Institut für Informatik — TU München
Scientific **C**omputing in **C**omputer **S**cience
Dipl.Inf. T. Auckenthaler, Dipl.Inf. W. Eckhardt

WS 2010/11

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

Zur Übung am 30.11.2010

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 auf dem Rechner `lx64ia2.lrz-muenchen.de` für $p = 16$ die erreichte Bandbreite ($n/time$) in Abhängigkeit der Nachchrichtengröße.

Aufgabe 10 „Zweidimensionales Glätten“(3 Punkte)

In der Bildverarbeitung werden zum Glätten von Bildern so genannte Tiefpassfilter eingesetzt, die die hochfrequenten Anteile des Bildinhaltes, also feine Strukturen im Ortsraum, abschwächen. Typische Vertreter sind bspw. das Rechteckfilter oder das Gauss-Filter. Das Gauss-Filter hat dabei eine gleichmäßige Wirkung auf hochfrequente Bildanteile wie Kanten und isolierte Störungen, die im Gegensatz zum Rechteckfilter weniger verwischt werden. Als Operator lässt sich das Gauss-Filter durch

$$F_G = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

aufschreiben. Durch Anwendung des Operators F_G auf Grauwertbilder, werden somit sukzessive die Grauwerte der einzelnen Pixel zusammen mit den Werten ihrer acht Nachbarn verarbeitet und dadurch geglättet. Die mehrmalige Anwendung des Operators F_G auf ein Grauwertbild hat dementsprechend einen stärkeren Glättungseffekt zur Folge.

Erstellen Sie mit MPI ein paralleles Programm `filter.c`, das ein Grauwertbild durch mehrmalige Anwendung des Operators F_G glättet. Als Grauwertbild ist eine Matrix $G^{n\times n}$ der Größe $n \times n$ zu verwenden.
Für eine parallele Bearbeitung ist die Matrix G 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. Nach jedem Glättungsschritt müssen benachbarte Prozesse ihre Randwerte austauschen. Dies soll mithilfe eines selbst definierten Datentyps (z. B. MPI_Vector) realisiert werden. Nach i Glättungsschritten sind die einzelnen Teilbildbereiche dem Masterknoten zu schicken, der diese zu einem Gesamtbild zusammenfügt und zur visuellen Betrachtung in eine Datei schreibt.
Bestimmen Sie die Laufzeit Ihres Programms für unterschiedliche Werte von n und testen Sie dieses für eine unterschiedliche Substrukturierung der Matrix G mit einer festen Anzahl an Prozessen (12 und 16). Skizzieren Sie Ihre Ergebnisse in einem Diagramm und diskutieren Sie den Speedup sowie die parallele Effizienz!

Viel Erfolg beim Bearbeiten!

Die Abgabe ist bis 21.12.2010, 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_Performance_Computing_-_Winter_10`