Übungen zur Vorlesung

Paralleles Höchstleistungsrechnen

Dr. S. Lang, D. Popović

Abgabe: 19. Januar 2010 in der Übung

Übung 28 GNBP with MPI (20 Punkte)

In dieser Übung parallelisieren wir das N-Körper-Problem mit *MPI*. Eine ausführliche Einführung in das N-Körper-Problem findet sich auf dem letzten Aufgabenblatt, bei dem wir es durch Kacheln und Verwenden von *OpenMP* beschleunigt haben. In der *MPI*-Version des Codes hält jeder Prozess nur einen Teil der Koordinaten und Massen. Der Einfachheit halber können Sie für diese Übung $N \, mod \, p = 0$ annehmen.

1. Implementieren Sie ein *MPI*-paralleles Lösungsschema für das N-Körper-Problem, in dem die beteiligten Prozesse im Ring blockierend kommunizieren. Verwenden Sie Färben der Kanten, um Deadlocks zu vermeiden. Es steht Ihnen das Gerüst `nbody_mpi.c` zur Verfügung, in dem Hilfsroutinen wie das Generieren der Anfangsbedingungen und das Schreiben oder Lesen der `VTK`-Dateien schon parallelisiert sind. Der bereitgestellte Code funktioniert für einen einzelnen Prozess.

   Parallelisieren Sie nun die Routine `accelerate_mpi`, so dass sie das Ring-Kommunikationsschema implementiert. Führen Sie dann Rechnungen mit denselben Werten für N, für die Sie auch Messungen mit Kacheln und *OpenMPI* gemacht haben, durch. Messen Sie die Rechenzeiten und berechnen Sie den Speed-Up des parallelen Programms. Überprüfen Sie auch die Rechenergebnisse, in dem Sie die Resultate der parallelen Rechnung mit den Ergebnissen der sequentiellen Variante mit dem `fuzzy_diff`-Skript vergleichen.

2. Entwerfen Sie nun ein nicht-blockierendes Kommunikationsschema für den Ring (`MPI_Isend, MPI_Irecv`). Bei Verwenden nicht-blockierender Kommunikation kann das Färben der Kanten entfallen. Ein Prozess kann nun mit schon vorhandenen Daten rechnen, während er gleichzeitig auf weitere Daten wartet. Am Ende einer Iteration müssen natürlich alle Daten angekommen sein, wozu man zum Beispiel die Funktion `MPI_Waitall` verwenden kann. Kopieren Sie für diesen Teil die benötigten Code-Fragmente, so dass Sie beide *MPI*-parallelen Varianten zur Verfügung haben.

   Führen Sie nun die gleichen Messungen wie unter 1. aus und vergleichen Sie die Rechenzeiten und den Speed-Up zwischen beiden parallelen Varianten.