

Übungen zur Vorlesung
Paralleles Höchstleistungsrechnen
Dr. S. Lang, D. Popović

Abgabe: 22. Dezember 2009 in der Übung

Übung 23 Kommunikationszeiten mit Cut-Through-Routing (4 Punkte)

In der Vorlesung haben Sie die Kommunikationszeiten T für verschiedene Basis-Operationen (One-to-All, ...) auf verschiedenen Verbindungstopologien für *Store-And-Forward*- und *Cut-Through-Routing* kennengelernt. Die Analyse ist sehr wichtig, weshalb wir hier die essentiellen Punkte noch einmal wiederholen wollen.

- (a) Stellen Sie eine Übersichtstabelle der Kommunikationszeiten für folgende Kommunikations-Operationen zusammen:
1. One-to-all Broadcast
 2. All-to-all Broadcast
 3. One-to-all mit individuellen Nachrichten
 4. All-to-all mit individuellen Nachrichten

Betrachten Sie nur das *Cut-Through-Routing* und geben Sie die Zeiten T für Kommunikation auf dem Ring, $2D$ -Feld (ohne Rückverbindungen) und Hypercube an. Es wurden nicht alle Topologien besprochen, d.h. die Tabelle kann lückenhaft sein. Oftmals ist *Cut-Through-Routing* nicht besser als *Store-and-Forward-Routing*.

- (b) Für welche Topologien gibt es Vorteile des *Cut-Through-Routing* gegenüber dem *Store-and-Forward-Routing*?

Übung 24 Isoeffizienz-Analyse: Matrix-Vektor-Multiplikation (10 Punkte)

In der Vorlesung wird gezeigt, dass für eine Matrix-Vektor-Multiplikation $\mathbf{y} = A\mathbf{x}$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$, $A \in \mathbb{R}^{M \times N}$, $M, N \in \mathbb{N}$ bei einer blockweisen Aufteilung der Matrix über ein Feld (ohne Rückverbindungen) mit *Cut-Through-Routing* eine Isoeffizienz-Funktion der Komplexität $\Theta(P(ld\sqrt{P})^2)$ existiert.

Untersuchen Sie nun für $N = M$ den Fall, daß die Matrix A und der Vektor \mathbf{x} zeilenweise auf P Prozessoren, die im Hypercube angeordnet sind, verteilt sind, wie in Abbildung 0.3 angedeutet. Der Vektor \mathbf{y} habe die gleiche Verteilung wie Vektor \mathbf{x} und verwendet werde wieder *Cut-Through-Routing*. Da \mathbf{x} auf jedem Prozessor zur Verfügung stehen muss, benötigen Sie für diesen Fall die Zeit $T_{All-to-All-Broadcast-HC}$:

$$T \approx t_s \log P + t_w m(P - 1)$$

für Nachrichten der Länge m .

- (a) Wie ist nun die parallele Ausführungszeit bestehend aus Rechen- und Kommunikations-Zeit (Die Kommunikation besteht nur in dem *All-to-All-Broadcast-HC*)?
- (b) Von welcher Komplexität ist für diese Lösung die Iso-Effizienzfunktion (Hinweis: Sie können die Terme von $T_o = pT_p - W$ getrennt betrachten, der algorithmisch komplexere „gewinnt“)? Welches ist der dominierende Term?

Übung 25 Superlinearer Speed-Up bei der Tiefensuche (6 Punkte)

Eine Möglichkeit, einen n -ären Baum zu Durchsuchen, ist die Tiefensuche (*Depth First Search*, DFS), bei der die Teilbäume zunächst bis zum tiefsten Knoten durchsucht wird, bevor zur letzten

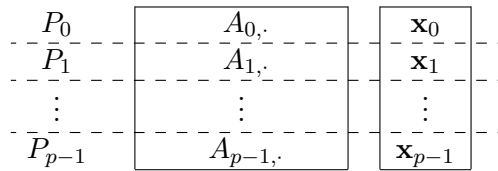


Abbildung 0.3: Zeilenweise Aufteilung einer Matrix A und eines Vektoren \mathbf{x} für das Matrix-Vektor-Produkt.

Verzweigung zurückgekehrt wird. Diese Suche kann durch Aufteilen auf Prozessoren parallelisiert werden. Für Baum-Suchalgorithmen definiert man den *Search Overhead Factor* über $\kappa = W_p/W_s$, wobei W_p die Anzahl der parallel von allen Prozessoren durchsuchten Knoten und W_s die Anzahl der sequentiell durchsuchten Knoten ist. Es sollte immer $\kappa \geq 1$ gelten, sonst ist der parallele Algorithmus mit $P = 1$ (nur ein Prozessor) schneller als der sequentielle. Das führt zu superlinearem Speed-Up.

- (a) Betrachten Sie Abbildung 0.4, links. Geben Sie die Anzahl der durchsuchten Knoten bei sequentieller und paralleler Suche an, falls der Baum mittig getrennt auf 2 Prozessoren aufgeteilt ist.
- (b) Rechts in der Abbildung ist der Zielknoten ausgetauscht. Betrachten Sie wiederum die Dauer der sequentiellen und parallelen Suche (mit 2 Prozessoren). In welcher „Zeit“ (Anzahl besuchter Knoten) wird nun der Zielknoten gefunden?
- (c) Können Sie κ für beide Fälle angeben? Kann es zur hier zu „Superlinearität“ kommen, und können Sie den Fall skizzieren, in der sie auftritt?
- (d) Woran liegt es, das bei diesem Problem superlinearer Speed-Up auftreten kann? Was muss für den sequentiellen Algorithmus verändert werden, damit der Speed-Up nicht superlinear wird (Hinweis: Haben wir den bestmöglichen sequentiellen Algorithmus betrachtet?)?

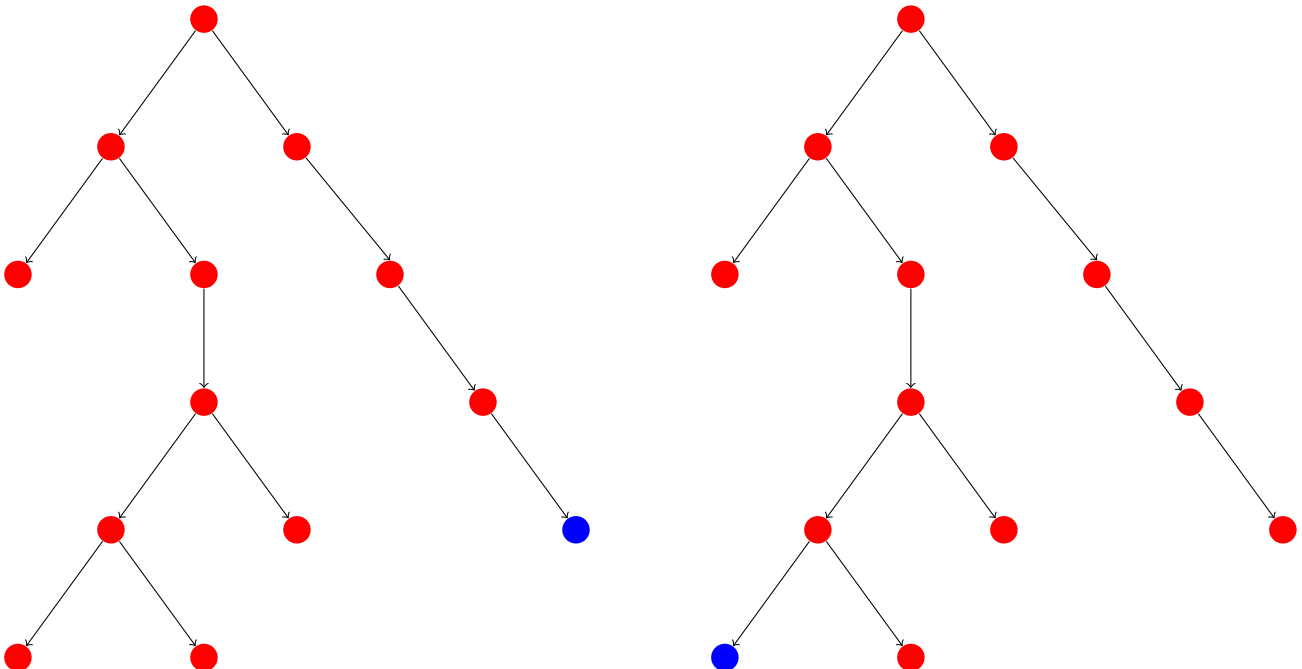


Abbildung 0.4: Suchbäume für die sequentielle oder parallele Depth First Search. Der Zielknoten der jeweiligen Suchen ist blau markiert.