Verteilte numerische Algorithmen

Wahlveranstaltung im Hauptstudium

Prof. Dr. Peter Buchholz
Informatik IV Modellierung und Simulation
GB V R 406a
Tel: (0231)755 4746
Email: peter.buchholz@udo.edu

Daten und Orte:

Vorlesung:	Mo 12.15-14.00 HG I HS 2
Übung	Di 10.15-12.00 GB V R420 (ab 19.4.05) (Alternativen noch diskutieren)
Schwerpunktgebiete	2 & 3

Leistungsnachweis:

Aktive (!!) Teilnahme an den Übungen

Material und Literatur zur Vorlesung

Postscriptdateien mit Folien im Internet:
http://ls4-www.cs.uni-dortmund.de/Lehre/05-42783.html
(Folienwerden vorlesungsbegleitend verfügbar gemacht)

Literatur (kleine Auswahl)

- T. Rauber, G. Rünger. Parallele und verteilte Programmierung. Springer 2000.
  (in der Bibliothek vorhanden)
- D. P. Bertsekas, J. N. Tsitsiklis. Parallel and distributed computing - numerical methods.
  Prentice Hall 1989.
- I. Foster. Designing and building parallel programs. Addison-Weseley 1995.
  (*http://www-unix.mcs.anl.gov/dbpp/*)
- H. Schwandt. Parallele Numerik - Eine Einführung. Teubner 2003.
- G. Alefeld, I. Lenhardt, H. Obermaier. Parallele numerische Verfahren. Springer 2002

weitere Literatur auf der www Seite

Gliederung

1.	Einführende Beispiele und Problemdefinition
2.-3.	Prinzipien paralleler Rechnerarchitekturen
4.	Parallele Programmiermodelle
5.	Laufzeitanalyse paralleler Programme
6.-7.	Parallele Programmierung
8.-10.	Lineare Gleichungssysteme
11.	Nichtlineare Gleichungssysteme
12.	Schnelle Fourier-Transformation
13.	Optimierungsansätze
14.	Asynchrone Verfahren

Ergänzung durch andere Vorlesungen, Seminare und Praktika aus dem Gebiet der Modellierung und Simulation

<u>Themenbereiche</u>:

Leistungs- und Zuverlässigkeitsanalyse von
Rechen-, Kommunikations- und Materialflusssystemen,
Software zur Modellierung und Analyse,
Modellierung und Simulation ...

1 Einführung und Übersicht

Themenbereiche der Vorlesung:

- Numerische Algorithmen
- Verteilte/Parallele Systeme/Programme

Anwendungsgebiete der Numerik:

- Simulation von realen oder geplanten Abläufen auf einem Computer
- Analyse von Rechensystemen
- Strömungsberechnungen
- Wettervorhersage
- Chartanalyse
- ...

Optimierung von Abläufen

- Bestimmung kürzester Wege
- Transportzeitoptimierung
- ...

Algorithmen für die meisten Probleme sind bekannt, aber

- Aufwand wächst exponentiell mit Problemgröße
  (z.B. bei vielen Optimierungsproblemen)
- Genaue Lösung erfordert detaillierte Untersuchung und hohen Aufwand
  (z.B. bei Diskretisierung kontinuierlicher Vorgänge)

⇒ hohe Anforderungen an Rechenzeit und Speicherplatz

Welche Lösungen gibt es?

1. effizientere Programmierung (kaum Potenzial)
2. effizientere Algorithmen (Existenz !?)
3. schnellere Rechner (bei gegebener Technologie nur durch Parallelisierung)

Nur 3. ist realistische Alternative

⇒ hohe Investitionen im Bereich der Parallelrechner

- umfangreiche öffentliche Programme (USA, Japan)
- Zentren für Parallelrechnen in Deutschland
- Investitionen in großen Firmen

Aber auch (billigere) Alternative vorhanden:

Nutzung verteilter Rechner (NOWs, PC Netze) wird immer bedeutsamer

Ziel der Parallelverarbeitung:

Größere Probleme schneller als mit sequentiellen Rechnern lösen

Aufgabengebiete der Informatik

- Entwicklung paralleler Rechner
  - Hardware
  - Kommunikationsnetze
  - Protokolle
- Entwicklung paralleler Programmierumgebungen
  - Programmbibliotheken zur Kommunikation
  - Parallele Programmiersprachen
  - Parallelisierende Compiler und Laufzeitsysteme
- Entwicklung paralleler Algorithmen
  - theoretische Analyse paralleler Laufzeiten
  - praktische Realisierung paralleler Algorithmen

Ablauf im Idealfall:

- Auswahl eines geeigneten sequentiellen Algorithmus
- Automatische Generierung einer parallelen Variante

⇒ optimaler/guter paralleler Algorithmus

Szenario leider nicht realistisch, da

- gute sequentielle Algorithmen nicht zwangsläufig zu guten parallelen Algorithmen führen

Parallelisierung abhängt von

- den Daten
- der verwendeten Zielarchitektur

⇒ automatische Parallelisierung nur mit starken
Einschränkungen möglich

meisten großen Anwendungen werden zumindest teilweise "per Hand" parallelisiert

Vorgehen bei der Parallelverabeitung:

- Zerlegung eines großen Problems in möglichst autonome Teilprobleme
- (potenziell) parallele Lösung der Teilprobleme durch Threads, Tasks, Prozesse
- dazu notwendig Mapping der Prozesse auf Prozessoren (statisch zu Beginn oder dynamisch zur Laufzeit)

typischerweise Teilprobleme nicht
unabhängig voneinander lösbar

Granularität der Teilprobleme bestimmt

- notwendige Interaktion und
- maximale Anzahl paralleler Prozesse

viele parallele Prozesse ⇒hohe potenzielle Parallelität

wenige parallele Prozesse ⇒geringer Overhead

⇒ Kompromiss (problem-, architekturabhängig)

Abhängigkeiten durch Daten- oder Kontrollfluss erfordert Synchronisation/Kommunikation

⇒ parallele Programme beinhalten
Synchronisations-/Kommunikationsanweisungen

Je nach Programmierstil unterschiedliche Arten der Kommunikation

- über geteilten Speicher (virtuell oder real)
- über Nachrichtenaustausch

Ziel:

Architekturunabhängige Realisierung der Kommunikation!

Was behandelt die Vorlesung:

Teil 1.1 (recht kurz) parallele Architekturen

- Ebenen der Parallelität
- Speicherorganisation
- Verbindungsnetze
- Beispiele realer Systeme
- Workstation- / PC-Netze

Teil 1.2 parallele Programmierung

- Informationsaustausch
- Laufzeitanalyse
- Programmierung mit gemeinsamen Speicher (Pthreads)
- Programmierung mit Nachrichtenaustausch (MPI, PVM)

Teil 2 numerische Algorithmen

- lineare Gleichungssysteme
- nichtlineare Gleichungssysteme
- Fourier Transformation
- Optimierungsverfahren

Praktische Anwendung (in den Übungen)
für Workstation-/PC-Netze d.h.
Kommunikation über Nachrichtenaustausch

- (sehr) grobe Granularität
- aber breite Verfügbarkeit

Verwendung von MPI/PVM, Pthreads:
frei verfügbare Software für Unix/Linux Systeme

Bereitstellung

- einer Infrastruktur für die Generierung und den Ablauf paralleler Prozesse
- von zahlreichen Funktionen zur Kommunikation zwischen Prozessen

Anbindung an Fortran und C
(MPI-2/PVM auch an C++)

$\Rightarrow$ numerische Software i.d.R. in Fortran seltener in C

$\Rightarrow$ hier vorgestellte Beispiele in C

Durch MPI/PVM (fast) architekturunabhängige
$\Rightarrow$ Entwicklung verteilter/paralleler Software