

Inhaltsverzeichnis

Vorwort	V
Inhaltsverzeichnis	VII
Abbildungsverzeichnis	XI
Tabellenverzeichnis	XIII
1 Einleitung	1
1.1 Motivation	1
1.2 Problemstellung.....	2
1.3 Gliederung der Arbeit.....	4
2 Zielsetzung und Beitrag der Arbeit	5
2.1 Zielsetzung	5
2.1.1 Konzept: Musterbasierte Suche.....	7
2.1.2 Konzept: Generierung optimierbarer Softwarearchitekturen	8
2.1.3 Konzept: Korrektheitsverifikation und Performanzoptimierung	9
2.2 Beitrag	11
2.3 Abgrenzung	12
2.4 Thesen	13
3 Grundlagen musterbasierter Parallelisierung	15
3.1 Grundbegriffe	16
3.1.1 Definitionen.....	16
3.1.2 Beispiel zu den Definitionen.....	18

3.2	Klassifikation von Analyseverfahren zur Parallelisierung	19
3.2.1	Taxonomie zu Analyseverfahren	19
3.2.2	Beispiel zu parallelisierenden Analyseverfahren	28
3.2.3	Datenstrukturen zum Erfassen der Analyseergebnisse	29
3.3	Klassifikation der Parallelverarbeitung	33
3.3.1	Aufgabenparallelität mittels <i>Master/Worker</i>	33
3.3.2	Datenparallelität mittels Gebietszerlegung	35
3.3.3	Fließbandparallelität mittels <i>Software-Pipelines</i>	36
3.4	Musterkataloge zur Parallelprogrammierung	38
3.5	Verfahren zum Auffinden von Parallelisierungsfehlern in paralleler Software	40
3.6	Verfahren zur Optimierung paralleler Softwarearchitekturen	42
4	Diskussion verwandter Arbeiten	45
4.1	Taxonomie der Softwareparallelisierung	45
4.2	Verfahren und Werkzeuge zur automatischen Parallelisierung	47
4.2.1	DOALL- und DOACROSS-Schleifenparallelisierung	47
4.2.2	Automatische Schleifenparallelisierung mit dem Polytopmodell	48
4.2.3	BONES: <i>Source-to-source</i> -Übersetzer für Grafikprozessoren	50
4.2.4	SAMBAMBA: Laufzeitadaptive Parallelisierung	52
4.2.5	Erkennung hierarchischer <i>Software-Pipelines</i>	53
4.2.6	Transformationen zur Erleichterung der Parallelprogrammierung	56
4.2.7	DISCOPOP: Erkennung von Parallelisierungspotenzial	58
4.2.8	Parallelisierungswerkzeuge	59
4.3	Ansätze zur Erkennung von Parallelisierungspotenzial	61
4.3.1	Abhängigkeitsinduzierte Analyse des kritischen Pfads	62
4.3.2	Laufzeitinduzierte Analyse des kritischen Pfads	63
4.3.3	KREMLIN: Neuentwurf des Werkzeugs GPROF für das Mehrkernzeitalter	63
4.3.4	PARCEIVE: Interaktive Parallelisierung durch dynamische Analyse	65
4.4	Ansätze zur Erkennung von Entwurfsmustern	66
4.4.1	Vergleich von Mustererkennungswerkzeugen	67
4.4.2	Erkennung von Softwarearchitekturen und Entwurfsmustern	68
4.5	Explizite Parallelprogrammierung	68
4.5.1	XJAVA: Objektorientierte Stromverarbeitung in Java	69
4.5.2	ATUNE: Performanzoptimierung paralleler Architekturen	69
4.6	Zusammenfassung, Vergleich und Bewertung	70
5	Konzepte und Lösungsansätze zur musterbasierten Parallelisierung	73
5.1	<i>AutoParPROC</i> : Erweiterbares Rahmenwerk zur Softwareparallelisierung	74
5.1.1	Anforderungen und Aufbau	75
5.1.2	Erweiterbarkeit des Erkennungsverfahrens	77
5.1.3	Erweiterbarkeit der Transformationsphase	80
5.1.4	Erweiterbarkeit der Verifikationsphase	82
5.1.5	Verwendungsmöglichkeiten für Entwickler	84
5.1.6	Fazit	85

5.2	<i>AutoPar_{PAT}</i> : Konzept der musterbasierten Suche.....	86
5.2.1	Arten der Parallelverarbeitung und Softwarearchitekturen.....	87
5.2.2	Aufgabenparallelität mit der <i>Master/Worker</i> -Architektur	88
5.2.3	Datenparallelität mittels Gebietszerlegung	92
5.2.4	Fließbandparallelität mit der <i>Pipeline</i> -Architektur	94
5.2.5	Datenerhebung zur musterbasierten Suche	98
5.2.6	Anwendungsbeispiel	102
5.2.7	Fazit	103
5.3	<i>AutoPar_{ARCH}</i> : Konzept der optimierbaren parallelen Softwarearchitekturen	103
5.3.1	Sprachanforderungen und Sprachkonzepte	104
5.3.2	Sprachentwurf	106
5.3.3	Transformation optimierbarer paralleler Softwarearchitekturen.....	115
5.3.4	Anwendungsbeispiel	119
5.3.5	Fazit	120
5.4	<i>AutoPar_{TEST}</i> : Konzept der Verifikation und Optimierung.....	120
5.4.1	Testfallbasierte Erkennung von Parallelitätsfehlern	120
5.4.2	Optimierung der Performanz paralleler Softwarearchitekturen	124
5.5	Zusammenfassung	127
6	Implementierung von <i>AutoPar</i>	129
6.1	Gesamtüberblick über die Softwarearchitektur von <i>AutoPar</i>	130
6.2	Implementierung des Parallelisierungsrahmenwerks <i>AutoPar_{PROC}</i>	131
6.2.1	Betriebsmodi von <i>AutoPar_{PROC}</i>	132
6.2.2	Betriebsmodus: Automatische Parallelisierung	133
6.3	Implementierung der Mustererkennung in <i>AutoPar_{PAT}</i>	134
6.3.1	Quellcodeanalyse in <i>SourceCodeAnalysis</i>	134
6.3.2	Erfassen von Abhängigkeiten und Laufzeitdaten in <i>AssemblySteps</i>	139
6.3.3	Suche nach Zielmustern in <i>ParallelizationCandidate</i>	142
6.3.4	Spezifikation von Zielmustern in <i>ParallelizationCandidate</i>	144
6.4	Implementierung von <i>AutoPar_{ARCH}</i>	145
6.4.1	Identifikation spezifizierter Softwarearchitekturen.....	145
6.4.2	Die parallele Laufzeitbibliothek <i>AutoPar_{RT}</i>	146
6.5	Implementierung von <i>AutoPar_{TEST}</i>	148
6.5.1	Erkennung von Parallelisierungsfehlern	148
6.5.2	Performanzoptimierung der <i>Tuning</i> -Parameter.....	149
6.6	Implementierung der grafischen Bedienoberfläche	151
6.7	Zusammenfassung	152
7	Evaluierung	153
7.1	Fallstudien.....	154
7.1.1	Videostrombearbeitung in <i>Video Processing</i>	155
7.1.2	Desktopsuche	156
7.1.3	Generische Datentypen in <i>Power Collections</i>	157
7.1.4	Geometrische Algorithmen in <i>Computational Geometry</i>	157

7.1.5	Strahlenverfolgung in <i>Ray tracing</i>	158
7.1.6	Das Sortierverfahren <i>MergeSort</i>	159
7.2	Experimentelle Ergebnisse	160
7.2.1	Reduzierung des Such- und Programmieraufwands in <i>AutoPar_{PAT}</i>	161
7.2.2	Präzision und Ausbeute des Suchverfahrens in <i>AutoPar_{PAT}</i>	165
7.2.3	Beschleunigung des Transformationsverfahrens <i>AutoPar_{ARCH}</i>	173
7.2.4	Kosten des Gesamtverfahrens <i>AutoPar</i>	179
7.2.5	Benutzerstudie zur Werkzeugintegration in MICROSOFT VISUAL STUDIO	182
7.3	Erfüllung der Thesen	187
7.4	Zusammenfassung der Evaluierungsergebnisse	188
8	Zusammenfassung und Ausblick	189
8.1	Zusammenfassung der Arbeit	189
8.2	Ausblick und zukünftige Arbeiten	192
8.2.1	Hinzunahme weiterer Muster	192
8.2.2	Verarbeitung paralleler Software	193
8.2.3	Heterogene Parallelisierung	193
8.2.4	Modellbasierte Fehlererkennung	194
8.2.5	Musterbasierte Fehlerkorrektur	194
8.2.6	Fazit	194
Anhänge		197
A.	Liste eigener Publikationen	198
B.	Vorstudie zur Parallelisierung von Aufgabenparallelität mittels Futures	200
C.	Datenerhebung zur musterbasierten Parallelisierung	202
D.	Das Paket KIT.PhD.AutoPar.Detection	203
E.	Sprachgrammatik der <i>Tunable Architecture Description Language (TADL)</i>	204
F.	Das Paket KIT.PhD.AutoPar.Transformation	205
G.	Die <i>Tuning</i> -Datei <i>TuningParameters.xml</i>	206
H.	Fragebogen: Werkzeugeigenschaften (Gruppen <i>G_{PAT}</i> und <i>G_{INTEL}</i>)	207
I.	Fragebogen: Gewünschte Eigenschaften eines Parallelisierungswerkzeugs (<i>G_{Man}</i>)	209
J.	Auswertung der Fragebögen	210
K.	Die Werkzeugintegration <i>PATTY</i>	211
Literaturverzeichnis		213

Musterbasierte Parallelisierung sequenzieller
Anwendungen

Konzept und Implementierung eines Verfahrens zur
Softwaretransformation

Molitorisz, K.

2016, XIII, 226 S. 89 Abb., 9 Abb. in Farbe., Softcover

ISBN: 978-3-658-15094-5