Übungen zur Vorlesung

Paralleles Höchstleistungsrechnen

Dr. S. Lang, D. Popović

Abgabe: 11. Januar 2012 in der Übung

Übung 21 MPI: Kommunikationszeiten (5 Punkte)

Wir wollen die Zeiten messen, die bei den MPI-Sende-Routinen `MPI_Send`, `MPI_Ssend`, `MPI_Bsend` und `MPI_Rsend` (synchrones bzw. asynchrones Senden, ...) benötigt werden, bis

1. der aufrufende Knoten wieder die Kontrolle über das Programm nach Beenden des jeweiligen Sende-Befehls erlangt hat,
2. der Empfänger die Nachricht ganz erhalten hat.

Teilaufgabe (a)

Schreiben Sie dazu ein Programm, das per Kommandozeilen-Parameter eine dieser Routinen zum Versenden einer Nachricht an einen anderen Prozess wählt. Die Größe der Nachricht soll ebenso als Programmparameter übergeben werden können. Der Sender mißt die Zeit, die es dauert, bis das Send-Kommando abgeschlossen ist, d.h. er wieder die Kontrolle über das Programm erhält, und der Empfänger die Zeit, die er benötigt, die Nachricht vollständig zu empfangen. Um zuverlässige Werte zu erhalten, sollte es möglich sein, diese Prozedur mehrmals hintereinander auszuführen und die Zeiten dann zu mitteln. Die Anzahl Wiederholungen sollte an die Dauer der Nachrichtenübermittlung angepasst sein, d.h. schnelle Übertragungen werden häufiger wiederholt. Das ganze muss natürlich nicht automatisch geschehen.

Messen Sie für jede Sende-Art die Zeiten mit mehreren Nachrichtengrößen. Die Nachrichtengröße soll dabei von wenigen Byte bis zu einigen MBytes variieren.

Teilaufgabe (b)

Die Routine `MPI_Send` kann blockierend (synchron) oder asynchron arbeiten. Überlegen Sie sich einen Test, wie Herausgefunden werden kann, ab welcher Nachrichtengröße n synchron gesendet wird. Implementieren Sie den Test und finden Sie das passende n im Pool heraus.

Freiwillige Zusatzaufgabe

Wiederholen Sie die Messungen aus Teilaufgabe (a), arbeiten Sie aber nur lokal auf einem Rechner arbeiten (wie das geht, siehe Übung 20). Wie verhalten Sich die Zeiten nun?

Hinweise

- Da wir auch den Overhead durch die Kommunikation messen wollen, messen wir die Echtzeit unter Verwendung von `MPI_Wtime()`.
- Synchronisieren Sie die Prozesse vor dem Starten der Zeitmessung mittels einer `MPI_Barrier()`.
- Profiling und Zeitmessung für MPI-Programme ist nicht trivial. Hier sind einige Links auf hilfreiche alte und neue Dokumente:
  - Ein Beispiel wie wir es in Teilaufgabe (a) implementieren wollen:
    http://cpansearch.perl.org/src/JOSH/Parallel-MPI-0.03/contrib/perf/mpi_timing.c
  - Pitfalls in der Zeitmessung:
    http://www.mcs.anl.gov/research/projects/mpi/mpptest/hownot.html

– MPI Profiling Suite:
  http://www.mcs.anl.gov/research/projects/mpi/mpptest/
– Ein recht ausführliches Paper:
  http://perlplexity.org/Publications/hoefler-collmea.pdf

Übung 22 MPI: Matrixmultiplikation (10 Punkte)

In den Übungen 4 und 9 haben wir zwei (quadratische) Matrizen $A, B \in \mathbb{R}^{n \times n}$ miteindander multipliziert, $C = A \cdot B$, und die Flop-Raten mit Tiling bzw. *OpenMP* gemessen. Wir wollen das nun unter Verwendung von *MPI* wiederholen und die Skalierbarkeit des Problems hinsichtlich der Anzahl der Prozesse untersuchen.

Aufgabe

Schreiben Sie ein Programm, das die Matrizen-Multiplikation *MPI*-parallel ausführt. Auf der Homepage steht eine sequentielle Variante inklusive Tiling bereit (Übersetzen mit `g++ -fopenmp mm_vanilla.c`), die Sie natürlich nicht verwenden müssen. Sie dürfen eine beliebige Strategie implementieren (erklären Sie kurz Ihr Konzept), z.B. die der Jacobi-Iteration. Initialisieren Sie die Matrizen so, daß sie vollbesetzt sind, etwa $a_{ij} = b_{ij} = i + j,\ i, j = 0 \ldots n-1$. Messen Sie die Rechenzeit in s und FLOP-Rate für $m = 1, 2, 4, 6, 8, 12, 16, 20, 25, 32$ beteiligte Prozesse und für die Matrix-Größen (Grundseite der Matrix) $n = 256, 512, 1024, 2048$. Verwenden Sie zur Zeitmessung wie gehabt `MPI_Wtime()` und synchronisieren vor Beginn der Zeitmessung mittels einer `MPI_Barrier()`. Plotten Sie Diagramme Rechenzeit über Problemgrößen sowie des Speed-Ups über Prozessoranzahl P für die verschiedenen Problemgrößen. Diskutieren Sie kurz die Ergebnisse. Falls noch vorhanden, können Sie auch mit den alten Ergebnissen der Aufgaben 4 und 9 vergleichen.

Freiwillige Zusatz-Aufgabe

Verwenden Sie auf den einzelnen Rechenknoten zusätzlich ein Tiling, um den Cache besser auszunutzen, und vergleichen Sie die Messungen mit den Ergebnissen der ungekachelten Variante.