Technology
Arts Sciences
TH Köln

Lehrveranstaltung
PPRA - Parallelprogrammierung und Rechnerarchitekturen

Version: 3 | Letzte Änderung: 29.04.2022 08:41 | Entwurf: 2 | Status: Entwurf

Allgemeine Informationen

Langname	Parallelprogrammierung und Rechnerarchitekturen
Anerkennende LModule	PPRA_BaTIN
Verantwortlich	Prof. Dr. Lothar Thieling Professor Fakultät IME
Niveau	Bachelor
Semester im Jahr	Sommersemester
Dauer	Semester
Stunden im Selbststudium	60
ECTS	5
Dozenten	Lehrbeauftragte(r) / Thieling
Voraussetzungen	grundlegende prozedurale Programmierkenntnisse Grundkenntnisse in Multitasking-Programmierung Grundlegende Funktionsweise eines Von-Neumann-Rechners Grundlagen der Digitaltechnik (Automaten, Hardware-Beschreibungssprache)
Unterrichtssprache	deutsch
separate Abschlussprüfung	Ja

Abschlussprüfung

Details

Die Studierenden sollen in einer schriftlichen Klausur folgende Kompetenzen nachweisen: 1.) Sicherer Umgang mit grundlegenden Begrifflichkeiten, Mechanismes und Konzepten. 2.) Parallelprogrammierung unter Verwendung gängiger Entwurfwerkzeuge (z.B. MPI und CUDA). 3.) Entwicklung von Problemlösungen, die prädestiniert sind für den Einsatz von Parallelrechnersystemen.

Mindeststandard

Mindestens 50% der möglichen Gesamtpunktzahl.

Prüfungstyp

Die Studierenden sollen in einer schriftlichen Klausur folgende Kompetenzen nachweisen: 1.) Sicherer Umgang mit grundlegenden Begrifflichkeiten, Mechanismes und Konzepten. 2.) Parallelprogrammierung unter Verwendung gängiger Entwurfwerkzeuge (z.B. MPI und CUDA). 3.) Entwicklung von Problemlösungen, die prädestiniert sind für den Einsatz von Parallelrechnersystemen.

Vorlesung / Übungen

Lernziele

Kenntnisse

Grundlagen der Parallelprogrammierung
Einleitung
Ansatz/Grundidee
Datenabhängigkeiten und Synchronisierung
Parallele Computerarchitekturen
Klassifikation
MMID
SIMD

Design paralleler Programme
Entwicklungsprozeß
Dekomposition Muster
Vollständig parallel
Task Parallelität (inkl. Task Pool)
Devide and Conquer
Pipeline (bzw. allgemeiner Task Graph)
Data Parallel (Geometric Data)
Recursive Data

Design paralleler Programme
Programm Struktur Muster zur Parallelprogrammierung
Master Slave (Master Worker)
Fork and Join
Singe Program Multiple Data (SPMD)
Multiple Program Multiple Data (MPMD)
Map-Reduce
Loop Parallelism
Zuordnung von Programm Struktur Mustern zu Dekompositions-Muster

---

Design paralleler Programme
Metriken zur Leistungsbestimmung (Performance Metrics)
Speedup
Amdahl's Gesetz
Effizienz
Skalierbarkeit

Leistungseinbuße
Lastausgleich (Load Balancing)
Messung der Leistung (Performance)

Einordnung von Standardbibliotheken hinsichtlich der vorangestellten Designmöglichkeiten und deren Nutzung auf Basis von Design-Pattern
MPI (distributed memory)
CUDA (GPU-Programming)

Rechnerarchitekturen (gem. Von-Neumann)
Konzeptionelle Komponenten zur Leistungssteigerung bzgl. ...
Speicher
Processing Units
GPU (s.o.)
Kommunikation
Protection

Implementierung der vorangestellten Konzepten in konkreten Rechnerarchitekturen
IA32e (AMD64)
ARM

Nicht-Von-Neuman-Architekturen
Anbindung von FPGAs an Von-Neumann-Architekturen
Neuronale Netze implementiert in FPGAs

Fertigkeiten

Die Studierenden sind in der Lage,
- den Aufbau, die Organisation und das Operationsprinzip von Rechnersystemen zu erörtern,
- den Zusammenhang zwischen Hardware-Konzepten und den Auswirkungen auf die Software zu analysieren, um effiziente Programme erstellen zu können,
- aus dem Verständnis über die Wechselwirkungen von Technologie, Rechnerkonzepten und Anwendungen die grundlegenden Prinzipien des Entwurfs nachzuvollziehen und anzuwenden,
- Rechnerkonzepte zu bewerten und zu vergleichen.

Die Studierenden sind in der Lage,
- Architekturmerkmale von Parallelrechner zu beschreiben,
- Parallelrechner, Programmierparadigmen und Designpattern zu bewerten und bezüglich einer Anwendung auszuwählen,
- Parallelrechner zu programmieren

Systemverhalten aus spezifizierenden Texten herleiten
technische Texte erfassen
implizite Angaben erkennen und verstehen
fehlende Angaben
erkennen
ableiten
erfragen

Aufwand Präsenzlehre

Typ	Präsenzzeit (h/Wo.)

Vorlesung	2
Übungen (ganzer Kurs)	1
Übungen (geteilter Kurs)	1
Tutorium (freiwillig)	0

Separate Prüfung

keine

Praktikum

Lernziele

Fertigkeiten

siehe Fertigkeiten, die unter "Vorlesung/Übung->Lernziele->Fertigkeiten" aufgeführt sind

zielgerichtetes Handhaben der Software-Entwicklungsumgebungen

komplexere Aufgaben in einem Kleinteam bewältigen

Erarbeitung von komplexeren Problemlösungen aus mindestens einem Bereich der rechen/datanintensiven Algorithmen, der Signalverarbeitung, der Künstlichen Intelligenz oder der Grafischen Animation, die pädestiniert sind für den Einsatz von Parallelrechnern.

Aufwand Präsenzlehre

Typ	Präsenzzeit (h/Wo.)
Praktikum	1
Tutorium (freiwillig)	0

Separate Prüfung

keine