Effiziente externe Algorithmen

durch Simulation paralleler Algorithmen

Olaf Bonorden

21. Juni 1999

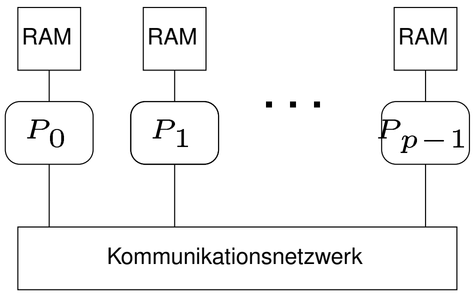
Überblick

- BSP*– und EM–BSP* Modelle und Kostenanalyse
- Prohabilistische Simulation von BSP*–Algorithmen
  - Ein–Prozessor Computer
  - Parallelrechner
- Deterministische Simulation von CGM–Algorithmen
  - Ein–Prozessor Computer
  - Parallelrechner

BSP* Modell

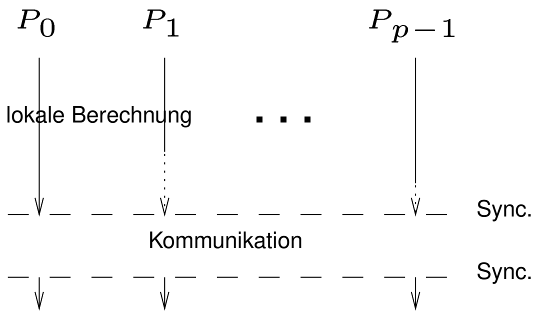
Distributed Memory Machine

- Prozessoren besitzen lokalen Speicher
- Kommunikation über ein vollständiges Netzwerk



Algorithmen

- jeder Algorithmus besteht abwechselnd aus *Berechnungs*– und *Kommunikations–Supersteps*
- nach jedem Superstep *Synchronisation* aller Prozessoren



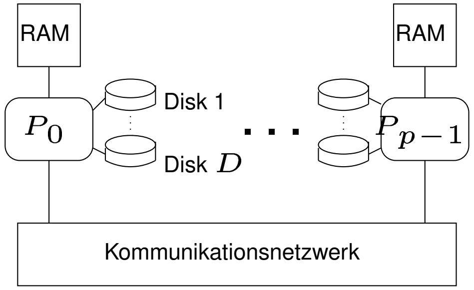
Kostenanalyse

- Charakterisierung eines Parallelrechners durch Parameter:
  - p Anzahl der Prozessoren
  - b minimale Blockgröße für effiziente Kommunikation
  - g Zeit pro übertragenem Datenblock
  - L minimale Synchronisationszeit (Latenzzeit)
- Kosten pro Superstep i:
  - Berechnung: w_i Maximum ausgeführter Operationen über alle Prozessoren
  - Kommunikation: $g\frac{h_i}{b}$, falls h_i Maximum der gesendeten/empfangenen Worte pro Prozessor,

    im BSP*–Modell zählt dabei jede Nachricht der Länge $< b$ Worte als b Worte
  - Synchronisation: L
- gesamte Laufzeit bei m Supersteps: $\sum_{i=1}^{m} w_i + \frac{g}{b} \sum_{i=1}^{m} h_i + mL$

EM–BSP* Modell

- pro Prozessor D Festplatten
- zusätzliche Parameter:
  - M Größe des lokalen Speichers
  - D Anzahl Festplatten pro Prozessor
  - B Blockgröße der Festplatten
  - G Zeit pro Transfer eines Blockes



- zusätzliche I/O–Kosten pro Superstep: $G \cdot \max_j \{w^j_{I/O}\}$ mit $w^j_{I/O}$ Anzahl I/O–Zugriffe in Prozessor j

c–Optimalität

Definition 1 *Sei* $\mathcal{A}$ *der beste sequentielle Algorithmus für ein festes Problem, sei* $T(\mathcal{A})$ *die Laufzeit im worst case. Sei* $c \geq 1$ *konstant. Ein EM–BSP* Algorithmus* $\mathcal{A}'$ *heißt* c–optimal, *falls gilt:*

- *Das Verhältnis zwischen Berechnungszeit von* $\mathcal{A}'$ *und* $T(\mathcal{A})/p$ *ist* $c + o(1)$*.*
- *Das Verhältnis zwischen Kommunikationszeit von* $\mathcal{A}'$ *und* $T(\mathcal{A})/p$ *ist* $o(1)$*.*
- *Das Verhältnis zwischen I/O–Zeit von* $\mathcal{A}'$ *und* $T(\mathcal{A})/p$ *ist* $o(1)$*.*

Datenformate

Definition 2 *Eine gegebene Menge von Daten ist in* Blockformat $:\Leftrightarrow$ *die Daten sind aufgeteilt in Einheiten der Größe mindestens B.*

Definition 3 *Daten im Blockformat sind im* aufeinanderfolgenem Format, *falls*

1. *die Anzahl der Blöcke pro Festplatte ist $\leq 1+$minimale Anzahl Blöcke pro Platte, und*
2. *die Blöcke sind in aufeinanderfolgenden Spuren auf den Festplatten gespeichert.*

$\Rightarrow$ Daten im aufeinanderfolgenden Format können schnell von allen Festplatten parallel gelesen werden

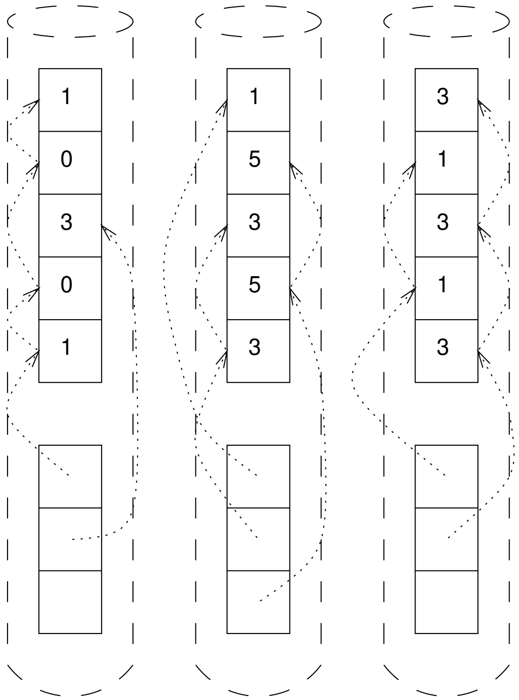
Prohabilistische Simulation mit einem Prozessor

Algorithmus SeqSuperstep

for $i = 0$ to $\frac{v}{k} - 1$ **do**
    Lese Kontexte der virtuellen Prozessoren $V_{ik}, \ldots, V_{(i+1)k-1}$
    Lese die von den k simulierten Prozessoren empfangenen Nachrichten
    Simuliere die lokalen Berechnungen der k virtuellen Prozessoren
    Schreibe die Blöcke der gesendeten Nachrichten
    Schreibe die geänderten Kontexte $V_{ik}, \ldots, V_{(i+1)k-1}$
end for
Reorganisiere die gespeicherten Nachrichten (aufeinanderfolgendes Format)

Details Schritt 5 (Schreiben der Nachrichten)

- zerlege Nachrichten in Blöcke der Größe B
- in $k\frac{\gamma}{BD}$ Runden werden jeweils D Blöcke geschrieben
- Blöcke mittels zufälliger Permutation auf die Festplatten verteilt
- Blöcke auf jeder Platte nach Zieladresse in D Körbe eingeteilt
- jeder Korb enthält Nachrichten für $\frac{v}{D}$ Ziele
- jeder Korb als verkettete Liste gespeichert

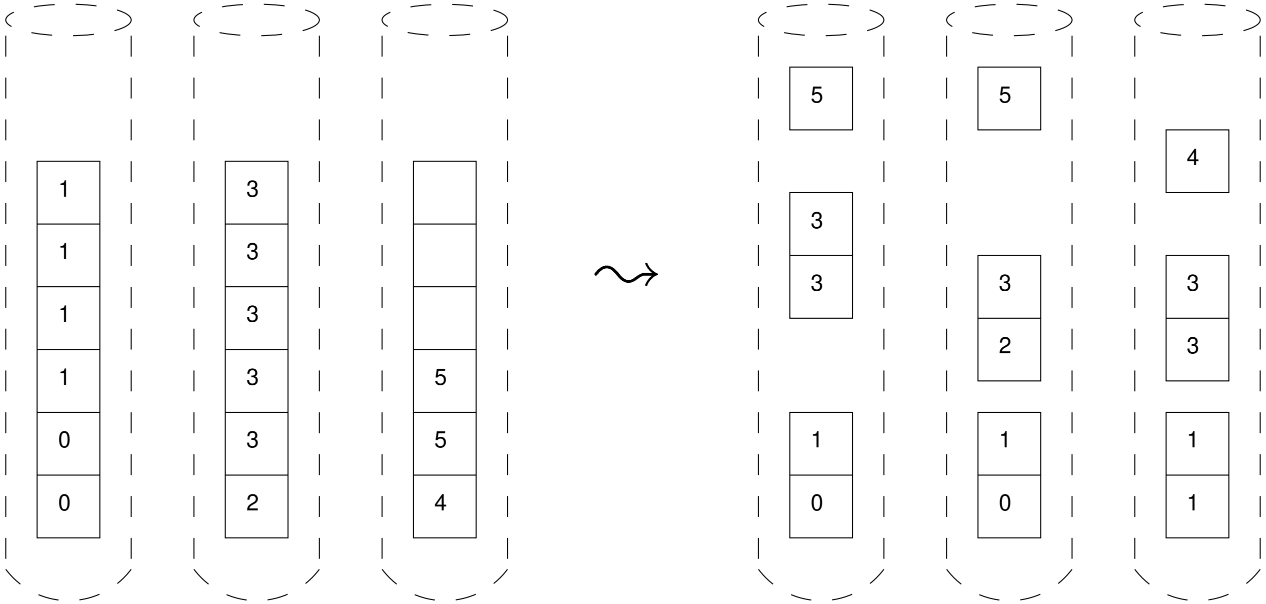


Beispiel: $v = 6$, $D = 3$ Körbe

Details Schritt 8, Reorganisation

(Algorithmus SimulateRouting)

1. Kopieren von Korb i auf Festplatte i, dazu parallel für $d = 0, \ldots, D-1$:
   - lese den j–ten Block aus Korb d von Festplatte $j + d \bmod D$
   - schreibe Block auf Festplatte d
2. Kopieren der Körbe in aufeinanderfolgendes Format



Ergebnis

Satz 1 *Ein c–optimaler v–Prozessor BSP* Algorithmus mit Berechnungszeit $\beta + \lambda L$, Kommunikationszeit $g\alpha/b + \lambda L$ und lokaler Speichergröße μ kann auf einem 1–Prozessor EM–BSP* Computer mit Berechnungszeit $(1 + o(1))v\beta$ und I/O–Zeit $O(G\lambda\frac{lv\mu}{DB})$ mit Wahrscheinlichkeit $1 - e^{-\Omega(l \log l \cdot \log(M/B))}$ für eine Konstante $l \geq 1$, $\beta = \omega(\lambda\mu)$, $v \geq kD \log \frac{M}{B}$, $M = \Theta(k\mu)$, $b \geq B$ und k eine beliebige ganze Zahl, simuliert werden.*

Insbesondere gilt: Die Simulation eines c–optimalen BSP Algorithmus führt zu einem c–optimalen EM–BSP* Algorithmus.*

Prohabilistische Simulation mit p Prozessoren

Idee: Analog sequentieller Situation, jeder reale Prozessor simuliert $\frac{v}{p}$ virtuelle

Problem: Kommunikation zwischen realen Prozessoren eventl. unabalanciert

Lösung: Zufällige Zwischenziele

Algorithmus ParSuperstep

for $j = 1$ to $\frac{v}{pk}$ **do**
    for $i = 0$ to $p - 1$ **in parallel do**
        Lese Blöcke für Stapel j und sende sie an die richtigen Zielprozessoren
        Simuliere die Berechnungen der virt. Prozessoren $i\frac{v}{p}, \ldots, (i+1)\frac{v}{p} - 1$
        Sende alle Nachrichten in Blöcken der Größe B zu zufälligen Prozessoren
    end for
end for

Deterministische Simulation von CGM–Algorithmen

Definition 4 *Ein* CGM–Algorithmus *ist ein BSP–Algorithmus mit der Eigenschaft, daß in jedem Kommunikations–Superstep jeder Prozessor $O\left(\frac{N}{V}\right)$ Worte senden und empfängt (h–Relation mit $h = O\left(\frac{N}{V}\right)$).*

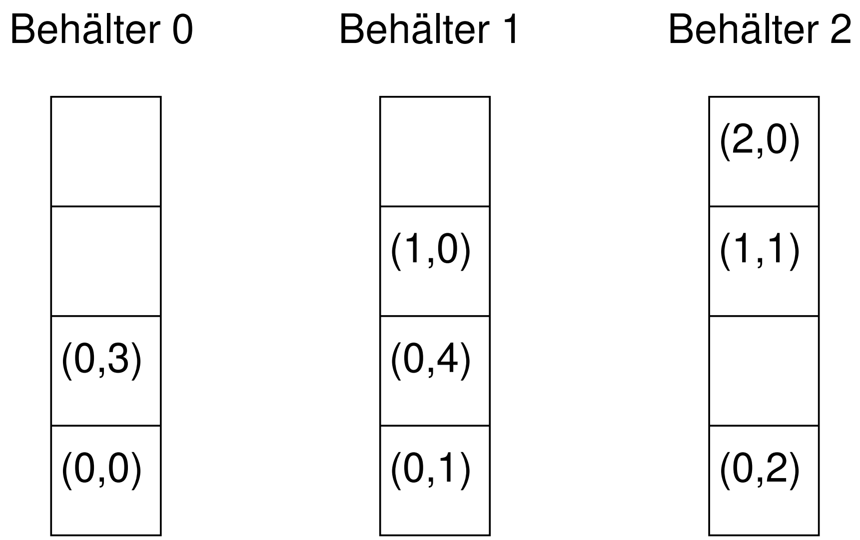
Problem: Obwohl die Kommunikationsmenge begrenzt ist, können die Größen einzelner Nachrichten stark varieren.

Lösung: Algorithmus BalancedRouting ersetzt Kommunikations–Superstep durch zwei balancierte Runden

Algorithmus BalancedRouting

(v Prozessoren)

1. Sende Wort l der Nachricht von Prozessor i an j an Prozessor $i + j + l \bmod v$
2. Sende Nachrichtenteile an richtige Ziele



$(i, j) \widehat{=}$ Wort j der Nachricht an Prozessor i

Algorithmus DetSeqSuperstep

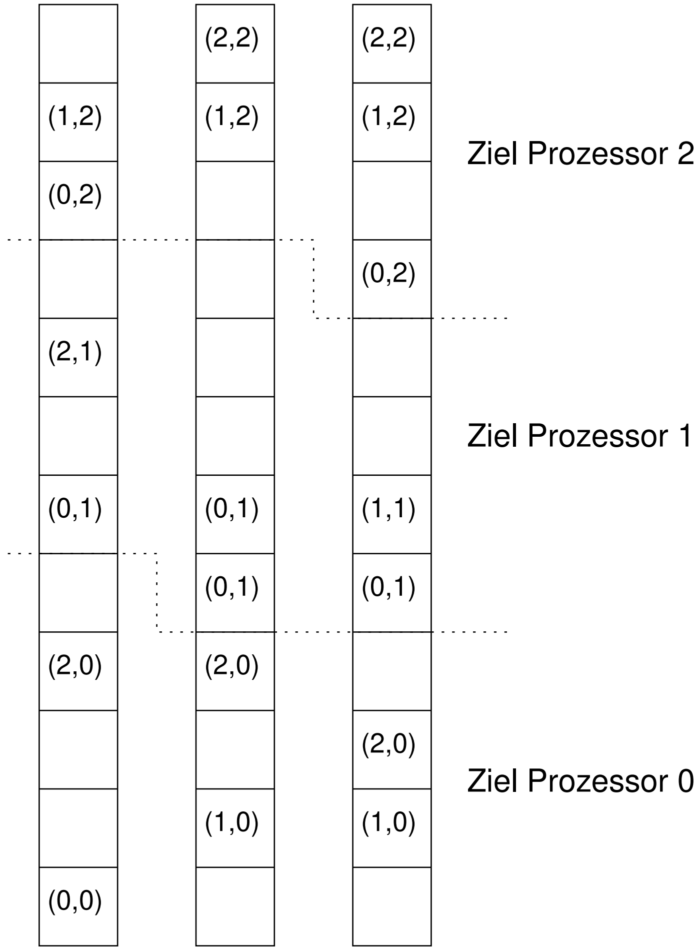
for $i = 0$ to $v - 1$ **do**
    Lese Kontext des virtuellen Prozessors i in den Speicher
    Lese empfangene Blöcke für Prozessor i von den Festplatte
    Simuliere lokale Berechnungen von Prozessor i
    Schreibe gesendete Nachrichten in gestaffeltem Format
    Schreibe geänderten Kontext auf die Festplatten
end for

Gestaffeltes Format

- sei b' die maximale Anzahl Blöcke pro Nachricht
- Nachrichten an Prozessor j beginnen auf Platte $b_j = jb' \bmod D$

Größe der Nachricht von i an j:

	$j=0$	$j=1$	$j=2$
$i=0$	1	4	2
$i=1$	2	1	3
$i=2$	3	1	2



Ergebnis

Satz 2 *Ein v–Prozessor–CGM–Algorithmus $\mathcal{A}$ mit λ Supersteps, lokaler Speichergröße μ, Laufzeit $\beta + g \cdot O(\frac{N}{v}) + \lambda L$ und Nachrichtengröße $\Theta(\frac{N}{v^2})$ kann auf einer Ein–Prozessor EM-CGM mit Zeit $v\beta + O(\lambda v \mu) + G\lambda \cdot O(\frac{v\mu}{DB})$ für $M = \Theta(\mu)$, $B = O(\frac{N}{v^2})$ und $N = \Omega(vDB)$ simuliert werden.*

Insbesondere führt die Simulation zu einem c–optimalen EM–CGM Algorithmus, falls $\mathcal{A}$ c–optimal, $\beta = \omega(\lambda\mu)$ und $G = DB \cdot o\left(\frac{\beta}{\lambda\mu}\right)$.

Deterministische Simulation mit p Prozessoren

1: **for** $i = 0$ to $p - 1$ **in parallel do**

2:     **for** $j = 0$ to $\frac{v}{p} - 1$ **do**

3:         Lese Kontext des virtuellen Prozessors $\frac{v}{p}i + j$ in den Speicher

4:         Lese empfangene Blöcke für Prozessor $\frac{v}{p}i + j$ von den Festplatte

5:         Simuliere lokale Berechnungen von Prozessor $\frac{v}{p}i + j$

6:         Sende alle Nachrichten zu den Prozessoren, auf denen die virtuellen Zielprozessoren simuliert werden

7:         Schreibe in Zeile 6 empfangene Nachrichten in gestaffeltem Format auf die Festplatten, analog Algorithmus DetSeqSuperstep

8:         Schreibe geänderten Kontext auf die Festplatten

9:     **end for**

10: **end for**

Zusammenfassung

- Prohabilistische Simulation von BSP*–Algorithmen ([DDH97])
- Deterministische Simulation von CGM–Algorithmen ([DHMD99])

$\leadsto$ effiziente EM–Algorithmen für Ein– und Multiprozessorcomputer

Literatur

[DDH97] DEHNE, FRANK, WOLFGANG DITTRICH and DAVID HUTCHINSON: *Efficient External Memory Algorithms by Simulating Coarse-Grained Parallel Algorithms*. In *Annual ACM Symposium on Parallel Algorithms and Architecture*. ACM, 1997.

[DHMD99] DEHNE, FRANK, DAVID HUTCHINSON, ANIL MAHESHWARI and WOLGANG DITTRICH: *Reducing I/O Complexity by Simulating Coarse Grained Parallel Algorithms*. In *Annual ACM Symposium on Parallel Algorithms and Architecture*. ACM, 1999.