Übungen zur Vorlesung

Paralleles Höchstleistungsrechnen

Dr. S. Lang, D. Popović

Abgabe: 3. November 2009 in der Übung

Übung 3 Verklemmungsfreies Routing im Hypercube (5 Punkte)

(a) Skizzieren Sie, wie es in einem Verbindungs-Netzwerk mit $4D$-Hypercube-Struktur zu einer Verklemmung kommen kann.
(b) Entwerfen Sie einen verklemmungsfreien Routing-Algorithmus für den d-dimensionalen Hypercube. (Tipp: Zerlegen Sie das Netzwerk in d Teilnetzwerke).

Übung 4 Cube Connected Cycles (5 Punkte)

Cube Connected Cycles (CCC) können aus Hypercubes der Dimension d konstruiert werden, indem jeder Knoten des Hypercubes durch einen d-elementigen (inneren) Ring ersetzt wird. Ein Beispiel für den Fall $d = 3$ zeigt Abbildung 0.1. Die CCC wurden eingeführt, um den Knotengrad eines Verbindungsnetzwerkes zu reduzieren, ohne den Durchmesser signifikant zu erhöhen. In dieser Aufgabe sollen Sie daher Knotengrad und Durchmesser von CCC's untersuchen.

(a) Wie hoch ist die Anzahl e an Prozessoren eines CCC, der aus einem d-Hypercube konstruiert wird?
(b) Wie hoch ist der Knotengrad des CCC aus (a)?
(c) Wie ist der Durchmesser des Netzwerks in Abhängigkeit der Dimension d? Welche Komplexität ergibt sich daraus für den Durchmesser in Abhängigkeit der Anzahl der Prozessoren e?
Tipp: Nutzen Sie ein dimension-order Routing: Fangen Sie an einem Knoten des inneren Rings an und überlegen Sie, wieviele Schritte notwendig sind, um zu einem Hypercube-Ring, also zu einem Ring der Dimension $d-1$ des ursprünglichen Hypercubes zu gelangen. In diesem muss dann der maximale Abstand zu einem anderen Ursprungs-Knoten gegangen werden. Für den CCC wird dieser Knoten wiederum durch einen inneren Ring ersetzt, so dass erneut eine maximale Anzahl Verbindungen in diesem inneren Ring jeden seiner Knoten erreicht. Abschliessend kann man sich überlegen, dass die Schritte im ersten inneren Ring eliminiert werden können, und die verbleibende Anzahl Verbindungen aufsummieren.

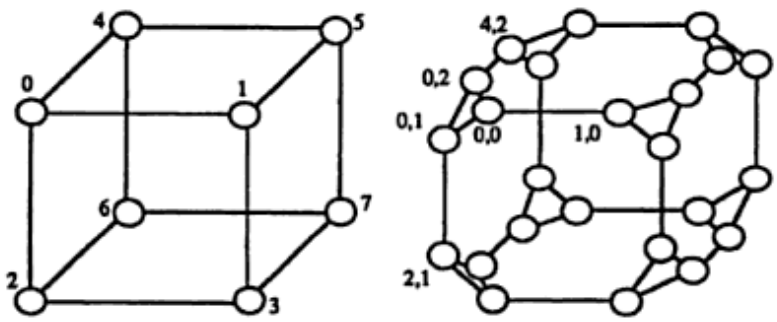

Abbildung 0.1: Cube Connected Cycles-Verbindungsnetzwerk konstruiert aus einem $3D$-Hypercube. Aus: Parhami, B.: *Introduction to Parallel Processing: Algorithms and Architectures.*

Übung 5 Messen von MFLOPS für $3D$-Probleme (10 Punkte)

In Aufgabe 2, Übungsblatt 1, haben wir die Anzahl von MFLOPS gemessen und verschiedene Strategien zur Cache-Nutzung untersucht. In dieser Übung wiederholen wir die Messungen an einem erweiterten Gauss-Seidel-Algorithmus in $3D$. Dazu sei ein Gebiet in d Dimensionen wie gehabt definiert über

$$\Omega_n^d = \Big\{ (i_0, \ldots, i_{d-1}) \in \mathbb{Z}^d \mid \forall\, 0 \leq k < d : 0 \leq i_k < n \Big\}.$$

Für eine Gitterfunktion $u^m : \Omega_n^3 \to \mathbb{R}$ definiert die Iteration

$$u^{m+1}(\alpha) = \frac{1}{26}\Bigg(\sum_{\substack{\beta \in [0,1]^d \\ \beta \neq (0,0,0)}} u^{m+1}(\alpha - \beta) + u^m(\alpha + \beta) \Bigg) \qquad \alpha \in [1, n-1]^3$$

ein $3D$-Gauss-Seidel-Verfahren, das zusätzlich zum Einfluss der direkten Nachbarn noch die der Diagonal-Nachbarn berücksichtigt. Implementieren Sie dieses Verfahren in der Programmiersprache C/C++ mit geeigneten Datenstrukturen Ihrer Wahl.

(a) Bestimmen Sie die Anzahl der Fließkommaoperationen und ermitteln Sie die Geschwindigkeit des Programmes in „Millionen Fließkommaoperationen pro Sekunde“. Fertigen Sie Plots der Anzahl der MFLOPS oder User time über Problemgröße an (z.B. mit *gnuplot*). Beginnen Sie mit kleinen n, etwa $n = 10, 20, 30, \ldots$ und erhöhen Sie sukzessive, bis die Datenmenge auf jeden Fall nicht mehr in die Caches passt (Beispiel: $n = 100$ ergibt mit 8 Byte für eine Gleitkommazahl die Datenmenge von ungefähr $100^3 * 8 = 8MB$, übliche Caches sind mehrere MB groß. Können Sie an Hand der Kurven auf die Cache-Hierarchie Ihres Rechners rückschliessen?

(b) Bauen Sie ein Kacheln des Problems mit variabler Blockgröße ein und wiederholen Sie die Messung aus (a). Wie verändern sich die Kurven qualitativ?

(c) Für eine feste Problemgröße aus (b), die sicher nicht mehr in den $L1$-Cache passt, plotten Sie nun die MFLOP-Rate über der Blockgröße. Sie sollten lokalisieren können, wann die Blöcke zu groß für den $L1$-Cache werden, und dass sich die Kurve der ohne das Tiling anpasst.

Zur Zeitmessung können Sie die Funktionen, welche in `timer.h` zur Verfügung gestellt werden, verwenden. Den Header und Hinweise zur Verwendung finden Sie auf der Vorlesungs-Homepage. Führen Sie die Zeitmessung bei kleiner Problemgröße n mehrmals hintereinander in einer Schleife aus, um Zeitmessfehler zu verringern. Initialisieren Sie die Felder mit sinnvollen Daten, z. B. $u(i, j) = i + j$. Übersetzen Sie das Programm mit maximaler Optimierungsstufe. Für den `GNU C/C++` Compiler ist etwa `-O3 -funroll-loops` empfehlenswert.

Zusatzpunkte

2 ZP Testen Sie das Programm mit naiver und/oder gekachelter Struktur auf verschiedenen Rechnern und stellen Sie die MFLOPS-Problemgrößen-Kurven der verschiedenen Rechner in einem Plot gegenüber. Diskutieren Sie etwaige Unterschiede.