Günther Bengel | Christian Baun |
Marcel Kunze | Karl-Uwe Stucky

# Masterkurs Parallele und Verteilte Systeme

Grundlagen und Programmierung von Multicore-prozessoren, Multiprozessoren, Cluster und Grid

► Mit Online-Service

STUDIUM

VIEWEG+
TEUBNER

Günther Bengel | Christian Baun | Marcel Kunze | Karl-Uwe Stucky

Masterkurs Parallele und Verteilte Systeme

Günther Bengel | Christian Baun |
Marcel Kunze | Karl-Uwe Stucky

# Masterkurs Parallele und Verteilte Systeme

Grundlagen und Programmierung von Multicore-prozessoren, Multiprozessoren, Cluster und Grid

Mit 103 Abbildungen

STUDIUM

VIEWEG+
TEUBNER

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
<http://dnb.d-nb.de> abrufbar.



Höchste inhaltliche und technische Qualität unserer Produkte ist unser Ziel. Bei der Produktion und Auslieferung unserer Bücher wollen wir die Umwelt schonen: Dieses Buch ist auf säurefreiem und chlorfrei gebleichtem Papier gedruckt. Die Einschweißfolie besteht aus Polyäthylen und damit aus organischen Grundstoffen, die weder bei der Herstellung noch bei der Verbrennung Schadstoffe freisetzen.

1. Auflage 2008



Lektorat: Sybille Thelen | Andrea Broßler

Vieweg+Teubner ist Teil der Fachverlagsgruppe Springer Science+Business Media.
www.viewegteubner.de




Umschlaggestaltung: KünkelLopka Medienentwicklung, Heidelberg
Druck und buchbinderische Verarbeitung: Wilhelm & Adam, Heußenstamm
Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier.
Printed in Germany

ISBN 978-3-8348-0394-8

# Vorwort

Die Entwicklung von Computern steht heute an einem Wendepunkt. Nach Jahrzehnten stetiger Steigerung der Rechengeschwindigkeit baut heute kein Hardware-Hersteller mehr schnellere sequentielle Prozessoren. Ein klarer Trend zu Computer-Architekturen, die parallele Abläufe unterstützen, und zur Parallelisierung von Programmen ist erkennbar. So sind beispielsweise Multicore-Chips zu erwarten, die bis 2009 bis zu 64 und bis 2015 bis zu 128 integrierte Prozessoren aufweisen. Zusammen mit der Weiterentwicklung von Cluster-Architekturen in homogener und heterogener Rechnerlandschaft sowie mit dem rasch voranschreitenden Ausbau von Grids mit heterogener Zusammensetzung ist hier eine klare Richtung vorgegeben.

Die neuen Architekturen können aber nur dann sinnvoll genutzt werden, wenn die Software den angebotenen Parallelismus auch nutzt, wobei heute noch die Hardware die Entwicklungsgeschwindigkeit vorgibt und der Softwareentwicklung vorauseilt. Parallele Architekturen und deren Programmierung verlassen damit ihre bisherige Nische des Hochleistungsrechnens und werden zukünftig zum Standard. Sie erweitern unsere vernetzte Welt und bieten etwa Wissenschaftlern Zugriff auf nahezu unbegrenzte Rechenleistung und Speicherkapazität. Im kommerziellen Bereich, um mit IBM's Zauberformel "Business on demand" zu argumentieren, wird IT-Dienstleistung an jedem Ort und zu jeder Zeit mit beliebig großen Rechen- und Speicheranforderungen verfügbar. Kaum ein Bereich, in dem heute schon Rechner eingesetzt werden, wird von der allgegenwärtigen Vielfalt an Rechenressourcen ausgenommen bleiben.

Derzeit bahnt sich High Performance Computing (HPC) rasch einen Weg über die Grenzen von Hochschulen und Forschungseinrichtungen hinaus. Die neuen, schlüsselfertigen HPC-Systeme ermöglichen nun auch (fast) jeder Forschungseinrichtung und jedem Unternehmen den Betrieb von enorm leistungsstarken Rechnersystemen. Diese besitzen eine offene Architektur, die sich von einzelnen Racks auf Cluster im Petascale-Bereich ausdehnen lässt. In Grids erfolgt der Rechnerverbund sogar domänenübergreifend, also mit Ressourcen, die verschiedenen Organisationen zugeordnet sind. Ein weltweites Supercomputing in einer zuvor nie gekannten Größenordnung wird Realität.

Peer-to-Peer-Computing (P2P) spielt im Bereich des Hochleistungsrechnens und der Leistungssteigerung durch Parallelität kaum eine Rolle und hat nur wenig Einfluss auf das Cluster- und Grid-Computing. Der ursprüngliche Plan, das P2P-Computing mit in dieses Werk aufzunehmen, wurde aus diesem Grund und zu Gunsten einer größeren Tiefe des übrigen Stoffes aufgegeben.

Der Aufbau des Buches orientiert sich nach einer Einleitung mit Historie und einem allgemeinen Überblick zunächst an der führenden Rolle der Hardwareentwicklung, die der Software-Entwicklung in der Regel immer vorauseilt.

Schon 1987 hat Greg Papadopoulos, heute Chief Technology Officer und Executive Vice President of Research and Development bei Sun Microsystems, Inc., das Hinterherhinken der Software gegenüber der parallelen Hardware folgendermaßen charakterisiert:

„It appears to be easier to build parallel machines than to use them."[1]

Und Sutter und Larus äußern sich folgendermaßen:

„The concurrency revolution is primarily a software revolution. The difficult problem is not building multicore hardware, but programming it in a way that lets mainstream application benefit from the continued exponential growth in CPU performance." [2]

Kapitel 2 beschreibt zunächst die Grundlagen der parallelen Hardware für Einprozessorsysteme und die Rechnerarchitekturen für den Aufbau von Multiprozessoren. Wir starten mit dem Instruction Level Parallelismus und Thread-Level Parallelismus und führen hin zum Simultaneous Multithreading. Bei Multiprozessoren unterscheiden wir zwischen Architekturen mit gemeinsamem Speicher (eng gekoppelten Multiprozessoren) und verteiltem Speicher (lose gekoppelten Multiprozessoren).

Bei den eng gekoppelten Multiprozessoren betrachten wir die Cachekohärenzprotokolle, die Architektur und die Thread-Programmierung von Multicoreprozessoren. Anschließend gehen wir auf die Organisation von Multiprozessorbetriebssystemen und hauptsächlich auf das

---

[1] Papadopoulos G.: The new dataflow architecture being built at MIT. In: Proceedings of the MIT-ZTI-Symposium on Very High Parallel Architectures, November 1987.

[2] Sutter H., Larus J.: Software and the concurreny revolution. ACM Queue, Vol. 3, No. 7, 2005.

Symmetrische Multiprocessing ein. Schwergewicht bei den Multiprozessorbetriebssystemen sind die parallelen Prozesse und deren Synchronisation. Die Synchronisationsverfahren umfassen die hardwarenahen Locksynchronisationsverfahren bis hin zu den klassischen Semaphoren, aber auch das neuere Verfahren des Transactional Memory.

Bei den lose gekoppelten Multiprozessoren zeigen wir, nach der Darstellung von deren Architektur, wie durch die Implementierung eines verteilten gemeinsamen Speichers die lose gekoppelte Architektur in die eng gekoppelte Architektur überführbar ist. Als Beispiel für ein lose gekoppeltes System dient das Load Balancing und High Throughput Cluster Google.

Gemäß der zuvor gemachten Aussage, dass die Software der parallelen Hardware hinterherhinkt und bei der Software ein Nachholbedarf besteht, sind die Programmiermodelle für parallele Architekturen von zentraler Bedeutung und nehmen mit Kapitel 3 den größten Umfang des Werkes ein. Die Unterteilung von Kapitel 2 in eng gekoppelte und lose gekoppelte Multiprozessoren gibt die Unterteilung der Programmiermodelle in Kapitel 3 vor. Der erste Teil befasst sich mit dem Client-Server-Modell, das auf die Hardwarearchitektur keine Rücksicht zu nehmen braucht. Eine Einführung in service-orientierte Architekturen, die hauptsächlich auf verteilten Rechnern basieren, enthält der zweite Teil. Der dritte Teil behandelt die Programmiermodelle für gemeinsamen Speicher und der vierte Teil die Modelle und Programmierverfahren für verteilten Speicher. Es wurde versucht, nicht nur die beiden vorherrschenden Modelle OpenMP für gemeinsamen Speicher und das Message Passing Interface (MPI) für verteilten Speicher zu besprechen, sondern auch die älteren Verfahren und ganz neue Entwicklungen zu behandeln, die gerade im Entstehen und in der Entwicklung sind.

Ältere Programmiermodelle für gemeinsamen Speicher sind Unix mit den fork- und join-Systemaufrufen, Threads und das Ada-Rendezvous. Neuere Modelle sind in Programmiersprachen wie Unified Parallel C und Fortress realisiert.

Ältere Programmiermodelle für verteilten Speicher sind bei den nebenläufigen Modellen Occam und der Parallel Virtual Machine (PVM) zu finden. Weitere ältere kooperative Modelle sind die TCP/IP-Sockets. Nicht ganz so alt sind der Java Message Service (JMS) und für die entfernten Aufrufe der Remote Procedure Call (RPC), die Common Object Request Broker Architecture (CORBA) und die Remote Method Invocation (RMI). Neuere Entwicklungen sind das .NET-Remoting und die

Service Oriented Architecture (SOA) und deren Implementierungsbasis, die Web-Services und der XML-RPC.

Zur Illustration der Programmierverfahren wurde, wo es vom Umfang her möglich und für das Programmiermodell angepasst war, das Erzeuger-Verbraucher-Problem gewählt.

Kapitel 4 beschreibt den parallelen Softwareentwurf und definiert die Leistungsmaße und Metriken für Parallele Programme. Die eingeführten Leistungsmaße führen zu einer Bewertung der nachfolgend besprochenen Parallelisierungstechniken und -verfahren.

Das Werk legt den Schwerpunkt auf die Darstellung der Parallelität und der parallelen Prozesse. Dass die parallelen Prozesse weltweit auf die Rechner verteilt werden ist dabei nur ein Nebenaspekt. Deshalb erläutert Kapitel 5 (Verteilte Algorithmen) nur die mit den verteilten Algorithmen auftretende Problematik des Fehlens von Gemeinsamkeiten. Zur Lösung oder Umgehung dieser Problematik werden die grundlegenden und somit wichtigsten verteilten Basisalgorithmen vorgestellt.

Besonderes Gewicht legen wir mit Kapitel 6 auf das Thema Rechenlastverteilung. Die Beschreibung der statischen Lastverteilung erläutert das Scheduling-Problem, gibt einen Überblick über verschiedene Jobmodelle einschließlich Workflows und diskutiert Beispiele für Verfahren. Der Abschnitt zur dynamischen Lastverteilung unterscheidet zwischen zentralen und dezentralen Verfahren und erläutert die Migration, die Unterbrechung und Verschiebung bereits laufender Prozesse. Den Abschluss bildet eine Einführung in das Grid Scheduling, das auf Besonderheiten der domänenübergreifenden Architektur Rücksicht nehmen muss und für das erste Lösungen verfügbar sind.

Kapitel 7 geht auf Virtualisierungstechniken ein, mit denen das Problem des Ressourcenmanagements in verteilten Systemen elegant gelöst werden kann. Oftmals werden Ressourcen wie CPU und Speicher nicht optimal genutzt, und die Virtualisierung bietet hier ein großes Potenzial zur Effizienzsteigerung. Alle modernen Prozessoren bieten heute entsprechende Funktionen. Anwendungsvirtualisierung hilft darüber hinaus bei der Verwaltung von Software und bei der aus Kostengründen immer häufiger diskutierten Rezentralisierung von IT-Services.

Kapitel 8 beschreibt die Entwicklung des Cluster-Computing. Besonderes Gewicht hat die Klassifikation der unterschiedlichen Arten von Clustern mit ihren typischen Einsatzgebieten, sowie die Beschreibung der eingesetzten Technologien.

Kapitel 9 definiert den Begriff des Grid-Computing und klassifiziert die Unterscheidungsmöglichkeit von verschiedenen Grid-Systemen. Die populärsten Grid Middleware-Systeme mit ihren notwendigen Protokollen und Diensten werden vorgestellt. Zusätzlich beschreibt Kapitel 9 die Grid-Softwarepakete, welche die Verwaltung eines Grid und die Arbeit damit vereinfachen.

Von Kapitel 1 bis 6.1 ist Prof. Bengel der Autor, Abschnitt 6.1.1.3 bis zum Ende von Kapitel 6 verfasste Dr. Stucky, Kapitel 7 hat sich Dr. Kunze vorgenommen, Abschnitt 8 hat C. Baun erstellt und Kapitel 9 wurde in Zusammenarbeit von C. Baun und M. Kunze erstellt.

Der Stoff wurde so umfassend wie möglich dargestellt. Dies betrifft besonders die parallelen Programmiermodelle in Kapitel 3, dem vom Umfang her mächtigsten Abschnitt des Werkes. Dadurch eignet sich das Buch sehr gut als Einstiegs- und Nachschlagewerk. Die tiefe Untergliederung der einzelnen Abschnitte und die systematische Darstellung des Stoffes unterstützen dies. Die Vielzahl von Literaturhinweisen erleichtert dem Leser den noch tieferen Einstieg in die Thematik und die selbstständige Vertiefung des Stoffes. Dadurch ist das Werk auch sehr gut zum Selbststudium geeignet.

Das Buch ist eher forschungsorientiert ausgelegt, und die einzelnen Abschnitte sind in sich abgeschlossen. Durch das Umfassende und den großen Umfang des Werkes konnte in den einzelnen Abschnitten eine große Tiefe erreicht werden. Dadurch lassen sich prinzipiell wie aus einem Modulkasten auch mehrere Masterkurse mit verschiedener Ausrichtung auf dem Gebiet der Parallelen und Verteilten Systeme zusammenstellen und konzipieren. Einzelne Abschnitte oder Teile davon können aber auch in Vorlesungen oder Seminare im Bachelor-Studiengang einfließen.

Vom Forschungszentrum Karlsruhe vom Institut für wissenschaftliches Rechnen danken wir Frau Dr. Jie Tao für die kritische Durchsicht der Hardwarerealisierung von Client-Server-Systemen und die Verbesserung und Richtigstellung des MESI-Protokolls.

Hr. Dipl.-Phys. Klaus-Peter Mickel (komm. Leiter des Instituts für Wissenschaftliches Rechnen) danken wir für die Bereitstellung von Ressourcen und Unterstützung während der Erstellung des Werkes.

Herr Prof. Dr. Georg Winterstein, Dekan der Fakultät für Informatik, Hochschule Mannheim und dem Rektor Prof. Dr. Dietmar v. Hoyningen-Huene dankt Prof. Bengel für die Genehmigung eines Forschungsfreisemesters im Sommersemester 2007 am Forschungszentrum Karls-

ruhe. Ohne dieses Forschungssemester und der Unterstützung durch Herrn Klaus-Peter Mickel wäre dieses Werk in solchem Umfang und Tiefe nicht möglich gewesen.

Vielen Dank an Anja Langner, die das Zeichnen einiger Abbildungen in diesem Buch übernommen hat und den Abbildungen ein professionelleres Aussehen gegeben hat.

Frau Dipl.-Bibl. Maria Klein von der Hochschulbibliothek der Hochschule Mannheim möchten wir unseren Dank aussprechen für die schnelle Beschaffung der aktuellsten Neuerscheinungen, sowie der für dieses Werk notwendigen großen Anzahl von Literatur.

Mit dem Buch steht auch ein kostenloser Online-Service zur Verfügung. Die Internet-Adresse der Web-Seiten ist

```
http://www.pvs.hs-mannheim.de
```

Die folgenden Informationen können auf den Web-Seiten gefunden werden:

- Informationen über die Autoren mit Email-Adresse, die zum Senden von Anmerkungen, Kommentaren und Berichtigungen verwendet werden kann.

- Alle Abbildungen des Buches zum Herunterladen; sie lassen sich in der Lehre einsetzen und wieder verwenden.

- Alle Programmbeispiele des Buches zum Herunterladen. Sie sollen den Leser ermuntern, die Programme auszuprobieren, und dienen zur Gewinnung von praktischer Erfahrung mit den Techniken der parallelen und verteilten Programmierung.

- Ein Erratum, d.h. Korrekturen zu Fehlern, die erst nach der Drucklegung des Buches gefunden wurden.

- Aktuelle Informationen zu Weiter- und Neuentwicklungen bzgl. der im Buch beschriebenen Technologien.

Die Web-Seiten werden kontinuierlich weiterentwickelt und ausgebaut.

Zum Schluss noch eine Zukunftsvision: In einer total vernetzten Welt sind Rechenleistung, Speicherkapazität und andere Ressourcen als Dienste von jedem Computer aus zugreifbar. Der Einsatz von Rech-

nern in täglich genutzten Geräten sowie die mobile Verfügbarkeit von Internetzugängen ermöglichen sogar den Zugriff von buchstäblich jedem beliebigen Ort aus[3].

Durch diese Technologien erhält der Mensch eine nahezu unbegrenzte Vielfalt von Möglichkeiten, sein Leben, seine Arbeit und Freizeit sowie seine Umgebung zu gestalten. Gleichzeitig sind sie aber auch eine Herausforderung, da sie in völlig neuer Art und Weise und in bisher unbekanntem Umfang in das Leben jedes Einzelnen eingreifen. Wir hoffen, dieses Werk hilft Ihnen bei der aktiven und verantwortungsvollen Mitgestaltung dieser Zukunftsvision.

Altrip, Mannheim, Karlsruhe im Dezember 2007

Günther Bengel
Christian Baun
Marcel Kunze
Karl-Uwe Stucky

[3] Siehe hierzu auch Mattern F. (Hrsg.): Total vernetzt. Szenarien einer informatisierten Welt. Springer Verlag 2003.

# Inhaltsverzeichnis