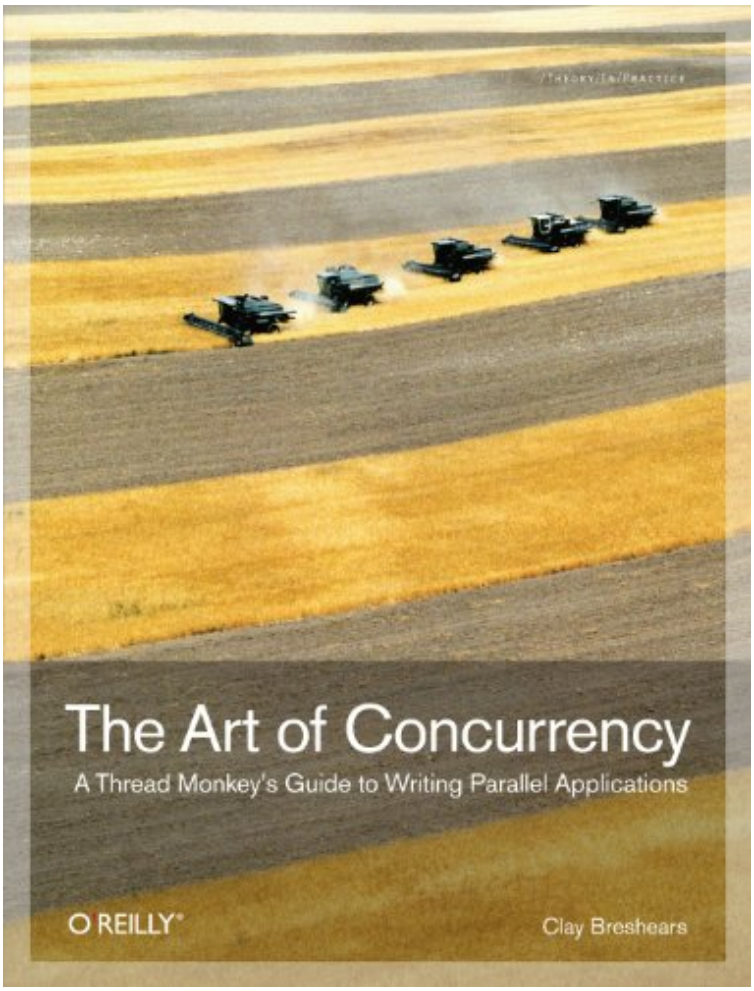
The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications

Von Clay Breshears
**Download PDF | ePub | DOC | audiobook | ebooks*



Produktinformation -Verkaufsrang: #1123074 in eBooksVerffentlicht am: 2009-05-07Erscheinungsdatum: 2009-05-07File Name: B002L4EXD8 | File size: 37.Mb

Von Clay Breshears : The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications before purchasing it in order to gage whether or not it would be worth my time, and all praised The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications:

KundenrezensionenHilfreichste Kundenrezensionen17 von 18 Kunden fanden die folgende Rezension hilfreich. Von der Kunst der Gleichzeitigkeit - Clay Breshears Einfhrung in thread-basierte ProgrammierungVon Janne SchulzEinleitungEs ist noch gar noch so lange her, da wurde man als Programmierer von seinen Freunden und Kollegen ein wenig schief angeschaut, wenn man in vertrauter Runde auf die Frage, an was man denn gerade arbeite, geantwortet hat, das man eine Thread-basierte Software entwickle. Viele der Entwickler rmpften zumindest innerlich mit der Nase - lautete doch der neuste Hype in Sachen paralleler Programmierung und Ereignisverarbeitung "event-

based" und nicht (mehr) "thread-based" [1].Nicht so bei dem Autor Clay Breshears: er hat ein Buch ber die Kunst der parallelen Programmierung (genauer: ber die "Kunst der Gleichzeitigkeit") geschrieben, in dem er Threads pragmatisch und vollkommen undogmatisch verwendet. Das mag auch damit zusammenhngen, das die modernen Prozessoren durch die Multikern-Technologie zu echter Parallelitt fhig sind (soweit dies von dem Betriebssystem und/oder der Programmiersprache untersttzt wird).BeschreibungDas Buch beginnt klassisch und wie bei O'Reilly gewohnt: mit einem Kapitel, indem sich der Autor ber die anvisierte Zielgruppe Gedanken macht. In diesem Fall richtet sich Clay Breshears an Studenten, professionelle Softwareentwickler und ambitionierte Programmierer, die vor numerischen Algorithmen und theoretischen berlegungen nicht zurckschrecken. Er fhrt einige Grundbegriffe der parallelen Programmierung ein, die im weiteren Verlauf des Buches immer wieder auftauchen werden. Er stellt dabei zunchst ganz grundstzliche Fragen in den Raum und nhert sich so dem "Paradigma" der parallelen Programmierung.(Keywords: shared-memory, distributed-memory, dividing work, communication, design, locksSeiten: 18)Im zweiten Kapitel steht die Frage nach "Was ist denn parallel und was nicht?" im Vordergrund. Er untersucht unterschiedliche Szenarien und versucht diese in parallele Handlungsstrnge zu dekomponieren. Wo dies nicht gelingt, erbringt er einen Hinweis auf die Nicht-Parallelisierung des Problems.(Keywords: task decomposition, granularity, boss-worker, scheduling, data decompositionSeiten: 21)Das dritte Kapitel steht ganz im Zeichen von Performanz und Verifikation. Er verdeutlicht die Problematik anhand von Algorithmen, die sich nicht korrekt verhalten und leitet daraus Richtlinien ab, mit denen man einige der kritischen Bereiche umgehen kann.Neben der korrekten Funktionsweise der Programme untersucht er die Performanz der Algorithmen und interessiert sich dabei besonders fr die Verbesserung der Laufzeit (Speedup).(Keywords: critical section, speedup, conditionsSeiten: 22)Im vierten Kapitel fasst er die vorherigen Erkenntnisse und Gedanken zusammen und prsentiert acht Grundregeln fr ein gutes Design von multithreaded Applikationen.(Seiten: 7)Kapitel fnf ist vier Thread-Bibliotheken oder -umgebungen gewidmet, mit denen er in den folgenden Kapiteln die Algorithmen (in verkrzter Form) beschreiben wird. Er unterscheidet zwischen implizitem Threading mit den Vertretern "OpenMP" und "Intel TBB (Thread Building Blocks)" sowie explizitem Threading mit POSIX Threads und Windows Threads by Microsoft.(Keywords: implicit threading, explicit threading, librariesSeiten: 11)Mit dem sechsten Kapitel beginnt der algorithmische und numerische Teil des Buches: die ersten fnf Kapitel dienten der Einfhrung in die Welt der parallelen Programmierung, nun stellt er konkrete Algorithmen und Probleme vor, whlt dazu eine der Thread-Libraries aus und entwickelt Lsungen fr das Problem. Jeder wichtige Algorithmus wird anschlieend nach vier Kriterien beurteilt. Die Kriterien lauten:1. Effizienz2. Einfachheit ('simplicity')3. Portabilitt und4- SkalierbarkeitEr beginnt mit parallelen Summen und Prfix-Suchen.(Keywords: parallel sum, PRAM algorithm, medianSeiten: 28)Das siebte Kapitel gilt "MapReduce" - einem Framework ganz hlich wie "divide-and-conquer" o.. Zunchst untersucht er das "Mapping" der Daten (und deren Dekomposition) um sich danach auf das "Reduzieren" zu konzentrieren. Im Verlauf des Kapitels wird ein weiterer Mechanismus zur Synchronisierung von parallelen Prozessen vorgeschlagen: das "Barrier Object".(Keywords: map, reduce, google, barrier object, key-value,Seiten: 17)Kapitel acht ist ausschlielich der Sortierung gewidmet. Er stellt mindestens fnf Sortierverfahren vor, samt naiver Implementierung, Verbesserungen und Beurteilung nach dem vier-Kritierien Schema (siehe Kapitel sechs).(Keywords: bubblesort, odd-even transposition, shellsort, quicksort, radix-sort,Seiten: 37)Das Thema des neunten Kapitels ist "Suchen". Er unterscheidet zwischen unsortierten und bereits geordneten Daten und untersucht darauf jeweils eine Such-Strategie.(Keywords: binary search, sorted, unsorted,Seiten: 9)Im zehnten und vorletzten Kapitel werden "Graphen" und fr diesen Bereich typische Fragestellungen ("minimal spanning tree" etc.) behandelt.(Keywords: depth-first, shortest path, rubik's cube, breadth-first, floyd, kruskal, primSeiten: 24)Das letzte Kapitel ist eine Vorstellung von Tools, die bei der Entwicklung und Programmierung von parallelen Applikationen helfen sollen.(Keywords: gdb, dbx, debugger, hot-spots,Seiten: 6)MeinungDer Autor hat einen angenehm zu lesenden, professionellen und przisen Schreibstil. Er vermittelt seine langjhrige Erfahrung glaubhaft, ohne jedoch berheblich oder abgehoben zu wirken. Das Buch im Ganzen und jedes Kapitel im Einzelnen ist klar gegliedert und wirkt in seiner Struktur aufgerumt und bersichtlich. Die Probleme sind gut beschrieben, und dort wo ntig, illustriert. Grafiken sind sparsam aber effektiv eingesetzt.Clay Breshears fhrt kompetent in die Thematik ein und wird ihrer Komplexitt in dem Rahmen, den er sich fr das Buch gesetzt hat, gerecht. Leider sind viele Beispiele zu "akademisch", d.h. er arbeitet mit Datenstrukturen, die recht klein dimensioniert (aber damit auch anschaulicher) sind. So ist es nicht so nicht ganz einfach, davon auf reale Probleme zu schlieen, um ein Gefhlfr die Verbesserungen durch parallele Algorithmen zu bekommen. Auch fehlen dem Buch ein paar Benchmarks, die er in einem kleine Testszenario ermittelt hat und deren Ergebnisse er kommentiert.Obwohl er bei Intel arbeitet, gibt es leider auch keine Informationen zur Architektur von Multikern-Prozessoren, die das Verstndnis u.U. noch weiter gefrdert htten.FazitDas er ein "Freund der Sortierung" ist, lsst sich bereits an der Seitenzahl der Kapitel ablesen. Aber das tut dem runden Eindruck, den das Buch hinterlsst, keinen Abbruch. Die zuvor annvisierte Leserschaft wird mit dem Buch von Clay Breshears bestens zurechtkommen und interessierten Neulingen bietet er eine kompetente Einfhrung in die Thematik. Dieses Buch hat mich zu einigen neuen Ideen inspiriert - weshalb ich es nicht zuletzt deshalb empfehlen kann.--[1]:Dieser Eindruck wird durch die imposante Anzahl von gefundenen Seiten und Dokumenten besttigt, wenn man an Google eine Suchanfrage mit "thread-based vs event-based" schickt. Meine letzte

Anfrage ergab ca. 59 Millionen Seiten!(30.10.2009, 19:00 Uhr).5 von 5 Kunden fanden die folgende Rezension hilfreich. Nur fr "Thread-Monkeys"Von Jrn DinklaMit dem Titel "The Art of Concurrency" wird ein hoher Anspruch geweckt, der nicht erfllt wird. Der Untertitel "A Thread Monkey's Guide to Writing Parallel Applications" wird dem Inhalt schon eher gerecht, aber auch hier gibt es noch einige Kritikpunkte.In diesem Buch wird die threadbasierte Programmierung von Shared-Memory-Rechnern mit POSIX Threads (pthreads), Windows Threads, OpenMP und mit Intel Thread-Building-Blocks in einer C-hnlichen-Sprache erlutert. Fr Java-Programmierer ist es daher nur bedingt geeignet.Die theoretischen Grundlagen von Parallelitt und Nebenlufigkeit werden nur ganz kurz anhand der PRAM erlutert. Andere Modelle, z. B. das Aktor-Modell, Petri-Netze oder Prozess-Kalkle fehlen vllig. Ebenfalls nichts ber "Futures" und "Promises".Was der Autor hingegen schn macht ist, wie er erklrt, wie man aus einem Algorithmus fr die PRAM threadbasierten Code fr Shared-Memory-Architekturen herleitet. Wie man etwas am besten parallelisiert und da aus seiner reichhaltigen Erfahrung schpft.Leser mit gengend Interesse an dieser Programmierung finden hier viele Beispiele. Gegner, die behaupten das Code mit Threads und Shared-Memory leicht kompliziert wird und zu nicht wartbarem Code-Chaos fhrt allerdings auch.Insgesamt werden 13 Algorithmen (davon 6 Sortierverfahren) parallelisiert. Mir persnlich ist nicht klar, warum die Mglichkeiten der Parallelisierung nicht an 2-3 Beispielalgorithmen gezeigt werden. Was will mir dieses Buch beibringen? Die grundlegenden Algorithmen? Diese werden in anderen Bchern (Cormen, Leiserson, Rivest, etc.) besser und ausfhrlicher erklrt. Auch ist die verwendete C-hnliche Programmiersprache didaktisch ungeeignet.Leider gibt es auch keine bungsaufgaben im Buch, so dass der Leser leider nicht selber aktiv werden und das Gelernte berprfen kann.Als ich dann bei Wikipedia unter "Minimal Spanning Tree" folgendes las "Typically, parallel algorithms are based on Bor'vka algorithm'Prim's and especially Kruskal's algorithm do not scale as well to additional processors." , habe ich mich schon gefragt, warum gerade die letzten beiden im Buch implementiert werden.Fazit: Der Untertitel beschreibt das Buch recht gut, ein Buch ber die Kunst des Programmierens, wie z. B. bei Donald Knuths "The Art of Computer Programming" ist es allerdings bei weitem nicht.

KurzbeschreibungIf you're looking to take full advantage of multi-core processors with concurrent programming, this practical book provides the knowledge and hands-on experience you need. The Art of Concurrency is one of the few resources to focus on implementing algorithms in the shared-memory model of multi-core processors, rather than just theoretical models or distributed-memory architectures. The book provides detailed explanations and usable samples to help you transform algorithms from serial to parallel code, along with advice and analysis for avoiding mistakes that programmers typically make when first attempting these computations.Written by an Intel engineer with over two decades of parallel and concurrent programming experience, this book will help you:Understand parallelism and concurrencyExplore differences between programming for shared-memory and distributed-memoryLearn guidelines for designing multithreaded applications, including testing and tuningDiscover how to make best use of different threading libraries, including Windows threads, POSIX threads, OpenMP, and Intel Threading Building BlocksExplore how to implement concurrent algorithms that involve sorting, searching, graphs, and other practical computationsThe Art of Concurrency shows you how to keep algorithms scalable to take advantage of new processors with even more cores. For developing parallel code algorithms for concurrent programming, this book is a must.