

DIE MULTICORE-REVOLUTION UND IHRE BEDEUTUNG FÜR DIE SOFTWAREENTWICKLUNG

Die Taktraten von Mikroprozessoren lassen sich kaum noch steigern und die Industrie setzt nun für weitere Leistungsverbesserungen auf Multicore-Prozessoren, die mehrere CPUs auf einem Chip integrieren. Moderne Laptops, PCs oder Server werden somit zu echten Parallelrechnern, die für jeden erschwinglich sind. Dieser Artikel skizziert die aktuellen Trends und die neuen Herausforderungen für die Softwareentwicklung.

Aktuelle Hardware-Trends

Haben Sie schon bemerkt, dass ab dem Jahr 2002 die Taktraten bei Mikroprozessoren kaum noch gestiegen sind? Wir haben eine Schallmauer erreicht, die kaum überwindbare Grenzen bei Hitzeentwicklung und Energieverbrauch setzt (siehe Abb. 1). Selbst wenn diverse Techniken zur Reduzierung des Energieverbrauchs entwickelt werden, ist die Hoffnung auf einen anhaltenden exponentiellen Anstieg der Taktraten unrealistisch. Die obere Kurve steigt aber noch, sodass die Mooresche Regel der Steigerung der Transistordichte

ungebremst ist. Als Konsequenz setzen Hardwarehersteller nun auf explizite Parallelität durch so genannte Multicore-Prozessoren, um die Performance trotzdem weiter zu verbessern.

Multicore-Prozessoren integrieren mehrere unabhängige Kerne sowie gemeinsam genutzte Cache-Speicher auf einem Chip. Das erhältlich Angebot ist breit: Intel und AMD vermarkten Quad-Core-Prozessoren, SUN bietet den Niagara-2-Chip mit acht Kernen an (siehe Abb. 2), IBM hat den Cell-Prozessor mit neun Prozessoren entwickelt. Darüber hinaus hat Intel bereits

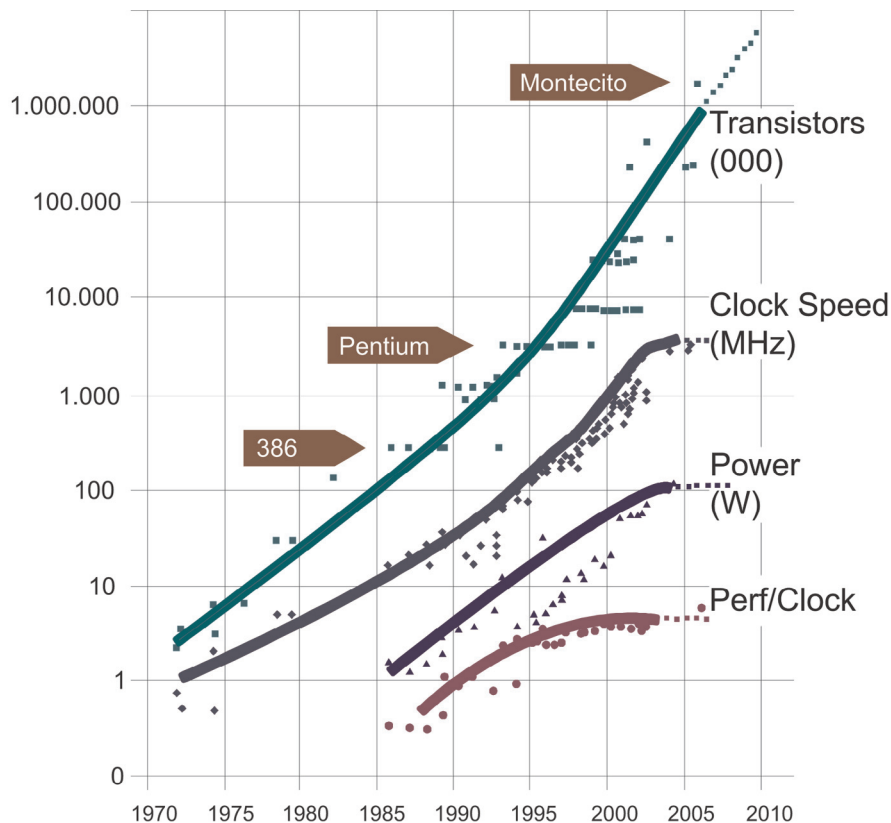


Abb. 1: Trends in der Hardware-Entwicklung.



Dr. Victor Pankratius
 (E-Mail: pankratius@ipd.uka.de)
 ist Leiter der Young Investigator Group „Software Engineering für Multicore-Systeme“ am Institut für Programmstrukturen und Datenorganisation (IPD) der Universität Karlsruhe und Sprecher des GI-Arbeitskreises Software Engineering für parallele Systeme (SEPAS).



Prof. Walter F. Tichy
 (E-Mail: tichy@ipd.uka.de)
 bekleidet seit 1986 den Lehrstuhl für Softwaretechnik der Fakultät für Informatik an der Universität Karlsruhe. Von 2002 bis 2004 war er Dekan der Informatik-Fakultät. Seit 1998 ist er außerdem Direktor für Softwaretechnik am Forschungszentrum Informatik in Karlsruhe.

einen Prototypen mit 80 Kernen vorgestellt. Doch das ist noch nicht der Rekord: Kaum wahrgenommen wurde der 2005 von CISCO entwickelte Metro-Netzwerkprozessor mit 192 Kernen, die alle auf einer Fläche von 3,24 cm² untergebracht sind. Würde man statt der damaligen 130nm-Technologie aktuellere 45nm-Technologie einsetzen, könnte man heute sogar 1.500 Prozessoren auf einen Chip packen. Man spricht schon von Manycore-Chips mit hunderten oder mehr Kernen.

Was bedeutet das für die Softwareentwicklung?

Es dürfte deutlich zu sehen sein, dass sich für die Softwareentwicklung derzeit ein klassischer Paradigmenwechsel ankündigt: der Übergang vom sequenziellen zum parallelen Rechnen auf breiter Front. Parallelverarbeitung ist nicht mehr auf wenige Anwendungsbereiche – wie das wissenschaftliche Rechnen, Datenbanken, oder Parallelismus auf Instruktionsebene – beschränkt, sondern wird nun durch Multi-

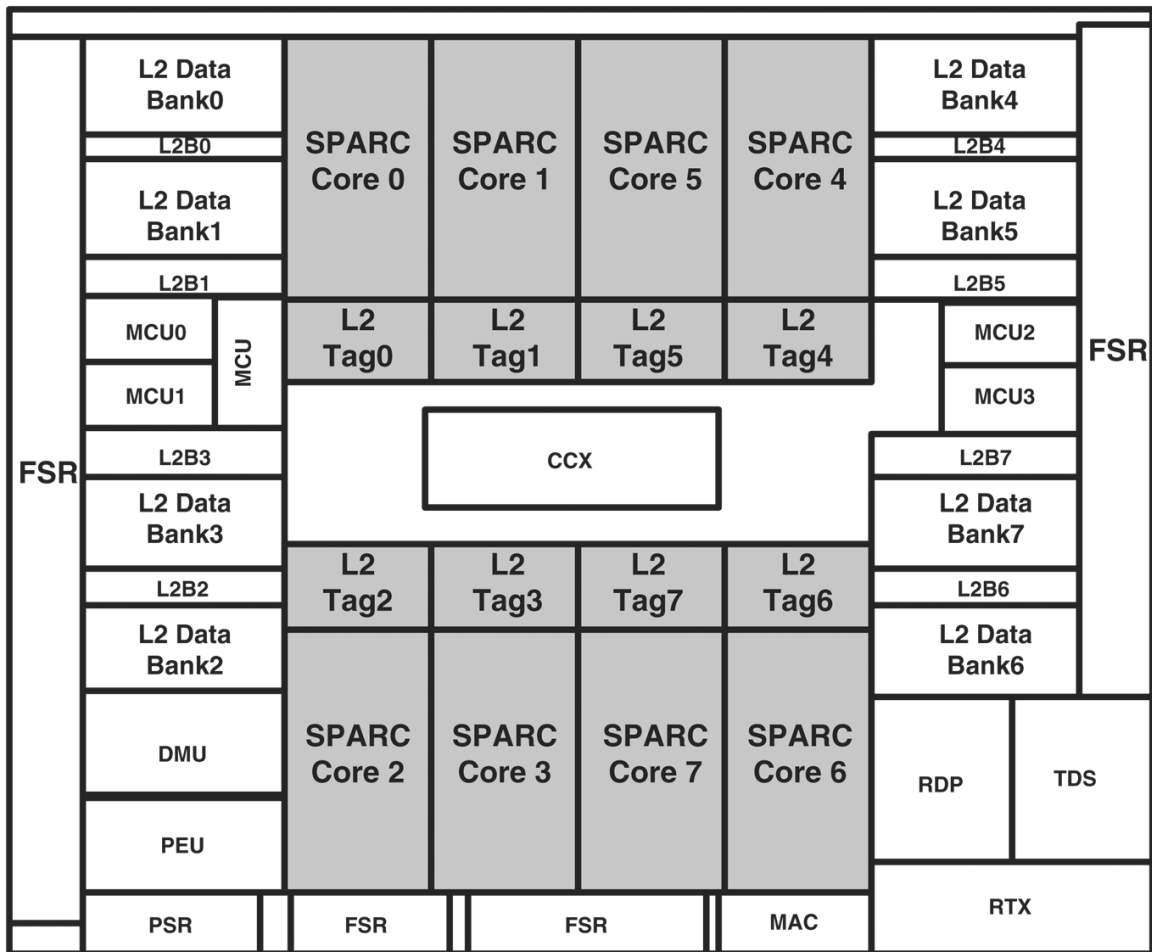


Abb. 2: Architekturskizze des SUN Niagara 2.

core-Prozessorchips für jeden erschwinglich.

Nutzer werden zunehmend nach Software verlangen, die das volle Potenzial ihrer Rechner ausschöpft. Warum sollten sie sonst den neuesten Multicore-Rechner mit mehr Kernen kaufen? Das bedeutet für Softwareentwickler, dass sie nun leistungshungrige Anwendungen aller Art parallelisieren müssen. Gleichzeitig wächst auch die Bandbreite der möglichen Anwendungsgebiete: Von Büroanwendungen über Multimedia und Spielen bis hin zu Web-basierten Informationssystemen oder geschäftskritischen Anwendungen in Unternehmen und eingebetteten Systemen kommt alles für Parallelisierung in Frage.

Falls es allerdings nicht gelingen sollte, die durch Parallelität weiter steigende Rechenleistung durch die Software zu nutzen, könnten wir auf eine zweite Softwarekrise zusteuern.

Kann man die Parallelisierung Compilern überlassen?

Es wäre naiv zu glauben, dass Parallelisierung durch Compiler alleine bewältigt

werden kann. Schon für relativ einfache, numerische Programme funktioniert sie nicht zufriedenstellend. Bei komplexen, nicht-numerischen Anwendungen wird es noch deutlich schwieriger werden. Der Grund ist verständlich: Die automatische Parallelisierung entspricht der Herleitung eines parallelen Algorithmus für ein Problem, für das nur eine sequenzielle Implementierung – und damit nicht einmal eine Spezifikation – vorliegt. Schon die automatische Herleitung sequenzieller Algorithmen aus präzisen Spezifikationen ist kaum praktikabel; im parallelen Fall ist das Problem noch weiter verschärft. Softwaretechniker werden wohl auf absehbare Zeit die Parallelisierung nicht an Automaten delegieren können.

Worauf müssen sich Softwareentwickler einstellen?

Softwareingenieure werden nicht umhinkommen, sich Gedanken über Parallelität in ihren Anwendungen zu machen. Insbesondere müssen sie neue Herangehens-

weisen entwickeln und lernen, Probleme von Anfang an im Lichte der Parallelität zu betrachten. Selbst wenn Werkzeuge bestimmte Aufgaben automatisieren könnten, ist zusätzliches Kontextwissen über parallele Softwareentwicklung unerlässlich. Neue Lernphasen im Umgang mit Parallelität werden notwendig sein, da die Ausbildung von Softwareentwicklern meist nur auf Sequenzialität ausgerichtet ist.

Erste von uns durchgeführte Fallstudien (vgl. [Pan08-a], [Pan08-b]) deuten darauf hin, dass in einigen Multicore-Anwendungen der Hebel für große Beschleunigung auf höheren Abstraktionsebenen liegt. Die Ausnutzung von parallelen Entwurfsmustern, wie z. B. Master-Worker oder Erzeuger-Verbraucher, trägt mehr zur parallelen Performance bei als beispielsweise eine fein-granulare Parallelisierung auf Schleifenebene. Was die Parallelisierung von sequenziellen Anwendungen angeht, zeichnet sich ab, dass ein hoher Refaktorisierungsaufwand notwendig sein kann, um das sequenzielle Programm für die Parallelisierung vorzubereiten. Er liegt auf

der Hand, dass diese beispielhaft genannten Themenbereiche zur Softwaretechnik gehören und damit viele Entwickler betreffen werden.

Wie weit ist die Softwaretechnik?

Die Softwaretechnik kann aus dem langjährigen Einsatz der Parallelität im Bereich des wissenschaftlichen Rechnens sehr viel lernen. Aber das allein wird nicht reichen. Verglichen mit den großen Anwendungen auf PCs und Servern, die in die Millionen Zeilen Code gehen, sind numerische Anwendungen klein und arbeiten mit einer überschaubaren Menge von Datenstrukturen wie Vektoren, Matrizen und einigen unregelmäßigen Gebietszerlegungen. Dagegen werden die funktionsreichen Anwendungen auf PCs und Servern sehr viel mehr Möglichkeiten zur Parallelisierung bieten, womöglich auf mehreren Ebenen gleichzeitig. Ein weiterer, wichtiger Unterschied ist, dass die Entwickler wissenschaftlicher Software meist auch die einzigen Benutzer dieser Software sind. Entsprechend sind Qualitäten wie Benutzerfreundlichkeit, Zuverlässigkeit, Robustheit oder Wartbarkeit dort weniger wichtig als für Software, die von Tausenden oder Millionen von Menschen eingesetzt wird, die eine Lebensdauer von Jahrzehnten hat, unternehmenskritische Daten verwaltet oder sicherheitskritische Anlagen steuert.

Die Forschung ist sich der Probleme bewusst und wird gerade aktiv, brauchbare

Methoden, Konzepte und Werkzeuge zu entwickeln, die es jedem Softwareentwickler ermöglichen, korrekte und effizient ausführbare parallele Programme systematisch zu erstellen. Die Herausforderung für die Softwaretechnik besteht daher darin, die Parallelisierung von großen Anwendungen bei hohen Komplexitäts- und Qualitätsanforderungen zu meistern.

Helfen Sie mit!

Auch Sie, verehrte Leserinnen und Leser, können mit Ihrem Know-how und Ihrer Erfahrung helfen, dieses spannende Gebiet voranzubringen. Der Arbeitskreis *Software-Engineering für parallele Systeme (SEPAS)* der Gesellschaft für Informatik bietet ein Austauschforum für Wissenschaftler und Praktiker. Registrieren Sie sich kostenlos unter [SEPAS], um sich regelmäßig über die Aktivitäten des Arbeitskreises zu informieren. Nehmen Sie an den Treffen teil, diskutieren Sie mit uns aktuelle Fragestellungen, berichten Sie über Probleme oder machen Sie selbst konkrete Lösungsvorschläge für die Entwicklung von Multicore-Anwendungen. Wir würden uns über Ihr Feedback sehr freuen.

Fazit

Wir befinden uns an einem wichtigen Wendepunkt: Multicore-Rechner machen die Entwicklung paralleler Software auf breiter Front erforderlich. Um das volle

Potenzial der Hardware nutzen zu können, müssen wir uns umstellen und Parallelität schon beim Anfang der Softwareentwicklung berücksichtigen. Wir haben nun die seltene Möglichkeit, eine fundamentale Umwälzung in der Informatik mitzerleben, und bekommen die Chance, neuartige Anwendungen zu entwickeln. Nutzen wir sie. ■

Literatur & Links

[IWMSE] International Workshop on Multicore Software Engineering 2008 (IWMSE08), Leipzig, 11. Mai, 2008, siehe: www.multicore-systems.org/iwmse

[Pan08-a] V. Pankratius, C. Schaefer, A. Jannesari, W.F. Tichy, Software Engineering for Multicore Systems – An Experience Report, in: Proc. Of International Workshop on Multicore Software Engineering (IWMSE), im Rahmen der ICSE2008, ACM, 2008

[Pan08-b] V. Pankratius, W.F. Tichy, Parallelizing BZip2. A Case Study in Multicore Software Engineering, Technischer Bericht, Institut für Programmstrukturen und Datenorganisation (IPD), Universität Karlsruhe, 2008

[SEPAS] GI-Arbeitskreis Software Engineering für parallele Systeme (SEPAS), siehe: www.multicore-systems.org/gi-ak-sepas

[Tic08] W. Tichy, V. Pankratius, Herausforderung Multikern-Systeme, Software Engineering Tagung 2008, München 2008