Universität Heidelberg Wintersemester 2008/09

Aufgabenblatt 5 24. 11. 2008

Übungen zur Vorlesung

"Paralleles Höchstleisungsrechnen "

Prof. Dr. P. Bastian, Ch. Engwer

Abgabe bis 2. 12. 2008 an christian.engwer@iwr.uni-heidelberg.de

---

ÜBUNG 1 SPEEDUP

In der Vorlesung wurde das N-Körper-Problem besprochen. Es wurden ausserdem verschiedene Optimierungen, sowie die Threadparallelisierung gezeigt.

Eine wichtige Optimierung ist, dass man nicht alle Kräfte ausrechnen muss, sondern die Symetrie der Kräfte berücksichtigt, die Kraft, die Körper A auf Körper B auswirkt, wirkt mit umgekehrtem Vorzeichen von Körper B auf Körper A. Daher ist es ausreichend eine obere Dreiecksmatrix aufzustellen, die linke untere Matrix kann dann nachträglich berechnet werden.

Betrachten wir nun den parallel Fall:

Wir habe P Prozessoren, und $N = k \cdot P$ Körper. Wir müssen das obere Dreieck einer $N \times N$ Matrix berechnen. Hierzu werden die Zeilen der Matrix zyklisch auf die Prozessoren aufgeteilt. Durch die Dreiecksstruktur ist der Rechenaufwand pro Zeile aber nicht konstant, das hat eine Ungleichverteilung des Rechenaufwandes zur Folge.

1. Was ist der Rechenaufwand im sequentiellen Fall ($P = 1$).
2. Welches Rechenaufwand ergibt sich für den Prozessor mit dem geringsten Aufwand, welcher für den mit dem größten Aufwand?
3. Quantifizieren Sie die "imbalance".
4. Berechnen Sie den Speedup der durch die Parallelisierung mit P Prozessoren erreicht wird.

ÜBUNG 2 PARALLELE BERECHNUNG VON π

Hinweis: Für die Programmieraufgabe haben Sie 2 Wochen Zeit. Senden Sie bei der Abgabe ebenfalls den Sourcecode ein.

π kann durch Integration des Kreisflächeninhalts

$$\pi r^2 = \int\limits_{\mathbb{R}^2} f(x)\,dx \qquad \text{mit} \quad f(x) = \begin{cases} 1 & |x| < r \\ 0 & \text{sonst} \end{cases}$$

bestimmt werden.

Zur Vereinfach wählen wir $r = 1$ und nutzen zusätzlich die Symetrie aus, so dass man nur über ein Viertel der Kreises integrieren muss. Es muss dann auch nicht mehr über den ganzen rechten oberen Quadranten von $\mathbb{R}^2$ integriert werden, sondern nur noch über das Einheitsquadrat.

$$\pi = 4 \cdot \int\limits_{\mathbb{R}^2, |x|>0,} f(x)\,dx \qquad \text{mit} \quad f(x) = \begin{cases} 1 & |x| < 1 \\ 0 & \text{sonst} \end{cases}$$

Um das Integral numerisch auszuwerten wird das Einheitsquadrat in kleinere Quadrate zerlegt. Beim Integrieren wird dann über die Flächen all der Quadrate summiert, deren Schwerpunkt im Inneren des Einheitskreises liegt.

Um eine geforderte Genauigkeit des Ergebnisses zu gewährleisten muss über entsprechend kleine Teilquadrate summiert werden. Dies würde aber zu einem Quadratischen Anstieg des Rechenaufwandes führen. Um das zu vermeiden soll nur um den Rand herum verfeinert werden, dazu verwenden wir folgende rekursiver Algorithmus:

- Wir beginnen mit dem Einheitsquadrat.
- Wenn ein Quadrat teilweise innerhalt, teilweise ausserhalb des Kreises liegt wird es in vier Unterquadrate verfeinert.

Die Rekursion wird beendet, wenn man oft genug verfeinert hat, um die gewünschte Genauigkeit zu erhalten.

1. Überlegen Sie sich eine Formel, die Ihnen eine Abschätzung des Fehlers bei einer gewissen Rekursionstiefe gibt. Aus dieser Abschätzung können Sie dann für eine geforderte Genauigkeit die erforderliche Rekursiontiefe bestimmen.

   Hinweis: Die Fläche des Viertelkreises können Sie durch ein Dreieck nach unten abschätzen. Den Fehler können Sie durch Summe der Fehler in den Quadraten auf dem Rand abschätzen, und die Länge des Randes kann durch die doppelte Kantenlänge nach oben abgeschätzt werden.

2. Implementieren Sie ein Programm das mit Hilfe der numerischen Integration π näherungsweise bestimmt.

   - Die Anzahl der Verfeinerungen soll das Programm automatisch bestimmen, so dass ein, vom User angegebener, maximaler relativer Fehler garantiert wird.

     Hinweis: wenn Sie Ihr Programm testen, denken Sie daran, dass Sie eine endliche Genauigkeit bei Ihren Floatingpoint Operationen haben. Die Fehler die dort gemacht werden akkumulieren sich auch wieder. Von daher sollten Sie bei Ihren Vorgaben nicht zu ambitioniert sein.
   - Verwenden Sie die oben besprochene Optimierung, so dass nur um den Rand herum verfeinert wird.
   - Parallelisieren Sie das Programm mit Hilfe von P-Threads.
   - Um eine gute Lastverteilung zu erhalten verwenden Sie einen "Erzeuger-Verbraucher" Ansatz. In unserem Fall kann jeder Prozess als Erzeuger und Verbraucher agieren, je nachdem, ob er ein Quadrat verfeinert, oder nicht.

     Hinweis: Denken Sie auch darüber nach, wann es Sinn macht einen Auftrag in den Puffer zu stellen und wann man ihn besser selber abarbeitet.