Institut für Informatik — TU München
Scientific **C**omputing in **C**omputer **S**cience
T. Auckenthaler, A. Heinecke

WS 2011/12

Masterpraktikum Scientific Computing (High Performance Computing) Übungsblatt 3: Distributed Memory Parallelisierung (MPI)

Zur Übung am 29.11.2011

Aufgabe 8 „Das Gesetz von Amdahl“(1 Punkt)

Der Speedup Sp eines parallelen Programms mit p Prozessen im Vergleich zu seiner seriellen Version (mit nur einem Prozess) lässt sich über das Gesetz von Amdahl

$$Sp = \frac{1}{s + (1-s)/p}$$

ausdrücken, wobei $s \in [0, 1]$ den seriellen Anteil des Programms bestimmt. Damit kann schließlich die parallele Effizienz $Eff = Sp/p$ des Programms berechnet werden.

a) Für die parallele Version eines numerischen Gleichungslöser haben Sie einen seriellen Anteil von 10 % identifiziert. Wie groß darf die maximale Anzahl an Prozessen p werden, damit das Programm eine parallele Effizienz von mindestens 70 % erreicht?

b) Das Gesetz von Amdahl stellt gewissermaßen einen sehr pessimistischen Ansatz zur Bewertung paralleler Programme dar. Erklären Sie dieses Verhalten in Abhängigkeit der maximalen Anzahl an Prozessen p. Gibt es alternative Bewertungsmöglichkeiten?

Aufgabe 9 „Broadcast“(2 Punkte)

Implementieren Sie mit MPI eine Methode `broadcast`, die ein Array mit n `doubles` - ausgehend von einem root-Prozess (Prozess 0) - an alle anderen Prozesse schickt. Probieren Sie dabei drei unterschiedliche Algorithmen aus:

- In der Variante `trivial` schickt der root-Prozess die Daten an alle anderen $p - 1$ Prozesse.
- In der Variante `tree` werden die Daten baumartig verteilt. D.h., Prozess 0 schickt die Daten zunächst an Prozess $p/2$. Anschließend kümmert sich Prozess 0 rekursiv um die Verteilung der Daten auf Prozess 1 bis $p/2 - 1$, Prozess $p/2$ entsprechend um die Verteilung auf Prozess $p/2 + 1$ bis p.
- Entwickeln und implementieren Sie eine Variante `bonus`, in der jeder Prozess nur noch maximal $O(n)$ an Daten verschicken muss. Auf dem sogenannten „critical path“ sollen auch nur $O(n)$ Daten verschickt werden.

Ermitteln Sie für die implementierten Algorithmen jeweils den Kommunikationsaufwand (Anzahl an MPI-Nachrichten und Anzahl an gesendeten Bytes) auf dem “critical path”.
Messen Sie die erreichte Bandbreite ($n/time$) in Abhängigkeit der Nachchrichtengröße und vergleichen Sie die Ergebnisse mit denen der MPI-Routine `MPI_Bcast`.

Aufgabe 10 „Paralleles CG“(3 Punkte)

Das konjugierte Gradienten Verfahren (kurz CG-Verfahren) ist ein iteratives Verfahren zur Lösung von symmetrischen positiv definiten Gleichungssystemen der Form $Ax = b$.
In `poisson.cpp` ist eine sequentielle Implementierung des CG-Verfahrens vorgegeben. Erweitern Sie den Code zu einer mit MPI parallelisierten Variante.
Für eine parallele Bearbeitung ist die Matrix A in gleichgroße Rechtecke (z. B. 2×2, 3×3 oder 3×4) zu unterteilen. Entsprechend dieser Verteilung ist eine virtuelle Topologie zu generieren, die als Grundlage für sämtliche Kommunikation zwischen den einzelnen Prozessen dient. Während des Algorithmus müssen benachbarte Prozesse ihre Randwerte austauschen. Dies soll mithilfe eines selbst definierten Datentyps (z. B. MPI_Vector) realisiert werden.
Bestimmen Sie die Laufzeit Ihres Programms für unterschiedliche Matrixgrößen und testen Sie unterschiedliche Substrukturierungen der Matrix A mit einer festen Anzahl an Prozessen (z.B. 1×16, 2×8 und 4×4). Skizzieren Sie Ihre Ergebnisse in einem Diagramm und diskutieren Sie den Speedup sowie die parallele Effizienz!

Viel Erfolg beim Bearbeiten!

Die Abgabe ist bis 05.12.2011, 9.00 Uhr möglich.

Alle Programme finden Sie zum Herunterladen auf der Praktikumsseite unter:
`http://www5.in.tum.de/wiki/index.php/Masterpraktikum_Scientific_Computing_-_High`