

# Abschlussbericht des Projekts Ru 422/7-5 (Kennwort: DiME-2)

*Datenlokale Iterationsverfahren  
zur effizienten Lösung partieller Differentialgleichungen*

## 1 Allgemeine Angaben

### **Antragsteller:**

- Prof. A. Bode, München
- Prof. U. Rude, Erlangen
- Prof. W. Karl, Karlsruhe (war München)

### **Institut/Lehrstuhl:** Kooperation zwischen

- Lehrstuhl für Rechnerarchitektur und Rechnerorganisation /  
Parallelrechnerarchitektur (LRR), I 10, Fakultät für Informatik  
Technische Universität München  
Boltzmannstr. 3, 85748 Garching bei München
- Lehrstuhl für Systemsimulation, Department Informatik,  
Universität Erlangen-Nürnberg (FAU)  
Cauerstraße 6, 91058 Erlangen
- Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung,  
Institut für Technische Informatik (ITEC), Fakultät für Informatik  
Universität Karlsruhe (TH)  
Zirkel 2, 76131 Karlsruhe

**Thema des Projekts:** DiME - Datenlokale Iterationsverfahren zur effizienten Lösung partieller Differentialgleichungen

**Berichtszeitraum:** 1.10.2003 - 31.12.2007

Es standen noch Personalmittel aus, die bis November 2007 ausgegeben wurden. Die über den offiziellen Berichtszeitraum für das Projekt hinaus durchgeführten Arbeiten sind in diesem Bericht eingeschlossen.

## 2 Zusammenfassung

Im vorhergehenden „DiME“ wurden verschiedene Cache- und Speicherlayout-Optimierungen zur effizienten Lösung partieller Differentialgleichungen mit Hilfe datenlokaler Iterationsverfahren untersucht. Dabei wurden v. a. RISC-Architekturen (wie Alpha-Prozessoren) als Grundlage benutzt, und es stand die Suche nach möglichen Optimierungsstrategien im Vordergrund. Im vorliegenden Projekt „DiME-2“ wurde erkannt, dass eine naive Nutzung dieser Strategien auf heutigen Prozessoren, insbesondere der x86-Architektur, häufig nicht erfolgreich ist, da komplexe Architektureigenschaften, die zu der enormen Leistungsfähigkeit wesentlich beitragen, nicht berücksichtigt werden. Dazu zählen automatische Vorauslademechanismen und die dabei verwendeten Heuristiken, die Abhängigkeit der verfügbaren Speicherleistung von Zugriffsmustern und die Erweiterung um Vektoreinheiten (SIMD).

Cache- und Speicherlayout-Optimierungen – so wie sie bereits in DiME empfohlen werden – können aber zu teils großen Leistungssteigerungen führen, wenn sie unter stärkerer Berücksichtigung der jeweiligen Zielarchitektur angewandt werden, was mitunter zeitaufwendig handoptimierten Assemblercode erfordert. Neben Blocking-Techniken, die eine möglichst häufige Nutzung von in den Cache geladenen Daten ermöglichen, ist die Maximierung der effektiven Speicherbandbreite entscheidend. Je nach Architektur und Anwendung sind hier ein günstiges Speicherlayout, die Einstreuung von Vorausladeoperationen oder sogar die Verwendung dedizierter Threads sinnvoll, die den effizienten Speicherzugriff übernehmen. Eine zweischneidige Entwicklung wurde bei den Compilern festgestellt, die viele Optimierungstechniken zunehmend automatisch ausführen können, aber dabei eingesetzte Optimierungsstrategien zum Schlechteren verändern können. Entwicklungen in der Prozessortechnik und Compiler-Technik stellen zudem oft widersprüchliche Anforderungen, zwischen denen ein guter Kompromiss gefunden werden muss; häufig sollte ein Speicherlayout z. B. die Anforderungen der immer wichtigeren SIMD-Rechenwerke bezüglich Ausrichtung genauso erfüllen wie die Anzahl der Datenströme entsprechend der Fähigkeiten der automatischen Prefetch-Einheiten begrenzen, zugleich aber eine effiziente Adressberechnung durch den Compiler ermöglichen. Experimente mit der Cell Broadband Engine haben zudem gezeigt, dass viele der untersuchten oder entwickelten Optimierungsverfahren auch bei einschneidenden architekturellen Änderungen übertragen und somit auf zukünftigen Plattformen eingesetzt werden können.

Nachdem immer weniger eine Standardstrategie zur Optimierung einer Anwendung angegeben werden kann, ist eine genaue vorherige Analyse von Algorithmus und Zielplattform unverzichtbar. Dabei muss bei Optimierungsschritten regelmäßig überprüft werden, ob die implementierten Techniken den gewünschten Erfolg bringen. Um die Ursache eines Leistungsengpasses feststellen zu können, sind detaillierte Analysen des Laufzeitverhaltens mit entsprechenden Werkzeugen unverzichtbar. Neben klassischen Techniken wie manueller oder Compiler-gestützter Instrumentierung oder der Nutzung von „Performance Countern“ erweist sich hier die Verwendung von Simulatoren als besonders sinnvoll, um Vermutung zu überprüfen oder neue Hinweise zu erhalten. Neben der Simulation verschiedener Architekturen oder Prozessorversionen können hier auch komplexere Zusammenhänge erfasst oder ausgewertet werden — zum Beispiel, welcher Anteil an Daten in den Cache geladen wurde, ohne jemals genutzt zu werden, und welcher Anteil vor einer Verdrängung mehrfach genutzt wurde. Die in diesem Projekt entwickelten Werkzeuge (Cachesimulator, Visualisierung) wurden als Open-Source veröffentlicht, und werden dabei nicht nur in verschiedenen Open-Source-Entwicklungskreisen (wie KDE, OpenOffice) oder Software-Unternehmen eingesetzt, wie verschiedene Anfragen dazu gezeigt haben, sondern auch in anderen Forschungsvorhaben (etwa „Adaptor“ vom Fraunhofer-Institut) und in der Lehre.

## 3 Arbeits- und Ergebnisbericht

### 3.1 Ausgangsfragen und Zielsetzung

Das Projekt ist eine Fortsetzung des von der DFG geförderten Projekts „DiME“.

Forschungsschwerpunkt ist die Frage, auf welche Weise sich ein konkretes Problem, nämlich die Lösung einer diskretisierten, partiellen Differentialgleichung, möglichst effizient auf aktueller Rechnerhardware mit Hilfe eines iterativen Lösungsverfahrens berechnen lässt. Ein wesentliches Merkmal des untersuchten Problems ist der wiederkehrend nötige Zugriff auf große Datenmengen, die in diesem Falle aus dem Modellgitter bestehen. Dadurch wird die Speicheranbindung der begrenzende Faktor für die Leistungsfähigkeit des Verfahrens.

Das ursprünglichen Projekt („DiME“) erreichte durch algorithmusneutrale Änderungen eines herkömmlichen Mehrgitterverfahrens in der Implementierung auf den damals untersuchten Hardwareplattformen einen teilweise erheblichen Geschwindigkeitsgewinn (bis 400 % auf Alpha-Prozessoren) bei zweidimensionalen Modellen. Der Fortsetzungsantrag wirft deshalb v. a. die Frage nach algorithmischen Verbesserungen auf, die durch Adaptivität eine geringere Anzahl von Gleitkommaoperationen benötigen und deshalb eine weitere Verbesserung versprechen. Meilensteine des Antrags sind dementsprechend Grundlagenarbeit zur Entwicklung von sogenannten Patch-adaptiven Verfahren (PAM), eine Untersuchung ihrer praktischen Einsetzbarkeit durch Einbau in praxisrelevanten Code, sowie eine Ergänzung der bereits in DiME entwickelten Bibliothek.

Unabhängig von algorithmischen Verbesserungen wurden bereits in der Anfangsphase des Fortsetzungsprojekts folgende Probleme identifiziert, die es notwendig erschienen ließen, sich in DiME-2 stärker mit der Auswirkung architektureller Rahmenbedingungen auf die Effizienz der entwickelten Optimierungsverfahren zu befassen und weitere Verbesserungen zu erarbeiten:

- Basierend auf den guten Ergebnissen der in DiME entwickelten Optimierungen auf Alpha-Rechnern war davon auszugehen, dass sich diese auch auf anderen Architekturen beobachten lässt. Überraschender Weise ist die erreichbare Verbesserung z. B. auf der Intel-x86-Architektur (Pentium 4) deutlich niedriger als erwartet.
- Für dreidimensionale Modellgitter scheint eine Verbesserung durch Cache-Optimierungen kaum möglich zu sein; architekturelevant ist in diesem Fall der hohe Verschachtelungsgrad von Schleifen und die Notwendigkeit, gleichzeitig auf viele örtlich im Speicher getrennt liegende Bereiche zugreifen zu müssen.

Wegen der hohen Relevanz der Intel-Architektur und der ungeklärten Probleme in 3D war deshalb eine genauere Untersuchung nötig, um die Ergebnisse von DiME auch auf zukünftig relevanten Architekturen vertreten zu können – parallel zur Forschung an PAM-Algorithmen. Ziel war dabei, durch Erarbeitung der zu Grunde liegenden architekturellen Probleme allgemein nutzbare Hinweise für Cache-Optimierungen geben zu können, sowie universell einsetzbare Werkzeuge zu erhalten.

Dabei kann die Untersuchung der Ursachen mit Hilfe von Cache-Simulation einen wesentlichen Vorteil bringen: Hypothesen, die durch reale Beobachtungen nahegelegt werden, können detailliert anhand der Simulation überprüft werden. Es wurde erkannt, dass die in DiME verwendeten Simulatoren nicht ausreichen; diese erfordern

eine manuelle Instrumentierung zur Erzeugung der Speicherzugriffssequenz, woraus a posteriori das Cache-Verhalten einer dabei vorgegebenen Cache-Hierarchie ermittelt wird.

- Eine manuelle Erzeugung der Speicherzugriffssequenz wird sehr schnell zu komplex; außerdem kann sie prinzipbedingt nicht das Verhalten des vom Compiler generierten, tatsächlich abgearbeiteten Codes ermitteln.
- Die zur detaillierten Analyse nötige vollständige Zugriffssequenz des Maschinencodes ist so groß (leicht im dreistelligen Milliardenbereich), dass eine nachträgliche Cache-Simulation unmöglich und eine Online-Verarbeitung nötig wird.
- Eine Erweiterung der simulierten Modelle ist nötig, um der Realität moderner Cache-Architekturen besser zu entsprechen: so ist der Einbezug von automatischen Prefetch-Verfahren der Hardware wichtig.
- Eine Simulation kann weitere, aus den Primärdaten (z. B. Anzahl Cachetrefefer) abgeleitete Metriken berechnen, die genauere Hinweise auf sinnvolle Optimierungen geben.

Zusammen mit den oben erwähnten Beobachtungen bzgl. der Intel-Architektur stellt die im folgenden beschriebene teilweise Neuorientierung der Arbeiten in DiME-2 sicher, dass die Ergebnisse von DiME – also die Notwendigkeit von Cache-Optimierungen für den betrachteten Algorithmus – auch für moderne Architekturen gültig bleiben.

## 3.2 Durchgeführte Arbeiten

Die Arbeiten im Projekt beziehen sich zum einen auf die Entwicklung des Patchadaptiven Glätters, zum anderen auf die Weiterentwicklung der Cache-Optimierungstechniken, deren Voraussetzung die Entwicklung geeigneter Analysewerkzeuge ist. Wie erwartet war die in diesem Projekt gewählte Kooperation zwischen Know-How-Trägern im wissenschaftlichen Rechnen (Erlangen) und in der Rechnerarchitektur (München) von großem Vorteil: Auf Münchener Seite wurden die Werkzeuge zur Simulation moderner Cache-Hierarchien entwickelt, auf Erlanger Seite die Algorithmusverbesserungen erarbeitet und Anwendungen einbezogen. Cache-Optimierungen wurden mit den entstandenen Werkzeugen gemeinsam analysiert, Verbesserungen erarbeitet sowie verifiziert.

### 3.2.1 Algorithmische Verbesserungen

Basierend auf den im Rahmen des vorhergehenden Projekts durchgeführten Arbeiten für ein volladaptives Mehrgitterverfahren auf Punktebene wurde ein Patchadaptiver Glätter für Mehrgitterverfahren entwickelt [KCR05] .

### 3.2.2 Analysewerkzeuge zum Speicherzugriffsverhalten

Basierend auf Vorarbeiten, u. a. im Rahmen von EP-Cache (BMBF-Förderung), entstand ein Werkzeug[WKT04] zur Analyse des Laufzeitverhaltens von Anwendungen. Dieses besteht aus zwei Komponenten: Erstere überwacht den Ablauf eines Programms mit Hilfe von Laufzeitinstrumentierung und kann daraus – ohne dass ein

spezielles Compilieren notwendig wäre – sowohl dynamische Aufrufgraphen als auch Cache-Simulationsdaten erzeugen. Diese werden von der Visualisierungskomponente anschaulich und übersichtlich dargestellt. Die Werkzeugkomponenten „Callgrind“ und „KCachegrind“ wurden – u. a. wegen Rahmenbedingungen aus Vorarbeiten – als Open Source veröffentlicht<sup>1</sup>. Sie werden inzwischen nicht nur in weiteren Open-Source-Projekten wie KDE gewinnbringend eingesetzt, sondern – wie EMail-Kommunikation mit Beratungscharakter zu Unternehmen weltweit zeigen – auch in etlichen Firmen, wie Trolltech und Sun<sup>2</sup>. Die Visualisierung wird auch in dem Werkzeug Adaptor<sup>3</sup>, das in dem oben erwähnten Projekt EP-Cache entstanden ist, am Fraunhofer Institut eingesetzt.

Wichtige Ergänzung des Cache-Simulators aus Sicht des Projekts ist der Einbau einer „Hardware-Prefetch“-Einheit, die das vorausladende Cache-Verhalten z. B. eines Intel Pentium 4 simuliert [WT06b]; dadurch kann quantitativ ermittelt werden, in welcher Weise dieses Architekturmerkmal den ständig abbrechenden, kurzen Datenströmen des optimierten Mehrgittercodes, insbesondere in 3D, entgegenarbeitet. Zudem ließen sich daraus weitere Cache-Optimierungsstrategien wie Block-Prefetching [27] ableiten.

Eine weitere relevante Erweiterung ist die Berechnung von Benutzungsmetriken des Caches [WT05]. Diese können quantitative Aussagen treffen, z. B. welcher Prozentsatz der Daten in den Cache geladen wird, ohne wirklich benutzt zu werden, und dadurch die effektive Bandbreite vermindert. Zudem bekommt man Hinweise, wie oft im Cache liegende Daten benutzt werden. Für die x86-Architektur lässt sich als Optimierungstechnik direkt ableiten, dass man für nur einmal benutzte Daten „Non-Temporal Loads“ einsetzen sollte, die zu weniger Verdrängung und besserer Cache-Nutzung führen.

Beide Verbesserungen sind in der öffentlich verfügbaren Version des Werkzeugs Callgrind enthalten.

### 3.2.3 Weiterentwicklung der Verfahren zur Cache-Optimierung

Zunächst stellte sich die Frage nach den Ursachen für die unerwartet niedrigen Leistungen der in Rahmen von DiME optimierten 3D-Glätter und des 2D-Glätters auf modernen Rechnerarchitekturen wie etwa den aktuellen Intel- verglichen mit den älteren Alpha-Prozessoren. Eine Analyse des optimierten Codes von DiME mit Hilfe des Simulators konnte belegen, dass sich die Anzahl der Hauptspeicherezugriffe gegenüber der unoptimierten Variante tatsächlich halbiert hatte: bei zwei Iterationen in der feinsten Glättungsphase im Mehrgittercode ist eine weitere Verbesserung nicht möglich. Trotzdem konnte der Code die mögliche Rechenleistung der CPU bei weitem nicht erreichen. Eine verringerte Anzahl der Speicherezugriffe allein bewirkt also nicht zwangsläufig eine Verringerung der Laufzeit.

Im Rahmen einer Studienarbeit [Stü05] wurde der Rot-Schwarz-Gauss-Seidel-Glätter direkt in Maschinensprache implementiert. Ziel war es, ein besseres Verständniss für den Instruktionssatz moderner Prozessoren zu bekommen und mögliche Limitierungen durch den Compiler auszuschließen. Hierbei konnte gezeigt werden, dass die entwickelten Optimierungstechniken auch auf modernen Architekturen sehr gute

---

<sup>1</sup>Callgrind ist enthalten in einer Suite von Fehlererkennungs- und Laufzeitanalyse-Werkzeugen für Linux, siehe <http://www.valgrind.org>. Homepage von KCachegrind: <http://kcachegrind.sf.net>

<sup>2</sup>Für eine Beschreibung der Nutzung von Callgrind / KCachegrind für die von Sun mitbetriebenen Entwicklung von OpenOffice, einer Open-Source-Suite von Büroanwendungen, siehe <http://tools.openoffice.org/profiling/callgrind-howto.html>.

<sup>3</sup>T. Brandes. Adaptor. Homepage. <http://www.scai.fraunhofer.de/291.0.html>

Ergebnisse bringen, sofern man die speziellen Anforderungen der jeweiligen Architektur berücksichtigt.

Auch in Bezug auf die Cache-Optimierungen für den 3D-Fall, die bisher zu kaum nennenswerten Verbesserungen geführt hatten, konnte durch eine Implementierung in Maschinensprache eine Beschleunigung um den Faktor 3 erreicht werden. Dies liegt allerdings immer noch deutlich unterhalb der Leistung eines vergleichbar optimierten 2D-Codes. Eine Vermutung war, dass interne Limitierungen der x86-Architekturen eine bessere Leistung für den 3D-Code verhindern. Daher wurde der 3D-Mehrgittercode auf der Intel-IA64-Architektur in Maschinensprache implementiert [Stü06]. Hierbei konnte gezeigt werden, dass auch für den 3D-Fall – eine ausgewogene Architektur vorausgesetzt – eine Leistungssteigerung um den Faktor 5 erreicht werden kann.

Wesentlichen Anteil an den erwähnten Verbesserungen haben Verfahren zum Vorausladen von Daten, die in den Assembler-Varianten geschickt in die Berechnungen eingebaut werden. Motiviert durch die seit kurzem verfügbaren Mehrkernprozessoren mit gemeinsamem Cache wurde als Ergänzung der obigen Arbeiten untersucht, wie gut ein Vorausladen von Daten in einen parallel laufenden, sogenannten „Helper-Thread“, ausgelagert werden kann. Es konnte gezeigt werden, dass auf Intel x86-Prozessoren mit der Core-Mikroarchitektur durch dieses Verfahren eine um bis zu 30 % bessere Bandbreite zum Hauptspeicher und damit eine deutliche Beschleunigung gegenüber einem 2D-Multigrid-Codes in C erreicht werden kann [WT08]. Diese Technik kann also zeit- und kostenintensive Assembleroptimierung ersparen.

### 3.2.4 Cache-Optimierungen und zukünftige Architekturen

Der Aufwand zur Erhaltung von Cache-Koheränz wird bei zukünftigen Vielkern-Architekturen stark zunehmen, so dass nach Alternativen für transparente Zwischenspeicher gesucht wird. Eine bereits eingesetzte Lösung dafür ist es, jedem Kern einen privaten Zwischenspeicher zur Verfügung zu stellen, entweder zusätzlich zu oder anstatt eines Cache-Speichers. Die Cell Broadband Engine z. B. verfügt neben einem klassischen Allzweck-Prozessorkern mit zwei Cache-Ebenen über acht auf Speicherbandbreite und Berechnungen optimierte Prozessorkerne, die auf einem lokalen Speicher arbeiten und die nötigen Operanden mit Hilfe von asynchronen Speichertransfers vom und in den gemeinsamen Hauptspeicher übertragen müssen. Anhand dieser Architektur, deren nächste Prozessorgeneration im Roadrunner-Projekt<sup>4</sup> den ersten Supercomputer mit Peta-FLOP-Leistung ermöglichen soll, wurde untersucht, ob und wie weit sich die entwickelten Optimierungen von transparenten auf lokale Zwischenspeicher übertragen lassen.

### 3.2.5 Praxisrelevante Anwendungen

**LBM** Im Bereich der Anwendung der entwickelten Techniken wurden die im Projekt RU 422/7-3 begonnenen Arbeiten im Bereich der Strömungssimulation mit dem Lattice-Boltzmann-Verfahren konsequent weitergeführt. Die Lattice-Boltzmann-Methode ist ein besonders herausfordernder Fall, da zum einen in 3D bei jedem „Update“ 19 Werte in mehreren Speicherströmen gelesen und geschrieben werden müssen, zum anderen innerhalb des inneren Schleifenkörpers eine große Anzahl von Berechnungen durchgeführt wird. Es wurde gezeigt, dass sich nach Anwendung von Cache-Blocking-Techniken die Leistungsbeschränkung von der Bandbreite des Speichers hin zu einer arithmetischen Limitierung durch Registerabhängigkeiten verschiebt.

<sup>4</sup>Los Alamos National Laboratory: Roadrunner Homepage. <http://www.lanl.gov/roadrunner>

Theoretische Abschätzungen haben gezeigt, dass eine weitere Leistungssteigerung möglich ist, falls es gelingt diese Abhängigkeiten aufzubrechen [THR05].

**HHG** Im Rahmen des durch KONWIHR geförderten „gridlib“-Projekts wurde die Patch-adaptive Gitterbibliothek HHG (Hybrid Hierarchical Grids) implementiert [Ber05]. Bei diesem Ansatz wird ein unstrukturiertes Eingabegitter regelmäßig innerhalb der Elemente verfeinert. Durch dieses Vorgehen wird die Datenlokalität entscheidend verbessert, und es können sehr gute Leistungswerte erreicht werden. Hierbei flossen in enger Zusammenarbeit mit dem DiME-Projekt Techniken ein, die im Rahmen von DiME entwickelt wurden. Es wurde gezeigt, dass die Ein-Prozessor-Leistung auch in massiv-parallelem Kontext von entscheidender Bedeutung ist. Unter Verwendung des HHG-Ansatzes gelang mit  $3.2 \cdot 10^{11}$  Unbekannten in 93 s auf 9170 Prozessorkernen die unseren Wissens nach bisher größte Finite-Elemente-Rechnung [GR07].

**Optischer Fluss** In der Bildregistrierung und Videokompression ist die Lösung des Poisson-Problems in Echtzeit wünschenswert. Erst der Einsatz eines hochoptimierten Mehrgitterlösers macht dies möglich. Eingesetzt werden diese Algorithmen z. B. in der Medizintechnik [SKR07].

### 3.2.6 Erweiterung der Funktionsbibliothek

Durch die neuen Erkenntnisse über die Anwendung der Optimierungstechniken auf modernen Architekturen mussten die elementaren Bausteine, die zur Erstellung einer selbstoptimierenden Bibliothek notwendig sind, neu überdacht werden. Wegen der erweiterten Optimierungsstrategien müsste für die automatischen Architekturanpassung ein deutlich größerer Parameterraum durchsucht werden; dies kann reduziert werden, indem im Vorfeld relevante Kenndaten einer Plattform in Mikrobenchmarks in Erfahrung gebracht werden. Diese wurden implementiert.

Da moderne Architekturen jedoch sensitiv auf die Art der verwendeten Instruktionen reagieren und die Anzahl der relevanten Architekturen überschaubar ist, wurde eine selbstoptimierende Bibliothek zugunsten von plattformspezifischen, handoptimierten Bausteinen verworfen. Dies ist auch dadurch motiviert, dass es aufgrund mangelnder Qualität des Assemblercodes vieler Compiler, die heutzutage eingesetzt werden, oft notwendig ist, den Algorithmus in Maschinensprache zu implementieren, um die gewünschte Leistung zu erreichen.

## 3.3 Darstellung und Diskussion der Ergebnisse

Durch Implementierung von Mikrobenchmarks und des Rot-Schwarz-Gauss-Seidel-Glätters für den Intel Pentium 4 und AMD Athlon 64 konnten die folgenden Regeln abgeleitet werden, um effizienten Code zu erzielen: (1) Nutzung von SIMD-Instruktionen und (2) Nutzung und Ergänzung automatischer Vorauslademechanismen in den Cache.

Die meisten modernen Prozessoren unterstützen Vektoroperationen („Single Instruction Multiple Data“, SIMD), die dieselbe Operation auf mehrere Operanden gleichzeitig anwenden. 16 Byte breite Register werden verwendet, die im Falle doppelt genauer Gleitkommawerte zwei und für einfache Genauigkeit vier Werte enthalten. Wichtig ist, dass bei Nutzung von SIMD-Operationen sehr effizient auf Speicher zugegriffen wird, was sich in einer höheren verfügbaren Bandbreite widerspiegelt.

Daneben lässt sich der Algorithmus in wesentlich weniger Instruktionen formulieren. Obwohl er nur 64Bit breite Rechenwerke besitzt und Berechnungen doppelter Genauigkeit intern sequentiell berechnet werden, ist dies speziell für den Pentium 4 von Bedeutung. Dieser kann nur eine x86-Instruktion pro Takt dekodieren, zudem profitiert der Instruktionsscheduler von der geringeren Anzahl potentieller Abhängigkeiten durch SIMD. Um SIMD-Operationen effizient im Löser zu verwenden, mussten die roten und schwarzen Punkte des Gitters in separaten Feldern abgespeichert werden. Dies hat zudem den Vorteil einer besseren Ausnutzung von Cachezeilen.

Die Zugriffsverhersage-Mechanismen modernen Prozessoren versuchen, voraussichtlich bald benötigte Daten in die Caches vorausladen. Die tatsächlichen Zugriffsmuster sollten also für die eingesetzten Heuristiken erfassbar sein. Daneben existieren spezielle Instruktionen, mit denen Vorausladen von Daten explizit angestoßen werden kann. Dies wird entweder notwendig, wenn das Speicherzugriffsmuster zu komplex ist oder die integrierte Zugriffsvorhersage nicht effizient genug arbeitet. Vorauslade-Instruktionen wurden bei Bedarf in den dann leider notwendigen, zeitaufwendig handoptimierten Assemblercode eingebaut. Allerdings kann bei Mehrkernarchitekturen mit einem gemeinsam genutzten Cache gezieltes Vorausladen auch in einen parallel zur Anwendung laufenden Thread ausgelagert werden, was Assemblercodierung dadurch vermeidet.

Die beschriebenen Techniken können im Prinzip auch von Compilern genutzt werden. In der Praxis ist es für sie aber sehr schwer, die nötigen Informationen aus der Hochsprache zu extrahieren. In vielen Fällen können obige Techniken daher nicht oder nicht ausreichend vom Compiler eingesetzt werden. Dies wiegt umso schwerer, als viele moderne Prozessoren sehr sensitiv darauf reagieren, welche Instruktionen verwendet werden. Es ist festzustellen, dass die klassische RISC-Architektur, in der der Compiler ein integraler Bestandteil des Gesamtentwurfs war, zunehmend komplexen Instruktionssätzen weicht. Um die Leistungsfähigkeit der CPU auszuschöpfen, ist es dort oft unabdingbar, den Algorithmus zumindest in Teilen in Maschinensprache zu implementieren.

Speziell für die 3D-Codes auf der x86-Architektur, die durch die geringe Anzahl an Registern benachteiligt ist, gibt es einen weiteren Grund für die schlechten Ergebnisse: Compiler erzeugen ineffizienten Code für die verschachtelten Schleifenkonstrukte, die durch Blocking-Techniken bedingt sind, und machen so die Vorteile der besseren Cache-Ausnutzung wieder zunichte. Durch direkt in Maschinensprache implementierten Code konnte gezeigt werden, dass prinzipiell auch für den 3D-Fall auf der x86-Architektur effizienter Code erzeugt werden kann. Die Leistung des 2D-Glätters konnte so gegenüber der vorigen optimierten Version nochmals um den Faktor 2 gesteigert werden, was nahezu einer Verbesserung um den Faktor 5 gegenüber der unoptimierten Version entspricht. Für den 3D-Fall wurde gegenüber der unoptimierten Version eine Steigerung um den Faktor 3 erreicht, auch hier können die entwickelten Techniken also eine deutliche Leistungssteigerung bringen. Die geringere absolute Leistung gegenüber dem 2D-Fall liegt an der höheren Anzahl gleichzeitig benutzter Speicherströme, die zu einer geringeren effektiven Hauptspeicherbandbreite führen. Um obige Vermutungen bezüglich des 3D-Mehrgitter-Codes zu untermauern, wurde auf der IA64-Architektur ein optimierter Mehrgitterlöser in Maschinensprache implementiert. Es stellte sich heraus, dass auf einer geeigneten Architektur auch der 3D-Code in vollem Umfang von den Cache-Optimierungen profitiert. Es konnte eine Steigerung der Gleitpunktleistung um den Faktor 5 erreicht werden. Wie auch auf der x86-Architektur zeigte sich der große Einfluss des konkreten Instruktioncodes. Von seiner Qualität hängt es ab, ob der durch Cache-Blocking-Techniken mögliche Geschwindigkeitszuwachs wirklich ausgeschöpft werden kann. Als wei-

teres Ergebnis stellte sich heraus, dass bei geeignetem Blocken der Speicherbus so stark entlastet wird, dass arithmetische Probleme ihm gegenüber als leistungsbeschränkende Faktoren in den Vordergrund rücken [STR08].

Eine besondere Bedeutung kommt allerdings stets dem Speicherlayout zu, dessen Wahl die Geschwindigkeit stark beeinflusst und eine Voraussetzung für weitere Optimierungsverfahren darstellt [DZH<sup>+</sup>05].

Die Experimente mit der Cell-BE [SGRR07, KSFR07] als Vertreter zukünftiger Architekturen konnten zeigen, dass insbesondere Optimierungen des Speicherlayouts und Blocking-Techniken auch hier effizient eingesetzt werden können. Techniken, die durch Nutzung mehrerer Speicherbereiche eine parallele Kommunikation und Berechnung erlauben („Multibuffering“), entkoppeln Hauptspeicherzugriffe mit beschränkter Bandbreite und hoher Latenz einerseits und Berechnungen auf dem schnellen lokalen Speicher andererseits, und erlauben so zumindest bei regelmäßigen Zugriffsmustern die getrennte Optimierung beider Aspekte. Um die Speicherbandbreite möglichst gut zu nutzen, muss das Speicherlayout gleichzeitig benötigte Operanden so speichern, dass sie in wenigen Blöcken und mit Ausnutzung mehrerer Speicherkanäle und dahinterliegender -bänke ermöglichen. Es sollte zudem die Verarbeitung durch SIMD-Operationen begünstigen. Da für die Berechnung alle Operanden mit niedriger Latenz zwischen lokalem Speicher und Registern übertragen werden können, können Art und Scheduling der Instruktionen leichter für optimalen Durchsatz gewählt werden.

### **Entstandene Werkzeuge**

Die im Rahmen der Arbeiten zur Cache-Optimierung entstandenen Werkzeuge zur Analyse des Speicherzugriffsverhaltens können als ein eigenständiges Ergebnis des Projekts betrachtet werden. Zum einen wurde eine robuste Online-Cache-Simulation ermöglicht, die eine einfache Benutzung durch Instrumentierung zur Laufzeit ohne Änderung des Programms erreicht. Dies wird ergänzt durch eine Visualisierungs-umgebung, die eine übersichtliche Darstellung der Messdaten im Aufrufgraph vornimmt. Eine wesentliche Eigenschaft des Simulators ist eine Erweiterung des zugrunde liegenden einfachen Cache-Modells hinsichtlich Strategien zur Zugriffsvorhersage und zum Vorausladen, was zu „Best-Case“-Aussagen führt. In [WT06b] wird gezeigt, dass das erweiterte Modell hohe Realitätsnähe speziell im Hinblick auf betrachtete x86-Prozessoren besitzt. Ein Unterscheidungsmerkmal zu anderen Cache-Simulatoren, die übliche Metriken wie Trefferraten von Cache-Zugriffen ausgeben, ist der Einbezug von neuartigen Cache-Nutzungsmetriken in das Messergebnis: Diese umfassen die Anzahl tatsächlich verwendeter Bytes und die Anzahl der Zugriffe auf eine Cache-Zeile vor ihrer Verdrängung („Wie gut wird der Aufwand des Hauptspeicherzugriffs kompensiert?“) [WT05].

Schließlich entstand im Laufe des Projekts eine Microbenchmark-Plattform zur Charakterisierung der Speicherhierarchie.

### **Vorschläge zu Verbesserungen zukünftiger Prozessor-Architekturen**

Eine *bessere Beeinflussung von Hardware-Heuristiken* zum Vorausladen von Daten durch Programmcode ist sinnvoll, etwa durch die Einführung einer Instruktion zum Vorausladen ganzer Speicherblöcke. Dies verbessert den Befehlsdurchsatz und erreicht eine Überlappung von Speicherzugriff und Berechnungen. Der Nutzen wurde mit Hilfe eines Helper-Threads auf einer Mehrkern-Architektur belegt. Ein anderes Beispiel ist eine Instruktion, die es ermöglicht, dem Prozessor das Ende

eines Ladestroms mitzuteilen. Gerade bei vielen kleinen Schleifen, wie sie durch Cache-Blocking entstehen, wird durch Vorausladen des sequentiellen Zugriffs in jeder Schleife häufig zuviel geladen. Dies mindert die effektive Bandbreite und verdrängt sinnvolle Cache-Inhalte.

Ein Laden in *breitere SIMD-Register* kann den Durchsatz zum Hauptspeicher wesentlich verbessern.

3D-Simulationen benötigen *effiziente Unterstützung mehrerer Datenströme*: Engpässe können hier zwischen allen Speicherebenen entstehen, von internen Caches bis zum DRAM-Speicher.

### 3.4 Wirtschaftliche Verwertbarkeit

Die im Projekt entstandenen Werkzeuge zur Leistungsanalyse und zum Cache-Verhalten sind frei verfügbar. Sie können eingesetzt werden (und werden das auch) bei der Entwicklung von C/C++/Fortran-Code. Ein effektiver Einsatz der Werkzeuge benötigt jedoch Support- und Schulungs- Dienstleistungen. Solche Leistungen könnten gegen Bezahlung angeboten werden.

Der entstandene prototypische Code des Mehrgitterlösers in 2D und 3D kann — bei notwendiger Erweiterung hinsichtlich der Qualitätssicherung und Dokumentation — vermarktet werden.

### 3.5 Kooperationspartner

- *Dr.Ing. Ben Bergen, Los Alamos National Laboratory (LLNL), Los Alamos, NM, USA:*
  - Diskussionen über Performancemetriken
  - Einfließen von DIME Techniken in das HHG Framework
- *Dr.Ing. Markus Kowarschik, Siemens Medical Solutions, Erlangen:*
  - Diskussionen über die weitere Vorgehensweise in DIME
  - Wissenstransfer in der Anwendung bildgebende Verfahren in der Medizintechnik
- *Frank Hülsemann PHD, Electricité de France, Paris, Frankreich:*
  - Diskussionen über numerische Algorithmen (insbesondere Finite Elemente)
  - Wissenstransfer in verschiedene Anwendungen der Simulation
- *EP-Cache Projekt, BMBF gefördertes Projekt mit ähnlicher Zielsetzung wie DIME im Werkzeugbereich, beteiligte Organisationen: :*
  - Zentrum für Hochleistungsrechnen, TU Dresden
  - SCAI, Fraunhofer Gesellschaft St. Augustin
  - Lehrstuhl für Rechnertechnik und Rechnerorganisation, TU München

### 3.6 Nachwuchsförderung

Das Projekt Ru 422/7-5 trug wie folgt zur Förderung des wissenschaftlichen Nachwuchses bei:

- Promotion des Projektmitarbeiters *Markus Kowarschik*,  
Lehrstuhl für Informatik 10 (Systemsimulation), Universität  
Erlangen-Nürnberg,  
abgeschlossen im Juli 2004 [Kow04]
- Promotionsprojekt des Projektmitarbeiters *Jan Treibig*,  
Lehrstuhl für Informatik 10 (Systemsimulation),  
Universität Erlangen-Nürnberg,  
voraussichtlicher Abschluss 2008
- Promotionsprojekt des Projektmitarbeiters *Markus Stürmer*,  
Lehrstuhl für Informatik 10 (Systemsimulation),  
Universität Erlangen-Nürnberg,  
begonnen Juni 2006
- Promotionsprojekt des Projektmitarbeiters *Stefan Donath*,  
Lehrstuhl für Informatik 10 (Systemsimulation),  
Universität Erlangen-Nürnberg,  
begonnen Oktober 2006
- Habilitation des wissenschaftlichen Mitarbeiters *Josef Weidendorfer*, Dr.,  
Lehrstuhl für Informatik X (Rechnerarchitektur), TU München,  
voraussichtlich 2008
- Studentische Arbeiten:
  - *Jens Wilke*: LSS, Studienarbeit, abgeschlossen [Wil03],
  - *Klaus Iglberger*: LSS, Bachelor's Thesis, abgeschlossen [Igl03],
  - *Iris Christadler*: LSS, Studienarbeit, abgeschlossen [Chr03],
  - *Stefan Donath*: LSS, Bachelor's Thesis, abgeschlossen [Don04],
  - *Manfred Hauser*: LRR, Diplomarbeit, abgeschlossen [Hau04],
  - *Stefan Hamerl*: LRR, Diplomarbeit, abgeschlossen [Ham04],
  - *Dietrich Christopheit*: LRR, Diplomarbeit, abgeschlossen [Chr04],
  - *Stefan Lukowitz*: LRR, Diplomarbeit, abgeschlossen [Luk05],
  - *Simon Hausmann*: LSS, Bachelor's Thesis, abgeschlossen [Hau05],
  - *Markus Stürmer*: LSS, Studienarbeit, abgeschlossen [Stü05],
  - *Markus Stürmer*: LSS, Diplomarbeit, abgeschlossen [Stü06],
  - *Johannes Habich*: LSS, Bachelor's Thesis, abgeschlossen [Hab06],
  - *Aditya Nitsure*: LSS, Master's Thesis, abgeschlossen [Nit06],
  - *Vlasia Anagnostopoulou*: LRR, Diplomarbeit, abgeschlossen [Ana06],
  - *Daniel Ritter*: LSS, Master's Thesis, laufend.

## 4 Anhang

Viele Vortragsfolien, Technische Berichte und Preprints können im Internet von der DiME-Projektseite oder der Homepage des Lehrstuhl für Systemsimulation abgerufen werden:

<http://www10.informatik.uni-erlangen.de/Research/Projects/DiME/>  
<http://www10.informatik.uni-erlangen.de/Publications/>

### 4.1 Präsentationen und Vorträge

- [1] DONATH, STEFAN: *Cache-Optimized Implementations of the Lattice Boltzmann Methods*. ASIM05, IMMD10, Erlangen, September 2005.
- [2] GÖTZ, JAN und MARKUS STÜRMER: *Simulation of Blood Flow in the Human Brain Using the Cell Processor*. The Fourth International Conference for Mesoscopic Methods in Engineering and Science (ICMMES), München, Juli 2007.
- [3] HAGER, GEORG, THOMAS ZEISER, JAN TREIBIG und GERHARD WELLEIN: *Optimizing performance on modern HPC systems: learning from simple kernel benchmarks*. The 2nd Russian-German Advanced Research Workshop on Computational Science and High Performance Computing, German-Russian Center for Computational Technologies and High Performance Computing, Stuttgart, März 2005.
- [4] HÄRDTLEIN, JOCHEN: *Blocking Techniques with Fast Expression Templates*. POOSC-Workshop, 19th European Conference on Object-Oriented Programming, University of Glasgow, Glasgow, Juli 2005.
- [5] IGLBERGER, KLAUS et al.: *Cache Optimization for the Lattice Boltzmann Method in 3D*. CSE Conference 2005 SIAM, Orlando, USA, Februar 2005.
- [6] KOWARSCHIK, MARKUS: *Cache Performance Optimizations for Iterative Linear Solvers and the Lattice Boltzmann Method*. TU Wien, Vienna, Austria, Mai 2004.
- [7] KOWARSCHIK, MARKUS, IRIS CHRISTADLER und ULRICH RÜDE: *Patch-Adaptive Relaxation in Multigrid Algorithms*. PARA'04 State-of-the-Art in Scientific Computing, Technical University of Denmark, Lyngby, Copenhagen, Denmark, Juni 2004.
- [8] NITSURE, ADITYA, KLAUS IGLBERGER, ULRICH RÜDE, CHRISTIAN FEICHTINGER, GERHARD WELLEIN und GEORG HAGER: *Optimization of Cache Oblivious Lattice Boltzmann Method in 2D and 3D*. ASIM 2006 - 19th Symposium Simulationstechnique, ASIM, Hannover, September 2006.
- [9] RÜDE, ULRICH: *Cache-Optimizations for Numerical Algorithms*. Dagstuhl-Seminar on Cache-Oblivious and Cache-Aware Algorithms, Juli 2004.
- [10] RÜDE, ULRICH: *Challenges and potentials of emerging multicore architectures*. 3rd Joint HLRB and KONWIHR Result and Reviewing Workshop, LRZ, Garching, Dezember 2007.
- [11] RÜDE, ULRICH, BENJAMIN BERGEN, CHRISTOPH FREUNDL und TOBIAS GRADL: *Massively Parallel Multigrid for Finite Elements*. 13th Copper Mountain Conference on Multigrid Methods, Copper Mountain, USA, März 2007.
- [12] RÜDE, ULRICH, MARKUS KOWARSCHIK, ARNDT BODE und JOSEF WEIDENDORFER: *Performance Prediction, Analysis, and Optimization of Numerical Methods on Cache-Based Computer Architectures*. Half-Day Tutorial at the

- 31st Annual International Symposium on Computer Architecture (ISCA 2004), Munich, Germany.
- [13] RÜDE, ULRICH und JAN TREIBIG: *Optimizing Numerical Simulation Kernels for Current Instruction and Memory Architectures*. ScalPerf05, Bertinoro, Italy, Oktober 2005.
  - [14] RÜDE, ULRICH et al.: *Experiences with the design of cache-aware numerical algorithms*. ScalPerf04, Bertinoro, Italy, September 2004.
  - [15] RÜDE, ULRICH et al.: *Performance-/Cache-Optimierungen in der numerischen Simulation am Beispiel von Lattice-Boltzmann-Verfahren*. ASIM-Workshop der GI, Siegen, Germany, März 2004.
  - [16] STÜRMER, MARKUS: *Fluid Simulation using the Lattice Boltzmann Method on the Cell Processor*. Zentralinstitut für Angewandte Mathematik des Forschungszentrums Jülich, Jülich, April 2007.
  - [17] STÜRMER, MARKUS: *Optimizing Fluid Simulation and other scientific applications on the Cell*. SFB 716, Stuttgart, Juni 2007.
  - [18] STÜRMER, MARKUS, HARALD KÖSTLER und ULRICH RÜDE: *A fast full multigrid solver for applications in image processing*. 13th Copper Mountain Conference on Multigrid Methods, Copper Mountain, USA, März 2007.
  - [19] STÜRMER, MARKUS, JAN TREIBIG und ULRICH RÜDE: *Optimizing a 3D Multigrid for the IA-64 Architecture*. ASIM 2006 - 19th Symposium Simulationstechnique, ASIM, Hannover, September 2006.
  - [20] TREIBIG, JAN et al.: *Performance analysis of the Lattice Boltzmann Method on x86-64 Architectures*. ASIM 2005, Erlangen, Germany, September 2005.
  - [21] TREIBIG, JAN und GEORG HAGER: *Why is the Performance Productivity Poor on Modern Computer Architectures*. Seminar Architectures and Algorithms for Petascale Computing, Dagstuhl, Germany, Februar 2006.
  - [22] WEIDENDORFER, JOSEF: *Simulation by Runtime Instrumentation and Profile Visualization*. Demonstration on Workshop on Automatic Performance Analysis (WAPA 2005), Dagstuhl, Germany, Dezember 2005.
  - [23] WEIDENDORFER, JOSEF: *Single Processor Performance: Analysis with Callgrind / KCachegrind*. Talk at 1st Parallel Tools Workshop, HLRS, Stuttgart, Juli 2007.
  - [24] WEIDENDORFER, JOSEF: *Single Processor Performance: Analysis with Callgrind / KCachegrind*. Talk at Multicore-Training, LRZ, Garching, Juli 2007.
  - [25] WEIDENDORFER, JOSEF: *Understanding Memory Access Bottlenecks on Multicore*. Talk at ParCo 2007, September 2007.
  - [26] WEIDENDORFER, JOSEF, MARKUS KOWARSCHIK und CARSTEN TRINITIS: *A Tool Suite for Simulation Based Analysis of Memory Access Behavior*. International Conference on Computational Science (ICCS 2004), Krakov, Poland, Juni 2004.
  - [27] WEIDENDORFER, JOSEF und CARSTEN TRINITIS: *Cache Optimization for Iterative Numerical Codes Aware of Hardware Prefetching*. International Conference on Applied Parallel Computing (PARA'04), Copenhagen, Denmark, Juni 2004.
  - [28] WELLEIN, GERHARD, THOMAS ZEISER, GEORG HAGER, ADITYA NITSURE, KLAUS IGLBERGER und ULRICH RÜDE: *Introducing a Cache-Oblivious Blocking Approach for the Lattice Boltzmann Method*. ICMMES 06, Hampton (VA), USA, Juli 2006.

## 4.2 Überblick der entstandenen Publikationen

- [Ana06] ANAGNOSTOPOULOU, VLASIA: *Exploiting multi-core processors for memory-bound numerical codes by using prefetching techniques*. Lehrstuhl für Rechnerarchitektur und Rechnerorganisation / Parallelrechnerarchitektur (LRR), Informatik 10, Fakultät für Informatik, Technische Universität München, Germany, Oktober 2006. Diplomarbeit.
- [Ber05] BERGEN, B.: *Hierarchical Hybrid Grids: Data Structures and Core Algorithms for Efficient Finite Element Simulations on Supercomputers*. Doktorarbeit, FAU Erlangen, 2005.
- [BGHR06] BERGEN, B., T. GRADL, F. HÜLSEMANN und U. RÜDE: *A Massively Parallel Multigrid Method for Finite Elements*. Computing in Science and Engineering, 8(6):56–62, Dezember 2006.
- [BHR05] BERGEN, B., F. HÜLSEMANN und U. RÜDE: *Is  $1.7 \times 10^{10}$  Unknowns the Largest Finite Element System that Can Be Solved Today?* In: *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, Seite 5, Washington, DC, USA, 2005. IEEE Computer Society. Preprint version as Technical Report 05-3.
- [Chr03] CHRISTADLER, I.: *Patch-adaptive Relaxation als Glätter im Mehrgitterverfahren*. Lehrstuhl für Informatik 10 (Systemsimulation), Institut für Informatik, University of Erlangen-Nuremberg, Germany, Dezember 2003. Studienarbeit.
- [Chr04] CHRISTOPHEIT, DIETRICH: *Vervollständigung von Profiling-Messungen durch Kombination*. Lehrstuhl für Rechnerarchitektur und Rechnerorganisation / Parallelrechnerarchitektur (LRR), Informatik 10, Fakultät für Informatik, Technische Universität München, Germany, August 2004. Diplomarbeit.
- [Don04] DONATH, S.: *On Optimized Implementations of the Lattice Boltzmann Method on Contemporary Architectures*. Bachelor's Thesis, August 2004.
- [DRHB07] DOUGLAS, C. C., U. RÜDE, J. HU und M. L. BITTENCOURT: *A Guide to Designing Cache Aware Multigrid Algorithms*. Technischer Bericht 07-3, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, April 2007.
- [DZH<sup>+</sup>05] DONATH, S., T. ZEISER, G. HAGER, J. HABICH und G. WELLEIN: *Optimizing Performance of the Lattice Boltzmann Method for Complex Structures on Cache-based Architectures*. In: HÜLSEMANN, F., M. KOWARSCHIK und U. RÜDE (Herausgeber): *18th Symposium Simulationstechnique ASIM 2005 Proceedings*, Band 15 der Reihe *Frontiers in Simulation*, Seiten 728–735. ASIM, SCS Publishing House, September 2005.
- [FGRB07] FREUNDL, C., T. GRADL, U. RÜDE und B. BERGEN: *Petascale Computing: Algorithms and Applications*, Kapitel Towards Petascale Multi-level Finite Element Solvers. Chapman & Hall/CRC, Dezember 2007.
- [Göt06a] GÖTZ, J.: *Numerical Simulation of Blood Flow with Lattice Boltzmann Methods*. Master's Thesis, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, Juli 2006.
- [Göt06b] GÖTZ, J.: *Simulation of bloodflow in aneurysms using the Lattice Boltzmann method and an adapted data structure*. Technischer Bericht 06-6, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, 2006.

- [GR07] GRADL, TOBIAS und ULRICH RÜDE: *Massively Parallel Multilevel Finite Element Solvers on the Altix 4700*. inSiDE, 5(2):24–29, 2007.
- [Hab06] HABICH, J.: *Improving computational efficiency of Lattice Boltzmann methods on complex geometries*. Bachelor's Thesis, Dezember 2006.
- [Ham04] HAMERL, STEFAN: *Entwicklung eines komponentenbasierten Designs zur interaktiven Visualisierung und Steuerung von Profiling-Messungen*. Lehrstuhl für Rechnertechnik und Rechnerorganisation/Parallelrechnerarchitektur (LRR), Informatik 10, Fakultät für Informatik, Technische Universität München, Germany, Juli 2004. Diplomarbeit.
- [Hau04] HAUSER, MANFRED: *Assemblerbasierte Optimierungen für EPIC- und CISC-Architekturen bei iterativen numerischen Codes*. Lehrstuhl für Rechnertechnik und Rechnerorganisation/Parallelrechnerarchitektur (LRR), Informatik 10, Fakultät für Informatik, Technische Universität München, Germany, Juli 2004. Diplomarbeit.
- [Hau05] HAUSMANN, S.: *Optimization and Performance Analysis of the Lattice Boltzmann Method on x86-64 based Architectures*. Bachelor's Thesis, April 2005. Betr. J. Treibig.
- [HLP06] HÄRDTLEIN, J., A. LINKE und C. PFLAUM: *Blocking Techniques with Fast Expression Templates*. Technischer Bericht 06-8, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, November 2006.
- [HZTW05] HAGER, G., T. ZEISER, J. TREIBIG und G. WELLEIN: *Optimizing Performance on Modern HPC Systems: Learning From Simple Kernel Benchmarks*. Conference Paper, März 2005. 2nd Russian-German Advanced Research Workshop on Computational Science and High Performance Computing, German-Russian Center for Computational Technologies and High Performance Computing, Stuttgart, 16.03.2005.
- [Igl03] IGLBERGER, K.: *Cache Optimizations for the Lattice Boltzmann Method in 3D*. Lehrstuhl für Informatik 10 (Systemsimulation), Institut für Informatik, University of Erlangen-Nuremberg, Germany, September 2003. Bachelor thesis.
- [KCR04] KOWARSCHIK, M., I. CHRISTADLER und U. RÜDE: *Towards Cache-Optimized Multigrid Using Patch-Adaptive Relaxation*. Technischer Bericht 04-8, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, 2004.
- [KCR05] KOWARSCHIK, M., I. CHRISTADLER und U. RÜDE: *Towards Cache-Optimized Multigrid Using Patch-Adaptive Relaxation*. In: DONGARRA, J., K. MADSEN und J. WASNIEWSKI (Herausgeber): *PARA 2004 Proceedings*, Band 3732 der Reihe *Lecture Notes in Computer Science*, Seiten 901–910. Springer Verlag, Dezember 2005.
- [Kow04] KOWARSCHIK, M.: *Data Locality Optimizations for Iterative Numerical Algorithms and Cellular Automata on Hierarchical Memory Architectures*. Cauerstrasse 6, 91058 Erlangen, Germany, Juli 2004. SCS Publishing House, ISBN 3-936150-39-7.
- [KSFR07] KÖSTLER, H., M. STÜRMER, C. FREUNDL und U. RÜDE: *PDE based Video Compression in Real Time*. Technischer Bericht 07-11, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, August 2007.

- [Luk05] LUKOWITZ, STEFAN: *Ermittlung des Quellcodebezugs von Speicherzugriffen auf Datenobjekte*. Lehrstuhl für Rechnertechnik und Rechnerorganisation/Parallelrechnerarchitektur (LRR), Informatik 10, Fakultät für Informatik, Technische Universität München, Germany, Mai 2005. Diplomarbeit.
- [NIR<sup>+</sup>06] NITSURE, A., K. IGLBERGER, U. RÜDE, C. FEICHTINGER, G. WELLEIN und G. HAGER: *Optimization of Cache Oblivious Lattice Boltzmann Method in 2D and 3D*. In: BECKER, M. und H. SZCZEBICKA (Herausgeber): *Simulationstechnique - 19th Symposium in Hannover, September 2006*, Band 16 der Reihe *Frontiers in Simulation*, Seiten 265–270. ASIM, SCS Publishing House, September 2006.
- [Nit06] NITSURE, A.: *Implementation and optimization of a cache-oblivious Lattice Boltzmann algorithm*. Master's Thesis, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, August 2006.
- [PKW<sup>+</sup>03a] POHL, T., M. KOWARSCHIK, J. WILKE, K. IGLBERGER und U. RÜDE: *Optimization and Profiling of the Cache Performance of Parallel Lattice Boltzmann Codes*. *Parallel Processing Letters*, 13(4):549–560, 2003.
- [PKW<sup>+</sup>03b] POHL, T., M. KOWARSCHIK, J. WILKE, K. IGLBERGER und U. RÜDE: *Optimization and Profiling of the Cache Performance of Parallel Lattice Boltzmann Codes in 2D and 3D*. Technischer Bericht 03–8, Lehrstuhl für Informatik 10 (Systemsimulation), University of Erlangen-Nuremberg, Germany, Juli 2003.
- [PTD<sup>+</sup>04] POHL, T., N. THÜREY, F. DESERNO, U. RÜDE, P. LAMMERS, G. WELLEIN und T. ZEISER: *Performance Evaluation of Parallel Large-Scale Lattice Boltzmann Applications on Three Supercomputing Architectures*. November 2004. Supercomputing Conference 04.
- [SGRR07] STÜRMER, M., J. GÖTZ, G. RICHTER und U. RÜDE: *Blood Flow Simulation on the Cell Broadband Engine using the Lattice Boltzmann Method*. Technischer Bericht 07-9, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, September 2007.
- [SKR07] STÜRMER, M., H. KÖSTLER und U. RÜDE: *A fast multigrid solver for applications in image processing*. Technischer Bericht 07-6, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, Mai 2007. Accepted for publication in *Numerical Linear Algebra with Applications*.
- [STR06] STÜRMER, M., J. TREIBIG und U. RÜDE: *Optimizing a 3D Multigrid Algorithm for the IA-64 Architecture*. In: BECKER, M. und H. SZCZEBICKA (Herausgeber): *Simulationstechnique - 19th Symposium in Hannover, September 2006*, Band 16 der Reihe *Frontiers in Simulation*, Seiten 271–276. ASIM, SCS Publishing House, September 2006.
- [STR08] STÜRMER, M., J. TREIBIG und U. RÜDE: *Optimizing a 3D Multigrid Algorithm for the IA-64 Architecture*. *International Journal of Computational Science and Engineering (IJCSE)*, 2008. Accepted for publication.
- [Stü05] STÜRMER, M.: *Optimierung des Red-Black-Gauss-Seidel-Verfahrens auf ausgewählten x86-Prozessoren*. Studienarbeit, August 2005. Betr. J. Treibig.

- [Stü06] STÜRMER, M.: *Optimierung von Mehrgitteralgorithmen auf der IA-64 Rechnerarchitektur*. Diplomarbeit, Mai 2006. Betr. J. Treibig.
- [THR05] TREIBIG, J., S. HAUSMANN und U. RÜDE: *Performance Analysis of the Lattice Boltzmann Method on x-86-64 Architectures*. In: HÜLSEMANN, F., M. KOWARSCHIK und U. RÜDE (Herausgeber): *18th Symposium Simulationstechnique ASIM 2005 Proceedings*, Band 15 der Reihe *Frontiers in Simulation*, Seiten 736–741. ASIM, SCS Publishing House, September 2005.
- [Wil03] WILKE, J.: *Cache Optimizations for the Lattice Boltzmann Method in 2D*. Lehrstuhl für Informatik 10 (Systemsimulation), Institut für Informatik, University of Erlangen-Nuremberg, Germany, Februar 2003. Studienarbeit.
- [WKT04] WEIDENDORFER, J., M. KOWARSCHIK und C. TRINITIS: *A Tool Suite for Simulation Based Analysis of Memory Access Behavior*. In: *Proc. of the 2004 Int. Conf. on Computational Science (ICCS2004), Part III*, Band 3038 der Reihe *Lecture Notes in Computer Science (LNCS)*, Seiten 440–447, Krakov, Poland, Juni 2004. Springer.
- [WPKR03a] WILKE, J., T. POHL, M. KOWARSCHIK und U. RÜDE: *Cache Performance Optimizations for Parallel Lattice Boltzmann Codes*. In: *Proc. of the EuroPar-03 Conf.*, Band 2790 der Reihe *Lecture Notes in Computer Science (LNCS)*, Seiten 441–450. Springer, 2003.
- [WPKR03b] WILKE, J., T. POHL, M. KOWARSCHIK und U. RÜDE: *Cache Performance Optimizations for Parallel Lattice Boltzmann Codes in 2D*. Technischer Bericht 03–3, Lehrstuhl für Informatik 10 (Systemsimulation), University of Erlangen-Nuremberg, Germany, Juli 2003.
- [WT05] WEIDENDORFER, JOSEF und CARSTEN TRINITIS: *Collecting and Exploiting Cache-Reuse Metrics*. In: *ICCS 2005: 5th International Conference on Computational Science*, Band 3515 der Reihe *LNCS*, Seiten 191–198. Springer, Mai 2005.
- [WT06a] WEIDENDORFER, JOSEF und CARSTEN TRINITIS: *Block Prefetching for Numerical Codes*. In: *Proc. of the ASIM-06 Conf.*, *Frontiers in Simulation*. SCS, 2006.
- [WT06b] WEIDENDORFER, JOSEF und CARSTEN TRINITIS: *Cache Optimizations for Iterative Numerical Codes Aware of Hardware Prefetching*. Band 3732 der Reihe *Lecture Notes in Computer Science*, Seiten 921–927. Springer, 2006.
- [WT08] WEIDENDORFER, JOSEF und CARSTEN TRINITIS: *Off-loading Application controlled Data Prefetching in Numerical Codes for Multi-Core Processors*. *Int. J. High Performance Computing and Networking*, 2008. Accepted for publication.
- [WZHD06] WELLEIN, G., T. ZEISER, G. HAGER und S. DONATH: *On the single processor performance of simple Lattice Boltzmann kernels*. *computers & fluids*, 35(8–9):910–919, November 2006. ISSN 0045-7930.