Prof. Andreas Goerdt
Andreas Berger

TU Chemnitz
Wintersemester 2002/03

Parallele Algorithmen
Nachtrag zur 1. Übung

Präfix-Summen

Wir betrachten das Präfix-Summen-Problem. Sei $\odot$ ein assoziativer Operator. Zu einer gegebenen Folge von n Elementen a_1, a_2 ... a_n ist die Folge der „Präfix-Summen" s_1, s_2 ... s_n zu berechnen:

$$s_i = \bigodot_{k=1}^{i} s_k = a_1 \odot a_s \odot \ldots \odot a_i.$$

Wir suchen einen (arbeits-)optimalen parallelen Algorithmus. Es ist klar, dass für die Berechnung dieser Summen eine Arbeit von $\Omega(n)$ zu bewältigen ist. Dabei zählen wir für die Anwendung eines $\odot$-Operators Zeit bzw. Arbeit $\Theta(1)$.
Eine Klammerung des Ausdrucks für die Summen ergibt unter Ausnutzung der Assoziativität für gerade i:

$$\begin{aligned} s_i &= (a_1 \odot a_2) \odot (a_2 \odot a_4) \odot \ldots \odot (a_{i/2-1} \odot a_{i/2}) \\ &= \bigodot_{k=1}^{\frac{i}{2}} (a_{2k-1} \odot a_{2k}). \end{aligned}$$

Entsprechend für ungerade i

$$s_i = \left(\bigodot_{k=1}^{\lfloor \frac{i}{2} \rfloor} (a_{2k-1} \odot a_{2k}) \right) \odot a_i.$$

Setzen wir also für $1 \leq i \leq \left\lfloor \frac{n}{2} \right\rfloor$

$$a'_i = a_{2i-1} \odot a_{2i}$$

und s'_i die entsprechenden Präfix-Summen, dann können wir die gesuchten Summen folgendermaßen darstellen

$$s_i = \begin{cases} s'_{i/2} & \text{, falls } i \text{ gerade} \\ s'_{\lfloor i/2 \rfloor} + a_i & \text{, falls } i \text{ ungerade.} \end{cases}$$

Zu Folge $(a)_i$ lässt sich $(a')_i$ mit Arbeit $\Theta(n)$ in Zeit $\Theta(1)$ ebenso berechnen wie die Lösung $(s)_i$ aus $(s')_i$. Schließlich ist für $n = 1$ trivialerweise $s_1 = a_1$ Lösung für das Präfix-Summen-Problem.
Wir können die Präfix-Summen daher mit folgendem rekursiven Algorithmus berechnen:

> Eingabe: die Folge a_i für $1 \leq i \leq n$ und die Anzahl n
> Ausgabe: die Folge s_i der Präfix-Summen
> 1. $s_1 := a_1$
> $n = 1$? ja: fertig, Ende
> 2. for $i = 1$ to $\left\lfloor \frac{n}{2} \right\rfloor$ pardo

$a'_i := a_{2i-1} + a_{2i}$

3. `berechne rekursiv die Präfix-Summen für` a'_i `in` s'_i
4. `for` $i = 2$ `to` n `pardo`

$$s_i := \begin{cases} s'_{i/2} & , falls\ i\ gerade \\ s'_{(i-1)/2} + a_i & , falls\ i\ ungerade \end{cases}$$

- Die Korrektheit ergibt sich induktiv aus den obigen Betrachtungen.
- Parallele Zeit:
  - pro Rekursion $\mathcal{O}(1)$,
  - Anzahl der Rekursionen: $\lfloor \log n \rfloor + 1$
    daher insgesamt $T^{\infty}(n) = \mathcal{O}(\log n)$
- Arbeit:
  - $W(n) = W\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \mathcal{O}(n) = \mathcal{O}(n + \frac{n}{2} + \frac{n}{4} + \ldots) = \mathcal{O}(n)$
- Der Algorithmus kann auf einer EREW-PRAM realisiert werden.

Der angegebene Algorithmus ist damit optimal. Mit p Prozessoren können die Präfix-Summen also in paralleler Zeit $\mathcal{O}(\frac{n}{p} + \log n)$ berechnet werden.

Eine Anwendung: Addition großer Zahlen

Die Verwendung des uniformen Kostenmaßes ist für Operationen wie Addition oder Multiplikation nur dann gerechtfertigt, wenn wir uns auf einen festen Zahlenbereich beschränken. Anders sieht es bei Problemen aus, für die sich die Berechnungskomplexität gerade durch die Verwendung großer Zahlen ergibt.
Wir betrachten die Addition zweier n-Bit Zahlen. Für einen sequentiellen Algorithmus stellt $\Omega(n)$ sicherlich eine untere Schranke für die benötigte Zeit dar. Andererseits liefert bereits die „Schulmethode“, wobei die Zahlen entsprechend ihrer binären Darstellung beginnend von der kleinstwertigen Position an unter Berücksichtigung des Übertrags (im folgenden auch carry) addiert werden, einen $\mathcal{O}(n)$-Algorithmus. Allerdings haftet dieser Methode der Nachteil der Sequentialität an. Zur Bestimmung der i-ten Stelle des Ergebnisses wird das Carry der jeweils vorangehenden Stelle benötigt.
Seien nun a und b die zu addierenden Zahlen mit den Binärdarstellungen $a = a_0 + 2a_1 + 4a_2 + ... + 2^{n-1}a_{n-1}$ und $b = b_0 + ...$ Wir bezeichnen mit c_i das einlaufende Carry in die Stelle i. Könnten wir c_i schneller bestimmen, wäre auch das Problem der Addition schneller gelöst.
Blicken wir etwas „intuitiver“ auf die Entstehung eines Übertrags, so zeigt sich: ein Übertrag existiert in Stelle i,

- falls $a_{i-1} = b_{i-i} = 1$ oder
- falls $c_{i-1} = 1$ und $a_{i-1} = 1$ oder $b_{i-1} = 1$.

Setzen wir nun $g_i = a_i \wedge b_i$ (wir verwenden boolesche Logik mit $0 = FALSE$ und $1 = TRUE$) und $p_i = a_i \vee b_i$, so ergibt sich für Carry c_i :

$$c_i = g_{i-1} \vee (p_{i-1} \wedge c_{i-1}). \tag{1}$$

(Ein Carry tritt auf, falls es in der vorangegangenen Stelle generiert wird (g) oder ein vorhandenes Carry weitergereicht (p wie progariert) wird.)
Setzen wir die Gleichung für c_{i-1} ein, so erhalten wir mit

$$c_i = \underbrace{g_{i-1} \vee (g_{i-2} \wedge p_{i-1})}_{g'} \vee (\underbrace{p_{i-1} \wedge p_{i-2}}_{p'} \wedge c_{i-2})$$

eine Gleichung zur Berechnung des Carrys über zwei Positionen hinweg. Wiederum stellen g' und p' die notwendigen Informationen bereit. Wir definieren nun einen Operator $\oplus$:

$$(g_1, p_1) \oplus (g_2, p_2) = (g_2 \vee (g_1 \wedge p_2)\, ,\, p_2 \wedge p_1).$$

Dieser ist offensichtlich assoziativ. Wegen (1) und $c_0 = 0$ gilt dann für $(g', p') = (g_0, p_0) \oplus (g_1, p_1) \oplus ... \oplus (g_{i-1}, p_{i-1})$

$$c_i = g'.$$

(Formaler Beweis: Induktion!)
Also können die Überträge mit dem Präfix-Summen-Algorithmus in paralleler Zeit $\mathcal{O}(\log n)$ bestimmt werden und eine Addition ist in der selben Zeit möglich.
Es gilt daher:
Zwei natürliche Zahlen a und b können auf einer EREW-PRAM in Zeit $\mathcal{O}(\log\log(a+b))$ mit einer Arbeit von $\mathcal{O}(\log(a+b))$ addiert werden.
Zwei n -Bit-Zahlen können damit auch von einem Schaltkreis aus $\mathcal{O}(n)$ Gattern mit einer Schaltkreistiefe $\mathcal{O}(\log n)$ addiert werden.
zusätzliche Aufgabe:
Man zeige, dass zwei n-Bit Zahlen in paralleler Zeit $\mathcal{O}(\log n)$ multipliziert werden können!