Photogrammmetrische Woche 1979
Institut für Photogrammetrie
Schriftenreihe, Heft 6, 1980

PARALLELISMUS - ASPEKT ZUKÜNFTIGER DATENVERARBEITUNG

H. Müller, Erlangen

1. Datenverarbeitung im Menschen und im klassischen Universalrechner

Datenverarbeitung ist prinzipiell so alt wie die Menschheit selbst. Im weitesten Sinne kann auch die Gehirntätigkeit als Datenverarbeitung angesehen werden. Sieht der Mensch mit dem Auge eine Gefahr auf sich zukommen und reagiert mit einer Handlung zur Abwendung dieser Gefahr, so ist ein wesentlicher Teil dieses Vorgangs Datenverarbeitung. Die mit dem Auge aufgenommenen optischen Daten, die über Nervenleitungen zum Gehirn übermittelt werden, werden dort verarbeitet zur Information "Gefahr". Dieses interne Gefahr-Datum wird weiterverarbeitet unter Verwendung von im Gedächtnis gespeicherter "Erfahrung". Ergebnis ist möglicherweise ein "Wunsch"-Datum: "Man sollte sich aus der Gefahrenzone entfernen". Dieses Wunsch-Datum wird weiterverarbeitet zu Signalen an gewisse Muskel-Gruppen, die das Weglaufen schließlich realisieren. Daß dabei im Menschen viele Dinge parallel ablaufen, will ich an dieser Stelle nicht weiter untersuchen.

Die Grundstruktur maschineller Datenverarbeitung im klassischen Universalrechner sieht sehr ähnlich aus. Über ein Eingabe-Gerät (Tastatur, Lochkartenleser, automatische Beleglese-Maschine usw.) werden Daten in das System eingegeben, in einem Prozessor verarbeitet mit Zwischenspeicherung von Zwischen-Ergebnissen und schließlich die Verarbeitungsergebnisse durch ein Ausgabe-Gerät (z.B. Drucker, Bildschirm) ausgegeben. Häufig sind Eingabe- und Ausgabe-Gerät in einem Gerät (Terminal) zusammengefaßt.

Zum Verständnis der Probleme paralleler Datenverarbeitungsvorgänge ist eine etwas genauere Kenntnis der Form der Datenverarbeitung im klassichen Universalrechner erforderlich. Jede Datenverarbeitung ist ein Prozeß, der - meist in vielen einfachen Einzelschritten - aus Eingabe-Daten mit gewissen Zwischenergebnissen schließlich Ausgabe-Daten erzeugt. Welche Einzelschritte in welcher Reihenfolge nacheinander oder gleichzeitig auszuführen sind, beschreibt man üblicherweise durch einen Algorithmus bzw. maschinennäher durch ein Programm. Als konkretes Beispiel sei die Bildung des inneren Produktes zweier n-dimensionaler Vektoren $\underline{a} = (a_1,\ldots,a_n)$ und $\underline{b} = (b_1,\ldots,b_n)$ betrachtet. Die mathematische Beschreibung ist durch die Formel

$$IP = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$$

gegeben.

Aus dieser Formel kann man folgenden (sequentiellen) Algorithmus ablesen (für n = 8):

1.	Berechne	$ZE\ 1 = a_1 b_1$
2.	"	$ZE\ 2 = a_2 b_2$
3.	"	$ZE\ 3 = ZE1 + ZE2$
4.	"	$ZE\ 4 = a_3 b_3$
5.	"	$ZE\ 5 = ZE3 + ZE4$
...		
14.	"	$ZE14 = a_8 b_8$
15.	Resultat:	$IP = ZE13 + ZE14$

In der Programmiersprache ALGOL 60 kann dieser Algorithmus durch folgendes Programm(-Stück) beschrieben werden:

```
S  := 0;
For I := 1 step 1 until N do
 S := S  + A[I]* B[I];
IP := S
```

Dieses Programm einer höheren Programmiersprache muß für die direkte Verarbeitung durch die Maschine noch in Maschinensprache übersetzt werden. Greifen wir zur Veranschaulichung eine Anweisung des ALGOL-Programms heraus: Die zur Anweisung "S := S + A[I]* B[I]" gehörige Maschinenbefehlsfolge hat etwa folgendes Aussehen:

Hole a_i in den Akkumulator A

Hole b_i in das Operandenregister B

Multipliziere (A:=A*B)

Hole S nach B

Addiere (A:=A+B)

Bringe A nach Speicherplatz von S

Ein großer Teil der Operationen sind Daten-Transporte vom bzw. zum Speicher. Zu den oben enthaltenen Hole- und Bringe-Befehlen kommen noch Hole-Nächsten-Befehl-Anweisungen hinzu, da auch das Programm im Speicher enthalten ist. Bei hoher Verarbeitungsgeschwindigkeit des Prozessors kann es daher zu Engpässen auf dem für den Daten-Transport zuständigen "Bus" zwischen Prozessor und Speicher kommen. Zu Beginn der Computer-Zeit war dies kein Problem, weil die Verarbeitungszeiten im Prozessor wesentlich größer als die Transportzeiten waren. Durch die enorme Entwicklung der Halbleiter-Technologie liegen aber heute die Schaltzeiten im Bereich von Nanosekunden (1 ns = 10^{-9}s). Als physikalische Grenze macht sich dabei und insbesondere beim Daten-Transport die endliche Ausbreitungsgeschwindigkeit elektromagnetischer Wellen (300 000 km/s = 30 cm/ns) bemerkbar. Diese prinzipielle Grenze der Verarbeitungsgeschwindigkeit kann dadurch überwunden werden, daß man Teile einer Datenverarbeitungsaufgabe abspaltet und parallel von einem oder mehreren anderen Rechnern bearbeiten läßt. Günstig für diesen Gedanken wirken sich natürlich auch die stark gesunkenen Kosten für die Hardware aus.

2. Parallel-Datenverarbeitung

Soll eine Datenverarbeitungsaufgabe von mehreren Rechnern übernommen werden, spielt natürlich die Aufteilbarkeit in Teilaufgaben eine wesentliche Rolle. Für theoretische Untersuchungen macht man dabei die Annahme, daß man beliebig viele Prozessoren zur Verfügung hat und stellt die Frage, wie viele Schritte braucht eine Berechnung bei maximaler Parallelisierung. Betrachten wir in dieser Richtung das Beispiel "Inneres Produkt". Haben wir acht Prozessoren, so können die Produkte $a_1b_1,\ldots,a_8b_8$ parallel berechnet werden in einem ersten Schritt. In einem zweiten Schritt können jeweils zwei Zwischenergebnisse (durch vier Prozessoren) addiert werden. Im dritten Schritt fassen zwei Prozessoren wieder je zwei Zwischenergebnisse additiv zusammen, im vierten Schritt ergibt sich IP. Benutzen wir // als Zeichen für "parallel", ergibt sich damit folgender Parallel-Algorithmus:

1. $Z_1 := a_1b_1$ // $Z_2 := a_2b_2$ //...// $Z_8 := a_8b_8$
2. $Z_{12} := Z_1+Z_2$ // $Z_{34} := Z_3+Z_4$ // $Z_{56} := Z_5+Z_6$ // $Z_{78} := Z_7+Z_8$
3. $Z_{1234} := Z_{12}+Z_{34}$ // $Z_{5678} := Z_{56}+Z_{78}$
4. $IP := Z_{1234}+Z_{5678}$

Durch Parallelisierung war es also möglich, mit 4 Schritten im Vergleich zu 15 Schritten beim sequentiellen Algorithmus auszukommen, allerdings wurden dafür acht Prozessoren benötigt.

Dies Beispiel zeigt, daß - in gewissen Grenzen - Rechenzeit durch Rechenraum (Anzahl an verfügbaren Prozessoren) austauschbar ist.

Wie bei der Datenverarbeitung allgemein hat man auch bei der parallelen Datenverarbeitung zwei Teilbereiche zu betrachten, den der Hardware - also der Maschinen, die zur parallelen Datenverarbeitung fähig sind - und den der Software - also Programmiersprachen und Programme zur Beschreibung paralleler Algorithmen. Beide Bereiche beeinflussen sich zwar gegenseitig, könnten hier aber parallel abgehandelt werden. Hier zeigt sich nun eine prinzipielle Schwierigkeit beim Parallelisieren: Der Mensch ist gewöhnt, sequentiell zu denken. Es wird sicher erst einer längeren Lernphase bedürfen, bis Programmierer in der Lage sind, die im zu programmierenden Problem steckenden Parallelismen auch durch parallele Programme zu beschreiben.

3. Parallelismus im Bereich der Hardware

Es gibt inzwischen einige unkonventionelle Rechensysteme, die mehr oder weniger gut für parallele Datenverarbeitung geeignet sind. Als unechte Parallelverarbeitung ist hier zunächst "Multiprogramming" zu nennen, eine Methode, die schon relativ früh verwendet wurde, um Wartezeiten, die durch langsame Ein/Ausgabe-Operationen verursacht wurden, zu vermeiden. Dabei werden mehrere Programme scheinbar gleichzeitig verarbeitet und Wartezeiten eines Programms durch Ausführung eines Stücks eines anderen Programms sinnvoll genutzt.

Gewisse Parallelität ist auch schon bei einigen klassischen Universalrechnern vorhanden, z.B. in Form eines Parallel-Addierers, der beim Addieren alle Stellen der binären Operanden gleichzeitig verarbeitet. Auch bei der Steuerung durch ein Mikroprogramm werden teilweise mehrere Mikrooperationen parallel ausgeführt.

Vektorprozessoren (z.B. MAP) sind so konstruierte Rechner, daß alle Komponenten eines Vektors gleichzeitig verarbeitet werden können. Der 1. Schritt des parallelen Algorithmus für das innere Produkt ist ein Elementarschritt eines Vektorprozessors.

Feldrechner sind Rechenanlagen, deren wesentlicher Bestandteil ein quadratisches Feld von n x n Prozessoren ist, die, gesteuert durch eine Überwachungseinheit, zu jedem Zeitpunkt alle dieselbe Operation durchführen. Günstig lösen lassen sich auf diese Weise Probleme, die man in Matrix-Form formulieren kann und bei denen für jedes Matrix-Element gleichartige Operationen auszuführen sind. Drei Vertreter dieser Klasse sind Burroughs Illiac IV, ICL DAP (Distributed array processor) und Goodyear STARAN.

Während bei Feldrechnern alle Prozessorelemente die gleiche Operation ausführen, können in Multiprozessor-Systemen die einzelnen Prozessoren weitgehend unabhängig voneinander arbeiten. Die Verknüpfung geschieht über gemeinsame Speicher, auf die mehrere Prozessoren zugreifen können. Beispiele von Multiprozessor-Systemen, die sich noch im Forschungs- und Entwicklungsstadium befinden, sind C.mmp, X-Tree und EGPA. C.mmp ist ein von der Carnegie-Mellon-University untersuchtes System, bestehend aus 16 PDP 11-Computern und 16 Speicherblöcken und einem Verbindungsnetzwerk (Kreuzschienenverteiler), das die Verbindung von jedem Prozessor zu jedem Speicherblock ermöglicht. X-Tree ist ein baumartig verknüpftes Rechnernetz. Hinter dem Kürzel EGPA (Erlanger General Purpose Array) verbirgt sich eine an der Universität Erlangen-Nürnberg entworfene und teilweise (mit AEG 80/60-Rechnern) realisierte pyramidenartige Rechnerstruktur. Das Prinzip, jeden Prozessor mit jedem Speicherblock zu verbinden, läßt sich mit vertretbarem Aufwand nur für relativ kleine Anzahlen realisieren. In X-Tree und EGPA wird die Speicherzugriffsmöglichkeit der Prozessoren deshalb lokal begrenzt. Über Zwischenstationen ist letztlich aber auch hier ein Informationsaustausch zwischen beliebigen Prozessoren möglich.

Die bisher erwähnten Rechensysteme hatten alle eine feste Struktur. Neuerdings beschäftigt man sich auch mit rekonfigurierbaren Rechnern, deren Struktur an die jeweilige Aufgabe anpaßbar ist.

Schließlich sind in diesem Zusammenhang die Kommunikations-Netze zu nennen, bei denen mehrere Groß-Rechenanlagen durch Datenübertragungswege miteinander verbunden sind. Das bekannteste derartige Netz ist das ARPA-Netz, das eine Vielzahl von Computern in den USA zu einem Verbund-Netz zusammenschließt.

Zusammenfassend kann man sagen, von der Technologie-Seite gesehen, lassen sich Rechensysteme hoher Komplexität relativ preiswert aufbauen. Die Probleme liegen überwiegend auf der nun zu betrachtenden Software-Seite, die die zur Steuerung der Rechner erforderlichen Programme liefern muß.

4. Parallelismus im Bereich der Software

Die Datenverarbeitung auf dem klassischen Universalrechner wurde durch ein Programm gesteuert. Wie geschieht die Steuerung eines Systems von Prozessoren? Grundsätzlich wird auch ein Mehrprozessorsystem durch ein oder mehrere Programme gesteuert, nur muß hierbei klar beschrieben sein, welche Teilaufgaben parallel verarbeitet werden können oder sollen. Die meisten höheren Programmiersprachen haben keine Sprach-Elemente, die Parallelität auszudrükken gestatten. Ausnahmen sind z.B. APL (A Programming Language von Inversen), eine Sprache, die relativ mächtige Grundoperationen (z.B. Inneres Produkt) besitzt, ALGOL 68, wo Parallelität durch das Trennzeichen "," ausgedrückt werden kann und sequentielle Abarbeitung durch das Trennzeichen ";" erzwungen wird, sowie "Concurrent PASCAL", eine vorwiegend zum Zwecke der Betriebs-System-Programmierung entworfene Erweiterung der Sprache PASCAL.

Diese Vernachlässigung der Parallelität hat vermutlich zwei Gründe. Zum einen die historische Entwicklung aus den Maschinensprachen, die nur sequentielle Abarbeitung eines Programms kennen, zum anderen die sequentielle Denkweise des menschlichen Gehirns.

Bei der Analyse eines komplexen Systems ist der Mensch leicht geneigt, die im System gleichzeitig ablaufenden Vorgänge zeitlich nacheinander zu analysieren. Um parallele Algorithmen zu entwerfen, bedarf es daher eines gewissen Umdenkungsprozesses. Ein krasses Beispiel dafür, wie aus einem von Natur aus parallelen Algorithmus durch die Programmiersprache ein sequentielles Programm wird, ist die Matrizenaddition. Aus der Formel $c_{ij} = a_{ij} + b_{ij}$ $(i,j=1,..,n)$ wird das Programm

```
For I := 1 step 1 until n do
   For J := 1 step 1 until n do
      c[I,J] := a[I,J] + b[I,J]
```

Eine Parallelität zulassende Formulierung könnte etwa lauten:

```
For all I,J with 1 ≦ I ≦ n and 1 ≦ J ≦ n do
   c[I,J] := a[I,J] + b[I,J]
```

Charakteristisch für die Datenverarbeitung im klassischen Universalrechner ist die Verarbeitung von genau einem Wort pro Befehl ("word-at-a-time-style"). Diese Denkweise und der sich daraus ergebende Programm-Stil sind 1977 von John Backus, der wesentlich an der Entwicklung der Programmiersprachen FORTRAN und ALGOL beteiligt war, kritisiert worden. Er propagiert eine gänzlich andersartige Methode, die "Funktionale Programmierung", eine auf dem aus der Rekursionstheorie bekannten Lambda-Kalkül aufbauende Beschreibung von Berechnungsverfahren, die ohne Variablen auskommt und die die Beschreibung paralleler Vorgänge leicht ermöglicht.

Einen kleinen Eindruck mag als Beispiel ein funktionales Programm für das innere Produkt geben:

$$IP = (/+) \circ (\alpha\times) \circ Trans$$

Photogrammmetrische Woche 1979
Institut für Photogrammetrie
Schriftenreihe, Heft 6, 1980

Dabei bedeuten einige der vorkommenden Zeichen Funktionen, andere bedeuten Operationen, die auf Funktionen operieren. Andeutungsweise sei die Semantik der vorkommenden Zeichen erklärt durch:

$\circ$ (Komposition) : $[fog](x) = f(g(x))$

/ (Insertion) : $[/+](x_1,\ldots,x_n) = x_1 + x_2 + \ldots + x_n$

α (apply to all): $[\alpha f](x_1,\ldots,x_n) = (f(x_1),\ldots,f(x_n)$

$$\text{Trans}((x_{11},x_{12},\ldots,x_{1m}),\ldots,(x_{n1},\ldots,x_{nm})) = ((x_{11},x_{21},\ldots,x_{n1}),\ldots,(x_{1m},\ldots,x_{nm}))$$

$+$ Addition; $\times$ Multiplikation

Eine "Berechnung" für das innere Produkt von $\underline{a} = (1,2,3)$ mit $\underline{b} = (6,5,4)$ kann man kurz so beschreiben:

$$IP(\langle\langle 1,2,3\rangle,\langle 6,5,4\rangle\rangle) =$$

$$\text{Insert Add}(\text{Apply to all Mult}(\underbrace{\text{Trans}(\langle\langle 1,2,3\rangle,\langle 6,5,4\rangle\rangle)}_{\langle\langle 1,6\rangle,\langle 2,5\rangle,\langle 3,4\rangle\rangle}))$$

$$\underbrace{\langle 1\cdot 6, 2\cdot 5, 3\cdot 4\rangle = \langle 6,10,12\rangle}$$

$$\langle 6 + 10 + 12\rangle = \langle 28\rangle$$

Diese Art der Berechnung legt die Einzelschritte nicht mehr als nötig fest. Ein typischer Kandidat für Parallelverarbeitung ist der Operator "apply to all". Eine Aufgabe der Rechnerarchitektur wird es sein, Hardware-Strukturen zu entwickeln, die möglichst effizient derartige funktionale Programme bearbeiten können. Ob sich diese funktionale Programmiermethode durchsetzen wird, kann heute noch nicht abgeschätzt werden, aber die Tendenz ist, nicht die Programmiersprachen den Rechnern anzupassen, sondern umgekehrt für Sprachen, die sich gut zur Problembeschreibung eignen, geeignete Rechner zu entwickeln.

5. Fallstudie: Sichtbarkeitsproblem bei der Zentralprojektion von Höhenlinien

In den Jahren 1972 bis 1977 wurde im Institut für Mathematische Maschinen und Datenverarbeitung der Universität Erlangen-Nürnberg mit Förderung durch das BMVg ein "Erlanger Projekt Ineraktive Kartographie (EPIK)" durchgeführt, dessen Hauptziel Entwurf und Realisierung eines interaktiven topographischen Informationssystems war. Insbesondere enthält dieses System für ein bestimmtes Gebiet (Teil des Wiesent-Tals in der fränkischen Schweiz) die Daten für eine Höhenlinienkarte. Aus diesen Daten kann man (auf dem Bildschirm eines Terminals) ein Bild erzeugen, das die Sicht von einem gewählten Blickpunkt (z.B. Flugzeug) auf das Gebiet simuliert (Zentralprojektion). Bei ungünstigem Verlauf - z.B. bei steilem Gelände oder bei niedriger Blickpunktwahl - müssen dabei nicht sichtbare Abschnitte von Höhenlinien berechnet und bei der Ausgabe im Bild unterdrückt werden. Dieses Sichtbarkeitsproblem ist sehr rechenintensiv. Es soll nun untersucht werden, wie dieses Problem zu lösen ist, insbesondere im Hinblick auf Parallelisierbarkeit.

Bei der Zentralprojektion wird jeder Geländepunkt über einen Strahl vom Blickpunkt aus auf eine Bildebene projiziert. Das Bild einer durch einen Polygonzug approximierten Höhenlinie wird durch Verbindung der Bilder der Ecken dieses Polygonzuges gewonnen. Ein Teilstück einer Höhenlinie ist nicht sichtbar, wenn es im "Schatten" einer Höhenlinie höheren Niveaus liegt. Wegen der Approximation durch Geradenstücke reduziert sich das Problem lokal darauf, die relative Lage einer Strecke AB (Teil der Höhenlinie auf Niveau n) und eines Vierecks V (Schatten einer Teilstrecke der Höhenlinie auf Niveau n+1) zu bestimmen. Zu lösen sind die Teilprobleme (1) Liegen A und/oder B im Inneren

Photogrammmetrische Woche 1979
Institut für Photogrammetrie
Schriftenreihe, Heft 6, 1980

des Vierecks V? und (2) Gibt es Schnittpunkte von AB mit den Seiten von V? Diese Fragen führen auf die Lösung von sechs linearen Gleichungssystemen. Soweit eine grobe Analyse des Lösungsverfahrens. Nun zu der Frage der Parallelisierbarkeit. Ich will zwei extreme Möglichkeiten andeuten.

Verfahren I: Parallelisierung auf "oberster Ebene":

Zerteile das Gelände in Sektoren vom Blickpunkt aus und führe die Sichtbarkeitsberechnungen für jeden Sektor parallel aus. Offenbar sind die Sichtbarkeitseigenschaften in den einzelnen Sektoren unabhängig voneinander. Zusatzaufwand ist nur bei der Aufteilung in Sektoren und beim Zusammensetzen der Teilbilder erforderlich.

Verfahren II: Parallelisierung auf "unterster Ebene":

Löse für jedes Strecken-Vierecks-Paar die sechs Gleichungs-Systeme parallel auf sechs Prozessoren. Bei dieser Lösung ist allerdings ein sehr viel größerer Synchronisationsaufwand zu leisten. Für jedes Strecken-Vierecks-Paar sind die Ergebnisse der sechs Prozessoren miteinander zu verknüpfen.

Ob eines der beiden Verfahren günstiger als das andere ist oder ob man beide Verfahren miteinander mischen sollte, ist vermutlich von der zur Verfügung stehenden Hardware abhängig. Vergleichende Versuche sind vorgesehen.

6. Zusammenfassung

Ich habe versucht, Ihnen einen Aspekt zukünftiger Datenverarbeitung - den Parallelismus - vorzustellen. Die Technologie ermöglicht, Hardware zur Verfügung zu stellen, die auch hohe Anforderungen an Parallelverarbeitung erfüllen kann. Große Schwierigkeiten bestehen jedoch noch in der Formulierung paralleler Algorithmen und in der Steuerung paralleler Datenverarbeitungsprozesse.

Die volle Ausnutzung der Möglichkeiten von Multiprozessoren verlangt letztlich vom Anwender eine Problembeschreibung, aus der die parallel bearbeitbaren Teilprobleme klar erkennbar sind.

Literatur

J. BACKUS: Can Programming be liberated from the von Neumann Style? A functional style and its algebra of programms. Communication ACM, Vol. 21, No. 8 (1978).

P. BRINCH-HANSEN: The architecture of concurrent pascal. Prentice Hall (1977).

P.H. ENSLOW: What is a distributed data processing system? Computer, Vol. 11, No. 1 (1978).

W. HÄNDLER: Aspects of parallelism in computer architecture. in: Parallel Computers - Parallel Mathematics, M. Feilmeier (ed.), Int. Ass. for Math. and Computers in Simulation (1977).

W. HÄNDLER e.a.: EPIK Erlanger Projekt Interaktive Kartographie. Arbeitsberichte des IMMD, Universität Erlangen-Nürnberg, Bd. 11, Nr. 6 (1978).

G. NAGY/S. WAGLE: Geographic data processing. Computing Surveys, Vol. 11, No. 2 (1979).

Photogrammmetrische Woche 1979
Institut für Photogrammetrie
Schriftenreihe, Heft 6, 1980

Zusammenfassung

Durch eine enorme Entwicklung der Halbleitertechnologie sind die materiellen Hilfsmittel der Datenverarbeitung (Hardware) billig geworden. Unkonventionelle Rechensysteme erlauben durch Parallel-Verarbeitung eine Erhöhung der Rechenleistung. Es wird über verschiedene Möglichkeiten und Probleme der parallelen Datenverarbeitung berichtet.

Parallelism - an aspect of future data processing

Abstract

Data-processing hardware has become cheap due to the tremendous progress made in semiconductor technology. Increased efficiency has become possible by the use of unconventional computer systems which allow parallel processing. The paper reports on several possibilities and problems of parallel data processing.

Parallélisme - un aspect du traitement futur des données

Résumé

L'extraordinaire développement de la technologie des semiconducteurs a fait baisser considérablement les prix du matériel rentrant dans la constitution physique des systèmes de traitement des informations (hardware). Des systèmes de calcul nouveaux permettent, grâce à un traitement en parallèle des informations, d'augmenter considérablement les capacités de calcul. Les différentes possibilités de traitement parallèle des informations et leurs problèmes respectifs sont exposés au cours de cette conférence.

Paralelismo - aspecto del procesamiento futuro de datos

Resumen

El enorme progreso de la tecnología de semiconductores ha hecho baratos los medios materiales (hardware) empleados en el procesamiento de datos. Sistemas de cómputo no convencionales permiten aumentar el rendimiento por procesamiento paralelo. Se informa acerca de varias posibilidades y problemas del procesamiento paralelo de datos.

Prof. Dr. rer. nat. Horst Müller
Institut für Mathematische Maschinen und Datenverarbeitung
(Informatik) der Universität Erlangen-Nürnberg
D-8520 Erlangen, Martensstraße 3