extensible Power-Aware Resource Management Framework“. In: *2017 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2017, Orlando, FL, USA, May 29 - June 2, 2017*. 2017, S. 957–966. DOI: 10.1109/IPDPS.2017.107. URL: https://doi.org/10.1109/IPDPS.2017.107.

[67] *Samsung™ DDR4 SDRAM Memory, Product Guide*. Accessed: 2015-06-30. URL: http://www.samsung.com/global/business/semiconductor/file/product/DDR4_Product_guide_Dec13.pdf.

[68] Ralf Schneider. *Analyse kontinuumsmechanischer, anisotroper Materialparameter mikrostrukturierter Volumina mit Hilfe direkter mechanischer Simulation. Dissertation.* HLRS. Höchstleistungsrechenzentrum, Universität Stuttgart, 2016. URL: https://books.google.de/books?id=XLh5AQAACAAJ.

[69] Andrey Semin. „Intel technologies, tools and techniques for power and energy efficiency analysis“. In: 2014. URL: http://www.ena-hpc.org/2014/pdf/intel.pdf.

[70] Anand Lal Shimpi. „Intel's Haswell Architecture Analyzed: Building a New PC ans a New Intel“. In: *AnandTech* (2012). URL: http://www.anandtech.com/show/6355/intels-haswell-architecture/.

[71] *Technische Beschreibung CRAY XC40 (Hazel Hen)*. Accessed: 2017-12-12. URL: http://www.hlrs.de/de/systems/cray-xc40-hazel-hen/.

[72] *Thick Film Chip Resistor SMD WIDERSTAND 0603 RC 1608 1% 0,1W), Technische Beschreibung.* Accessed: 2015-01-18. URL: http://files.voelkner.de/425000-449999/433081-da-01-en-SMD_WIDERSTAND_0603_RC_1608_10K0_1__0_1W.pdf.

[73] My Ton, Brian Fortenbery und William Tschudi. *DC Power for Improved Data Center Efficiency.* Accessed: 2018-2-12. März 2008. URL: http://www.chip2grid.com/docs/DCDemoFinalReport.pdf.

[74] Alistair Hart (CRAY UK) u. a. *D2.6.3 - Power measurement across algorithms.* Accessed: 2017-12-12. Feb. 2014. URL: https://www.cresta-project.eu/images/docs/deliverables/year_3/D2.6.3_Power_measurement_across_algorithms.pdf.

[75] *Unitronic LiYCY 2x0.14, Technische Beschreibung.* Accessed: 2015-01-18. URL: http://edgecdn.lappgroup.com/fileadmin/documents/technische_doku/datenblaetter/unitronic/DB0034302DE.pdf.

[76] C. Vömel und A. Meister. *Numerik linearer Gleichungssysteme: Eine Einführung in moderne Verfahren. Mit MATLAB-Implementierung von C. Vömel.* Vieweg+Teubner Verlag, 2007. ISBN: 9783834804310. URL: https://books.google.de/books?id=80Jzr0aN8RYC.

[77] Markus Wittmann u. a. „An analysis of energy-optimized lattice-Boltzmann CFD simulations from the chip to the highly parallel level“. In: *CoRR* abs/1304.7664 (2013). URL: http://arxiv.org/abs/1304.7664.

[78] Л. З. Румшиский. *Справочное пособие: Математическая обработка результатов экспериментов.* Москва: Издательство НАУКА, УДК 518, 1971.

[79] П. Л. Калантаров А. А. Цейтлин. *Расчет индуктивностей: Справочная книга.* 3. Издание. Ленинград: ЭНЕРГОАТОМИЗДАТ Ленинградское отделение, 1986.

A. Testcluster

I. Hardwarekomponente

In diesem Abschnitt wird der Kern des Testclusters beschrieben, dessen Hardwarekomponenten im weiteren Verlauf bezüglich der Energieeffizienz geprüft werden.[1] In Abb. A.1 ist das prinzipielle Schema des Testclusters und sein Foto dargestellt. Der Cluster besteht aus drei Rechenknoten, einem Embedded-Board "Myriad2 ", einem Messsystem, einem Front-End und einem NAS-Server. Die Cluster-Komponenten sind über zwei Switches verbunden: *Ethernet-Switch* 1000 Mbit und *Infiniband-Switch* 56 Gbit (FDR). Der Hochgeschwindigkeits-Switch *Infiniband* verbindet drei Rechenknoten, die für die Ausführung der parallelen Algorithmen vorgesehen sind.

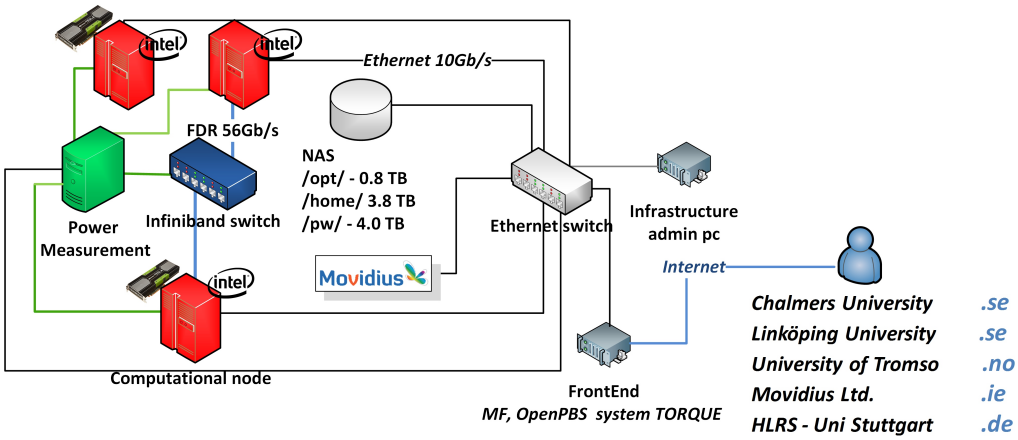

Abb. A.1.: Testcluster

Auf allen Rechenknoten und Servern wird das Betriebssystem "Scientific Linux release 6.6 (Carbon)" mit dem OS-Kernel 2.6.32 verwendet.

[1] Das Cluster wird laufend mit den steigenden Anforderungen erweitert.

I.I. Technische Beschreibung des Testclusters

Das Testcluster wurde in einem HLRS-Serverraum installiert. Die Umgebungstemperatur wird mit einer Klimaanlage auf $\sim$ 25 °C konstant gehalten. Das Cluster besteht aus folgenden Komponenten:

- **node01 und node02**:
  1 × Intel Server Chassis P4308XXMHGC;
  1 × Intel Server Board S2600COE;
  2 × Intel Xeon CPU: E5-2690 v2 - 10 cores; 25MB L3 smart cache; 3.0 GHz;
  4 mem. ch. DDR3 1866 MHz = 59.7 GB/s[41];
  2 × Intel Heat sink <200b>AUPSRCBT;
  8 × Hynix Memory 4 GB DDR3 HMT351U7EFR8C-RD - 1866 MHz PC3-14900
  CL13 - ungepuffert - Dual Rank - ECC[37];
  1 × GPU im node01: TeslaK40c - GPU Clock: 745 MHz; Shading Units: 2880;
  GDDR5 12288 MB; 288 GB/s; PCIe3.0 8GT/s x16[59];
  1 × GPU im node02: AMD FirePro S10000;
  GDDR5 12288 MB; 480 GB/s; PCIe3.0 8GT/s x16[3];
  1 × Connect-IB HCA, Mellanox MCB193A-FCAT; single-port QSFP;
  FDR IB (56Gb/s); PCIe3.0 8GT/s x16[54];
  1 × Hard disk: 500GB WD5003AZEX Black; 1 × SSD: 128GB Vertex450;
  1 × Schaltnetzteil: node01: *DPS-750XB A REV02F*; node02: *EartWatts EA-650*;

- **node03**:
  1 × Chenbro 4U 17.5"Compact Industrial Server Chassis RM42300;
  1 × Gigabyte® Server Board - MD70-HB0 (Rev. 1,2);
  2 × Intel Xeon CPU: E5-2680 v3 - 12 cores; 30MB L3 smart cache; 2.5 GHz;
  4 mem. ch. DDR4 2133 MHz = 68 GB/s[40];
  2 × Enermax® Heat sink ETS-T40-W;
  8 × SAMSUNG DRAM 16GB Samsung DDR4-M393A2G40DB0-CPB - 2133 MHz
  CL15 - Registered DIMM - Dual Rank - ECC[67];
  1 × GPU: TeslaK80 - 2x Kepler GK210; GPU Clock: 560-875 MHz; Shading Units: 4992;
  GDDR5 24 GB; 480 GB/sec; PCIe3.0 8GT/s x16[60];
  1 × Connect-IB HCA, Mellanox MCB193A-FCAT, single-port QSFP,
  FDR IB (56Gb/s), PCIe3.0 8GT/s x16[54];
  1 × Intel® Ethernet Server Adapter I350-T2 - PCI Express 2.1 x4 Low Profile - 1000Base-T x 2;
  1 × Hard disk: 500GB WD5003AZEX Black; 1 × SSD: 240GB Vertex460A;
  1 × Schaltnetzteil: *EartWatts EA-650*;

- **Infiniband-Switch**:
  1 × FDR-Infiniband-Switch Mellanox MSX6018F-1SFS, 18 FDR (56Gb/s) Anschlüsse im 1U Switch;

- **Ethernet-Switch**:
  1 × Cisco Catalyst 3750X-24T-S, 24 10/100/1000 Anschlüsse im 1U Switch;

- **Frontend**:
  2 × Intel® Xeon® E5-2609 v2, 2.50 GHz, 4-Core, 10MB Cache,
  8 × 8 GB DDR3;
  4 × 1TB SATA3-HDD Seagate Constellation ES.3;

- **NAS**:
  1 × Intel® Xeon® E3-1220 v3 3100MHz 8MB Cache 4Core;
  2 × 8GB DDR3;
  2 × 1TB WD Caviar Red NAS HDD 64MB;
  3 × 2TB WD Caviar Red NAS HDD 64MB;

- **Info structure PC**:
  1 × Supermicro® X10SBA;

- **Messystem**:

  1 × Addi-System mit 4 A/D-Wandler APCIE-3021-16[2];

I.II. Messgruppen des Testclusters

Die Messgruppen im Testcluster sind in Tab. A.1 , Tab. A.2 und Tab. A.3 zusammengefasst. Eine Messgruppe besteht aus den Hardwarekomponenten, die an die gleiche Stromquelle angeschlossen sind. Sowohl die Spannung als auch die Stromstärke werden gemessen. Allerdings, wird die Spannung für die Gruppen $ATX5V$, $ATX12V_{node02}$, $ATX12V_{node03}$ und $COOLER$ wegen der Knappheit der Messkanäle und dem relativ kleinen Anteil vom gesamten Stromverbrauch nicht vermessen.[2] Zum Beispiel wird für die Messgruppe $ATX5V$ die Spannung V_i als 5.0 V angenommen.
Die Gruppe „CPU“ besteht aus einem Prozessor und den angeschlossenen Speichermodulen. Der Leistungsverlust der VRMs wird somit mitgemessen. Zusätzlich werden oft die CPU-Lüfter mit dem Strom über den gleichen Stromanschluss wie der Prozessor versorgt. Wenn dieses der Fall ist, wird das Stromsignal eine zusätzliche Komponente mit der Frequenz im Bereich von etwa 100 Hz haben. Diese Frequenz ist proportional zur Rotationsgeschwindigkeit des Lüfters. Mit Hilfe eines einfachen Adapters kann die Stromversorgung auf eine andere Stromquelle umgeschaltet werden.

Tab. A.1.: Messgruppen des Rechenknotens *node01* mit zwei *Ivy Bridge* Prozessoren *Intel Xeon E5-2690 v2* und einem Grafikprozessor (GPU) TeslaK40[59]

Messgruppe	Messstelle	Hardwarekomponenten
CPU1	4-Pin-CPU Stromkabel	1. CPU[41]; 4x4 GB DDR3[37];
CPU2	4-Pin-CPU Stromkabel	2. CPU[41]; 4x4 GB DDR3[37]
GPU1	8-Pin-GPU Stromkabel	GPU TeslaK40[59];
ATX12V	12-Volt ATX-Stromkabel	PCI-e bridge; InfiniBand-Adapter[54]; GPU-Anteil
NODE01	AC-Kaltgerätekabel	Rechenknoten *node01*

[2]Es ist möglich, statt einer Analog-Eingabekarte pro Rechenknoten mehrere Eingabekarten zu einem Rechenknoten zuzuweisen.

Tab. A.2.: Messgruppen des Rechenknotens *node02* mit zwei *Ivy Bridge* Prozessoren *Intel Xeon E5-2690 v2*

Messgruppe	Messstelle	Hardwarekomponenten
CPU1	4-Pin-CPU Kabel	1. CPU[41]; 4x4 GB DDR3[37]
CPU2	4-Pin-CPU Kabel	2. CPU[41]; 4x4 GB DDR3[37]
ATX12V	12-Volt ATX-Kabel	PCI-e bridge; InfiniBand-Adapter[54]
ATX5V	5-Volt ATX-Kabel	5V-Hauptplatinenkomponente
COOLER	12-Volt Fans (optional)	2 CPU and 3 Chassis fans
NODE02	AC-Kaltgerätekabel	Rechenknoten *node02*

Tab. A.3.: Messgruppen des Rechenknotens *node03* mit zwei *Haswell* Prozessoren *Intel Xeon E5-2680 v3*

Messgruppe	Messstelle	Hardwarekomponenten
CPU1	4-Pin-CPU Kabel	1. CPU[40]; 4x16 GB DDR4[67]
CPU2	4-Pin-CPU Kabel	2. CPU[40]; 4x16 GB DDR4[67]
GPU1	8-Pin-GPU Stromkabel	GPU TeslaK40[60];
ATX12V	12-Volt ATX-Kabel	PCI-e bridge; InfiniBand-Adapter[54]; GPU-Anteil
COOLER	12-Volt Lüfter	zwei CPU- und drei Gehäuselüfter
NODE03	AC-Kaltgerätekabel	Rechenknoten *node03*

I.III. Schaltnetzteile des Testclusters

Die Rechenknoten des Testclusters verwenden unterschiedlichen Typen der Schaltnetzteile:

- **Intel DPS-750XB A REV02F**: Das ist ein Server-Schaltnetzteil mit einer nominalen Leistungsabgabe von 750 Watt. Die maximale Stromstärke für den +12VDC Ausgang beträgt 62.0 A.
- **Antec EartWatts EA-650**: Das ist ein Desktop-Schaltnetzteil mit einer nominalen Leistungsabgabe von 650 Watt. Die summierte maximale Stromstärke für die +12VDC Ausgänge beträgt 48.0 A. Laut Herstellerangaben erreichen die Ripple-/Noise-Spannungen (Peak-Peak) maximal 120 mV. Die 120 mV

wird als eine maximale Ripple-/Noise-Spannung im standardisierten “ATX12V Power Supply Design Guide“[12] von „*formfactors.org*“ spezifiziert.

Ein Schaltnetzteil vom Typ *Intel DPS-750XB* ist im Rechenknoten *node01* installiert. Alle anderen Knoten sind mit den Schaltnetzteilen vom Typ *EartWatts EA-650* ausgestattet.

B. Messeinrichtung

I. A/D-Wandler

Das in dieser Arbeit beschriebene Messsystem verwendet für die zeitliche Aufzeichnung des Spannungs- und Stromverlaufs die 16-Bit-Analog-Eingabekarten *APCIe-3021*[2]. Eine Karte ist mit acht differenziellen Analogsignal-Eingängen ausgestattet. Die totale Durchsatzrate der Karte beträgt $f_{addi} = 100kHz$. Das analoge Signal im Bereich von 0.0 V bis 1.0 V wird vom A/D-Wandler in ein digitales Signal mit einer Stufenfunktion der Stufenbreite (LSB) von $LSB = 2^{-16}$ V gewandelt. Somit wird dem Bereich zwischen zwei benachbarten digitalen Größen ein Wert zugewiesen, was zum Quantisierungsfehler führt.
Der Quantisierungsfehler wird durch die integrale Nichtlinearität (INL) und durch die differenzielle Nichtlinearität (DNL) angegeben. Eine ideale Stufenfunktion besitzt eine konstante Stufenbreite von 1 LSB. Die differenzielle Nichtlinearität gibt die maximale Differenz zwischen idealer und tatsächlicher Stufenbreite. Die maximale Differenz der verwendeten Eingabekarten ist nach [2] gleich $\pm$ 1 LSB.
Durch die unterschiedlichen Stufenbreiten wird die Übertragungsfunktion zwischen dem Analog- und dem Digital-Signal keine gerade Linie. Die Integrale Nichtlinearität (INL) beschreibt die Abweichung zwischen der Übertragungsfunktion und der geraden Linie. Die verwendeten Eingabekarten erweisen die Abweichung typischerweise von $\pm 1/2$ LSB und maximal von ± 2 LSB.
Das Analogsignal wird vor dem A/D Wandler mit einem Tiefpassfilter *RC-Glied* gefiltert. Dadurch werden die Hochfrequenzkomponenten eines Analogsignals vom A/D Wandler nicht erfasst. Die Bandbreite des Tiefpassfilters (-3 dB) beträgt 159 kHz.
Die Eingabekarte kann die aufgezeichneten Messwerte regelmäßig über das DMA-Protokoll in den Speicher des Messsystems schreiben. Danach können diese Daten von mit Sofware bearbeitet werden.
Neben der elektrischen Spannung wird auch die genauere Uhrzeit jeder einzelnen Konvertierung bekannt. In einem Messsystem können mehrere *APCIe-3021* Karten unabhängig voneinander verwendet werden. Die weiteren Details zur Analog-

Eingabekarte kann der Leser in der technischen Anleitung [2] finden. Der Bereich zwischen zwei benachbarten digitalen Größen wird dabei einem Digitalwert zugeordnet und führt zu einem sogenannten Quantisierungsfehler.

II. Strommessung mit einem Messwiderstand

Der Strom wird über die Differenz der elektrischen Potenzialen am Messwiderstand vom Type *PBV R010* ±0.5 %[55] gemessen. Der Messwiderstand ist in einer Reihenschaltung mit dem Verbraucher eingebaut. Der Kalibrierungskoeffizient für die Berechnung der Stromstärke nach Gl. (2.1) ist $c_{shunt} = \frac{1.0}{R} = 100\ \Omega^{-1}$ gleich und ist daher mit der Genauigkeit von ±0.5 % bekannt. Das elektrische Schema der Strommessung ist in Abb. B.1 dargestellt.

Abb. B.1.: **Links:** Elektrisches Schema der Messung am Messwiderstand: R=0.01 Ω - Messwiderstand *PBV R010*; 1 - geschirmtes Kabel; 2 - 12-V Bus-Schiene; 3 - Signalleitungen zu Analogsignal-Eingängen eines A/D-Wandlers. **Rechts:** Elektrisches Schema der Messung am Messwiderstand mit der Berücksichtigung der parasitären reaktiven Komponenten: 5 - parasitäre Induktivität des Messwiderstandes; 6 - parasitäre Induktivität der Signalleitung 7,8 - parasitäre Kapazität von Signalleitung und Messwiderstand;

Nach den Herstellerangaben ist die Induktivität eines Messwiderstandes vom Type *PBV R010* nicht größer als 5.0 nH. Und die parasitäre Kapazität des Messwiderstandes ist nicht größer als 1.0 pF.
Für die Signalleitungen wurde das Kabel *Unitronic LiYCY 2x0.14* [75] verwendet. Die Länge einer Signalleitung ist 1.4m. Mit den Herstellerangaben kann die parasitäre Induktivität und Kapazität der Leitungen ausgerechnet werden. Die parasitäre Induktivität und die parasitäre Kapazität sind dementsprechend gleich $\frac{160.0\ \frac{\mu H}{km}}{1.4\ m} = 939.4$ nH und $160.0\ \frac{nF}{km} = 0.224$ nF .

III. Spannungsmessung mit einem Spannungsteiler

Die Spannungsmessung muss über einen Spannungsteiler durchgeführt werden, da die Analog-Eingabekarte *APCIe-3021* ein Signal von maximal 10.0 V aufzeichnen kann. Der Spannungsteiler besteht aus zwei Chip-Widerständen *SMD 1068* 220 Ω und 10 kΩ (Toleranz ±1.0 %)[72]. Die Chip-Widerstände sind auf einem SMT-Adapter montiert[56].

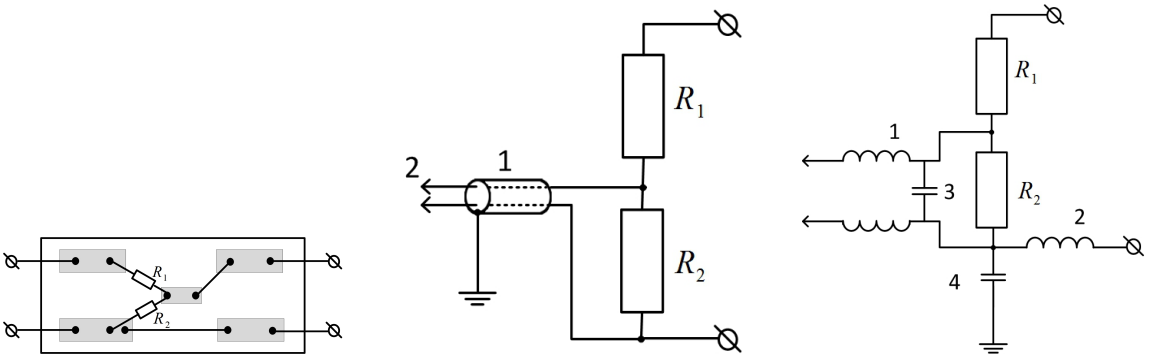

Abb. B.2.: **Links:** Platine des Spannungsteilers. **Mitte:**: Elektrisches Schema der Messung mit dem Spannungsteiler: R_1 - Widerstand *SMD 1068* 10 kΩ; R_2 - Widerstand *SMD 1068* 220 Ω; 1 - geschirmtes Kabel; 2 - Zu Analogsignal-Eingängen eines A/D-Wandlers. **Rechts:** Elektrisches Schema der Messung am Messwiderstand mit der Berücksichtigung der parasitären reaktiven Komponenten: 1- parasitäre Kabelinduktivität; 2 - parasitäre Induktivität des Spannungsteilers; 3,4 - parasitäre Kapazität des Kabels und Spannungsteiles;

Nach Gl. (B.1) [79] kann die parasitäre Induktivität des SMT-Adapters abgeschätzt werden.

$$L_{divider} = 0.0002 * l * (ln\frac{4l}{b} + 0.0224 * (\frac{b}{2l}) + 0.5)[\frac{\mathrm{H}}{\mathrm{m}}] \tag{B.1}$$

Die Koeffizienten l und b aus Gl. (B.1) geben die Länge und die Breite der Streifenleitung auf der Platine des Spannungsteilers (siehe Abb. B.2) an. Die parasitäre Induktivität des Spannungsteilers wurde mit 45.0 nH geschätzt. Die parasitäre Kapazität ist 0.3 pF gleich.

IV. Hochlast-Widerstand RB50

Für die Abschätzung der Störsignale, die während des Betriebes eines Schaltnetzteiles erzeugt werden, wurde ein Hochlast-Widerstand vom Type *RB50-2R2-J* 2.2 Ω an

die Stromquellen angeschlossen. Wegen der hohen elektrischen Leistung von 65 W ist der Hochlast-Widerstand mit einem Kühlkörper und einem Lüfter ausgestattet.

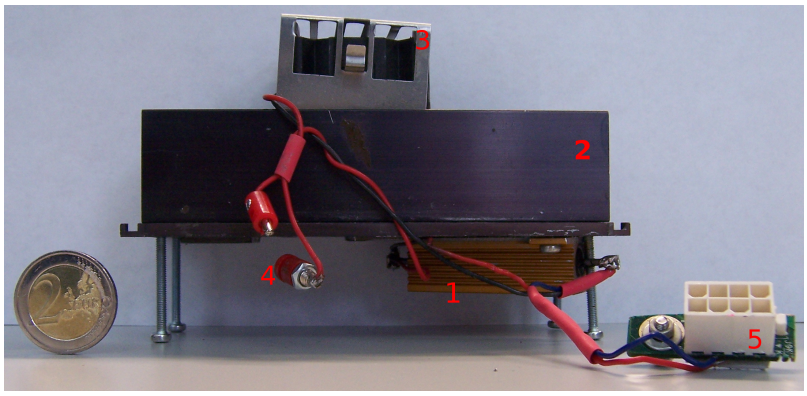

Abb. B.3.: Messeinrichtung mit einem Hochlast-Widerstand vom Type *RB50-2R2-J* 2.2 Ω und mit einem aktiven Kühlkörper: 1 - Hochlast-Widerstand; 2 - Kühlkörper; 3 - Lüfter; 4 - Ein/Ausschalter für den Lüfter; 5 - 8-poliger Anschluss für +12VDC Ausgang eines Schaltnetzteiles;

Wegen der hohen elektrischen Leistung von ≈ 65 W ist der Widerstand mit einem Kühlkörper und einem Lüfter ausgestattet, der allerdings während der Testversuche meistens ausgeschaltet blieb. Die Temperatur des Kühlkörpers stabilisiert sich bei etwa 116 °C. Bei der Verwendung des Akkumulators (siehe Abschnitt V) wurden die Messungen bei den Temperaturen zwischen ≈ 40 °C und 116 °C durchgeführt.

V. Messung mit Akku

Für die Untersuchung des Störsignals wurde als Stromquelle für die Prozessoren und den Hochlast-Widerstand ein Lithium-Polymer-Akkumulator anstatt eines Schaltnetzteiles verwendet. Die ATX-Stromversorgung wird weiterhin mit einem Schaltnetzteil realisiert. Der Spannungsbereich im Betrieb liegt zwischen 11.4 V und 12.5 V abhängig vom Ladezustand.

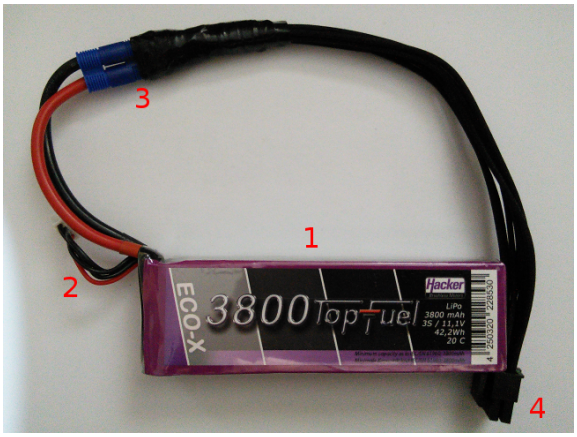

Abb. B.4.: Lithium-Polymer-Akkumulator mit einem 8-poligen CPU-Stecker: 1 - Lithium-Polymer-Akkumulator *Top Fuel*, 11.1V, 3800 mA h; 2 - Ladekabel; 3 - Adapter für ein Stromkabel für CPU; 4 - 8-poliger CPU-Stecker

VI. Messsystem-Integration in Rechenknoten

Eine Hauptplatine[1] eines Rechenknotens verfügt über mehrere Buchsen zur Verbindung eines Schaltnetzteiles. Zum Beispiel verfügen die gängigen „Dual Socket“ Hauptplatinen über einen 24-poligen ATX- und zwei 8-polige CPU-Stecker, die an einem Schaltnetzteil angeschlossen sind. Sowohl die Spannung als auch die Stromstärke werden an diesen Stromanschlüssen gemessen.

In Abb. B.5 links ist ein Sockel des Rechenknotens „node03“ dargestellt. Es sind folgende Hauptkomponenten zu erkennen: ein Prozessor (*1*), die Spannungsreglermodule VRM (*2*) für die Spannungsanpassung nach den Forderungen des Prozessors, vier Speichermodule (*3a*, *3b*), zwei VRMs (*4a*, *4b*) für die Spannungsanpassung nach der Forderung des Speichers und die 8- und 24-poligen Büchsen (*5*, *6*) für den Anschluss am Schaltnetzteil.

In Abb. B.5 rechts ist ein Verlängerungskabel für den Anschluss des 8-poligen CPU-Steckers eines Schaltnetzteils an der Hauptplatine (*7* und *8*) dargestellt. Das Kabel ist mit einem Messwiderstand (*11*) und zwei zusätzlichen Signalleitungen (*9* und *10*) der Länge von 1.4 m erweitert. Die Signalleitungen übertragen die Spannungs- und Stromverläufe an eine Analog-Eingabekarte. Daher werden die Hardwarekomponenten zusammen gemessen, die diegleiche Stromquelle besitzen. Somit teilen sich die Hauptkomponenten in mehreren Messgruppen auf (siehe Abschnitt I.II).

[1] In der englischsprachigen Literatur wird von „Mainboard“ gesprochen.

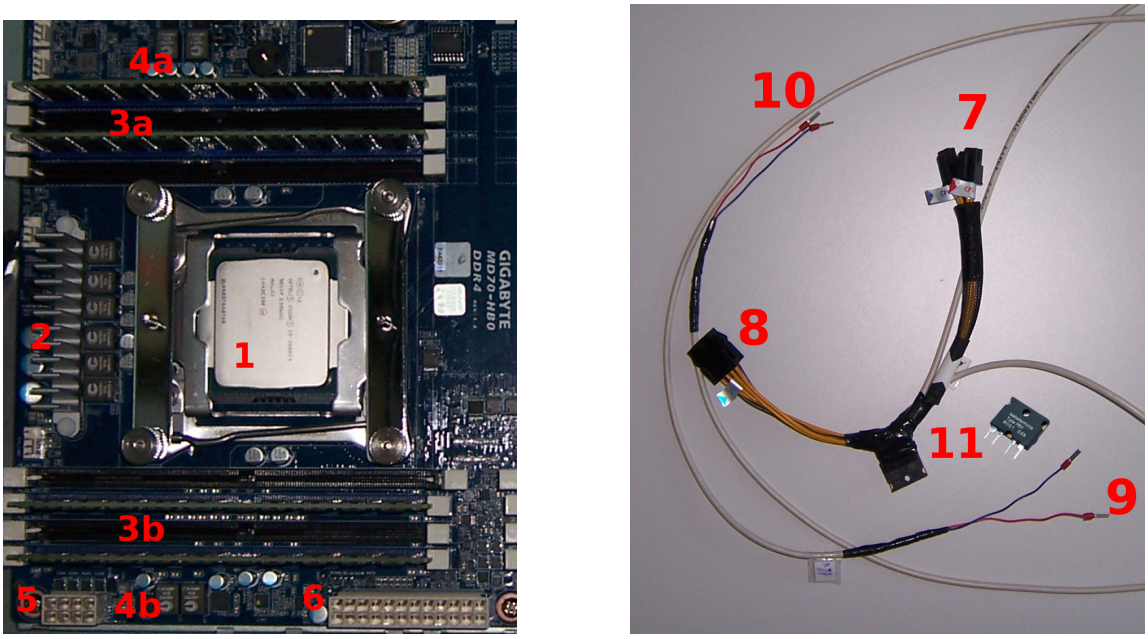

Abb. B.5.: Hauptplatine und Messeinrichtung

VII. Messung in Wechselstrom

Die Spannungs- und Stromverläufe am AC-Kaltgerätekabel werden mit Hilfe von zwei Präzisions-Stromwandlern *TZ77*[63] aufgenommen. Ein induktionsarmer Leistungswiderstand FPA100 2.2 kΩ ±5.0%[51] wird als ein Verbraucher für den Spannungsverlauf verwendet.
Abb. B.6 stellt eine entwickelte Messeinrichtung dar, die für die Aufzeichnung mit einem A/D-Wandler sowohl des Strom- als auch des Spannungsverlaufs benutzt werden kann. Der Spannungsverlauf wird am ersten Präzisions-Stromwandler aufgenommen, durch den der Strom über den passiven Leistungswiderstand FPA100 fließt. Für die Kalibrierung der Spannungsmessung wird das mit dem A/D-Wandler aufgenommene Signal mit der Anzeige eines Multimeters verglichen, das an die Messbuchsen angeschlossen wird. Über den zweiten Präzisions-Stromwandler fließt der Strom zu einem angeschlossenen Verbraucher. Damit wird der Stromverlauf aufgezeichnet. Für die Kalibrierung der Strommessung schaltet man das Gerät in den Betriebsmodus „3. Der Wert des passiven Widerstandes FPA100 ist mit 5 % Tolleranz bekannt.

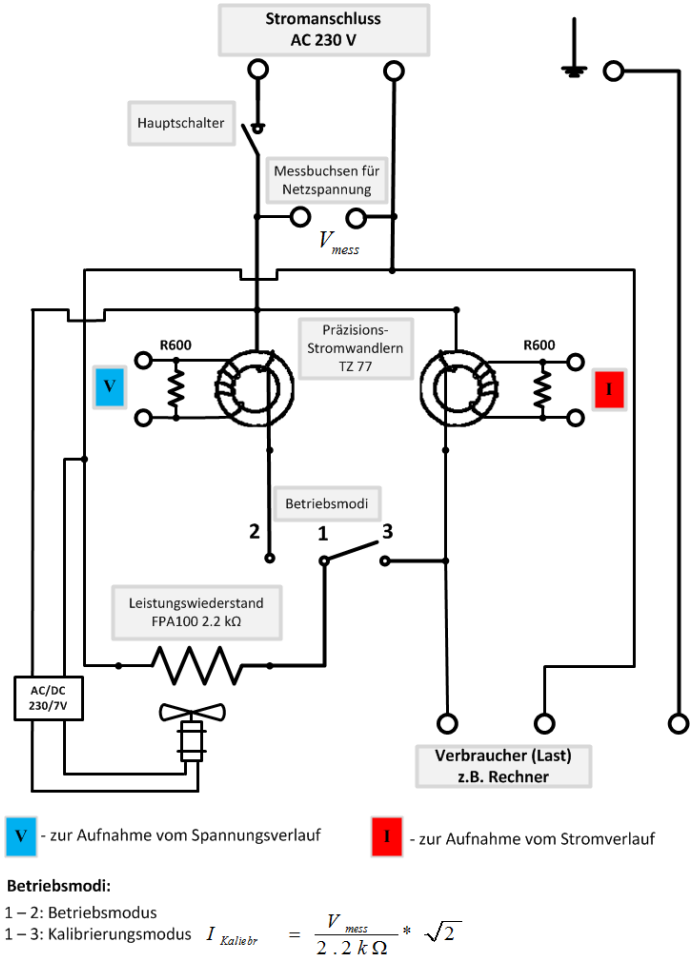

Abb. B.6.: Messeinrichtung für die Aufnahme der Spannungs- und Stromverläufe am AC-Kaltgerätekabel

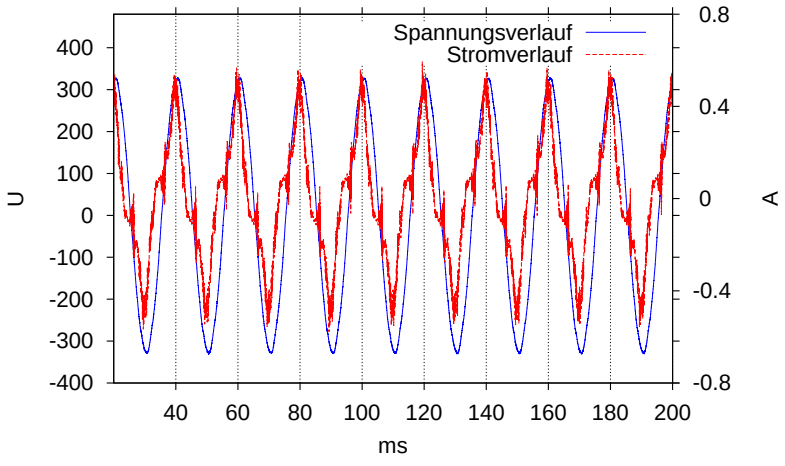

Abb. B.7.: Spannungs- und Stromverlauf für das Schaltnetzteil *Corsair VS550* mit einem Hochlast-Widerstand

Abb. B.7 stellt den Spannungs- und Stromverlauf für das Schaltnetzteil *Corsair VS550* dar, an den ein Hochlast-Widerstand vom Type *RB50-2R2-J* angeschlossen war. Zur Berechnung einer Wirkleistung multipliziert man die momentanen Messwerte und bildet daraus einen Mittelwert (siehe Gl. (B.2)).

$$P = \frac{1}{N} \sum_{n=0}^{N-1} V(n) \times I(n) \tag{B.2}$$

Die Messeinrichtung B.4 wurde für die Messung des Stromverbrauchs eines Desktop-Rechners verwendet. Für den Testcluster wurden zwei neue Geräte entwickelt. Das erste Gerät wird ausschließlich für die Spannungsprofile verwendet: Der aufgenommene Spannungsverlauf kann für die Berechnung des Stromverbrauchs aller Rechenknoten verwendet werden, die an der gleichen Phase angeschlossen sind. Das elektrische Schema und das Foto des Gerätes ist in Abb. B.8 dargestellt.

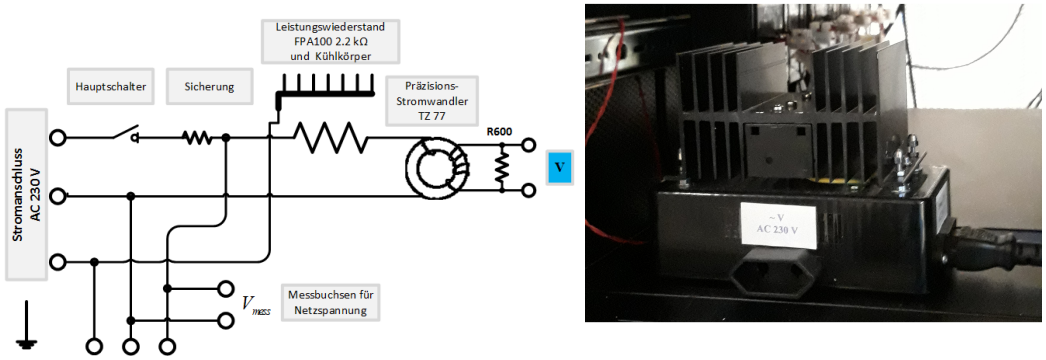

Abb. B.8.: **Links:** Elektrisches Schema der Messeinrichtung zur Aufnahme des Spannungsverlaufs am AC-Kaltgerätekabel. **Rechts:** Foto der Messeinrichtung zur Aufnahme des Spannungsverlaufs.

Das zweite Gerät wird für den Stromverlauf verwendet. Ein Gerät hat zwei Anschlüsse für die Versorgung der Rechenknoten. Das elektrische Schema und das Foto von drei im Testcluster verwendeten Geräten sind in Abb. B.9 dargestellt.

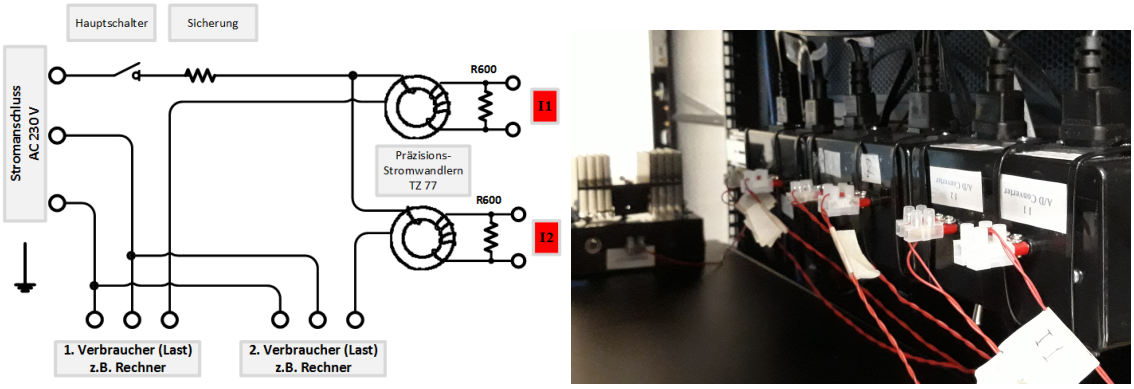

Abb. B.9.: **Links:** Elektrisches Schema der Messeinrichtung zur Aufnahme des Stromverlaufes am AC-Kaltgerätekabel. **Rechts:** Foto von drei Messeinrichtungen zur Aufnahme des Stromverlaufes. Im Hintergrund ist die Messeinrichtung zur Aufnahme des Spannungsverlaufs zu sehen.

VIII. Kalibrierungsfehler

Die Kalibrierungsparameter der Messeinrichtung wurden mit einem Multimeter bestimmt. Die Messgenauigkeit des Multimeters *VC950* für die Gleichspannung beträgt ±0.03%. Die Messgenauigkeit für die Wechselspannung mit einer Netzfrequenz von 50 Hz beträgt ±1.0 %. Der Koeffizient c_{shunt} aus Gl. (2.1) wurde aus den Herstellerangaben zur Toleranz von 0.5% für den Messwiderstand berechnet. Der Koeffizient c_{AC_I} aus Gl. (2.2) wurde mit den Toleranzangaben von 5% für den Leistungswiderstand (siehe Abschnitt VII) berechnet.
Der resultierende Kalibrierungsfehler einer Messgruppe ist weniger als ±0.6 %. Im Fall vom *ATX5V* erhöht sich der Fehler auf ±2.5 % (experimentell bestimmt), sofern die Spannung der Gruppe als konstant angenommen wird. Der Kalibrierungsfehler der Messung am AC-Kaltgerätekabel beträgt weniger als ±6.0 %. Die Messgenauigkeit bezieht sich auf eine konstante Last.

IX. Spannungsverlauf am Hochlast-Widerstand

Wegen der Einschränkung der Eingabekarte, die Signale von maximal 12 V aufzuzeichnen, muss das Signal um $c_{divider} \sim \frac{R_1+R_2}{R_2}$ mal geschwächt werden (siehe Abschnitt 2.1.3). Der dimensionslose Koeffizient $c_{divider}$ wurde mit der Genauigkeit von 0.03% für den verwendeten Spannungsteiler bestimmt und beträgt 45.580. Für die nachfolgenden Diagramme wird die Umrechnung der Messdaten nicht durchgeführt:

Es werden die Rohdaten des Messsystems analysiert.

IX.I. Spannungsverlauf und Schaltnetzteil

Der aufgezeichnete Spannungsverlauf der beiden Schaltnetzteiler ist in Abb. B.10 für 20 ms dargestellt. Die Messfrequenz eines differenziellen Messkanals der Analog-Eingabekarte betrug dabei 50.0 kHz.[2] Die Rohdaten sind mit roten Kreuzchen dargestellt. Zur Filterung wird ein Median-Filter nach Gl. (2.3) verwendet. Die gefilterten Messwerte x'_i sind grün und blau angekreuzt. Die grün dargestellten Messwerte sind mit den Parametern $n_l = n_r = 5$ und blau mit $n_l = n_r = 20$ gefiltert. Zwei Toleranzgrenzen für die Abweichung vom Mittelwert sind mit den horizontalen Linien gekennzeichnet: $\pm 0.5\%$ und $\pm 1.0\%$. Die in den folgenden Abbildungen angegebenen Parameter σ und μ sind nach Gl. (B.3) für die gefilterten Messdaten mit $n_l = n_r = 5$ berechnet.

$$\mu = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i;\ \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{B.3}$$

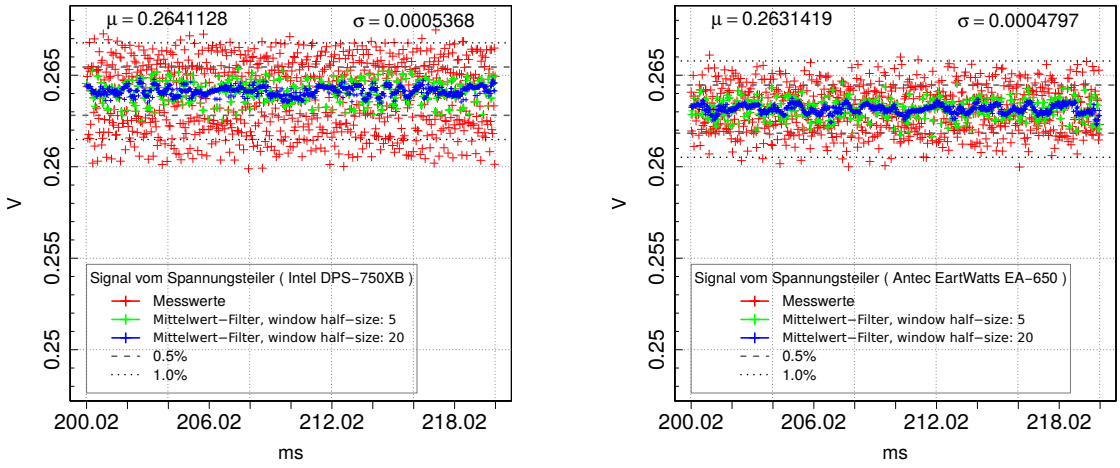

Abb. B.10.: Spannungsverlauf am Hochlast-Widerstand 2.2 Ω im Messkreis mit dem Schaltnetzteil *Intel* (links) und mit dem Schaltnetzteil *Antec* (rechts). Die Spannung wurde parallel am Spannungsteiler gemessen.

Vergleicht man die beiden Diagramme in Abb. B.10 nach der Streuung der Messdaten, so stellt man fest, dass das Signal des Schaltnetzteiles *Antec* eine bessere

[2] 1000 Messungen werden innerhalb des Zeitintervalls von 20 ms durchgeführt.

Qualität hat. Abgesehen von wenigen Ausnahmen streuen die Messwerte vom *Antec* innerhalb des 1-prozentigen Toleranzbereiches. Die Standardabweichungen für die gefilterten Daten sind $\sigma_{Intel} = 0.0005368$ und $\sigma_{Antec} = 0.0004797$: Der relative Unterschied beträgt 11.9 %.

Zwei Histogramme sind in Abb. B.11 für die Rohmessdaten der beiden Schaltnetzteile dargestellt.[3] Zum Vergleich sind außerdem zwei Dichtefunktionen der Normalverteilung gezeichnet. Die Normalverteilung ist durch den Mittelwert und durch die Standardabweichung parametrisiert, die nach Gl. (B.3) für die Rohdaten berechnet sind.

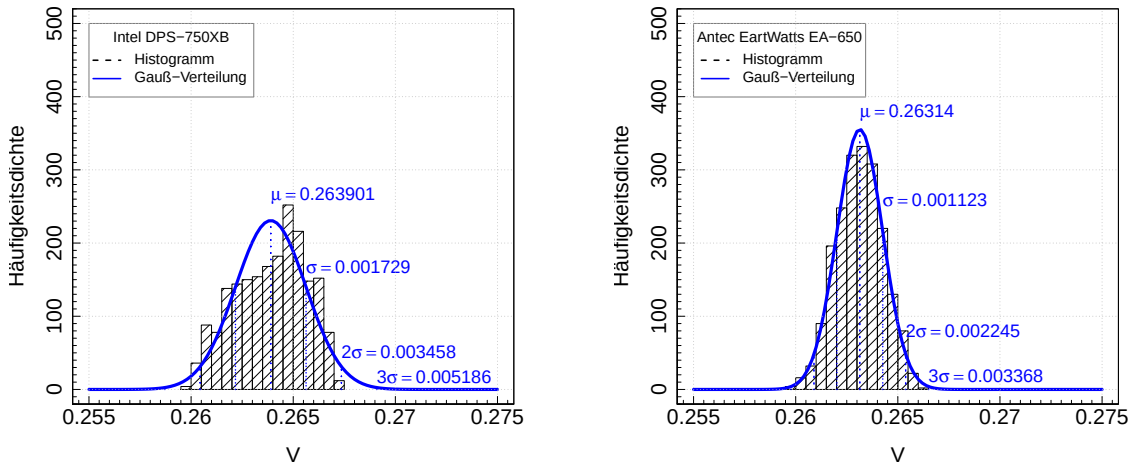

Abb. B.11.: Histogramm der Spannungswerte im Messkreis mit dem Schaltnetzteil *Intel* (links) und mit dem Schaltnetzteil *Antec* (rechts). Zusätzlich ist die Dichtefunktion der Normalverteilung dargestellt.

Es ist deutlich zu erkennen, dass die Verteilung der Messdaten für das Schaltnetzteil *Antec* viel näher an der Normalverteilung als die für das Schaltnetzteil *Intel* ist: Die Messwerte sind symmetrischer und weniger verstreut.

IX.II. Spannungsverlauf und Akkumulator

Für die nächste Messung verwenden wir statt eines Schaltnetzteiles einen Lithium-Polymer-Akkumulator, an den der Hochlast-Widerstand 2.2 Ω angeschlossen wird.

[3] Die beiden Histogramme sind aus den Stichproben jeweils mit 1000 Messwerten berechnet. Die Histogrammwerte sind mit der Anzahl der Messdaten und der Histogramm-Klassenbreite normiert, so dass die gesamte Fläche der Balken gleich eins ist.

Abb. B.12 stellt einen Spannungsverlauf am Spannungsteiler und das Histogramm der Messwerte dar. Der Unterschied zwischen dem Spannungsverlauf eines Schaltnetzteils und dem Spannungsverlauf des Akkumulators ist deutlich zu erkennen. Die Streuung des Lithium-Polymer-Akkumulators ist um mehr als 40-mal geringer als die Streuung des Schaltnetzteiles *Antec*. Dies bedeutet, dass der größte Anteil der Störungen vom Schaltnetzteil verursacht wird. Hiermit wird die Spannungsmessung durch die internen Störungen der Analog-Eingabekarte und der Umgebungsstrahlung gering beeinflusst.

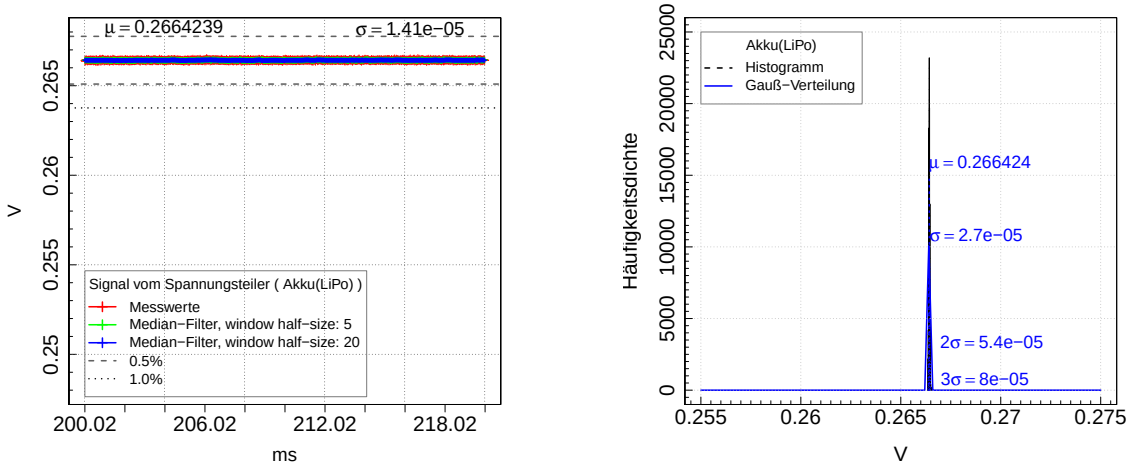

Abb. B.12.: **Links:** Spannungsverlauf am Hochlast-Widerstand 2.2 Ω im Messkreis mit dem Lithium-Polymer-Akkumulator. **Rechts:** Histogramm der Spannungswerte. Zusätzlich ist die Dichtefunktion der Normalverteilung dargestellt. Die Spannung wurde parallel am Spannungsteiler gemessen. Die Messfrequenz beträgt 50.0 kHz.

Zum Vergleich: Die Rohmesswerte streuen sich bei der Verwendung des Akkumulators im Intervall zwischen 0.2663 V und 0.2665 V. Somit beträgt die maximale Differenz zwischen den Messdaten[4] $\Delta_{min,max} = 0.2 \times 10^{-3}$. Ein vergleichbarer Messwiederholungsfehler für das Schaltnetzteil *Antec* kann erst durch die Anwendung des Mittelwert-Filters mit dem Patch-Window über 27 Messwerte erreicht werden.

[4]Die Differenz von 0.2×10^{-3} V entspricht der Betriebspannung von $45.580 * 0.2 \times 10^{-3} = 9.1$ mV im Messkreis (siehe Gl. (2.1)).

IX.III. Einfluss der Messfrequenz auf Spannungsmessung

In Abb. B.13 sind zwei Messungen mit der Messfrequenz von 50.0 kHz und mit der Messfrequenz von 20.0 kHz dargestellt. Die Spannung wurde für das Schaltnetzteil *Antec* aufgezeichnet. Die Dauer jeder Messung beträgt 20 ms. Die Diagramme bestehen entsprechend aus 1000 und 400 einzelnen Messpunkten.
Obwohl der Spannungsverlauf im Falle der Messfrequenz von 20 kHz mehr geglättet ist, haben die beiden Messungen einen ähnlichen Mittelwert μ und die Standardabweichung σ.

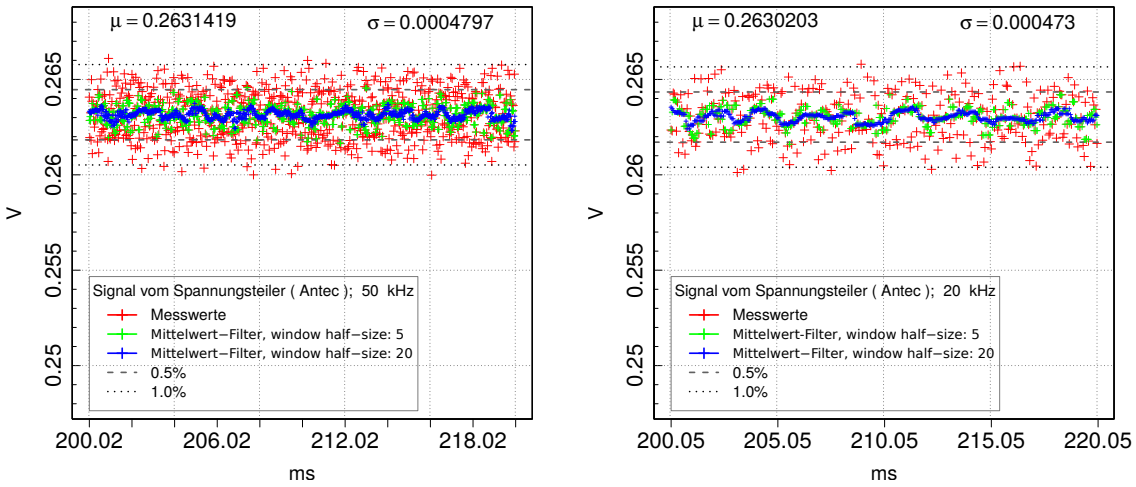

Abb. B.13.: Spannungsverlauf am Hochlast-Widerstand 2.2 Ω im Messkreis mit dem Schaltnetzteil *Antec*. Die Spannung wurde parallel am Spannungsteiler mit der Messfrequenz von 50 kHz (links) und mit der Messfrequenz von 20 kHz (rechts) gemessen.

In Tab. B.1 vergleicht man die Messungen der Spannung am Schaltnetzteil *Antec* und dem Hochlast-Widerstand, die mit den Frequenzen von 50.0 kHz und von 20.0 kHz durchgeführt wurden. In der ersten Spalte der Tabelle Tab. B.1 ist die Messfrequenz angegeben und in den zweiten und dritten Spalten sind die Dauer und die Anzahl der Messungen in einer Strichprobe angegeben.
Die weiteren Spalten enthalten das Minimum und das Maximum der fünfzig nacheinander folgenden Stichproben. Die Spalte $\Delta_{min,max}$ zeigt, wie groß die Differenz zwischen den minimalen und maximalen Messwerten ist.
Die zwei nächsten Spalten zeigen das Intervall und die Differenz der Mittelwerte. Und die letzte Spalte enthält den relativen Unterschied zwischen den Mittelwerten der fünfzig Stichproben.

Es ist deutlich zu erkennen, dass die Genauigkeit der Messung von der Messfrequenz sehr schwach abhängt.

Tab. B.1.: Statistik der Stichproben aus der Messung am Spannungsteiler im Messkreis mit dem Schaltnetzteil *Antec*. Die Spannung wurde parallel am Spannungsteiler gemessen. Die Messwerte sind nicht mit dem Koeffizienten $c_{divider} = 45.580$ skaliert.

kHz	ms	#	[min;max]	$\Delta_{min,max}$	$[\mu_{min};\mu_{max}]$	$\Delta_{\mu_{min},\mu_{max}}$	$\Delta_{\mu_{min},\mu_{max}}\%$
50	$2.00 * 10^{1}$	1000	[0.2593;0.2666]	$0.73 * 10^{-2}$	[0.2631;0.2632]	$0.10 * 10^{-3}$	$0.4 * 10^{-1}$
50	$8.00 * 10^{0}$	400	[0.2593;0.2665]	$0.72 * 10^{-2}$	[0.2631;0.2632]	$0.10 * 10^{-3}$	$0.4 * 10^{-1}$
50	$3.20 * 10^{0}$	160	[0.2597;0.2665]	$0.68 * 10^{-2}$	[0.2631;0.2632]	$0.10 * 10^{-3}$	$0.4 * 10^{-1}$
50	$1.34 * 10^{0}$	67	[0.2600;0.2665]	$0.65 * 10^{-2}$	[0.2631;0.2632]	$0.10 * 10^{-3}$	$0.4 * 10^{-1}$
50	$5.40 * 10^{-1}$	27	[0.2600;0.2665]	$0.65 * 10^{-2}$	[0.2630;0.2633]	$0.33 * 10^{-3}$	$1.3 * 10^{-1}$
50	$2.20 * 10^{-1}$	11	[0.2600;0.2661]	$0.61 * 10^{-2}$	[0.2629;0.2634]	$0.55 * 10^{-3}$	$1.9 * 10^{-1}$
50	$1.00 * 10^{-1}$	5	[0.2602;0.2661]	$0.59 * 10^{-2}$	[0.2627;0.2636]	$0.10 * 10^{-2}$	$3.4 * 10^{-1}$
50	$6.00 * 10^{-2}$	3	[0.2603;0.2661]	$0.58 * 10^{-2}$	[0.2625;0.2640]	$0.15 * 10^{-2}$	$5.7 * 10^{-1}$
20	$5.00 * 10^{1}$	1000	[0.2592;0.2665]	$0.73 * 10^{-2}$	[0.2629;0.2631]	$0.17 * 10^{-3}$	$0.8 * 10^{-1}$
20	$2.00 * 10^{1}$	400	[0.2592;0.2665]	$0.73 * 10^{-2}$	[0.2629;0.2630]	$0.14 * 10^{-3}$	$0.4 * 10^{-1}$
20	$8.00 * 10^{0}$	160	[0.2597;0.2661]	$0.64 * 10^{-2}$	[0.2629;0.2631]	$0.20 * 10^{-3}$	$0.8 * 10^{-1}$
20	$3.35 * 10^{0}$	67	[0.2600;0.2661]	$0.61 * 10^{-2}$	[0.2628;0.2631]	$0.29 * 10^{-3}$	$1.1 * 10^{-1}$
20	$1.35 * 10^{0}$	27	[0.2600;0.2658]	$0.58 * 10^{-2}$	[0.2627;0.2634]	$0.64 * 10^{-3}$	$2.7 * 10^{-1}$
20	$5.50 * 10^{-1}$	11	[0.2600;0.2658]	$0.58 * 10^{-2}$	[0.2623;0.2636]	$0.14 * 10^{-2}$	$4.9 * 10^{-1}$
20	$2.50 * 10^{-1}$	5	[0.2601;0.2658]	$0.57 * 10^{-2}$	[0.2620;0.2637]	$0.17 * 10^{-2}$	$6.5 * 10^{-1}$
20	$1.50 * 10^{-1}$	3	[0.2601;0.2655]	$0.54 * 10^{-2}$	[0.2618;0.2641]	$0.23 * 10^{-2}$	$8.7 * 10^{-1}$

X. Stromverlauf

Im Folgenden wird die Differenz der elektrischen Potentiale am Messwiderstand vom Type *PBV R010* 0.01 Ω $\pm 0.5\%$[55] aufgezeichnet, um die Stromstärke nach Gl. (2.1) zu berechnen. Der Widerstand muss am positiven Pol vor dem Verbraucher eingesetzt werden, da die Grundleitungen einer Hautplatine höchstwahrscheinlich kurzgeschlossen sind. Sonst wären die Messungen verfälscht, weil erst ein Teil des elektrischen Stromes durch den Messwiderstand am negativen Pol flösse. Die Einzelheiten der Messwiderstand-Anwendung sind in Anhang B auf Seite 124 zu finden.
Die aufgezeichnete Potentialdifferenz wird in die Stromstärke durch die Multiplikation mit dem Koeffizient $c_{shunt} = 100.0\ \frac{1}{\Omega}$ umgerechnet. Für die nachfolgenden Diagramme wird die Umrechnung der Messdaten nicht durchgeführt: Es werden die Rohdaten des Messsystems analysiert.

X.I. Stromverlauf und Schaltnetzteil

Wie im Fall der Messung am Spannungsteiler wird im Folgenden der Einfluss der verschiedenen Stromquellen analysiert. Abb. B.14 stellt den unskalierten Stromverlauf dar, der am Messwiderstand mit der Analog-Eingabekarte aufgenommen wurde. Die Messfrequenz für einen differenziellen Messkanal betrug 50.0 kHz. In den beiden Diagrammen ist ein Abschnitt von 20 ms dargestellt.

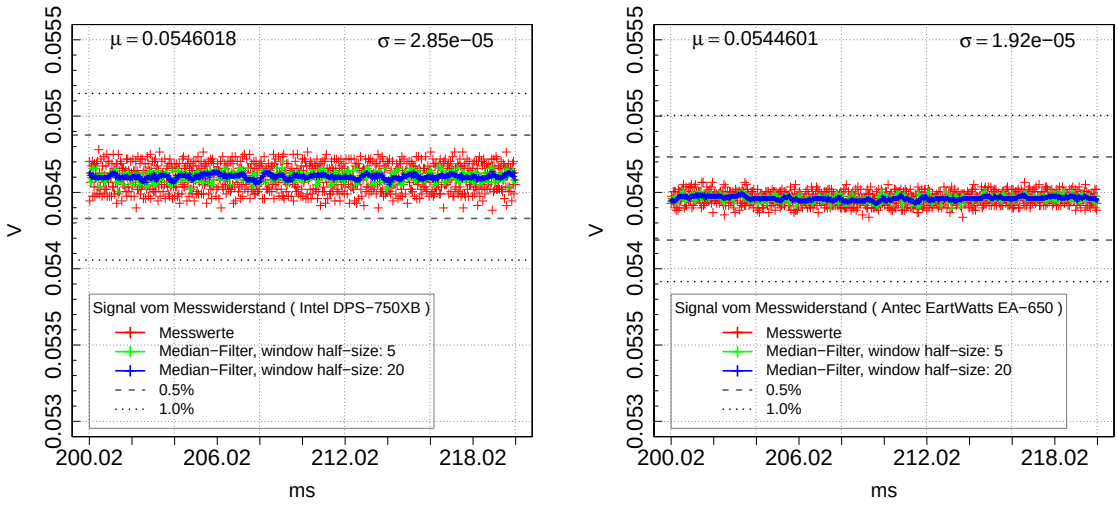

Abb. B.14.: Stromverlauf am Hochlast-Widerstand 2.2 Ω im Messkreis mit dem Schaltnetzteil *Intel* (links) und mit dem Schaltnetzteil *Antec* (rechts). Die Differenz der elektrischen Potentiale wurde am Messwiderstand *PBV R010* gemessen. Die Messwerte sind nicht mit dem Koeffizient $c_{shunt} = 100.0\ \frac{1}{\Omega}$ skaliert. Die Messfrequenz war 50 kHz.

Die Diagramme bestehen dementsprechend aus 1000 Messungen. Die Streuung der Messdaten ist beim Schaltnetzteil *Antec* deutlich niedriger als die Streuung beim Schaltnetzteil *Intel*. Es ist deutlich zu erkennen, dass sich die Streuung im Vergleich zu Abb. B.10 allgemein etwa auf die Hälfte verringerte. Zwei Histogramme für die Rohmessdaten der beiden Schaltnetzteile sind in Abb. B.15 dargestellt. Die Strommessung für das Schaltnetzteil *Antec* ist symmetrischer und weniger verstreut als die für das Schaltnetzteil *Intel*.
Zusätzlich kann festgestellt werden, dass der Quantisierungsfehler bei der Auswertung der Messdaten für das Schaltnetzteil *Antec* deutlich zu erkennen ist. Die noch

zu unterscheidende Differenz zwischen zwei nächstliegenden Messwerten beträgt typischerweise $\pm 1 \times \text{LSB} \approx 7.629 \times 10^{-6}$ V, jedoch in seltenen Fällen bis zu $\pm 2 \times \text{LSB} \approx 3.052 \times 10^{-5}$ V.[5] Dies bedeutet, dass die Auflösung der Messung der Stromstärke im Messkreis zwischen ± 7.629 mA und ± 30.518 mA liegt. Der Mittelwert der gefilterten Rohdaten μ und der Mittelwert der nicht gefilterten Rohdaten $\underline{\mu}$ unterscheiden sich voneinander um einen noch kleineren Betrag:

$$\begin{aligned} \mu_{Intel} - \underline{\mu}_{Intel} &= 0.0546018 - 0.054598 &&= 3.8 * 10^{-6}; \\ \mu_{Antec} - \underline{\mu}_{Antec} &= 0.0544601 - 0.054460 &&= 1.0 * 10^{-6}; \end{aligned} \tag{B.4}$$

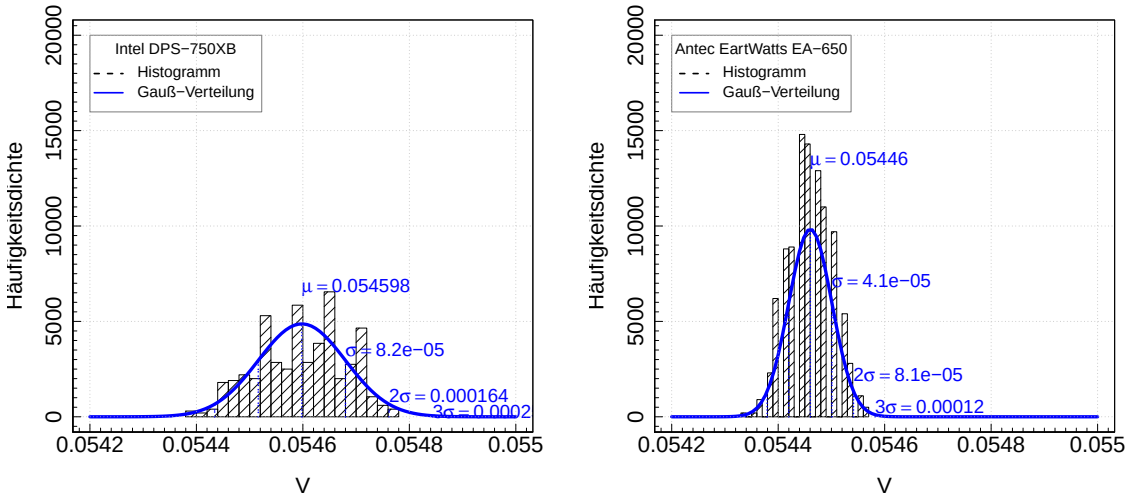

Abb. B.15.: Histogramm der Stromwerte im Messkreis mit dem Schaltnetzteil *Intel* (links) und mit dem Schaltnetzteil *Antec* (rechts). Die Differenz der elektrischen Potentiale wurde am Messwiderstand *PBV R010* gemessen. Zusätzlich ist die Dichtefunktion der Normalverteilung dargestellt. Der Quantisierungsfehler der Analog-Eingabekarte von $\frac{1}{2}\times$ LSB ist im rechten Histogramm deutlich zu erkennen. Die Messfrequenz beträgt 50.0 kHz.

X.II. Stromverlauf und Akkumulator

In Abb. B.16 ist der Fall der Strommessung dargestellt, bei dem der Lithium-Polymer-Akkumulator deutlich weniger Störungen als ein Schaltnetzteil produziert. Die Streuung der Messwerte ist um etwa 1.5-mal geringer.
Hiermit wird angenommen, dass die Störungen der Messkarte und der Umgebung

[5] Der Quantisierungsfehler entsteht bei der Digitalisierung des Analogsignals (siehe Anhang B auf Seite 123).

einen vergleichsweise geringeren Einfluss auf den Wiederholungsfehler der Spannung und der Stromstärke ausüben. Allerdings ist der relative Einfluss der Störung auf die Resultate deutlich größer als bei der Spannungsmessung.
Zum Vergleich: Die Rohmesswerte bei der Verwendung des Akkumulators *Akku* streuen im Intervall zwischen 0.055 71 V und 0.055 53 V. Die Differenz beträgt dabei $\Delta_{min,max} = 0.18 * 10^{-4}$. Wenn diese Differenz in Ampere konvertiert wird, beträgt sie 1.8mA.

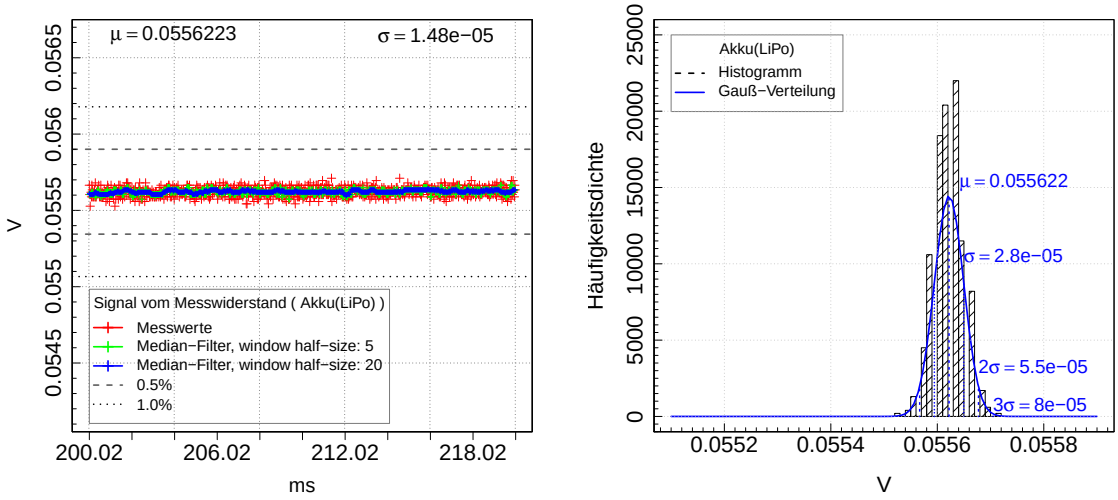

Abb. B.16.: **Links:** Stromverlauf am Hochlast-Widerstand 2.2 Ω im Messkreis mit dem Lithium-Polymer-Akkumulator. **Rechts:** Histogramm der Stromwerte. Zusätzlich ist die Dichtefunktion der Normalverteilung dargestellt. Die Differenz der elektrischen Potentiale wurde am Messwiderstand *PBV R010* gemessen. Die Messwerte sind nicht mit dem Koeffizient $c_{shunt} = 100.0\ \frac{1}{\Omega}$ skaliert. Die Messfrequenz beträgt 50.0 kHz.

C. Oszillogramme

In diesem Anhang sind die weiteren Oszillogramme dargestellt, die für die in Abschnitt 2.2.1 beschriebene Anwendung aufgezeichnet wurden.

I. Oszillogramm für Spannungssignal am Schaltnetzteil

Die Spannung wurde parallel zum Hochlast-Widerstand 2.2 Ω [51] gemessen, der direkt an den +12VCD Ausgang des Schaltnetzteiles *Antec EartWatts* angeschlossen wurde.

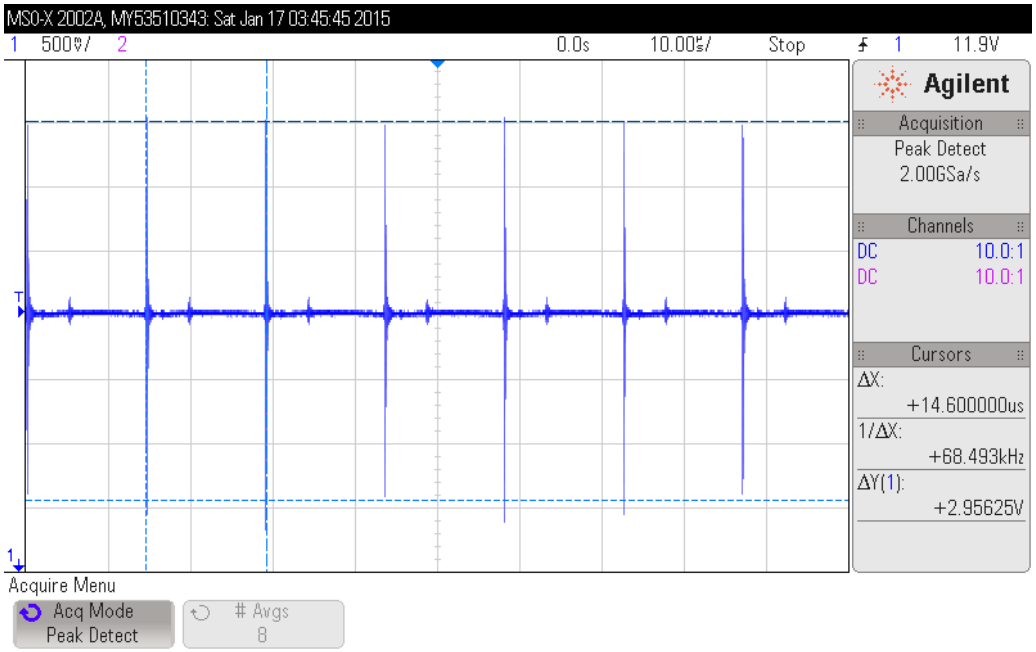

Abb. C.1.: Oszillogramm für den 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oziloskop Mode ist „Peak detect“.

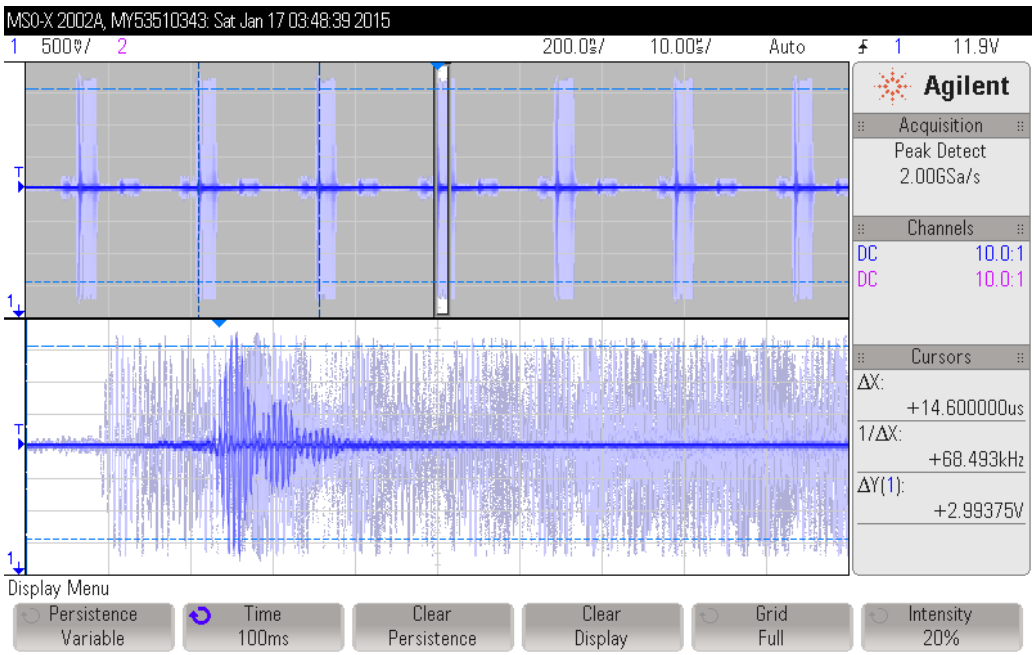

Abb. C.2.: Oszillogramm für den 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oszilloskop Mode ist „Peak Detect Mode to Find a Glitch“.

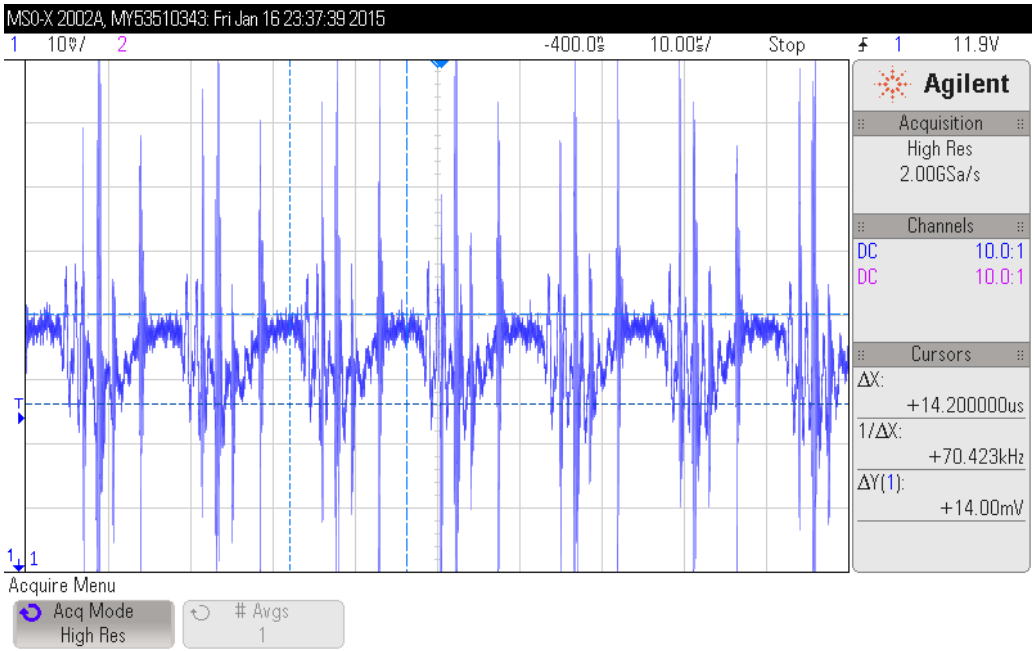

Abb. C.3.: Oszillogramm für den 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oszilloskop Mode ist „High resolution“.

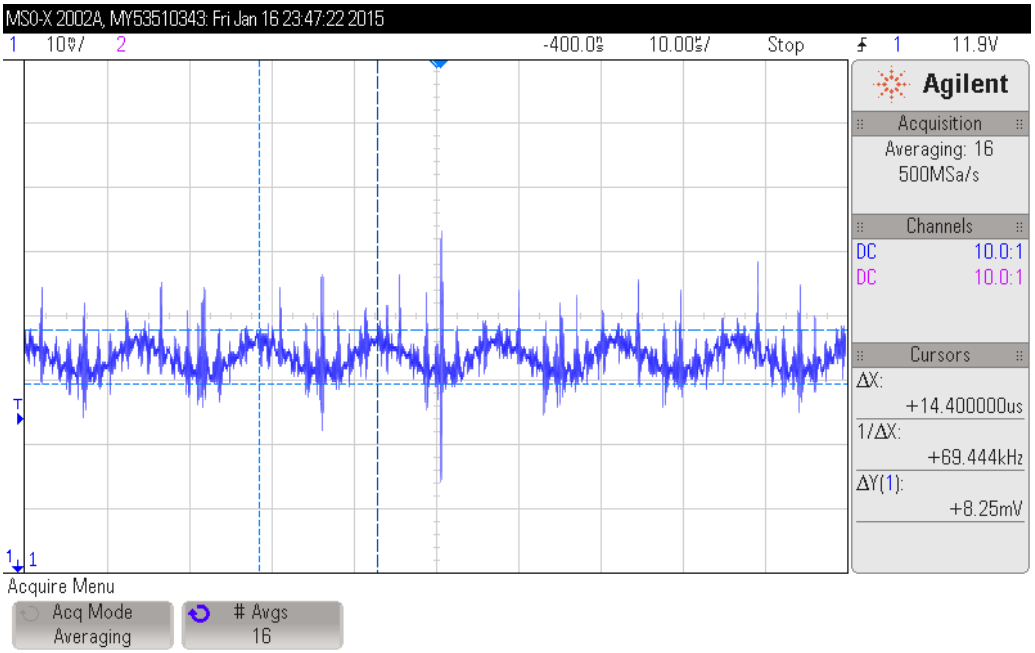

Abb. C.4.: Oszillogramm für den 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oziloskop Mode ist „Average over 16 samples“.

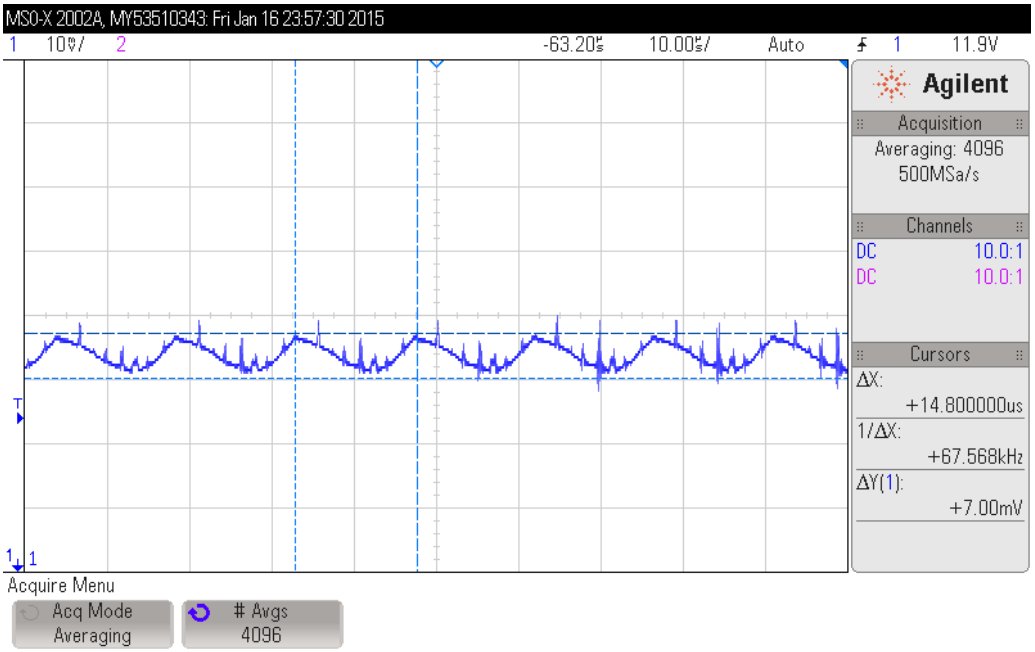

Abb. C.5.: Oszillogramm für den 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω [51]. Der Oziloskop Mode ist „Average over 4096 samples“.

II. Oszillogramme für Spannungssignal am Spannungsteiler

Die Spannung wurde parallel zum Hochlast-Widerstand 2.2 Ω vom Type *PB504 8R2* am Spannungsteiler gemessen. Der Hochlast-Widerstand wurde direkt an den +12VCD Ausgang des Schaltnetzteiles *Antec EartWatts* angeschlossen. Messung von 12 V Ausgang

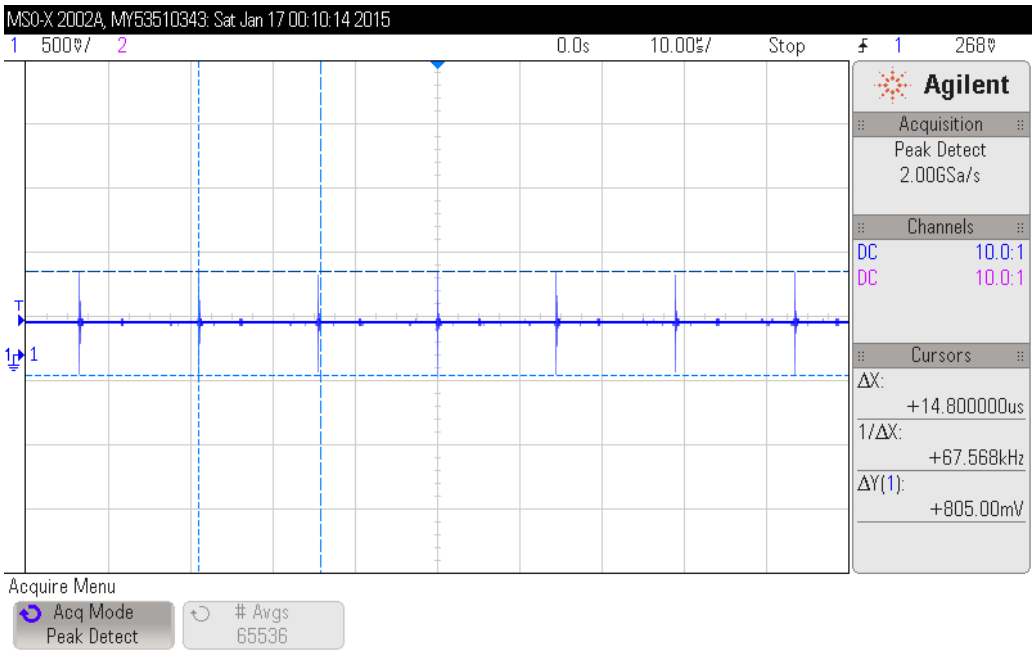

Abb. C.6.: Oszillogramm für den Spannungsteiler 220 Ω / 10 kΩ am 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Osziloskop Mode ist „Peak detect“.

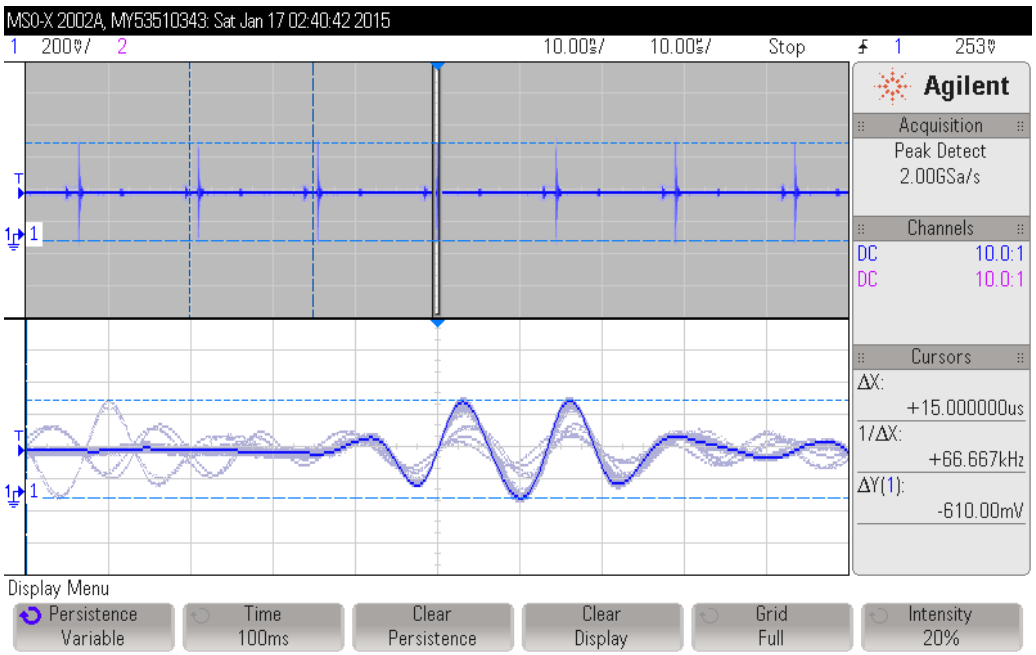

Abb. C.7.: Oszillogramm für den Spannungsteiler 220 Ω / 10 kΩ am 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oziloskop Mode ist „Peak Detect Mode to Find a Glitch“.

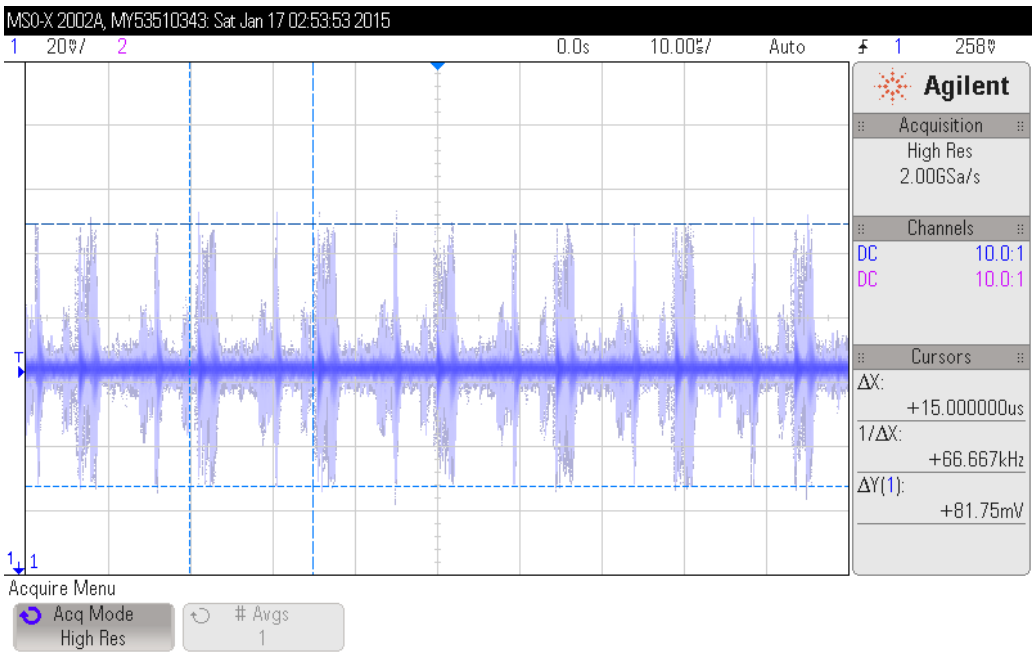

Abb. C.8.: Oszillogramm für den Spannungsteiler 220 Ω / 10 kΩ am 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oziloskop Mode ist „High resolution“.

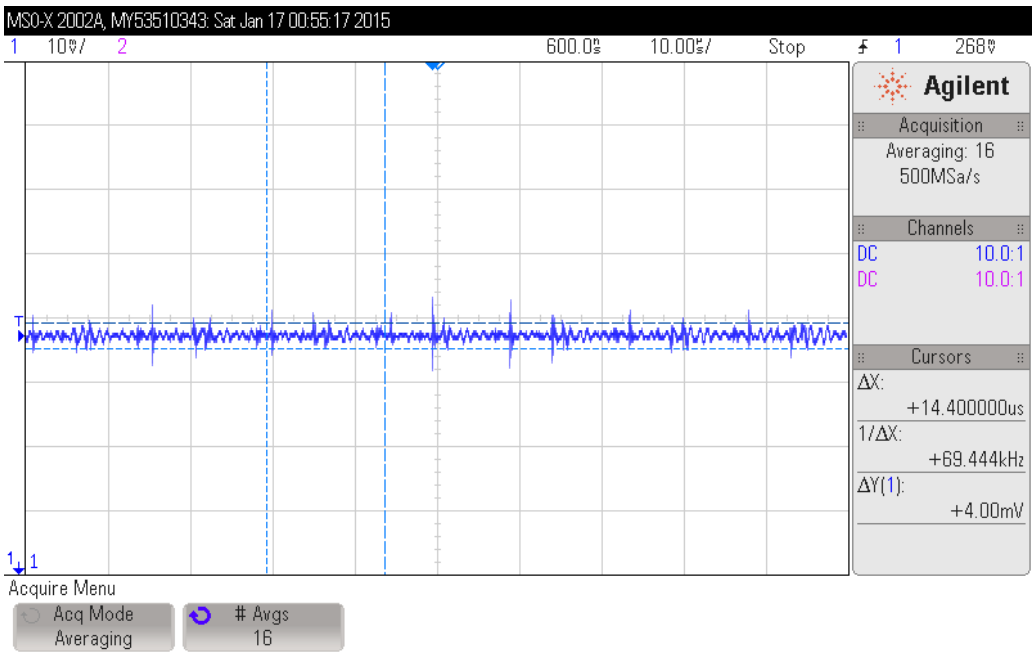

Abb. C.9.: Oszillogramm für den Spannungsteiler 220 Ω / 10 kΩ am 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oszilloskop Mode ist „Average over 16 samples“.

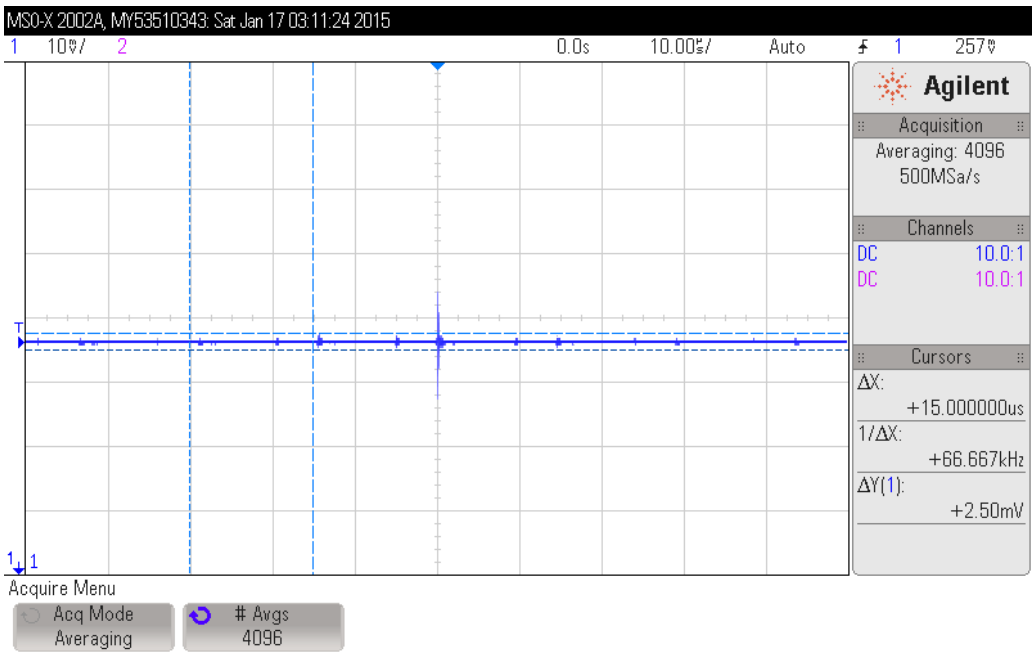

Abb. C.10.: Oszillogramm für den Spannungsteiler 220 Ω / 10 kΩ am 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oszilloskop Mode ist „Average over 4096 samples“.

III. Oszillogramme für Stromsignal

Im Folgenden wird die Differenz der elektrischen Potentiale am Messwiderstand vom Type *PBV R010* 0.01 Ω ±0.5% [55] dargestellt. Mit Hilfe dieses Signales kann die Stromstärke nach Gl. 2.1 berechnet werden.

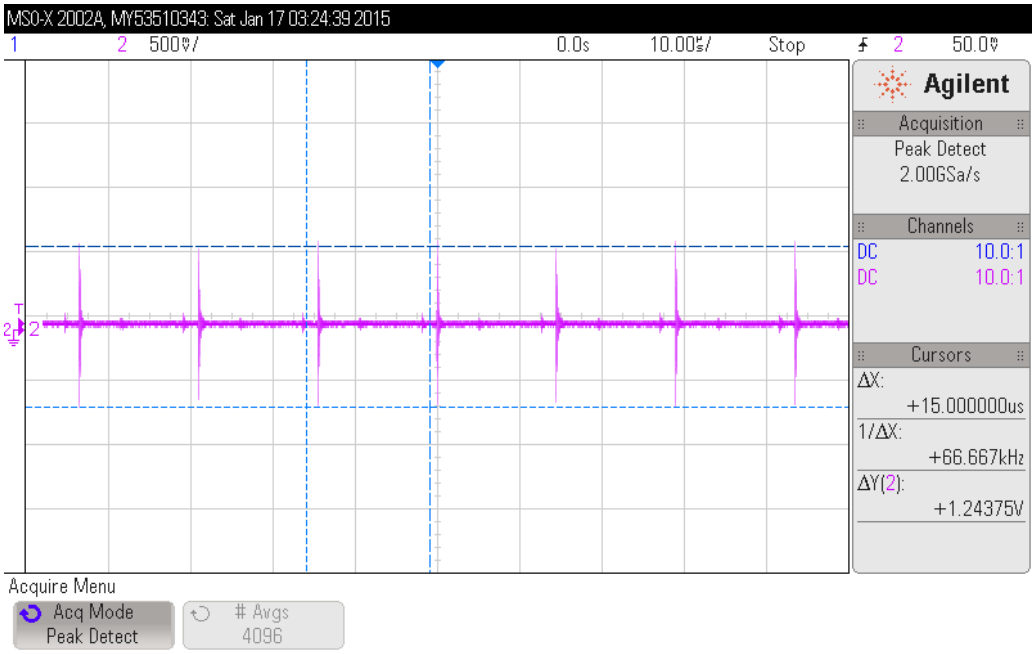

Abb. C.11.: Oszillogramm für den Shunt 0.01 Ω am 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oziloskop Mode ist „Peak detect“.

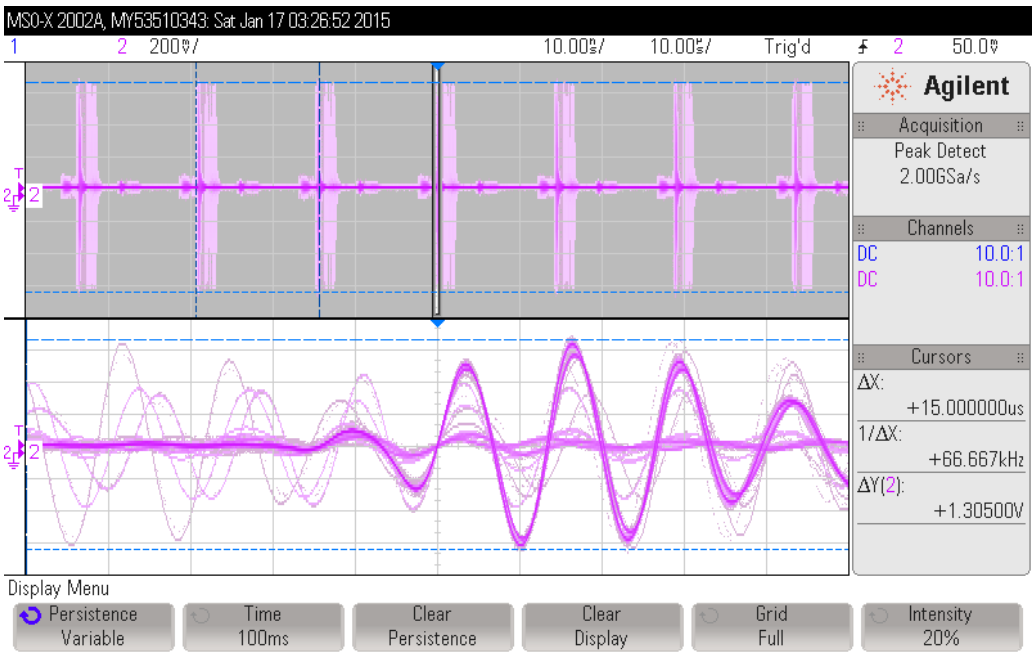

Abb. C.12.: Oszillogramm für den Shunt 0.01 Ω am 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oszilloskop Mode ist „Peak Detect Mode to Find a Glitch“.

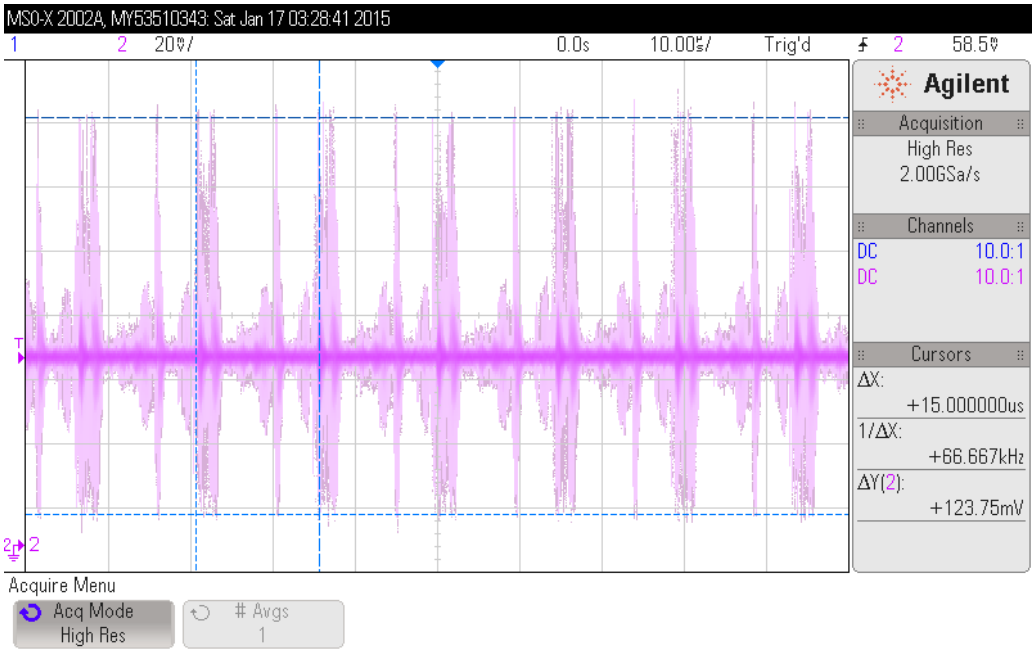

Abb. C.13.: Oszillogramm für den Shunt 0.01 Ω am 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oszilloskop Mode ist „High resolution“.

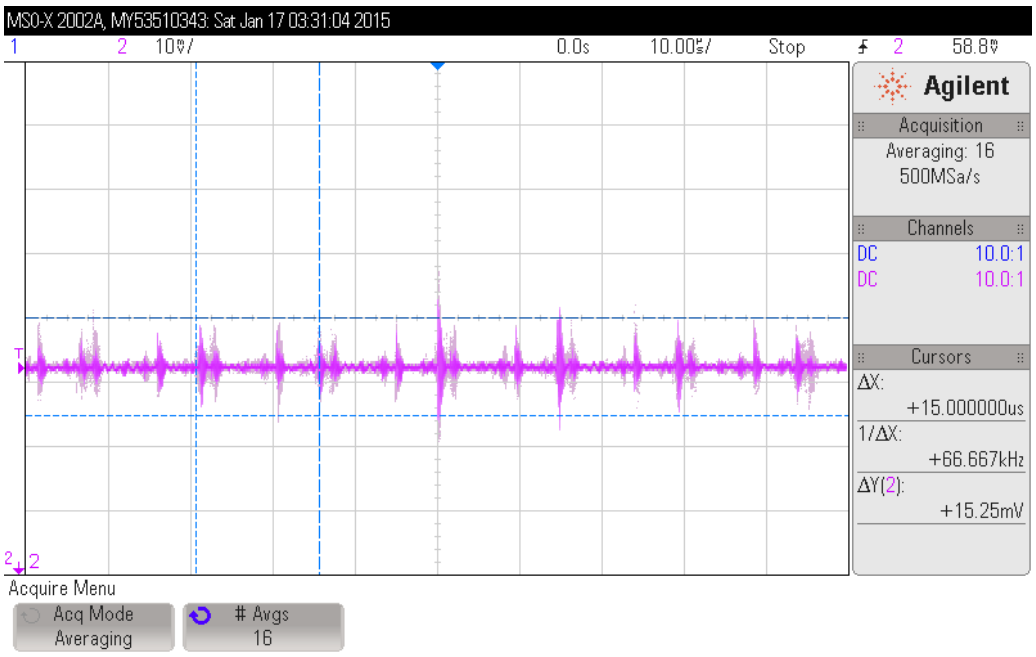

Abb. C.14.: Oszillogramm für den Shunt 0.01 Ω am 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oszilloskop Mode ist „Average over 4 samples“.

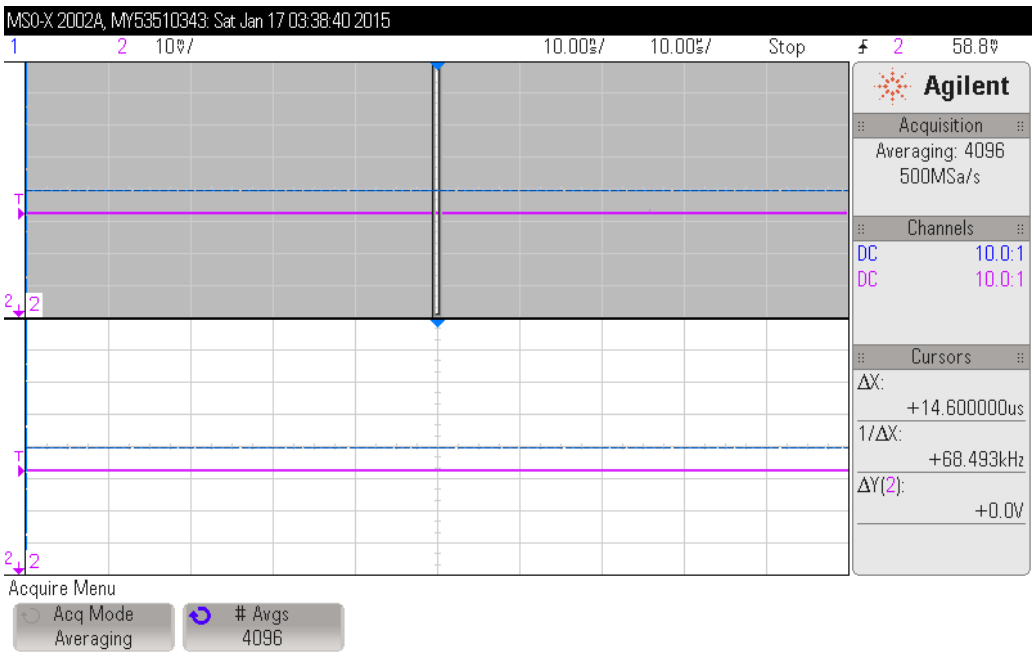

Abb. C.15.: Oszillogramm für den Shunt 0.01 Ω am 12V Ausgang (8-pin CPU1) des Schaltnetzteiles *Antec EartWatts EA-450* mit angeschlossenem Hochlast-Widerstand 2.2 Ω. Der Oszilloskop Mode ist „Average over 4096 samples“.

D. Betrieb ohne Rechenlast

I. Zähler für Betrieb ohne Rechenlast

In der Tabelle Tab. D.1 sind die System-Laufzeitparameter angegeben, die mit dem Tool „Intel pcm“[43] für den Rechenknoten *node03*[1] während des Betriebes ohne Rechenlast aufgezeichnet wurden. Die Messung dauerte 100 Sekunden. Das Tool gibt folgende Informationen aus:

- *SKT*: Socket-Id (SKT0 - der ertste Prozessor; SKT1 - der zweite Prozessor).
- *EXEC*: Anzahl der Instruktionen in einem Takt bei der nominalen Frequenz.
- *FREQ*: Verhältnis der tatsächlichen zur nominalen Frequenz des Prozessors: $\frac{unhalted\_clock\_ticks}{invariant\_time\_ticks}$
- *AFREQ*: Verhältnis der tatsächlichen zur nominalen Frequenz des Prozessors, wenn der Prozessor im aktiven Zustand ist (C0-State): $\frac{unhalted\_clock\_ticks\_c0}{invariant\_time\_ticks}$
- *C6-State %*: Zeitanteil, währenddessen die Prozessorkerne in idle power state waren.
- *C6-PState %*:Zeitanteil, währenddessen der Prozessor in idle power state war.
- *CPU W*: Stromverbrauch des Prozessors [W] (nach RAPL-Zählern)
- *RAM W*: Stromverbrauch des Speichers [W] (nach RAPL-Zählern)

Tab. D.1.: System-Laufzeitparameter während des Betriebes ohne Rechenlast auf dem Rechenknoten *node03*. Die Daten sind vom Tool „Intel pcm“ aufgezeichnet.

SKT	EXEC	FREQ	AFREQ	C6-State%	C6-PState%	CPU W	RAM W
SKT0	$1.32 * 10^{-4}$	$1.66 * 10^{-4}$	$5.28 * 10^{-1}$	99.98	78.28	9.39	1.26
SKT1	$4.13 * 10^{-5}$	$5.17 * 10^{-5}$	$4.81 * 10^{-1}$	99.84	78.18	9.90	1.83

Auffallend ist der große Unterschied zwischen den Werten in den Spalten „FREQ“

[1] Der Rechenknoten *node03* des Testclusters ist mit zwei Prozessoren *Haswell E5-2680v3* ausgestattet (siehe Abschnitt I.I auf Seite 118).

und „AFREQ“. Im energiesparenden Zustand *C6-PState* verbrachten die Prozessoren erst 78 % der Zeit: nur wenn alle Prozessorkerne im *C6-State* sind, kann der gesamte Prozessor in den Zustand *C6-PState* versetzt werden.
In der Tabelle Tab. D.2 sind die System-Laufzeitparameter angegeben, die mit dem Tool „Intel pcm“ für den Rechenknoten *node02*[2] während des Betriebes ohne Rechenlast aufgezeichnet wurden. Im Unterschied zum *Haswell* Prozessor konnte der *Ivy Bridge* nicht in den energiesparenden Zustand *C6-PState* versetzt werden.

Tab. D.2.: System-Laufzeitparameter während des Betriebes ohne Rechenlast auf dem Rechenknoten *node02*. Die Daten sind vom Tool „Intel pcm“ aufgezeichnet.

SKT	EXEC	FREQ	AFREQ	C6-State%	C6-PState%	CPU W	RAM W
SKT0	$8.52 * 10^{-5}$	$3.00 * 10^{-4}$	$7.13 * 10^{-1}$	99.99	0.00	15.89	1.57
SKT1	$9.14 * 10^{-5}$	$1.55 * 10^{-4}$	$4.61 * 10^{-1}$	99.97	0.00	15.70	1.57

II. Spannungs- und Stromverlauf (Haswell)

In Abb. D.1 sind zwei Histogramme des Spannungs- und Stromverlaufs angegeben, die für den ersten Prozessor *Haswell E5-2680v3* des Rechenknotens „node03“ während des Betriebes ohne Rechenlast aufgenommen wurden.

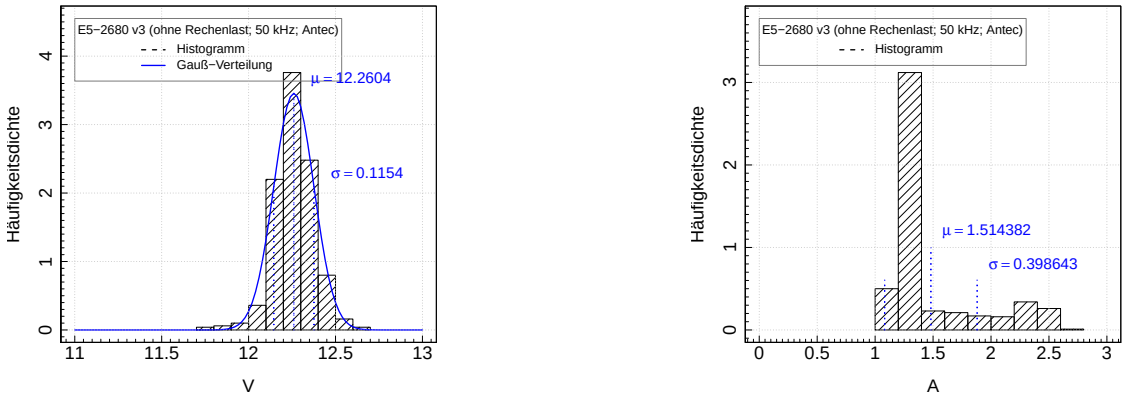

Abb. D.1.: Histogramme der Spannungswerte (links) und der Stromwerte (rechts) im Messkreis mit dem Prozessor *Haswell E5-2680v3* und dem Schaltnetzteil *Antec* während des Betriebes ohne Rechenlast

[2] Der Rechenknoten *node02* des Testclusters ist mit zwei Prozessoren *Ivy Bridge E5-2690v2* ausgestattet (siehe Abschnitt I.I auf Seite 118).

Abb. D.2 und Abb. D.3 stellen den Verlauf der elektrischen Leistung der beiden Prozessoren des Rechenknotens „node03“ für 5 ms dar. Die Messung wurde mit vier Messkanälen (25 kHz) durchgeführt. Das Diagramm D.2 zeigt die Rohmessdaten und das Diagramm D.3 die gefilterten Messdaten. Die Spannung wurde dabei mit dem Mean-Filter ($n_l = n_r = 5$) und der Strom mit dem Median-Filter ($n_l = n_r = 1$) gefiltert. Die elektrische Leistung der beiden Prozessoren haben ein sehr ähnliches Verhalten, allerdings gibt es doch zwei Unterschiede. In den Sekunden 1164 und 1165 fällt die Leistung des zweiten Prozessors rasant ab, während der erste Prozessor noch für eine halbe Millisekunde aktiv bleibt. Der zweite Unterschied besteht darin, dass die elektrische Leistung des ersten Prozessors im *C6-PState* um ~ 3 W größer ist. Zur Kontrolle der Messeinrichtung wurden die 4-Pin-CPU Stromkabel der beiden Prozessoren vertauscht. Dies hatte keinen Einfluss auf das Resultat.

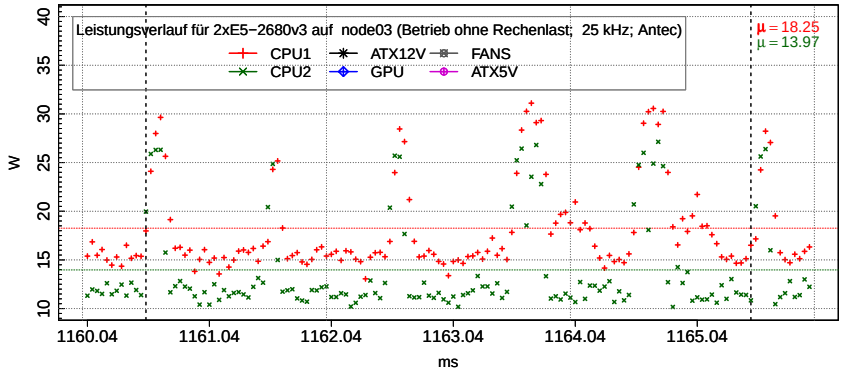

Abb. D.2.: Verlauf der elektrischen Leistung in Messkreisen mit zwei Prozessoren *Haswell* und mit dem Schaltnetzteil *Antec* während des Betriebes ohne Rechenlast

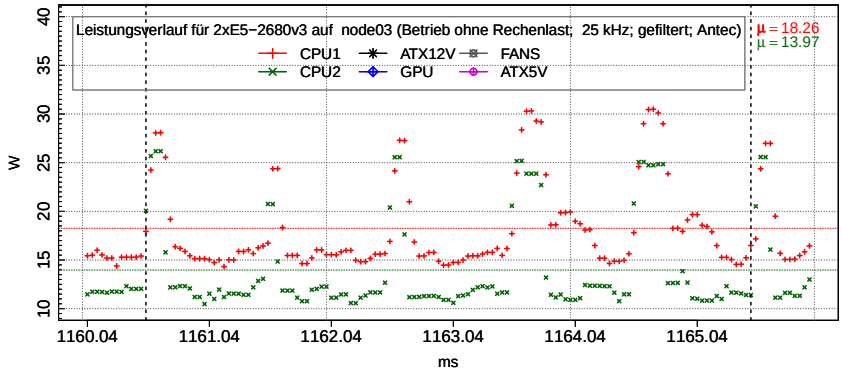

Abb. D.3.: Verlauf der elektrischen Leistung in Messkreisen mit zwei Prozessoren *Haswell* und mit dem Schaltnetzteil *Antec* während des Betriebes ohne Rechenlast

In Abb. D.4 ist der Verlauf der elektrischen Leistung der beiden Prozessoren des Rechenknotens „node03“ für das Zeitintervall von 10 s dargestellt. Wegen der geringen Auflösung der Darstellung mussten die Filterparameter geändert werden, um die Differenzen in der elektrischen Leistung über die größeren Zeitintervalle zu erkennen. Die Messdaten wurden mit dem Mittelwert-Filter $n_l = n_r = 1250$ für die beiden Signale gefiltert. Die kurzen Impulse wurden dadurch geglättet. Der Mittelwert μ erhöhte sich um weniger als 1 W. Es ist deutlich zu erkennen, dass sich die elektrischen Leistung nicht nur innerhalb von wenigen Millisekunden, sondern auch im Sekundenbereich änderte. Jede zweite Sekunde erhöhte sich der Stromverbrauch für Hunderte von Millisekunden.
Welche Dienste die betrachteten Muster des Leistungsverlaufs verursachten, muss in dieser Arbeit folgenden Untersuchungen geklärt werden.

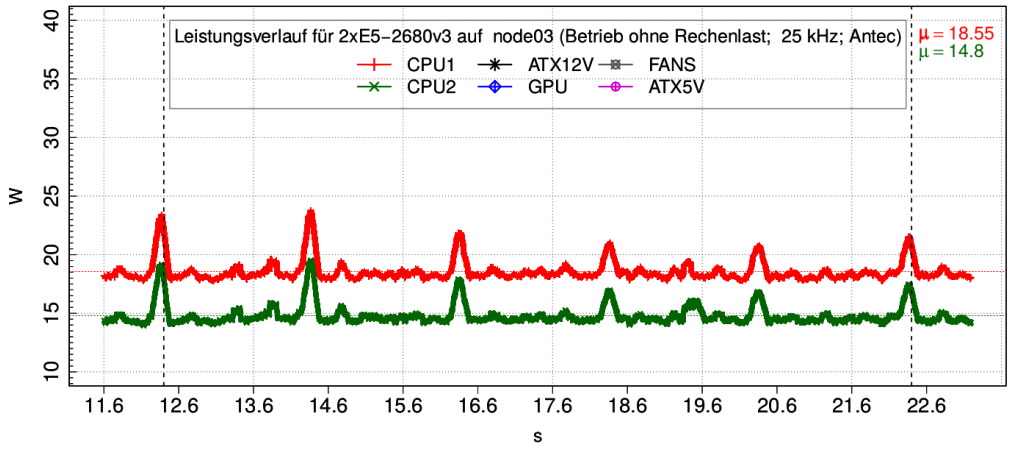

Abb. D.4.: Verlauf der elektrischen Leistung in Messkreisen mit zwei Prozessoren *Haswell* und mit dem Schaltnetzteil *Antec* während des Betriebes ohne Rechenlast. Die Messfrequenz beträgt 25 kHz. Für das Diagramm wurden die Messdaten mit dem Mittelwert-Filter $n_l = n_r = 1250$ auf das Spannungssignal und auf das Stromsignal angewendet.

III. Spannungs- und Stromverlauf (Ivy Bridge)

Abb. D.5 stellt den Stromverlauf des ersten Prozessor *Ivy Bridge* des Rechenknotens „node02“ und dessen Speichermodule für 10 ms während des Betriebes ohne Rechenlast dar. Im Unterschied zum Verlauf der elektrischen Leistung in Abb. 3.2 wurde ein Zeitabschnitt der Messung dargestellt, während dessen die Stromstärke durchschnittlich um 10 % kleiner als im Falle der Abb. 3.2 war. Es ist zu erkennen,

dass die Periode der Oszillation zweimal größer geworden ist.

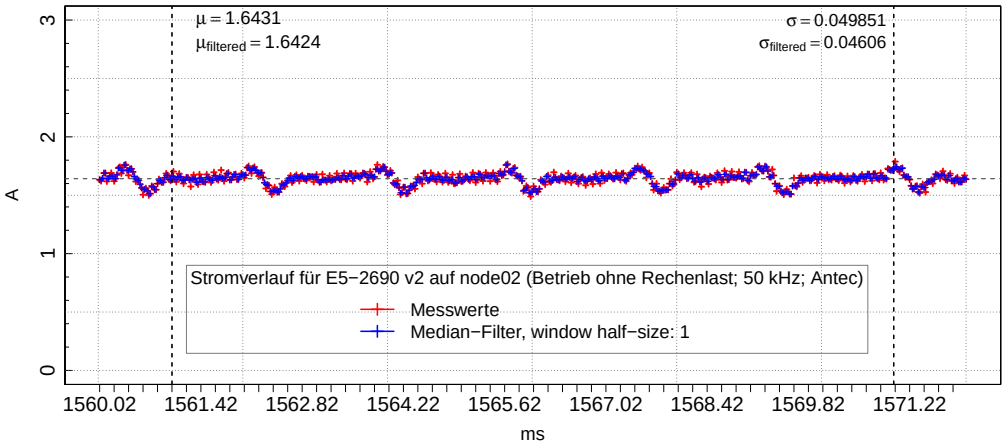

Abb. D.5.: Stromverlauf des ersten Prozessors E5-2690v2 auf *node02* während des Betriebes ohne Rechenlast. Der durchschnittliche Stromwert beträgt 1.64 A. Die Messfrequenz beträgt 50 kHz.

Die Oszillationen sind nicht mehr zu erkennen, wenn anstatt eines Schaltnetzteiles ein Lithium-Polymer-Akkumulator verwendet wird. Dies ist deutlich in Abb. D.6 zu erkennen.

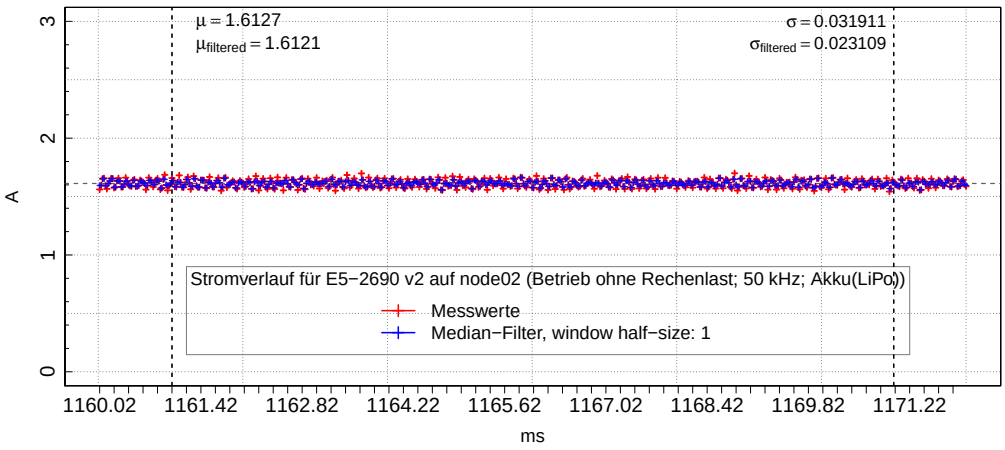

Abb. D.6.: Stromverlauf im Messkreis mit einem Prozessor *Ivy Bridge* und mit dem Akkumulator während des Betriebes ohne Rechenlast. Sowohl der Spannung- als auch der Stromverlauf sind gleichzeitig mit der Frequenz 50 kHz aufgezeichnet.

In Abb. D.7 ist der Verlauf der elektrischen Leistung der beiden Prozessoren des Rechenknotens „node02“ für das Zeitintervall von 10 s dargestellt. Im Unterschied zum

Prozessor *Haswell* musste der Mittelwert-Filter mit einem kleineren *Patch-Window* $n_l = n_r = 12$ angewendet werden, um die Änderungen in der elektrischen Leistung zu erkennen. Ein anderer Unterschied ist, dass der zweite Prozessor mehr Strom als der Erste verbraucht.

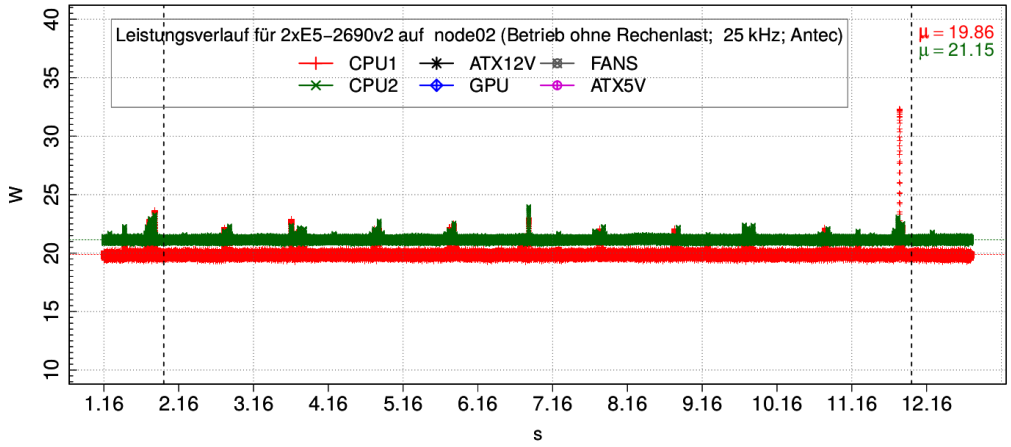

Abb. D.7.: Verlauf der elektrischen Leistung in Messkreisen mit zwei Prozessoren *Ivy Bridge* und mit dem Schaltnetzteil *Antec* während des Betriebes ohne Rechenlast. Die Messfrequenz beträgt 25 kHz. Für das Diagramm wurden die Messdaten mit dem Mittelwert-Filter $n_l = n_r = 12$ auf das Spannungssignal und auf das Stromsignal angewendet.

IV. Akkumulator als Stromquelle

IV.I. Prozessor Ivy Bridge

In diesem Teil vergleiche ich die Messresultate für das Schaltnetzteil *Antec* und dem Lithium-Polymer-Akkumulator *Akku(LiPo)*. Als Analysewerkzeug werden genau so, wie im Kapitel 2, die Histogramme für die Stichproben von 500 Messungen verwendet. 500 Messungen entsprechen einem Zeitintervall von 10 ms. In Abb. D.8 ist die Verteilung der Messdaten für das Schaltnetzteil *Antec* links und für den Lithium-Polymer-Akkumulator rechts dargestellt. Das Schaltnetzteil verursacht die wellenartigen Oszillationen mit der Standardabweichung $\sigma_{Antec} = 0.09830$. Die Messdaten sind im Bereich von $3 * \sigma_{Antec} = 0.2949$ oder $\pm 2.44\%$ verstreut. Das ist um 5.76-mal schlechter als das Resultat der Messung für den Hochlast-Widerstand (siehe das rechte Diagramm in Abb. B.11) und noch mal um 40-mal schlechter, wenn der Hochlast-Widerstand an den Akkumulator angeschlossen ist.

In Abb. D.8 rechts ist das Histogramm für den Spannungsverlauf im Messkreis mit dem Akkumulator dargestellt. Die Verstreuung der Messdaten reduziert sich um 38.11% auf $3 * \sigma_{Akku} = 0.1825$ und die Häufigkeitsdichte nähert sich stark der entsprechenden Normalverteilung. Allerdings ist die Verstreuung immerhin noch um 2.59 schlechter als im Fall vom Hochlast-Widerstand. Für den Vergleich muss die Standardabweichung der Stichprobe für den Hochlast-Widerstand mit dem Koeffizient $c_{divider} = 45.580$ skaliert (siehe Abschnitt IX auf Seite 131).

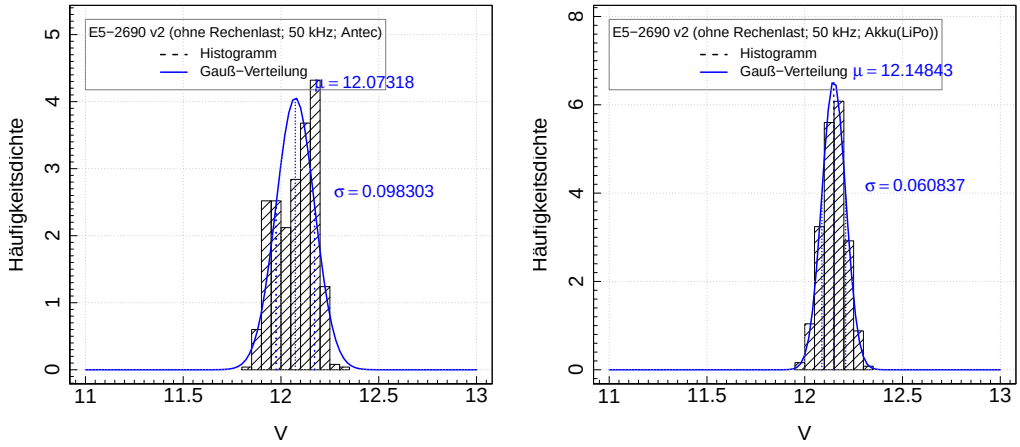

Abb. D.8.: Histogramme des Spannungsverlaufs im Messkreis mit dem Prozessor *Ivy Bridge* und mit dem Schaltnetzteil *Antec* (links) und mit dem Akkumulator (rechts)

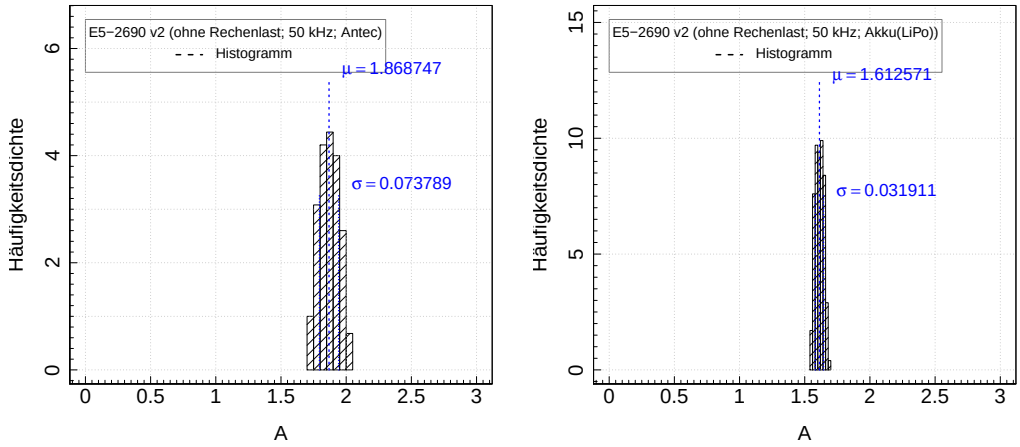

Abb. D.9.: Histogramme des Stromverlaufs im Messkreis mit dem Prozessor *Ivy Bridge* und mit dem Schaltnetzteil *Antec* (links) und mit dem Akkumulator (rechts)

In Abb. D.9 sind die Histogramme für den Stromverlauf dargestellt. Die Standardabweichung der Messproben verringert sich von $\sigma_{Antec} = 0.073\,79$ um mehr als zweimal auf $\sigma_{Akku} = 0.031\,91$. Im Vergleich zur Messung vom Hochlast-Widerstand unterscheidet sich die Standardabweichung um 11.4-mal.
Es ist zu erkennen, dass die Asymmetrie des Spannungssignals durch das Schaltnetzteil verursacht wird. Wenn das Schaltnetzteil mit einem Akkumulator ersetzt wird, werden die Störsignale stark reduziert.

IV.II. Prozessor Haswell

Wie zu erwarten ist, gibt es keine großen Unterschiede im Spannungsverlauf des Prozessors *Haswell* zwischen dem Schaltnetzteil und dem Akkumulator. Es ist deutlich in Abb. D.10 zu erkennen. Die rasanten Änderungen in der elektrischen Leistung des Prozessors verursachen einen Spannungsabfall (siehe Abschnitt 3.1.1 auf Seite 20).

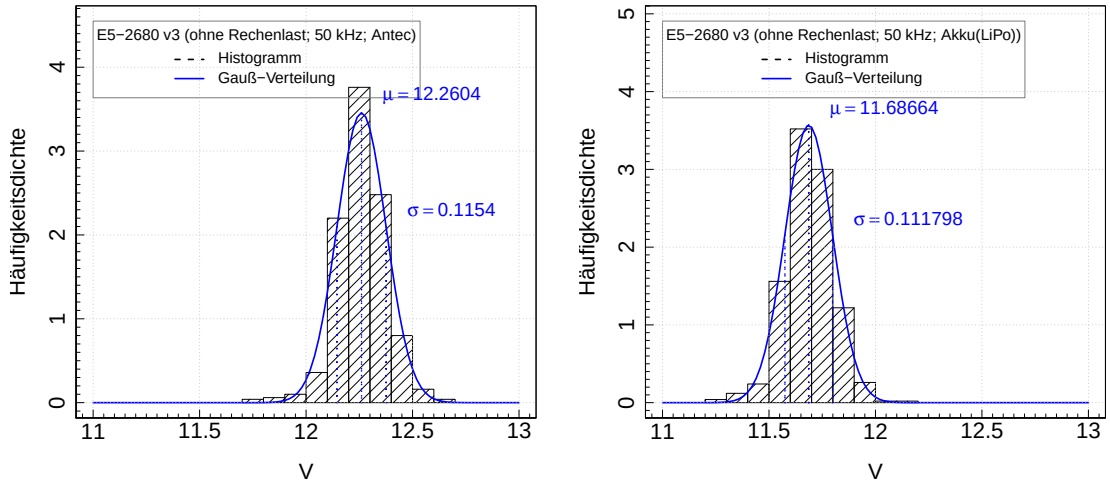

Abb. D.10.: **Links:** Histogramm des Spannungsverlaufs im Messkreis mit dem Prozessor *Haswell* und mit dem Schaltnetzteil *Antec*. **Rechts:** Histogramm des Spannungsverlaufs im Messkreis mit dem Prozessor *Haswell* und mit dem Akkumulator.

Die Stromprofile für *Haswell* mit Antec und *Akku* kann der Leser in Abb. D.11 sehen. Es gibt zwar einige Unterschiede in den Histogrammen, die Verläufe unterscheiden sich aber kaum. Es deutet daraufhin, dass die Störungen vom Prozessor und *VRM* etwa der gleichen Ordnung sind wie die vom Schaltnetzteil.

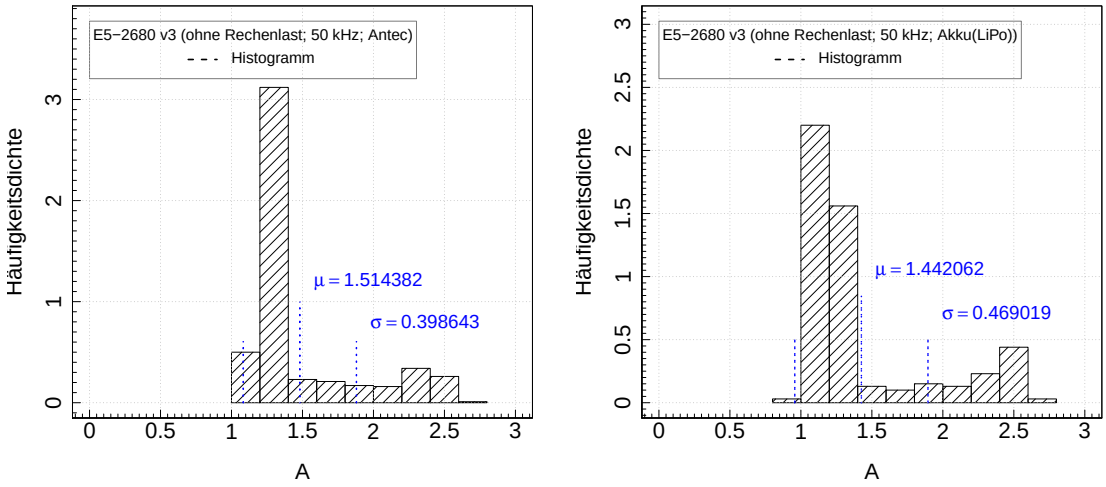

Abb. D.11.: **Links:** Histogramm des Stromverlaufs im Messkreis mit dem Prozessor *Haswell* und mit dem Schaltnetzteil *Antec*. **Rechts:** Histogramm des Stromverlaufs im Messkreis mit dem Prozessor *Haswell* und mit dem Akkumulator.

Bei der Verwendung des Lithium-Polymer-Akkumulators wurde eine Reduzierung des durchschnittlichen Stromverbrauchs registriert. Die Messungen zeigten, dass der Stromverbrauch vom Prozessor *Ivy Bridge* um $\sim$ 10% senkte und vom Prozessor *Haswell* um $\sim$ 5%.

Erste Versuche laufen bereits, die Stromversorgung eines Hochleistungsrechners mit dem DC-Strom vollständig zu realisieren, wie es zum Beispiel im Jahr 2008 in „Lawrence Berkeley National Laboratory“ gemacht wurde [73]. Die Resultate zeigten eine Erhöhung der Energieeffizienz eines Versuchssystems um bis zu 28 %. Die Autoren weisen darauf hin, dass die bessere Energieeffizienz durch die Abschaffung der Verluste bei der AC/DC-Konvertierung erreicht wurde. Die Gründe für Erhöhung der Energieeffizienz, die in dieser Arbeit erläutert wurde, müssen in dieser Arbeit folgenden Untersuchungen geklärt werden.

V. Elektrischen Leistung der Rechenknoten

Abb. D.12 stellt den Verlauf der elektrischen Leistung der Messgruppen des Testclusters dar (siehe Anhang A auf Seite 119). Die Messungen wurden mit der Frequenz 12.5 kHz durchgeführt. 125 davon sind zwischen zwei gestrichelten vertikalen Linien gezeichnet. Die entsprechenden Durchschnittswerte μ sind im rechten oberen Eck zu

finden.
In Abb. D.12 oben sind die Daten für vier Messgruppen des Rechenknotens „node01“ zusammengefasst: *CPU1*, *CPU2*, *GPU* und *ATX12V*. Die durchschnittliche elektrische Leistung des ersten Prozessors und dessen Speichermodules beträgt 21.62 W. Die Leistung der Messgruppe *CPU2* beträgt 21.89 W. Die Stromversorgung für den Akzelerator *TeslaK40* wird aus zwei Quellen realisiert. Die Versorgung erfolgt über ein 12-Volt-ATX-Stromkabel und über ein 8-Pin-GPU-Stromkabel. Der 12-Volt ATX-Bus wird von zwei PCIe-Karten geteilt: vom Akzelerator *TeslaK40* und vom InfiniBand-Adapter. Die durchschnittliche Leistung der Messgruppe *ATX12V* beträgt 7.54 W. Das Stromkabel 8-Pin-GPU versorgt den Akzelerator *TeslaK40* mit 9.06 W.
In Abb. D.12 Mitte ist die elektrische Leistung der Messgruppen des Rechenknotens „node02“ dargestellt. Das Diagramm zeigt den Leistungsverlauf von zwei neuen Gruppen: *ATX5* und *FANS*. Die Messgruppe *ATX5* verbraucht im Durchschnitt 13.02 W. Diese Messgruppe beinhaltet die Hardwarekomponenten der Hauptplatine, derer Betriebsspannung 5 V ist. Die durchschnittliche elektrische Leistung des Gehäuse-Lüfters (spezifiziert mit 12 V DC, 1.73 A) und von zwei CPU-Lüfter wird unter der Bezeichnung *FANS* angegeben. Die Lüfter werden nicht gesteuert und laufen mit einer hohen und konstanten Rotationsgeschwindigkeit. Grund dafür ist, dass die Stromversorgung mit einem zusätzlichen Adapter realisiert ist, sonst würden die Lüfter die Energie aus der gleichen Quelle wie die Prozessoren ziehen. Das Signal des Gehäuse-Lüfters ist deutlich höher als das Signal des CPU-Lüfters. Die elektrische Leistung des CPU-Lüfters beträgt 3.12 W (experimentell bestimmt), alle Lüfter zusammen brauchen 20.79 W. Die große Variation der Leistung ist durch die Arbeitsweise eines Lüftermotors verursacht. Die Rotationsgeschwindigkeit des Lüfters ist dabei proportional zu der Hauptfrequenz des Signals. Die Rotationsgeschwindigkeit ist nicht bekannt.
Mit Hilfe der Messgruppe *ATX12V* kann die elektrische Leistung des Akzelerators *TeslaK40* des Rechenknotens *node01* geschätzt werden, da die beiden Rechenknoten mit den identischen Hautplatinen und den identischen InfiniBand-Adapter ausgestattet sind. Es ist anzunehmen, dass der InfiniBand-Adapter 4.80 - Δ_{PCIe} W braucht. Der Anteil Δ_{PCIe} ist unbekannt und steht für den eigenen Verbrauch des *PCIe*-Interfaces. Daraus folgt, dass der Anteil der elektrischen Leistung des Akzelerators *TeslaK40* in der Messgruppe *ATX12V* $7.32 - 4.80 = 2.52$ W beträgt.
Abb. D.12 unten stellt die elektrische Leistung der Messgruppen des Rechenknotens „node03“ dar. Es ist deutlich zu erkennen, dass sich die elektrische Leistung der beiden Prozessoren voneinander unterscheiden und sich im Millisekunden-Takt ändern. Die elektrische Leistung des Akzelerators *TeslaK80* ist etwa doppelt so hoch

wie die elektrische Leistung des Akzelerators *TeslaK40*. Die Tesla-Karten müssen auch im Leerlauf von dem vorinstallierten Treiber kontrolliert werden. Die elektrische Leistung wird durch das Setzen einer niedrigen Frequenz der Akzelerator-Einheiten reduziert. Wenn der Treiber inaktiv ist (*non-persistent Mode*), werden der GDDR5-Speicher und die Streaming-Multi-Prozessoren (SM) mit einer hohen Frequenz getaktet. Für den GDDR5-Speicher beträgt die höchste Frequenz 2.505 GHz. Die SM-Frequenz ist mit 875 MHz spezifiziert. In *persistent Mode* werden die beiden Komponenten mit 324 MHz getaktet. Die niedrige Frequenz reduziert die elektrische Leistung der Messgruppe *GPU* von $\sim$ 33 W auf $\sim$ 21 W. Die elektrische Leistung der Messgruppe *ATX12V* erhöht sich dabei von $\sim$ 16 W auf $\sim$ 21 W. Im Rechenknoten sind drei *PCIe*-Karten installiert: ein Akzelerator *TeslaK80*, ein InfiniBand-Adapter und ein Ethernet-Adapter mit zwei Ethernet-Ports. Der Ethernet-Adapter verbraucht im Durchschnitt $\sim$ 4.2 W.
Es ist deutlich zu erkennen, dass die Verläufe der Messgruppen *CPU* und *GPU* korrelieren (siehe Abb. 3.1.1).

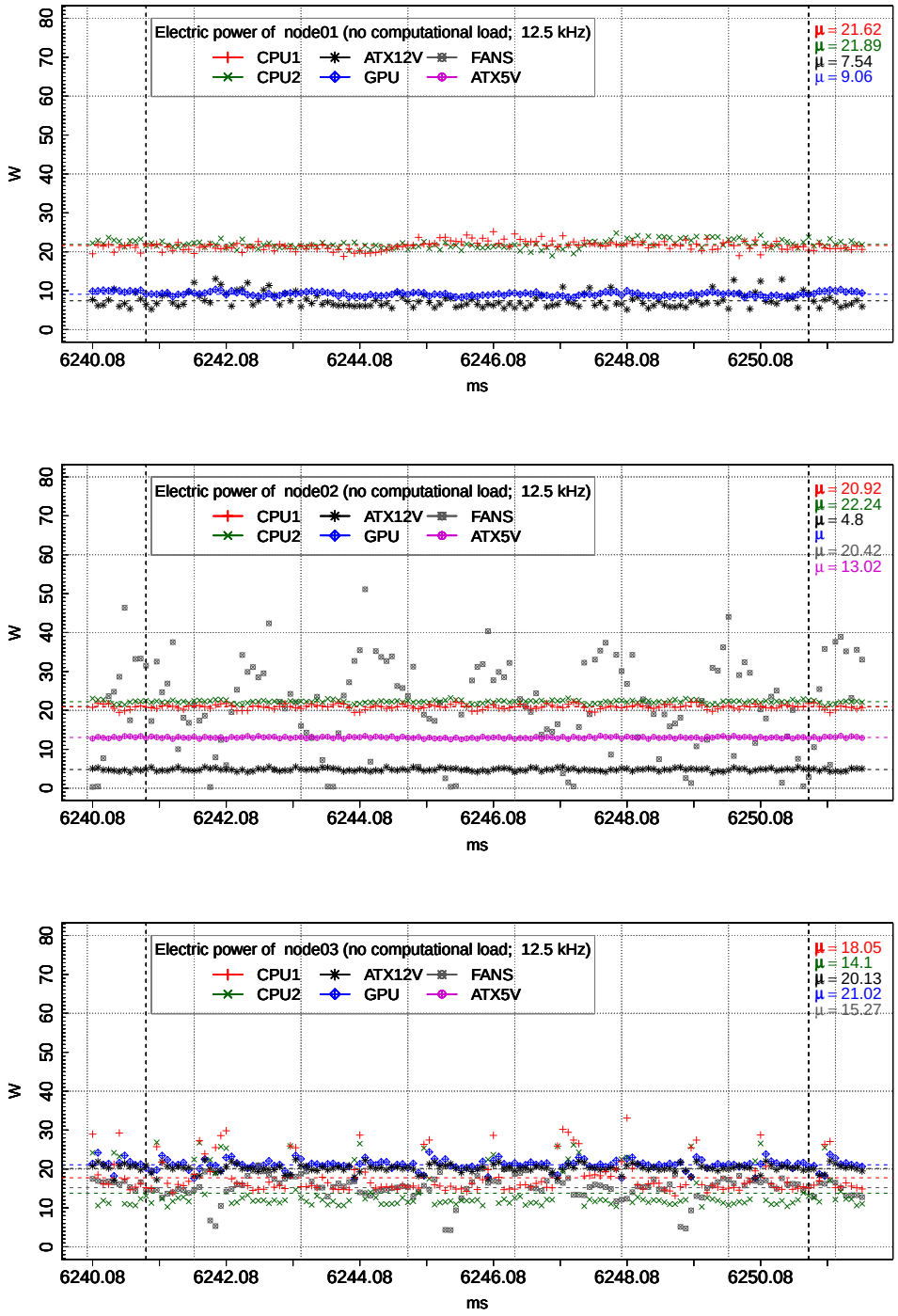

Abb. D.12.: Leistungsverlauf der Rechenknoten *node01*, *node02* und *node03* während des Betriebes ohne Rechenlast. Die Messfrequenz beträgt 12.5 kHz.

Tabelle Tab. D.3 fasst die durchschnittliche elektrische Leistung der oben betrachteten Messgruppen zusammen. Die Durchschnittswerte sind für eine Messung von 100 s gebaut. Zusätzlich ist die elektrische Leistung der Rechenknoten angegeben, die am AC-Kaltgerätekabel gemessen wurde.

Tab. D.3.: Durchschnittliche elektrische Leistung der Rechenknotenkomponenten während des Betriebes ohne Rechenlast

Rechenknoten	Messgruppe	Elektrische Leistung [W]
node01	CPU1	21.59
node01	CPU2	21.89
node01	ATX12V	7.32
node01	GPU	9.24
node01	Σ	60.04
node01	NODE01	106
node02	CPU1	20.89
node02	CPU2	22.28
node02	ATX12V	4.80
node02	ATX5V	13.02
node02	COOLER	20.79
node02	Σ	81.78
node02	NODE02	102
node03 non-persistent Mode	CPU1	18.71
node03 non-persistent Mode	CPU2	14.91
node03 non-persistent Mode	ATX12V	15.60
node03 non-persistent Mode	GPU	33.83
node03 non-persistent Mode	COOLER	15.49
node03 non-persistent Mode	Σ	98.54
node03 non-persistent Mode	NODE03	134
node03 persistent Mode	CPU1	18.81
node03 persistent Mode	CPU2	14.95
node03 persistent Mode	ATX12V	20.15
node03 persistent Mode	GPU	21.28
node03 persistent Mode	COOLER	15.52
node03 persistent Mode	Σ	90.71
node03 persistent Mode	NODE03	126

VI. Vergleich zwischen RAPL-Zähler und A/D-Messung

Die modernen Intel-Prozessoren sind mit den „Running Average Power Limit“ Zählern (RAPL) ausgestattet, aus denen der Stromverbrauch der Prozessoren und der Speichermodule ausgelesen werden kann. Im Betrieb ohne Rechenlast unterscheiden sich die aufgezeichneten RAPL-Zähler von der gemessenen elektrischen Leistung. Die Werte sind in der Tabelle Tab. D.4 zusammengefasst. Es ist anzumerken, dass der Prozessor *Ivy Bridge* kein eigenes Messsystem besitzt. Die elektrische Leistung wird innerhalb der „RAPL-Einheit“ des Prozessors approximiert. Bei dem *Haswell* Prozessor ist dagegen mit ein Messsystem integriert. Die Genauigkeit der RAPL-Zähler ist unbekannt. Nach „Package C-State Power Spezification“ [22] verbraucht ein Prozessor *Haswell E5-2680v3* im energiesparenden Zustand *C6-PState* 13 W. Das ist deutlich höher, als die vom Tool „Intel pcm“ aufgezeichnete elektrische Leistung.

Tab. D.4.: Vergleich zwischen der Ausgabe von RAPL-Zählern und den Messwerten während des Betriebes ohne Rechenlast auf *node03* und *node02*.

CPU	CPU+RAM W (RAPL)	CPU+RAM W (Messung)	ΔW
CPU1 node03	10.65	18.71	8.06
CPU2 node03	11.73	14.91	3.18
CPU1 node02	17.46	20.89	3.43
CPU2 node02	17.27	22.28	5.01

E. Approximation der Kernel-Operation *Add*

In diesem Anhang sind die zusätzlichen Details zum Verfahren zur Approximation der Rechenleistung und der elektrischen Leistung der Kernel-Operation *Add* zusammengefasst. Das Verfahren ist in Kapitel 4 ausführlich erklärt.

I. Fall des Haltens der Daten im Cache

Im Folgenden betrachten wir die gemessene Rechenleistung für den Fall $l = L1$. Wir verwenden die durchschnittlichen Werte der gemessenen Rechenleistung. Der Koeffizient $\beta_{0\,l,p}$ in Gl. (4.6) wird auf Null gesetzt. Die Begründung dafür ist, dass die Rechenleistung mit der CPU-Frequenz 0.0 auch Null ist. Im ersten Schritt berechnet man die Koeffizienten $\beta_{1\,l,p,\gamma}$ für die unterschiedlichen Werte von $\gamma_{l,p}$. Der Exponent γ iteriert dabei über den Bereich von 0.0 bis 4.0 mit einem Schritt von 0.01. Die Approximation mit dem kleinsten Fehler im L2-Norm wird ausgewählt.
Tab. E.1 gibt die berechneten Koeffizienten und die Fehler der Interpolation an. Es ist deutlich zu erkennen, dass die Rechenleistung im Falle des Haltens der Daten im L1-Cache linear von der CPU-Frequenz abhängt, weil die Potenz $\gamma_{L1,p}$ für alle p annähernd gleich Eins ist.

Tab. E.1.: Koeffizienten der Rechenleistungs-Approximation $R_I(f)_{L1,p} = \beta_{1\,L1,p} \times f^{\gamma_{L1,p}}$ für *Ivy Bridge E5-2690 v2* und *Haswell E5-2680 v3* im Falle des Haltens der Daten im L1-Cache

p	*Ivy Bridge E5-2690 v2*				*Haswell E5-2680 v3*			
	$\beta_{1,L1,p}$	$\gamma_{L1,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L^2\,p}$	$\beta_{1,L1,p}$	$\gamma_{L1,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L^2\,p}$
1	0.170×10^1	1.0	0.174×10^{-2}	0.0195	0.256×10^1	1.0	0.762×10^{-2}	0.049
2	0.338×10^1	1.0	0.434×10^{-2}	0.107	0.518×10^1	0.99	0.103×10^{-2}	0.168
3	0.504×10^1	1.0	0.276×10^{-2}	0.081	0.763×10^1	1.0	0.690×10^{-2}	0.208
4	0.676×10^1	1.0	0.419×10^{-2}	0.208	0.108×10^2	1.01	0.642×10^{-2}	0.228
5	0.839×10^1	1.01	0.438×10^{-2}	0.227	0.126×10^2	1.01	0.416×10^{-2}	0.217
6	0.101×10^2	1.01	0.326×10^{-2}	0.188	0.152×10^2	1.01	0.431×10^{-2}	0.225
7	0.117×10^2	1.01	0.463×10^{-2}	0.359	0.176×10^2	1.01	0.379×10^{-2}	0.239
8	0.134×10^2	1.01	0.393×10^{-2}	0.266	0.201×10^2	1.01	0.498×10^{-2}	0.336
9	0.151×10^2	1.01	0.546×10^{-2}	0.314	0.227×10^2	1.01	0.508×10^{-2}	0.443
10	0.167×10^2	1.01	0.364×10^{-2}	0.30	0.251×10^2	1.01	0.429×10^{-2}	0.481
11	-	-	-	-	0.276×10^2	1.01	0.309×10^{-2}	0.283
12	-	-	-	-	0.301×10^2	1.01	0.173×10^{-2}	0.186

In Tab. E.2 sind die Koeffizienten der Rechenleistungs-Approximation im Falle des Haltens der Daten im L2-Cache über die Thread-Anzahl p angegeben. Die Rechenleistung steigt mit der CPU-Frequenz linear und kann mit den relativen Fehlern unter 1 % approximiert werden. Die größten Fehler sind rot dargestellt.

Tab. E.2.: Koeffizienten der Rechenleistungs-Approximation $R_I(f)_{L2,p} = \beta_{1\,L2,p} \times f^{\gamma_{L2,p}}$ für *Ivy Bridge E5-2690 v2* und *Haswell E5-2680 v3* im Falle des Haltens der Daten im L2-Cache

p	*Ivy Bridge E5-2690 v2*				*Haswell E5-2680 v3*			
	$\beta_{1,L2,p}$	$\gamma_{L2,p}$	$\epsilon_{abs\,p}$	$\epsilon_{rel\,p}$	$\beta_{1,L2:q,p}$	$\gamma_{L2,p}$	$\epsilon_{abs\,p}$	$\epsilon_{rel\,p}$
1	0.627	1.0	0.278×10^{-2}	0.262×10^{-2}	0.801	1.01	0.747×10^{-2}	0.710×10^{-2}
2	0.124×10^1	1.01	0.253×10^{-1}	0.842×10^{-2}	0.161×10^1	1.0	0.743×10^{-2}	0.256×10^{-2}
3	0.190×10^1	1.0	0.204×10^{-1}	0.507×10^{-2}	0.238×10^1	1.0	0.112×10^{-1}	0.195×10^{-2}
4	0.250×10^1	1.0	0.366×10^{-1}	0.542×10^{-2}	0.319×10^1	1.0	0.823×10^{-1}	0.155×10^{-2}
5	0.313×10^1	1.0	0.193×10^{-1}	0.236×10^{-2}	0.400×10^1	1.0	0.432×10^{-1}	0.429×10^{-2}
6	0.375×10^1	1.0	0.532×10^{-1}	0.541×10^{-2}	0.481×10^1	1.0	0.157×10^{-1}	0.166×10^{-2}
7	0.440×10^1	1.0	0.596×10^{-1}	0.496×10^{-2}	0.557×10^1	1.0	0.950×10^{-2}	0.802×10^{-3}
8	0.499×10^1	1.00	0.462×10^{-1}	0.298×10^{-2}	0.637×10^1	1.0	0.330×10^{-1}	0.204×10^{-2}
9	0.565×10^1	1.01	0.929×10^{-1}	0.491×10^{-2}	0.716×10^1	1.0	0.151×10^{-1}	0.835×10^{-3}
10	0.623×10^1	1.00	0.805×10^{-1}	0.449×10^{-2}	0.796×10^1	1.0	0.193×10^{-1}	0.139×10^{-2}
11	-	-	-	-	0.876×10^1	1.0	0.246×10^{-1}	0.125×10^{-2}
12	-	-	-	-	0.953×10^1	1.0	0.557×10^{-1}	0.239×10^{-2}

In den Diagrammen der Abbildung Abb. E.1 sind die Koeffizienten der Rechenleistungs-Approximation $R_I(f)_{L2\,p} = \beta_{1\,L2,p} \times f^{\gamma_{L2,p}}$ für den Prozessor *Ivy Bridge* links und für den Prozessor *Haswell* rechts dargestellt. Zusätzlich ist die Interpolation der Koeffi-

zienten β_1 (dunkelgrün markiert) und γ (blau markiert) mit den Kurven angezeigt. Wie im Falle des Haltes der Daten im L1-Cache, ergibt sich aus dem Verlauf von β_1 und γ, dass die Rechenleistung linear mit der Anzahl der aktiven Prozessorkerne und der CPU-Frequenz skaliert. Der Skalierungsfaktor des Prozessors *Haswell* ist deutlich höher als der Skalierungsfaktor des Prozessors *Ivy Bridge*.

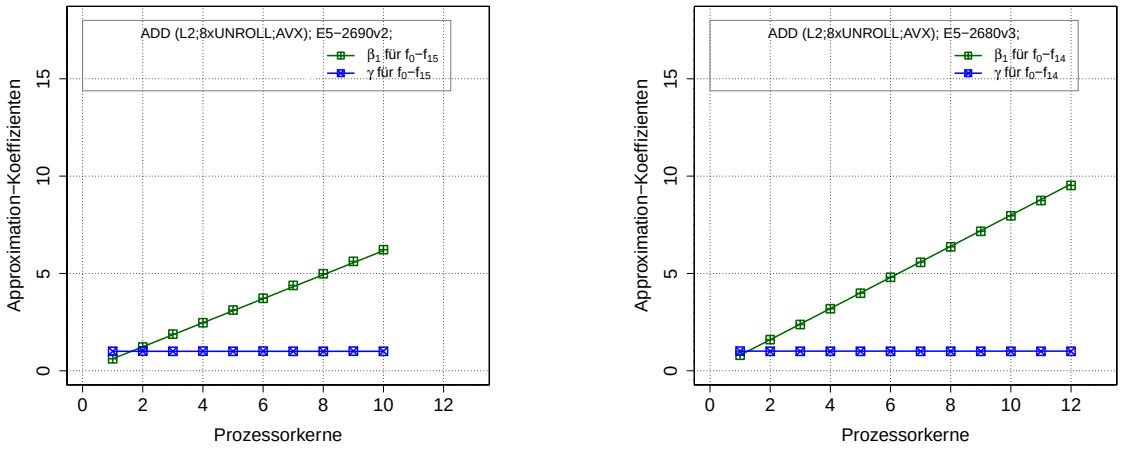

Abb. E.1.: Koeffizienten der Rechenleistungs-Approximation $R_I(f)_{l2\,p} = \beta_{1\,l2,p} \times f^{\gamma_{l2,p}}$ im Falle des Haltens der Daten im L2-Cache

In Tab. E.3 sind die Koeffizienten der Approximation der elektrischen Leistung angegeben. Der relative Fehler ist sowohl für *Ivy Bridge* als auch für *Haswell* kleiner als drei Prozent. Sowohl die Potenz als auch der statische Anteil der Approximation der elektrischen Leistung des Prozessors *Haswell* sind durchschnittlich höher als beim Prozessor *Ivy Bridge*.

Tab. E.3.: Koeffizienten der Approximation der elektrischen Leistung $P_I(f)_{L2,p} = \alpha_{1\,L2,p} \times f^{\lambda_{L1,p}}$ für *Ivy Bridge E5-2690 v2* und *Haswell E5-2680 v3* im Falle des Haltens der Daten im L2-Cache

p	*Ivy Bridge E5-2690 v2*					*Haswell E5-2680 v3*				
	$\alpha_{0,L2,p}$	$\alpha_{1,L2,p}$	$\lambda_{L2,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L2\,p}$	$\alpha_{0,L2,p}$	$\alpha_{1,L2,p}$	$\lambda_{L2,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L2\,p}$
1	22.0017	1.2434	2.32	0.236×10^{-1}	0.143×10^{-1}	26.4833	2.4767	1.91	0.958×10^{-2}	0.323×10^{1}
2	22.6668	2.0935	2.13	0.300×10^{-1}	0.234×10^{-1}	29.9517	1.7112	2.49	0.232×10^{-2}	0.120×10^{1}
3	22.9242	3.0709	2.01	0.271×10^{-1}	0.278×10^{-1}	30.5222	2.6105	2.36	0.246×10^{-2}	0.114×10^{1}
4	25.9064	2.8153	2.19	0.306×10^{-1}	0.223×10^{-1}	31.5211	3.3215	2.32	0.233×10^{-2}	0.148×10^{1}
5	26.1323	3.6872	2.12	0.243×10^{-1}	0.295×10^{-1}	31.4270	4.3796	2.22	0.617×10^{-2}	0.267×10^{1}
6	26.9276	4.2891	2.13	0.265×10^{-1}	0.337×10^{-1}	33.4614	4.523	2.29	0.226×10^{-2}	0.132×10^{1}
7	27.9744	4.8289	2.13	0.250×10^{-1}	0.372×10^{-1}	35.4327	4.557	2.39	0.159×10^{-2}	0.132×10^{1}
8	29.6246	4.9030	2.21	0.255×10^{-1}	0.448×10^{-1}	36.6089	4.905	2.42	0.195×10^{-2}	0.167×10^{1}
9	30.8524	5.3224	2.23	0.300×10^{-1}	0.501×10^{-1}	37.6641	5.6115	2.40	0.187×10^{-2}	0.177×10^{1}
10	32.026 15	5.6625	2.26	0.267×10^{-1}	0.553×10^{-1}	38.9732	6.1088	2.42	0.184×10^{-2}	0.187×10^{1}
11	-	-	-	-	-	40.3618	6.4502	2.45	0.183×10^{-2}	0.196×10^{1}
12	-	-	-	-	-	40.6092	7.2701	2.39	0.174×10^{-2}	0.221×10^{1}

In Tab. E.4 sind die Koeffizienten $\beta_{1\,L3*}$ und γ_{L3*} für die unterschiedliche Anzahl der Threads p angegeben. Die Rechenleistung des Prozessors *Ivy Bridge* wächst mit der Erhöhung der CPU-Frequenz linear. Der maximale relative Fehler der Approximation liegt bei 0.38 %.

Die Rechenleistung des Prozessors *Haswell*, verhält sich anders. Je mehr Prozessorkerne aktiv sind, desto langsamer steigt die Rechenleistung mit der CPU-Frequenz. Dies wird mit der Abnahme der Koeffizienten $\gamma_{L3,p}$ während der Steigung von p bestätigt. Der maximale relative Fehler liegt bei 3.4%.

Tab. E.4.: Koeffizienten der Rechenleistungs-Approximation $R_I(f)_{L3,p} = \beta_{1\,L3,p} \times f^{\gamma_{L3,p}}$ auf *Ivy Bridge* (links) und *Haswell* (rechts) im Falle des Haltens der Daten im L3-Cache

p	*Ivy Bridge E5-2690 v2*				*Haswell E5-2680 v3*			
	$\beta_{1,L3,p}$	$\gamma_{L3,p}$	ϵ_{rel}	ϵ_{L^2}	$\beta_{1,L3,p}$	$\gamma_{L3,p}$	ϵ_{rel}	ϵ_{L^2}
1	0.376	1.0	0.25×10^{-2}	0.73×10^{-2}	0.411	0.95	0.30×10^{-1}	0.29×10^{-1}
2	0.752	1.0	0.31×10^{-2}	0.18×10^{-1}	0.820	0.96	0.20×10^{-1}	0.48×10^{-1}
3	0.112×10^{1}	1.0	0.22×10^{-2}	0.17×10^{-1}	0.137×10^{1}	0.94	0.19×10^{-1}	0.13
4	0.149×10^{1}	1.0	0.27×10^{-2}	0.27×10^{-1}	0.185×10^{1}	0.92	0.22×10^{-1}	0.18
5	0.185×10^{1}	1.0	0.22×10^{-2}	0.28×10^{-1}	0.231×10^{1}	0.92	0.23×10^{-1}	0.24
6	0.211×10^{1}	1.0	0.29×10^{-2}	0.45×10^{-1}	0.279×10^{1}	0.92	0.21×10^{-1}	0.29
7	0.255×10^{1}	1.0	0.26×10^{-2}	0.49×10^{-1}	0.323×10^{1}	0.92	0.21×10^{-1}	0.34
8	0.292×10^{1}	1.0	0.34×10^{-2}	0.75×10^{-1}	0.370×10^{1}	0.91	0.21×10^{-1}	0.40
9	0.325×10^{1}	1.0	0.38×10^{-2}	0.84×10^{-1}	0.418×10^{1}	0.90	0.21×10^{-1}	0.45
10	0.359×10^{1}	1.0	0.28×10^{-2}	0.46×10^{-1}	0.467×10^{1}	0.89	0.23×10^{-1}	0.56
11	-	-	-	-	0.514×10^{1}	0.88	0.25×10^{-1}	0.68
12	-	-	-	-	0.562×10^{1}	0.87	0.30×10^{-1}	0.77

II. Hauptspeicher

II.I. Speicherzugriff im „Temporal“-Modus

Aus der Praxis kennen wir, dass die theoretische Bandbreite des Hauptspeichers oft viel kleiner als die tatsächliche Bandbreite ist. Besonders ausgeprägt ist es, wenn die Speicherung der Daten im „Temporal“-Modus durchgeführt wird. Sowohl GNU- als auch Intel-Compiler übersetzen die meisten Programme inklusive der Kernel-Operation *Add* mit den Instruktionen für den „Temporal“-Modus. „Temporal“ bedeutet, dass die Daten nicht direkt im Speicher geschrieben, sondern über alle Hierarchiestufen des Caches zum Speicher geschoben werden.[1] Einerseits liegt der Grund dafür an der Architektur und der Funktionsweise der Speichermodule und andererseits liegt es daran, dass nicht alle Operationen während der Ausführung überlappend ausgeführt werden können. Im weiteren Verlauf wird die Ausführung der Kernel-Operation *Add* betrachtet, wenn die gelesenen und geschriebenen Daten so groß sind, dass sie nicht in den Cache passen. Daher müssen diese über alle Speicher-Hierarchiestufen in beide Richtungen transportiert werden.

II.II. Rechenleistung

Im Abschnitt 4.5.4.1 wurde gezeigt, dass sich die Rechenleistung des Prozessors *Ivy Bridge* über die CPU-Frequenz f nicht nach Gl. (4.6) direkt interpolieren lässt. In der Tabelle Tab. E.5 sind die entsprechenden Koeffizienten und die Interpolationsfehler für alle p aufgelistet. Während der relative Fehler für den Prozessor *Haswell* klein ist, steigt dieser auf mehr als zehn Prozente für den Prozessor *Ivy Bridge*.

[1] Im „Non-Temporal“-Modus werden die Daten direkt im Speicher geschrieben, ohne eine temporale Kopie davon in den Caches zu hinterlassen[20].

Tab. E.5.: Koeffizienten der Rechenleistungs-Approximation $R_I(f)_{RAM,p} = \beta_{1\,RAM,p} \times f^{\gamma_{RAM,p}}$ im Falle des Haltens der Daten im Hauptspeicher. Der Interpolationsfehler für den Prozessor *Ivy Bridge* übersteigt zehn Prozent.

p	*Ivy Bridge E5-2690 v2*				*Haswell E5-2680 v3*			
	$\beta_{1,RAM,p}$	$\gamma_{RAM,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L_2\,p}$	$\beta_{1,RAM:q,p}$	$\gamma_{RAM,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L_2\,p}$
1	0.2412	0.73	0.108×10^{-1}	0.279×10^{-1}	2.11×10^{-1}	0.76	0.144×10^{-1}	0.532×10^{-1}
2	0.4538	0.70	0.293×10^{-1}	0.711×10^{-1}	4.92×10^{-1}	0.67	0.102×10^{-2}	0.102×10^{-2}
3	0.6473	0.64	0.646×10^{-1}	0.157	9.01×10^{-1}	0.41	0.308×10^{-1}	0.269×10^{-1}
4	0.820	0.53	0.899×10^{-1}	0.226	1.167	0.28	0.276×10^{-1}	0.204×10^{-1}
5	0.9472	0.44	0.106	0.270	1.369	0.19	0.281×10^{-1}	0.186×10^{-1}
6	0.106×10^{1}	0.39	0.115	0.283	1.526	0.12	0.235×10^{-1}	0.144×10^{-1}
7	0.1023×10^{1}	0.38	0.118	0.293	1.634	0.07	0.180×10^{-1}	0.102×10^{-1}
8	0.1033×10^{1}	0.37	0.118	0.292	1.705	0.03	0.100×10^{-2}	0.579×10^{-2}
9	0.1036	0.36	0.117	0.288	1.737	0.02	0.746×10^{-2}	0.421×10^{-2}
10	0.1039	0.35	0.113	0.277	1.757	0.01	0.218×10^{-2}	0.122×10^{-2}
11	-	-	-	-	1.761	0.01	0.566×10^{-2}	0.318×10^{-2}
12	-	-	-	-	1.762	0.01	0.403×10^{-2}	0.227×10^{-2}

Wie im Falle des L3-Caches, unterscheiden sich die „Uncore„- und die CPU-Frequenzen. Abb. E.2 zeigt das.

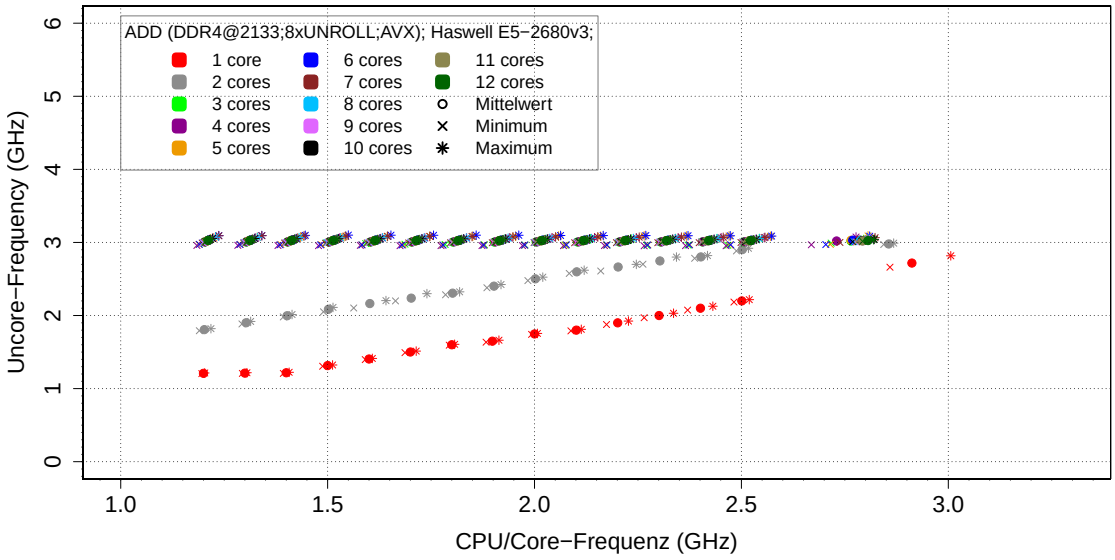

Abb. E.2.: Abhängigkeit zwischen der „Uncore„-Frequenz und der CPU-Frequenz im Prozessor *Haswell* im Falle des Haltens der Daten im Hauptspeicher

Hier zeigt sich im Vergleich zum Fall des Haltens der Daten im L3-Cache eine im Durchschnitt höhere „Uncore“-Frequenz (siehe Abb. 4.15).

Den Grund für das beobachtete Verhalten erklärt Abb. E.3, in der die diskrete Ableitung der gemessenen Rechenleistung nach Gl. (4.22) abgebildet ist. Der steile Abstieg der Ableitung für den Prozessor *Ivy Bridge* wiederholt sich für alle $p > 3$. Dieses Verhalten kann mit der verwendeten Interpolationsmethode nicht abgebildet werden.

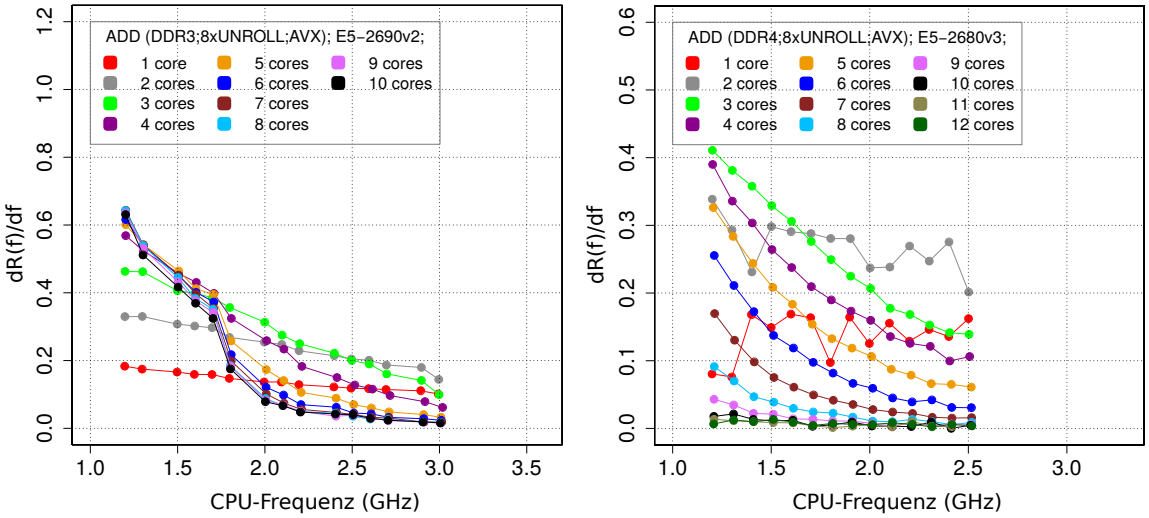

Abb. E.3.: Erste Ableitung der Rechenleistung nach der CPU-Frequenz im Falle des Haltens der Daten im Hauptspeicher

Daher wird der Definitionsbereich für die CPU-Frequenz des Prozessors *Ivy Bridge* in zwei disjunkte Teilmengen aufgeteilt: $F_1 \equiv [f_0; \ldots; f_4]$ und $F_2 \equiv [f_5; \ldots; f_{15}]$. Die gemessenen Werte der Rechenleistung werden für diese Definitionsbereiche getrennt interpoliert. Tab E.6 listet die neuen berechneten Koeffizienten auf.

Tab. E.6.: Koeffizienten der Rechenleistungs-Approximation $R_I(f)_{RAM,p} = \beta_{1\,RAM,p} \times f^{\gamma_{RAM,p}}$ für den Prozessor *Ivy Bridge E5-2690 v2*. Die gemessenen Werte der Rechenleistung werden für zwei disjunkte CPU-Frequenz-Definitionsbereiche getrennt interpoliert.

p	$\forall f \in F_1 \equiv [f_0; \ldots; f_4]$				$\forall f \in F_2 \equiv [f_5; \ldots; f_{15}]$			
	$\beta_{1,RAM,p}$	$\gamma_{RAM,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L_2\,p}$	$\beta_{1,RAM,p}$	$\gamma_{RAM,p}$	$\epsilon_{rel\,p}$	$\epsilon_{L_2\,p}$
1	0.2385	0.78	0.301×10^{-2}	0.1351×10^{-2}	0.2555	0.66	0.543×10^{-2}	0.101×10^{-1}
2	0.4473	0.78	0.163×10^{-2}	0.181×10^{-2}	0.4868	0.63	0.260×10^{-1}	0.332×10^{-1}
3	0.6333	0.76	0.491×10^{-2}	0.487×10^{-2}	0.7299	0.51	0.473×10^{-1}	0.728×10^{-1}
4	0.7931	0.71	0.642×10^{-2}	0.93×10^{-2}	0.9729	0.35	0.459×10^{-1}	0.743×10^{-1}
5	0.9111	0.65	0.725×10^{-2}	0.126×10^{-1}	0.1170×10^{1}	0.21	0.294×10^{-1}	0.629×10^{-1}
6	0.9646	0.62	0.764×10^{-2}	0.145×10^{-1}	0.1252×10^{1}	0.16	0.290×10^{-1}	0.526×10^{-1}
7	0.9787	0.628	0.812×10^{-2}	0.167×10^{-1}	0.1289×10^{1}	0.13	0.219×10^{-1}	0.482×10^{-1}
8	0.9882	0.61	0.888×10^{-2}	0.177×10^{-1}	0.1301×10^{1}	0.12	0.198×10^{-1}	0.441×10^{-1}
9	0.9908	0.60	0.918×10^{-2}	0.179×10^{-1}	0.1297×10^{1}	0.12	0.219×10^{-1}	0.437×10^{-1}
10	0.9940	0.59	0.823×10^{-2}	0.182×10^{-1}	0.1290×10^{1}	0.12	0.225×10^{-1}	0.412×10^{-1}

Die Rechenleistungs-Approximation über die CPU-Frequenz-Definitionsbereiche F_1 und F_2 werden in Abb. 4.18 über zwei getrennte Kurven angezeigt.

2. Approximation der Rechenleistung: Im vorangegangenen Abschnitt wurde gezeigt, dass die Aufteilung der CPU-Frequenz in den unterschiedlichen Bereichen die Genauigkeit der Approximation erheblich erhöhte. Im nächsten Schritt werden die berechneten Koeffizienten über die Anzahl der Prozessorkerne interpoliert. Wie im Falle des Haltens der Daten im L3-Caches für den Prozessor *Haswell* wird für die bessere Genauigkeit die Menge P der aktiven Prozessorkerne in mehrere Teilmengen aufgeteilt. Anhand der Diagramme für die Ableitung der Rechenleistung unterteile ich P in vier Teilmengen sowohl für *Ivy Bridge* als auch für *Haswell*:

- *Haswell:* $P \equiv [p_1; p_2] \cup [p_3; p_4] \cup [p_5; p_6] \cup [p_7; p_{10}]$
- *Ivy Bridge:* $P \equiv [p_1; p_2] \cup [p_3; p_4] \cup [p_5; p_7] \cup [p_8; p_{12}]$

Somit ergibt sich die Rechenleistungs-Approximation über die CPU-Frequenz und die Thread-Anzahl. Die Koeffizienten dazu sind in Tab. E.7 und E.8) aufgelistet.

Tab. E.7.: Koeffizienten der Rechenleistungs-Approximation $R_{II}(f,p)_{RAM} = (\tilde{\beta}_{1,RAM} \times p^{\tilde{\gamma}_{RAM}}) \times f^{(\tilde{\tilde{\beta}}_{1,RAM} \times p^{\tilde{\tilde{\gamma}}_{RAM}})}$ für den Prozessor *Ivy Bridge*

P	F	$\tilde{\beta}_{1,RAM}$	$\tilde{\gamma}_{RAM}$	$\tilde{\tilde{\beta}}_{1,RAM}$	$\tilde{\tilde{\gamma}}_{RAM}$	$\epsilon_{rel,RAM}$
$(p_1, ..., p_2)$	$(f_0, ..., f_4)$	0.2385	0.91	0.780	0.0	0.30×10^{-2}
$(p_1, ..., p_2)$	$(f_5, ..., f_{15})$	0.2555	0.93	0.660	-0.07	0.232×10^{-1}
$(p_3, ..., p_4)$	$(f_0, ..., f_4)$	0.2683	0.78	0.9893	-0.24	0.6642×10^{-2}
$(p_3, ..., p_4)$	$(f_5, ..., f_{15})$	0.2436	1.0	2.151	-1.31	0.361×10^{-1}
$(p_5, ..., p_6)$	$(f_0, ..., f_4)$	0.5506	0.31	0.9877	-0.26	0.7592×10^{-2}
$(p_5, ..., p_6)$	$(f_5, ..., f_{15})$	0.6461	0.37	2.310	-1.49	0.2×10^{-1}
$(p_7, ..., p_{10})$	$(f_0, ..., f_4)$	0.8914	0.05	0.7984	-0.13	0.123×10^{-1}
$(p_7, ..., p_{10})$	$(f_5, ..., f_{15})$	1.252	0.02	0.2286	-0.29	0.1678×10^{-1}

Tab. E.8.: Koeffizienten der Rechenleistungs-Approximation $R_{II}(f,p)_{RAM} = (\tilde{\beta}_{1,RAM} \times p^{\tilde{\gamma}_{RAM}}) \times f^{(\tilde{\tilde{\beta}}_{1,RAM} \times p^{\tilde{\tilde{\gamma}}_{RAM}})}$ für *Haswell*

P	F	$\tilde{\beta}_{1,RAM}$	$\tilde{\gamma}_{RAM}$	$\tilde{\tilde{\beta}}_{1,RAM}$	$\tilde{\tilde{\gamma}}_{RAM}$	$\epsilon_{rel,RAM}$
$(p_1, ..., p_2)$	$(f_0, ..., f_{14})$	0.2116	1.21	0.74	-0.14	0.053×10^{-1}
$(p_3, ..., p_4)$	$(f_0, ..., f_{14})$	0.3317	0.91	1.937	-1.37	0.341×10^{-1}
$(p_5, ..., p_7)$	$(f_0, ..., f_{14})$	0.562	0.55	17.830	-2.79	0.235×10^{-1}
$(p_8, ..., p_{12})$	$(f_0, ..., f_{14})$	1.394	0.1	51.308	-3.58	0.843×10^{-2}

In den Diagrammen der Abb. E.4 sind die Koeffizienten der Rechenleistungs-Approximation $R_I(f)_{RAM\,p} = \beta_{1\,RAM,p} \times f^{\gamma_{RAM,p}}$ für die Kernel-Operation *Add* auf *Ivy Bridge E5-2690 v2* (links) und *Haswell E5-2680 v3* (rechts) dargestellt. Die rote und lila Farbe markieren die Koeffizienten der höheren CPU-Frequenzen $f_5 \leq f \leq f_{15}$. Die schwarze und graue Farbe markieren die niedrigen CPU-Frequenzen $f_0 \leq f \leq f_4$. Anhand der Diagramme kann man nicht nur die beiden Prozessoren vergleichen, sondern auch einige Aussagen über die Skalierung der Prozessoren nach der CPU-Frequenz und den Prozessorkernen machen, die erst schwer aus den Messdaten ablesbar sind. Der Koeffizient γ zeigt die Skalierung nach der CPU-Frequenz und der Koeffizient β zeigt die Skalierung nach den Prozessorkernen. Während γ für *Ivy Bridge* niemals Null wird, nähert sich dieser Koeffizient für *Haswell* bei $p > 7$ sehr schnell der Null.

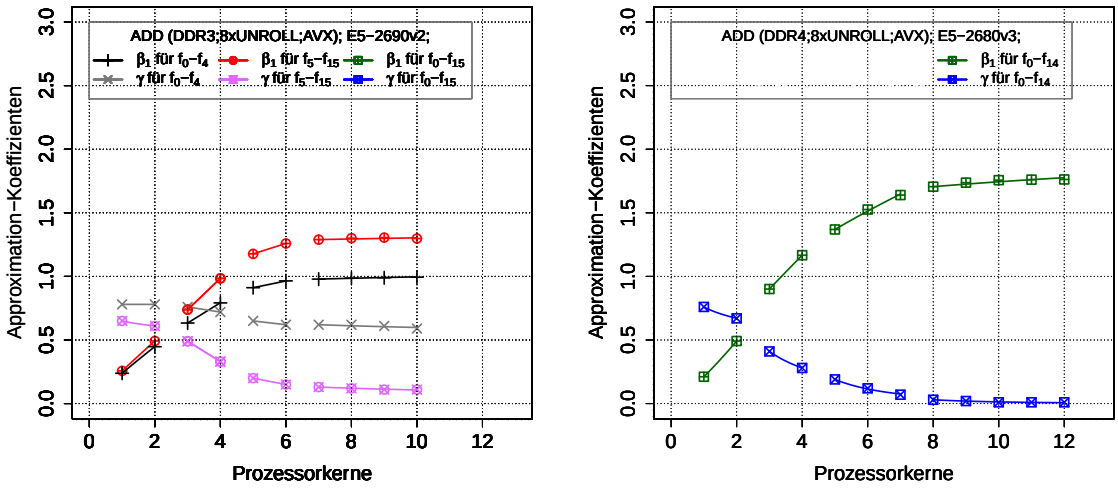

Abb. E.4.: Approximations-Koeffizienten für die Rechenleistungs-Approximation $R_I(f)_{RAM,p} = \beta_{1\,RAM,p} \times f^{\gamma_{RAM,p}}$ des Kernels *Add* auf *Ivy Bridge E5-2690 v2* (oben) und *Haswell E5-2680 v3* (unten) im Falle des Haltens der Daten im Hauptspeicher

II.III. Elektrische Leistung

Tab. E.9 listet die Koeffizienten der Approximation der elektrischen Leistung im Falle des Haltens der Daten im Hauptspeicher. Die Potenzen wurden für alle p einzeln berechnet.

Tab. E.9.: Koeffizienten der Approximation der elektrischen Leitung $P_I(f)_{RAM,p} = \alpha_{0\,L1,p} + \alpha_{1\,L1,p} \times f^{\lambda_{L1,p}}$ für *Ivy Bridge E5-2690 v2* und *Haswell E5-2680 v3*

p	*Ivy Bridge E5-2690 v2*					*Haswell E5-2680 v3*				
	$\alpha_{0,\mathrm{RAM,p}}$	$\alpha_{1,\mathrm{RAM,p}}$	$\lambda_{\mathrm{RAM,p}}$	ϵ_{relp}	ϵ_{L_2p}	$\alpha_{0,\mathrm{RAM,p}}$	$\alpha_{1,\mathrm{RAM,p}}$	$\lambda_{\mathrm{RAM,p}}$	ϵ_{relp}	ϵ_{L_2p}
1	25.506	1.914	2.1	0.23×10^{-1}	1.7	36.013	1.247	2.7	0.1×10^{-1}	1.2
2	27.421	3.76	1.84	0.27×10^{-1}	2.7	38.398	8.188	1.54	0.1×10^{-1}	1.9
3	28.816	5.845	1.64	0.22×10^{-1}	3.0	57.113	12.469	1.03	0.05×10^{-1}	0.9
4	33.494	5.298	1.76	0.24×10^{-1}	2.9	70.318	7.358	1.44	0.05×10^{-1}	1.0
5	35.017	5.994	1.77	0.24×10^{-1}	3.4	77.363	5.79	1.68	0.05×10^{-1}	11
6	36.628	6.284	1.82	0.24×10^{-1}	3.9	81.826	5.433	1.81	0.07×10^{-1}	1.4
7	38.309	6.24	1.9	0.26×10^{-1}	4.5	85.922	5.005	1.95	0.08×10^{-1}	1.7
8	40.051	6.123	1.99	0.26×10^{-1}	5.0	89.636	4.580	2.09	0.08×10^{-1}	1.7
9	40.729	6.502	2.01	0.27×10^{-1}	5.4	93.374	4.035	2.28	0.1×10^{-1}	1.7
10	42.247	6.443	2.08	0.26×10^{-1}	5.8	96.222	3.945	2.38	0.1×10^{-1}	2.2
11	-	-	-	-	-	98.711	3.802	2.49	0.11×10^{-1}	2.7
12	-	-	-	-	-	100.400	4.107	2.49	0.13×10^{-1}	3.0

Tab. E.10 listet die Koeffizienten der Approximation der elektrischen Leistung im Falle des Haltens der Daten im Hauptspeicher, wenn die Potenz p nach der L2-Norm der einzelnen Gruppen ausgewählt wird.

Tab. E.10.: Die Koeffizienten der Approximation der elektrischen Leistung $P_I(f)_{RAM,p} = \alpha_{0\,L1,p} + \alpha_{1\,L1,p} \times f^{\lambda_{L1,p}}$ für *Ivy Bridge E5-2690 v2* und *Haswell E5-2680 v3*

p	*Ivy Bridge E5-2690 v2*					*Haswell E5-2680 v3*				
	$\alpha_{0,\mathrm{RAM,p}}$	$\alpha_{1,\mathrm{RAM,p}}$	$\lambda_{\mathrm{RAM,p}}$	ϵ_{relp}	ϵ_{L_2p}	$\alpha_{0,\mathrm{RAM,p}}$	$\alpha_{1,\mathrm{RAM,p}}$	$\lambda_{\mathrm{RAM,p}}$	ϵ_{relp}	ϵ_{L_2p}
1	24.410	2.736	1.81	3.9×10^{-1}	2.3	33.728	2.89	1.91	0.5×10^{-1}	3.1
2	27.235	3.904	1.81	2.6×10^{-1}	2.7	41.989	5.286	1.91	0.27×10^{-1}	2.8
3	32.179	3.266	2.11	0.39×10^{-1}	4.7	67.983	3.213	2.11	0.29×10^{-1}	4.0
4	35.947	3.452	2.11	0.32×10^{-1}	4.0	75.258	3.374	2.11	0.13×10^{-1}	2.6
5	37.73	3.956	2.11	0.34×10^{-1}	4.5	80.223	3.559	2.11	0.09×10^{-1}	2.0
6	39.127	4.416	2.11	0.32×10^{-1}	4.9	83.848	3.886	2.11	0.09×10^{-1}	1.9
7	40.198	4.842	2.11	0.33×10^{-1}	5.1	86.999	4.196	2.11	0.097×10^{-1}	1.9
8	41.169	5.304	2.11	0.284×10^{-1}	5.2	89.769	4.482	2.11	0.09×10^{-1}	1.7
9	41.733	5.77	2.11	0.3×10^{-1}	5.5	92.265	4.839	2.11	0.1×10^{-1}	2.0
10	42.559	6.217	2.11	0.26×10^{-1}	5.8	94.4	5.254	2.11	0.14×10^{-1}	2.8
11	-	-	-	-	-	96.075	5.673	2.11	0.18×10^{-1}	3.8
12	-	-	-	-	-	97.549	6.131	2.11	0.19×10^{-1}	4.2

F. Optimierter Pipeline-Verlauf (ECM)

Die im Abschnitt 5.1.2.3 modellierte Pipeline für die Ausführung der Kernel-Operation *Add* enthält Lücken, die sogenannten „Bubbles“. In diesem Teil wird versucht, diese Lücken in der Pipeline zu eliminieren, um den Durchsatz der Pipeline zu erhöhen. Die Lücke in der Zeile „*Port 1*“ der Tabelle Tab. 5.1 entstand, weil der Port 2 sowohl für die Speicherung als auch für das Laden benutzt wurde. Der Grund dafür ist, dass die im Befehl „STA“ verwendete Adressierungsart für das Feld $c[]$ indirekt ist (siehe den Abschnitt 5.1.2.2). Um das zu vermeiden, wird die entsprechende direkte Adresse mit einem separaten Befehl „LEA“ berechnet. Dieser Mikrobefehl kann durch den Port 5 durchgeführt werden. Dies ermöglicht parallel zu zwei Ladebefehlen die Speicherung der Daten durch den Port 7 (siehe Abschnitt 5.1.2.1). Der Quelltext F.1 zeigt eine neue nach dem ECM-Modell schnellere Implementierung des Kernels *Add*:

Quelltext F.1: Implementierung des Kernels *Add* in der Assemblersprache mit einer absoluten Adressierung für die Speicherung eines AVX-Registers.

```
lea        (%r11,%rax,8), %r8
vmovapd    (%r13,%rax,8), %ymm0
vaddpd     (%r12,%rax,8), %ymm0, %ymm2
vmovupd    %ymm2, (%r8)
```

Die Tabelle Tab. F.1 zeigt den entsprechenden Ausführungsverlauf. Alle Daten, die für die Durchführung von zwei Vektoroperationen „ADD“ notwendig sind, werden zwischen dem L1-Cache und der Ausführungseinheit im Laufe von zwei Takten übertragen. Somit ergibt sich, dass die Ausführungszeit 2 $cycle/CL$ beträgt. Dies entspricht der Rechenleistung von $8/2$ $Flop/cycles = 4$ $Flop/cycle$ oder der Bandbreite von 96 $B/cycle$.

Tab. F.1.: Optimierter Verlauf der Kernel-Operation *Add* im Falle des Haltens der Daten im L1-Cache in einer *Haswell*-Pipeline in Takten der CPU-Frequenz

	Verlauf von Mikrobefehlen in der Pipeline über die Zeit						
	1	2	3	4	5	6	7
Port 2	$LD(a_{0-3})$	$LD(a_{4-7})$	$LD(a_{8-11})$	$LD(a_{12-15})$	$LD(a_{16-19})$	$LD(a_{20-23})$	$LD(a_{24-27})$
Port 3	$LD(b_{0-3})$	$LD(b_{4-7})$	$LD(b_{8-11})$	$LD(b_{12-15})$	$LD(b_{16-19})$	$LD(b_{20-23})$	$LD(b_{24-27})$
Port 1		$ADD(c_{0-3})$	$ADD(c_{4-7})$	$ADD(c_{8-11})$	$ADD(c_{12-15})$	$ADD(c_{16-19})$	$ADD(c_{20-23})$
Port 7			$STA(\tilde{c_{0-3}})$	$STA(\tilde{c_{4-7}})$	$STA(\tilde{c_{8-11}})$	$STA(\tilde{c_{12-15}})$	$STA(\tilde{c_{16-19}})$
Port 5	$LEA(c_{0-3})$	$LEA(c_{4-7})$	$LEA(c_{8-11})$	$LEA(c_{12-15})$	$LEA(c_{16-19})$	$LEA(c_{20-23})$	$LEA(c_{24-27})$

Die Ergebnisse der Ausführung der Kernel-Operation *Add* auf dem Prozessor *Haswell E5-2680v3* zeigen keine großen Unterschiede unabhängig davon, ob die Rechenkernkomponente „*Fast LEA*“ benutzt wurde oder nicht. Der genauere Grund dafür ist mir unbekannt.

G. Speichermodul-Architektur

In diesem Anhang sind die Basisprinzipien des Aufbaus eines SDRAM-Modules zusammengefasst, weil die dahinter stehende Technologie einen großen Einfluss auf die Rechen- und die elektrische Leistung der Prozessoren hat. Die detaillierte technische Anleitung zum Aufbau und zur Funktionsweise des Speichers kann der interessierte Leser z.B. in [38], [9] und [46] finden.

I. SDRAM-Modul

Ein Speichermodul besteht aus mehreren geschalteten Chips, die nach außen eine gemeinsame Schnittstelle zum integrierten Speichercontroller (IMC) haben. Die Modulbezeichnung „*Dual Rank DDR3-SDRAM 4 GB 512M×72*“ bedeutet, dass das 4 GB-Modul insgesamt aus 16 DDR3-SDRAM-Chips gebaut ist.[1] Die Chips werden gleichzeitig identisch angesteuert. Die Steuerung läuft synchron zur steigenden Flanke der 4-fachen Taktfrequenz des Speicherkerns. Die IO-Ausgabe läuft dagegen mit einer 4-fachen Frequenz und liefert die Daten sowohl auf den steigenden als auch auf den fallenden Taktflanken des Taktsignals.
In Abb. G.1 links ist eine von zwei Seiten, der sogenannte Speicherrank, eines Moduls dargestellt. Acht SDRAM-Chips und ein zusätzlicher ECC-Chip sind auf der Skizze einer Seite der Modul-Platine zu sehen. Der ECC dient zur Erkennung der 1-Bit-Fehler, die während der Übertragung zwischen den SDRAM-Chips und dem Speichercontroller auftreten können. Mithilfe vom ECC-Protokoll werden die 1-Bit-Fehler erkannt und korrigiert.
Jeder abgebildete Chip besitzt einen 8-Bit-IO-Bus zum IMC. Daraus ergibt sich in der Bezeichnung die Zahl 72. Eine 8-Bit-IO-Leitung wird entweder von einem Chip des ersten Speicherranks oder von einem Chip des zweiten Speicherranks verwendet.
In Abb. G.1 rechts ist ein DDR3-SDRAM-Chip schematisch dargestellt. Der Chip ist in drei logischen Einheiten aufgeteilt: einen Speicherkern, einen IO-Puffer und eine

[1] Die Chips-Anzahl ist mit der Gleichung $4\ [\mathrm{GB}] / \frac{512}{2}\ [\mathrm{MB}]$ berechnet. Der angegebene Wert von 512 MB wird durch zwei geteilt, weil das Modul aus zwei Speicherranks besteht.

Steuerungseinheit. Die Steuerungseinheit SDRAM-Logik ist ein Automat, der vom IMC in verschiedene Zustände versetzt wird, um den Datentransfer zwischen dem Chip und dem IMC aufrechtzuerhalten. Die dargestellten Busse der Breite von 64 bit werden in der Literatur als globale Leitungen bezeichnet.
Der Chip speichert 256 MB in 2×2^{30} Kondensatoren, die in acht Bänke aufgeteilt sind. Die Bänke sind ein Teil des Speicherkerns. Der Speicherkern ist mit $f_{kern} = 233.25$ MHz getaktet und arbeitet auf der steigenden Flanke des Takt-Signals. Der IO-Puffer ist eine Schnittstelle zwischen dem Speicherkern und dem IMC. Die 64 bit der Daten aus einer Bank werden in einem Takt ausgelesen und über die 8-Bit-Leitung entweder zum oder vom IMC übertragen. Der IO-Puffer arbeitet sowohl auf der fallenden, als auch auf der steigenden Flanke des 4-fachen Takt-Signals des Speicherkerns.

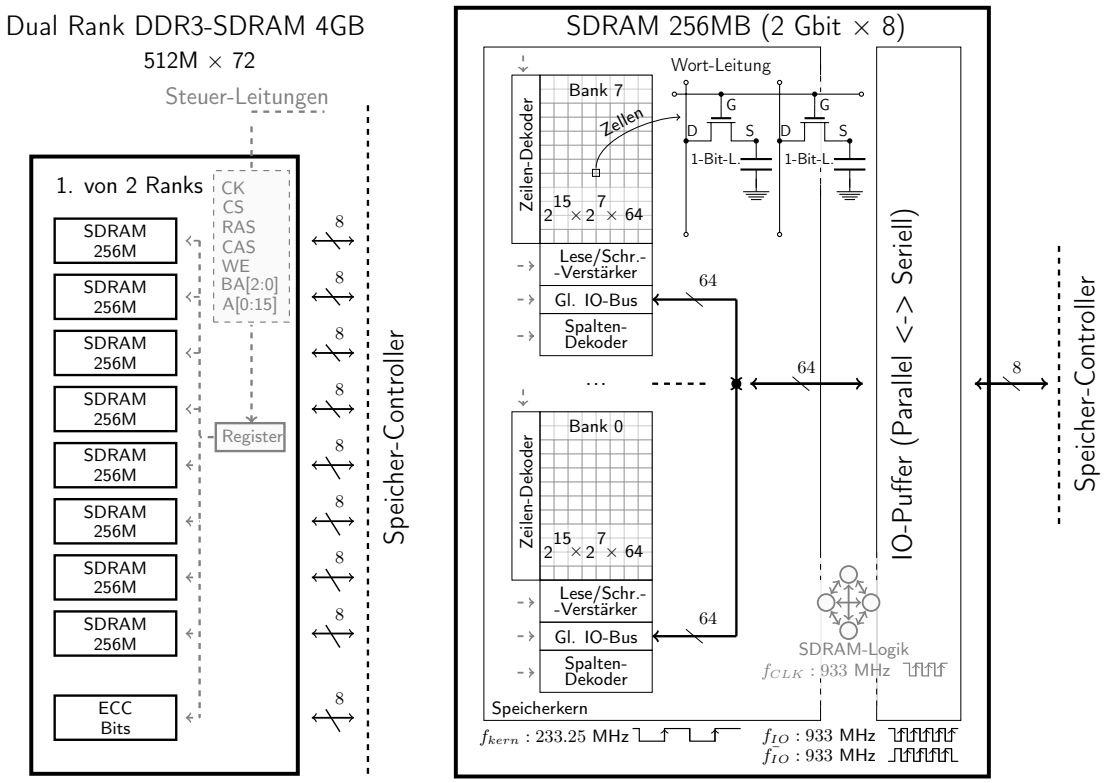

Abb. G.1.: In der Abbildung links ist einer von zwei Ranks eines DDR3-SDRAM-Speichermoduls dargestellt. Ein Rank ist mit acht SDRAM-Chips und einem ECC-Chip bestückt. Das rechte Schema zeichnet ein SDRAM-Chip. Der Chip speichert 256 MB in 2×2^{30} Kondensatoren, die in acht Bänken aufgeteilt sind. Die 64 bit der Daten aus einer Bank werden in einem Takt der Speicherkernfrequenz f_{kern} ausgelesen und über die 8-Bit-IO-Leitung entweder zum oder vom IMC sowohl auf der fallenden, als auch auf der steigenden Flanke des 4-fachen Takt-Signals des Speicherkerns übertragen.

II. SDRAM-Chip

Der betrachtete 2 Gbit × 8 DDR3-SDRAM-Chip speichert 2 Gbit der Daten und überträgt diese entweder zum IMC oder vom IMC durch eine 8-Bit-Leitung. Die Daten werden in acht Speicherzellen-Arrays flüssig[2] gehalten. Eine Zeile im Array, eine sogenannte „Page“, besteht aus 8×2^{10} Speicherzellen, die durch je einen Transistor als Schalter und einen Kondensator realisiert sind (siehe in Abb. G.1 rechts vom Array der Bank 7). Der Ladezustand eines Transistors entscheidet über den entsprechenden Binärwert der Daten. Zum Lesen oder Schreiben eines Bits muss der Transistor im leitenden Zustand sein. Der Zeilen-Decoder aktiviert eine entsprechende Wortleitung, um durch die Gates alle Transistoren einer Zeile zu öffnen. Jeder der 8×2^{10} Kondensatoren der selektierten Zeile wird damit mit einem Lese/Schreib-Verstärker über eine 1-Bit-Leitung gekoppelt. Der Ladungszustand der Kondensatoren kann kurz nach der Aktivierung gemessen oder geändert werden.
Man ist gezwungen, die Anzahl der Leitungen, derer Länge und Fläche im Chip einzugrenzen, um die elektrische Leistung und die Selbstinduktivität des Chips zu limitieren. Deshalb kann nur eine kleine Anzahl der Bits im Lese/Schreib-Verstärker mit den globalen IO-Leitungen zeitgleich verbunden werden.
In einem Takt der Speicherkern-Frequenz f_{kern} ist ausschließlich ein Lese/Schreib-Verstärker mit dem IO-Puffer gekoppelt. 64 nacheinander folgende Bits werden parallel entweder gelesen oder geschrieben. Diese parallele Sequenz der Daten wird im IO-Puffer serialisiert und über die 8-Bit-Leitung mit der 4-fachen Frequenz $f_{IO} = 4 \times f_{kern}$ auf den steigenden und fallenden Flanken entweder zum IMC oder vom IMC übertragen. Das Verfahren wird „n × Prefetching“ genannt.[3] Die DDR3-SDRAM- und DDR4-SDRAM-Chips benutzen „8 × Prefetching“ und greifen auf 8 bit zu, um 1 bit zwischen dem Chip und dem IMC zu übertragen.
Somit können 8 bit der Daten mit der Speicher-Frequenz $f_{MEM} = 1866$ MHz vom DDR3-SDRAM-Chip zum Ring-Bus des Prozessors über den IMC übertragen werden (siehe Abb. 5.6).

[2] Die Bedeutung des Begriffes „flüssig“ ist in Abschnitt VI erläutert.

[3] Ein DDR1-SDRAM-Chip benutzt „2 × Prefetching“ und greift in der Bank auf 2 bit der Daten zu, um 1 bit zu übertragen. Ein DDR2-SDRAM-Chip benutzt „4 × Prefetching“.

III. SDRAM-Steuerung

Ein SDRAM-Chip ist ein Automat mit mehreren Zuständen und Übergängen, deren Zeitspezifikation in der SDRAM-Logik programmiert ist. Somit muss das Steuerprotokoll kein Handshake-Verfahren zwischen dem Speicher und dem Speichercontroller implizieren. Dies unterscheidet ein DRAM von einem SDRAM.
Wie oben erwähnt, werden die SDRAM-Chips synchron zum Taktsignal der IO-Puffer-Frequenz gesteuert. In Abb. G.1 ist diese Frequenz zusätzlich als f_{CLK} dargestellt. Das Signal an der „Chip Select“-Steuer-Leitung aktiviert alle Chips eines Speicherranks (in Abbildung G.1 als „CS“ gekennzeichnet).
Ein Speicherzugriff wird in vier Phasen durchgeführt: der Zeilen-Auswahl, der Spalten-Auswahl, dem Datentransfer und der Zeilen-Schließung. Der Prozess wird mit einer Steuer-Sequenz gesteuert. Die wichtigsten Signale zur Steuerung eines Speicherzugriffs und deren Folgen sind:

- *CK:* Die „Clock“-Leitung dient zum Takten des Chips mit der Frequenz f_{CLK}.

- *BA[2:0]:* Mit dem „Bank Adress“-Bus wird eine von acht Bänken für die nachfolgende Datenübertragung selektiert.

- *RAS:* Mit dem „Row Address Strob“-Signal wird die Aktivierung einer Speicherzeile im Array gestartet. Der Zeilen-Decoder öffnet durch eine Wortleitung gleichzeitig alle Transistoren der zu aktivierenden Zeile und lässt die Spannung an den 1-Bit-Leitungen abhängig vom Ladezustand der Kondensatoren erhöhen oder herabsetzen. Der Lese/Schreibe-Verstärker ist danach bereit die Ladezustände zu lesen und zu schreiben. Der Steuer-Befehl zur Aktivierung einer Speicherzeile wird als „ACTIVE“ bezeichnet. Bevor man eine andere Zeile in der Bank auswählt, muss die aktive Zeile geschlossen sein, weil die Bank nur eine einzige geöffnete Zeile haben kann. Der entsprechende Befehl wird „PRECHARGED“ genannt.

- *CAS:* Mit dem „Column Address Strob“-Signal findet die Übertragung der 64 nacheinander folgenden Bits zwischen dem Lese/Schreibe-Verstärker und dem IMC statt. Zu diesem Zweck werden die entsprechenden globalen IO-Leitungen mit dem IO-Puffer verbunden. Diese Operation wird „READ“ oder „WRITE“ genannt.

- *WE:* Mit dem Signal „Write Enable“ wird angegeben, ob es sich während des „CAS“ -Signales um einen Lese- oder einen Schreibzugriff handelt.

- *A[14:0]:* Mit dem „Adress“-Bus wird während des „RAS“-Signals die Adresse der selektierten Zeile festgelegt. Damit kann Jede einzelne aus 2^{15} Zeilen identifiziert werden. Während des „CAS“-Signals werden die ersten 10 Leitungen *A[9:0]* für die Adresse der zu übertragenen Spalten benutzt. Die Granularität der Spaltenadressierung ist dabei 1 Byte. Die restlichen Leitungen sind für die weiteren Spezifikationen reserviert. Muss die aktive Zeile nach dem Lesen zugemacht werden, so kann das entsprechende Signal vom IMC durch die Leitung *A[12]* gesendet werden. Mit der Leitung *A[10]* kann der Parameter *BL* für den „Burst“-Modus[4] eingestellt werden.

Wie oben erwähnt, werden alle Operationen des Speichers synchron zum Taktsignal durchgeführt. Dabei muss die Latenzzeit der unterschiedlichen Steuer-Signale berücksichtigt werden. In der Tabelle Tab. G.1 sind beispielsweise ein paar Latenzzeiten für DDR3-SDRAM-Chips aufgelistet.[5]

Tab. G.1.: In der Tabelle sind einige Latenzzeiten für einen DDR3-SDRAM-Chip (1866 MHz) angegeben

Bezeichnung	**Latenzzeit [ns]/[cycles]**	**Beschreibung**
t_{CK}/CK	1.07/1	Dauer eines Taktes der IO-Puffer-Frequenz f_{IO};
t_{RCD}/RCD	13.91/13	Das Intervall zwischen „RAS“- und „CAS“-Signalen der gleichen Zeile;
t_{RAS}/RAS	13.91/13 — 13.91+ $9 \times t_{REFI}/66$	Das Intervall zwischen dem „RAS“- und dem „PRECHARGED“ -Befehl ist zulässig zwischen zwei nacheinander folgenden Speicherzellen -Auffrischungen der selektierten Zeile ;
t_{RP}/RP	13.91/13	Das Intervall zwischen einer „PRECHARGED“-Operation und dem „RAS“-Signal auf der identischen Bank;
t_{RRD}/RRD	5.35/5	Das Intervall zwischen zwei „RAS“-Signalen auf der identischen Bank;
t_{CAS}/CAS (CL)	13.91/13	Die Verzögerung zwischen einem „CAS“-Signal und dem ersten entsprechenden Bit der Daten auf den globalen IO-Leitungen;
t_{CCD}/CCD	4.28/4	Die minimale Zeit zwischen zwei „CAS“-Signalen;
t_{REFI}/REFI	7.8/4 3.9/4	Die durchschnittlichen Zeitintervalle zwischen nacheinander folgenden Auffrischungen der Bänken beim Temperaturintervall $0°C \leq T_{case} \leq 85°C \leq T_{case}$ und beim Temperaturintervall $85°C < T_{case} \leq 95°C \leq T_{case}$ (Siehe Abschnitt VI);

[4]Der „Burst“-Modus ist in Abschnitt VI beschrieben.

[5]Mit dem in der Tabelle benutzten Wert t_{REFI} bezeichnet man das durchschnittliche Zeitintervall zwischen zwei nacheinander folgenden Auffrischungen der Bänke. Der typische Wert ist 7.8 µs. Mehr Details über die Speicherzellen-Auffrischung siehe im Abschnitt VI.

IV. Burst-Modus

Als „Burst“-Modus bezeichnet man einen optimierten Zugriff auf die Speicherzellen einer Zeile. Ohne die Allgemeinheit der weiteren Aussagen zu limitieren, werden wir einen Lesezugriff betrachten. Der Schreibzugriff erfolgt symmetrisch dazu. Im „Burst“-Modus werden nach einem „RAS“- und einem „CAS“-Signal in jedem Takt der Frequenz f_{kern} achtmal nacheinander 64 bit der Daten zum IO-Puffer übertragen. Wenn der „Burst“-Modus ausgeschaltet ist, kosten die sonst notwendigen 7 „CAS“-Signale für die Übertragung von 64 B eine zusätzliche Zeit von bis zu $7 \times t_{RAS}$ (siehe Tabelle Tab. G.1). Allerdings kann dabei auf die beliebigen Speicherzellen der Zeile zugegriffen werden.[6]
Wenn die nacheinander folgenden Zugriffe die gleiche Speicherbank und nicht die gleiche Zeile referenzieren, muss die aktuell geladene Zeile mit dem „PRECHARGED“ geschlossen werden. Die zusätzliche Zeit dafür ist t_{RP} (siehe Tabellen Tab. G.1). Wenn dagegen die nacheinander folgenden Zugriffe im „Burst“-Modus auf den unterschiedlichen Bänken ausgeübt werden, können noch mehr Daten ohne Unterbrechung entweder ausgelesen oder geschrieben werden. In Abb. G.2 ist ein vereinfachtes Zeitdiagramm von zwei nacheinander folgenden Speicherzugriffen im „Burst“-Modus dargestellt. Die Latenzzeiten sind nicht im Maßstab gehalten. Die Signal-Leitung „RAS“ wird eingeschaltet, um den ersten „RAS“-Befehl im SDRAM-Chip zu bearbeiten. Neben dem „RAS“-Signal werden eine Bank- und eine Zeile-Adresse gesetzt („$b1$“ und „Row_n“ dem entsprechend). Die Bank- und die Zeile-Adresse werden über die Leitungen $BA[2:0]$ und $A[14:0]$ übertragen. Nach der Latenzzeit t_{RCD} kann die erste Spaltenadresse ausgewählt werden. Dies wird mit dem „CAS“-Signal und der entsprechenden Belegung des Adressessbusses $A[0:9]$ durchgeführt. Über die Leitung $A[10]$ wird zusätzlich ein Signal darüber geschickt, wie die Zeile mit „PRECHARGED“ geschlossen werden muss. Im einfachsten Fall benutzt man ein Signal zum automatischen Schließen der Zeile nach dem Zugriff (nicht in der Abbildung gezeigt). Der Chip braucht die Latenzzeit t_{CL}, um die Daten vom Lese/Schreibe-Verstärker zum IMC über IO-Puffer zu übertragen. Nach t_{CL} stehen die ersten 64 bit der Daten dem IMC zur Verfügung. Mit dem „Burst“-Modus werden 8×64 bit der Daten aus der Zeile „Row_n“ der Bank „$b1$“ und ab der Spalte „Col_n“ gelesen. Damit die Daten auch danach ohne Verzögerung an den IMC geliefert werden könnten, wird gleich nach dem ersten „RAS“-Signal, jedoch frühesten nach t_{RRD}, ein neues „RAS“-Signal für eine andere Bank gesendet. Dementsprechend sendet der IMC nach t_{RCD} ein

[6] Die Adressierung der Spalten ist dadurch limitiert, dass mit einem „CAS“-Signal 64 nacheinander folgende Bits gelesen werden und dass die Spalten-Adressierung eine 1 Byte-Granularität hat.

nachfolgendes „CAS“-Signal. Wenn die Latenzzeiten nicht beibehalten werden, kann eine unzulässige Situation auftreten, wo unterschiedliche Bänke gleichzeitig auf die globalen IO-Leitungen zugreifen könnten.
Der dargestellte „Burst“-Modus ist mit der „Burst“-Länge $BL = 8$ konfiguriert. Die Speichermodule können auch im „Burst“-Modus mit der Länge $BL = 4$ arbeiten. In diesem Fall werden 4×64 bit der Daten entweder ausgelesen oder geschrieben. Dabei ist allerdings zu beachten, dass wegen der Latenzzeiten die Lücken in der Datenausgabe auftreten und dadurch die Bandbreite verringert wird.

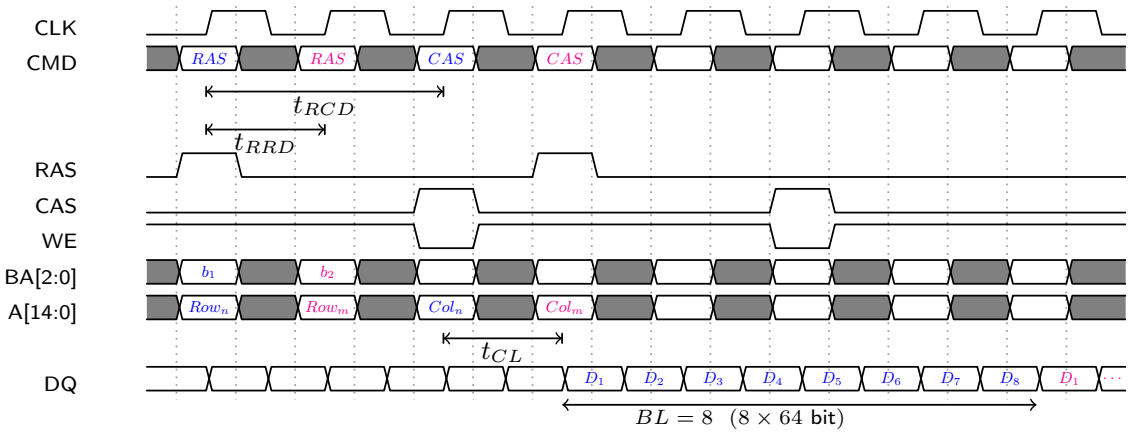

Abb. G.2.: In der Abbildung ist ein vereinfachtes Zeitdiagramm von zwei nacheinander folgenden Speicherzugriffen im „Burst“-Modus dargestellt.

V. Erweiterung für DDR4-SDRAM

Der JEDEC Standard für DDR4-SDRAM [44] erweitert den Speichermodulaufbau von DDR3-SDRAM mit einer zusätzlichen Hierarchiestufe. Die Bänke werden in verschiedenen Bankgruppen erfasst. Dabei teilen diese die globalen IO-Leitungen innerhalb jeder Gruppe. Neben der Bank-Adresse muss der IMC die Bankgruppe-ID an die SDRAM-Steuerungseinheit senden. Durch die zusätzliche Adressierung können sich die Latenzzeiten in Takten erhöhen. DDR4-SDRAM wird mit einer niedrigen Spannung betrieben und kann unter anderem dadurch mit einer höheren Dichte der Speicherzellen bestückt und mit einer höheren Speicherkern-Frequenz als DDR3-SDRAM getaktet werden.

VI. Speicherzellen-Auffrischung

Die Speicherzellen müssen regelmäßig aufgefrischt werden. Dies kann ein weiterer limitierender Faktor für die effektive Bandbreite sein. Eine Speicherzelle besteht aus einem Transistor und einem Kondensator. Die Speicherung eines bit geschieht durch die Ladung des Kondensators. Wenn eine Speicherzelle einem Zugriff nicht ausgesetzt wird, bleibt der Transistor in einem gesperrten, nicht leitenden Zustand. Allerdings bleiben unerwünschte Stromflüsse, die sogenannten Leckströme, innerhalb des Transistors. Die Leckströme können die Ladung des Kondensators mit der Zeit verändern. Dazu kommt auch die Selbstentladung des Kondensators. Um den Informationsverlust zu verhindern, wird die Ladung innerhalb eines bestimmten Zeitintervalls wiederhergestellt. Da der Silikonfertigungsprozess einige zufällige Variationen in der Qualität der Speicherzellen verursacht, wird ein genügend kleines Intervall gewählt, um die Zustandsstabilität für alle Zellen im Chip zu gewährleisten. Das Intervall wird als „*retention time*“ t_{RT} bezeichnet. Im Normalbetrieb, wenn die Temperatur zwischen 0 °C und 85 °C ist, müssen alle Kondensatoren eines Moduls mindestens einmal innerhalb von $t_{RT} = 64$ ms aufgefrischt werden. Wenn die Temperatur höher als 85 °C ist, verringert sich das Intervall auf die Hälfte. Typischerweise werden die Chips im „*Auto-Refresh (AR)*“ Mode betrieben. Der Speichercontroller muss regelmäßig den „AR“-Befehl an alle Bänke eines Speichermoduls schicken. Weiteres wird vom Chip selbst organisiert, so dass die Array-Zeilen nacheinander in den Leseverstärker eingelesen und danach zurückgeschrieben werden. Dies ist äquivalent zur Ausführung von zwei nacheinander folgenden „RAS“- und „PRECHARGED“-Befehlen. Während der Auffrischung darf die Speicherbank nicht gelesen oder geschrieben werden. Durchschnittlich wird der „AR“-Befehl alle 7.8 µs im Chip ausgeführt, um die Speicherzellen aller acht Bänken aufzufrischen. Diese Zeit wird in der Literatur t_{REFI} genannt. Tabelle Tab. G.2 zeigt, wie lange die Auffrischung aller Bänken bei der Temperatur zwischen 0 °C und 85 °C abhängig von der Dichte der Bänke dauert. Ob die Auffrischung und die Speicherzugriffe überlappend ausgeführt werden, hängt davon ab, wie die Speicherzugriffe organisiert sind.

Tab. G.2.: Auffrischungszeit t_{RFC} für die verschiedenen Speicherkonfigurationen. Der Prozess muss mindestens einmal innerhalb von $t_{RT} = 64$ ms durchgeführt werden.

SDRAM Bank	1 Gb	2 Gb	4 Gb	8 Gb	16 Gb	32 Gb
DDR3	110 ns	160 ns	300 ns	350 ns	-	-
DDR4	-	160 ns	260 ns	350 ns	480 ns	640 ns

H. Modellierte Bandbreite im Falle des Haltens der Daten im Cache

In einem Takt der CPU-Frequenz f_{CPU} kann der Prozessor *Ivy Bridge* aus dem L1-Cache in die AVX-Register 2×16 B laden und 1×16 B schreiben. Die Interfacebreite des *Haswell* Prozessors verdoppelte sich gegenüber der *Ivy Bridge*. Somit ergibt sich, dass die erreichte Bandbreite kleiner als die theoretische (siehe Gleichung Gl. (5.4)) um 15% für den *Ivy Bridge* und um 37% für den *Haswell* ist.
Die Durchsatzanalyse der unterschiedlichen Stream-Operationen mit ECM-Modell nach [32] zeigt, dass der Datentransfer zwischen den verschiedenen Hierarchiestufen des Speichersystems höchstwahrscheinlich nicht überlappend erfolgt. Aus dieser Tatsache ergibt sich, dass die Bandbreite der Kernel-Operation im L2-Cache aus der Messung im L1-Cache und der theoretischen Bandbreite des L2-Caches abgeleitet werden kann. Bei dem ECM-Modell wird vorausgesetzt, dass die Rechenleistung bzw. Bandbreite mit der CPU-Frequenz und der Prozessorkernanzahl linear skaliert. Wie im vorherigen Kapitel gezeigt wurde, stimmt diese Annahme im Falle des L1- und L2-Caches für die beiden Prozessoren. Im Falle des L3-Caches des *Haswell* Prozessors degradiert die Rechenleistung bei den höheren CPU-Frequenzen. Um das Modell aufzubauen, werden die folgenden Bezeichnungen eingeführt:

$$
\begin{array}{ll}
T_{StufeA \leftrightarrow StufeB} & : \text{Die Zeit in cycles zur Übertragung aller Daten zwischen} \\
 & \quad \text{zwei Hierarchiestufen des Speichers } \textit{StufeA} \text{ und } \textit{StufeB}. \\
 & \quad \text{Die cycles werden in der Frequenz der langsamsten Stufe berechnet;} \\
Q_{StufeA \leftrightarrow StufeB} & : \text{Die Datenmenge in GB zum Übertragen zwischen} \\
 & \quad \text{zwei Hierarchiestufen } \textit{StufeA} \text{ und } \textit{StufeB}; \\
B_{StufeA \leftrightarrow StufeB} & : \text{Die Bandbreite in } {}^{\text{B}}/_{\text{cycle}} \text{ zwischen } \textit{StufeA} \text{ und } \textit{StufeB} \\
B^{TH} & : \text{Die theoretische Bandbreite in } {}^{\text{B}}/_{\text{cycle}}; \\
B^{DTM} & : \text{Die modellierte Bandbreite in } {}^{\text{B}}/_{\text{cycle}};
\end{array}
\tag{H.1}
$$

Unter Berücksichtigung, dass der Datenmengentransfer zwischen den *L2*- und *L1*-Caches höher als zwischen dem *L1*-Cache und den AVX-Registern ist (siehe Ab-

schnitt 5.1.4), ergibt sich folgendes Verhältnis:

$$\begin{array}{ll} Q_{L2\leftrightarrow L1} & = \frac{N_{read}+2\times N_{write}}{N_{read}+N_{write}} \times Q_{L1\leftrightarrow AVX}; \\ N_{read}, N_{write} & \textit{- Anzahl der Variablen zum Lesen/Schreiben für ein} \text{ FLOP}; \end{array} \tag{H.2}$$

Mit 5.3 und H.2 kann Gleichung Gl. (H.3) für die Bandbreite und die Ausführungszeit aufgestellt werden:

$$\begin{array}{ll} T_{L2\leftrightarrow AVX} & = T_{L2\leftrightarrow L1} + T_{L1\leftrightarrow AVX} \overset{ECM}{\Leftrightarrow} \textit{ Der Datentransport ist nicht überlappend;} \\ T_{L2\leftrightarrow AVX} & \overset{5.3}{=} \frac{Q_{L2\leftrightarrow L1}}{B_{L2\leftrightarrow AVX}} \overset{H.2}{=} \frac{4}{3} \times \frac{Q_{L1\leftrightarrow AVX}}{B_{L2\leftrightarrow AVX}}; \\ T_{L2\leftrightarrow L1} & \overset{5.3}{=} \frac{Q_{L2\leftrightarrow L1}}{B_{L2\leftrightarrow L1}} \overset{H.2}{=} \frac{4}{3} \times \frac{Q_{L1\leftrightarrow AVX}}{B_{L2\leftrightarrow L1}}; \\ T_{L1\leftrightarrow AVX} & \overset{5.3}{=} \frac{Q_{L1\leftrightarrow AVX}}{B_{L1\leftrightarrow AVX}}; \end{array} \tag{H.3}$$

I. Effizienz des Datentransfers im Falle des Haltens der Daten im Cache

Zusätzlich wird die Effizienz $EF_{DTML2\leftrightarrow AVX}$ des Datentransfers berechnet, die zeigt, um wie viel Prozent sich die gemessene Bandbreite von der modellierten Bandbreite unterscheidet. Für die Berechnung des relativen Fehlers der Effizienz werden nach [78] die entsprechenden relativen Fehler summiert. Somit ergibt sich folgende Gleichung für die modellierte Bandbreite $B_{DTML2\leftrightarrow AVX}$ und für deren Unterschied zur gemessenen Bandbreite $B_{MESSL2\leftrightarrow AVX}$:

$$\begin{array}{ll} \frac{1}{B^{DTM}_{L2\leftrightarrow AVX}} & = \frac{1}{B^{TH}_{L2\leftrightarrow L1}} + \frac{3}{4} \times \frac{1}{B^{MESS}_{L1\leftrightarrow AVX}}; \\ EF^{DTM}_{L2\leftrightarrow AVX} & = \frac{B^{MESS}_{L2\leftrightarrow AVX}}{B^{DTM}_{L2\leftrightarrow AVX}} \times 100\%; \\ \epsilon_{rel}{}^{DTM}_{EF_{L2\leftrightarrow AVX}} & = \epsilon_{rel}{}^{MESS}_{B_{L1\leftrightarrow L1}} + \epsilon_{rel}{}^{MESS}_{B_{L2\leftrightarrow AVX}}; \end{array} \tag{H.4}$$

Es ist anzunehmen, dass sich die durchschnittliche Bandbreite im Falle des Haltens der Daten im L2-Cache zwischen dem L1-Cache und den AVX-Registern nicht ändert, somit kann die Bandbreite $B_{DTML2\leftrightarrow L1}$ zwischen den L2- und L1-Caches mit der

folgenden Gleichung berechnet werden (abgeleitet von Gl. (H.4)):

$$\begin{aligned} \frac{1}{B^{DTM}_{L2\leftrightarrow L1}} &= \frac{1}{B^{MESS}_{L2\leftrightarrow AVX}} - \frac{3}{4} \times \frac{1}{B^{MESS}_{L1\leftrightarrow AVX}}; \\ \epsilon_{rel}{}^{DTM}_{B_{L2\leftrightarrow AVX}} &= \max(\epsilon_{rel}{}^{MESS}_{B_{L1\leftrightarrow AVX}}, \epsilon_{rel}{}^{MESS}_{B_{L1\leftrightarrow AVX}}); \end{aligned} \tag{H.5}$$

II. DTM-Modell im Falle des Haltens der Daten im L2-Cache

Die theoretische Bandbreite $B_{THL2\leftrightarrow L1}$ der beiden Prozessoren *Ivy Bridge* und *Haswell* ist in [18] dementsprechend mit 32 $^{\text{B}}/_{\text{cycle}}$ und 64 $^{\text{B}}/_{\text{cycle}}$ angegeben. Für die gemessene Bandbreite $B_{MESSL1\leftrightarrow AVX}$ werden die approximierten Werte aus Gl. (5.4) benutzt. Nach dem Einsetzen der Werte in Gleichung Gl. (H.4) ergibt sich Gleichung Gl. (H.6) für die Kernel-Operation *Add* im Falle des Haltens der Daten im L2-Cache. Zum Vergleich wird ebenfalls die gemessene Bandbreite aus Gl. (5.4) angegeben:

$$\begin{aligned} &\textbf{Ivy Bridge (E5-2690v2)} && : \\ &B^{DTM}_{L2\leftrightarrow AVX} && = (\tfrac{1}{32} + \tfrac{3}{4} \times \tfrac{1}{40.7897})^{-1} = 20.1463\ ^{\text{B}}/_{\text{cycle}}\ (\pm 1.6071\%); \\ &B^{MESS}_{L2\leftrightarrow AVX} && = 19.73696\ ^{\text{B}}/_{\text{cycle}}\ (\pm 2.8178\%); \\ &EF^{DTM}_{L2\leftrightarrow AVX} && = 97.97 \pm 4.42\% \wedge EF_{DTM_{L2}} \leq 100\%; \\ &B^{DTM}_{L2\leftrightarrow L1} && = 30.98\ ^{\text{B}}/_{\text{cycle}}\ (\pm 2.8178\%); \\ &\textbf{Haswell (E5-2680v3)} && : \\ &B^{DTM}_{L2\leftrightarrow AVX} && = (\tfrac{1}{64} + \tfrac{3}{4} \times \tfrac{1}{61.366})^{-1} = 35.9108\ ^{\text{B}}/_{\text{cycle}}\ (\pm 1.8783\%); \\ &B^{MESS}_{L2\leftrightarrow AVX} && = 25.61734\ ^{\text{B}}/_{\text{cycle}}\ (\pm 1.8633\%); \\ &EF^{DTM}_{L2\leftrightarrow AVX} && = 71.34(\pm 3.74)\%; \\ &B^{DTM}_{L2\leftrightarrow L1} && = 37.29\ ^{\text{B}}/_{\text{cycle}}\ (\pm 1.81783\%); \end{aligned} \tag{H.6}$$

Somit ergibt sich, dass die tatsächliche Bandbreite $B_{DTML2\leftrightarrow L1}$ im *Ivy Bridge* zwischen L2- und L1-Caches sehr nah an der theoretischen Bandbreite von 32 $^{\text{B}}/_{\text{cycle}}$ ist. Das heißt, dass der *Ivy Bridge* über 96% der vollen Bandbreite zwischen L2- und L1-Caches nutzt. Im Gegensatz dazu zeigt der *Haswell* weniger als $\sim$ 58%

der theoretischen Bandbreite und dadurch sinkt die Effizienz des Rechnens auf $\sim$ 71%. Hierbei ist zu beachten, dass die Effizienz des Datentransports zwischen L1 und AVX-Registern als 100% angenommen ist, weil die Messresultate B_{L1}^{MESS} in Gl. (H.6) eingesetzt sind.

III. DTM-Modell im Falle des Haltens der Daten im L3-Cache

Der Fall des Haltens der Daten im L3-Cache ist wegen der vielen Unbekannten in der Funktionsweise des Ring-Busses komplexer geworden. Ich vereinfache das Modell mit einer im vorherigen Abschnitt begründeten Annahme, dass die Daten in den L3-Segmenten größtenteils lokal zu den entsprechenden Prozessorkernen gehalten werden (siehe Abschnitt 5.2.2.1). Trotzdem kann der Ring ein Flaschenhals sein. Dies wird durch die durchgeführten Messungen bestätigt (siehe Gl. (5.4)). Die Bandbreite des L3-Caches beträgt nach [18] für die beiden Prozessoren 32 $^{\text{B}}/_{\text{cycle}}$. Im Unterschied zum L2-Cache wird der Vorfaktor $\frac{3}{4}$ nicht mehr benötigt, weil dieser in $B_{mess_{L2}}$ schon berücksichtigt ist.

Ivy Bridge (E5-2690v2) :

$$
\begin{aligned}
B_{L3\leftrightarrow AVX}^{DTM} &= (\tfrac{1}{32} + \tfrac{1}{19.73696})^{-1} = 12.207573 \ ^{\text{B}}/_{\text{cycle}} \ (\pm 2.8\%); \\
B_{L3\leftrightarrow AVX}^{MESS} &= 11.5520 \ ^{\text{B}}/_{\text{cycle}} \ (\pm 1.2\%); \\
EF_{L3\leftrightarrow AVX}^{DTM} &= 94.63 \pm 4.0\%; \\
B_{L3\leftrightarrow L2}^{DTM} &= 27.856 \ ^{\text{B}}/_{\text{cycle}} \ (\pm 2.8\%);
\end{aligned}
$$

Haswell (E5-2680v3) :

$$
\begin{aligned}
B_{L3\leftrightarrow AVX}^{DTM} &= (\tfrac{1}{32} + \tfrac{1}{25.61734})^{-1} = 14.227572 \ ^{\text{B}}/_{\text{cycle}} \ (\pm 1.9\%); \\
B_{L3\leftrightarrow AVX,1\leq p\leq 2}^{MESS} &= 13.1616 \ ^{\text{B}}/_{\text{cycle}} \ (\pm 3.0\%); \\
EF_{L3\leftrightarrow AVX,1\leq p\leq 2}^{DTM} &= 92.77(\pm 4.9)\%; \\
B_{L3\leftrightarrow L2,1\leq p\leq 2}^{DTM} &= 27.0691 \ ^{\text{B}}/_{\text{cycle}} \ (\pm 1.9\%); \\
B_{L3\leftrightarrow AVX,11\leq p\leq 12}^{MESS} &= 14.96 \ ^{\text{B}}/_{\text{cycle}} \ (\pm 3.0\%); \\
EF_{L3\leftrightarrow AVX,11\leq p\leq 12}^{DTM} &= 105.1(\pm 4.9)\% \wedge EF_{L3,P_2}^{DTM} \leq 100\%; \\
B_{L3\leftrightarrow L2,11\leq p\leq 12}^{DTM} &= 33.9483 \ ^{\text{B}}/_{\text{cycle}} \ (\pm 1.9\%);
\end{aligned}
\tag{H.7}
$$

Die Ergebnisse für die beiden Prozessoren zeigen, dass der L3-Cache eine Effizienz

über $\sim 80\%$ hat. Der *Ivy Bridge* tauscht die Daten zwischen L2- und L3-Caches mit 27.856 $^{\mathrm{B}}/_{\mathrm{cycle}}$. Gleichung Gl. (5.4) für die Approximation der Bandbreite bestätigt, dass die Bandbreite für wenige Threads mit der CPU-Frequenz linear skaliert. Erst mit steigender Anzahl der Prozessorkerne wird die lineare Skalierbarkeit verlangsamt. Die Bandbreite B_{DTM} zwischen den unterschiedlichen Hierarchiestufen des Caches ist im *Ivy Bridge* fast identisch. Dies ist anders im *Haswell*. Besonders groß ist der Unterschied während der Ausführung mit einem oder zwei Prozessorkernen. Der Datentransfer zwischen den L2- und L3-Caches findet mit der Frequenz f_{RING} statt und die Berechnungen werden auf der Basis von f_{CPU} durchgeführt. Aus diesem Grund übersteigt die berechnete Bandbreite von 33.9483 $^{\mathrm{B}}/_{\mathrm{cycle}}$ die theoretische Bandbreite von 32 $^{\mathrm{B}}/_{\mathrm{cycle}}$: bei den niedrigeren CPU-Frequenzen ist f_{RING} höher als f_{CPU} und bei der höheren CPU-Frequenzen ist der Exponent für f in der Approximation der Bandbreite kleiner als Eins.

I. Hauptspeicher-Frequenz

I. Einfluss der Hauptspeicher-Frequenz auf die Bandbreite

Im vorhergehenden Abschnitt 5.2.3 habe ich im Modell die Beschränkung eingeführt, dass keine Daten mit einem „Prefetcher“ aus dem Speicher geholt werden, die eventuell bei den voranstehenden Befehlen während der Ausführung eines Kernels benötigt werden. Eine Reihe der Messungen, die mit den unterschiedlichen Einstellungen der Speicherfrequenz f_{MEM} im BIOS durchgeführt wurden, zeigen, dass diese Vereinfachung die Resultate der Modellierung der Kernelausführung nicht beeinflusst.
In Abb. I.1 oben ist die Bandbreite der Kernel-Operation *Add* auf dem Prozessor *Ivy Bridge* für die unterschiedlichen Speicherfrequenzen und unterschiedlichen CPU-Frequenzen für einen Thread, 3 und 10 Threads dargestellt. Die Speicherfrequenz wurde im Bereich von 888 MHz bis 1867 MHz geändert. In Abb. I.1 unten ist die Bandbreite für die Ausführung des Kernels auf dem Prozessor *Haswell* mit einem Thread, 3 und 12 Threads für die Speicherfrequenzen im Bereich von 1333 MHz bis 2133 MHz dargestellt.
Die Ausführung mit einem Thread, 2 (nicht dargestellt) und 3 Threads hängen von der Speicher-Frequenz f_{MEM} nicht ab oder ihre Abhängigkeit ist sehr gering. Der Grund dafür ist, dass die Menge der vom CA und weiterhin vom IMC angefragten Daten schon mit einer niedrigeren Frequenz f_{MEM} über die Speicherkanäle zum Ringbus geliefert werden kann. Wenn „Prefetcher“ aktiv wäre, könnten mehr Daten im Voraus aus dem Speicher geholt werden. Und die resultierende Bandbreite der Kernel-Operation hätte dann einen höheren Wert. Die Bandbreite mit 10 (*Ivy Bridge*) und 12 (*Haswell*) Threads steigt dagegen mit der Frequenz f_{MEM}, weil die Menge der benötigten Daten viel höher ist, als vier Speichermodule liefern könnten.

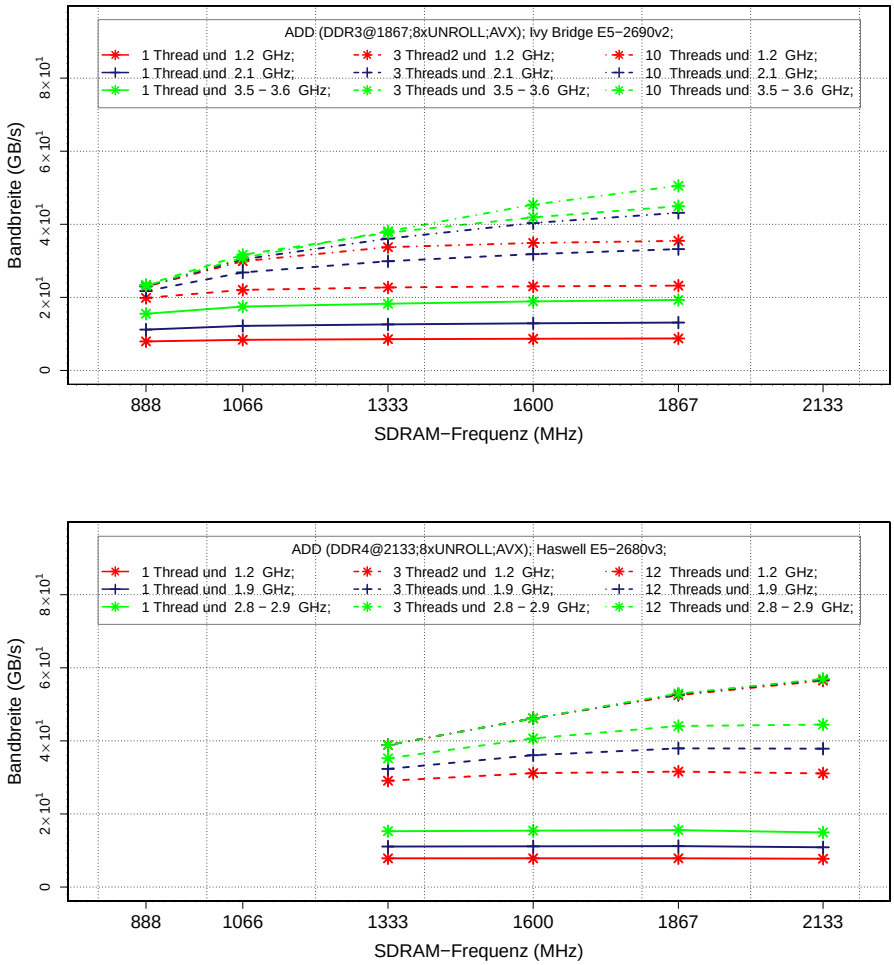

Abb. I.1.: Bandbreite der Kernel-Operation *Add* auf den Prozessoren *Ivy Bridge E5-2690v2* (oben) und *Haswell E5-2680v3* (unten) im Fall des Haltens der Daten im Hauptspeicher. Die SDRAM-Frequenz wurde im BIOS eingestellt.

II. Einfluss der Hauptspeicher-Frequenz auf die elektrische Leistung

Die Erhöhung der Hauptspeicherfrequenz im BIOS hat in vielen Fällen keinen oder einen geringen Einfluss auf die elektrische Leistung des Prozessors und des Speichers, wenn die Bandbreite des Hauptspeichers ausreichend ist. Dies ist der Fall, wenn die Anzahl der Threads und die CPU-Frequenz niedrig sind. Dies ist in Abb. I.2 und in Abb. I.3 deutlich zu erkennen.

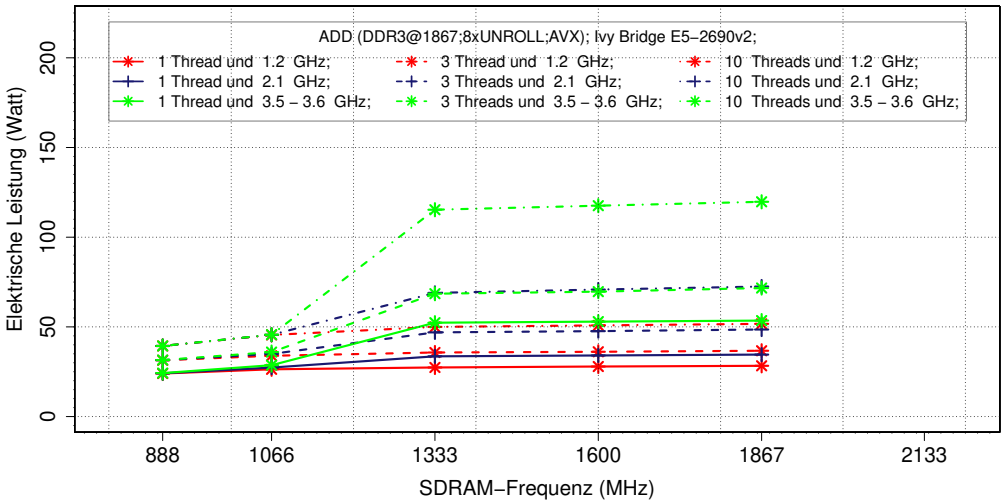

Abb. I.2.: Elektrische Leistung der Kernel-Operation *Add* auf den Prozessoren *Ivy Bridge E5-2690v2* im Fall des Haltens der Daten im Hauptspeicher. Die SDRAM-Frequenz wurde im BIOS eingestellt.

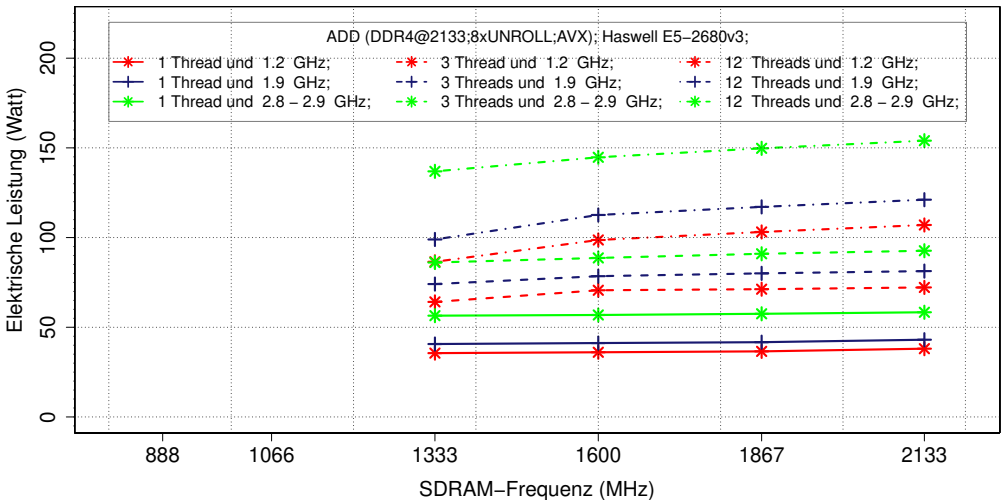

Abb. I.3.: Elektrische Leistung der Kernel-Operation *Add* auf den Prozessoren *Haswell E5-2680v3* im Fall des Haltens der Daten im Hauptspeicher. Die SDRAM-Frequenz wurde im BIOS eingestellt.

Eine saubere Trennung der Messung von der elektrischen Leistung der Speichermodule und der Prozessoren könnte uns eine detaillierte Auswertung der Änderungen in der elektrischen Leistung der Speichermodule liefern. Weiterführende Untersuchungen zu diesem Thema sind im Rahmen dieser Arbeit aus Zeitgründen nicht erfolgt, es erscheint jedoch sinnvoll, das Messsystem mit dieser Fähigkeit zu erweitern.

J. DTM-Modell

Das von mir entwickelte „Data-Transfer-Memory Modell“ (DTM) erklärt das Verhalten der Bandbreite einer Kernel-Operation, wenn die CPU-Frequenz und die Anzahl der Threads geändert werden. Das Modell enthält einen freien Parameter, nämlich den Überlappunkgskoeffizient K. Der Koeffizient K wird im Bereich zwischen 0 und 1 gesucht, so dass die Abweichung zwischen modellierter und gemessener Bandbreite minimal wird. Das DTM-Modell wurde im Abschnitt 5.3 dieser Arbeit erläutert.
Die Diagramme in Abb. J.2 zeigen den Überlappungskoeffizienten K, bei dem der relative Fehler der Bandbreitenapproximation minimal ist. Zum besseren Überblick werden die Werte nicht für alle möglichen CPU-Frequenzen angezeigt. Wenn der Koeffizient gleich eins ist, zeugt dies davon, dass die Berechnungen aus dem L2-Cache zu 100% Prozent überlappend durchgeführt werden, während die Daten aus dem Speicher zu den L3-Segmenten transportiert werden.

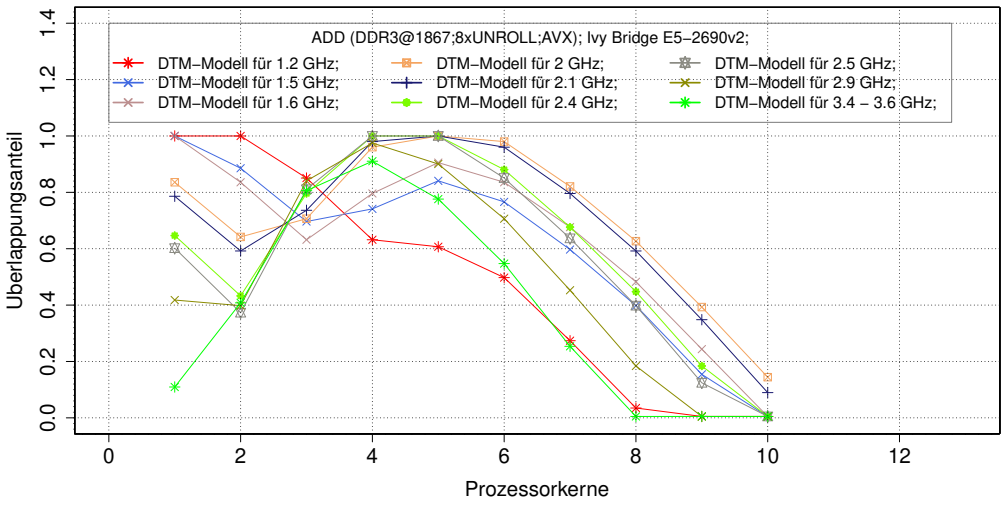

Abb. J.1.: Überlappungskoeffizient des DTM-Modells für die Ausführung der Kernel-Operation *Add* auf dem Prozessor *Ivy Bridge E5-2690v2* im Falle des Haltens der Daten im Hauptspeicher

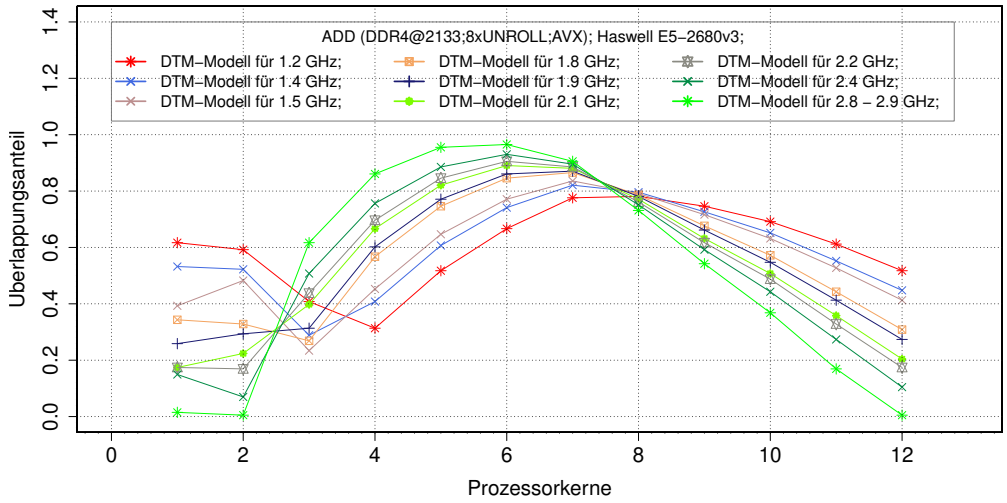

Abb. J.2.: Überlappungskoeffizient des DTM-Modells für die Ausführung der Kernel-Operation *Add* auf dem Prozessor *Haswell E5-2680 v3* im Falle des Haltens der Daten im Hauptspeicher

Es ist bemerkenswert, dass die Energiekosten der Kernel-Operation *Add* auf dem Prozessor *Ivy Bridge* bei der Ausführung mit 5 Threads und der CPU-Frequenz von 1.6 GHz minimal sind und der Überlappungskoeffizient für 1.6 GHz den maximalen Wert genau mit 5 Threads annimmt. Der Prozessor *Haswell* hat die minimalen Energiekosten bei der Ausführung mit 7 Threads und der CPU-Frequenz von 1.2 GHz. Zwar hat K den maximalen Wert entlang der Kurve für 1.2 GHz mit 8 Threads, die Werte des Überlappungskoeffizientes für 7 und 8 Threads unterscheiden sich minimal voneinander.

I. DTM-Modell für die Kernel Add3 und Copy

Dieser Anhang beinhaltet die Resultate des DTM-Modells für zwei weitere Kernel-Operationen *Copy* und *Add3* im Falle des Haltens der Daten im Hauptspeicher, wenn diese auf dem Prozessor *HaswellE5-2680v3* ausgeführt werden.
Der Kernel *Copy* `a[i]=b[i]` besteht aus zwei Mikrobefehlen. Der erste kopiert die Daten aus dem Speicher in ein AVX-Register und der zweite schreibt den Inhalt des Registers in den Speicher unter einer anderen Adresse zurück. Eine Implementierung des Kernels ist im Quelltext J.1 gezeigt.

Quelltext J.1: Eine Implementierung der Kernel-Operation *Copy* in Assemblersprache mit einer indirekten Adressierung sowohl für das Laden als auch für die Speicherung eines AVX-Registers.

```
...
vmovapd   (%r12,%rax,8), %ymm0
vmovapd   %ymm0, (%rb13,%rax,8)
...
```

Der Kernel *Add3* $a[i] = b[i] + c[i] + d[i]$ ist eine Erweiterung des Kernels *Add* mit einem zusätzlichen Array d. Eine Implementierung des Kernels ist im Quelltext J.2 gezeigt.

Quelltext J.2: Eine Implementierung der Kernel-Operation *Add3* in Assemblersprache mit einer indirekten Adressierung für das Laden und mit einer direkten Adressierung für die Speicherung eines AVX-Registers.

```
...
vmovapd   (%r15,%rax,8), %ymm0
vaddpd    (%r14,%rax,8), %ymm0, %ymm1
vaddpd    (%r12,%rax,8), %ymm1, %ymm2
lea       (%r13,%rax,8), %rsi
vmovupd   %ymm2, (%rsi)
...
```

Das Diagramm in Abb. J.4 zeigt sowohl das ECM-Modell und das DTM-Modell als auch die Messresultate für die Kernel-Operationen *Copy* auf dem Prozessor *Haswell E5-2680v3*.

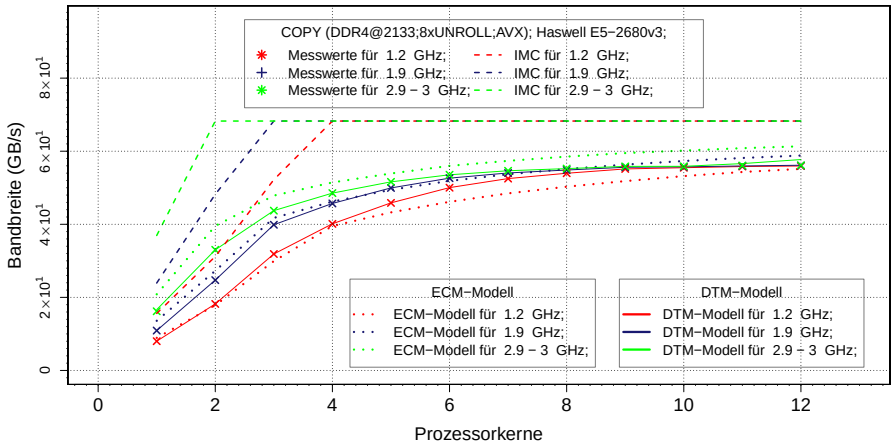

Abb. J.3.: Im Diagramm ist die mit ECM- und DTM-Modell berechnete Bandbreite der Kernel-Operation *Copy* auf dem Prozessor *Haswell E5-2680v3* im Falle des Haltens der Daten im Hauptspeicher dargestellt. Zum Vergleich sind zusätzlich die Messwerte mit Kreuzchen angegeben. Die gestrichelten Linien zeigen den Verlauf für die Bandbreite zwischen dem IMC und den Speichermodulen.

Das Diagramm in Abb. J.4 zeigt den Verlauf des Überlappungskoffezienten K für die Kernel-Operationen *Copy* auf dem Prozessor *Haswell E5-2680v3*.

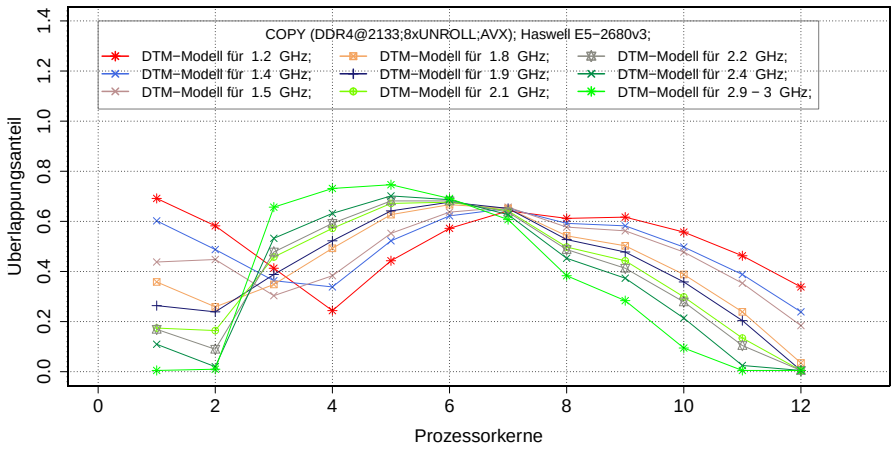

Abb. J.4.: Überlappungskoeffizient des DTM-Modells für die Ausführung der Kernel-Operation *Copy* auf dem Prozessor *Haswell E5-2680v3* im Falle des Haltens der Daten im Hauptspeicher

Die Diagramme in Abb. I und in Abb. J.6 zeigen die beiden Modelle und den Überlappungskoffezient K für die Kernel-Operation *Add3* auf dem Prozessor *Haswell*

E5-2680v3.

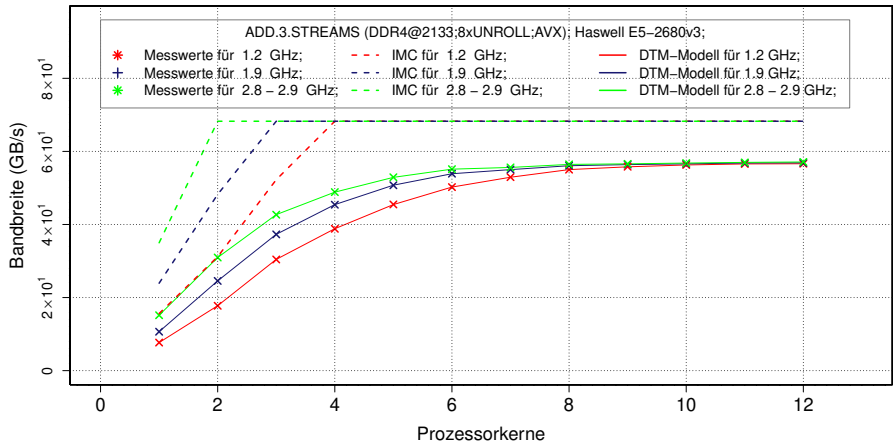

Abb. J.5.: Mit ECM- und DTM-Modell berechnete Bandbreite der Kernel-Operation Kernels *Add3* auf dem Prozessor *Haswell E5-2680v3* im Falle des Haltens der Daten im Hauptspeicher. Zum Vergleich sind zusätzlich die Messwerte mit Kreuzchen angegeben. Die gestrichenen Linien zeigen den Verlauf für die Bandbreite zwischen dem IMC und den Speichermodulen.

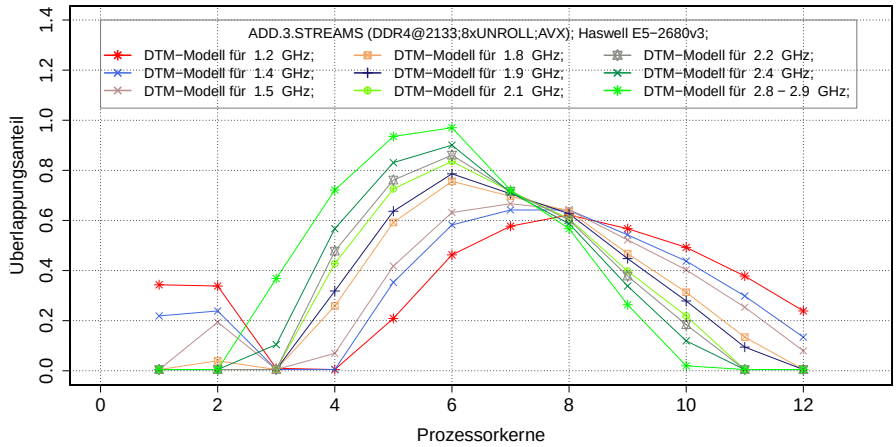

Abb. J.6.: Im Diagramm ist der Überlappungskoeffizient des DTM-Modells für die Ausführung der Kernel-Operation *Add3* auf dem Prozessor *Haswell E5-2680v3* im Falle des Haltens der Daten im Hauptspeicher dargestellt.

Es ist zu erkennen, dass die minimalen und maximalen Energiekosten mit dem Verlauf des Koeffizienten K korrelieren. Die minimalen Kosten wurden bei 7 Threads und der minimalen CPU-Frequenz von 1.2 GHz erreicht. Die maximalen Kosten wurden mit einem Thread und der gleichen CPU-Frequenz von 1.2 GHz erreicht. Die

Tabelle Tab. J.1 zeigt die erwähnten Energiekosten.

Tab. J.1.: Die minimalen und maximalen Energiekosten der Kernel-Operationen *Copy*, *Add* und *Add3* auf dem Prozessor *Haswell E5-2680v3*.

Kernel	*MIN*		*MAX*	
	J/GFlop	J/GB	J/GFlop	J/GB
Copy	-	1.74438	-	4.87262
Add	55.81644	1.74426	158.4040	4.95013
Add3	35.80656	1.74438	101.18931	5.05947

Sechs weitere Diagramme zeigen in Abb. J.7 und in Abb. J.8 die Bandbreite für die Kernel-Operationen *Copy* und *Add3* zwischen den einzelnen Komponenten des DTM-Modells (siehe Abschnitt 5.3). Die Diagramme zeigen die Ausführung der Kernel-Operationen mit den CPU-Frequenzen 1.2 GHz, 1.9 GHz und dem Turbo-Mode.

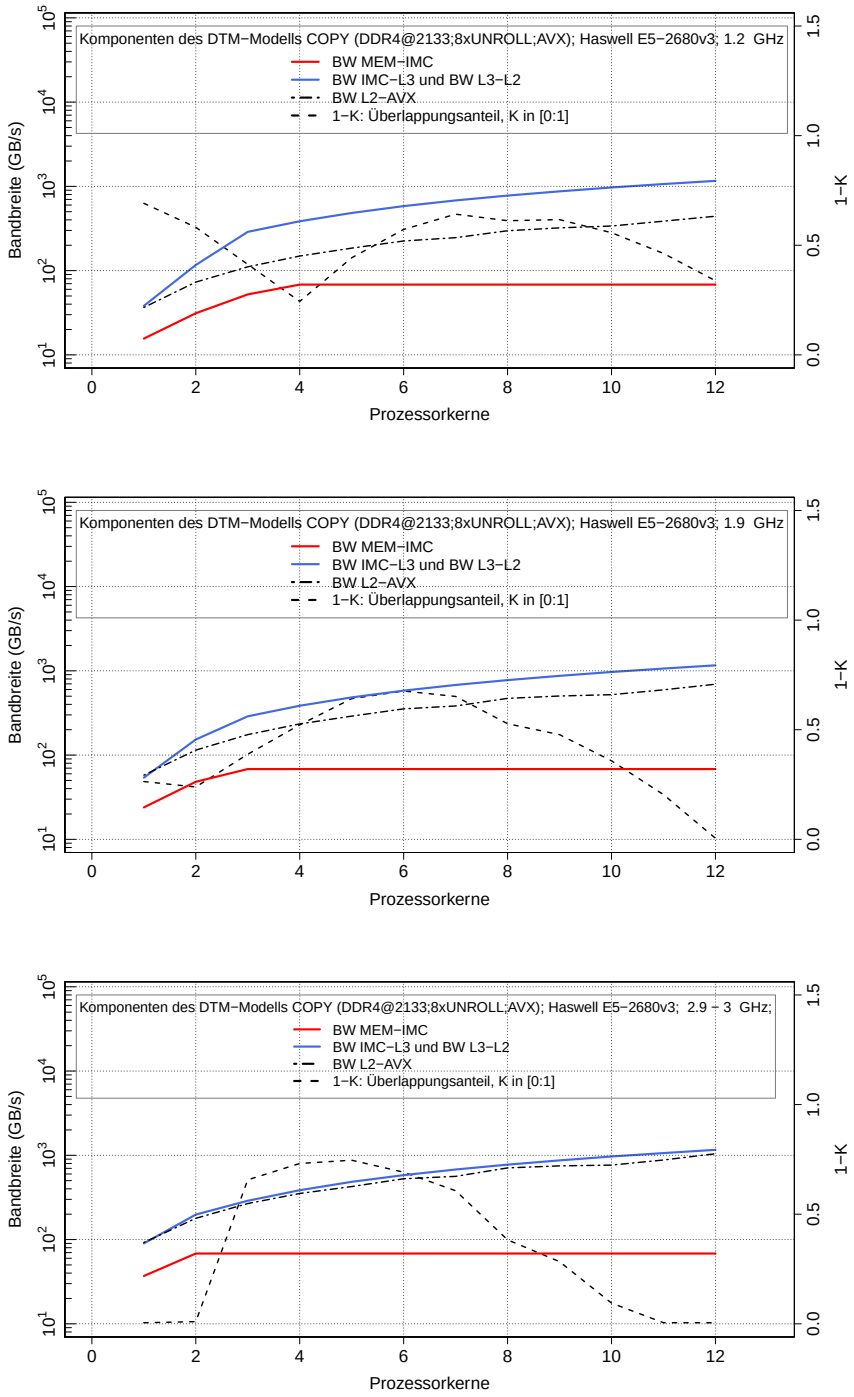

Abb. J.7.: Die Diagramme zeigen die Komponenten des DTM-Modells für die Ausführung der Kernel-Operation *Copy* auf dem Prozessor *Haswell E5-2680v3* im Falle des Haltens der Daten im Hauptspeicher, wenn die CPU-Frequenz auf 1.2 GHz (oben), 1.9 GHz (Mitte) und Turbo-Mode (unten) gesetzt war.

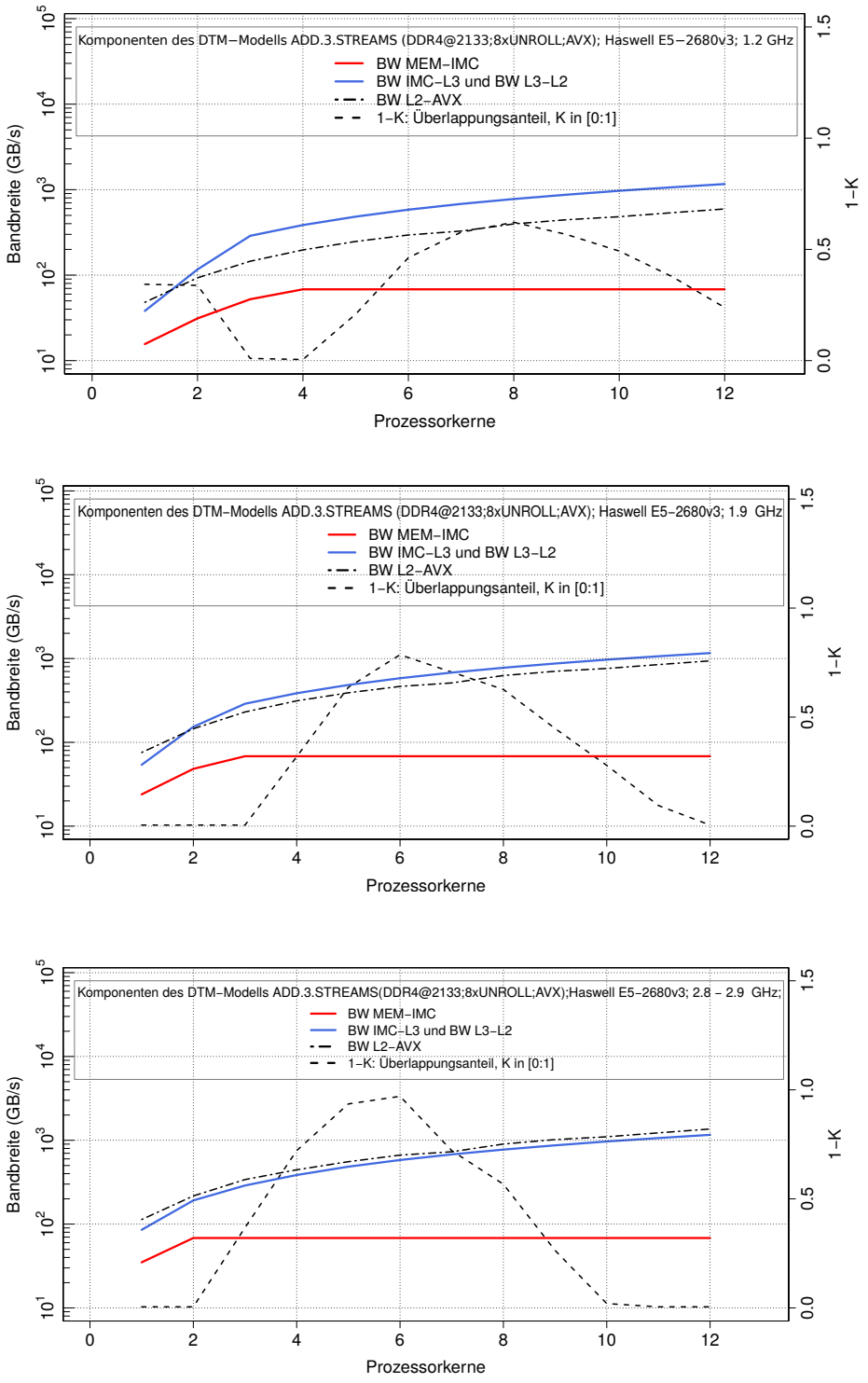

Abb. J.8.: Die Diagramme zeigen die Komponenten des DTM-Modells für die Ausführung der Kernel-Operation *Add3* auf dem Prozessor *Haswell E5-2680v3* im Falle des Haltens der Daten im Hauptspeicher, wenn die CPU-Frequenz auf 1.2 GHz (oben), 1.9 GHz (Mitte) und Turbo-Mode (unten) gesetzt war.

Eigenständigkeitserklärung

Hiermit erkläre ich, daß ich die vorliegende Dissertation mit dem Thema „Energieeffizienz von Prozessoren in High Performance Computing Anwendungen der Ingenieurwissenschaften“ selbstständig und ohne unerlaubte fremde Hilfe angefertigt habe. Es wurden von mir ausschließlich die angegebenen Quellen und Hilfen in Anspruch genommen.

Dmitry Khabi Stuttgart im November 2018