

---

---

Konzepte zur Optimierung der  
Skalierbarkeit von parallelen  
Fahrzeugkollisionsberechnungen und ihre  
industrielle Realisierbarkeit

---

---

Josef Weidendorfer



Institut für Informatik  
Lehrstuhl für Rechnerorganisation

**Konzepte zur Optimierung der Skalierbarkeit von  
parallelen Fahrzeugkollisionsberechnungen und ihre  
industrielle Realisierbarkeit**

Josef Weidendorfer

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität  
München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. H. M. Gerndt

Prüfer der Dissertation:

1. Univ.-Prof. Dr. A. Bode

2. Univ.-Prof. Dr. E. Jessen, emeritiert

Die Dissertation wurde am 11.12.2002 bei der Technischen Universität München eingereicht  
und durch die Fakultät für Informatik am 13.02.2003 angenommen.





Research Report Series

Lehrstuhl für Rechnertechnik und  
Rechnerorganisation (LRR-TUM)  
Technische Universität München

<http://www.bode.in.tum.de/>

Editor: Prof. Dr. A. Bode

Vol. 29

---

---

**Konzepte zur Optimierung  
der Skalierbarkeit von parallelen  
Fahrzeugkollisionsberechnungen und  
ihre industrielle Realisierbarkeit**

---

---

**Josef Weidendorfer**

**SHAKER**

**V E R L A G**

Aachen 2003

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

*Weidendorfer, Josef:*

Konzepte zur Optimierung der Skalierbarkeit von parallelen  
Fahrzeugkollisionsberechnungen und ihre industrielle  
Realisierbarkeit / Josef Weidendorfer.

Aachen : Shaker, 2003

(Research Report Series/Lehrstuhl für Rechnertechnik und Rechner-  
organisation (LRR-TUM), Technische Universität München ; Vol. 29)

Zugl.: München, Techn. Univ., Diss., 2003

ISBN3-8322-1421-6

Copyright Shaker Verlag 2003

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen  
oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungs-  
anlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 3-8322-1421-6

ISSN 1432-0169

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen

Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9

Internet: [www.shaker.de](http://www.shaker.de) • eMail: [info@shaker.de](mailto:info@shaker.de)

---

---

# Kurzfassung

---

---

Parallele Anwendungen im Bereich des Hochleistungsrechnens erfordern neben der Implementierung geeigneter numerischer Algorithmen die effiziente Nutzung der in einem Parallelrechner bereitgestellten Kommunikationsstruktur und die Integration dynamischer Lastbalancierung, falls verwendete Algorithmen unvorhersehbares und damit statisch schlecht aufteilbares Lastaufkommen besitzen. Die Verflechtung dieser teilweise plattformabhängigen Programmteile, die bei Benutzung üblicher Programmiermodelle notwendigerweise entsteht, führt zu unflexibler Software, die nur schwer Weiterentwicklung, Wartung und Anpassung an neue Plattformen erlaubt.

Zur Lösung dieses Problems stellt die vorliegende Arbeit ein Programmiermodell für die Klasse der feldbasierten iterativen Anwendungen vor, das es erlaubt, die eigentlichen Berechnungen des Programms von der notwendigen Kommunikation und Synchronisation zwischen den parallel ablaufenden Prozessen zu trennen. Dadurch wird eine unabhängig arbeitende Lastbalancierung ermöglicht. Obwohl das Modell dem Programmierer gemeinsamen Speicher zur Verfügung stellt, kann es mit *Message Passing* implementiert werden. Grundlage dabei ist, daß die Spezifikation a priori bekannter Abhängigkeiten von Anweisungen, deren Zugriffsverhalten auf gemeinsame Daten explizit anzugeben ist, die automatisierte Berechnung der zwischen den Anweisungen zur Konsistenzhaltung durchzuführenden Kommunikation erlaubt. Vermieden wird dadurch der Zeitverlust, der bei einer üblichen Realisierung von verteilt gemeinsamem Speicher entsteht, in der erst auf Anforderung ein Abgleich zur Konsistenzsicherung ausgeführt wird.

Die Praxistauglichkeit des Programmiermodells wird durch den erfolgreichen Einsatz in einer Industrianwendung, einer Software zur Simulation von Fahrzeugkollisionen gezeigt. Diese Simulationen werden von der Automobilindustrie neben physikalischen Kollisionsversuchen heutzutage deshalb verstärkt eingesetzt, da bei gleicher Entwicklungszeit weitaus mehr Konstruktionsalternativen erprobt werden können und damit die Sicherheit von Fahrzeuginsassen bei Verkehrsunfällen gesteigert werden kann. Der hohe Rechenleistungsbedarf macht allerdings Parallelisierung zwingend erforderlich. Ein diese Arbeit begleitendes Projekt untersuchte die Skalierungsprobleme, die in der bei der BMW AG benutzten Simulationssoftware PAM-CRASH<sup>TM</sup> auftreten. Der Einsatz des oben beschriebenen Programmiermodells zur Einführung von Lastbalancierung kann diese Probleme reduzieren, wie Verbesserungen in der Laufzeit praxisrelevanter Simulationsläufe belegen.



---

---

# Danksagung

---

---

Voraussetzung für das Entstehen dieser Arbeit war das Zustandekommen einer Kooperation zur Untersuchung von Skalierbarkeitsproblemen bei Kollisionssimulationen. Diese wurde von Prof. Dr. Arndt Bode (LRR-TUM — Lehr- und Forschungseinheit Rechnertechnik und Rechnerorganisation der Technischen Universität München), Prof. Dr. Thomas Ludwig (LRR-TUM), Dr.-Ing. Michael Holzner (BMW AG) und Christian Tanasescu (SGI — Silicon Graphics, Inc.) in die Wege geleitet unter Mitwirkung von Jan Clinckemaiilie (ESI Group — Engineering Systems International). Ich danke Prof. A. Bode, daß er mir die interessante Aufgabe der Durchführung des daraus entstandenen Projekts überlassen hat.

Dabei war ich auf die ständige Unterstützung von verschiedenen Seiten angewiesen. Zum einen haben Prof. A. Bode und die Mitarbeiter des Lehrstuhls, v. a. Dr. habil. Peter Luksch die Arbeit durch ihr Fachwissen helfend begleitet.

Zum anderen bin ich dankbar für die große Unterstützung durch die BMW AG, insbesondere den Kollegen der Crash-Simulationsabteilung (v. a. Horst-Uwe Mader), der SGI Administrationsgruppe um Martin Saller und den ESI-Mitarbeitern vor Ort (Fabrice Payne, Martin Oehm). Deren Hilfeleistung und die Bereitstellung der verfügbaren Hardware waren essentiell für die Programmierung und Bewertung des entstandenen Prototyps.

Dieser Prototyp hätte ohne die Überlassung wichtiger Teile der PAM-CRASH Software durch die ESI Group nicht entstehen können. Bei der Einarbeitung in den Quellcode und der Interpretation der Performanzprobleme standen Jan Clinckemaiilie (ESI) und Herve Chevanne (SGI) stets hilfreich zur Seite.

Desweiteren möchte ich Hrn. Prof. E. Jessen danken, daß er das Amt des Zweitgutachters für diese Arbeit übernommen hat und mit vielen kleinen Tipps zur Verbesserung beigetragen hat.

Nicht zuletzt danke ich meiner Frau für die Geduld und ständige Aufmunterung in der letzten Phase des Schreibens dieser Arbeit.

Josef Weidendorfer



---

---

# Inhaltsverzeichnis

---

---

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Kollisionssimulation im Automobilbau . . . . .	1
1.1.1	Gründe für den Einsatz von Kollisionssimulationen . . . . .	1
1.1.2	Notwendigkeit der Nutzung paralleler Simulationssoftware . . . . .	2
1.1.3	Konkreter Rahmen der Arbeit . . . . .	3
1.2	Zielsetzung . . . . .	3
1.2.1	Probleme paralleler numerischer Anwendungen . . . . .	3
1.2.2	Anforderungen an Simulationsanwendungen aus Benutzersicht . . . . .	4
1.2.3	Anforderungen an Simulationsanwendungen aus Entwicklersicht . . . . .	6
1.2.4	Motivation der angestrebten Lösung . . . . .	7
1.2.5	Konkrete Anforderungen einer Lösung . . . . .	9
1.3	Aufbau der Arbeit . . . . .	10
<b>2</b>	<b>Simulationen in der Strukturmechanik</b>	<b>11</b>
2.1	Modellbildung und Analyse . . . . .	12
2.1.1	Erzeugung des Modellgitters . . . . .	13
2.1.2	Zuordnung von Materialeigenschaften . . . . .	13
2.1.3	Festlegung von Randbedingungen . . . . .	14
2.1.4	Simulationsrechnung . . . . .	14
2.1.5	Nachbearbeitung und Interpretation der Ergebnisse . . . . .	14
2.2	Die Methode der finiten Elemente . . . . .	15
2.2.1	Die Variationsrechnung . . . . .	15
2.2.2	Näherungsweise Lösung eines Variationsproblems . . . . .	16
2.2.3	Nutzung der Variationsrechnung in der FEM . . . . .	17
2.2.4	Nichtlineare zeitabhängige Probleme . . . . .	19
2.2.5	Beschleunigungsmöglichkeiten für die FEM . . . . .	20
2.3	Eigenheiten von Kollisionssimulationen . . . . .	23
2.3.1	Anforderungen . . . . .	23
2.3.2	Eingangsdaten . . . . .	24
2.3.3	Kontaktbehandlung . . . . .	25
2.4	Informatische Sichtweise . . . . .	27
2.4.1	Datenstrukturen . . . . .	27
2.4.2	Grobstruktur eines FEM-Lösers . . . . .	28
<b>3</b>	<b>Konzepte der Parallelisierung</b>	<b>33</b>
3.1	Überblick . . . . .	33
3.1.1	Leistungsmaße für parallele Anwendungen . . . . .	33

---

3.1.2	Nutzung von Parallelität . . . . .	35
3.1.3	Schritte zu automatisierter Parallelisierung numerischer Anwendungen . . . . .	37
3.2	Architekturen für Parallelrechner . . . . .	38
3.2.1	Klassifikationsschemata . . . . .	38
3.2.2	Beispiele von Parallelrechnern . . . . .	39
3.3	Entwicklung paralleler numerischer Anwendungen . . . . .	41
3.3.1	Automatische Parallelisierung . . . . .	42
3.3.2	Parallele Programmiermodelle und -paradigmen . . . . .	43
3.3.3	Lastaufteilung . . . . .	47
3.3.4	Fehlerbeseitigung und Performanzoptimierung . . . . .	51
3.3.5	Nutzung in einer Produktionsumgebung . . . . .	53
<b>4</b>	<b>Parallelisierung einer Kollisionssimulation</b> . . . . .	<b>55</b>
4.1	Die SMP-Version von PAM-CRASH . . . . .	55
4.1.1	Programmstruktur eines expliziten FEM-Lösers . . . . .	56
4.1.2	Ergebnis einer Compilerparallelisierung . . . . .	57
4.1.3	Nachteile der SMP-Version . . . . .	58
4.2	Die MPP-Version von PAM-CRASH . . . . .	59
4.2.1	Nutzung von Nachrichtenaustausch . . . . .	59
4.2.2	Aufteilung eines Modellgitters . . . . .	60
4.2.3	Zweiteilung der Kontaktsuche . . . . .	60
4.2.4	Programmstruktur des explizit parallelisierten Lösers . . . . .	61
4.3	Analyse der Performanzprobleme in der MPP-Version . . . . .	62
4.3.1	Synchronisation beim Kommunikationsaustausch . . . . .	62
4.3.2	Durch FE-Berechnung erzeugte Last . . . . .	63
4.3.3	Durch Kontaktberechnung erzeugte Last . . . . .	63
4.3.4	Untersuchte Lastfälle . . . . .	64
4.3.5	Verfahren zur Analyse von Performanzproblemen . . . . .	64
4.4	Erkannte Lastbalancierungsprobleme in der Kontaktberechnung . . . . .	70
4.4.1	Globale Kontaktsuche . . . . .	70
4.4.2	Lokale Kontaktsuche und -behandlung . . . . .	72
4.5	Konkrete Lösungsmöglichkeiten . . . . .	74
4.5.1	Hierarchische Kontaktsuche . . . . .	74
4.5.2	Getrennte Datenaufteilung . . . . .	75
4.5.3	Überpartitionierung . . . . .	75
4.5.4	Parallele Nutzung verschiedener Programmiermodelle . . . . .	76
4.6	Ein verallgemeinerter Lösungsansatz . . . . .	76
4.6.1	Anforderungen . . . . .	76
4.6.2	Lösungs idee . . . . .	78
<b>5</b>	<b>Ein Programmiermodell für anwendungsintegrierte Lastbalancierung</b> . . . . .	<b>81</b>
5.1	Anforderungen . . . . .	81
5.1.1	Implementierbarkeit mittels Nachrichtenaustausch . . . . .	81
5.1.2	Ermöglichung einer automatisierten Lastbalancierung . . . . .	81
5.2	Aktionssysteme als Basis des Modells . . . . .	82
5.2.1	Grundlegende Begriffe . . . . .	82
5.2.2	Übertragung eines sequentiellen in ein paralleles Aktionssystem . . . . .	83

5.2.3	Aktionssysteme mit gemeinsamem Speicher . . . . .	85
5.3	Aktionssysteme mit scheinbar gemeinsamem Speicher . . . . .	87
5.3.1	Einführung von prozessorlokalem Speicher . . . . .	87
5.3.2	Erweiterte Bernsteinbedingung . . . . .	88
5.3.3	Simulation eines gemeinsamen Speichers . . . . .	89
5.3.4	Vereinfachungen der Simulation . . . . .	92
5.4	Unterscheidung von Speicherbereichen . . . . .	95
5.4.1	Rein prozessorlokal genutzte Bereiche . . . . .	95
5.4.2	Unterteilung von Bereichen für die gemeinsame Nutzung . . . . .	96
5.4.3	Maßnahmen zur vereinfachten Realisierbarkeit . . . . .	97
5.5	Anwendungsorientierte Eigenschaften . . . . .	98
5.5.1	Parallele Reduktionsoperationen . . . . .	98
5.5.2	Vermeidung von Deadlocks bei der Realisierung mit Nachrichtenaustausch . . . . .	99
5.6	Nutzung von iterativem Programmverhalten . . . . .	102
5.6.1	Einführung von Wiederholungsangaben . . . . .	102
5.6.2	Parallel ablaufende Mikro-Aktionssysteme . . . . .	103
5.7	Umsetzung paralleler Programmierparadigmen . . . . .	104
5.7.1	Master-Slave . . . . .	104
5.7.2	Workpool . . . . .	105
5.7.3	Pipeline . . . . .	107
5.8	Abbildung von Aktionssystemen auf reduzierte Prozessoranzahlen . . . . .	108
5.8.1	Statische Aufteilung paralleler Aktionen . . . . .	108
5.8.2	Dynamische Aufteilung paralleler Aktionen . . . . .	110
5.8.3	Verteilte Aufteilungsbestimmung . . . . .	111
5.8.4	Lastabschätzungen im Modell . . . . .	115
<b>6</b>	<b>Realisierung des Programmiermodells im EPaCTS-Framework</b> . . . . .	<b>119</b>
6.1	Nutzung aus Entwicklersicht . . . . .	119
6.1.1	Simulation einer 1-dimensionalen Temperaturobreitung . . . . .	120
6.1.2	Sequentielles Programm . . . . .	120
6.1.3	Paralleles Programm mit Nutzung des EPaCTS Frameworks . . . . .	120
6.2	Architekturüberblick . . . . .	122
6.2.1	Der Kommunikationsmanager . . . . .	122
6.2.2	Die Monitor-Komponente . . . . .	123
6.2.3	Der Migrationsmanager und die Lastbalancierungskomponente . . . . .	124
6.2.4	Schnittstelle zu Debugger und Ablaufverfolgung . . . . .	124
6.3	Das virtuelle Feld als zentrale gemeinsame Datenstruktur . . . . .	125
6.3.1	Zugriffsrelationen . . . . .	125
6.3.2	Zugriffsrechte . . . . .	126
6.4	Kommunikationsentscheidungen . . . . .	127
6.5	Unterstützung für automatische Lastbalancierung . . . . .	129
6.5.1	Nutzung von Anwendungswissen in der Lastbalancierung . . . . .	129
6.5.2	Auslösung einer Lastbalancierung . . . . .	130
6.5.3	Visualisierung der Programmphasen im Beispiel . . . . .	131
6.5.4	Anwendungsunabhängige Lastbalancierungsalgorithmen . . . . .	132
6.6	Verbesserungen durch Zerteilung von Berechnungsphasen . . . . .	133

---

6.6.1	Zweiteilung eines lesenden Zugriffszustandes . . . . .	133
6.6.2	Vermeidung von Deadlock-Problemen . . . . .	134
6.6.3	Bestimmung unmittelbar ausführbarer Aktionen . . . . .	134
6.6.4	Zweiteilung eines schreibenden Zugriffszustands . . . . .	135
6.7	Ergänzungen für komplexere Anwendungen . . . . .	135
6.7.1	Gleichzeitige Nutzung virtuelle Felder . . . . .	135
6.7.2	Nutzung eines virtuellen Felds in unterschiedlichen Algorithmen . . . . .	136
<b>7</b>	<b>Nutzung des EPaCTS-Frameworks für PAM-CRASH MPP</b>	<b>139</b>
7.1	Benutzung virtueller Felder . . . . .	139
7.1.1	Ideales Vorgehen . . . . .	140
7.1.2	Reales Vorgehen im Projekt . . . . .	140
7.2	Einführung von Lastbalancierung . . . . .	140
7.2.1	Datenaufteilung für verschiedene Algorithmen . . . . .	140
7.2.2	Lastbalancierung für die Kontaktberechnung . . . . .	142
7.3	Benutzte Lastbalancierungsalgorithmen . . . . .	145
7.3.1	Exakte Neuaufteilung von Lasten . . . . .	145
7.3.2	Balancierung durch Lastverschiebung . . . . .	146
7.4	Priorisierung einer Plattform . . . . .	147
7.5	Koexistenz von virtuellen Feldern und MPI . . . . .	148
<b>8</b>	<b>Ergebnisse</b>	<b>151</b>
8.1	Lastvermindernde Optimierungen . . . . .	151
8.2	Stauchung eines Rechteckstabes . . . . .	153
8.2.1	Zeitmessungen . . . . .	154
8.2.2	Lastbalancierung in regulären Iterationen . . . . .	156
8.2.3	Verhalten der Lastbalancierung bei externer Störung . . . . .	157
8.2.4	Lastverhältnisse in Iterationen mit globaler Kontaktsuche . . . . .	159
8.3	Frontalcrash eines Fahrzeugs . . . . .	161
8.3.1	Zeitmessungen . . . . .	161
8.3.2	Lastbalancierung in regulären Iterationen . . . . .	164
8.3.3	Lastbalancierung in Iterationen mit globaler Kontaktsuche . . . . .	167
8.3.4	Einfluß externer Störungen . . . . .	170
8.4	Lastbalancierung bei größeren Fahrzeugmodellen . . . . .	171
8.5	Diskussion der weiterhin vorhandenen Skalierungsprobleme . . . . .	172
8.6	Gewonnene Erkenntnisse . . . . .	173
8.6.1	Allgemeine Erfahrungen . . . . .	174
8.6.2	Ratschläge für PAM-CRASH MPP . . . . .	175
<b>9</b>	<b>Zusammenfassung und Ausblick</b>	<b>177</b>
9.1	Behandeltes Problem . . . . .	177
9.2	Konzeptuelle Lösung . . . . .	179
9.3	Praktische Anwendung der Lösung . . . . .	180
9.4	Ausblick . . . . .	182
9.4.1	Nutzung der konzeptuellen Arbeiten in weiterführender Forschung . . . . .	182
9.4.2	Vorteile des praktischen Einsatzes des Programmiermodells . . . . .	182
9.4.3	Fortführung der Untersuchungen im Umfeld von PAM-CRASH . . . . .	183

<b>Literaturverzeichnis</b>	<b>185</b>
<b>Index</b>	<b>197</b>



---

---

# Abbildungsverzeichnis

---

---

2.1	Ein einfaches zweidimensionales FE-Modell . . . . .	12
2.2	Eine Funktionenschar als Lösungsraum eines Variationsproblems . . . . .	16
2.3	Finite Elemente in einer Dimension . . . . .	18
2.4	Finite Elemente in zwei Dimensionen . . . . .	18
2.5	Finite Elemente in drei Dimensionen . . . . .	19
2.6	Lastfälle für Kollisionsversuche . . . . .	24
2.7	Wirkungsweise einer “Penalty Method” . . . . .	26
2.8	Ein FE-Gitter als bipartiter Graph . . . . .	27
2.9	Datenstruktur für einen FE-Knoten . . . . .	28
2.10	Datenstruktur für ein FE-Element . . . . .	28
2.11	Programmstruktur eines impliziten FEM-Lösers . . . . .	29
2.12	Programmstruktur eines expliziten FEM-Lösers . . . . .	30
3.1	Verbindungsstrukturen bei Parallelrechnern . . . . .	39
3.2	Parallele Programmierparadigmen . . . . .	44
4.1	Programmstruktur eines expliziten FE-Lösers . . . . .	56
4.2	Scheduling-Verfahren bei automatischer Schleifenparallelisierung . . . . .	58
4.3	Aufteilung eines FE-Modellgitters . . . . .	60
4.4	Programmstruktur PAM-CRASH MPP, E/A-Strang . . . . .	61
4.5	Programmstruktur PAM-CRASH MPP, Arbeiter-Strang . . . . .	62
4.6	CAMAS-Modellgitter am Ende der Simulationsrechnung . . . . .	65
4.7	Graphische Visualisierung eines Laufzeitprofils . . . . .	66
4.8	Time-History-Plot eines Zeitschritts in PAM-CRASH . . . . .	69
4.9	Visualisierung der Lastungleichheit in einem Supercycle . . . . .	71
4.10	Visualisierung der Lastungleichheit in der Kontaktbehandlung . . . . .	73
5.1	Abhängigkeitsgraph für parallelisierten Code . . . . .	86
5.2	Konstruktion für scheinbar gemeinsamen Speicher . . . . .	90
5.3	Skizze zu Satz 5.6 . . . . .	93
5.4	Skizze zu Satz 5.7 . . . . .	95
5.5	Aktionssystem mit 3 unterschiedlichen gemeinsamen Speicherbereichen . . . . .	96
5.6	Deadlock-Gefahr bei Nutzung von Nachrichtenaustausch . . . . .	100
5.7	Feste Kommunikationsreihenfolge zur Deadlock-Vermeidung . . . . .	102
5.8	Aufteilung in parallele kleine Aktionssysteme . . . . .	103
5.9	Mikro-Aktionssysteme für das Master/Slave Paradigma . . . . .	105
5.10	Mikro-Aktionssysteme für das Workpool Paradigma . . . . .	106

5.11	Mikro-Aktionssystem $G_j$ für das Pipeline-Paradigma . . . . .	107
5.12	Benutzung von Aufteilungen . . . . .	109
5.13	Migrationsgruppen für ein Rechnercluster . . . . .	113
5.14	Migrationsgruppen für ein 2-dimensionales Prozessgitter . . . . .	113
6.1	Simulation einer 1-dimensionalen Temperaturlausbreitung . . . . .	119
6.2	Sequentieller C-Code für eine 1-dimensionale Relaxation . . . . .	120
6.3	Parallele 1-dimensionale Relaxation im EPaCTS-Framework . . . . .	121
6.4	Zugriffsrelationen bei der 1-dimensionalen Relaxation . . . . .	122
6.5	Module des EPaCTS-Frameworks . . . . .	123
6.6	Abänderungen der Relaxation für migrierbare Aktionen . . . . .	130
6.7	Visualisierung des Relaxationsablaufs mit Lastbalancierung . . . . .	131
7.1	Vergleich getrennter und gemeinsamer Datenaufteilung (I) . . . . .	141
7.2	Vergleich getrennter und gemeinsamer Datenaufteilung (II) . . . . .	142
7.3	Lastbalancierungsalgorithmus zur Neuaufteilung . . . . .	145
7.4	Lastbalancierungsalgorithmus für Migrationsentscheidungen . . . . .	146
8.1	Detailverbesserungen einer Lastverminderung . . . . .	153
8.2	Schematische Darstellung des Trägermodells . . . . .	153
8.3	Trägermodell, reguläre Iterationen bei 0, 40, 80, 120 und 160 Minuten . . . . .	156
8.4	Wirkung einer Lastbalancierung . . . . .	158
8.5	Auswirkungen von Störungen auf die Lastbalancierung . . . . .	159
8.6	Trägermodell, Lastunausgeglichenheit im Supercycle . . . . .	160
8.7	CAMAS-Modell, reguläre Iteration bei Simulationsbeginn . . . . .	164
8.8	CAMAS-Modell, reguläre Iteration bei Simulationsende . . . . .	166
8.9	CAMAS-Modell, Iteration mit globaler Kontaktsuche bei Simulationsbeginn . . . . .	168
8.10	CAMAS-Modell, Iteration mit globaler Kontaktsuche bei Simulationsende . . . . .	169
8.11	Einfluß externer Störungen bei einem 60-Prozessor-Lauf . . . . .	171
8.12	Visualisierung der Iterationen eines großen Fahrzeugmodells . . . . .	172

---

---

# Tabellenverzeichnis

---

---

4.1	Verwendete Modelle für genauere Untersuchung . . . . .	64
4.2	Laufzeiten für die MPP-Version von PAM-CRASH . . . . .	64
4.3	Profiling: Laufzeiten von Funktionen bei 12 Prozessoren . . . . .	65
4.4	Zeitanteil von Funktionen abhängig vom Modell . . . . .	67
4.5	Zeitstatistik von PAM-CRASH MPP . . . . .	68
8.1	Auswirkung von Lastverminderung in der Kontaktberechnung . . . . .	152
8.2	Trägermodell, Laufzeitverbesserungen durch Lastbalancierung . . . . .	154
8.3	Trägermodell, Kontaktberechnung ohne Lastbalancierung . . . . .	155
8.4	Trägermodell, Kontaktberechnung mit Lastbalancierung . . . . .	155
8.5	CAMAS-Modell, Laufzeitverbesserungen durch Lastbalancierung . . . . .	162
8.6	CAMAS-Modell, Kontaktberechnung ohne Lastbalancierung . . . . .	163
8.7	CAMAS-Modell, Kontaktberechnung mit Lastbalancierung (RI) . . . . .	163
8.8	CAMAS-Modell, Kontaktberechnung mit Lastbalancierung (SC+RI) . . . . .	163
8.9	CAMAS-Modell, Dauer einer regulären Iteration ohne/mit Lastbalancierung	165
8.10	CAMAS-Modell, Dauer eines Supercycles ohne/mit Lastbalancierung . . . . .	168