Kapitel 1
Klassifikationen

Es werden Parallelrechner klassifiziert nach verschiedenen Gesichtspunkten:

1.1. Klassifikation nach der Kopplung der rechnenden Einheiten

Von Interesse ist die Art der Kopplung zwischen den rechnenden Einheiten.

1.1.1. Enge Kopplung

Die rechnenden Einheiten sind über gemeinsam genutzte Ressourcen aneinander gekoppelt

- gemeinsamer Takt (s. zellulare Automaten, Coprozessoren)
- gemeinsame Speicherbänke, über die der Datenaustausch läuft
  (DMA, Graphikprozessoren, Multiprozessoranlagen)
- Bus, Punkt-zu-Punkt-Verbindungen oder Verbindungsnetzwerke
  Sie müssen sich für den Datenaustausch miteinander synchronisieren über Busvergabe- und Synchronisationsleitungen (Feldrechner, neuronale Netze)

1.1.2. Lose Kopplung

Die rechnenden Einheiten sind über ein Kommunikationssystem gekoppelt (LAN, Internet). Es sind unabhängige Einheiten, die sich zur Kommunikation über Protokolle miteinander synchronisieren. Sie bilden für geeignete Algorithmen sehr leistungsfähige Parallelrechner.

1.2. Klassifikation nach der Art von Daten- und Befehlsströmen

1.2.1. SISD - single instruction, single data

Ein Instruktionsstrom verarbeitet einen Datenstrom.

Daten wandern durch den Rechner und werden von einem Befehlsstrom manipuliert: Einzelrechner ohne Befehlspipelining und ohne superskalare Funktionen.

1.2.2. SIMD - single instruction, multiple data

Ein Befehl wirkt auf viele Daten parallel.

Parallelrechner können so betrieben sein: zellulare Automaten, Feldrechner.

In Abwandlung läuft auf allen Rechnern das gleiche Programm, aber mit jeweils verschiedenen Daten; datenabhängig werden ggf.verschiedene Zweige des Programms durchlaufen. Rechner mit Vektorarithmetik oder MMX-Befehlen fallen ebenfalls in diese Kategorie.

1.2.3. MISD - multiple instruction, single data

Auf ein Datum werden parallel viele Befehle angewandt. Das ist der Fall z. B. bei der Analyse von Daten, die auf verschiedene Aspekte hin untersucht werden. Ein anderer Fall sind Operationen auf Vektoren, bei denen zu einem Zeitpunkt verschiedene Komponenten in unterschiedlichen Stufen einer Pipeline sind.

1.2.4. MIMD - multiple instruction, multiple data

Es können parallel mehrere Threads in einem Rechner laufen. Im Extremfall liegt ein lose gekoppeltes Rechnernetz vor.

1.3. Klassifikation nach der Parallelität in Algorithmen

1.3.1. Rein sequentieller Code

Keine Parallelität möglich -> normaler Rechner

1.3.2. Sequentieller Code

Teile parallel ausführbar -> superskalare Rechner

1.3.3. Austausch mit Nachbarn nach vielen Schritten

Schleifen unabhängig parallel

```
for i = 1 to N do
 y(i) := f(x(i))
 z := g(y(i))
endfor
```

y(i) := f(x(i)) auf N Rechnern parallel rechenbar nach einer Vor- und Nachverarbeitung:

- verteile x(i) auf die Rechner {im Extremfall muß ein Rechner nur seine Nummer i kennen und x ist konstant};
- berechne parallel y(i) { macht Sinn, wenn die Berechnung von f(x(i)) lange dauert im Vergleich zu der Verteilung der x(i) }
- fasse die Ergebnisse zusammen: z := g(y(i)) { meist ein einfacher Schritt }

Wenn die Rechnung lange dauert und nur wenige Daten auszutauschen sind, können die Rechner in einem Netz dafür parallel geschaltet werden.

Beispiele:

- Knacken des Schlüssels einer mit dem US-Datenverschlüsselungsstandard (data encryption standard, DES) codierten Nachricht

  (es sind $2^{56} \approx 7{,}2 \cdot 10^{16}$ mögliche Schlüssel zu testen)

  Es seien 10.000 Rechner im Netz verfügbar. Dann muß ein Rechner $7{,}2 \cdot 10^{12}$ Schlüssel durchtesten. Es dauere ein Test 10 µs ==> 100.000 Tests/s

  Dann dauert die Suche $7{,}2 \cdot 10^{7}$ s oder länger als zwei Jahre.

  Mit schnellen Workstations, bei denen ein Test 1 µs dauert, kann die Suche in 85 Tagen durchgeführt werden.

- Faktorisierung des Produkts zweier großer Primzahlen

  (Auf der Schwierigkeit dieser Faktorisierung beruht die Sicherheit von Autentifizierun gen im Netz und die Schlüsselübertragung mit dem RSA-Verfahren)

  Für $Q = p_1 * p_2$ mit p_1 und p_2 ca. 20stellige Primzahlen ist die Suche noch nicht machbar.
  Sei $2^{62} < p1 \le 2^{63} \approx 10^{19}$ und $2^{64} < p2 \le 2^{65}$

  ==> Q ist mit 128 Bit darstellbar.

  Die Suche dauert dann rund $5 \cdot 10^{18}$ Schritte, da nur ungerade Zahlen berücksichtigt werden müssen.

  Quadratisches Sieb nach Pomerance: $Q = (a+b)(a-b) = a^2 - b^2$

  Sei $(R-1)^2 < Q \le R^2$

  initialisiere `a := R;  b := 0;  S := a^2 - Q - b^2`

```
while S ≠ 0 do
    if S < 0    {* a ist noch zu klein, erhöhe um 1*}
      then
        begin S := S + 2a + 1;
              b := (2a + 1 + b^2)^1/2; {ganzzahlig abgerundet}
              a := a + 1;
        end
      else
        begin S := S - 2b - 1;
              b := b + 1;
        end;
p1 := (a + b);  p2 := (a - b).
```

Aufteilung auf N Rechner: Rechner i wird initialisiert mit

$a := R + K\cdot(i - 1)$; $b := \{(R + K\cdot(i - 1))^2 - Q)^{1/2}$ {ganzzahlig abgerundet}

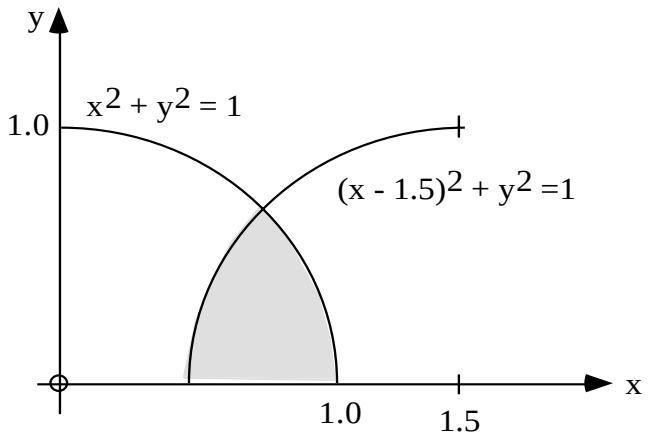
mit $K := R/(2N)$

- Monte-Carlo-Rechnungen

  Es werden parallel N Zufallszahlen erzeugt. Mit ihnen wird eine Rechnung durchgeführt, danach erneut Zufallszahlen erzeugt und die Ergebnisse statistisch ausgewertet. Man erhält eine gute Überdeckung des Parameterraumes.
  Der einfache statistische Fehler ist bei M Versuchen ~ $(M)^{1/2}$.

  $M = 10^6$ Versuche ergeben einen statistischen Fehler von 1‰. Verteilung der M Versuche auf N << M Rechner.

  Z. B. Flächenberechnung eines Viertelkreises: Rückführen der Integration auf die Bestimmung des Flächenanteils des Viertelkreises am umschließenden Rechteck.



```
for all i=1 to N do
   l(i):=0;
   for k=1 to M/N do
   x:=rdn; y:=rdn; if x² + y² ≤ 1 then l(i):=l(i) + 1;
   endfor
integral:=summe(l(i))/M
```

- Integration als Obersumme:

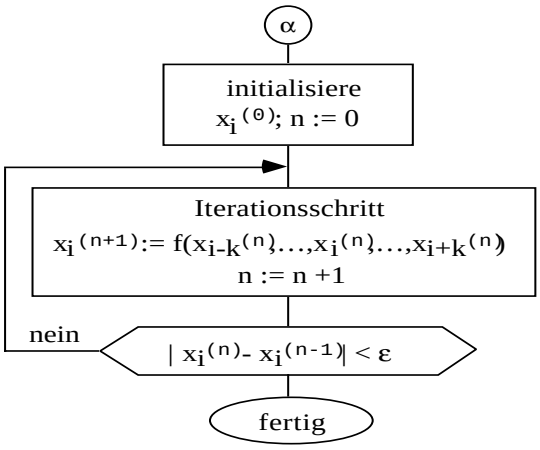
$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{N} f\left(a + (b-a)\frac{i}{N}\right)\frac{(b-a)}{N}$$

Nach der Berechnung von f(a + (b-a)i/N) ist über alle Teilsummen zu summieren. Dafür wird ein Kommunikationsmedium gebraucht.

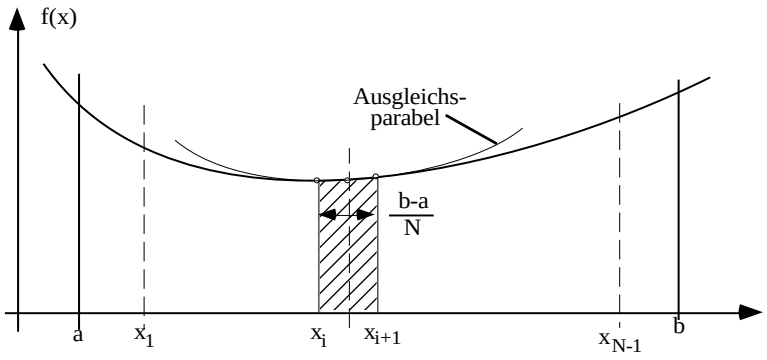
1.3.4. Algorithmen mit Datenaustausch zwischen Nachbarn

y(i) := f(g(x(i-1)), g(x(i)), g(x(i+1))) i = 2 ,…, N-1 oder zyklisch mit N

Hier ist ein Verbindungsnetzwerk erforderlich, das den parallelen Datenaustausch ermöglicht. Für alle i wird parallel g(x(i)) gerechnet und dann von den Nachbarn übernommen. Das ist besonders wichtig bei Algorithmen mit Iterationen.



Beispiele: numerische Integration nach der Simpson-Formel,



$$\int_a^b f(x)dx \approx \frac{(b-a)}{6N} \sum_{i=1}^{N} \{ f(a + \frac{(b-a)}{N}(i-1)) + 4f(a + \frac{(b-a)}{N}(i- \frac{1}{2})) + f(a + \frac{(b-a)}{N} i)\}$$

numerische Differentiation, Finite Elemente-Methoden.

1.3.5. Algorithmen mit Datenaustausch mit vielen Nachbarn

Beispiel: neuronale Netze: Bei jedem Schritt ist für alle i = 1,…, N zu rechnen

$$y_i = \sum_{k \neq i} w_{i\,k}\, x_k$$

mit Gewichten w_{ik}, die i. allg. nur für einige Kombinationen von i und k ungleich Null sind. Die Verbindungen zur Datenübertragung sollten entsprechend konfigurierbar sein.

Weitere Beispiele: numerische Integration partieller Differentialgleichungen mit Mehrgitterverfahren.

1.4. Übersicht über die Rechnerentwicklung der letzten 30 Jahre

Transistoren /Chip	1970 <1000	1980 <100.000	1985 < 500.000	1990 1 Million	1995 10 Millionen	2000 1 Milliarde
Taktrate	1 MHz	10 MHz	< 50 MHz	100 Mhz	500 MHz	1GHz
Hauptspeicher	wenige kBytes	< 1 MByte	<32 MBytes	100 MByte	Gigabyte	100 GBytes
Architektur	Bitparallel (4 Bits) mikroprogram-miert sequentielle Befehlsaus-führung	Bitparallel (8 Bits) µ-programmiert Pipelines	Bitparallel (16 Bits) RISC, Pipelines, Caches,	Befehlsparallel, 32-Bit-Architek. Verzweigungs-vorhersage, Befehlsausf. außerhalb der Reihe, Multimedia	Befehlsparallel, 32- und 64-Bit-Architekturen, Superskalar, spekulative Befehlsausf.	Threadparallel, 64-Bit-Architek. Superskalar Traceprocessor, multithreaded, Multiskalar
Compile	C, Pascal p4 Datenfluß-analyse	GNU retargetable compilers, Vektorisierung, Programmier-Umgebungen	ML, Tcl/Tk, interproze-durale Analyse	C++, HPF/Fortran 90, IDL Compiler, visuelle Programmier-sprachen	Java,JVM, mobile code, proof carrying code	Web-program-mieren, Robuste Compiler-infrastruktur
Betriebs-systeme	Unix Time sharing	remote procedure calls (RPC), verteilte File-systeme	Mikrokerne (Mach, Chorus), X Windows, Name file server	Verteilte Objekte CORBA	beweglicher Code, Web-Sicherheit	globales Betriebssystem
Netzwerke	ARPAnet FTP, Telenet Ehernet, LAN	Internet, TCP/IP	Domain name system, Netzwerk Verwaltungs-protokoll	ATMs, HTTP, Mbone, MIME	Web, IPv6, Internet 2, aktive Netzwerke	Next Generation Internet, World Wide Web mit 10^9 Knoten