
Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Technische Informatik
Prof. Dr. rer. nat. Wolfgang Karl

Zentralübung Rechnerstrukturen: Parallelismus, Parallele Programmierung und Verbindungsstrukturen

Aufgabenblatt 3 – Musterlösung

1 Leistungsbewertung

1.1

- $E(n) = \frac{S(n)}{n} \Rightarrow E(16) = \frac{S(16)}{16} = \frac{8}{16} = 0,5$
  $S(n) = \frac{T(1)}{T(n)} \Rightarrow T(16) = \frac{T(1)}{S(16)} = \frac{80}{8} = 10$
  $I(16) = \frac{P(n)}{T(n)} \Rightarrow I(16) = \frac{P(16)}{T(16)} = \frac{16}{10} = 1,6$

  Hier ist uns ein logischer Fehler in der Aufgabenstellung unterlaufen. Bitte beachten Sie die Anmerkungen in den (korrigierten) Folien zur Übung!

- Amdahls Gesetz:
  $T(n) = T(1) * \left(\frac{(1-a)}{n} + a\right)$
  $T(n) = T(1) * \frac{1-a+na}{n}$
  $\Rightarrow \quad 10 = 80 * \frac{1-a+16a}{16} = 80 * \frac{1+15a}{16} = 5 * (1+15a)$
  $\Rightarrow \quad 15a = 1 \quad \Rightarrow \quad a = \frac{1}{15} \approx 6,7\,\%$ des Programmcodes sind nur sequentiell ausführbar.

1.2

- Parallele Ausführungszeit von P Prozessoren: $T_{par} = 20\,\% + \frac{80\,\%}{P} * T_{seq}$
  Beschleunigung: $S = \frac{T_{seq}}{T_{par}}$
  Effizienz: $E = \frac{S}{P}$

	$P=2$	$P=4$	$P=8$	$P=16$	$P=32$
Beschleunigung	1,67	2,5	3,33	4	4,44
Effizienz	0,83	0,625	0,42	0,25	0,14

Die Skalierbarkeit ist sehr schlecht. Grund dafür ist, daß ein großer Anteil vom Code nicht parallelisierbar ist. Die Ausführungszeit von diesem Anteil bestimmt einen zunehmenden Anteil der gesamten Ausführungszeit mit steigender Anzahl von Prozessoren.

Beschleunigung $S = \frac{T}{\left(20\% + \frac{80\%}{P}\right) * T} \xrightarrow{\text{P sehr groß}} \frac{1}{20\%} = 5$

- Beschleunigung $S = \frac{T}{\left(20\% + \frac{80\%}{64} + 1\% * 64\right) * T} = 1,17$
  Effizienz $E = \frac{1,17}{64} = 0,018$

1.3

- Betrachten wir zunächst den einfacheren Fall, d.h. die Installation eines zweiten Prozessors. Dies macht das vorhandene System zu einem speichergekoppelten System vom Typ „UMA“, da beide Prozessoren auf den gleichen in dem System installierten Speicher zugreifen.

  Aufgrund der Angaben gilt somit $n = 2$ und $a = 1 - 0,25 = 0,75$. Eingesetzt in die Formel für die Beschleunigung $S(n)$ unter Brüchsichtigung von Amdahls Gesetz ergibt sich folglich:
  $S(n) = \frac{T(1)}{T(n)} = \frac{T(1)}{T(1) * \left(\frac{1-a}{n} + a\right)} = \frac{n}{1 - a + n * a}$
  $S(n) = \frac{2}{0,25 + 2 * 0,75} = \frac{2}{1,75} \approx 1,14$

- Für das zweite – strikt sequentielle – System gilt, daß kein Parallelisierungaufwand $R(n)$ vorliegt, d.h. $R(n) = 1$. Dies ist auch für das optimale parallele System der Fall, für welches $\frac{T(1)}{n} = T_{opt}$ die Laufzeit darstellt. Sei nun α der Anteil des Programms, der nicht optimiert werden kann (d.h. die Entsprechung von a), so läßt sich die neue Ausführungszeit T_{new} als die von α abhängige gewichtete Summe der optimierten und alten Ausführungszeit beschreiben:

  $T_{new} = (1 - \alpha) * T_{opt} + \alpha * T_{old}$

  Weiterhin gilt aufgrund der Angaben für die optimierbare Programmteile $T_{old} = T_{opt} * 2$ bzw. $T_{opt} = \frac{1}{2} * T_{old}$, d.h. für die Beschleunigung ergibt sich:

  $S' = \frac{T_{old}}{T_{new}} = \frac{T_{old}}{(1-\alpha) * T_{opt} + \alpha * T_{old}} = \frac{T_{old}}{\frac{(1-\alpha)}{2} * T_{old} + \alpha * T_{old}} = \frac{2}{(1-\alpha) + 2 * \alpha}$

  Somit folgt aus $\alpha = 1 - 0,3 = 0,7$ unmittelbar $S' = \frac{2}{0,3 + 2 * 0,7} \approx 1,18 > S(2)$

Aufgrund der Ausführungszeiten ist also die Koprozessorlösung dem SMP-System für diese Anwendung vorzuziehen.

2 Verbindungsstrukturen

2.1 Statische Verbindungsstrukturen

a) Der Verbindungsgrad eines Knotens ist definiert als die Anzahl der Verbindungen, die von dem Knoten zu anderen Knoten bestehen. Gemäß Aufgabenstellung in diesem Fall also 4.

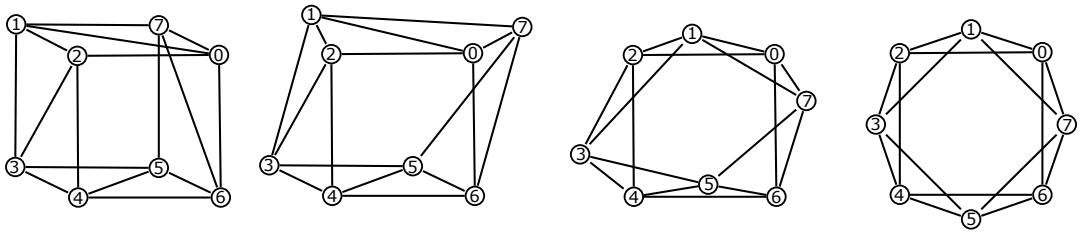
Der Diameter oder Durchmesser ist die maximale Distanz zwischen zwei Knoten, hier in der Aufgabe 2.

Schneidet man einen Graphen in zwei gleich große Teile und betrachtet die Menge der Kanten, die diesen Schnitt kreuzen, so bezeichnet man die Kardinalität der kleinsten dieser Kantenmengen über alle möglichen Schnitte als die minimale Bisektionsbreite. Die minimale Bisektionsbreite ist hier 6.

Die Diskonnektivität ist definiert als $\frac{\#Knoten}{minimale\ Bisektionsbreite}$, d.h. hier beträgt sie $\frac{8}{6} \approx 1,33$.

Die Kosteneffektivität ist definiert als:
$Verbindungsgrad * \max(Durchmesser,\ Diskonnektivität)$
und ergibt sich somit zu $4 * \max\left(2, \frac{4}{3}\right) = 8$

b) Es handelt sich hier um einen chordalen Ring.



c) Es liegt Redundanz vor. Da der Verbindungsgrad jedes Knotens 4 ist und bidirektionale Leitungen verwendet werden, können bis zu drei Leitungen ausfallen und dennoch jeder Knoten von einem anderen erreicht werden. Allerdings kann beim Ausfall einer Kante der Durchmesser steigen, das heißt es könnten längere Wege notewendig sein.

d)

	Aufgabe 2	Ring	2D-Gitter	Baum	Hyperkubus
Verbindungsgrad	4	2	$2-4$	$1-3$	$\log_2 N$
Durchmesser	$\lfloor\sqrt{N}\rfloor$	$\lfloor N/2\rfloor$	$2\left(\sqrt{N}-1\right)$	$2\left(\lceil\log_2 N\rceil - 1\right)$	$\log_2 N$
min. Bisektionsbreite	6	2	$\sqrt{N}$	1	$N/2$
Diskonnektivität	$N/6$	$N/2$	$\sqrt{N}$	N	2
Kosteneffektivität	$4\lfloor\sqrt{N}\rfloor$	N	$8\left(\sqrt{N}-1\right)$	$3N$	$(\log_2 N)^2$

2.2 Dynamische Verbindungsstrukturen

a) Ja.

b) Nein. Beweis durch Widerspruch.

Annahme: jede Permutation kann generiert werden.
Gesucht: mindestens eine Permutation, für die die Annahme nicht gilt.

- Bei einer paarweisen Mischpermutation (Kreisverschiebung), hier also Verbindung von P0 und P1 mit M6 bzw. M7 gibt es nur einen möglichen Verbindungsweg, der gleichzeitig für beide Verbindungen benutzt werden müßte
  $\Rightarrow$ Blockierung
  Eine weitere Blockierung tritt bei Permutationen mit P6 $\rightarrow$ M4 und P7 $\rightarrow$ M5 auf.
- Bei einer Kreuzpermutation (Randpaare vertauschen), tritt das gleiche Problem wie bei der Mischpermutation auf.

c) Schon bei zwei Verbindungen kann eine Blockierung auftreten: z.B. bei P0 $\rightarrow$ M0 und P1 $\rightarrow$ M1

d) Nein. Auf der einen Seite gibt es für bestimmte Paare mehre Möglichkeiten (vgl. P2 $\rightarrow$ M4), aber ebenso gibt es Paare, bei denen schon der Ausfall einer Verbindung die Weiterleitung ausschließt (z.B. P0 $\rightarrow$ M6).

3 Parallele Programmierung

a) SMP-Knoten mit P Prozessoren und einem gemeinsamen Speicher

Sequentielle Zeit $T(1)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{n^2}_{\text{Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 7n^2$$

Parallele Zeit $T(P)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{\frac{n^2}{P}}_{\text{par. Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 6n^2 + \frac{n^2}{P}$$

Beschleunigung $S(P)$:

$$S(P) = \frac{T(1)}{T(P)} = \frac{7n^2}{6n^2 + \frac{n^2}{P}} = \frac{7}{6 + \frac{1}{P}} < \frac{7}{6} \approx 1,17$$

Das Problem ist hierbei, daß der Speicher als limitierender Faktor wirkt und damit die Beschleunigung stark beschränkt ist.

b) NUMA-Architektur mit P Prozessoren und P lokalen, aber gemeinsamen Speichern

Sequentielle Zeit $T(1)$:

$$\underbrace{5n^2}_{\text{Daten aus Speicher holen}} + \underbrace{n^2}_{\text{Berechnung}} + \underbrace{n^2}_{\text{Zurückschreiben}} = 7n^2$$

Parallele Zeit $T(P)$:

$$\underbrace{\underbrace{\frac{4*2*n^2}{P}}_{\substack{\text{4 Werte der Nachbarknoten aus entferntem Speicher}}} + \underbrace{\frac{n^2}{P}}_{\substack{\text{eigener Wert im lokalen Speicher}}}}_{\text{Daten aus Speicher holen}} + \underbrace{\frac{n^2}{P}}_{\substack{\text{parallele Berechnung}}} + \underbrace{\frac{n^2}{P}}_{\substack{\text{Zurückschreiben in lokalen Speicher}}} = \frac{11n^2}{P}$$

Beschleunigung $S(P)$:

$$S(P) = \frac{T(1)}{T(P)} = \frac{7n^2}{\frac{11n^2}{P}} = \frac{7}{11}P$$

$$\text{für } P = 2: \quad S(2) = \frac{14}{11} \approx 1,27$$

$$\text{für } P = 3: \quad S(3) = \frac{21}{11} \approx 1,90$$

$\Rightarrow$ Die Beschleunigung skaliert mit $\frac{7}{11}$ der Problemgröße (linear).

4 Verständnisfragen

a) Konstanter Verbindungsgrad, einfache Erweiterbarkeit, einfaches Routing, hohe Fehlertoleranz,...

b) Einfacher Aufbau, einfaches Routing, einfaches Adressierungsschema,...

c) Shared-Memory Programmiermodell:
   - Vorteile: Niedrige Latenzen, hohe Kommunikationsbandbreite
   - Nachteile: Skalierbarkeit

d) Message-Passing Programmiermodell:
   - Vorteile: Skalierbarkeit
   - Nachteile: im Vergleich zu Shared-Memory hohe Latenzen, komplexe Programmierung über gezielte Nachrichten